Safe and Sample-Efficient Reinforcement Learning

Michael Luo
Ashwin Balakrishna
Brijen Thananjeyan
Ion Stoica, Ed.
Ken Goldberg, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-101
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-101.html

May 14, 2021

Acknowledgement

I would like to acknowledge and thank the entire Recovery RL team, in particular, Ashwin Balakrishna and Brijen Thananjeyan, for being supportive, responsive mentors and introducing me to the field of safe RL. I would also like to thank Prof. Ion Stoica and Ken Goldberg for advising and guiding me on the path to make Reinforcement Learning (RL) practical for real-life applications. Most importantly, I would like to thank my family for supporting me during the quarantine.

Safe and Sample-Efficient Reinforcement Learning[1]

by

Michael Luo

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ion Stoica, Chair
Professor Ken Goldberg

Spring 2021

[1]This thesis is adapted from **Recovery RL: Safe Reinforcement Learning with Learned Recovery Zones** by Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, & Ken Goldberg and **MESA: Offline Meta-RL for Safe Adaptation and Fault Tolerance** by Michael Luo, Ashwin Balakrishna, Brijen Thananjeyan, Suraj Nair, Julian Ibarz, Jie Tan, Chelsea Finn, Ken Goldberg, & Ion Stoica. It is recommended to cite these papers over this report.

The dissertation of Michael Luo, titled Safe and Sample-Efficient Reinforcement Learning, is approved:

Chair	Ion Stoica Ion Stoica (May 14, 2021 13:37 PDT)	Date	May 14, 2021
	Ken Goldberg Ken Goldberg (May 14, 2021 13:17 PDT)	Date	May 14, 2021

University of California, Berkeley

Safe and Sample-Efficient Reinforcement Learning

Abstract

Safe and Sample-Efficient Reinforcement Learning

by

Michael Luo

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ion Stoica, Chair

Reinforcement learning (RL) provides a flexible and general-purpose framework for learning new behaviors through interaction with the environment. However, safe exploration is critical to deploying reinforcement learning algorithms in risk-sensitive, real-world environments. Learning new tasks in unknown environments requires extensive exploration, but safety requires limiting exploration.

To navigate this tradeoff, we propose Recovery RL, an algorithm which (1) efficiently leverages offline data to learn about constraint violating zones *before* policy learning and (2) *separating* the goals of improving task performance and constraint satisfaction across two policies: a task policy that only optimizes the task reward and a recovery policy that guides the agent back to safety when constraint violation is likely. Recovery RL can be applied on top of any RL algorithm. Simulation and physical experiments across 7 continuous control domains, including two contact rich manipulation tasks and an image-based navigation task, suggest that Recovery RL trades off constraint violations and task successes 2-80x more efficiently than the next best prior methods, which jointly optimize task performance and safety via constrained optimization or reward shaping.

Next, we generalize the problem of safe exploration to the transfer learning setting, where there is assumed access to environments of similar dynamics. In this setting, safe exploration is recasted as an offline meta-reinforcement learning problem, where the objective is to leverage datasets of safe and unsafe behavior across different environments to quickly adapt learned safety measures to new environments with unseen, perturbed dynamics. We propose MEta-learning for Safe Adaptation (MESA), an approach which meta-learns a safety measure and stacks on top of Recovery RL. Simulation experiments across 5 continuous control domains suggest that MESA can leverage datasets from prior environments to reduce constraint violations in unseen environments by up to 2x while maintaining task performance compared to prior algorithms that do not learn transferable risk measures.

To my parents, advisors, and research collaborators.

I would like to acknowledge and thank the entire Recovery RL team, in particular, Ashwin Balakrishna and Brijen Thananjeyan, for being supportive, responsive mentors and introducing me to the field of safe RL. I would also like to thank Prof. Ion Stoica and Ken Goldberg for advising and guiding me on the path to make Reinforcement Learning (RL) practical for real-life applications. Most importantly, I would like to thank my family for supporting me during the quarantine.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reinforcement learning (RL) provides a general framework for robots to acquire new skills, and has shown promise in a variety of robotic domains such as navigation [46], locomotion [21], and manipulation [30, 40]. However, enforcing constraints on the agent's behavior to encourage safety during learning and exploration is challenging, since constraint violating states and the states leading to them may be initially unknown and must be learned from experience. Thus, safe exploration requires navigating a tradeoff: learning new skills through environmental interaction requires exploring a wide range of possible behaviors, but learning safely forces the agent to restrict exploration to constraint satisfying states.

We consider a RL formulation subject to constraints on the probability of unsafe future behavior and design an algorithm that can effectively balance the often conflicting objectives of task directed exploration and safety. Most prior work in safe RL integrates constraint satisfaction into the task objective to jointly optimize the two. While these approaches are appealing for their generality and simplicity, there are two key aspects which make them difficult to apply in practice. First, the inherent objective conflict between exploring sufficiently to learn a good task policy and limiting exploration to avoid constraint violations can lead to suboptimalities in policy optimization. Second, sufficiently exploring the environment to learn about constraint structure necessitates a significant amount of constraint violations during learning. However, this can result in the agent taking uncontrolled actions which can damage both itself and the environment.

We take a step towards addressing these issues with two key algorithmic ideas. First, inspired by recent work in robust control [17, 5, 19, 35], we represent the RL agent with two policies: the first policy focuses on optimizing the unconstrained task objective (task policy) and the second policy takes control when the task policy is in danger of constraint violations in the near future (recovery policy). Instead of modifying the policy optimization procedure to encourage constraint satisfaction, which can introduce suboptimality in the learned task policy [45], the recovery policy can be viewed as defining an alternate MDP for the task policy to explore within in which constraint violations are unlikely. Separating the task policy and the recovery policy makes it easier to balance task performance and safety, and allows us to apply off-the-shelf RL algorithms for learning each. Second, we leverage offline data to learn a recovery set, which indicates regions of the MDP in which future constraint violations are likely, and a recovery policy, which is queried within this

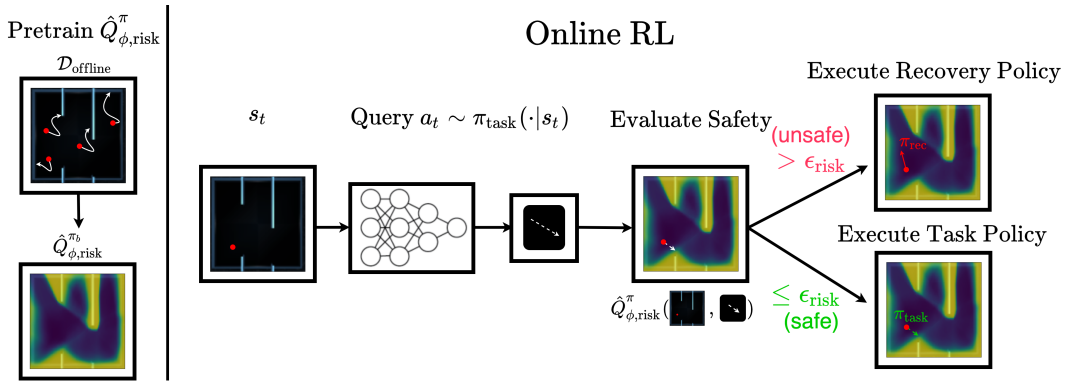


Figure 1.1: **Recovery RL:** We illustrate Recovery RL on a 2D maze navigation task where a constraint violation corresponds to hitting a wall. Recovery RL first learns safety critic $\hat{Q}^{\pi}_{\phi,\text{risk}}$ with offline data from some behavioral policy π_b, which provides a small number of controlled demonstrations of constraint violating behavior as shown on the left. For the purposes of illustration, we visualize the average of the $\hat{Q}^{\pi}_{\phi,\text{risk}}$ learned by Recovery RL over 100 action samples. Then, at each timestep, Recovery RL queries the task policy π_{task} for some action a at state s, evaluates $\hat{Q}^{\pi}_{\phi,\text{risk}}(s,a)$, and executes the recovery policy π_{rec} if $\hat{Q}^{\pi}_{\phi,\text{risk}}(s,a) > \epsilon_{\text{risk}}$ and π_{task} otherwise. The task policy, recovery policy, and safety critic are updated after each transition from agent experience.

set to prevent violations. This offline data can be collected by a human or an agent under human supervision to provide controlled examples of constraint violations, such as gently tipping over a glass rather than aggressively knocking the glass over and shattering it. Thus, the agent is able to observe constraint violations and learn from them without the task policy directly having to experience too many uncontrolled examples of these violations during learning.

We present Recovery RL, a new algorithm for safe robotic RL. Unlike prior work, Recovery RL (1) can effectively leverage offline data of constraint violations to learn about constraints *before* interacting with the environment, and (2) uses separate policies for the task and recovery to learn safely without significantly sacrificing task performance. We evaluate Recovery RL against 5 state-of-the-art safe RL algorithms on 6 navigation and manipulation domains in simulation, including a visual navigation task, and find that Recovery RL trades off constraint violations and task successes 2 - 80 times more efficiently than the next best prior method. We then evaluate Recovery RL on a constrained image-based reaching task on a physical robot and find that Recovery RL trades off constraint violations and task successes 12 times more efficiently than the next best prior algorithm.

Next, we address the challenge of safe exploration in the transfer learning setting. To motivate transfer learning, a challenge with Recovery RL is that these offline transitions are required to be in an environment with the same dynamics as that in which the agent is deployed, which is not always be practical in risk-sensitive environments where a large number of constraint violations could be exceedingly costly or dangerous. Additionally, shifting dynamics is a ubiquitous phenomenon in real robot hardware: for example losses in battery voltage [51] or wear-and-tear in manipulators or actuators [29]. These changes can drastically change the space of safe behaviors, as the robot may need to compensate for unforeseen differences in the robot dynamics. Furthermore, these changes in

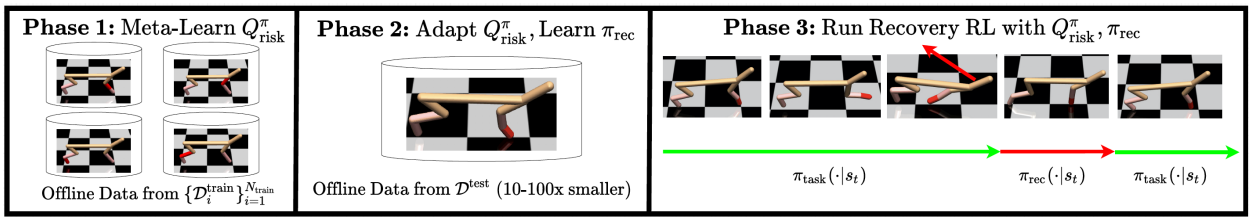


Figure 1.2: **MEta-learning for Safe Adaptation (MESA)**: MESA takes a 3 phase approach to learn a transferable risk measure for safe reinforcement learning. In Phase 1, MESA uses offline datasets of interactions from a number of training environments with different dynamics (in this case a HalfCheetah with a different disabled leg segment in each environment) to meta-learn a safety critic Q^{π}_{risk}. In Phase 2, MESA adapts this safety critic to a test environment with unseen dynamics (in this case HalfCheetah has a disabled leg that was unseen in training) using a small dataset (10-100x smaller than the training datasets) of offline transitions from the test environment and learns a recovery policy to descend the learned safety critic and protect the agent from constraint violations. Finally, in Phase 3, MESA uses the adapted safety critic and recovery policy for safe reinforcement learning with Recovery RL. In this case, MESA is able to prevent constraint violations by encouraging the agent to avoid letting the HalfCheetah's head collide with the ground.

dynamics will often not be immediately observable for a robot control policy, motivating algorithms which can identify and adapt to these changes based on *interaction* in the environment.

To address this, we aim to transfer knowledge about safety between environments with different dynamics, so that the agent can rapidly learn to be safe when learning in a test environment with previously unseen dynamics. Our insight is that a robot should be able to leverage offline datasets across previous deployments, with knowledge of only safe and unsafe states in these datasets, to rapidly learn to be safe in new environments without task specific information. This is motivated by the idea of fault tolerance in robotics: an intelligent agent should be able to leverage prior experiences to quickly adapt to unexpected changes in the environment dynamics without exhibiting unsafe behaviors, such as a legged locomotion system learning to stabilize even when it suddenly loses power in one of its joints.

The contributions of this work are (1) casting safe RL as an offline meta-reinforcement learning problem [39, 12], where the objective is to leverage fully offline data from training and test environments to learn how to be safe in the test environment; (2) MEta-learning for Safe Adaptation (MESA), which meta-learns a risk measure that is used for safe reinforcement learning in new environments with previously unseen dynamics; (3) simulation experiments across 5 continuous control domains which suggest that MESA can cut the number of constraint violations in half in a new environment with previously unseen dynamics while maintaining task performance compared to that of prior algorithms.[1]

[1]Individual Contributions: **Recovery RL**: Generated experiment results for both Object Extraction environments and ablations in Sections 5.1,5.2; **MEta-learning for Safe Adaptation (MESA)**: Proposed MESA and conducted all experiments and ablations in Sections 7.1, 7.2

Chapter 2

Related Work

Prior work has studied safety in RL in several ways, including imposing constraints on expected return [1, 57], risk measures [24, 50, 54, 56], and avoiding regions of the MDP where constraint violations are likely [14, 18, 5, 60, 7, 62]. We build on the latter approach, and design algorithms which uses a learned recovery policy to keep the RL agent within a learned safe region of the MDP.

2.1 Safe Reinforcement Learning

Jointly Optimizing for Task Performance and Constraint Satisfaction: A popular strategy in algorithms for safe RL involves modifying the policy optimization procedure of standard RL algorithms to simultaneously reason about both task reward and constraints using methods such as trust regions [1], optimizing a Lagrangian relaxation [57, 43, 52], or constructing Lyapunov functions [9, 10]. The most similar of these works to Recovery RL is [52]. Srinivasan et al.[52] trains a safety critic, which estimates the probability of future constraint violation under the current policy, and optimizes a Lagrangian objective function to limit the probability of constraint violations while maximizing task reward. Unlike Srinivasan et al. [52], which uses the safety critic to modify the task policy optimization objective, Recovery RL uses it to determine when to execute a learned recovery policy which minimizes the safety critic to keep the agent in safe regions of the MDP. This idea enables Recovery RL to more effectively balance task performance and constraint satisfaction than algorithms which jointly optimize for task performance and safety.

Closest to MESA, Zhang et al. [66] designs a model-based RL algorithm which leverages unsafe data from a variety of training environments with different dynamics to predict whether the agent will encounter unsafe states and penalize its reward if this is the case. Unlike Zhang et al. [66], MESAexplicitly optimizes for adaptation and decouple information about constraints from the reward function, making it possible to efficiently learn transferable notions of safety. Additionally, we learn a risk measure in a fully offline setting, and do not assume direct access to the training environments.

Restricting Exploration with an Auxiliary Policy: Another approach to safe RL explicitly re-

stricts policy exploration to a safe subset of the MDP using a recovery or shielding mechanism. This idea has been explored in [17, 5], which utilize Hamilton-Jacobi reachability analysis to define a task policy and safety controller, and in the context of shielding [19, 35, 2]. In contrast to these works, which assume approximate knowledge of system dynamics [17, 5, 19, 35, 2] or require precise knowledge of constraints apriori [2], Recovery RL learns information about the MDP, such as constraints and dynamics, from experience and can scale to high-dimensional state spaces. Additionally, Recovery RL reasons about probabilistic constraints rather than robust constraints, allowing it to estimate a safe set without a dynamics model. Han et al. [23] and Eysenbach et al. [14] introduce reset policies which are trained jointly with the task policy to reset the agent to its initial state distribution, ensuring that the task policy only learns behaviors which can be reset [14]. However, enforcing the ability to fully reset can be impractical or inefficient. Inspired by this work, Recovery RL instead executes approximate resets to nearby safe states when constraint violation is probable. Similar to Recovery RL, Richter and Roy [46] learns the probability of constraint violation conditioned on an action plan to activate a hand-designed safety controller. In contrast, Recovery RL uses a learned recovery mechanism which can be broadly applied across different tasks.

Leveraging Demonstrations for Safe RL and Control: Finally, there has also been significant prior work investigating how demonstrations can be leveraged to enable safe exploration. Rosolia and Borrelli [47], Thananjeyan et al.[58] introduce model predictive control algorithms which leverage initial constraint satisfying demonstrations to iteratively improve their performance with safety guarantees, and Thananjeyan et al.[60] extends these ideas to the RL setting. In contrast to these works, Recovery RL learns a larger safe set that explicitly models future constraint satisfaction and also learns the problem constraints from prior experience without task specific demonstrations. Additionally, Recovery RL can be applied with either model-free or model-based RL algorithms while [60, 58] require a dynamics model to evaluate reachability-based safety online.

2.2 Meta Reinforcement Learning

There is a rich literature [49, 6, 42, 61, 25] studying learning agents that can efficiently adapt to new tasks. In the context of reinforcement learning, this problem, termed *meta-reinforcement learning* [13, 63, 16], aims to learn RL agents which can efficiently adapt their policies to new environments with unseen transition dynamics and rewards. A number of strategies exist to accomplish this such as recurrent or recursive policies [13, 63, 38], gradient based optimization of policy parameters [16, 26], task inference [44, 27, 15], or adapting dynamics models for model-based RL [48, 41]. One of the core challenges studied in many meta-RL works is efficient exploration [53, 44, 67, 37], since the agent needs to efficiently explore its new environment to identify the underlying task. Unlike all of these prior works, which focus on learning transferable policies, we focus on learning *risk measures* which can be used to safely learn new tasks in a test environment with previously unseen dynamics. Additionally, we study learning these measures in the context of offline meta-RL, and learn from purely offline datasets of prior interactions in various environments with different

dynamics.

The *offline meta reinforcement learning* problem [39, 12, 34] considers a setting in which the agent learns from a set of offline data from each training task, and adapts to the test environment conditioned only on a small set of offline transitions. Critically, this setting is particularly well suited to the problem of safe RL, because it has potential to enable an agent to be safe in an environment with previously unseen dynamics, conditioned on a small set of experiences from that environment. In this work, we formalize safe reinforcement learning as an offline meta-RL problem and present an algorithm to adapt *a safety critic* to new environments and use this adapted safety critic for safe reinforcement learning.

Srinivasan et al. [52] and Thananjeyan et al. [59] also leverage prior offline data from previous interactions to learn how to be safe. However, unlike these works, which assume that prior data is collected in an environment with the same dynamics as the test environment, MESA learns to leverage experience from a variety of environments with different dynamics in addition to a small amount of data from the test environment. This choice makes it possible to avoid excessive constraint violations in the test environment, in which constraint violations may be costly, by leveraging prior experience in safer environments or from accident logs from previous deployments.

One option for meta-learning for safe RL is using meta-learning for sim-to-real domain adaptation where data can be collected safely and at scale in simulated environments [4]. By contrast, MESA explicitly reasons about safety constraints in the environment to learn adaptable risk measures. Additionally, while prior work has also explored using meta-learning in the context of safe-RL [20], specifically by learning a single safety filter which keeps policies adapted for different tasks safe, we instead adapt *the risk measure itself* to unseen dynamics and fault structures.

Chapter 3

Preliminaries

3.1 Constrained Markov Decision Processes

In safe reinforcement learning, an agent interacts with a Constrained Markov Decision Process (CMDP) [3], defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, C, \rho_0, \gamma, \gamma_{\text{risk}})$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ is the action space, the transition dynamics function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ maps the current state and action to a probability distribution of next states, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $C : \mathcal{S} \to \{0, 1\}$ is a constraint function which indicates whether a state is constraint violating, $\rho_0 : \mathcal{S} \to [0, 1]$ is the starting state distribution, and $\gamma, \gamma_{\text{risk}} \in [0, 1]$ are the discount factors for the rewards and constraint values. As in prior work [52, 59], we assume constraint violations end the episode immediately. The expected return for a policy $\pi : \mathcal{S} \to \mathcal{A}$ is $R(\pi) = \mathbb{E}_{\pi,\rho_0,P}\left[\sum_t^\infty \gamma^t R(s_t, a_t)\right]$. The discounted probability of future constraint violation for policy π is $Q^\pi_{\text{risk}} = \mathbb{E}_{\pi,\rho_0,P}\left[\sum_t^\infty \gamma^t_{\text{risk}} C(s_t)\right] = \mathbb{E}_{\pi,\rho_0,P}\left[\sum_t^\infty \gamma^t_{\text{risk}} \mathbb{P}\left(C\left(s_t\right) = 1\right)\right]$. Unlike unconstrained RL, safe RL agents seek to optimize:

$$\pi^* = \arg\max_\pi \left\{R^\pi : Q^\pi_{\text{risk}} \leq \epsilon_{\text{risk}}\right\} \tag{3.1}$$

where ϵ_{risk} is a hyper-parameter that defines how safe the agent should be.

3.2 Safety Critics for Safe RL

Recent work [59, 52] investigates ways to estimate the discounted future probability of catastrophic constraint violations under the current policy:

$$\begin{aligned} Q^\pi_{\text{risk}}(s_t, a_t) &= \mathbb{E}_\pi\left[\sum_{t'=t}^{\infty} \gamma^{t'-t}_{\text{risk}} C(s_{t'}) | s_t, a_t\right] \\ &= C(s_t) + (1 - C(s_t))\gamma_{\text{risk}} \mathbb{E}_\pi\left[Q^\pi_{\text{risk}}(s_{t+1}, a_{t+1}) | s_t, a_t\right]. \end{aligned} \tag{3.2}$$

. This is different from the standard Bellman equations for solving MDPs due to the assumption that episodes terminate when $C(s_t) = 1$. In practice, algorithms search over a parametric function class:

$\left\{Q^{\pi}_{\psi,\text{risk}}(s_t, a_t) : \psi \in \Psi\right\}$, where ψ is a particular parameter vector and Ψ is its possible values. This function is trained by minimizing an MSE loss function with respect to a target function on a dataset of transitions $\{(s_t, a_t, C(s_t), s_{t+1})_i\}_{i=1}^{N}$ collected in the environment:

$$J_{\text{risk}}(s_t, a_t, s_{t+1}; \psi) = \\ \frac{1}{2}\left(Q^{\pi}_{\psi,\text{risk}}(s_t, a_t) - (C(s_t) + (1 - C(s_t))\gamma_{\text{risk}} \mathop{\mathbb{E}}_{a_{t+1}\sim\pi(\cdot|s_{t+1})}[Q^{\pi}_{\psi,\text{risk,targ}}(s_{t+1}, a_{t+1})])\right)^2 \tag{3.3}$$

where $Q^{\pi}_{\psi,\text{risk,targ}}$ is a target network and $C(s_t)$ denotes whether state s_t is constraint violating. In practice, we use a target network to create the targets as in prior work [52, 22]. The safety critic can be used for constrained policy search, by either optimizing a Lagrangian function [57, 52, 8] with it or filtering dangerous actions [52, 59].

3.3 Meta-learning

Consider a task distribution $p(\mathcal{M})$ where tasks are sampled via $\mathcal{M}_i \sim p(\mathcal{M})$. In the RL setting, each task corresponds to an MDP, all of which share the same state and action spaces but may have varying dynamics (e.g. varying controller impedance for a legged robot). The goal in MESA is to learn risk measures that rapidly adapt to new environments, such as when a robot's actuator loses power and it is forced to compensate with only the remaining actuators. We will briefly discuss how functions can be initialized for rapid adaptation to new tasks by training on similar tasks.

Meta-learning learns a model explicitly optimized for adaptation to a new task from $p(\mathcal{M})$. Let $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{M}_i}(f_\theta)$ be the parameters θ after a single gradient step from optimizing $\mathcal{L}_{\mathcal{M}_i}(f_\theta)$. Model-Agnostic Meta-Learning (MAML) [16] optimizes the following meta-objective at meta-train time:

$$\min_\theta \mathbb{E}_{\mathcal{M}_i\sim p(\mathcal{M})}\left[\mathcal{L}_{\mathcal{M}_i}(f_{\theta_i'})\right] = \\ \min_\theta \mathbb{E}_{\mathcal{M}_i\sim p(\mathcal{M})}\left[\mathcal{L}_{\mathcal{M}_i}(f_{\theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{M}_i}(f_\theta)})\right] \tag{3.4}$$

After meta-training, to quickly adapt to a new test environment, MAML computes a task-specific loss function from an unseen task and updates θ with several gradient steps using a standard RL algorithm.

Chapter 4

Recovery RL

Here we outline the central ideas behind Recovery RL. In Section 3.2, we reviewed how to learn a safety critic to estimate the probability of future constraint violations. Then, in Section 4.1, we will show how this safety critic can be used to define the recovery policy for Recovery RL and the recovery set in which it is activated. In Section 4.2, we will discuss how the safety critic and recovery policy are learned from offline data and, in Section 4.3, we will discuss implementation details.

4.1 Defining a Recovery Set and Policy

Recovery RL executes a composite policy π in the environment, which selects between a task-driven policy π_{task} and a recovery policy π_{rec} at each timestep based on whether the agent is in danger of constraint violations in the near future. To quantify this risk, we use Q^{π}_{risk} to construct a recovery set that contains state-action tuples from which π may not be able to avoid constraint violations. Then if the agent finds itself in the recovery set, it executes a learned recovery policy instead of π_{task} to navigate back to regions of the MDP that are known to be sufficiently safe. Specifically, define two complimentary sets: the safe set $\mathcal{T}^{\pi}_{\text{safe}}$ and recovery set $\mathcal{T}^{\pi}_{\text{rec}}$:

$$\mathcal{T}^{\pi}_{\text{safe}} = \{(s,a) \in \mathcal{S} \times \mathcal{A} : Q^{\pi}_{\text{risk}}(s,a) \leq \epsilon_{\text{risk}}\} \qquad \mathcal{T}^{\pi}_{\text{rec}} = \mathcal{S} \times \mathcal{A} \setminus \mathcal{T}^{\pi}_{\text{safe}}$$

We consider state-action tuple (s,a) safe if in state s after taking action a, executing π has a discounted probability of constraint violation less than ϵ_{risk}.

If the task policy π_{task} proposes an action $a^{\pi_{\text{task}}}$ at state s such that $(s, a^{\pi_{\text{task}}}) \notin \mathcal{T}^{\pi}_{\text{safe}}$, then a recovery action sampled from π_{rec} is executed instead of $a^{\pi_{\text{task}}}$. Thus, the recovery policy in Recovery RL can be thought of as projecting π_{task} into a safe region of the policy space in which constraint violations are unlikely. The recovery policy π_{rec} is also an RL agent, but is trained to minimize $\hat{Q}^{\pi}_{\phi,\text{risk}}(s,a)$ to reduce the risk of constraint violations under π. Let $a_t^{\pi_{\text{task}}} \sim \pi_{\text{task}}(\cdot|s_t)$ and

$a_t^{\pi_{\text{rec}}} \sim \pi_{\text{rec}}(\cdot|s_t)$. Then π selects actions as follows:

$$a_t = \begin{cases} a_t^{\pi_{\text{task}}} & (s_t, a_t^{\pi_{\text{task}}}) \in \mathcal{T}_{\text{safe}}^{\pi} \\ a_t^{\pi_{\text{rec}}} & (s_t, a_t^{\pi_{\text{task}}}) \in \mathcal{T}_{\text{rec}}^{\pi} \end{cases} \tag{4.1}$$

Recovery RL acts as a filtering mechanism that aims to block proposed actions that are likely to lead to unsafe states, equivalent to modifying the environment that π_{task} operates in with new dynamics:

$$P_{\epsilon_{\text{risk}}}^{\pi_{\text{rec}}}(s'|s,a) = \begin{cases} P(s'|s,a) & (s,a) \in \mathcal{T}_{\text{safe}}^{\pi} \\ P(s'|s,a^{\pi_{\text{rec}}}) & (s,a) \in \mathcal{T}_{\text{rec}}^{\pi} \end{cases} \tag{4.2}$$

We train $\hat{Q}_{\phi,\text{risk}}^{\pi}$ on samples from π since π_{task} is not executed directly in the environment, but is rather filtered through π.

It is easy to see that the proposed recovery mechanism will shield the agent from regions in which constraint violations are likely if $\hat{Q}_{\phi,\text{risk}}^{\pi}$ is correct and executing π_{rec} reduces its value. However, this poses a potential concern: while the agent may be safe, how do we ensure that π_{task} can make progress in the *new* MDP defined in equation 4.2? Suppose that π_{task} proposes an unsafe action $a_t^{\pi_{\text{task}}}$ under $\hat{Q}_{\phi,\text{risk}}^{\pi}$. Then, Recovery RL executes a recovery action $a_t^{\pi_{\text{rec}}}$ and observes transition $(s_t, a_t^{\pi_{\text{rec}}}, s_{t+1}, r_t)$ in the environment. However, if π_{task} is updated with this observed transition, it will not learn to associate its proposed action $(a_t^{\pi_{\text{task}}})$ in the new MDP with r_t and s_{t+1}. As a result, π_{task} may continue to propose the same unsafe actions without realizing it is observing the result of an action sampled from π_{rec}. To address this issue, for training π_{task}, we *relabel all actions with the action proposed by* π_{task}. Thus, instead of training π_{task} with executed transitions (s_t, a_t, s_{t+1}, r_t), π_{task} is trained with transitions $(s_t, a_t^{\pi_{\text{task}}}, s_{t+1}, r_t)$. This ties into the interpretation of defining a safe MDP with dynamics $P_{\epsilon_{\text{risk}}}^{\pi_{\text{rec}}}(s'|s,a)$ for π_{task} to act in since all transitions for training π_{task} are relabeled as if π_{task} was executed directly.

Algorithm 1 Recovery RL

Require: $\mathcal{D}_{\text{offline}}$, task horizon H, number of episodes N

Pretrain π_{rec} and $\hat{Q}_{\phi,\text{risk}}^{\pi}$ on $\mathcal{D}_{\text{offline}}$ ▷ Section 4.2
$\mathcal{D}_{\text{task}} \leftarrow \emptyset$, $\mathcal{D}_{\text{rec}} \leftarrow \mathcal{D}_{\text{offline}}$
$s_0 \leftarrow$ `env.reset()`
for $i \in \{1, \dots N\}$ **do**
  for $t \in \{1, \dots H\}$ **do**
    if $C(s_t) = 1$ or `is_terminal`(s_t) **then**
      $s_t \leftarrow$ `env.reset()`
    $a_t^{\pi_{\text{task}}} \sim \pi_{\text{task}}(\cdot|s_t)$ ▷ Query task policy
    ▷ Check if task policy will be unsafe
    if $(s_t, a_t^{\pi_{\text{task}}}) \in \mathcal{T}_{\text{rec}}^{\pi}$ **then**
      $a_t \sim \pi_{\text{rec}}(\cdot|s_t)$ ▷ Select recovery policy
    else
      $a_t = a_t^{\pi_{\text{task}}}$ ▷ Select task policy
    Execute a_t, observe s_{t+1}, $r_t = R(s_t, a_t)$, $c_t = C(s_t)$
    ▷ Relabel transition
    $\mathcal{D}_{\text{task}} \leftarrow \mathcal{D}_{\text{task}} \cup \{(s_t, a_t^{\pi_{\text{task}}}, s_{t+1}, r_t)\}$
    $\mathcal{D}_{\text{rec}} \leftarrow \mathcal{D}_{\text{rec}} \cup \{(s_t, a_t, s_{t+1}, c_t)\}$
    Train π_{task} on $\mathcal{D}_{\text{task}}$, π_{rec} and $\hat{Q}_{\phi,\text{risk}}^{\pi}$ on $\mathcal{D}_{\text{rec}}$ ▷ Eq. 3.3

4.2 Offline Pretraining

To convey information about constraints before interaction with the environment, we provide the agent with a set of transitions $\mathcal{D}_{\text{offline}}$ that contain constraint violations for pretraining. While this requires violating constraints in the environment, a human may be able to carefully demonstrate these unsafe transitions in a relatively controlled manner (e.g. gently tipping over a glass) so that the robot does not need to accidentally learn them online (e.g. knocking the glass off the table). We pretrain $\hat{Q}^{\pi}_{\phi,\text{risk}}$ by minimizing Equation 3.3 over offline batches sampled from $\mathcal{D}_{\text{offline}}$. We additionally pretrain π_{rec} using the data in $\mathcal{D}_{\text{offline}}$. Note that any RL algorithm can be used to represent π_{task} while any off-policy RL algorithm can be used to learn π_{rec}. For some environments in which exploration is challenging, we utilize a separate set of task demonstrations to initialize π_{task} to expedite learning.

Recovery RL first pretrains $\hat{Q}^{\pi}_{\phi,\text{risk}}$ and recovery policy π_{rec} on a set of transitions $\mathcal{D}_{\text{offline}}$ containing constraint violations. During online RL training, the agent actually executes π, which is an algorithmic selection between policy π_{task} and π_{rec}. This process is summarized in Algorithm 1 and Figure 1.1.

4.3 Practical Implementation

Recovery Policy: In principle, any off-policy RL algorithm can be used to learn π_{rec}. In this paper, we explore both model-free and model-based RL algorithms to learn π_{rec}. For model-free recovery, we perform gradient descent on the safety critic $\hat{Q}^{\pi}_{\phi,\text{risk}}(s, \pi_{\text{rec}}(s))$, as in the popular off-policy RL algorithm DDPG [36]. For model-based recovery, we perform model predictive control (MPC) over a learned dynamics model f_θ. For lower dimensional tasks, we utilize the PETS algorithm from Chua et al. [11] to plan over a learned stochastic dynamics model while for tasks with visual observations, we utilize a VAE based latent dynamics model.

Task Policy: We utilize the popular maximum entropy RL algorithm SAC [22] to learn π_{task}, but note that any RL algorithm could be used. Details on the implementation of both policies can be found in the supplement.

Chapter 5

Recovery RL Experiments

In the following experiments, we aim to study whether Recovery RL can (1) more effectively trade off task performance and constraint satisfaction than prior algorithms, which jointly optimize for both and (2) effectively leverage offline data for safe RL.

Domains: We evaluate Recovery RL on a set of 6 simulation domains (Figure 5.1) and an image-based constrained reaching task on a physical robot (Figure 5.3). All experiments involve policy learning under state space constraints, in which a constraint violation terminates the current episode. This makes learning especially challenging, since constraint violations directly preclude further exploration. This setting is reflective of a variety of real world environments, in which constraint violations can require halting the robot due to damage to itself or its surrounding environment.

We first consider three 2D navigation domains: Navigation 1, Navigation 2, and Maze. Here, the agent only observes its position in 2D space and experiences constraint violations if it hits obstacles, walls, or workspace boundaries. We then consider three higher dimensional tasks to evaluate whether Recovery RL can be applied to contact rich manipulation tasks (Object Extraction, Object Extraction (Dynamic Obstacle)) and vision-based continuous control (Image Maze). In the object extraction environments, the goals is to extract the red block without toppling any blocks, and in the case of Object Extraction (Dynamic Obstacle), also avoiding contact with a dynamic obstacle which moves in and out of the workspace. Image Maze is a shorter horizon version of Maze, but the agent is only provided with image observations rather than its (x, y) position in the environment.

We then evaluate Recovery RL on an image-based constrained reaching task on the da Vinci Research Kit (dVRK) [31] where the robot must guide its end effector within 2 mm of a target position from two possible starting locations while avoiding a stay-out zone for the end effector in the center of the workspace. The dVRK is cable-driven and has relatively imprecise controls, motivating closed-loop control strategies to compensate for these errors [28]. Furthermore, the dVRK system has been used in the past to evaluate safe RL algorithms [60] due to its high cost and the delicate structure of its arms, which make safe learning critical. Exact environment, task, and data collection details can be found in the supplement for all simulation and physical experiments.

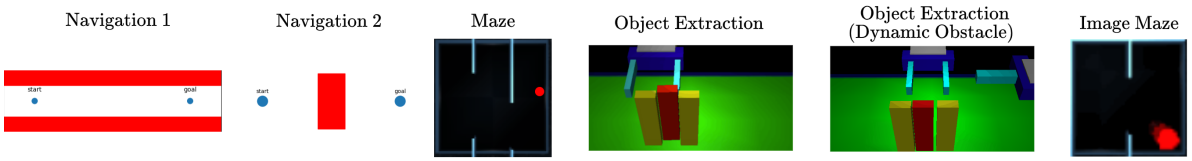


Figure 5.1: **Simulation Experiments Domains:** We evaluate Recovery RL on a set of 2D navigation tasks, two contact rich manipulation environments, and a visual navigation task. In Navigation 1 and 2, the goal is to navigate from the start set to the goal set without colliding into the obstacles (red) while in the Maze navigation tasks, the goal is to navigate from the left corridor to the red dot in the right corridor without colliding into walls/borders. In both object extraction environments, the objective is to grasp and lift the red block without toppling any of the blocks or colliding with the distractor arm (Dynamic Obstacle environment).

Evaluation Metric: Since Recovery RL and prior methods trade off between safety and task progress, we report the ratio of the cumulative number of task successes and the cumulative number of constraint violations at each episode to illustrate this (higher is better). We tune all algorithms to maximize this ratio, and task success is determined by defining a goal set in the state space for each environment. To avoid issues with division by zero, we add 1 to the cumulative task successes and constraint violations when computing this ratio. This metric provides a single scalar value to quantify how efficiently different algorithms balance task completion and constraint satisfaction. We do not report reward per episode, as episodes terminate on task completion or constraint violation. In the supplementary material, we also report additional metrics for each experiment: cumulative task successes and cumulative constraint violations. For all experiments, we replicate each run across 3 random seeds and report the mean and standard error.

Comparisons: We compare Recovery RL to algorithms which ignore constraints (Unconstrained) and enforce constraints by implementing constraints into the policy optimization objective (LR, SQRL, RSPO) or employing reward shaping (RP, RCPO). Specifically, we compare Recovery RL to: **Unconstrained**, where the agent only optimizes for the task reward and ignores constraints, **Lagrangian Relaxation (LR)**, which minimizes $L_{\text{policy}}(s, a, r, s'; \pi) + \lambda(\mathbb{E}_{a\sim\pi(\cdot|s)}\left[\hat{Q}^{\pi}_{\phi,\text{risk}}(s, a)\right] - \epsilon_{\text{risk}})$, where L_{policy} is the policy optimization loss function used and the second term approximately implements the constraint $\hat{Q}^{\pi}_{\phi,\text{risk}}(s, a) \leq \epsilon_{\text{risk}}$, with both updated via dual gradient descent, **Safety Q-Functions for RL (SQRL) [52]**, which combines the LR method with a filtering mechanism to reject policy actions for which $\hat{Q}^{\pi}_{\phi,\text{risk}}(s, a) > \epsilon_{\text{risk}}$, **Risk Sensitive Policy Optimization (RSPO)** [50], where the agent minimizes $L_{\text{policy}}(s, a, r, s'; \pi) + \lambda_t(\mathbb{E}_{a\sim\pi(\cdot|s)}\left[\hat{Q}^{\pi}_{\phi,\text{risk}}(s, a)\right] - \epsilon_{\text{risk}})$, where λ_t is a sequence that decreases to 0, **Reward Penalty (RP)**, in which the agent observes a new reward function that penalizes constraint violations: $R'(s, a) = R(s, a) - \lambda C(s)$, and **Critic Penalty Reward Constrained Policy Optimization (RCPO) [57]**, where the agent optimizes the Lagrangian relaxation via dual gradient descent and the policy gradient trick. The policy gradient update maximizes $\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t(R(s_t, a_t) - \lambda\hat{Q}^{\pi}_{\phi,\text{risk}}(s_t, a_t))\right]$ and the multiplier update is the same as in LR.

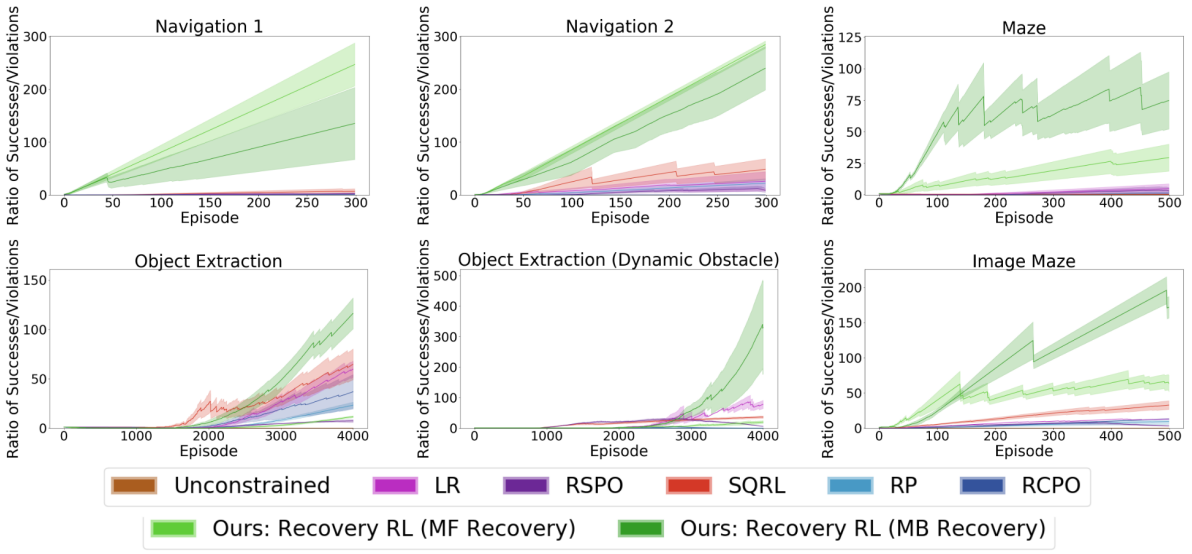


Figure 5.2: **Simulation Experiments:** In all navigation tasks, we find that Recovery RL significantly outperforms prior methods with both model-free and model-based recovery policies, while for the object extraction environments, Recovery RL with a model-based recovery policy significantly outperforms prior algorithms while Recovery RL with a model-free recovery policy does not perform as well. We hypothesize that this is due to the model-based recovery mechanism being better able to compensate for imperfections in $\hat{Q}^{\pi}_{\phi,\text{risk}}$. The sawtooth pattern occurs due to constraint violations, which result in a sudden drop in the ratio.

All of these algorithms are implemented with the same base algorithm for learning the task policy (Soft Actor Critic [22]) and all but Unconstrained and RP are modified to use the same safety critic $\hat{Q}^{\pi}_{\phi,\text{risk}}$ which is pretrained on $\mathcal{D}_{\text{offline}}$ for all methods. Thus, the key difference between Recovery RL and prior methods is how $\hat{Q}^{\pi}_{\phi,\text{risk}}$ is utilized: the comparisons use a joint objective which uses $\hat{Q}^{\pi}_{\phi,\text{risk}}$ to train a single policy that optimizes for both task performance and constraint satisfaction, while Recovery RL separates these objectives across two sub-policies. We tune all prior algorithms and report the best hyperparameter settings found on each task for the ratio based evaluation metric introduced above. See the supplement for ablations studying different hyperparameter choices for Recovery RL and the comparison algorithms, a detailed study of the importance of each component of Recovery RL, and further details on experimental setup and parameters.

5.1 Experiments

Simulation Experiments: We study the performance of Recovery RL and prior methods in all simulation domains in Figure 5.2. Results suggest that Recovery RL with both model-free and model-based recovery mechanisms significantly outperform prior algorithms across all 3 2D pointmass navigation environments (Navigation 1, Navigation 2, Maze) and the visual navigation environment (Image Maze). In the Object Extraction environments, we find that Recovery RL with model-

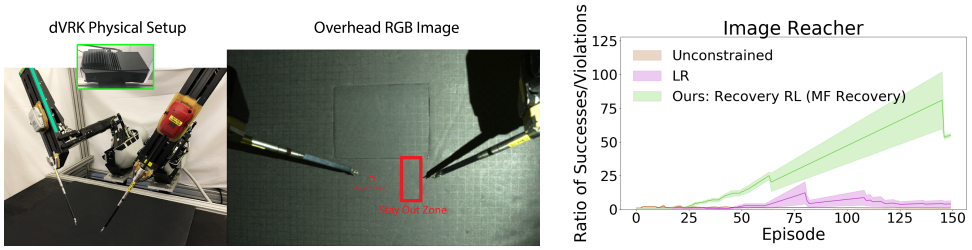


Figure 5.3: **Physical Experiment**: We evaluate Recovery RL on a constrained image-based reacher task on the dVRK with a stay out zone in the center of the workspace. We supply all algorithms with an overhead RGB image as input and find that Recovery RL significantly outperforms Unconstrained and LR.

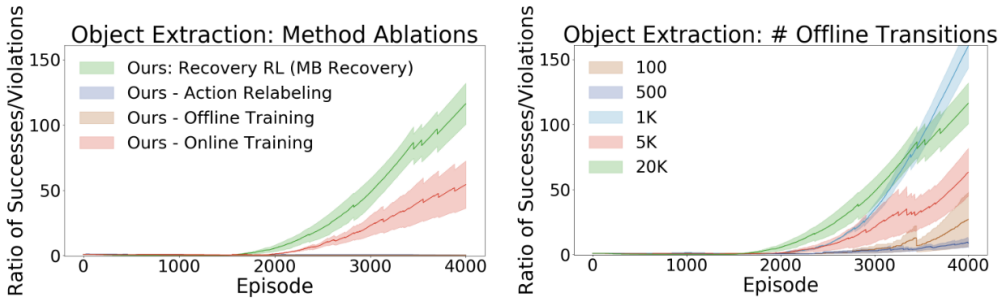


Figure 5.4: **Ablations:** We first study the affect of different algorithmic components of Recovery RL (left). Results suggest that offline pretraining of π_{rec} and $\hat{Q}^{\pi}_{\phi,\text{risk}}$ is critical for good performance, while removing online updates leads to a much smaller reduction in performance. Furthermore, we find that the action relabeling method for training π_{task} (Section 4.1) is critical for good performance. We then study the sensitivity of Recovery RL with model-based recovery to the number of offline transitions used to pretrain π_{rec} and $\hat{Q}^{\pi}_{\phi,\text{risk}}$ (right) and find that Recovery RL performs well even with just 1000 transitions in $\mathcal{D}_{\text{offline}}$ for the Object Extraction task, with performance degrading when the number of transitions is reduced beyond this point.

based recovery significantly outperforms prior algorithms, while Recovery RL with a model-free recovery mechanism does not perform nearly as well. We hypothesize that the model-based recovery mechanism is better able to compensate for noise in $\hat{Q}^{\pi}_{\phi,\text{risk}}$, resulting in a more robust recovery policy. We find that the prior methods often struggle as they tend to sacrifice either safety or task performance, while Recovery RL is generally able to effectively optimize for task performance in the safe MDP defined by the recovery policy. We study this further in the supplement.

Physical Experiment: We evaluate Recovery RL and prior algorithms on an image-based reaching task with delta-position control on the da Vinci Research Kit in Figure 5.3. See Figure 5.3 for an illustration of the experimental setup. We find that Recovery RL substantially outperforms prior methods, suggesting that Recovery RL can be used for visuomotor control on physical robots.

5.2 Ablations:

We ablate different components of Recovery RL and study the sensitivity of Recovery RL to the number of transitions in $\mathcal{D}_{\text{offline}}$ for the Object Extraction domain in Figure 5.4. Results suggest that Recovery RL performs much more poorly when π_{rec} and $\hat{Q}^{\pi}_{\phi,\text{risk}}$ are not pretrained with data from $\mathcal{D}_{\text{offline}}$, indicating the value of learning to reason about safety before environment interaction. However, when π_{rec} and $\hat{Q}^{\pi}_{\phi,\text{risk}}$ are not updated online, performance degrades much less significantly. A key component of Recovery RL is relabeling actions when training the task policy so that π_{task} can learn to associate its proposed actions with their outcome (Section 4.1). We find that without this relabeling, Recovery RL achieves very poor performance as it rarely achieves task successes. Additionally, we find that although the reported simulation experiments supply Recovery RL and all prior methods with $20,000$ transitions in $\mathcal{D}_{\text{offline}}$ for the Object Extraction task, Recovery RL is able to achieve good performance with just 1000 transitions in $\mathcal{D}_{\text{offline}}$, with performance significantly degrading only when the size of $\mathcal{D}_{\text{offline}}$ is reduced to less than this amount.

Chapter 6

MEta-learning for Safe Adaptation (MESA)

6.1 Problem Statement

We consider the offline meta-reinforcement learning problem setting introduced in [39, 12], in which the objective is to leverage offline data from a number of different tasks to rapidly adapt to an unseen task at test-time. We consider an instantiation of this setting in which tasks correspond to CMDPs $\{\mathcal{M}_i\}_{i=1}^{N}$, each with different system dynamics $p_i(s'|s,a)$, but which otherwise share all other MDP parameters, including the same state and action spaces and constraint function. Here the agent is not allowed to directly interact with any environment at meta-train time or meta-test time, but is only provided with a fixed offline dataset of transitions from environments. This setting is particularly applicable to the safe reinforcement learning setting, where direct environmental interaction can be risky, but there may be accident logs from prior robot deployments in various settings. We formalize the problem of learning about constraints in the environment in the context of offline meta-reinforcement learning, in which the agent is provided with offline data from N_{train} training environments $\{\mathcal{M}_i^{\text{train}}\}_{i=1}^{N_{\text{train}}}$ with varying system dynamics and must rapidly adapt to being safe in a new environment $\mathcal{M}^{\text{test}}$ with unseen system dynamics. The intuition is that when dynamics change, the states which violate constraints remain the same, but the behaviors that lead to these states may be very different. Thus, we consider the problem of using data from a number of training environments to optimize the safe RL objective in equation 3.1.

We assume that the agent is provided with a set of N_{train} datasets of offline transitions $\mathcal{D}^{\text{train}} = \{\mathcal{D}_i^{\text{train}}\}_{i=1}^{N_{\text{train}}}$ from training environments with different dynamics in addition to a small dataset $\mathcal{D}^{\text{test}}$ of offline transitions from the test environment $\mathcal{M}^{\text{test}}$, in which the agent is to be deployed. The agent's objective is to leverage this data to optimize the safe RL objective in equation 3.1 in MDP $\mathcal{M}^{\text{test}}$ by learning some task τ in MDP $\mathcal{M}^{\text{test}}$ while minimizing constraint violations.

6.2 Algorithm Description

We introduce MEta-learning for Safe Adaptation (MESA), a 3-phase procedure to optimize the objective in Section 6.1. First, MESA uses datasets of offline transitions from the training envi-

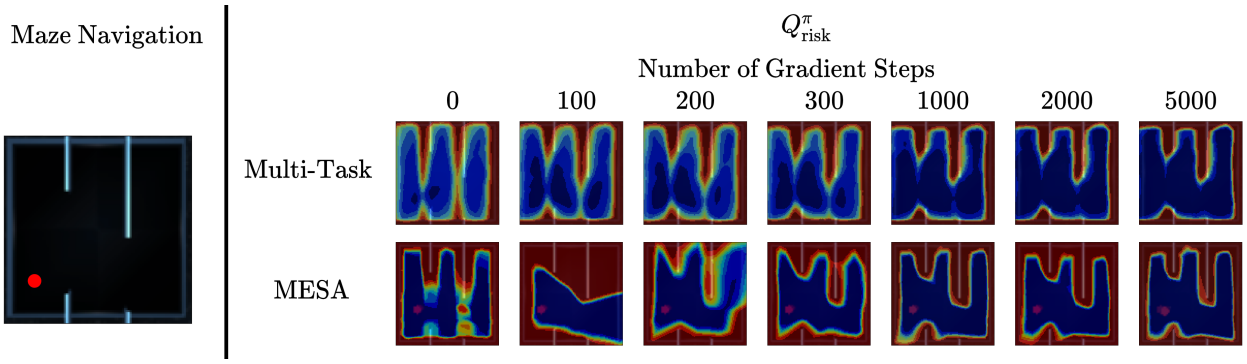


Figure 6.1: **Safety Critic Adaptation Visualizations:** For purposes of illustration, we evaluate MESA and a Multi-Task learning comparison on a simple Maze Navigation task (left) from [59] in which the objective is for the agent (the red dot) to navigate from a random point in the left column to the middle of the right column without colliding into any of the Maze walls or boundaries. Environments are sampled by changing the gaps in the walls (parameterized by $w_1, w_2 \sim \mathcal{U}(-0.1, 0.1)$), leading to significant changes in which behaviors are safe. On the left, we show heatmaps of the learned safety critic Q^{π}_{risk} when it is adapted to a new Maze with unseen wall gaps for the Multi-Task comparison (top) and MESA (bottom). Here bluer colors denote low probability of constraint violation while redder colors denote a higher probability, and the labels above the heatmaps indicate the number of gradient steps used for adaptation on $\mathcal{D}^{\text{test}}$. The Multi-Task learning comparison, which aggregates data from all environments to learn the safety critic and does not explicitly optimize for adaptation, is much slower to adapt to the new environment. However, MESA is able to leverage its learned prior to rapidly adapt to the new gap positions.

ronments to meta-learn a safety critic optimized for rapid adaptation (Phase 1). Then, we discuss how MESA adapts its meta-learned safety critic using a dataset of offline transitions from the test environment (Phase 2). This same dataset is also used to learn a recovery policy, which is trained to descend the safety critic and prevent the agent from visiting unsafe states as in Thananjeyan et al. [59], but we note that the learned safety critic can also be used in conjunction with other safe RL algorithms such as those from Srinivasan et al. [52], Bharadhwaj et al. [8]. Finally, the meta-learned safety critic and recovery policy are used and trained online for safe reinforcement learning when learning some downstream task τ in the testing environment (Phase 3). The full algorithm is summarized in Algorithm 2 and Figure 1.2. An illustration of the adaptation procedure for the safety critic is shown in Figure 6.1.

Phase 1: Meta-Learning Q^{π}_{risk}

Given offline transitions from N_{train} training environments, $\{\mathcal{D}_i^{\text{train}}\}_{i=1}^{N_{\text{train}}}$, we meta-learn the safety critic Q^{π}_{risk} using Model-agnostic Meta Learning [16] (Section 3). We utilize the following safety critic loss function for parameter ψ and transition (s_t, a_t, c_t, s_{t+1}):

$$\begin{aligned}\mathcal{L}_{\text{risk}}(\psi, (s_t, a_t, c_t, s_{t+1})) = (Q^{\pi}_{\psi,\text{risk}}(s_t, a_t) - (c_t \\ + \gamma_{\text{risk}}(1 - c_t)\mathbb{E}_{a_{t+1}\sim\pi(a_{t+1}|s_{t+1})}\left[Q^{\pi}_{\psi,\text{risk,targ}}(s_{t+1}, a_{t+1})\right]))^2_2\end{aligned} \tag{6.1}$$

in each meta-training iteration where c_t indicates a constraint violation at state s_t. Note that Q^{π}_{risk} is initially calibrated to data in the training datasets $\{\mathcal{D}_i^{\text{train}}\}_{i=1}^{N_{\text{train}}}$, which are collected by a mixture of policies.

The recovery policy is not trained with a MAML-style objective. Similar to the actor's loss function in DDPG [36], the recovery policy, parameterized by ω, aims to minimize the safety critic value for input state s_t:

$$\mathcal{L}_{\pi_{\text{rec}}}(\omega, s_t) = Q^{\pi}_{\psi,\text{risk}}(s_t, \pi_{\omega,\text{rec}}(\cdot|s_t)).$$

In principle, this loss function can be replaced with any off-policy algorithm objective, such as AWR and SAC. We overload the loss function notation to also represent empirical risk functions for an input dataset $\mathcal{D}$ such that

$$\mathcal{L}_{\text{risk}}(\psi, \mathcal{D}) = \mathbb{E}_{(s_t,a_t,c_t,s_{t+1})\sim\mathcal{D}}\left[\mathcal{L}_{\text{risk}}(\psi, (s_t, a_t, c_t, s_{t+1}))\right] \tag{6.2}$$

and

$$\mathcal{L}_{\pi_{\text{rec}}}(\omega, \mathcal{D}) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\mathcal{L}_{\pi_{\text{rec}}}(\psi, s_t)\right] \tag{6.3}$$

. When running MAML, we let $\mathcal{L}_{\mathcal{M}_i}(\cdot) = \mathcal{L}_{\text{risk}}(\cdot, \mathcal{D}_i^{\text{train}})$ where $\mathcal{D}_i^{\text{train}}$ denotes a dataset of offline transitions from environment i. The input to the loss is Q_ψ, and MAML optimizes ψ to minimize the time to adapt the safety critic to a new environment.

Phase 2: Test Time Adaptation

A previously unseen test environment $\mathcal{M}^{\text{test}}$ is sampled from $p(\mathcal{M})$ and the agent is supplied with a dataset of offline transitions $\mathcal{D}_{\text{test}}$ from $\mathcal{M}^{\text{test}}$. This dataset is much smaller than the training dataset, since data collection in the test environment results in costly constraint violations. As shown later in Section 7, $\mathcal{D}_{\text{test}}$ is 10-100x smaller than $\mathcal{D}_{\text{train}}$. We then perform M gradient steps with respect to $\mathcal{L}_{\text{risk}}(\psi, \mathcal{D}_{\text{test}})$ as defined in equation 6.2 to rapidly adapt safety critic $Q^{\pi}_{\text{risk},\psi}$ with respect to parameters ψ. We also use this same dataset to train a recovery policy $\pi_{\text{rec},\omega}$ to minimize $\mathcal{L}_{\pi_{\text{rec}}}(\omega, \mathcal{D}_{\text{test}})$ as defined in equation 6.3. Note that the learned $Q^{\pi}_{\text{risk},\psi}$ is initially calibrated with the policy used for data collection in the meta-training and meta-testing environments. Since these datasets are designed to show numerous examples of constraint violations, the resulting $Q^{\pi}_{\text{risk},\psi}$ measures the probability of constraint violation of a policy that is trying to violate constraints, and thus serves as a pessimistic initialization for online learning of some downstream task τ. This is a desirable property, as $Q^{\pi}_{\text{risk},\psi}$ will initially prevent constraint violations, and then become increasingly less pessimistic during online exploration when calibrated with the task policy for task τ.

Phase 3: Using Q^{π}_{risk} and π_{rec} for Safe RL

We initialize the safety critic and recovery policy with the adapted Q^{π}_{risk} and $\pi_{\text{rec},\omega}$ when learning a task τ in the test environment. Note that since the safety critic is learned offline in a task-agnostic

Algorithm 2 MEta-learning for Safe Adaptation (MESA)

Require: Training datasets $\mathcal{D}^{\text{train}} = \{\mathcal{D}_i^{\text{train}}\}_{i=1}^{N_{\text{train}}}$, adaptation dataset $\mathcal{D}^{\text{test}}$, task horizon H, safety threshold ϵ_{risk}, safety critic step sizes α_1 and α_2, recovery policy step size β.

for $i \in \{1, \dots N\}$ **do** ▷ Phase 1: Offline Meta-Learning
  for $j \in \{1, \dots K\}$ **do**
    Sample $\mathcal{D}_j^{\text{train}} \sim \mathcal{D}^{\text{train}}$
    $\psi_j' \leftarrow \psi - \alpha_1 \cdot \nabla_\psi \mathcal{L}_{\text{risk}}\left(\psi, \mathcal{D}_j^{\text{train}}\right)$
  for $j \in \{1, \dots K\}$ **do**
    Sample $\mathcal{D}_j^{\text{test}} \sim \mathcal{D}^{\text{test}}$
  $\psi \leftarrow \psi - \alpha_2 \cdot \sum_j \nabla_\psi \mathcal{L}_{\text{risk}}\left(\psi_j', \mathcal{D}_j^{\text{train}}\right)$
  $\omega \leftarrow \omega - \beta \cdot \nabla_\omega \mathcal{L}_{\pi_{\text{rec}}}\left(\omega, \mathcal{D}^{\text{test}}\right)$
for $i \in \{1, \dots M\}$ **do** ▷ Phase 2: Test Time Adaptation
  $\psi \leftarrow \psi - \alpha_1 \cdot \nabla_\psi \mathcal{L}_{\text{risk}}\left(\psi, \mathcal{D}^{\text{test}}\right)$
  $\omega \leftarrow \omega - \beta \cdot \nabla_\omega \mathcal{L}_{\pi_{\text{rec}}}\left(\omega, \mathcal{D}^{\text{test}}\right)$
$\mathcal{D}^{\text{task}} \leftarrow \emptyset$
while not converged **do** ▷ Phase 3: Recovery RL
  $s_1 \sim$ `env.reset()`
  for $t \in \{1, \dots H\}$ **do**
    $a_t^\pi, a_t^{\text{rec}} \sim \pi_\theta(\cdot|s_t), \pi_{\text{rec}}(\cdot|s_t)$
    if $Q_{\text{risk}}^\pi(s_t, a_t) \leq \epsilon_{\text{risk}}$ **then**
      $a_t = a_t^\pi$
    else
      $a_t = a_t^{\text{rec}}$
    Execute a_t, observe r_t, c_t, and s_{t+1}
    Add $(s_t, a_t^\pi, c_t, s_{t+1})$ to $\mathcal{D}^{\text{test}}$
    Add (s_t, a_t, r_t, s_{t+1}) to $\mathcal{D}^{\text{task}}$
    $\theta \leftarrow \theta - \gamma \cdot \nabla_\theta \mathcal{L}_\pi\left(\theta, \mathcal{D}^{\text{task}}\right)$
    $\psi \leftarrow \psi - \alpha_1 \cdot \nabla_\psi \mathcal{L}_{\text{safe}}\left(\psi, \mathcal{D}^{\text{test}}\right)$
    $\omega \leftarrow \omega - \beta \cdot \nabla_\omega \mathcal{L}_{\pi_{\text{rec}}}\left(\omega, \mathcal{D}^{\text{test}}\right)$
    if c_t **then**
      End episode

way, we can flexibly utilize the meta-learned safety critic and recovery policy to learn a previously unknown task τ in the test environment. As in the Recovery RL paper [59], both Q^{π}_{risk} and π_{rec} are updated online through interaction with the environment so that they are calibrated with the learned task policy for τ and improve their estimates over time.

Chapter 7

MESA Experiments

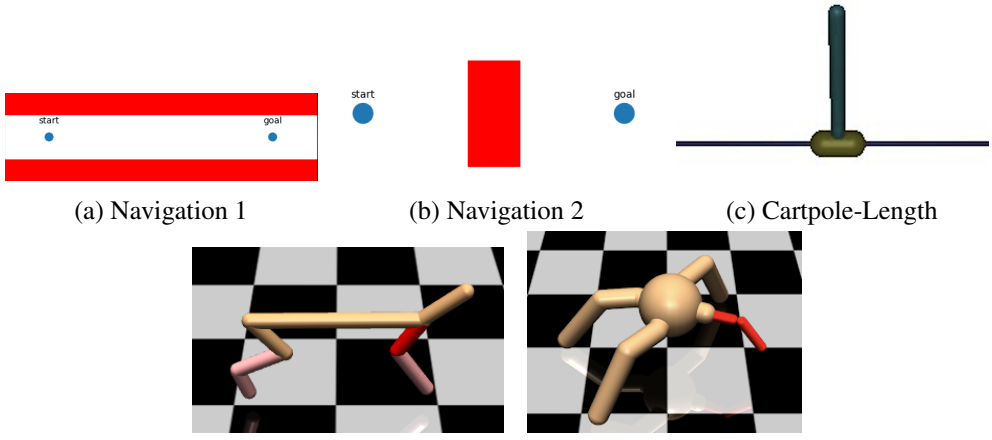
In experiments, we study the degree to which MESA can leverage offline data from environments with different dynamics to quickly learn how to be safe in a new test domain with modified, previously unseen dynamics from a small amount of experience in the new domain. To do this, we compare MESA with prior safe reinforcement learning algorithms and study the degree to which they can limit constraint violations when learning in a perturbed test environment with previously unseen dynamics. MEta-learning for Safe Adaptation (MESA) and all comparisons are built on top of the Soft Actor Critic (SAC) algorithm from Haarnoja et al. [22].

Comparisons: We compare MESA with the following algorithms:

- **Unconstrained:** A soft actor critic agent which only optimizes for task rewards and ignores constraints.
- **Recovery RL (RRL):** Uses data only from $\mathcal{D}_{\text{test}}$ to learn Q^{π}_{risk} and then uses Q^{π}_{risk} in conjunction with the Recovery RL algorithm [59].
- **Multi-Task Learning (Multi-Task):** Learns Q^{π}_{risk} from a combination of all data from both the training datasets $\{\mathcal{D}_i\}_{i=1}^{N_{\text{train}}}$ in phase 1 and then adapts in phase 2 using gradient steps on only the test dataset $\mathcal{D}_{\text{test}}$ (we use the same number of gradient steps for the Multi-Task comparison on $\mathcal{D}_{\text{test}}$ as for MESA). In phase 3, Multi-Task uses the learned Q^{π}_{risk} in conjunction with the Recovery RL algorithm [59] as in MESA and the RRL comparison.

The comparison to the Unconstrained algorithm allows us to evaluate the degree to which reasoning about constraints helps avoid them. The comparison to the Recovery RL algorithm allows us to understand how offline data from different environments allows MESA to learn about constraints in the test environment. Finally, the comparison to the Multi-Task Learning algorithm allows us to evaluate the benefits of specifically leveraging meta-learning to quickly adapt learned risk measures.

Experimental Procedure: We evaluate MESA and comparisons in terms of their ability to (1) efficiently learn some downstream task τ in the test environment and (2) satisfy constraints while doing



(a) Navigation 1 (b) Navigation 2 (c) Cartpole-Length

(d) HalfCheetah-Disabled (e) Ant-Disabled

Figure 7.1: **Simulation Domains:** MESA is evaluated on a set of 2D navigation and locomotion tasks in simulation. In Navigation 1 and Navigation 2, the agent learns to navigate from a beginning position to the goal while avoiding the obstacles (red walls). In the Cartpole-Length task, the goal is to keep the pole balanced on the cart while minimizing the number of times the pole falls beneath the rail or moves off the rail. Lastly, in the HalfCheetah-Disabled and Ant-Disabled tasks, the objective is to learn how to move forwards while minimizing the number of collisions with the ground of the head (HalfCheetah) or torso (Ant) during training.

so. Thus, we report learning curves (total rewards or cumulative task successes) and cumulative number of constraint violations for all algorithms and study whether MESA can leverage prior experience from different environments to learn safely in the test environment. During learning, episodes are terminated upon a constraint violation, making learning about constraints critical for safely learning in the test environment. To control for stochasticity in RL training, we report average performance over 3 to 5 random seeds with standard error shading for all learning curves.

Domains: We evaluate MESA and comparisons on 5 simulation domains which are illustrated in Figure 7.1. All domains we study have the property that the changes in the dynamics are not immediately observable in the agent's observation, motivating learning how to be safe from interaction experience when dynamics change. This is common in various practical settings, such as a robot with worn out joints or sudden power loss in a legged locomotion system. We first consider two 2D navigation domains from [59] in which the agent must navigate between a start set and goal set without colliding into red obstacles in a system with linear Gaussian dynamics. The environment distribution for both domains is defined by varying the coefficients of the A and B matrices in the transition dynamics function where $s_{t+1} = A \cdot s_t + B \cdot a_t + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$.

We then consider a cartpole task (Cartpole-Length) in which the agent must balance the cartpole system without letting the pole fall below the cart. Here environments are sampled by varying the length of the pole, where pole lengths for the training environments are sampled from $\mathcal{U}(0.4, 0.8)$ and the test environment corresponds to a pole of length 1. We also consider two legged locomotion

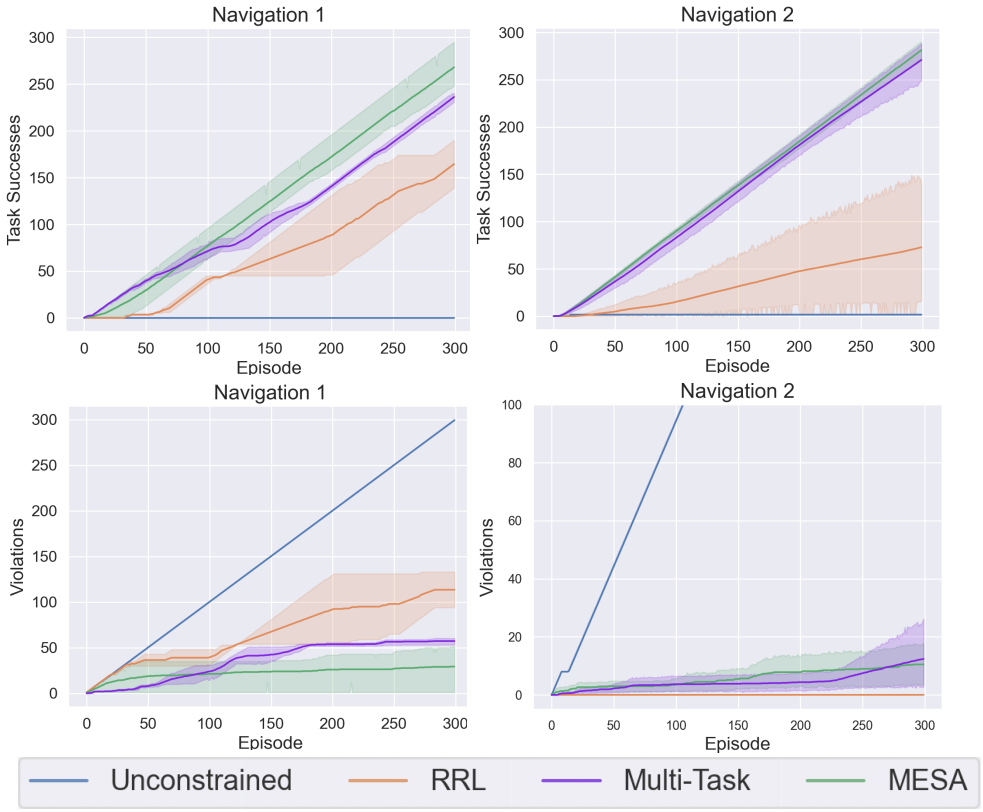


Figure 7.2: **Navigation Results: Top: Learning Curves During Phase 3.** We find that in the Navigation 1 task, MESA learns more efficiently than comparisons, but in the Navigation 2 task, MESA achieves very similar performance to the Multi-Task comparison, which we hypothesize is because small changes in the dynamics of the underlying linear system do not drastically affect the safety of different behaviors in Navigation 2 as the agent can simply learn to keep a large margin between itself and the obstacle. **Bottom: Cumulative Constraint Violations During Phase 3.** We find that MESA violates constraints less often than comparisons for Navigation 1, but performs very similarly to comparisons in terms of cumulative constraint violations in Navigation 2 as the agent can easily learn to keep a large margin between itself and the obstacle.

tasks, HalfCheetah-Disabled and Ant-Disabled, in which the agent is rewarded for running as fast as possible, but violates constraints given a collision of the head with the floor or torso with the floor for the HalfCheetah-Disabled and Ant-Disabled tasks respectively. For both HalfCheetah-Disabled and Ant-Disabled, environments are sampled by choosing a specific joint and simulating a loss of power (power loss corresponds to always providing zero motor torque to the joint), resulting in significantly different locomotion dynamics across environments. The Cartpole-Length and HalfCheetah-Disabled tasks are adapted from [66] while the Ant-Disabled task is adapted from [41].

Data Collection: For the navigation environments, offline datasets for both the training and testing datasets for meta-learning the safety critic are collected by executing a random policy where the

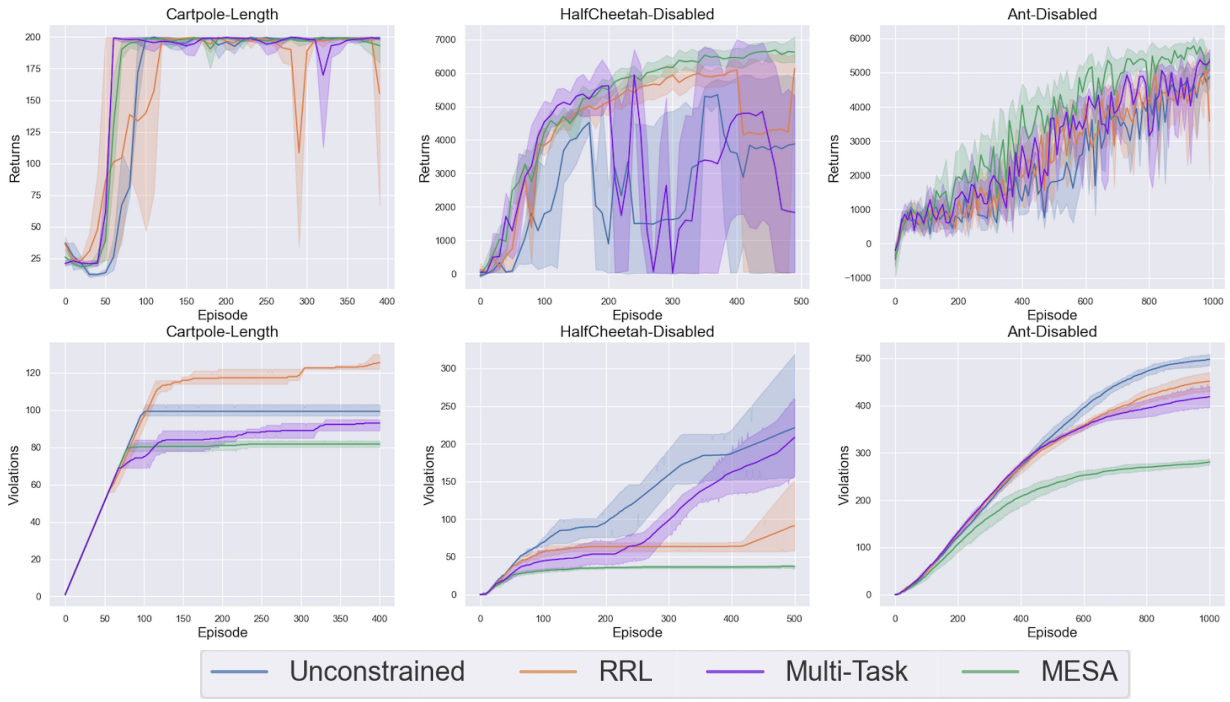
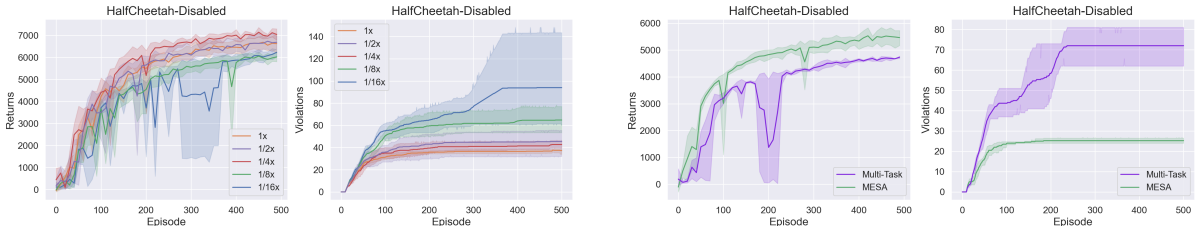


Figure 7.3: **Locomotion Results: Top: Learning Curves During Phase 3.** We find that across all tasks MESA achieves similar task performance as the best comparison algorithm, indicating that MESA is able to effectively learn in a test environment with previously unseen dynamics. **Bottom: Cumulative Constraint Violations During Phase 3.** We find that MESA violates constraints less often than comparisons on all tasks, and this difference is most significant on the HalfCheetah-Disabled and Ant-Disabled tasks, where MESA violates constraints significantly less often than comparisons. This suggests that MESA is able to effectively leverage its prior experiences across environments with different dynamics to rapidly adapt its risk measure to the test environment.

episode *does not terminate* upon constraint violation. We collect a total of 20-25 datasets for each of the sampled training environments, with each dataset consisting of 10000 transitions (680 and 1200 violations in Navigation 1 and Navigation 2 respectively), which is similar to the dataset size collected in [59]. However, the dataset in the test environment is **50-100x** smaller than each training task dataset ($\sim$100, 200 transitions with 15, 36 violations respectively).

Similarly, for locomotion environments, the datasets from the test environment are collected via a random policy rollout, where the episode does not terminate early upon constraint violations. To collect datasets from the training environments, we train SAC on each of the training environments and log the replay buffer from an intermediate checkpoint. For the HalfCheetah-Disabled and Ant-Disabled tasks, we collect 4 and 3 training datasets of 400 episodes (on average $\sim$400K transitions with 14K and 113K violations) respectively. The dataset from the testing environment consists of 40K transitions (2.4K, and 11.2K violations for HalfCheetah, Ant), which is **10x** smaller than before. For the Cartpole-Length task, 20 training datasets are generated, with each containing 200 episodes of data ($\sim$20K timesteps with 4.5K violations). The dataset from the testing environment



(a) Varying Test Dataset Sizes

(b) Test Task Generalization: Partial Joint Failures

Figure 7.4: **Ablation**: **Sensitivity to Test Dataset Size:** In Figure 7.4a, we investigate the sensitivity of MESA to the number of transitions in the test dataset used for adapting Q^{π}_{risk} for the HalfCheetah-Disabled task. We find that even with a test dataset 4 times smaller than used in the experiments in Section 5, MESA does not experience much degradation in performance. However, further reduction in the size of the test dataset make it difficult for MESA to learn a sufficiently accurate safety critic in the test environment, leading to more significant drops in performance. **Generalization to More Different Test Environment Dynamics:** In Figure 7.4b, we investigate MESA's and Multi-Task's generalization to partial joint failures in the HalfCheetah-Disabled task, where the training sets are kept the same. We find that MESA is able to significantly reduce the number of constraint violations compared to the Multi-Task comparison while also achieving superior task performance, suggesting that as differences in system dynamics increase between the training and testing environments, MESA is able to more effectively adapt risk measures across the environments.

contains 1K transitions (with 200 violations), which is **20x** smaller than before.

7.1 Experiments

Navigation Results: We evaluate the performance of MESA and comparisons in Figure 7.2. For both Navigation 1 and 2, unconstrained SAC performs poorly as it no mechanism with which to reason about constraints, and thus collides frequently and is unable to learn the task. In addition, for both environments, MESA violates constraints less often than the Multi-Task comparison, but the performance gap is somewhat small in these environments. We hypothesize that this is because in the Navigation environments, particularly Navigation 2, the space of safe behaviors does not change significantly as a function of the system dynamics, making it possible for the Multi-Task comparison to achieve strong performance by simply learning the safety critic jointly on a buffer of all collected data. For Navigation 2, we observe that the Recovery RL comparison violates the least constraints but does not succeed as often as the Multi-Task comparison and MESA. We hypothesize that since the safety critic for Recovery RL is trained only on the small dataset of transitions in the test environment, it is overly pessimistic when evaluating the risk of out-of-distribution states. This makes it mark all such states as dangerous, making it difficult to explore. This phenomenon is analogous to the over-estimation of Q-values observed in standard offline RL algorithms [32].

Locomotion Results: We also evaluate the performance of MESA on the set of 3 locomotion

environments in Figure 7.3. In Cartpole-Length, the Recovery RL comparison exhibits the most constraint violations, which suggests that the set of transitions from the test environment is too small to learn a sufficiently accurate safety critic. MESA and the Multi-Task comparison achieve somewhat similar performance in this environment, but both are able to leverage their learned prior from the training tasks to achieve somewhat fewer constraint violations than the Unconstrained and Recovery RL comparisons. We hypothesize that the difference in the dynamics between the training and test environments is insufficient for MESA and the Multi-Task comparison to gain sufficient benefit from data in the training environments.

The HalfCheetah-Disabled and Ant-Disabled environments present settings in which the dynamics are much more different between the training and testing environments. Accordingly, MESA significantly outperforms the other methods, including the Multi-Task comparison. We hypothesize that this is because the different training environments are sufficiently different in their dynamics that a safety critic and recovery policy trained jointly on all of them is unable to accurately represent the boundaries between safe and unsafe states. Thus, when adapting to an environment with unseen dynamics, the space of safe behaviors may be so different than in the training environments that the Multi-Task comparison cannot easily adapt. MESA mitigates this issue by explicitly optimizing the safety critic for rapid adaptation to environments with different dynamics.

7.2 Ablations

In ablations, we seek to answer the following questions: (1) how small can the dataset from the test environment be for MESA to safely adapt to new test environments? and (2) how well can MESA generalize to environments consisting of more significantly different dynamics (e.g. partial joint failures when only trained on datasets with examples of full joint failures)?

Test Dataset Size: We first investigate the sensitivty of MESA to the size of the test dataset. Figure 7.4a, we study performance when the test dataset is 1x, 1/2x, 1/4x, 1/8x, and 1/16x the size of the test dataset (40K transitions) used for the HalfCheetah-Disabled results reported in Section 5. We find that MESA can do well when given a test dataset 1/4 the size of the original test dataset (10K transitions, which is 10 episodes of environment interaction). This suggests that the test size dataset can be up to **40x** smaller than the training dataset sizes without significant drop in performance. We find that when the test dataset is reduced to 1/8 and 1/16 the size of the original test dataset, MESA exhibits degrading performance, as the safety critic has insufficient data to learn about constraints in the test environment.

Test Environment Generalization: Here we study how MESA performs when the test environments have more significantly different dynamics from those seen during training. To evaluate this, we consider the HalfCheetah-Disabled task, and train MESA using the same training datasets considered in Section 5, in which specific joints are selected to lose power. However, at test time, we evaluate MESA on a setting with partial power losses to joints, in which the maximum applicable power to certain joints is set to some k percent of the original maximum power, where

$k \in \mathcal{U}(0.5, 0.95)$. This is analogous to partial subsystem failures that can occur in real-world robotic systems. In, Figure 7.4b, we find that MESA achieves superior performance compared to the the Multi-Task comparison in terms of both task performance and constraint violations during training. This suggests that MESA could rapidly learn to be safe even with system dynamics that are out of the meta-training environment distribution.

Chapter 8

Conclusion

We present Recovery RL, a new algorithm for safe RL which is able to (1) efficiently leverage a small set of demonstrations of constraint violations to reduce the probability of constraint violations during learning and (2) effectively balance task-directed exploration and safety by decoupling them across a task policy and a recovery policy. We find that Recovery RL more effectively balances task performance and constraint satisfaction than 5 state-of-the-art prior algorithms for safe RL across 6 simulation domains and an image-based constrained reaching task on a physical robot. Results suggest that Recovery RL could scale well to robotic tasks with complex, contact rich dynamics and high dimensional state spaces such as images.

Next, we investigate safe exploration in the transfer learning setting. We formulate safe reinforcement learning as an offline meta-reinforcement learning problem and motivate how learning from offline datasets of unsafe behaviors in previous environments can provide a scalable and compelling way to learn tasks safely in new environments with unobserved change in system dynamics. We then present MEta-learning for Safe Adaptation (MESA), a new algorithm for learning a risk measure which can transfer knowledge about safety across environments with different dynamics. Results in simulation experiments suggest that MESA is able to achieve strong performance across 5 different robotic simulation domains and is able to effectively adapt to test environments with previously unseen dynamics.

All in all, Recovery RL and MESA are promising directions in safe RL that enable an agent to leverage prior knowledge of safe and unsafe behavior to minimize the number of constraint violations in the test environment. We observe that our agent's performance heavily depends on γ_{risk} and ϵ_{risk}. In addition, the learned safety boundaries are oftentimes too pessimistic, which can impair exploration. Hence, we hope to explore the following research directions:

- Learning Safety Boundaries without an Offline Dataset: To learn without datasets, we believe novel exploration algorithms are needed to guide the agent to quickly understand safety boundaries in as few constraint violations as possible.
- Hyperparameter Optimization: γ_{risk} and ϵ_{risk} should be automatically tuned and change based on the agent's history of constraint violations. For example, the agent should automatically

tune its behavior to be more risky if it wants to better understand an unknown region of the state space.

- Reducing Safety Critic's Pessimism: Learned safety boundaries oftentimes return pessimistic values for states visited outside of the training distribution. We believe that being more conservative in training, as shown by recent work in offline RL [33, 65], can help mitigate this issue.
- More Complex and Diverse Tasks: We hope to extend Recovery RL to learn in more complex manipulation tasks and MESA to meta-learn across a more diverse set of tasks, such as all 50 environments in [64].

RL faces many challenges before it can widely deployed in real-life applications. For an agent to be truly intelligent and reliable, not only must a robot learn to be safe, it must also learn to perform robustly in settings outside of training. During my PhD program at UC Berkeley, I hope to continue my work in safe RL and engage in new, exciting RL applications to better understand and address these roadblocks.

Bibliography

[1] Joshua Achiam et al. “Constrained Policy Optimization”. In: *Proc. Int. Conf. on Machine Learning*. 2017.

[2] Mohammed Alshiekh et al. “Safe Reinforcement Learning via Shielding”. In: 2018.

[3] Eitan Altman. *Constrained Markov Decision Processes*. 1999, p. 260. DOI: 10.1016/0167-6377(96)00003-X.

[4] Karol Arndt et al. “Meta Reinforcement Learning for Sim-to-real Domain Adaptation”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)* (2020).

[5] Somil Bansal et al. “Hamilton-Jacobi Reachability: A Brief Overview and Recent Advances”. In: *Conference on Decision and Control (CDC)*. 2017.

[6] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer.

[7] Felix Berkenkamp et al. “Safe Model-based Reinforcement Learning with Stability Guarantees”. In: *Proc. Advances in Neural Information Processing Systems*. 2017.

[8] Homanga Bharadhwaj et al. “Conservative Safety Critics for Exploration”. In: *arXiv preprint arXiv:2010.14497* (2020).

[9] Y. Chow et al. “A Lyapunov-based Approach to Safe Reinforcement Learning”. In: *NeurIPS*. 2018.

[10] Y. Chow et al. “Lyapunov-based Safe Policy Optimization for Continuous Control”. In: *ICML Workshop RL4RealLife*. 2019.

[11] Kurtland Chua et al. “Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models”. In: *Proc. Advances in Neural Information Processing Systems* (2018).

[12] Ron Dorfman, Idan Shenfeld, and Aviv Tamar. *Offline Meta Learning of Exploration*. 2021. arXiv: 2008.02598 [cs.LG].

[13] Yan Duan et al. *RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning*. 2016. arXiv: 1611.02779 [cs.AI].

[14] Benjamin Eysenbach et al. “Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning”. In: *International Conference on Learning Representations* (2018).

[15] Rasool Fakoor et al. *Meta-Q-Learning*. 2020. arXiv: 1910.00125 [cs.LG].

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. 2017. arXiv: 1703.03400 [cs.LG].

[17] Jaime F. Fisac et al. "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems". In: *IEEE Transactions on Automatic Control*. 2018.

[18] Jaime F. Fisac et al. "Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019.

[19] J. H. Gillula and C. J. Tomlin. "Guaranteed Safe Online Learning via Reachability: tracking a ground target using a quadrotor". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2012.

[20] Djordje Grbic and Sebastian Risi. *Safe Reinforcement Learning through Meta-learned Instincts*. 2020. arXiv: 2005.03233 [cs.LG].

[21] Tuomas Haarnoja et al. "Soft Actor-Critic Algorithms and Applications". In: *CoRR* (2018).

[22] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *Proc. Int. Conf. on Machine Learning* (2018).

[23] Weiqiao Han, Sergey Levine, and Pieter Abbeel. "Learning Compound Multi-Step Controllers under Unknown Dynamics". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2015).

[24] M. Heger. "Consideration of risk in reinforcement learning". In: *Machine Learning Proceedings*. 1994.

[25] Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. "Learning To Learn Using Gradient Descent". In: *IN LECTURE NOTES ON COMP. SCI. 2130, PROC. INTL. CONF. ON ARTI NEURAL NETWORKS (ICANN-2001*. Springer, 2001, pp. 87–94.

[26] Rein Houthooft et al. *Evolved Policy Gradients*. 2018. arXiv: 1802.04821 [cs.LG].

[27] Jan Humplik et al. *Meta reinforcement learning as task inference*. 2019. arXiv: 1905.06424 [cs.LG].

[28] Minho Hwang et al. "Efficiently Calibrating Cable-Driven Surgical Robots With RGBD Sensing, Temporal Windowing, and Linear and Recurrent Neural Network Compensation". In: *Robotics and Automation Letters (RAL)* (2020).

[29] Julian Ibarz et al. "How to Train Your Robot with Deep Reinforcement Learning: Lessons we Have Learned". In: *Int. Journal of Robotics Research (IJRR)* (2021).

[30] Dmitry Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". In: *Conference on Robot Learning (CoRL)* (2018).

[31] Peter Kazanzides et al. "An Open-Source Research Kit for the da Vinci Surgical System". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2014.

[32] Aviral Kumar et al. *Conservative Q-Learning for Offline Reinforcement Learning*. 2020. arXiv: 2006.04779 [cs.LG].

[33] Aviral Kumar et al. *Conservative Q-Learning for Offline Reinforcement Learning*. 2020. arXiv: 2006.04779 [cs.LG].

[34] Lanqing Li, Rui Yang, and Dijun Luo. "Efficient Fully-Offline Meta-Reinforcement Learning via Distance Metric Learning and Behavior Regularization". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=8cpHIfgY4Dj.

[35] Shuo Li and Osbert Bastani. "Robust Model Predictive Shielding for Safe Reinforcement Learning with Stochastic Dynamics". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020.

[36] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *Proc. Int. Conf. on Learning Representations* (2016).

[37] Evan Zheran Liu et al. *Decoupling Exploration and Exploitation for Meta-Reinforcement Learning without Sacrifices*. 2021. arXiv: 2008.02790 [cs.LG].

[38] Nikhil Mishra et al. *A Simple Neural Attentive Meta-Learner*. 2018. arXiv: 1707.03141 [cs.AI].

[39] Eric Mitchell et al. *Offline Meta-Reinforcement Learning with Advantage Weighting*. 2020. arXiv: 2008.06043 [cs.LG].

[40] Anusha Nagabandi et al. "Deep Dynamics Models for Learning Dexterous Manipulation". In: *Conference on Robot Learning (CoRL)* (2019).

[41] Anusha Nagabandi et al. *Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning*. 2019. arXiv: 1803.11347 [cs.LG].

[42] Devang K Naik and Richard J Mammone. "Meta-neural networks that learn by learning". In: *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. Vol. 1. IEEE. 1992, pp. 437–442.

[43] Fritz Wysotzki Peterr Geibel. "Risk-Sensitive Reinforcement Learning Applied to Control under Constraints". In: *Journal of Artificial Intelligence Rersearch*. Vol. 24. 2005.

[44] Kate Rakelly et al. *Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables*. 2019. arXiv: 1903.08254 [cs.LG].

[45] Alex Ray, Joshua Achiam, and Dario Amodei. "Benchmarking Safe Exploration in Deep Reinforcement Learning". In: *NeurIPS Deep Reinforcement Learning Workshop*. 2019.

[46] Charles Richter and Nicholas Roy. "Safe Visual Navigation via Deep Learning and Novelty Detection". In: *Robotics Science and Systems (RSS)* (2013).

[47] Ugo Rosolia and Francesco Borrelli. "Learning Model Predictive Control for Iterative Tasks. A Data-Driven Control Framework". In: *IEEE Transactions on Automatic Control* (2018).

[48] Steindor Saemundsson, Katja Hofmann, and Marc P. Deisenroth. "Meta Reinforcement Learning with Latent Variable Gaussian Processes". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. 2018. URL: https://arxiv.org/abs/1803.07551.

[49] Jurgen Schmidhuber. "Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook". Diploma Thesis. Technische Universitat Munchen, Germany, 14 5 1987. URL: http://www.idsia.ch/~juergen/diploma.html.

[50] Yun Shen et al. "Risk-sensitive Reinforcement Learning". In: *Neural Computation*. Vol. 26. 2014.

[51] Xingyou Song et al. "Rapidly Adaptable Legged Robots via Evolutionary Meta-Learning". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2020).

[52] Krishnan Srinivasan et al. "Learning to be Safe: Deep RL with a Safety Critic". In: *arXiv preprint arXiv:2010.14603* (2020).

[53] Bradly Stadie et al. "The Importance of Sampling inMeta-Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/d0f5722f11a0cc839fa2ca6ea49d8585-Paper.pdf.

[54] A. Tamar, Y. Glassner, and S. Mannor. "Policy Gradients Beyond Expectations: Conditional value-at-risk". In: *CoRR*. 2014.

[55] Pranjal Tandon. *PyTorch implementation of soft actor critic*. https://github.com/pranz24/pytorch-soft-actor-critic. 2020.

[56] Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. "Worst Cases Policy Gradients". In: *Conf. on Robot Learning (CoRL)* (2019).

[57] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. "Reward Constrained Policy Optimization". In: *Proc. Int. Conf. on Learning Representations*. 2019.

[58] Brijen Thananjeyan et al. "ABC-LMPC: Safe Sample-Based Learning MPC for Stochastic Nonlinear Dynamical Systems with Adjustable Boundary Conditions". In: *Workshop on the Algorithmic Foundations of Robotics*. 2020.

[59] Brijen Thananjeyan et al. "Recovery RL: Safe Reinforcement Learning with Learned Recovery Zones". In: *NeurIPS Robot Learning Workshop*. NeurIPS. 2020.

[60] Brijen Thananjeyan et al. "Safety Augmented Value Estimation from Demonstrations (SAVED): Safe Deep Model-Based RL for Sparse Cost Robotic Tasks". In: *Robotics and Automation Letters (RAL)* (2020).

[61] Sebastian Thrun and Lorien Pratt. "Learning to learn: Introduction and overview". In: *Learning to learn*. Springer, 1998, pp. 3–17.

[62] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. "Safe Exploration in Finite Markov Decision Processes with Gaussian Processes". In: *Proc. Advances in Neural Information Processing Systems*. 2016.

[63] Jane X Wang et al. *Learning to reinforcement learn*. 2017. arXiv: 1611.05763 [cs.LG].

[64] Tianhe Yu et al. *Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning*. 2019. arXiv: 1910.10897 [cs.LG].

[65] Tianhe Yu et al. *MOPO: Model-based Offline Policy Optimization*. 2020. arXiv: 2005.13239 [cs.LG].

[66] Jesse Zhang et al. "Cautious Adaptation for Reinforcement Learning in Safety-Critical Settings". In: *Proc. Int. Conf. on Machine Learning* (2020).

[67] Jin Zhang et al. *Learn to Effectively Explore in Context-Based Meta-RL*. June 2020.

Appendix A

Hyperparameters for Recovery RL and Comparisons

We use the same γ_{risk} and ϵ_{risk} for LR, RSPO, SQRL, and RCPO. For LR, RSPO, and SQRL, we find that the initial choice of λ strongly affects the overall performance of this algorithm and heavily tune this. We use the same values of λ for LR and SQRL, and use twice the best value found for LR in as an initialization for the λ-schedule in RSPO. We also heavily tune λ for RP and RCPO. These values are shown for each environment in the tables below.

Algorithm Name	Hyperparameter Format
LR	$(\gamma_{\text{risk}}, \epsilon_{\text{risk}}, \lambda)$
RP	λ
RCPO	$(\gamma_{\text{risk}}, \epsilon_{\text{risk}}, \lambda)$
MF Recovery	$(\gamma_{\text{risk}}, \epsilon_{\text{risk}})$
MB Recovery	$(\gamma_{\text{risk}}, \epsilon_{\text{risk}}, H)$

Table A.1: Hyperparameter Ordering.

	LR	RP	RCPO	MF Recovery	MB Recovery
Navigation 1	$(0.8, 0.3, 5000)$	1000	$(0.8, 0.3, 1000)$	$(0.8, 0, 3)$	$(0.8, 0.3, 5)$
Navigation 2	$(0.65, 0.1, 1000)$	3000	$(0.65, 0.1, 5000)$	$(0.65, 0, 1)$	$(0.65, 0.1, 5)$
Maze	$(0.5, 0.15, 100)$	50	$(0.5, 0.15, 50)$	$(0.5, 0, 15)$	$(0.5, 0.15, 15)$
Object Extraction	$(0.75, 0.25, 50)$	50	$(0.75, 0.25, 50)$	$(0.75, 0, 25)$	$(0.85, 0.35, 15)$
Object Extraction (Dyn. Obstacle)	$(0.85, 0.25, 20)$	25	$(0.85, 0.25, 10)$	$(0.85, 0.35)$	$(0.85, 0.25, 15)$
Image Maze	$(0.65, 0.1, 10)$	20	$(0.65, 0.1, 20)$	$(0.65, 0, 1)$	$(0.6, 0.05, 10)$
Image Reacher	$(0.55, 0.05, 1000)$	N/A	N/A	$(0.55, 0.05)$	N/A

Table A.2: Recovery RL and Comparisons Hyperparameters.

Appendix B

Hyperparameters for MESA and Comparisons

We report global hyperparameters shared across all algorithms in Table B.2 and additionally include domain specific hyperparameters in separate tables in Tables B.3, B.4, and B.5. We use the same base neural network architecture for the safety critic, recovery policy, actor for the task policy, and critic for the task policy. This base network is a fully connected network with 2 hidden layers each with 256 hidden units. For the task policy, we utilize the Soft Actor Critic algorithm from [22] and build on the implementation provided in [55].

B.1 Dataset Details

For all navigation environments, data is collected with a random policy initialized at pre-specified points in the map, with actions sampled uniformly. For Navigation 1, the agent is randomly initialized at various points in the tunnel. For Navigation 2, the agent is randomly initialized at different sides of the rectangular obstacle. Each episode lasts for 10 timesteps for both environments.

For the locomotion environments, datasets are collected by an early-stopped SAC run, where the episode does not end on constraint violation. The testing dataset is collected by a randomly initialized policy. Each episode consists of 1000 timesteps.

DATASET	N_{TRAIN}	$\lvert D_{\text{TRAIN}}\rvert$	$\lvert D_{\text{TEST}}\rvert$
Navigation 1	20	10K	100
Navigation 2	25	10K	200
Cartpole-Length	20	20K	1K
HalfCheetah-Disabled	4	400K	40K
Ant-Disabled	3	400K	40K

Table B.1: Dataset Hyperparameters.

HYPERPARAMETERS	UNCONSTRAINED	RRL	MULTI-TASK	MESA
Phase 1: Offline Training ($\mathcal{D}_{\text{train}}$)				
Total Iterations	—	—	10000	10000
Inner Batch Size $\lvert B_{\text{in}}\rvert$	—	—	—	256
Outer Batch Size $\lvert B_{\text{out}}\rvert$	—	—	—	256
Inner Adaptation Steps	—	—	—	1
Inner LR α_1	—	—	—	0.001
Outer LR α_2	—	—	—	0.00001
Task Batch Size K	—	—	—	5
Adam LR η	—	—	0.0003	—
Batch Size B	—	—	256	—
Phase 2: Offline Finetuning ($\mathcal{D}_{\text{test}}$)				
Total Iterations M	—	10000	500	500
Batch Size B	—	256	256	256
Adam LR	—	0.0003	η	α_1
Phase 3: Online Finetuning				
Adam LR	0.0003	0.0003	η	α_1
Batch Size B	256	256	256	256
Discount γ	0.99	0.99	0.99	0.99
γ_{risk}	—	0.8	0.8	0.8
ϵ_{risk}	—	0.1	0.1	0.1

Table B.2: Algorithm Hyperparameters.

HYPERPARAMETERS	UNCONSTRAINED	RRL	MULTI-TASK	MESA
Phase 2: Offline Finetuning ($\mathcal{D}_{\text{test}}$)				
Total Iterations M	—	2000	100	100
Batch Size B	—	64	64	64
Phase 3: Online Finetuning				
γ_{risk} (Navigation 2)	—	0.65	0.65	0.65
ϵ_{risk} (Navigation 1)	—	0.3	0.3	0.3

Table B.3: Navigation Hyperparameter Differences

HYPERPARAMETERS	UNCONSTRAINED	RRL	MULTI-TASK	MESA
Phase 1: Offline Training ($\mathcal{D}_{\text{train}}$)				
Total Iterations	—	—	15000	15000

Table B.4: HalfCheetah-Disabled Hyperparameter Differences.

HYPERPARAMETERS	UNCONSTRAINED	RRL	MULTI-TASK	MESA
Phase 1: Offline Training ($\mathcal{D}_{\text{train}}$)				
Total Iterations	—	—	15000	15000
Phase 3: Online Finetuning				
Risk Threshold ϵ_{risk}	—	0.3	0.3	0.3

Table B.5: Ant-Disabled Hyperparameter Differences.