Provable Safe Reinforcement Learning with Binary Feedback

Andrew Bennett
Cornell University

Dipendra Misra
Microsoft Research

Nathan Kallus
Cornell University

Abstract

Safety is a crucial necessity in many applications of reinforcement learning (RL), whether robotic, automotive, or medical. Many existing approaches to safe RL rely on receiving numeric safety feedback, but in many cases this feedback can only take binary values; that is, whether an action in a given state is safe or unsafe. This is particularly true when feedback comes from human experts. We therefore consider the problem of provable safe RL when given access to an offline oracle providing binary feedback on the safety of state, action pairs. We provide a novel meta algorithm, SABRE, which can be applied to any MDP setting given access to a blackbox PAC RL algorithm for that setting. SABRE applies concepts from active learning to reinforcement learning to provably control the number of queries to the safety oracle. SABRE works by iteratively exploring the state space to find regions where the agent is currently uncertain about safety. Our main theoretical results shows that, under appropriate technical assumptions, SABRE never takes unsafe actions during training, and is guaranteed to return a near-optimal safe policy with high probability. We provide a discussion of how our meta-algorithm may be applied to various settings studied in both theoretical and empirical frameworks.

1 INTRODUCTION

Reinforcement learning (RL) is an important paradigm that can be used to solve important dynamic decision-making problems in a diverse set of fields, such as robotics, transportation, healthcare, and user assistance. In recent years there has been a significant increase in interest in this problem, with many proposed solutions. However, in many such applications there are important safety considerations that are difficult to address with existing techniques.

Let us consider the running example of a cleaning robot, whose task is to learn how to vacuum the floor of a house. The primary goal of the robot, of course, is to learn to vacuum as efficiently as possible, which may be measured by the amount cleaned in a given time. However, we would also like to impose certain safety constraints on the robot's actions; for example, the robot should not roll off of a staircase where it could damage itself, it should not roll over electrical cords, or it should not vacuum up the owner's possessions. In this example, there are several desirable properties we would like a safety-aware learning algorithm to have, including:

1. The agent should avoid taking any unsafe actions, *even during training*
2. Since it is hard to concretely define a safety function from the robot's sensory observations *a priori*, we would like the agent to *learn* a safety function given feedback of observed states
3. Since the notion of safety is human-defined, and we would like the safety feedback to be manually provided by humans (*e.g.* the owner), we would want the agent to ask for *as little feedback as possible*
4. We would like to use *binary* feedback (*i.e.* is an action in a given state safe or unsafe) rather than numeric feedback, as this is more natural for humans to provide
5. Since the agent may need to act in real time without direct intervention, they should only ask for feedback *offline* in between episodes

Moving away from our specific example, the above five properties would be ideal for safe RL in many applications where safety is naturally human-defined. Unfortunately, there are no existing safe RL methods that can provably satisfy all of these properties. In particular, existing safe RL methods all fail to satisfy these properties for at least one of the following reasons: (1) they assume a safety function is fully known (*e.g.* Chow et al., 2018; Simão et al., 2021); (2) they learn safety using numeric feedback (*e.g.*

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

Wachi and Sui, 2020; Amani et al., 2021); or (3) they require a human-in-the-loop who can intervene and monitors the safety of every action taken in real time (Saunders et al., 2018). In addition, none of these existing methods address the issue of only asking for minimal feedback, which is an important consideration which places the problem at the interface of RL and active learning.

In this paper, we present a novel safe RL framework encompassing the above concerns, involving an offline safety-labeling oracle that can provide binary feedback on the safety of state, action pairs in between episodes. We provide a meta algorithm, SABRE, which can be applied to *any* MDP class within this safety framework, given a blackbox RL algorithm for the MDP class. This algorithm utilizes ideas from disagreement-based active learning, by iteratively exploring the state space to find all regions where there is disagreement on possible safety. Under some appropriate technical assumptions, including that the blackox RL algorithm can provably optimize any given reward, SABRE will satisfy all of the above five properties, and will return an approximately optimal safe policy with high probability. Importantly, we provide high-probability bounds on the number of samples and calls to the labeling oracle needed, and show that they are both polynomial, and that the latter is lower order than the former. Finally, we provide some discussion of how this meta-algorithm approach may be applied in various settings of both theoretical and practical interest.

Math Notation For any natural number $n \in \mathbb{N}$, we let $[n]$ denote the set $\{1, 2, \ldots, n\}$. For any countable set X, we let $\Delta(X)$ denote the space of all probability distributions over X. Given sets X and Y, we let $X \to Y$ denote the set of all functions from X to Y. Lastly, $\text{sign}(y)$ denotes the sign function which takes a value of 1 if $y \geq 0$, or -1 if $y < 0$.

2 PROBLEM SETUP

Markov Decision Process We consider learning in finite-horizon Markov decision processes (MDPs). A *reward-free* MDP is characterized by a tuple $(S, A, T, \cdot, H)$, where S is a given (potentially infinitely large) state space, A is a finite action space with $|A| = A$, $T : S \times A \to \Delta(S)$ is a transition operator, $\in \Delta(S)$ is an initial state distribution, and $H \in \mathbb{N}$ is the horizon. An MDP (with reward) is a tuple $(S, A, T, R, \cdot, H)$, where $R : S \times A \to [0, 1]$ is the reward function. We assume that the transition operator and reward function are time homogeneous; *i.e.*, T and R do not depend explicitly on the time index.[1] We will let (M, R) refer to the MDP we are learning in, where M is the corresponding reward-free MDP.

The agent interacts with an MDP over a series of rounds. In each round the agent successively takes actions according to some *policy*, which is a mapping from states to actions, in order to generate an *episode*. Specifically, in the n'th round for each $n \in \mathbb{N}$, the agent selects some policy $\pi_n \in S \to \Delta(A)$, which it then uses in order to generate an episode $\tau_n = (s_1^n, a_1^n, r_1^n, \ldots, s_H^n, a_H^n, r_H^n, s_{H+1}^n)$, where $s_1^n \sim (\cdot)$, and for each $h \in [H]$ we have $a_h^n \sim \pi_n(\cdot \mid s_h^n)$, $r_h^n = R(s_h^n, a_h^n)$, and $s_{h+1}^n \sim T(\cdot \mid s_h^n, a_h^n)$.

For any given policy π, we let $\mathbb{E}[\cdot]$ and $\mathbb{P}(\cdot)$ denote the expectation and probability operators over trajectories generated using π in the MDP (M, R). Also, for any policy π, we define the *value function* $V^\pi \in S \to \mathbb{R}$ according to $V^\pi(s) = \mathbb{E}[\sum_{h=1}^{H} R(s_h, a_h) \mid s_1 = s]$, and we similarly define the value of the policy according to $V(\pi) = \mathbb{E}_{s \sim (\cdot)}[V^\pi(s)]$.

Our main metric of success for reinforcement learning is *suboptimality* ($\texttt{SubOpt}$). For any policy class $\Pi \subseteq S \to \Delta(A)$ and any policy $\hat{\pi} \in \Pi$, we define $\texttt{SubOpt}(\hat{\pi}; \Pi) = \sup_{\pi \in \Pi} V(\pi) - V(\hat{\pi})$. Then, one of our primary goals is to learn a policy $\hat{\pi}$ that has low suboptimality with respect to a given set of policies that it must choose from, in a relatively small number of episodes.

Over the past few decades, many reinforcement learning algorithms have been developed that can provably obtain low suboptimality using a small number of episodes, for various classes of MDPs . Our focus is on adapting such RL algorithms, in order to additionally take into account safety considerations. For these reasons, we take a meta-learning approach, and assume access to a blackbox reinforcement learning algorithm $\texttt{Alg}$. We formalize this as follows:

Assumption 1. *We have access to a blackbox RL algorithm* $\texttt{Alg}$ *for a set of reward-free MDPs* $\mathcal{M}$ *that contains* M. *Specifically, given* any *reward function* $R' : S \times A \to [0, 1]$ *and policy class* Π *as input, along with any* $\varepsilon, \delta \in (0, 1)$, $\texttt{Alg}(R', \varepsilon, \delta, \Pi)$ *returns a policy* $\hat{\pi} \in \Pi$ *such that* $\texttt{SubOpt}(\hat{\pi}; \Pi) \leq \varepsilon$ *with probability at least* $1 - \delta$. *Furthermore, it is guaranteed to do so after at most* $n_{\texttt{Alg}}(\varepsilon, \delta)$ *episodes, for some* $n_{\texttt{Alg}}(\varepsilon, \delta) = \texttt{Poly}(\varepsilon, \log(1/\delta))$, *while only following policies in* Π.

We refer to $n_{\texttt{Alg}}(\varepsilon, \delta)$ as the *sample complexity* of $\texttt{Alg}$, and note that it implicitly depends on the time horizon H. We also note that the requirement that $\texttt{Alg}$ only follows policies in Π is important, as in practice we will call $\texttt{Alg}$ using classes of policies that are guaranteed to be safe.

Safety Considerations In addition to the general MDP setup presented above, we would also like to take safety considerations into account. Specifically, we will use a binary notion of safety as follows: we assume there exists a function $f^? : S \times A \to \{0, 1\}$ such that $f^?(s, a) = 1$ means that taking action a in state s is safe, and otherwise ac-

[1] This is w.l.o.g. since we can always include the current time index in the observation definition.

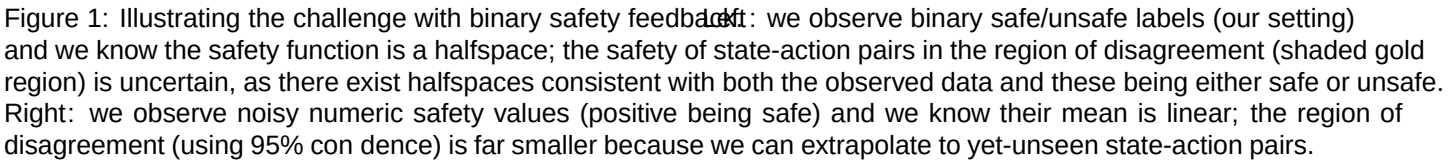

Figure 1: Illustrating the challenge with binary safety feedback. Left: we observe binary safe/unsafe labels (our setting) and we know the safety function is a halfspace; the safety of state-action pairs in the region of disagreement (shaded gold region) is uncertain, as there exist halfspaces consistent with both the observed data and these being either safe or unsafe. Right: we observe noisy numeric safety values (positive being safe) and we know their mean is linear; the region of disagreement (using 95% con dence) is far smaller because we can extrapolate to yet-unseen state-action pairs.

tion a in state s is unsafe. Importantly, we assume that the safety function $f^\star$ is unknown, and we require the agent to only take actions a in states s such that $f^\star(s,a) = 1$, even during training. That is, the agent must learn the safety relation, while simultaneously maintaining safety.

The assumption that the safety values are binary is in contrast to most of the literature, which mostly assumes that these values are numeric (i.e. takes values in $\mathbb{R}$ rather than $\{-1,1\}$, with safety given by $\mathrm{sign}(f^\star(s,a))$; see Section 5 for a detailed discussion). To see why this makes the problem more challenging, consider for example the example displayed in Figure 1. In this example, we consider the case where $\mathcal{S} = \mathbb{R}$, and the safety function is assumed to be linear. On the left we display a series of observed binary safety values for some fixed action, and on the right we display the corresponding observations when the safety values are numeric. In each case, we shade in gold the disagreement region where we are unsure about safety values. With the numeric observations, we can be sure of safety for almost all states except near the $f^\star(s,a) = 0$ margin, whereas with the binary observations we are unsure of safety for all states in between the two observed clusters. Note that this is true even though the numeric observations have noise, while the binary ones don't. In other words, when we receive binary safety feedback, we cannot easily extrapolate safety beyond the observed states.

In order to make the task of learning given these safety constraints feasible, we model $f^\star$ using some class $\mathcal{F} \subseteq \mathcal{S} \times \mathcal{A} \to \{-1,1\}$ of candidate safety functions. This allows us to ensure that $f^\star$ is learnable, by using a sufficiently well-behaved class $\mathcal{F}$. In particular, in our theory we will focus on classes $\mathcal{F}$ with finite Vapnik-Chervonenkis (VC) dimension (Dudley, 1987). Then, in order to make it feasible to guarantee safety during training, we make the following core assumption on the setting.

Assumption 2. The safety class $\mathcal{F}$ is correctly specified; that is, $f^\star \in \mathcal{F}$. Furthermore, there exists a policy $\pi_{\mathrm{safe}} \in \Pi$ that is known to always take safe actions.

To explain the importance of this assumption, let us define the set of safe policies corresponding to each $f \in \mathcal{F}$, by

$$\Pi_{\mathrm{safe}}(f) = \{\pi_{\mathrm{safe}}\} \cup [\{\pi \in \Pi : f(s, \pi(s)) = 1 \text{ a.s. } \forall s \in \mathcal{S}\}].$$

That is $\Pi_{\mathrm{safe}}(f)$ is the set of all policies that would be safe if f was the true safety function, along with the known safe policy. We similarly define the shorthand $\Pi^\star_{\mathrm{safe}} = \Pi_{\mathrm{safe}}(f^\star)$ for the actual set of safe policies. Then, the assumption that $\mathcal{F}$ is correctly specified allows us to reason about which policies are safe, since if $\pi \in \Pi_{\mathrm{safe}}(f)$ for all possible $f \in \mathcal{F}$ that are consistent with our observations so far, we are guaranteed that $\pi \in \Pi^\star_{\mathrm{safe}}$. Furthermore, the existence of a known safe policy π_{safe} allows the agent to avoid getting stuck with no known safe action to perform. Note that the definition of π_{safe} could be problem dependent. For example, in the previous cleaning robot example, the policy that always takes the "don't move" action could qualify. Alternatively, in different applications, π_{safe} could be defined for example by taking a special action that terminates the episode early, or has an expert take control.

Obtaining Safety Feedback A key motivation behind using binary safety feedback rather than continuous is for applications where safety is human-defined. In such applications, obtaining safety labels for (s,a) pairs may be expensive or otherwise burdensome. This motivates an active learning-style approach to the problem, where we have to explicitly ask for labels of (s,a) pairs, with an additional goal of minimizing the number of times we do so.

Concretely, our model for the labeling process is as follows. We assume access to a labeling oracle, which can be given (s,a) pairs and returns the corresponding values of $f^\star(s,a)$. We assume that this oracle can be queried an unlimited number of times in between episodes, and that

it can only be queried with states s that have been previously observed. The reason for the latter restriction is that in many applications the state corresponds to some kind of observation of the environment, such as an image, and therefore the total space of states S is not a-priori known.

Problem Setup Summary. Given a class of MDPs $\mathcal{M}$, a policy class Π, a blackbox RL algorithm Alg, a safety function class $\mathcal{F}$, a desired sub-optimality ϵ, and failure probability δ, we want to propose an online learning algorithm that, for any unknown $M \in \mathcal{M}$ and $f^\star \in \mathcal{F}$, interacts with the MDP over N rounds and returns a policy $\hat{\pi} \in \Pi$ such that, with probability at least $1-\delta$, we have:

1. $SubOpt(\hat{\pi}; \epsilon)$ ($\hat{\pi}$ is approximately optimal)
2. $\hat{\pi} \in \Pi_{safe}$ (the returned policy is safe)
3. $f^\star(s_h^n, a_h^n) = 1$ for all $h \in [H]$ and $n \in [N]$ (the agent never takes unsafe actions during training)

In addition, we would like to establish sample-complexity bounds on the number of episodes N needed to ensure the above, in terms of $1/\epsilon$, $\log(1/\delta)$, H, and $n_{Alg}(\epsilon)$. Finally, we would like to establish corresponding high-probability bounds on the total number of calls to the labelling oracle, which are lower order than the total sample complexity.

3 LEARNING OF SAFETY VIA ACTIVE LEARNING IN REINFORCEMENT LEARNING

First, we establish some basic definitions and a result that will form the basis for safety learning. For any safety-labeled dataset D consisting of $(s, a, f^\star(s,a))$ tuples, we define the corresponding *version space* of safety functions consistent with D by

$$V(D) = \{f \in \mathcal{F} : f(s,a) = c \quad \forall (s,a,c) \in D\}.$$

Similarly, for any given $a \in A$, we can define the corresponding *region of disagreement* of states where safety of that action is not known given D by

$$RD^a(D) = \{s : \exists f, f' \in V(D) \text{ s.t. } f(s,a) \neq f'(s,a)\}.$$

Note that these are both standard definitions in disagreement-based active learning (Hanneke, 2014).

Next, define the set of policies known to surely be safe given D as follows:

$$\Pi(D) = \bigcap_{f \in V(D)} \Pi_{safe}(f).$$

Note that given Assumption 2, $\Pi(D)$ is always ensured to be non-empty, as it will at least contain π_{safe}.

Given these definitions, we have the following lemma.

Lemma 1. *For any safety labeled dataset D, let*

$$U(D) = \sup_{\pi \in \Pi_{safe}} P(\exists h \in [H], a \in A : s_h \in RD^a(D)).$$

Then, for any D and $\hat{\pi} \in \Pi(D)$, we have

$$SubOpt(\hat{\pi}; \Pi_{safe}) \leq SubOpt(\hat{\pi}; \Pi(D)) + HU(D).$$

3.1 Challenges with a Naive Approach

This lemma in fact suggests a rather simple, naive approach, based on greedily trying to follow the blackbox RL algorithm, and switching to π_{safe} and querying the safety oracle whenever we reach a state where safety is not known. Before we outline our novel solution that addresses the challenges outlined in the previous section, we consider the limitations of this naive approach.

Given Lemma 1 and Assumption 1, we can easily ensure sub-optimality of at most $\epsilon + HU(D)$ for any target ϵ using a naive approach like above, where D is the labeled dataset incidentally collected following this approach. However, since this approach does nothing explicit to try to learn the safety and shrink the region of disagreement, it is difficult to provide any guarantees on how fast $U(D)$ shrinks.

A second issue is that this approach provides no control on the number of calls to the labelling oracle. Existing analyses from disagreement-based active learning provide bounds on the number of samples needed to shrink regions of disagreement under fixed distributions, but the agent may roll out with a different distribution every episode.

3.2 The SABRE Algorithm

Motivated by Lemma 1, as well as the limitations of the above naive baseline approach, we now present our novel algorithm in Algorithm 1. This algorithm is super cially similar to the baseline approach, in the sense that it builds a labeled dataset D over a series of rounds, and then estimates an optimal policy following $Alg(R; \Pi(D); \epsilon; \delta)$. However, the difference is in how the safety-labeled dataset D is constructed. Instead of successively trying to optimize the environmental reward, and labeling the states we happen observe in the process, our algorithm uses a strategy for constructing D that explicitly targets $U(D)$.

Our algorithm uses two loops to construct the safety-labeled dataset. The outer loop runs over N epochs. In each epoch it starts with the set Π_n of policies known to be safe at the start of the epoch, and holds this set fixed over the entire epoch. Within each epoch, the inner loop performs B iterations, alternating between the following two steps: (1) (approximately) optimize a policy within Π_n for hitting the region of disagreement with the current D, and (2) roll out with this policy form episodes to collect additional labeled data to expand D with. The reason for having these

Algorithm 1 SAfe Binary-feedback REinforcement learning (SABRE)

Input: Number of epochs N, number of iterations per epoch B, number of rollouts per batch m, accuracy parameters $_{explore}$ and $_R$, and probability parameters $_{explore}$ and $_R$, initial safety-labeled dataset $D_0^{(B)}$ (optional)
Output: Final policy $\hat{}$

for $n = 1; 2; \ldots; N$ do
  $_n$ $(D_{n-1}^{(B)})$ // define a safe policy class for the current safety labeled dataset $D_{n-1}^{(B)}$
  $D_n^{(0)}$ $D_{n-1}^{(B)}$
  for $i = 1; 2; \ldots; B$ do
    $R_n^{(i)}( )$ $1\{9\, a \in A : \in RD^a(D_n^{(i-1)})\}$ // safety reward that incentivizes visiting region of disagreement
    $\hat{}_n^{(i)}$ $Alg(R_n^{(i)}; _n; _{explore}; _{explore})$ // call blackbox RL method with safety exploration reward
    Roll out with $\hat{}_n^{(i)}$ for m episodes, and collect all observed states $S_n^{(i)}$
    Expand $D_n^{(i-1)}$ to $D_n^{(i)}$ by labelling all $s \in S_n^{(i)}$ and $a \in A$ such that $s \in RD^a(D_n^{(i-1)})$
return $Alg(R; _N^{(B)}); _R; _R)$ // return a safe policy by optimizing the environment reward R

two separate loops is that it allows us to derive our formal guarantees, as will be clear in the next section. However, it is possible that an improved analysis in future work could allow the algorithm to be simplified to a single loop.

Comparing with the naive approach discussed above, this algorithm follows a strategy for expanding D based on actually optimizing hitting the region of disagreement for data collection, which is better tailored to explicitly reducing $U(D)$. Furthermore, the data that is used for expanding D within each epoch is collected by rolling out with fixed policies, which allows stronger control on the number of times the safety labeling oracle will be called.

Finally, we note that this algorithm is fairly generic and abstract, and it is not immediately clear how to apply and scale it to practical scenarios. We provide a detailed discussion of this issue in Appendix A. First, we discuss how the constraint of the blackbox algorithm to policies in (D) may be easily implemented for most kinds of general RL algorithms, as long as we can tractably check whether $(s; a)$ is known to be safe or not given D. Then, we discuss how we may check the possible safety of $(s; a)$ given D for various safety classes of the form $F = f(s; a) \mapsto sign(g(s; a)) : g \in G\}$ for some base class $G \subseteq S \times A \to \mathbb{R}$. In particular, when G is a linear class safety checking reduces to solving linear programs, and when G is a kernel class safety checking reduces to solving quadratic programs. Finally, we discuss some heuristic approaches for reducing the amount of computation or labeling required for such implementations, or for dealing with more general safety classes F based on e.g. generic machine learning methods.

4 THEORETICAL ANALYSIS

We now provide a theoretical analysis of our proposed algorithm. First, by design SABRE only ever takes safe actions during training, and the final returned policy $\hat{}$ is always guaranteed to be safe. This holds with certainty, not just with high probability, since by definition (D) contains only safe policies, for any obtainable D. Given this, we will focus on establishing approximate suboptimality, along with high-probability bounds on the sample and labeling complexities.

Before we give our result, we must give some additional technical assumptions. First, we require that the MDP has reasonably low complexity, as follows.

Assumption 3. There exists some positive integer d such that, for any given set of policies $_{safe}$, there exists a set of policies $_1; \ldots; _d$ satisfying

$$\frac{1}{H}\sum_{h=1}^{H} P(s_h \in S) \le \sum_{i=1}^{d} \frac{1}{H}\sum_{h=1}^{H} P_{\pi_i}(s_h \in S);$$

for all measurable $S \subseteq S$ and $\in$.

We call the smallest d for which this assumption holds the policy-cover dimension of the MDP. We give examples of d for some common MDP classes in Section 4.1.

In addition, we need need to assume a bound on the disagreement coefficient of the distribution induced by any policy, which is a standard joint complexity measure of both distribution and hypothesis class, and is used to derive upper and lower bounds for active learning in binary classification (Hanneke, 2014). Formally, for any $h \in [H]$ and $f; f' \in F$, let $E_h(f; f')$ denote the event that $f(s_h; a) \neq f'(s_h; a)$ for some $a \in A$. That is, $E_h(f; f')$ denotes the event that f and f' do not fully agree on the safety of s_h. Then, for any $\in$, $h \in [H]$, and $r \ge 0$, we define the pseudometric $_{h;}$ on F and the corresponding ball $B_{h;}(r)$ about $f^?$ by

$$_{h;}(f; f') = P(E_h(f; f'))$$
$$B_{h;}(r) = \{f \in F : _{h;}(f; f^?) < r\}:$$

Then, for any $\in$, $h \in [H]$, and $r_0 \ge 0$, we define the

disagreement coefficient $_{h;\ }(r_0)$ by

$$_{h;\ }(r_0) = \sup_{r>r_0} \frac{P\ (9f; f\ ^0 2\ B_{h;\ }(r) : E_h(f; f\ ^0))}{r} :$$

Note that clearly by construction $_{h;\ }(r_0)$ is non-increasing in r_0. Then, we require the following technical assumption.

Assumption 4. There exists some fixed $d < 1$ such that $_{h;\ }(0) \ d$ for all $h 2 [H]$ and $2\ _{safe}$.

Note that for some problems we may have $_{h;\ }(0) = 1$. In this case, we can still obtain a similar PAC bound given a complex technical condition in terms of the rate of growth of $_{h;\ }(r)$ as $r\ !\ 0$, which we present in the appendix. However, we focus here on using Assumption 4 instead since it is much simpler and already covers many important settings, such as linear classifiers with bounded density (Hanneke, 2014). We also refer readers to Hanneke (2014) for a detailed discussion of known results on disagreement coefficients.

Given these assumptions, we are ready to present our main theoretical result.

Theorem 2. Let d_{VC} denote the VC dimension of F, and let some $;\ 2\ (0; 1)$ be given. Suppose we run the SABRE algorithm with $N = H$, $B = n_B(\ ; H; d\ )$, $m = n_m(\ ; \ ; H; d\ ; d\ ; d_{VC})$, $_{explore} = \frac{1}{8}H\ ^2$, $_R = \frac{1}{2}$, $_{explore} = \frac{1}{4NB}$, and $_R = \frac{1}{2}$, for some

$$n_B = O\ \log(\ \ ^1)\log(H)d$$

$$n_m = O\ \ ^1\log(\ \ ^1)H^3\log(H)^2d^2d_{VC} :$$

Then, under Assumptions 1 to 4, the returned policy $^{\wedge}$ satisfies $SubOpt(^{\wedge}; \ _{safe}) \ $ with probability at least $1\ \ $.

Ignoring log terms, it follows that the total sample complexity (number of episodes) of our algorithm is at most

$$n_{sample} = \tilde{O}\ H^4\ \ ^1\log(\ \ ^1)d\ d^2d_{VC}$$

$$+ Hd\ \ n_{Alg}(H\ ^2; \ ) :$$

Finally, under an additional event with probability at least $1\ \ $, the number of calls to the labeling oracle with the above settings is bounded by

$$n_{label} = \tilde{O}\ A\log(\ \ ^1)\log(\ \ ^1)H^2d\ d^2d_{VC} :$$

Importantly, Theorem 2 tells us that although we require a sample complexity that depends on $\ $ and H by at least $\tilde{O}(\ \ ^1H^4)$ (possibly greater depending on the complexity of the blackbox algorithm), the corresponding dependence for the sampling oracle is much smaller at $\tilde{O}(\log(\ \ ^1)H^2)$. This is very desirable compared with naive baselines, which may require a number of labels on the same order as the sample complexity. We also note that finite classes F have VC dimension at most $\log_2(|F|)$, so in the case that F is finite we can replace d_{VC} by $\log(|F|)$.

Proof Overview Here we provide a brief overview of the proof of Theorem 2. Full proof details, along with a generalized result using a relaxed version of Assumption 4, as discussed above, are provided in the supplement.

First, by Lemma 1 and the guarantees of the blackbox algorithm, if we can ensure $U(D_N^{(B)}) \ \frac{1}{2}H\ ^1$ with probability at least $1\ \ \frac{1}{2}\ $, then we have $SubOpt(^{\wedge}; \ _{safe})$ with probability at least $1\ \ $ by a union bound. Therefore, the proof focuses on establishing this bound.

Now, define

$$G(\ ; D) = \frac{1}{H}\sum_{h=1}^{H} P\ (9a : s_h\ 2\ RD_h^a(D))$$

and $\ = \frac{1}{4}H\ ^3$. Given Assumption 4, we show that our choice of m ensures that, after rolling out with any fixed $\ $ for m iterations to expand D to D^0, we will have $G(\ ; D^0)\ \ \frac{1}{2}\max(\ ; G(\ ; D))$ with high probability.

Furthermore, given Assumption 3 and any fixed $\ $, if we find $^{\wedge}$ such that $G(^{\wedge}; D)\ \ \sup_{\ 2\ }G(\ ; D)\ \ \frac{1}{2}\ $ (which is ensured by our choice of $_{explore}$ since the expected sum of rewards $R_n^{(i)}$ under $\ $ is given by $HG(\ ; D_n^{(i)})$) and expand D to D^0 such that $G(^{\wedge}; D^0)\ \ \frac{1}{2}\max(\ ; G(^{\wedge}; D))$ at least B times, then at end of the process we have $\sup_{\ 2\ }G(\ ; D)\ \ \ $ with certainty. That is, putting the above together, the inner loop of our algorithm ensures with high probability that $\sup_{\ 2\ _n}G(\ ; D_n^{(B)})\ \ \ $.

Finally, we show that after expanding D to D^0 to ensure that $\sup_{\ 2\ (\ D)}G(\ ; D^0)\ \ \ $ at least H times, we are ensured with certainty that at the end of this process we have $U(D)\ \ \frac{1}{2}H\ ^1$. This follows, intuitively, because after repeating the above process H times, we will always know safety values for the first H actions of any safe policy.

4.1 Examples

Finally we provide some examples that instantiate our theory to some particular classes of MDPs. For each MDP class M considered below we will provide a bound on d, an example blackbox algorithm Alg for this class, its sample complexity n_{Alg}, and the corresponding sample complexity of SABRE. We provide all of these bounds in Table 1, and provide details on how the bounds on d are derived in the appendix.

Tabular MDPs Tabular MDPs is a simple MDP class where S is finite, and no other structure is assumed. Letting $|S| = S$, it is easy to show that $d\ \ S$. In this example, we consider the UCB-VI algorithm (Azar et al., 2017), which is based on value iteration with an exploration reward bonus, and is known to be minimax optimal for this class. Note that Azar et al. (2017) only provided a regret bound; we provide details of how we converted this to a corresponding sample-complexity bound in the appendix.

M	d	bound	Alg	n_{Alg}	Corresponding SABRE sample complexity
Tabular MDP	S	UCB-VI	$\tilde{O}(H^3SA^2\log(1)^4)$	$\tilde{O}(H^8S^2A^2\log(1)^4d^2d_{VC})$
Block MDP	S	HOMER	$\tilde{O}(H^4S^8A^4\log(1))$	$\tilde{O}(H^9S^9A^4\log(1)^2d^2d_{VC})$
Low NNR MDP	d_{NNR}	Rep-UCB	$\tilde{O}(H^5d^4A^2\log(1))$	$\tilde{O}(H^{10}d^5A^2\log(1)^2d^2d_{VC})$

Table 1: Summary of example instantiations of our theory for different MDP classes M, and corresponding blackbox algorithms Alg. We consider UCB-VI (Azar et al., 2017), HOMER (Misra et al., 2020), and Rep-UCB (Uehara et al., 2021). In each case we give a bound on d, then n_{Alg}, and the corresponding sample complexity of SABRE.

Block MDP Block MDP is an MDP class where S can be general, but the transition dynamics are defined by an latent tabular MDP. Specifically, the observed states are sampled iid given the latent discrete state, and it is assumed that the observed state distributions for each latent state do not overlap.[2] Let S denote the number of discrete latent states. Then, it can be shown that we again have $d \le S$. Here we consider the HOMER algorithm of Misra et al. (2020), which works by systematically exploring the latent state space, while simultaneously learning a decoder to map observed to latent states.

Low NNR MDP Finally, we consider the class of MDPs with low non-negative rank (NNR). We define the NNR of an MDP as the smallest integer d_{NNR} such that the transition operator can be written as

$$T(s' \mid s, a) = \sum_{z=1}^{d_{NNR}} P(z \mid s, a) P(s' \mid z)$$

for some latent variable $z \in [d_{NNR}]$. It can easily be shown that this class generalizes both Block MDP and Tabular MDP, and that we always have $d \le d_{NNR}$. For this example we consider the Rep-UCB algorithm (Uehara et al., 2021), which is an algorithm with an optimism-based exploration bonus, which simultaneously learns the low-rank decomposition while exploring.

5 RELATED WORK

There has been a vast body of work on safe RL in recent years; for an overview see for example García and Fernández (2015), Gu et al. (2022), or Brunke et al. (2022). Past work has considered many different approaches, including heuristic deep reinforcement learning methods (Thomas et al., 2021; Luo and Ma, 2021), providing PAC bounds on both sample complexity and number of safety violations (Ding et al., 2021; HasanzadeZonuzy et al., 2021), safety in the context of transfer learning (Srinivasan et al., 2020), safe offline RL (Amani and Yang, 2022), safe multi-agent RL (Lu et al., 2021), or providing benchmark environments for safe RL (Ray et al., 2019).

Within the vast body of work on safe RL, there is a subarea of particular relevance that focuses on guaranteeing safety during all of training. One significant line of work here focuses on the setting where the safety function is unknown and numeric safety feedback is given, and work by leveraging smoothness assumptions on the safety function (Sui et al., 2015; Turchetta et al., 2016; Wachi et al., 2018; Wachi and Sui, 2020; Cheng et al., 2019). However, these papers require numeric feedback and deterministic state transitions, and are limited to continuous control-like settings. A related approach is taken by Berkenkamp et al. (2017), who also consider a similar continuous control-like setting, but define safety based on staying within the region of attraction of the system. A different line focuses on settings where the safety function is known, with diverse approaches including safe versions of value and policy iteration (Chow et al., 2018), reducing to blackbox optimal control problems given differentiability of trajectories (Jin et al., 2021), or using MDP abstractions (Alshiekh et al., 2018; Simão et al., 2021). However, these works are not relevant when safety must be learned. Other works in this sub-area include methods for low-rank MDPs with linear safety functions and numeric feedback (Amani et al., 2021), settings with cost-based safety functions given a total safety budget (Huang et al., 2022), or using a human-in-the-loop who can take control in real time (Saunders et al., 2018; Turchetta et al., 2020; Peng et al., 2022). However, these approaches all fail to meet our requirement of being able to provably ensure safety during training given offline binary feedback. Alternatively, Roderick et al. (2021) using "analogy" relationships to expand known sets of safe state, action pairs, but this is limited to settings where such a relationship is available. We also note that none of the existing papers in this sub-area consider the problem of actively acquiring safety feedback, or minimizing the amount of safety feedback required.

Another relevant sub-field within safe RL considers the very general constrained Markov decision (CMDP) setting (Altman, 1999), where safety is defined by constraints on the total accrued cost $\sum_{h=1}^{H} c(s_h, a_h)$ given one or more cost function c. These constraints may be e.g. bounds on the expected total cost, high probability bounds on the total cost, or other risk bounds on the distribution of the total cost. Note also that cost function in general is unrelated to

[2] This ensures that the latent states are fully recoverable from the observed states, and therefore the process is an MDP.

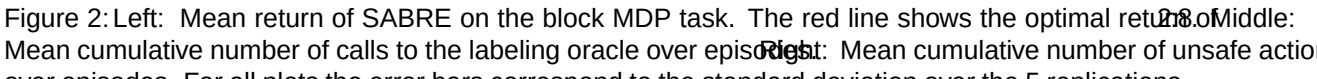

Figure 2: Left: Mean return of SABRE on the block MDP task. The red line shows the optimal return of 2.8. Middle: Mean cumulative number of calls to the labeling oracle over episodes. Right: Mean cumulative number of unsafe actions over episodes. For all plots the error bars correspond to the standard deviation over the 5 replications.

the reward function. Importantly, this is a very general setting that can model binary safety feedback on state, action pairs like we consider, and can even model high-probability safety constraints on states alone rather than state, action pairs (Wagener et al., 2021). There is a vast literature that tackles this more general setting, with approaches based on e.g. applying local policy optimization to the system's Lagrangian (Stooke et al., 2020; Chow et al., 2019; Ray et al., 2019), or safe versions local policy search (Zhang et al., 2020). Of particular interest, Cowen-Rivers et al. (2022) provides a method inspired by active learning, but unlike us they use active learning to generically encourage exploration within the region currently known to be safe, whereas we directly query labels for learning the safety relation. However, although these methods generally try to ensure safety during training, they do not actually provide formal guarantees on this, and indeed in their presented results these methods make some safety violations. In addition, again, none of these works consider the problem of actively acquiring safety feedback, or minimizing the amount of safety feedback required.

Finally, another relevant set of literature is on disagreement-based active learning in the realizable setting (Hanneke, 2014). In particular, both our algorithm and analysis adopt ideas from the CAL algorithm of Cohn et al. (1994), which is a simple approach that labels whenever we observe an input in the region of disagreement. This algorithm was first formally analyzed by Balcan et al. (2006), for agnostic active learning settings, and Hanneke (2007) showed that its sample- and label-complexity can be bounded in terms of the disagreement coefficient. In addition, Hanneke (2011) establishes similar results for the realizable setting. Other works related to this consider, for example, bounding disagreement coefficients for particular settings (for a detailed overview see Hanneke (2014) and references therein), or providing lower and upper bounds for the problem of realizable active learning (Dasgupta, 2005; Balcan et al., 2007). However, these works all consider standard binary classification settings where data is sampled from a fixed distribution, rather than RL settings where we would like to learn classifiers (for safety) that are accurate under the state distributions induced by a wide range of policies.

6 PROOF OF CONCEPT EXPERIMENTS

Finally, we provide some brief "proof of concept" experiments to help highlight the correctness of our theory. The focus here is to demonstrate that SABRE can achieve near-optimal return in a reasonable number of episodes in a non-trivial scenario, while never taking unsafe actions. Code for fully reproducing our experiments is available at https://github.com/CausalML/SABRE .

Environment We consider a particular Block MDP scenario, which was introduced in Section 4.1. This scenario has $A = 4$ actions, a time horizon of $H = 5$, and $S = 4H + 1$ discrete latent states. The agent receives an observed state s along with safety features $\phi(s, a)$ for every action a. The underlying latent state space contains four different paths the agent may take: an *unsafe* path that the agent follows by taking unsafe actions, a *high reward* path where the agent receives high rewards but learns the safety features more slowly; a *low reward* path where the agent receives low rewards but learns safety features more quickly; and a *safe* path which the agent reaches by following a_{safe} which is absorbing and gives no reward. The idea of the scenario is that, to safely optimize reward quickly, the agent should first follow the low-reward path to learn the safety function quickly, rather than greedily try to follow the high-reward path where the safety function is learned more slowly. We provide full details of this environment in the Appendix G.

Blackbox Algorithm We use Proximal Policy Optimization as our blackbox RL algorithm Schulman et al. (2017). PPO is a popular empirical RL method, and while not provably efficient, it is quite effective in practice for problems that do not require strategic exploration. We describe the specific details of our PPO implementation in Appendix G.

Safety Class We let the safety class be given by $\mathcal{F} = \{\text{sign}(w^\top \phi(s,a)) : \|w\|_1 \leq 1\}$. We note that checking whether $s, a \in RD^a(D)$ for any (s,a), as required by SABRE, can be reduced to solving linear programs. We provide details of this in Appendix G.

Results We ran SABRE, implemented with the PPO blackbox algorithm as discussed above on our Block MDP environment, over a total of 7000 episodes. In addition, for comparison we ran the PPO algorithm, directly optimizing reward without any safety considerations. We note that although the total number of episodes is the same for each algorithm, SABRE spends the first 1500 episodes exploring safety while ignoring return, the next 1000 episodes optimizing return, and the last 4500 episodes continuing to roll out with the estimated optimal safe policy. In comparison, the unsafe PPO baseline spends all 7000 episodes optimizing return. We repeated this over 5 random replications, and we present the results in Figure 2, with error bars corresponding to the variance over these replications.

The left figure shows the episodic return against episodes for both SABRE and PPO. In the case of SABRE, as expected, the agent initially receives negative reward as it explores the safety relation along the low reward path, and then at the end once safety is approximately known it computes a near-optimal policy. On the other hand, PPO takes a more direct path, although they both end up with a similarly optimal policy.

The middle figure shows the mean number of calls to the labeling oracle over the same set of episodes. The total number corresponds to only approximately 0.2% of the total number of state, action pairs encountered. This is very encouraging, since although the block MDP environment has a discrete latent state space, the actual state space is infinite, so there is no absolute maximum number of possible calls to the safety oracle.

Next, on the right we plot the mean number of safety violations of the two methods. As expected, the unsafe PPO baseline takes many unsafe actions, while SABRE never takes any unsafe actions, as guaranteed by our theory.

Finally, we provide a more detailed presentation of results in Appendix G. There, we provide similar results for a range of different values on the number of iterations used for safety exploration versus return optimization. In addition, we provide additional results for an implementation of the naive baseline approach described in Section 3.1 which greedily tries to optimize return using PPO while only taking actions known to be safe, and labeling observed states where safety is unknown at the same frequency as for SABRE. In summary, we find that both SABRE and this baseline approach perform very well in practice. However, SABRE is better able to learn the safely function accurately in fewer rounds of data collection for labeling, and consequently obtain a near-optimal policy faster. This is consistent with our theory, since unlike the baseline it can learn to follow the low reward path where it learns safety functions more quickly.

In summary, SABRE is able to achieve optimal return similar to a standard PPO baseline, while taking no unsafe actions, and only making a tiny number of oracle calls. This supports the theoretical findings described above.

7 CONCLUSION

We presented a novel algorithm, SABRE, which addresses the problem of safe RL, where the safety function must be learned via binary feedback that is actively acquired in between episodes. Under appropriate assumptions, SABRE is guaranteed to return a policy that only takes safe actions, and to never take unsafe actions during training. In addition, given access to a PAC blackbox RL algorithm for optimizing arbitrary reward functions in the underlying MDP class, it is guaranteed with high probability an approximately-optimal safe policy with polynomial sample complexity. Furthermore, it only requires labels for a relatively minimal number of state, action pairs, with a label complexity that scales in $H^2 \log(\frac{1}{\delta})$ as opposed to a sample complexity that scales in $H^4 \frac{1}{\delta}$.

There are multiple avenues for future research. First, improved concepts from realizable active learning could be used to further reduce the label complexity of the algorithm, by using a more sophisticated strategy rather than always labeling states in the region of disagreement (Dasgupta, 2005). Second, it is possible that an improved theoretical analysis could find settings under which the simpler baseline approach described in Section 3.1 has some provable guarantees. Indeed, it is apparent from the additional experimental results in Appendix G that this can be a strong algorithm, so it would be interesting to determine both whether it can obtain similar formal guarantees to SABRE, and also whether this baseline approach has any failure cases. Similarly, an improved theoretical analysis could allow SABRE to be simplified or improved; for example, it may be sufficient to have a single loop where the policy class is updated every iteration, or it may be possible to simultaneously perform reward optimization and safety learning. Future work may also consider how to relax the realizability assumption, and provide some kind of guarantees when function approximation is used to model $\mathcal{F}$. This is important, since a major limitation of our work is that the safety guarantees only hold as long as we satisfy Assumption 2. In addition, it may consider how to implement SABRE with more complex choices of $\mathcal{F}$ than the linear classes we considered in our experiments; for example, it may consider classes defined by kernels or neural nets. Finally, it would be exciting to see applications of our theory to practical safety problems.

References

M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

E. Altman. *Constrained Markov Decision Processes*, volume 7. CRC Press, 1999.

S. Amani and L. F. Yang. Doubly pessimistic algorithms for strictly safe off-policy optimization. In *2022 56th Annual Conference on Information Sciences and Systems (CISS)*, pages 113–118. IEEE, 2022.

S. Amani, C. Thrampoulidis, and L. Yang. Safe reinforcement learning with linear function approximation. In *International Conference on Machine Learning*, pages 243–253. PMLR, 2021.

M. G. Azar, I. Osband, and R. Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272. PMLR, 2017.

M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 65–72, 2006.

M.-F. Balcan, A. Broder, and T. Zhang. Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer, 2007.

F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8103–8112, 2018.

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

D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

A. I. Cowen-Rivers, D. Palenicek, V. Moens, M. A. Abdullah, A. Sootla, J. Wang, and H. Bou-Ammar. Samba: Safe model-based & active reinforcement learning. *Machine Learning*, 111(1):173–203, 2022.

S. Dasgupta. Coarse sample complexity bounds for active learning. *Advances in neural information processing systems*, 18, 2005.

D. Ding, X. Wei, Z. Yang, Z. Wang, and M. Jovanovic. Provably efficient safe exploration via primal-dual policy optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 3304–3312. PMLR, 2021.

S. Du, A. Krishnamurthy, N. Jiang, A. Agarwal, M. Dudik, and J. Langford. Provably efficient rl with rich observations via latent state decoding. In *International Conference on Machine Learning*, pages 1665–1674. PMLR, 2019.

R. M. Dudley. Universal donsker classes and metric entropy. *Annals of probability*, 15(4):1306–1326, 1987.

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

S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

S. Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th international conference on Machine learning*, pages 353–360, 2007.

S. Hanneke. Rates of convergence in active learning. *The Annals of Statistics*, pages 333–361, 2011.

S. Hanneke. Theory of active learning. *Foundations and Trends in Machine Learning*, 7(2-3), 2014.

A. HasanzadeZonuzy, A. Bura, D. Kalathil, and S. Shakkottai. Learning with safety constraints: Sample complexity of reinforcement learning for constrained mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7667–7674, 2021.

R. Huang, J. Yang, and Y. Liang. Safe exploration incurs nearly no additional sample complexity for reward-free rl. *arXiv preprint arXiv:2206.14057*, 2022.

W. Jin, S. Mou, and G. J. Pappas. Safe pontryagin differentiable programming. *Advances in Neural Information Processing Systems*, 34:16034–16050, 2021.

S. Lu, K. Zhang, T. Chen, T. Başar, and L. Horesh. Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8767–8775, 2021.

Y. Luo and T. Ma. Learning barrier certificates: Towards safe reinforcement learning with zero training-time vio-

lations. Advances in Neural Information Processing Systems, 34:25621–25632, 2021.

D. Misra, M. Henaff, A. Krishnamurthy, and J. Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In International conference on machine learning, pages 6961–6971. PMLR, 2020.

Z. Peng, Q. Li, C. Liu, and B. Zhou. Safe driving via expert guided policy optimization. In Conference on Robot Learning, pages 1554–1563. PMLR, 2022.

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

P. Rebeschini. Covering numbers bounds for rademacher complexity. chaining, November 2020.

M. Roderick, V. Nagarajan, and Z. Kolter. Provably safe pac-mdp exploration using analogies. In International Conference on Artificial Intelligence and Statistics, pages 1216–1224. PMLR, 2021.

W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans. Trial without error: Towards safe reinforcement learning via human intervention. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, pages 2067–2069, 2018.

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

T. D. Simão, N. Jansen, and M. T. Spaan. Alwayssafe: Reinforcement learning without safety constraint violations during training. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, pages 1226–1235, 2021.

K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn. Learning to be safe: Deep rl with a safety critic, 2020.

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

Y. Sui, A. Gotovos, J. Burdick, and A. Krause. Safe exploration for optimization with gaussian processes. In International Conference on Machine Learning, pages 997–1005. PMLR, 2015.

G. Thomas, Y. Luo, and T. Ma. Safe reinforcement learning by imagining the near future. Advances in Neural Information Processing Systems, 34:13859–13869, 2021.

M. Turchetta, F. Berkenkamp, and A. Krause. Safe exploration in finite markov decision processes with gaussian processes. Advances in Neural Information Processing Systems, 29:4312–4320, 2016.

M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal. Safe reinforcement learning via curriculum induction. Advances in Neural Information Processing Systems, 33:12151–12162, 2020.

M. Uehara, X. Zhang, and W. Sun. Representation learning for online and offline rl in low-rank mdps. arXiv preprint arXiv:2110.04652, 2021.

A. Wachi and Y. Sui. Safe reinforcement learning in constrained markov decision processes. In International Conference on Machine Learning, pages 9797–9806. PMLR, 2020.

A. Wachi, Y. Sui, Y. Yue, and M. Ono. Safe exploration and optimization of constrained mdps using gaussian processes. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

N. C. Wagener, B. Boots, and C.-A. Cheng. Safe reinforcement learning using advantage-based intervention. In International Conference on Machine Learning, pages 10630–10640. PMLR, 2021.

Y. Zhang, Q. Vuong, and K. Ross. First order constrained optimization in policy space. Advances in Neural Information Processing Systems, 33:15338–15349, 2020.

A DISCUSSION OF PRACTICAL IMPLEMENTATIONS OF SABRE

A.1 Implementation of Blackbox RL Algorithm with Policy Constraints

One concern an astute reader may have about our requirement of the blackbox algorithm in Assumption 1 is that it not only needs to be able to optimize any reward function, but it needs to be able to do so over any given policy set Π_0, while only taking actions in $\mathcal{A}_0$. While this may seem like a major requirement and limitation, we explain here why this is not so, and provide a general recipe for dealing with these constraints, as long as we can check whether any given action is known to be safe in any given state s given any safety-labeled dataset $\mathcal{D}$.

First, note that the policy sets Π_0 that we require $\mathcal{A}lg$ to be actually able to work with are all of the form $\Pi_{\text{safe}}(\cdot, \mathcal{D})$. These all look like Π, except with some constraints added on which actions are allowed in any given state based on whether or not that state, action pair is known to be safe or not given $\mathcal{D}$. Now, for any given $\mathcal{D}$, let us consider the MDP $M(\mathcal{D})$, which is defined identically to the actual MDP M, except whenever the agent would take an action a in state s that is not known to be safe according to $\mathcal{D}$, the MDP transitions following $T(\cdot \mid s, \pi_{\text{safe}}(s))$ rather than $T(\cdot \mid s, a)$. Next, note that if we add a layer of abstraction between the agent and the MDP, which checks any action the agent takes, and converts it to $\pi_{\text{safe}}(s)$ if it is not known to be safe, then interacting with M via this abstraction layer is equivalent to interacting with $M(\mathcal{D})$.

Now, let $\pi^\star$ be an α-optimal policy over Π for $M(\mathcal{D})$, and $\tilde{\pi}^\star$ be the policy given by following $\pi^\star$, and converting its action to $\pi_{\text{safe}}(s)$ whenever the action is not known to be safe given $\mathcal{D}$. By construction, following $\pi^\star$ in $M(\mathcal{D})$ is equivalent to following $\tilde{\pi}^\star$ in M, and also by construction the optimal return in M over $\Pi(\mathcal{D})$ is the same as the optimal return in $M(\mathcal{D})$ over Π. Therefore, it is clear that we have $\text{SubOpt}(\tilde{\pi}^\star; \Pi(\mathcal{D}))$ α. Furthermore, as long as the class Π is not defined in some absurd or pathological way, we should have $\pi^\star \in \Pi$, in which case we would also have $\tilde{\pi}^\star \in \Pi(\mathcal{D})$. That is, $\tilde{\pi}^\star$ would be an α-optimal policy over $\Pi(\mathcal{D})$ for M. Note that the above condition that $\pi^\star \in \Pi$ is trivial for example if Π consists of all policies, or all deterministic policies.

Given the above reasoning, as long as the MDP $M(\mathcal{D})$ is still within the class $\mathcal{M}$, we can implement the blackbox algorithm by estimating an α-optimal policy for $M(\mathcal{D})$ over $\Pi(\mathcal{D})$, using the kind of abstraction layer discussed above. Since $M(\mathcal{D}) \in \mathcal{M}$, the blackbox PAC guarantees of Assumption 1 will still apply to this estimation. The returned policy will then correspond to an α-optimal policy over $\Pi(\mathcal{D})$, given by converting potentially unsafe actions to $\pi_{\text{safe}}(s)$ as required. That being said, this is not necessarily the best approach for implementing, since optimizing over the duplicated $\pi_{\text{safe}}(s)$ actions may lead to both computational and statistical inefficiencies. More efficient implementations will be dependent on $\mathcal{M}$ and the choice of blackbox algorithm (we discuss a particular case for a better implementation of PPO with arbitrary $\Pi(\mathcal{D})$ in our experimental details).

Finally, we note that the above argument relied on the assumption that $M(\mathcal{D})$ was still in the class $\mathcal{M}$, so the PAC guarantees of the algorithm would still apply to $M(\mathcal{D})$. This is definitely the case for all of the MDP classes considered in Section 4.1. For example, in the case of low non-negative rank MDPs, the MDP $M(\mathcal{D})$ will have non-negative rank less than or equal to that of M. To see this, note that for any (s, a) that is known to be safe, we still have the decomposition $T(\cdot \mid s, a) = \sum_{z=1}^{d} P(z \mid s, a) P^0(\cdot \mid z)$, for some P and P^0 and for any (s, a) that is not known to be safe we have $T(\cdot \mid s, a) = \sum_{z=1}^{d} P(z \mid s, \pi_{\text{safe}}(s)) P^0(\cdot \mid z)$, which is a decomposition of the same rank. Similar logic easily extends to Tabular and Block MDPs.

A.2 Implementation of Safety Checking

Next, we discuss how we can determine the possible safety of a given (s, a) pair given an arbitrary safety labeled dataset $\mathcal{D}$. Here we will focus on safety classes $\mathcal{F}$ that take the form $\mathcal{F} = \{f(s, a) \mapsto \text{sign}(g(s, a)) : g \in \mathcal{G}\}$ for some base numeric class $\mathcal{G} \subseteq \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Note that this is without loss of generality, since for any $\mathcal{F} \subseteq \mathcal{S} \times \mathcal{A} \to \{\pm 1\}$ we have $\mathcal{F} = \{f(s, a) \mapsto \text{sign}(g(s, a)) : g \in \mathcal{F}\}$. In general given $\mathcal{F}$ specified in such a way, we can reduce the problem of checking the possible safety of (s, a) given $\mathcal{D} = \{(s_1, a_1, c_1), \ldots, (s_n, a_n, c_n)\}$ to computing the maximum and minimum possible values of $g(s, a)$ for $g \in \mathcal{G}$ such that $g(s_i, a_i) c_i \geq 0$ for all $i \in [n]$. If both minimum and maximum are non-negative then the action is known to be safe, if both are negative then the action is known to be unsafe, and otherwise we have $s \in \text{RD}^a(\mathcal{D})$. We discuss below some specific cases of $\mathcal{G}$ where this is a tractable optimization problem.

First, we consider the case that $\mathcal{G}$ is a linear class. Specifically, suppose that $\mathcal{G} = \{f(s, a) \mapsto w^\top \phi(s, a) : \|w\|_1 \leq 1\}$ for some fixed embedding function $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, and some $d \in \mathbb{N}$. In this case, the above safety checking problem reduces to computing the maximum and minimum possible values of $w^\top \phi(s, a)$ such that $w^\top \phi(s_i, a_i) c_i \geq 0$ for all $i \in [n]$, and

$1 \quad w_j \quad 1$ for all $j \in [d]$. We note that this is a linear program with $+2d$ constraints and d variables. Therefore, with linear G, checking safety reduces to linear programming. This is very appealing, since linear programming problems are very quick and tractable to solve, which suggests that such linear classes with an appropriate embedding function may be very practical.

For some scenarios, the above approach based on a linear safety class with a xed embedding function may not be reasonable. One approach in such settings may be to try to learn a safety features embedding function such that we can apply this method. For example, we may try to learn to be able to linearly classify observed training data using deep learning. We could then treat these learnt embeddings as the true ones, and hope that the class $\{(s,a) \mapsto \text{sign}(w^\top \hat{\phi}(s,a) \quad 0 : \|w\|_1 \quad 1\}$ contains $f^?$. Alternatively, we could be more robust by incorporating the error in these embeddings within our safety class; for example, we could consider $\mathcal{F}_{\text{err}} = \{\text{sign}(w^\top (\hat{\phi}(s,a) + g(s,a)) : \|w\|_1 \quad 1, \|g\|_\infty \quad \}$, for some maximum and norm on the error g. In the latter case, if we modeled the error by a reproducing kernel Hilbert space (RKHS), then checking the safety of any (s,a) could be reduced to solving some quadratically constrained linear programs (QCLPs). To see this, note that by the representer theorem, we can optimize over g of the form $g(\cdot,\cdot) = \sum_i \alpha_i K((s,a),(s_i,a_i))$ without loss of generality, in which case $\|g\|_K^2 = \alpha^\top L \alpha$, where K is the kernel function and $L_{i,j} = K((s_i,a_i),(s_j,a_j))$. Although quadratic programs are more expensive to solve than linear programs, this still may be a reasonable approach. Alternatively, if we used a generic class for modeling the error g, such as neural networks, we may be able to solve the corresponding min/max optimization problems using e.g. Lagrangian-based approaches.

An alternative to the above linear class would be to consider kernel-based classes, such as the norm-constrained ball of a reproducing kernel Hilbert space, which are much more expressive. If we use a universal kernel (such as the Gaussian kernel), then this approach has a slight complication that all all possible labellings are feasible, so all points will always be in the region of disagreement. Note that this is still an issue if we have a norm-constraint since $f(s,a) \mapsto \text{sign}(g(s,a)) : \|g\|_K \quad M\} = \{f(s,a) \mapsto \text{sign}(g(s,a)) : \|g\|_K \quad M'\}$ for every $M, M' \in (0, 1]$; this latter fact follows because $\text{sign}(cg(s,a)) = \text{sign}(g(s,a))$ for every $c > 0$. However, we could get around this by enforcing a minimum margin size in the separation of the safe and unsafe points; that is, we could enforce that $|g(s,a)| \geq 1$ for all (s,a), for some g such that $\text{sign}(g(\cdot)) = f^?(\cdot)$ and $\|g\|_K \quad M$. Then, we can determine the possible safety of any (s,a) by computing the minimum and maximum possible values of $g(s,a)$ such that $c_i g(s_i,a_i) \geq 1$ for all safety-labelled triplets (s_i,a_i,c_i), and $\|g\|_K \quad M$. Then, following the respresenter theorem, as described above, these min and max problems reduce to QCLPs.

A.3 Additional Heuristics

Finally, we discuss some additional heuristics that may be useful for practical implementations of SABRE.

In the case that we use a class $\mathcal{F}$ such that safety checking reduces to solving some convex optimization problems, we may apply some heuristics to reduce the number of such problems that need to be solved, or to simplify these problems. As an example, we could justify that some new point (s,a) is known to be safe/unsafe by nding feasible dual solutions to the max/min problems with the same sign; then, for example, if we cache previous dual solutions, then we could check the sign/feasibility of these solutions rst with the new (s,a) to possibly avoid re-solving. In addition, we may improve scalability by identifying redundant labeled triplets (s_i,a_i,c_i) that do not change the set of feasible solutions, and removing these; for example, it is easy to argue that if (s_i,a_i) is not in the region of disagreement given all other labeled triplets, then (s_i,a_i,c_i) is redundant. This may be helpful in reducing the number of constraints needed for solving the required convex optimization problems. Also, when we are required to label a batch of (s,a) at each stage of the inner loop of our algorithm, we may apply heuristics to decide what order to label these in, in order to maximize the chance that the safety of later pairs becomes known during the labeling so fewer labels are needed.

In addition, we could apply heuristic methods for using classes $\mathcal{F}$ for which safety checking does not easily reduce to solving some tractable convex optimization problems. For example, if we t the function $\hat{f}$ based on the labeled data using some generic class $\mathcal{F}$ and method that permits uniform con dence intervals, then we could decide on the possible values of $f^?(s,a)$ based on the con dence interval of $\hat{f}(s,a)$. As another example, we could consider using ensemble methods to estimate $f^?$, and decide on the possible values of $f^?(s,a)$ based on the range of predictions within the ensemble. We caution that such heuristic methods could represent a departure from our theory, and therefore our safety guarantees may no longer hold. However, they may be practical and perform well, and the safety guarantees may still mostly hold in practice if the heuristics are reasonably accurate.

B PROOF OF LEMMA 1

Proof. First, let some arbitrary $\pi^\star \in \arg\max_{\pi \in \Pi_{\text{safe}}} V(\pi)$ be given, and let us define a policy $\tilde{\pi}(\mathcal{D})$ as follows:

- If $s_h \in \widehat{RD}^a(\mathcal{D})$ for any $a \in \mathcal{A}$, then $\tilde{\pi}(\mathcal{D})(\cdot \mid s_h) = \pi^\star(\cdot \mid s_h)$
- Otherwise, $\tilde{\pi}(\mathcal{D})(\cdot \mid s_h) = \pi_{\text{safe}}(\cdot \mid s_h)$

Note that by construction we have $\tilde{\pi}(\mathcal{D}) \in \Pi(\mathcal{D})$, since it only ever takes actions that are known to be safe given $\mathcal{D}$. In addition, let E denote the event where $s_h \in \widehat{RD}^a(\mathcal{D})$ for any $a \in \mathcal{A}$ or $h \in [H]$. Note that under event E, it follows $\tilde{\pi}(\mathcal{D})$ takes actions identically to $\pi^\star$. Therefore, we have

$$\text{SubOpt}(\tilde{\pi}(\mathcal{D}); \pi_{\text{safe}}) \leq H P_{\tilde{\pi}(\mathcal{D})}(\neg E);$$

since the difference in the sum of rewards received between $\pi^\star$ and $\tilde{\pi}(\mathcal{D})$ can never be greater than H. Furthermore, by the definition of $U(\mathcal{D})$ we have $P_\pi(\neg E) \leq U(\mathcal{D})$ for every $\pi \in \Pi_{\text{safe}}$, so therefore

$$\text{SubOpt}(\tilde{\pi}(\mathcal{D}); \pi_{\text{safe}}) \leq H U(\mathcal{D}).$$

Then, given this, we can bound

$$\begin{aligned}
\text{SubOpt}(\hat{\pi}; \pi_{\text{safe}}) &= V(\pi^\star) - V(\hat{\pi}) \\
&= V(\tilde{\pi}(\mathcal{D})) - V(\hat{\pi}) + \text{SubOpt}(\tilde{\pi}(\mathcal{D}); \pi_{\text{safe}}) \\
&\leq \text{SubOpt}(\hat{\pi}; \Pi(\mathcal{D})) + \text{SubOpt}(\tilde{\pi}(\mathcal{D}); \pi_{\text{safe}}) \\
&\leq \text{SubOpt}(\hat{\pi}; \Pi(\mathcal{D})) + H U(\mathcal{D});
\end{aligned}$$

where the second equality follows by adding and subtracting $V(\tilde{\pi}(\mathcal{D}))$ and plugging in the suboptimality definition, the first inequality follows since $\tilde{\pi}(\mathcal{D}) \in \Pi(\mathcal{D})$ by construction, and the second follows from the previous bounds. This is our required bound, so we can conclude.

□

C PROOF OF THEOREM 2

Before we present the main proof, we present some additional lemmas from which the proof can be built. In these lemmas, we will define

$$\widehat{RD}(\mathcal{D}) = \bigcup_{a \in \mathcal{A}} \widehat{RD}^a(\mathcal{D});$$

for every safety-labeled dataset $\mathcal{D}$. Also, let

$$G(\pi; \mathcal{D}) = \frac{1}{H} \sum_{h=1}^{H} P_\pi(s_h \in \widehat{RD}(\mathcal{D}));$$

for any policy π and safety-labeled dataset $\mathcal{D}$.

First, the following lemma provides a guarantee on the behavior of the inner loop of SABRE, as long as n is sufficiently large.

Lemma 3. Let Assumption 3 be given. Suppose that for some given $i \in [N]$ in the execution of Algorithm 1, we have the events

$$G(\hat{\pi}_n^{(i)}; \mathcal{D}_n^{(i-1)}) \geq \sup_{\pi \in \Pi_n} G(\pi; \mathcal{D}_n^{(i-1)}) - \frac{1}{2} \quad \forall i \in [B]; \tag{1}$$

and

$$G(\hat{\pi}_n^{(i)}; \mathcal{D}_n^{(i)}) \geq \frac{1}{2} \max\left\{\alpha; G(\hat{\pi}_n^{(i)}; \mathcal{D}_n^{(i-1)})\right\} \quad \forall i \in [B]; \tag{2}$$

for some $\alpha \in (0, 1)$. In addition, suppose that $B \geq 8d \lceil \log_2(\alpha^{-1}) \rceil$. Then, under these conditions, we are ensured that

$$\sup_{\pi \in \Pi_n} G(\pi; \mathcal{D}_n^{(B)}) \leq \alpha.$$

Next, the following lemma analyzes the outer loop of SABRE, under the assumptions and events detailed in Lemma 3.

Lemma 4. Suppose that the conditions of Lemma 3 holds for every $n \in [N]$, with $\frac{1}{2}H^{-2}$ for some $\in (0; \frac{1}{2})$. Suppose also, that $N \geq H$. Then, the final dataset $D_N^{(B)}$ satisfies

$$U(D_N^{(B)}) \quad :$$

Next, we note that in order to apply the above lemmas, we need to be able to establish the conditions of Lemma 3. For (1), we can rely on the guarantees of the blackbox algorithm Alg. However, for (2), we can instead appeal to an additional lemma. For this lemma, we introduce the following assumption, which is a weaker version of Assumption 4.

Assumption 5. There exists some continuous, non-increasing function $^{?}: R^{+} \to R^{+}$ such that

$$^{?}(r) \quad _{h;}(r) \quad 8r > 0; h \in [H]; \quad \in _{safe}:$$

Furthermore, this function satisfies $\lim_{r \to 1} {}^{?}(r)r = 1$, and $\lim_{r \to 0} r\,{}^{?}r = 0$

Furthermore, we will define the function $_{min}: R^{+} \to R^{+}$, according to

$$_{min}(x) := \inf_{r > 0: r\,{}^{?}(r) \quad x} {}^{?}(r):$$

We note that given Assumption 5, $\{r > 0 : r\,{}^{?}(r) \quad x\}$ must be non-empty for every $x > 0$, so this is a well-defined function. Also, we note that in the case that $^{?}(0) < 1$, we clearly have $_{min}(x) \quad ^{?}(0)$ for every $x > 0$, since $^{?}$ is non-increasing.

Lemma 5. Let Assumption 4 be given. Consider some arbitrary $n \in [N]$ and $i \in [B]$ in the running of Algorithm 1. Then, as long as $m \quad n_m(\ ; \ _{min}; d_{VC}; \ ; H)$, for some

$$n_m(\ ; \ _{min}; d_{VC}; \ ; H) = O\left({}^{-1}\ _{min}(\ =32)^{-2} d_{VC} \log(1=\ ) \log(H) \right);$$

we have

$$G(\hat{}_n^{(i)}; D_n^{(i)}) \quad \frac{1}{2}\max\left\{ \ ; G(\hat{}_n^{(i)}; D_n^{(i\ 1)}) \right\};$$

with probability at least $1 \quad$.

Given the above lemmas, we can provide the main proof.

Proof of Theorem 2. First, we note that applying Lemma 5 with $= \ _{explore}$ and $= \frac{1}{4}H^{-3}$, the corresponding choice of m given by this lemma ensures that Lemma 5 holds at each iteration of the algorithm with probability at least $1 \quad _{explore}$. Then, applying a union bound over these NB events, we have that (2) holds for every $n \in [N]$ with probability at least $1 \quad =4$.

Next, given the guarantee of Alg, and the setting of $_{explore}$ and noting that $HG(\ ; D_n^{(i\ 1)})$ corresponds to the expected sum of rewards policy under $R_n^{(i)}$, we are ensured with probability at least $1 \quad _{explore}$

$$HG(\hat{}_n^{(i)}; D_n^{(i\ 1)}) \quad \sup_{2\ _n} HG(\ ; D_n^{(i\ 1)}) \quad \frac{1}{8}H^{-2}$$
$$() \quad G(\hat{}_n^{(i)}; D_n^{(i\ 1)}) \quad \sup_{2\ _n} G(\ ; D_n^{(i\ 1)}) \quad \frac{1}{2};$$

at any given $i \in [B]$ and $n \in [N]$. Then, by a union bound over all NB calls to Alg during the exploratory loops of the algorithm, and noting the value of $_{explore}$, we have that (1) holds for every $n \in [N]$ and $= \frac{1}{4}H^{-3}$ with probability at least $1 \quad =4$. Then, as long as B is at least $8d \log_2(\frac{1}{4}H^{3\ 1})e$, the conditions of Lemma 3 hold for every $n \in [N]$, so by Lemma 4 we have

$$U(D_N^{(B)}) \quad \frac{1}{2}H^{-1};$$

under the above high probability events. Then, by Lemma 1, we have

$$\begin{aligned}\mathrm{SubOpt}(\hat{\pi}; \pi_{\mathrm{safe}}) &\leq \mathrm{SubOpt}(\hat{\pi}; \mathcal{D}_N^{(B)})) + \mathrm{HU}(\mathcal{D}_N^{(B)}) \\ &\leq \mathrm{SubOpt}(\hat{\pi}; \mathcal{D}_N^{(B)})) + \frac{1}{2} .\end{aligned}$$

Furthermore, following the high-probability guarantee of the final call to Alg, and the settings of δ_R and ϵ_R, we have

$$\mathrm{SubOpt}(\hat{\pi}; \mathcal{D}_N^{(B)})) \leq \frac{1}{2} ;$$

with probability at least $1 - \delta/2$. Then, combining the above three high probability events, and the previous two bounds, a union bound gives us

$$\mathrm{SubOpt}(\hat{\pi}; \pi_{\mathrm{safe}}) \leq \epsilon ;$$

with probability at least $1 - \delta$, as required.

Next, let us analyze the above choices of m and B. Plugging in the settings for N and B, the above choices of m and B satisfy

$$\begin{aligned} m &= O\left(\epsilon^{-1} H^3 \log(H) \cdot \epsilon_{\min}(H^3/128)^2 d_{VC} \log((\delta/(4NB))^{-1})\right) \\ &= O\left(\epsilon^{-1} \log\log(\delta^{-1}) H^3 \log(H)^2 \log\log(H) \log(d) \cdot \epsilon_{\min}(H^3/128)^2 d_{VC} \log(\delta^{-1})\right) \\ &= O\left(\epsilon^{-1} \log(\delta^{-1}) H^3 \log(H)^3 \log(d) \cdot \epsilon_{\min}(H^3/128)^2 d_{VC} \log(\delta^{-1})\right) \end{aligned}$$

and

$$B = O\left(\log(\delta^{-1}) \log(H) d\right) .$$

Then, throwing away log-factors, and noting again that $N = H$, the total number of iterations is given by

$$\begin{aligned} n_{\mathrm{sample}}(\epsilon, \delta, H, d_{\mathcal{X}}, d_{\mathcal{A}}, d_{VC}) &= NB \cdot n_{\mathrm{Alg}}\left(\frac{1}{8}H^2, \delta/(4NB)\right) \\ &\quad + \tilde{O}\left(\epsilon^{-1} H^3 \cdot \epsilon_{\min}(H^3/128)^2 d_{VC} \log(\delta^{-1})\right) \\ &= \tilde{O}\left(Hd \cdot n_{\mathrm{Alg}}(H^2, \delta) + H^3 \cdot \epsilon_{\min}(H^3/128)^2 d_{VC} \log(\delta^{-1})\right) , \end{aligned}$$

where in the second equality we apply the assumption that $n_{\mathrm{Alg}}(\epsilon, \delta)$ is polynomial in ϵ^{-1} and $\log(\delta)$, so therefore hiding log terms we have $n_{\mathrm{Alg}}(\frac{1}{4}H^3, \delta/(4NB)) = \tilde{O}(n_{\mathrm{Alg}}(H^3, \delta))$.

Next, let us analyze the number of calls to the labeling oracle. Following the proof of Lemma 5, for any given h, after labeling at most k states for each $h \in [H]$, for some

$$k = O(\epsilon_{\min}(\epsilon/32)^2 d_{VC} \log(\delta^{-1}) \log(H)) ;$$

we are ensured that, with probability at least δ, the probability of encountering additional $s_h \in \mathrm{RD}(D)$ for each $h \in [H]$ is either reduced by a factor of 4, or that probability was already less than $\epsilon/8$. This implies that after at most

$$\tilde{O}(H \log(\delta^{-1}) \cdot \epsilon_{\min}(\epsilon/32)^2 d_{VC} \log(\delta^{-1}) A)$$

queries to the labeling oracle in a given iteration, we are ensured that the probability of encountering $s_h \in \mathrm{RD}(D)$ is at most $O(\epsilon)$ for every $h \in [H]$, with probability at most $1 - \delta$. Then, applying a union bound over the m rollouts in a single iteration with $\epsilon = \epsilon/m$, after the above maximum number of

$$\tilde{O}(H \log(m) \cdot \epsilon_{\min}(\epsilon/(32m))^2 d_{VC} \log(\delta^{-1})^2 A)$$

queries to the labeling oracle, we will make no further queries in that iteration with probability at least 1. Given this, replacing δ in this bound with $\delta/(2NB)$, and applying a union bound over all iterations, with probability at most $1-\delta$ we make no more than

$$\begin{aligned}&\tilde{O}\left(NBH\log(m)\log(NB)\,\varepsilon_{\min}(\delta/(64mNB))^{-2}d_{VC}\log(\delta^{-1})^2A\right)\\ &=\tilde{O}\left(H^2\log(\delta^{-1})d\,\varepsilon_{\min}(\delta/(64NmB))^{-2}d_{VC}\log(\delta^{-1})^2A\right)\end{aligned}$$

total calls to the labeling oracle.

Finally, we note that for the simplified version of the results presented in the main paper, we assumed that Assumption 4, which is stronger than Assumption 5, and implies that we have $\varepsilon_{\min}(x)\geq d$ for all $x>0$. Therefore, we can obtain the result in the main paper by replacing $\varepsilon_{\min}(\cdot)$ with d everywhere.

□

D PROOFS OF ADDITIONAL LEMMAS

D.1 Proof of Lemma 3

Proof. First, for any $i\in\{0,1,\ldots,B\}$, let

$$\Delta_i=\sup_{\hat{\theta}\in\Theta_n}G(\hat{\theta};D_n^{(i)}).$$

Under the events of the lemma statement, we will argue that

$$\Delta_{8d+i}\leq\max\left\{\varepsilon;\frac{1}{2}\Delta_i\right\}; \tag{3}$$

for every $i\leq B-8d$. Given this result the overall lemma easily follows, by chaining this result s times, where $s=d\log_2(\varepsilon^{-1})$. Under the assumption that $B\geq 8ds$, and the fact that Δ_i is non-increasing, this gives us $\Delta_B\leq\Delta_{8ds}\leq\max\{\varepsilon,2^{-s}\Delta_0\}$. Note that the last inequality follows since our choice of s satisfies $2^{-s}\Delta_0\geq\varepsilon$.

Now, let $\tilde{\pi}_1,\ldots,\tilde{\pi}_d\in\Pi_n$ be the policy cover of Π_n ensured by Assumption 3. Note that for any $\hat{\theta}\in\Theta_n$ and $i\in[B]$, we have

$$\begin{aligned}\sum_{k=1}^{d}G(\tilde{\pi}_k;D_n^{(i-1)})-G(\tilde{\pi}_k;D_n^{(i)})&=\sum_{k=1}^{d}\frac{1}{H}P_{\tilde{\pi}_k}(s_h\in RD(D_n^{i-1})\setminus RD(D_n^i))\\&\quad-\frac{1}{H}P_{\hat{\theta}}(s_h\in RD(D_n^{i-1})\setminus RD(D_n^i))\\&=G(\hat{\theta};D_n^{(i-1)})-G(\hat{\theta};D_n^{(i)}),\end{aligned}$$

where the first and last equalities follow because $RD(D_n^{(j)})\supseteq RD(D_n^{(i-1)})$.

Next, let some arbitrary $i\leq B-8d$ be given, and suppose that (3) does not hold. Given this, it must be the case that $\Delta_{i+j}>\varepsilon$ for all $j\in[8d]$, so for the corresponding iterations (2) can be strengthened to

$$G(\hat{\pi}_n^{(i+j)};D_n^{(i+j)})\geq\frac{1}{2}G(\hat{\pi}_n^{(i+j)};D_n^{(i+j-1)})\quad\forall j\in[8d]. \tag{4}$$

Then, for every $j \in [8d]$ we have

$$\begin{aligned}
&\sum_{k=1}^{d} G(\tilde{\ }_k; D_n^{(i+j-1)}) - G(\tilde{\ }_k; D_n^{(i+j)}) - G(\hat{\ }_n^{(i+j)}; D_n^{(i+j-1)}) + G(\hat{\ }_n^{(i+j)}; D_n^{(i+j)}) \\
&\quad > \frac{1}{2} G(\hat{\ }_n^{(i+j)}; D_n^{(i+j-1)}) \\
&\quad \geq \frac{1}{2} \cdot \frac{1}{2^{i+j-1}} - \frac{1}{4} \cdot \frac{1}{2^{i+8d}} \\
&\quad > \frac{1}{2} \max\left\{0, \frac{1}{2^i} - \frac{1}{4}\right\} \\
&\quad \geq \frac{1}{4} \max\left\{0, \frac{1}{2^i}\right\} \\
&\quad = \frac{1}{8^i},
\end{aligned}$$

where the strict inequality step follows from the assumption that (3) doesn't hold. Then, summing j and noting that this is a telescoping sum, we have

$$\begin{aligned}
\frac{1}{8}(8d)\ \ _i &< \sum_{k=1}^{d} G(\tilde{\ }_k; D_n^{(i)}) - G(\tilde{\ }_k; D_n^{(i+8d)}) \\
&\leq \sum_{k=1}^{d} G(\tilde{\ }_k; D_n^{(i)}) \\
&\leq d \sup_{2} G(\ ; D_n^{(i)}) \\
&= d\ \ _i,
\end{aligned}$$

But this is impossible, because we cannot have $_i < _i$. Therefore, we have proved by contradiction that (3) holds. Also, since we argued the above for an arbitrary B $8d$, (3) must hold for every such i, so we are done.

□

D.2 Proof of Lemma 4

Proof. We will prove that under these conditions, we are ensured that

$$\sup_{2\ _{\text{safe}}} P\left(9h \in [n] : s_h \in RD(D_n^{(B)})\right) \leq \frac{n}{H}, \tag{5}$$

for every $n \in [H]$. The required result then follows from this by plugging in $n = H$, since

$$U(D) = \sup_{2\ _{\text{safe}}} P\left(9h \in [H] : s_h \in RD(D)\right),$$

and because $U(D_N^{(B)}; \ _{\text{safe}}) \leq U(D_H^{(B)})$, as we assumed $N \leq H$.

In the remainder of this proof, we establish (5) for every $n \in [H]$ by forward induction on n.

Base Case

First, for the base case, we note that after the first inner loop, by Lemma 3 we are ensured that

$$\sup_{2\ _1} \frac{1}{H} \sum_{h=1}^{H} P\left(s_h \in RD(D_1^{(B)})\right) \leq \frac{1}{2H^2} \leq \frac{1}{H^2},$$

Therefore, by simple algebra, for every $\in \Pi_1$ we have

$$P_{\pi}(s_1 \in \mathrm{RD}(D_1^{(B)})) \geq \frac{1}{H} \;.$$

Now, for any policy π, the distribution s_1 is always the same (it is always equal to the initial state distribution), so therefore the previous inequality is equivalent to

$$\sup_{\pi \in \Pi_{\text{safe}}} P_{\pi}(s_1 \in \mathrm{RD}(D_1^{(B)})) \geq \frac{1}{H} \;;$$

which establishes the base case.

Inductive Case

Suppose that (5) holds for some arbitrary $n-1 \in [H-1]$. We will argue that it then holds for n. Now, let

$$\pi^\star \in \operatorname*{argmax}_{\pi \in \Pi_{\text{safe}}} P_{\pi}(\exists h \in [n] : s_h \in \mathrm{RD}(D_n^{(B)})) \;.$$

Also, let $\tilde{\pi}$ be some element of Π_n that is defined identically to $\pi^\star$ at every s_h such that $s_h \in \mathrm{RD}(D_n^{(B)})$, and is defined arbitrarily otherwise. Also, let E denote the event that $s_h \notin \mathrm{RD}(D_n^{(B)})$ for every $h \in [n-1]$. Note that under event E, $\pi^\star$ and $\tilde{\pi}$ take identical actions at the first $n-1$ time steps, so therefore the distributions of $s_1, \ldots, s_n$ conditional on E is the same under $\pi^\star$ and $\tilde{\pi}$. Therefore, we can bound

$$\begin{aligned}
&P_{\pi^\star}(\exists h \in [n] : s_h \in \mathrm{RD}(D_n^{(B)})) \\
&= P_{\pi^\star}(E) P_{\pi^\star}(\exists h \in [n] : s_h \in \mathrm{RD}(D_n) \mid E) \\
&\quad + P_{\pi^\star}(\neg E) P_{\pi^\star}(\exists h \in [n] : s_h \in \mathrm{RD}(D_n) \mid \neg E) \\
&\leq P_{\pi^\star}(E) P_{\pi^\star}(\exists h \in [n] : s_h \in \mathrm{RD}(D_n) \mid E) \\
&\quad + \frac{n-1}{H} \, P_{\pi^\star}(\exists h \in [n] : s_h \in \mathrm{RD}(D_n) \mid \neg E) \\
&\leq P_{\tilde{\pi}}(\exists h \in [n] : s_h \in \mathrm{RD}(D_n) \mid E) + \frac{n-1}{H} \\
&\leq \frac{P_{\tilde{\pi}}(\exists h \in [n] : s_h \in \mathrm{RD}(D_n))}{P_{\tilde{\pi}}(E)} + \frac{n-1}{H} \\
&\leq \frac{\sum_{h=1}^{t} P_{\tilde{\pi}}(s_h \in \mathrm{RD}(D_n))}{1 - \frac{n-1}{H}} + \frac{n-1}{H} \\
&\leq 2 \sum_{h=1}^{t} P_{\tilde{\pi}}(s_h \in \mathrm{RD}(D_n)) + \frac{n-1}{H} \;.
\end{aligned}$$

Note that in these bounds we apply the inductive assumption, which gives us

$$\begin{aligned}
P_{\tilde{\pi}}(\cdot \mid E) &= P_{\tilde{\pi}}(\exists h \in [n-1] : s_h \in \mathrm{RD}(D_n^{(B)})) \\
&\leq P_{\tilde{\pi}}(\exists h \in [n-1] : s_h \in \mathrm{RD}(D_{n-1}^{(B)})) \\
&\leq \frac{n-1}{H} \;;
\end{aligned}$$

for every $\tilde{\pi} \in \Pi_{\text{safe}}$. Also, we apply the assumption that $\geq \frac{1}{2}$, which gives us $(1 - \frac{n-1}{H})^{-1} \leq 2$, the fact that $P_{\tilde{\pi}}(\cdot \mid E) = P_{\pi^\star}(\cdot \mid E)$, and the fact that $P(A \mid B) \leq P(A)/P(B)$ for generic events A and B.

Next, given the assumption that we satisfy the conditions of Lemma 3 with $\frac{1}{2}H^{-2}$, we have

$$\sup_{\in \ _n} \frac{1}{H}\sum_{h=1}^{H} P\left(s_h \in RD(D_n^{(B)})\right) \leq \frac{1}{2H^2} :$$

This then implies that

$$\sum_{h=1}^{n} P\left(s_h \in RD(D_n^{(B)})\right) \leq \frac{1}{2H} ;$$

for every $\in \ _n$.

Then, noting that by construction $n \geq 2 \ _n$, putting together the prior bounds gives us

$$P\left(\ ^? \ \exists h \in [n] : s_h \in RD(D_n^{(B)})\right) \leq \frac{1}{H} + \frac{n-1}{H} \leq \frac{n}{H} :$$

Finally, noting the de nition of $^?$, the above is equivalent to

$$\sup_{\in \ _{safe}} P\left(\exists h \in [n] : s_h \in RD(D_n^{(B)})\right) \leq \frac{n}{H} ;$$

which completes the inductive case.

□

D.3 Proof of Lemma 5

Proof. Let D be an arbitrary label dataset, $\in$ be an arbitrary policy to run, and D^0 be the dataset that is obtained by running for m episodes, and for all encountered states s, labeling and adding to D all $(s_h; a)$ such that $s_h \in RD^a(D)$. It is suf cient to argue that, for any choice of and such that

$$G(\ ; D) \geq \frac{1}{2} \ ; \tag{6}$$

we have with probability at least 1 that after performing the above process for episodes that

$$G(\ ; D^0) \geq \frac{1}{2}G(\ ; D) :$$

In the remainder of the proof, we will consider an arbitrarily chosen and that satisfy (6), and we will reason about how large needs to be to ensure the above fact.

First, for every $h \in [H]$ and safety labeled dataset, let

$$_h(D) = P\left(s_h \in RD(D)\right) :$$

Note that according to this de nition, we have $G(\ ; D) = \frac{1}{H}\sum_{h=1}^{H} \ _h(D)$. Now, even though (6) ensures that we can bound $_h(D)$ from below on average across $h \in [H]$, there might be some for which $_h(D)$ is arbitrarily small, or even zero. Therefore, it may be very inef cient or even infeasible to try to ensure that $_h(D^0) \geq \frac{1}{2} \ _h(D)$ simultaneously for every $h \in [H]$.

Instead, we will proceed by de ning a target set $:= \{h \in [H] : \ _h \geq \frac{1}{8} \}$ of time indices for which we would like to

reduce $_h(D)$. Suppose we were able to ensure that $_h(D') \frac{1}{4} _h(D)$ for all $h 2 H$. Then, we would have

$$\begin{aligned}
\frac{1}{H}\sum_{h=1}^{H} {}_h(D') &\frac{1}{4}\frac{1}{H}\sum_{h2H} {}_h(D) + \frac{1}{H}\sum_{h2H} {}_h(D) \\
&\frac{1}{4}\frac{1}{H}\sum_{h=1}^{H} {}_h(D) + \frac{1}{8} \\
&\frac{1}{4}\frac{1}{H}\sum_{h=1}^{H} {}_h(D) + \frac{1}{4}\frac{1}{H}\sum_{h=1}^{H} {}_h(D) \\
&= \frac{1}{2}\frac{1}{H}\sum_{h=1}^{H} {}_h(D);
\end{aligned}$$

where in the final inequality we apply (6). Therefore, if we were able to prove that

$$_h(D') \quad \frac{1}{4} _h(D) \quad \text{w.p. at least } 1 \quad H^{-1} \quad 8h 2 H; \tag{7}$$

then our required result would follow by a union bound over $h 2 H$. Therefore, in the remainder of the proof we will consider an arbitrary h such that $h 2 H$.

Next, let r_h be defined such that

$$r_h \; ^?(r_h) = \frac{1}{4} _h(D):$$

Note that by Assumption 5, $^?(r)$ is continuous and non-decreasing in r with $\lim_{r!0} r \; ^?(r) = 0$, and $\lim_{r!1} r \; ^?(r) = 1$, so clearly such an r_h must exist. Furthermore, given the condition on $_h(D)$ we have $r_h \; ^?(r_h) \quad \frac{1}{32}$, which allows us to bound

$$^?(r_h) \quad \min(=32):$$

Now, let $P^U_{h;}$ denote the distribution of s_h induced by , conditional on $s_h 2 RD(D)$. Given the above definitions, for any given $f 2 B_{h;}(r_h)$, we can bound

$$\begin{aligned}
P^U_{h;}(9a : f(s_h;a) 6 f^?(s_h;a)) &= \frac{P(9a : f(s_h;a) 6 f^?(s_h;a))}{_h(D)} \\
&\frac{r_h}{_h(D)} \\
&= \frac{1}{4} _h(r_h^0) \quad 1 \\
&\frac{1}{4} \min(=32) \quad 1:
\end{aligned} \tag{8}$$

Next, suppose we draw k iid samples $s_h^{(1)}; : : : ; s_h^{(k)}$ from $P^U_{h;}$, and let $F_h = f f 2 F : f 2 B_{h;}(r_h)g$. Now, suppose further that it were the case that for every $f 2 F_h$ we had $f(s_h^{(i)};a) 6 f^?(s_h^{(i)};a)$ for some $i 2 [k]$ and $a 2 A$. If this were the case, then by labeling these k points we would eliminate all hypotheses outside of $B_{h;}(r_h)$. Therefore, letting D_k denote the dataset obtained by labeling all of these points and adding them to D, we would have

$$\begin{aligned}
_h(D_k) &= P(s_h 2 RD(V(D_k))) \\
&P(s_h 2 RD(B_h(r_h))) \\
&_{h;}(r_h) r_h \\
&^?(r_h) r_h \\
&= \frac{1}{4} _h(D);
\end{aligned}$$

which is our required condition. Therefore, we can proceed by: (1) finding k such that the above occurs with probability at least $1 \quad =(2H)$; and (2) finding m such that event $s_h 2 RD(D)$ occurs at least k times with probability at least $1 \quad =(2H)$. Then, applying a union bound we would be done. We will do these things one at a time.

Finding sufficiently large k

First, let us define the following stochastic process indexed by $\mathcal{F}_h$:

$$G_k(f) = \frac{1}{k}\sum_{i=1}^{k} c(s_h^{(i)}; f) - \mathbb{E}_{;h}^{U}[c(s_h; f)];$$

where

$$c(s; f) = \mathbb{1}\{\forall a \in \mathcal{A} : f(s, a) = f^{\star}(s, a)\}.$$

Then, we have

$$\sup_{f \in \mathcal{F}_h} \frac{1}{k}\sum_{i=1}^{k} c(s_h^{(i)}; f) - \sup_{f \in \mathcal{F}_h} \mathbb{E}_{;h}^{U}[c(s_h; f)] + \sup_{f \in \mathcal{F}_h} G_k(f)$$
$$1 - \frac{1}{4}\min(\varepsilon/32)^{-1} + \sup_{f \in \mathcal{F}_h} G_k(f);$$

where in the second inequality follows by applying (8). Furthermore, we have

$$\sup_{f \in \mathcal{F}_h} G_k(f) - \sup_{f \in \mathcal{F}_h} G_k(f) - \mathbb{E}[\sup_{f \in \mathcal{F}_h} G_k(f)] + \mathbb{E}[\sup_{f \in \mathcal{F}_h} G_k(f)];$$

where $\mathbb{E}[\sup_{f \in \mathcal{F}_h} G_k(f)]$ is the expected value of the random variable $\sup_{f \in \mathcal{F}_h} G_k(f)$ under the k iid draws from $P_{;h}^{U}$ (we will let this distribution be implicit in the rest of this sub-section of the proof). We will proceed by bounding these two terms one by one. Specifically, under any event where both terms are at most $\frac{1}{10}\min(\varepsilon/32)^{-1}$, we would have $\sup_{f \in \mathcal{F}_k} G_k(f) \leq \frac{1}{5}\min(\varepsilon/32)^{-1} < \frac{1}{4}\min(\varepsilon/32)^{-1}$. Therefore, we would have $\sup_{f \in \mathcal{F}_h} \frac{1}{k}\sum_{i=1}^{k} c(s_h^{(i)}; f) < 1$, which implies that for every $f \in \mathcal{F}_h$ we have $f(s_h^{(i)}; a) \neq f^{\star}(s_h^{(i)}; a)$ for some $i \in [k]$ and $a \in \mathcal{A}$, which is our required condition.

For the first of these terms, let $s_h^{(1)}; \ldots; s_h^{(k)}$ be an arbitrary sequence such that $s_h^{(i)} = s_h^{(i)}$ for all $i \neq j$, for some $j \in [k]$. Then, we can bound

$$\sup_{f \in \mathcal{F}_h} \frac{1}{k}\sum_{i=1}^{k} c(s_h^{(i)}; f) - \sup_{f \in \mathcal{F}_h} \frac{1}{k}\sum_{i=1}^{k} c(s_h^{(i)}; f)$$
$$\sup_{f \in \mathcal{F}_h} \frac{1}{k}\sum_{i=1}^{k} c(s_h^{(i)}; f) - c(s_h^{(i)}; f)$$
$$= \frac{1}{k}\sup_{f \in \mathcal{F}_h} c(s_h^{(j)}; f) - c(s_h^{(j)}; f)$$
$$\frac{1}{k}:$$

Therefore, we can apply the bounded differences inequality inequality to the first term, which gives us

$$\sup_{f \in \mathcal{F}_h} G_k(f) - \mathbb{E}[\sup_{f \in \mathcal{F}_h} G_k(f)] \quad \frac{1}{10}\min(\varepsilon/32)^{-1};$$

with probability at least $1 - \exp(-k\min(\varepsilon/32)^{2}/50)$. We can ensure that this event occurs with probability at least $1 - \delta/(2H)$, by noting

$$\exp(-k\min(\varepsilon/32)^{2}/50) \quad \delta/(2H)$$
$$\Leftrightarrow \quad k \quad 50\min(\varepsilon/32)^{2}\log(2H/\delta):$$

Next, for the second term above, we will follow a standard symmetrization argument. Let $\sigma_1; \ldots; \sigma_k$ be k iid Rademacher random variables (random variables such that $\text{Prob}(\sigma_i = 1) = \text{Prob}(\sigma_i = -1) = \frac{1}{2}$ for every $i \in [k]$). Also, let

$s_h^{(1)}, \ldots, s_h^{(k)}$ be a collection shadow variables that are distributed independent from and identically to $s_h^{(1)}, \ldots, s_h^{(k)}$. Then, we have

$$\begin{aligned}
\mathbb{E}[\sup_{f \in \mathcal{F}_h} G_k(f)] &= \mathbb{E}_{s_h^{(1:k)} \sim P^U_{\cdot,h}} \left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k c(s_h^{(i)}; f) - \mathbb{E}_{s_h \sim P^U_{\cdot,h}}[c(s_h; f)] \right] \\
&= \mathbb{E}_{s_h^{(1:k)} \sim P^U_{\cdot,h}} \left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k c(s_h^{(i)}; f) - \mathbb{E}_{s_h \sim P^U_{\cdot,h}}[c(s_h; f)] \right] \\
&\quad \mathbb{E}_{s_h^{(1:k)}, s_h^{(1:k)} \sim P^U_{\cdot,h}} \left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k c(s_h^{(i)}; f) - \frac{1}{k} \sum_{i=1}^k c(s_h; f) \right] \\
&= \mathbb{E}_{s_h^{(1:k)}, s_h^{(1:k)} \sim P^U_{\cdot,h}, \sigma_{1:k} \sim \mathrm{Unif}(\{-1,1\})} \left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k \sigma_i \left(c(s_h^{(i)}; f) - c(s_h; f) \right) \right] \\
&\quad 2\mathbb{E}_{s_h^{(1:k)} \sim P^U_{\cdot,h}, \sigma_{1:k} \sim \mathrm{Unif}(\{-1,1\})} \left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k \sigma_i c(s_h^{(i)}; f) \right];
\end{aligned}$$

where the first inequality follows from Jensen's, the subsequent equality follows by symmetry, and the final inequality follows since $\sup(a+b) \le \sup a + \sup b$, and since by symmetry on the definitions of $s_h^{(i)}$ and $s_h^{(i)}$. Next, suppressing the subscript in the expectation for brevity, we note that

$$\begin{aligned}
2\mathbb{E}\left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k \sigma_i c(s_h^{(i)}; f) \right] &= 2\mathbb{E}\left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k \sigma_i \mathbb{1}\{\forall a \in \mathcal{A} : f(s_h^{(i)}; a) = f^\star(s_h^{(i)}; a)\} \right] \\
&\quad 2\mathbb{E}\left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k \sigma_i \mathbb{1}\{f(s_h^{(i)}; a_h^{(i)}) = f^\star(s_h^{(i)}; a_h^{(i)})\} \right] \\
&= \mathbb{E}\left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k \sigma_i 2\mathbb{1}\{f(s_h^{(i)}; a_h^{(i)}) = f^\star(s_h^{(i)}; a_h^{(i)})\} - 1 \right] \\
&= \mathbb{E}\left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k \sigma_i f(s_h^{(i)}; a_h^{(i)}) \right];
\end{aligned}$$

where the inequality step follows because

$$\mathbb{1}\{\forall a \in \mathcal{A} : f(s_h^{(i)}; a)) = f^\star(s_h^{(i)}; a)\} \le \mathbb{E}_{a \sim P}\left[\mathbb{1}\{f(s_h^{(i)}; a)) = f^\star(s_h^{(i)}; a)\} \right] \quad \forall P;$$

and also by applying Jensen's inequality, the second equality follows because $\mathbb{E}[\sum_{i=1}^k \sigma_i] = 0$, and the final equality follows because $2\mathbb{1}\{f(s;a)) = f^\star(s;a)\} - 1 = f(s;a) f^\star(s;a)$, and by symmetry $\sigma_i f^\star(s_h^{(i)}; a_h^{(i)})$ and σ_i are identically distributed for each $i \in [k]$.

Next, note that the final bound is the Rademacher complexity of $\mathcal{F}_h$ (under some distribution of state, action pairs). Since by assumption $\mathcal{F}$ has VC dimension at most d_{VC}, so does $\mathcal{F}_h$, so by Rebeschini (2020, Theorem 5.6) we have

$$\mathbb{E}\left[\sup_{f \in \mathcal{F}_h} \frac{1}{k} \sum_{i=1}^k \sigma_i f(s_h^{(i)}; a)) \right] \le C \sqrt{\frac{d_{VC}}{k}};$$

for some universal constant C, regardless of the distribution over state, action pairs. Therefore, we have

$$\mathbb{E}\left[\sup_{f \in \mathcal{F}_h} G_k(f) \right] \le C \sqrt{\frac{d_{VC}}{k}};$$

so we can ensure that $\mathbb{E}[\sup_{f \in \mathcal{F}_h} G_k(f)] \le \frac{1}{10} \min(\varepsilon/32)^{-1}$ as long as

$$k \ge 100 C^2 \min(\varepsilon/32)^2 d_{VC}:$$

Therefore, putting the above together, as long as

$$k \geq \max\left\{ 50\,\epsilon_{\min}(\epsilon,\delta/32)^2 \log(2H/\delta),\; 100C^2\,\epsilon_{\min}(\epsilon,\delta/32)^2 d_{VC},\right.$$
$$\left. 50\,\epsilon_{\min}(\epsilon,\delta/32)^2 (2C^2 d_{VC} + \log(2H/\delta)) \right\},$$

then both bounds hold with probability at least $1 - \delta/(2H)$, and therefore we have our required event that every $h \in F_h$ disagrees with $h^\star$ on at least one of the sampled states $s_h^{(i)}$ with this probability.

Finding sufficiently large m

Next, for our target value of k samples from P, the the remaining question is how large does m need to be to ensure we have at least k samples of s_h where $s_h \in RD(D)$, with probability at least $1 - \delta/(2H)$? We know that for samples s_h, this event occurs with probability at least $\delta/8$ by the construction of H.

Now, suppose we set $m = k(\epsilon/8)^{-1} + t$ for some $t \geq 0$, and let $X_1, \ldots, X_m$ be a set of iid $\{0,1\}$-valued random variable such that $\text{Prob}(X_i = 1) = \epsilon/8$ for each $i \in [m]$. Also, let $X = \sum_{i=1}^m X_i$. Applying Bernstein's inequality, we have

$$\text{Prob}(X \leq k) = \text{Prob}(X \leq E[X] - t\,\epsilon/8)$$
$$\leq \exp\left(\frac{-\frac{1}{2}t^2(\epsilon/8)^2}{m(\epsilon/8)(1-\epsilon/8) + \frac{1}{3}t\,\epsilon/8} \right).$$

Now, in order to ensure that this probability is smaller than $\delta/(2H)$, it is sufficient to solve

$$\exp\left(\frac{-\frac{1}{2}t^2(\epsilon/8)^2}{m(\epsilon/8)(1-\epsilon/8) + \frac{1}{3}t\,\epsilon/8} \right) = \delta/(2H)$$
$$\Leftrightarrow \quad \frac{1}{2}(\epsilon/8)^2 t^2 - (4/3 - \epsilon/8)(\epsilon/8)\log(2H/\delta)\,t - (1-\epsilon/8)\log(2H/\delta)k = 0.$$

Now, letting t^+ be the greater solution of this quadratic equation,

$$t^\star = \frac{(4/3 - \epsilon/8)(\epsilon/8)\log(2H/\delta)}{(\epsilon/8)^2}$$
$$+ \sqrt{\frac{(4/3 - \epsilon/8)(\epsilon/8)\log(2H/\delta)^2 + 2(\epsilon/8)^2(1-\epsilon/8)\log(2H/\delta)k}{(\epsilon/8)^2}}$$
$$\leq 2(4/3 - \epsilon/8)(\epsilon/8)^{-1}\log(2H/\delta) + (\epsilon/8)^{-1}\sqrt{2(1-\epsilon/8)\log(2H/\delta)k}$$
$$\leq 3(\epsilon/8)^{-1}\log(2H/\delta)\sqrt{k},$$

where in the first inequality we apply apply the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b > 0$, and in the second inequality we note that $4/3 - \epsilon/8 \leq 1/3$ for any $\epsilon \in (0,1)$, so $(4/3 - \epsilon/8) \leq 1$, and $2(1 - \epsilon/8) \leq 1$.

Finally, we note that since the probability that $s_h \in RD(D)$ is at least $\epsilon/8$, the probability that we have fewer than k successes in m samples is no greater than $\text{Prob}(X \leq k)$, and therefore from the above bounds any choice of

$$m \geq (\epsilon/8)^{-1}\left[k + 3\log(2H/\delta)\sqrt{k} \right]$$

is sufficient to ensure that this probability is at most $\delta/(2H)$.

Putting Everything Together

Given the analysis, if we set

$$k \geq 50\,\epsilon_{\min}(\epsilon,\delta/32)^2 \left[2C^2 d_{VC} + \log(2H/\delta) \right]$$
$$\text{and} \quad m \geq 8\left[k + 3\log(2H/\delta)\sqrt{k} \right],$$

then our required result is ensured with probability at least $1-\delta$.

We note that this setting of m satisfies the growth bound

$$m = \frac{1}{\min(\epsilon/32)^2 d_{VC}} \log(1/\delta) \log(H) \Bigg);$$

which is our required result.

□

E DERIVATIONS OF POLICY COVER DIMENSION BOUNDS

Here we provide the derivations of our policy cover dimension bounds given in Section 4.1.

Tabular MDP Let some policy class Π^0 be given, and assume the MDP has S total states, and denote these by $S_1, \dots, S_S$. Then, for each $i \in [d]$, we define the policy $\pi_i \in \Pi^0$ according to

$$\pi_i = \operatorname*{argmin}_{\pi \in \Pi^0} \mathbb{E}^{\pi}\left[\sum_{h=1}^{H} \mathbb{1}\{s_h = S_i\}\right].$$

That is, π_i is the optimal policy in Π^0 for the reward function given by reaching the ith state. Now, let some arbitrary subset $\mathcal{S} \subseteq S$, and arbitrary $\pi \in \Pi^0$ be given. Then, we have

$$\begin{aligned}
\frac{1}{H}\sum_{h=1}^{H} P^{\pi}(s_h \in \mathcal{S}) &\le \frac{1}{H}\sum_{h=1}^{H}\sum_{s \in \mathcal{S}} P^{\pi}(s_h = s) \\
&\le \frac{1}{H}\sum_{h=1}^{H}\sum_{i=1}^{S} P^{\pi}(s_h = S_i) \\
&= \sum_{i=1}^{S} \mathbb{E}^{\pi}\left[\frac{1}{H}\sum_{h=1}^{H} \mathbb{1}\{s_h = S_i\}\right] \\
&\le \sum_{i=1}^{S} \mathbb{E}^{\pi_i}\left[\frac{1}{H}\sum_{h=1}^{H} \mathbb{1}\{s_h = S_i\}\right].
\end{aligned}$$

Since the above holds for arbitrary $\mathcal{S}$ and π, we have that $\{\pi_1, \dots, \pi_S\}$ is a policy cover for Π^0. That is, we always have a policy cover of size at most S, so $d \le S$.

Block MDP and Non-negative Rank MDP The reasoning here in both cases is similar to the tabular MDP case. In both cases, we can model the MDP as transitioning from (s,a) to an intermediate discrete latent state z following some distribution $P(z \mid s,a)$, and then sampling the next state s' following another distribution $P^0(s' \mid z)$. Let d be the number of discrete latent states, which corresponds to S for Block MDP and d_{NNR} for Non-negative Rank MDP in terms of our notation in Section 4.1. Also, for each $h \in [h]$, let z_h denote the discrete latent state that generated s_h at time h, and let $Z_1, \dots, Z_d$ denote the possible values of z. Then, following analogous reasoning as above, define

$$\pi_i = \operatorname*{argmin}_{\pi \in \Pi^0} \mathbb{E}^{\pi}\left[\sum_{h=1}^{H} \mathbb{1}\{z_h = Z_i\}\right];$$

for some given policy class Π^0. Also, let some arbitrary measurable set $S \subseteq S$ be given, along with some arbitrary policy $\in \Pi^0$. Then, we have

$$\begin{aligned}
\frac{1}{H}\sum_{h=1}^{H} P(s_h \in S) &= \frac{1}{H}\sum_{h=1}^{H}\sum_{i=1}^{d} P(z_h = Z_i) P^0(S \mid z = Z_i) \\
&= \sum_{i=1}^{d} P^0(S \mid z = Z_i) E\left[\frac{1}{H}\sum_{h=1}^{H} 1\{z_h = Z_i\}\right] \\
&\le \sum_{i=1}^{d} P^0(S \mid z = Z_i) E_{\pi_i}\left[\frac{1}{H}\sum_{h=1}^{H} 1\{z_h = Z_i\}\right] \\
&= \frac{1}{H}\sum_{h=1}^{H}\sum_{i=1}^{d} P_{\pi_i}(z_h = Z_i) P^0(S \mid z = Z_i) \\
&\le \sum_{i=1}^{d}\frac{1}{H}\sum_{h=1}^{H}\sum_{j=1}^{d} P_{\pi_i}(z_h = Z_j) P^0(S \mid z = Z_j) \\
&= \sum_{i=1}^{d}\frac{1}{H}\sum_{h=1}^{H} P_{\pi_i}(s_h \in S),
\end{aligned}$$

where the the above bounds we apply the fact that the distribution P^0 for generating s given z does not depend on either the policy or h, and the second inequality follows because we are just introducing additional non-negative summands. That is, we always have a policy cover of size d, so the policy cover dimension is at most d. So, for Block MDP we have $d \le |S|$, and for Low Non-negative Rank MDP we have $d \le d_{\text{NNR}}$.

F TABULAR MDP SAMPLE COMPLEXITY BOUND

The actual bound available in Azar et al. (2017) is a regret bound. Given N total episodes, as long as N is sufficiently large (specifically, if $N \ge S^3A$), and $H \ge SA$ (which is reasonable in non-trivial settings) they bound the total regret by

$$\text{Regret}(N) \le O\left(\sqrt{H^3 SAN \log(1/\delta)^2}\right),$$

with probability at least $1 - \delta$. Now, suppose we run UCB-VI over N total episodes, for large N, and let $\hat{\pi}$ denote the average policy over all these episodes. Then, under the same high-probability event, the suboptimality of this average policy must be bounded by $\text{Regret}(N)/N$. That is,

$$\text{SubOpt}(\hat{\pi};) \le O\left(\log(1/\delta)^2 \sqrt{\frac{H^3 SA}{N}}\right).$$

Now, suppose we wish the sub-optimality to be bounded by some given $\in (0, 1)$. Then solving for N, this corresponds to

$$\begin{aligned}
& O\left(\log(1/\delta)^2 \sqrt{\frac{H^3 SA}{N}}\right) \le \\
\Leftrightarrow \quad & N \ge O\left(H^3 SA \, ^{-2} \log(1/\delta)^4\right),
\end{aligned}$$

which is our presented sample complexity.

G EXPERIMENTAL DETAILS AND ADDITIONAL EXPERIMENTS

We show the promise of SABRE on a rich observation MDP setting called Block MDP Du et al. (2019); Misra et al. (2020). In this setting, the agent receives observations generated from a latent state. Further, no two states can generate the same observation. This is an expressive MDP setting which can capture non-trivial practical problems. Our goal is to discuss the implementation details for SABRE and show that it can achieve optimal return, while not taking unsafe actions, and minimizing the calls to safety oracle. Note that all details of our environment and methodology will also be available in our code release, which can be used to fully reproduce all of our results, which is currently withheld to preserve anonymity

Environment. For a given horizon H, an instance of the environment has $4H + 1$ states and $K = 4$ actions. One of the action is a known safe action a_{safe}. These states are labeled as $S = \{s_{1;0}\} \cup \{s_{i;h}\}_{i \in [4]; h \in [H]}$. For a state $s_{i;h}$, the index i denotes its type and h denotes the level. We view states with index $i \in \{1; 2\}$ as normal states, while states index $i = 3$ are considered safe states and those with $i = 4$ are unsafe states. The agent never observes the state but instead receives an observation $x \sim q(\cdot \mid x)$ generated by an emission process q. The rich-observation setting implies that no two different states can generate the same observation.

The agent starts deterministically in $s_{1;0}$. All latent state transitions occur deterministically. After taking h actions, the agent is in one of these four states: $\{s_{j;h}\}_{j \in [4]}$. Each environment instance has two action sequences $(a_1, \ldots, a_H)$ and $(a_1^0, \ldots, a_H^0)$ with $a_h \neq a_h^0; a_h \neq a_{\text{safe}}$ and $a_h^0 \neq a_{\text{safe}}$, for all $h \in [H]$. We view the a_h^0 as the *unsafe action* and a_h as the *continue action* for time step h.

For any $i \in \{1; 2\}$ and $h \in [H]$, transitions in the normal state $s_{i;h}$ operate as follows: taking the continue action a_h leads to the next normal state $s_{i;h+1}$ of same type, taking a_{safe} leads to the safe state $s_{3;h+1}$, taking the unsafe action a_h^0 leads to the unsafe state $s_{4;h+1}$, while the remaining action leads to the normal state normal state $s_{3-i;h}$ of the other type. All actions in the safe state $s_{3;h}$ leads to the next safe state $s_{3;h+1}$. Finally, in the unsafe state $s_{4;h}$, taking any action except a_{safe} leads to the next unsafe state $s_{4;h+1}$, while taking a_{safe} leads to the safe state $s_{3;h+1}$.

For a given transition $(s; a; s')$, the reward function $R(s; a; s')$ depends only on s'. For $s' = s_{i;h}$, the reward r is defined as follows: if $i = 1$ and $h < H$, then $r = 1=H$; for $i = 1$ and $h = H + 1$, $r = 2:0$, if $i = 2$ then $r = 1$, if $i = 3$ then $r = 0$ and if $i = 4$ then -1. The optimal return of 2.8 is achieved by staying on states with index 1.

An observation is stochastically generated from a state $s_{i;h}$ each time the agent visits the state. This is done by first concatenating two 1-sparse vectors encoding i and h, and adding independent Gaussian noise to each dimension sampled from a 0 mean and 0.1 standard deviation. Let the resultant vector be z' and it has $H + 5$ dimensions (4 dimensions to to encode i and $H + 1$ dimensions to encode h). We concatenate z' with a 0 vector of size $2^{\lceil \log_2 dH+5 \rceil} - (H + 5)$. Let z be the final vector and its dimensionality is given by $k = 2^{\lceil \log_2 dH+5 \rceil}$. The observation is generated by multiplying z with a Hadamard matrix of size $k \times k$ in order to mix its dimensions. We use Sylvester's construction to create a Hadamard matrix H_k of size $k \times k$. This is an inductive approach that works for $k = 2^l$ for some $l \in \mathbb{N}$. It defines $H_1 = [1]$ and $H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$ for every $n \in \mathbb{N}$. We borrowed the observation process from Misra et al. (2020).

In addition to the observation, the agent receives $(H + 1)B$-dimensional safety features $\{\phi(s; a)\}_{a \in A}$ for every action upon visiting a state. We assume that the safety function $f^?$ is given by $f^?(s; a) = w^T \phi(s; a) + b$ for some unknown $(w; b)$. In practice, these safety features can include readings from various safety devices attached to an agent (such as a LIDAR sensor, or temperature readings) which can be different from observation (e.g., camera image) which is used for dynamics. The agent receives a constant safe and unsafe feature for all states of type 3 and 4. States of type 2, have variance in safety features along all dimensions. Vising these states can help us learn safety function faster, but this comes at the cost of a negative reward associated with visiting these states. Finally, a state $s_{1;h}$ only have variance in safety features along the $\{(hB + 1), \ldots, hB + 1\}$ dimensions. Visiting $s_{1;h}$ gives higher reward but reveals the safety features more slowly. Therefore, an optimal agent must balance between optimizing reward and minimizing safety oracle calls by exploring safety features quickly.

SABRE Implementation. We use Proximal Policy Optimization (Schulman et al., 2017) (PPO) as our blackbox RL algorithm. PPO is a popular empirical RL method and while not a PAC RL algorithm, is quite effective in practice for problems that do not require strategic exploration. We define the policy class and the value network in PPO using a two-layer feed forward neural network with Leaky ReLU non-linearity.

We implement queries to region of disagreement by reducing it to linear programming. Formally, given a safety labeled dataset $D = \{(\phi(s_i; a_i); y_i)\}_{i=1}^n$, we solve for whether $\phi(s; a)$ is in $\text{RD}^a(D)$, by first returning False if $a = a_{\text{safe}}$, and otherwise, solving the following two linear programs to compute the value of $c_{\max}$ and $c_{\min}$:

$$
\begin{aligned}
c_{\max} = \max_{w;b} \; & w^T \phi(s; a) + b; \quad \text{such that;} \\
& \forall i \in [d]; \quad y_i(w^T \phi(s_i; a_i) + b_i) \geq 0; \\
& \|w\|_1 \leq 1; |b| \leq 1:
\end{aligned}
$$

and

$$c_{\min} = \min_{w;b} w^T \quad (s;a) + b; \quad \text{such that;}$$
$$8i \; 2 \; [d]; \quad y_i(w^T \quad (s_i;a_i) + b_i) \quad 0;$$
$$kwk_1 \quad 1; jbj \quad 1:$$

The given feature $(s;a)$ is in RD(D) if and only if either $c_{\max} \quad 0$ and $c_{\min} < 0$, or $c_{\max} > 0$ and $c_{\min} \quad 0$.

We implement (D) by mapping each policy 2 to a *known safe policy* $_D$ and defining $(D) = f \; _D \; j \quad 2 \quad g$. Let $(s;a) \; 2 \; \texttt{SurelySafe}(D)$ denote set membership in the set of surely safe state-action pairs. We define $_D$ given as

$$_D(a \; j \; s) = \frac{1 \; f(s;a) \; 2 \; \texttt{SurelySafe}(D)g \quad (a \; j \; s)}{\sum_{a^0 2 A} 1 \; f(s;a^0) \; 2 \; \texttt{SurelySafe}(D)g \quad (a^0 \; j \; s)}:$$

When the agent visits a state s, we allow the agent to receive the safety features for all actions $f \quad (s;a^0)g_{a^0 2 A}$. We test for set membership $(s;a) \; 2 \; \texttt{SurelySafe}(D)$ by first returning True if $a = a_{\text{safe}}$, and otherwise, solving the linear programs described above using the feature $(s;a)$ and returning True if and only if $c_{\max} \quad 0$ and $c_{\min} \quad 0$.

Our implementation of SABRE closely follows Algorithm 1. One notable departure for efficiency is that we update the region of disagreement incrementally, rather than in batch.

Hyperparameters. We list hyperparameters that we use for SABRE for Table 2 and for the PPO baseline in Table 3. Note that we do *not* provide explicit values for the hyperparameters $_{\text{explore}}$, $_R$, $_{\text{explore}}$, and $_R$ in SABRE. Instead, in practice these are implicitly determined by the other hyperparameters relating to the PPO blackbox algorithm. For example, the greater the total number of iterations of PPO is, the stronger guarantees we can make for the algorithm with respect to and ; *i.e.* the greater the number of iterations, the smaller the implicit and are. The hyperparameters N, B, and m in Table 2 correspond to those in Algorithm 1. The remaining hyperparameters in Table 2 are for the PPO blackbox algorithm, which are the same as the hyperparameters used by the unsafe PPO baseline that are listed in Table 3. Note that we use the same hyperparameter values for the PPO blackbox algorithm both when it is called on line 6 for safety learning, or line 9 for return optimization, in Algorithm 1. The hyperparameters used by PPO are the following:

- *Iterations of PPO* denotes the number of episodes used by each call to the PPO. Note that each time we call PPO in Algorithm 1 we start with a randomly initialized policy.
- *Gradient Clip* Is a maximum absolute value of all gradients for PPO updates; any gradients with absolute values above this threshold are clipped, which helps regularize and avoid extreme updates.
- *Number of PPO Updates* refers to the number of iterations of gradient descent optimization that we perform for each sampled batch. We use Adam optimization for training the policy parameters.
- *Ratio clipping coefficient* is a parameter used by PPO to control probability ratios, and limit how much the policy can change in each update.
- *Entropy coefficient* is the coefficient of entropy regularization for PPO, which encourages greater exploration and helps avoid convergence to a sub-optimal deterministic policy.

G.1 Additional Experiments

We also compare SABRE with a baseline SABRE-RO, which is based on the naive baseline algorithm described in Section 3.1. Like SABRE, SABRE-RO only ever takes actions that are known to be safe. In addition, to make the comparison fair, it labels data offline in batches at the same frequency as SABRE. However, instead explicitly exploring safety to directly learn $f^?$, it greedily tries to optimize return following the PPO algorithm (under the safety constraint). Specifically, each time it labels data, it sequentially labels $(s;a)$ for all previously observed s and $a \; 2 \; A$ where $s \; 2 \; \text{RD}^a(D)$, updating D after each call to the labeling oracle. We compare each method for various values of N and m.

Hyperparameter	Value
N	5
B	1
m	100
Iterations of PPO	1000
Learning Rate	0.001
Batch Size	32
Gradient Clip	20
Number of PPO Updates	10
Ratio Clipping Coefficient	0.1
Entropy Coefficient	0.01

Table 2: Hyperparameters for SABRE

Hyperparameter	Value
Iterations of PPO	6500
Learning Rate	0.001
Batch Size	32
Gradient Clip	20
Number of PPO Updates	10
Ratio Clipping Coefficient	0.1
Entropy Coefficient	0.01

Table 3: Hyperparameters for PPO

Results . We present mean return and cumulative oracle calls in Figure 3. We observed that SABRE-RO optimizes the environment reward which doesn't lead to the negative reward that SABRE accumulates early in the training. Further, if we consider the goal of achieving at least a return of 1.9 which is 1=2 the optimal return of 2.8, then SABRE achieves this for $N = 2$ and $m = 100$, with least number of oracle calls of around 70. This is possible since SABRE can quickly learn the safety function by reaching the safe but low reward path, and then using it to optimize the environment reward. However, overall SABRE-RO is a strong baseline and almost as competitive as the SABRE. We leave a more detailed study and questions of lower bound for *passive safety* learning for future work.