Optimistic Risk-Aware Model-based Reinforcement Learning

Romina Abachi *University of Toronto, Vector Institute* romina@cs.toronto.edu

Amir-massoud Farahmand *University of Toronto, Vector Institute* farahmand@vectorinstitute.ai

Abstract

We study the problem of planning with a learned model in model-based RL. In this setting, the agent should act safely with respect to its belief about the environment, as it may have errors in that belief. On the other hand, it needs to explore in order to learn good policies. We propose two algorithms based on the notion of risk-sensitivity over the uncertainty over parameters of the true MDP using a Bayesian RL framework. In the first, we propose optimizing the risk over the beliefs directly using a policy sub-gradient. In the second, we propose using the risk as a constraint to an optimistic or exploratory planner. We empirically evaluate our proposed algorithms and see improvement in certain cases over non-risk-aware and only-optimistic methods.

Keywords: Model-based Reinforcement Learning, Risk, Model-based, Exploration-Exploitation

1. Introduction

Model-based reinforcement learning (MBRL) consists of learning a model of the environment, planning using the learned model, and using the obtained policy to further interact with the environment, collect more data, and further improve the model and policy (Sutton, 1990). Variations of this procedure have gained considerable interest in recent years (Schrittwieser et al., 2020; Hafner et al., 2019; Ha and Schmidhuber, 2018).

A learned model, however, is often imprecise – it has some errors compared to the true dynamics of the environment. The sources of errors are the estimation error (i.e., the statistical error caused by having a finite number of samples) and possibly model approximation error (i.e., the true dynamics cannot be represented by any member of the model class due to limited capacity), see e.g., Györfi et al. (2002); Steinwart and Christmann (2008) for a discussion of estimation and function approximation errors in the supervised learning setting, Munos and Szepesvári (2008); Antos et al. (2008); Farahmand et al. (2016); Tosatto et al. (2017); Chen and Jiang (2019) in model-free RL, and Farahmand et al. (2017) in MBRL. Many MBRL algorithms ignore this error, and plan as if the learned model is the true model of the environment. This approach is called the *certainty equivalence principle* in control theory (Aström and Wittenmark, 1994). The underlying assumption of the certainty equivalence principle is that if and when the model becomes accurate enough, the difference between the policy that is optimal according to the model versus the true optimal policy of the environment becomes negligible. If there is no model approximation error, this might be asymptotically true. But oftentimes we cannot guarantee that there is no model approximation error when we model a complex environment. Moreover, in the finite sample regime, the estimation error might still be substantial. As a consequence, following this recipe may lead to a policy whose behaviour is disastrous in the real environment. The planning algorithm may exploit the errors in the model to improve its performance on the model, but these errors could lead to bad and unsafe outcomes in the true environment.

To mitigate this issue, we can either improve the model learning or use planners that explicitly incorporate the error in the model. To improve the model learning, we can use models that are more expressive (e.g., larger deep neural networks) or use decision-aware model learning approaches that focus on learning the model where it matters for the decision making while ignoring irrelevant aspects of the environment (Farahmand et al., 2017; D'Oro et al., 2020; Schrittwieser et al., 2020; Grimm et al., 2020; Lambert et al., 2020; Eysenbach et al., 2021). Although this is a worthy approach, in this work we instead turn our attention to the other part of the problem: planning with an imperfect model.

To deal with an imperfect model, we propose and study an online Bayesian risk-sensitive method. The method maintains a posterior over possible models of the environment, as is common in Bayesian RL, and the planner finds a policy that is risk-sensitive with respect to the uncertainty over the true model. Specifically, we use Conditional Value-at-Risk (CVaR) as the risk measure (Acerbi and Tasche, 2002; Shapiro et al., 2009). As opposed to the performance

measured according to the expected value function according to the posterior over the models, which is common in the Bayesian RL framework, CVaR measures the average performance of only those models whose value function belongs to a lower quantile of performance according to the posterior. This allows control over how risk-averse the policy is. Our first algorithmic contribution is a policy gradient-based algorithm for risk-sensitive planning (Section 3). This risk-aware Bayesian RL viewpoint has been studied in some previous work (cf. Angelotti et al. (2021) and Chapter 6 of Ghavamzadeh et al. 2015), but the specific PG-based method has not been proposed before (among some other differences). This approach has also some high-level similarities with the Robust Dynamic Programming (Iyengar, 2005; Nilim and El Ghaoui, 2005) literature, which focuses on robustifying against the worst-case statistically feasible model. The usual notion of robustness, however, can be too conservative as it considers the worst-case model. CVaR, on the other hand, provides a gentler notion of risk that considers a whole spectrum between the expected performance (risk neutral) to almost worst-case. We also note that our usage of risk is different from what is common in risk-sensitive RL, which focuses on uncertainty of return within a fixed MDP, see e.g., Chow et al. (2017) and references therein. In our work, the uncertainty is over the true model of the environment.

To allow the MBRL agent to explore the environment while avoiding risky behaviours, we introduce Optimistic Risk-Constrained planner in Section 4 as the other algorithmic contribution of this work. It modifies the optimistic planners and its stochastic variants such as PSRL (Osband and Van Roy, 2017), which are commonly used for balancing exploration-exploitation, by explicitly considering risk over uncertainty of the model within optimistic planning. We empirically evaluate a few variants and compare them with purely optimistic and purely risk-averse methods. We find that a CVaR constraint can improve performance in certain domains while having little effect in others.

2. Preliminaries

2.1 Markov Decision Processes

We consider the problem of learning to optimize a discounted infinite-horizon MDP $M = (\mathcal{X}, \mathcal{A}, \gamma, P, R, \mu_0)$ with state space $\mathcal{X}$, action space $\mathcal{A}$, discount factor γ, and starting state distribution μ_0 (Bertsekas and Tsitsiklis, 1996; Szepesvári, 2010; Sutton and Barto, 2018). The transition kernel $P : \mathcal{X} \times \mathcal{A} \to \mathcal{P}(\mathcal{X})$ and reward function $R : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathcal{R}$ are unknown, where $\mathcal{P}(\cdot)$ refers to the space of probability distributions defined over a set $\cdot$. The RL agent interacts with the MDP starting at time $t = 0$, takes action A_t at each timestep according to a policy π, transitions to state X_{t+1}, and observes a scalar reward $R_t = R(X_t, A_t, X_{t+1})$. We want to find a stationary Markov policy $\pi_M : \mathcal{X} \to \mathcal{P}(\mathcal{A})$ that maximizes policy's performance J_{μ_0}, i.e., $\pi_M = \arg\max_\pi J_{\mu_0}(\pi, P) = \sum_{x \in \mathcal{X}} \mu_0(x) V_P^\pi(x) = \mathbb{E}_{P, \pi, \mu_0}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t, X_{t+1})\right]$, where $V_P^\pi(x)$ is defined as the value function of the MDP with transition kernel P at state x. In the rest of the paper we omit μ_0 in our notation for J_{μ_0}. If there is no restriction over the policy, the optimal policy does not depend on the initial distribution μ_0. On the other hand, if the policy is restricted to a subset of all possible policies, the optimal policy within that class may depend on μ_0. If the policy is parameterized by θ, we simply use the parameters of the policy to refer to the policy, i.e., the policy performance and value functions would be $J(\theta, P)$ and $V_P^\theta(\cdot)$, respectively.

Suppose we have collected all the interactions of the environment in a dataset $\mathcal{D}$ consisting of tuples (X_t, A_t, R_t, X_{t+1}). We consider the Bayesian setup where given a prior distribution over unknown MDPs (transition kernel and reward function), we infer the posterior distribution $\Pr(M|\mathcal{D})$. The posterior represents our belief about the true MDP given the collected data so far. As more data is collected, the posterior concentrates more and more around P and R of the true MDP, assuming that prior probability is non-zero, i.e., $\Pr(M) > 0$. In much of this work we omit the dependence of the posterior and policy performance on the unknown function R only for notational convenience. However, we assume a posterior over reward functions is also inferred.

2.2 Risk Measures

A risk measure maps a random variable to a real-valued number. It is a measure of how risky a random variable behaves. Various risk measures have been proposed in the literature, see e.g., Shapiro et al. (2009). In this work, we focus on the Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR).

Let Z be a scalar random variable defined on a probability space $(\Omega, \mathcal{F}, P)$ with a finite mean, i.e., $\mathbb{E}[|Z|] < \infty$. Denote its cumulative distribution function (CDF) by $F_Z(z) = P(Z \leq z)$. The Value-at-Risk (VaR) with confidence level $\alpha \in (0,1)$ is defined as: $\text{VaR}_\alpha(Z) \doteq \max\{z \mid F_Z(z) \leq \alpha\}$. This is the greatest point $z \in \mathbb{R}$ at which the probability of random variable Z taking values smaller than z is α. This quantity is sometimes also referred to as the (lower) α-quantile. Note that as F_Z is non-decreasing and right-continuous, z attains the maximum.

To define the Conditional Value-at-Risk, we define an auxiliary functional first:

$$F_\alpha(Z; \xi) \doteq \xi + \frac{1}{1-\alpha}\mathbb{E}[(Z - \xi)^+] = \xi + \frac{1}{1-\alpha}\int (z - \xi)^+ P(dz), \tag{1}$$

where $(t)^+ = \min\{0, t\}$ (we define $(t)^+ = \max\{0, t\}$). The minimizer of this functional is CVaR of the random variable at level α, as shown by Rockafellar et al. (2000):

$$\text{CVaR}_\alpha(Z) = \min_{\xi \in \mathbb{R}} F_\alpha(Z; \xi). \tag{2}$$

The minimum is attained on a non-empty closed bounded interval, with VaR being on the left endpoint of that interval (Theorem 1 of Rockafellar et al. (2000) or Section 6.2.4 of Shapiro et al. 2009). Therefore, we have $\text{CVaR}_\alpha(Z) = F_\alpha(Z; \text{VaR}_\alpha(Z))$. When the CDF $F_Z(z)$ is continuous at $\text{VaR}_\alpha(Z)$, which means that there is no probability atom there, CVaR takes a simpler form of

$$\text{CVaR}_\alpha(Z) = \mathbb{E}[Z \mid Z \geq \text{VaR}_\alpha(Z)], \tag{3}$$

see Acerbi and Tasche (2002) or Theorem 6.2 of Shapiro et al. (2009). Thus, $\text{CVaR}_\alpha(Z)$ can be interpreted as the expected value of Z up to $\text{VaR}_\alpha(Z)$ on the lower α-quantile of the distribution. Note that $\text{CVaR}_1(Z) = \mathbb{E}[Z]$, and $\text{CVaR}_\alpha(Z)$ tends to $\operatorname{ess\,inf}(Z)$ as $\alpha \to 0$. CVaR is popular in financial applications in comparison to the VaR as it is useful for controlling rare, but potentially catastrophic events, that occur below the α-quantile and do not impact the VaR at all (Rockafellar et al., 2006). As opposed to VaR, CVaR is a coherent risk measure which gives it additional desirable properties (Artzner et al., 1999). As a result, we focus on optimizing CVaR in this work.

3. Risk-sensitive Planning for MBRL

We describe the risk-aware planning for model-based RL and propose our policy gradient-based algorithm for solving it. The risk in this context refers to parametric (epistemic) uncertainty: the uncertainty due to the unknown parameters of the MDP. This is in contrast to the inherent uncertainty present when interacting with a known MDP, that is, the uncertainty due to the return distribution, induced by the stochasticity of P and the policy (aleatoric uncertainty). This framework is sometimes referred to as Risk-aware Bayesian RL (Ghavamzadeh et al., 2015, Section 6).

Figure 1 illustrates why it is important to consider a risk-sensitive objective. Suppose that the agent has already collected some data D through interaction with the true MDP with the dynamics P. In a Bayesian framework, the agent's belief about the probability of each model in the space of all models is encoded through its posterior $(P|D)$. This uncertainty over the model induces an uncertainty on the performance of a policy. For any $P \sim (P|D)$, the agent evaluates the performance of π_1 as $J(\pi_1; P)$. This defines a distribution $(J(\pi_1)|D)$ over the performance of this policy, as indicated by a green curve in the figure.

Now suppose that the agent wants to select one of the two policies π_1 and π_2. Which one should it choose? One approach is to compare the average performance w.r.t. the posterior. This would be $\mathbb{E}_{P\sim(P|D)}[J(\pi_1; P)]$ and $\mathbb{E}_{P\sim(P|D)}[J(\pi_2; P)]$. In this figure, π_2 would be the preferred policy according to this measure. Looking at the distribution, however, we see that $(J(\pi_2)|D)$ has a fatter tail in the lower values of performance. This means that according to our posterior belief, it is more probable that policy π_2's performance would be significantly worse than π_1's, even though its average performance is not. If we define "safe" as the policy with a lower chance of low performances across the model space (and "risky" as vice versa), policy π_1 is safer than π_2 according to our posterior belief. This discussion leads us to the definition of a risk-sensitive objective and policy.

Definition 1 Given a posterior distribution $(P|D)$ over the model of the MDP, starting state distribution μ_0, and risk measure ρ, the risk-sensitive objective is defined as

$$\rho_{P\sim(P|D)}[J_{\mu_0}(\pi; P)]. \tag{4}$$

Figure 1: The distribution of policy performance for two policies π_1 and π_2 induced by the the posterior over models given D_n. Policy π_2 has higher expected performance, but higher probability of low-performance over models, i.e., it is less safe. On the right side of the figure we also illustrate which parts of the posterior are considered for the optimistic algorithms defined in Section 4.

The risk-sensitive policy over the policy space is defined as

$$\arg\max_{\pi \in \Pi} \rho[J_0(\pi; P)]. \tag{5}$$

Usually, ρ is simply taken as the expectation (Zintgraf et al., 2019), which is a risk-neutral measure. In this work we are interested in risk measures that lead to risk-averse behaviour. In particular, we choose CVaR, as introduced in Section 2.2, which is a versatile and commonly used risk measure. In our setting, the random variable Z mentioned in (2)-(3) becomes $Z = J(\pi; P)$, the policy performance under policy π and transition kernel $P \sim p(\cdot|D)$.

Our proposed risk-sensitive planning for MBRL algorithm, which is specified in detail in Algorithm 1, at the high-level works as follows: The agent interacts with the environment, and maintain a posterior over the model and the reward function. It then computes the (sub-) gradient of $\text{CVaR}_\alpha(J(\pi))$ w.r.t. π in order to update the policy. This procedure repeats. We explain the details in the rest of this section.

Computing $\text{CVaR}(J(\pi))$ exactly, which involves computing the integral (1), is not feasible. Instead, we use an empirical estimate of (1) using m sampled models $P_i \sim p(\cdot|D)$ $(i = 1, \dots, m)$:

$$\hat{F}(\nu, \pi) = \nu + \frac{1}{m} \sum_{i=1}^{m} (J(\pi; P_i) - \nu)^+. \tag{6}$$

We can compute the empirical estimate of $\text{CVaR}_\alpha(J(\pi))$ by noting that it is equal to $\hat{F}(\nu, \pi)$ with $\nu = \text{VaR}_\alpha(J(\pi))$. The value of $\text{VaR}_\alpha(J(\pi))$ itself is estimated as follow: we compute the performance $J_i = J(\pi; P_i)$ $(i = 1, \dots, m)$, sort them in increasing order $J_{(1)} \le J_{(2)} \le \dots \le J_{(m)}$, and then estimate $\text{VaR}_\alpha(J(\pi))$ as $J_{(\lfloor m\alpha \rfloor)}$. The empirical estimate of $\text{CVaR}_\alpha(J(\pi))$ is then equal to $\text{mean}(J_{(1)}, \dots, J_{(\lfloor m\alpha \rfloor)})$.

To perform the policy (sub-) gradient over $\text{CVaR}_\alpha(J(\pi))$, we need to compute the sub-gradient of $\hat{F}(\nu, \pi)$ with respect to π. We use the property that the posterior $p(P|D)$ is not dependent on π. Let ν be an estimate of VaR_α as specified above. Then the sub-gradient of (6) with respect to π is given by:

$$\partial_\pi \hat{F}(\nu, \pi) = \frac{1}{m} \sum_{i=1}^{m} \beta_i \nabla_\pi J(\pi; P_i), \quad \text{where } \beta_i = \begin{cases} 1 & \text{if } J(\pi; P_i) < \nu, \\ [0, 1] & \text{if } J(\pi; P_i) = \nu, \\ 0 & \text{if } J(\pi; P_i) > \nu. \end{cases} \tag{7}$$

In the rest of the paper, we may overload the notation for β_i as $\beta_i(J, \nu)$ to refer to the components that are being compared to determine its value. Note that this sub-gradient computation is simpler than the procedure introduced by Tamar et al. (2015), in which the distribution also depends on the parameter from which we take the gradient. Our procedure is similar to the procedure used by Rockafellar et al. (2000) in their application to hedging.

How many samples n should be collected? We want to have enough samples for the estimates to be accurate while not wasting storage and computation on collecting and solving (i.e. through policy evaluation) more samples than necessary. To answer this question, Briggs and Ying (2018) propose a stopping condition based on distribution-free confidence intervals. This is the condition we use on L24 and L25 of in Algorithm 1. The relative error tolerance ϵ_{rel} controls how wide the confidence intervals are, determines with what confidence level the α-quantile falls in the confidence interval. For example, if we want 95% confidence intervals, we would set $\alpha = 0.05$, and depending on ϵ_{rel}, we would need to collect enough data to get narrow enough intervals and have 95% confidence that the α-quantile falls in those intervals. We note that using this stopping condition to estimate risk measures in RL is similar to how it is used by Angelotti et al. (2021), which we discuss in more depth at the end of this section.

Our overall approach is presented in Algorithm 1. We use the notation $V[i]$ to refer to the ith item in the sequence V. In each iteration of the algorithm, we collect some transitions by acting on the environment with the current policy and update the posteriors P; R (L4 - L5). We then take policy (sub-)gradient (PG) steps to optimize the objective (4) (L7-L11), starting from the previous iteration's best policy $\theta^{(k)}_{(best)}$ as the initial point for optimization in the next round. We use the function EVALUATE RISK in each step of PG to evaluate the risk for each set of policy parameters. This function also calculates and returns the PG for each of the posterior samples used to estimate the (C)VaR. r $V_k[i]$ is the PG of the ith value function in the sorted V, which may be computed using any PG method such as REINFORCE or an actor-critic algorithm. Note that although in each call of EVALUATE RISK, new samples from the posterior are collected, we found empirically that collecting a large number of samples from the posterior after each update of the posteriors and performing the PG step on those samples was much more stable for optimization. Note that after each update of the posterior, the risk-sensitive objective (4) and the corresponding optimizer (5) change. Since the subgradient is not necessarily in the direction of increasing $\hat{F}$, we keep track of the best iterate $\theta^{(k)}_{best}$ among $\theta_1, \ldots, \theta_k$, i.e., $\hat{F}(\theta^{k}_{(best)}; \theta_{(best)}) = \max_{j=1,\ldots,k} \hat{F}(\theta_j; \theta_j)$; where θ_j refers to the VaR estimate of θ_j.

Solving the optimization problem (5) is also considered by Angelotti et al. (2021). Similar to our work, the approximate the risk measure by sampling from the posterior. There are, however, a few key differences. Their problem setup is the of ine (batch) setting, and not the online setting of this work. They do not empirically study the effect of nding a risk-sensitive policy on the performance of a continually interacting agent, as we do later in Section 5. From the algorithmic perspective, their approach to nding a good policy is based on an exhaustive search in the space of all policies or a subset thereof. On the other hand, the algorithm in this section is a policy gradient-based approach that gradually updates the policy in the direction of improvement.

4. Optimistic Risk-Constrained Planning for MBRL

There is a tradeoff between performance and safety. An agent that behaves safely with respect to its belief about the environment may be hesitant to learn much about its environment. This causes it to act too conservatively and degrades its performance. For example, consider a navigating robot in an unknown terrain. If the robot encounters a crevice or hole in the ground on its rst few steps, its model of the environment will be biased towards assuming that it is surrounded by unsafe terrain. A risk-averse planner may lead to a policy that causes it to simply stay in one spot, and never accomplish its intended task.

To have a high performance, the agent needs to carefully balance its exploration and exploitation. This can be done through optimistic methods such as UCRL methods (Jaksch et al., 2010) or Thompson/Posterior sampling (Strens, 2000; Osband et al., 2013), which can be interpreted as a stochastic optimistic approach (Osband and Van Roy, 2017). Nonetheless, this may lead to some behaviours that are risky according to the agent's belief about its environment.

Our desire is to have an agent that has good performance while reducing the number of catastrophic events that it faces. It is thus natural to consider safety as a constraint rather than its own objective. We formulate this as follows:

$$\text{OptimisticPlanner}(\cdot; J) \tag{8a}$$

$$\text{subject to } \rho(J(\cdot; P)) \leq b; \tag{8b}$$

where OptimisticPlanner $(\cdot; J)$ is an optimistic planner, ρ is a risk measure, and b is the acceptable level of risk. The optimistic planner depends on the posterior $p(P|D)$ and the policy performance $J(\cdot; P)$ and may be formulated in different ways. The risk measure ρ can be replaced with any risk measure over the posterior $p(P|D)$. Here, we use the CVaR, as in the previous section.

Algorithm 1 PG Algorithm for Risk-sensitive Planning

Input: prior distributions $P_0(P)$; $R_0(R)$, quantile order $\alpha \in (0,1)$, relative error tolerance ε_{rel}, initial state distribution μ_0, confidence level δ, policy learning rates $\eta(\cdot)$, convergence criterion ε, initial parameters θ_1.
Initialization: policy parameters $\theta^1_{(best)} = \theta_1$, Dataset $\mathcal{D} = \{\}$.
for $k = 1, 2, \ldots$ do
  Collect transition(s) with policy $\theta^k_{(best)}$ from the environment and add to data: $\mathcal{D} := \mathcal{D} \cup \{X_t, A_t, X_{t+1}, R_t\}$.
  Update posteriors $P(P|\mathcal{D})$; $R(R|\mathcal{D})$.
  Policy parameters $\theta_1 = \theta^k_{(best)}$. Re-initialize $k = 1$.
  repeat
    $V_k, r_{V_k}, \widehat{CVaR}(\theta_k, \alpha) =$ EVALUATE RISK$(\theta_k, \varepsilon_{rel}, \delta, \mu_0, P, R)$
    $\theta_{k+1} \leftarrow \theta_k + \eta(k) \frac{1}{N_k} \sum_{i=1}^{N_k} \psi_\theta(V_k[i], \alpha) r_{V_k}[i]$
    $k \leftarrow k + 1$
  until $\|\theta_k - \theta_{k-1}\| < \varepsilon$.
  Update behaviour policy: $\theta^k_{(best)} \leftarrow \arg\max_{j=1,\ldots,k} \widehat{CVaR}(\theta_j, \alpha)$
end for

function EVALUATE RISK$(\theta, \varepsilon_{rel}, \delta, \mu_0, P, R)$
  Set continue-sampling $= 1$, and $V = \{\}$, $r_V = \{\}$.
  while continue-sampling do
    for $j = 1, 2, \ldots$ in parallel do
      Sample $P_j \sim P(P|\mathcal{D})$, $R_j \sim R(R|\mathcal{D})$
      $J(\theta, P_j)$ $\leftarrow$ Policy Evaluation on model P_j, reward function R_j
      $V \leftarrow$ append $J(\theta, P_j)$
      $r_V \leftarrow$ append $r_{J(\theta, P_j)}$
    end for
    Sort V, r_V in increasing order of V. Denote $N = |V|$.
    Find r, s with $|s - r|$ minimal, subject to $\sum_{i=r}^{s-1} \binom{N}{i} (1-\alpha)^{N-i} > 1 - \delta$
    continue-sampling $= V[s] - V[r] > \varepsilon_{rel}(V[L] - V[1])$
  end while
  Estimate of VaR: $\hat{\nu} = V[\lfloor \alpha N \rfloor]$
  CVaR: $\widehat{CVaR}(\theta, \alpha) = \text{mean}(V[1], \ldots, V[\lfloor \alpha N \rfloor])$.
  return $V, r_V, \hat{\nu}, \widehat{CVaR}(\theta, \alpha)$
end function

We consider several possibilities for OptimisticPlanner in this work but this is not meant to be an exhaustive list nor the best possible that could be achieved. The objectives we consider here will be referred to as "PSRL", "Upper CVaR", and "MaxOpt", which we describe below.

PSRL: The OptimisticPlanner (θ, J) in PSRL (Osband and Van Roy, 2017) is a randomized algorithm that returns θ_{P_i} argmax $J(\theta, P_i)$, where $P_i \sim P(\cdot|\mathcal{D})$. We check whether the constraint is satisfied for P_i. If not, we resample $P_i \sim P(\cdot|\mathcal{D})$, and repeat the process. This procedure is computationally very expensive. Every verification of whether the constraint is satisfied for a given policy θ_{P_i} requires the computation of CVaR$(J(\theta_{P_i}))$, which itself requires many samples from the posterior. In addition, this procedure might need to be repeated many times if the constraint is a difficult one to satisfy.

Upper CVaR: We consider the opposite end of the value function distribution from the constraint. We define $\overline{CVaR}(Z) = \min\{\nu + \frac{1}{\alpha} E[(Z - \nu)^+]\}$, where $\alpha \in (0,1)$ is the quantile order defined on the upper tail of the distribution of Z. The OptimisticPlanner is then OptimisticPlanner $(\theta, J) = \arg\max \overline{CVaR}(J(\theta, P))$. Referring to Figure 1, this objective optimizes the average performance in the upper α quantile of the posterior. Therefore, this objective acts optimistically with respect to the agent's belief about the true MDP, while the constraint ensures that the agent does not have many bad events given its belief.

Algorithm 2 PG Algorithm for Optimistic Risk-Constrained MBRL

Input: prior distributions $\overset{P}{0}(P)$; $\overset{R}{0}(R)$, quantile order $\in (0;1)$, relative error tolerance $_{rel}$, initial state distribution $_0$, con dence level , policy learning rates , convergence criterion , initial policy parameters $_0$.
Initialization: policy parameters $\overset{(0)}{0} = {}_0$, Lagrangian parameter $= {}_0$, initial penalty parameter $_0$.
for $i = 0; 1; 2; : : :$ do
  Collect transition(s) from environment and add to data $D \; [\; f \; X_t; A_t; X_{t+1}; R_t g$
  Update posteriors $^P(PjD)$; $^R(RjD)$
  Set risk constraint $b \quad \max \; CVaR \; (J(\overset{(i)}{0}; P))$ (using Alg 1). Re-initialize $_0$, $_0$
  for $k = 0; 1; 2; : : :$ do
    repeat
      $V_k; r \quad V_k; \; ; \; CVaR \; = \textsc{Evaluate Risk}(\overset{(i)}{k}; \; _{rel}; \; ; \; ; \; \overset{P}{}; \; \overset{R}{})$
      Update: $\overset{(i)}{k+1} \quad \overset{(i)}{k} + \quad (k) \quad r \quad f \; (J(\overset{(i)}{k}; P)) + \frac{1}{N_k} \overset{N_k}{\underset{i=0}{P}} \; _i(V_k[i]; \;)r \; _{\overset{(i)}{k}} V_k[i]$
    until $\overset{(i)}{k+1} \quad \overset{(i)}{k}$.
    Update: $_{k+1} \quad (\; _k \quad \frac{1}{k}(CVaR \; (J(\overset{(i)}{k+1}; P)) \quad b))$
    Penalty Update: $_{k+1} \; 2 \; (0; \; _k)$
  end for
  Set policy parameters $\overset{(i)}{0} = \overset{(i \; 1)}{k+1}$.
end for

MaxOpt: Similarly, in "MaxOpt", the idea is to act as optimistically as possible with respect to samples from the posterior (maximum sample in Figure 1), instead of taking the UpperCVaR approach where we are "cautiously optimistic". The objective is then $\text{OptimisticPlanner} \quad (; J) = \arg \; \max_{P \; 2f \; P_1; :::; P_m g} J(\; ; P); \quad P_1; : : : ; P_m \quad (jD);$ where the number of models is determined with the stopping condition used to estimate the constraint CVaR. This is simply a practical heuristic as the samples for estimating the constraint can be reused.

If $\text{OptimisticPlanner} \quad (; J)$ can be written as the optimization of a function, we denote it as $\max \; f \; (J(\; ; P))$. To solve (8), we use an Augmented Lagrangian approach (Wright et al., 1999), with multiplier $\in \mathbb{R}$ and Lagrangian:

$$L(\; ; \;) = f \; (J(\; ; P)) + \quad (CVaR \; (J(\; ; P)) \quad b); \tag{9}$$

We present the full algorithm in Alg. 2. At each iteration of the outer loop, we collect data and update posteriors. Given penalty $_0 > 0$, at each iteration of the inner loop, we first find an approximate maximizer $_{k+1}$ of (9) by taking gradient ascent steps using the gradient of the Lagrangian: $rL(\; ; \; _k) = r \; f \; (J(\; ; P)) + \; _k @CVaR \; (J(\; ; P))$; where the first term is a subgradient if f is "Upper CVaR" or MaxOpt. We then update the multiplier $_k$ to obtain $_{k+1}$ using:

$$_{k+1} \quad (\; _k \quad r \quad L(\; _{k+1}; \; _k = \; _k)) = \quad (\; _k \quad \frac{1}{k}(CVaR \; (J(\; _{k+1}; P)) \quad b)); \tag{10}$$

where $$ is a projection operator onto $\mathbb{R}^+$. Finally, we make the penalty parameter $_{k+1}$ smaller than $_k$. The starting point for the next iteration is set to the solution $_k$ of the current iteration. As the penalty parameter becomes smaller, constraint violations are penalized more and the infeasibility in the solution $_k$ becomes smaller (Wright et al., 1999). Another element of Alg. 2 is in the choice of acceptable risk b. Ideally we would like this value to be adaptively chosen, and make sense based on the environment and collected data. As a result, we propose to set b as: $b \quad \max \; _2 \; CVaR \; (J(\overset{(i)}{0}; P))$; with $\overset{(i)}{0}$ the solution from the previous iteration of data collection.

5. Empirical Studies

We present preliminary results on Alg.'s 1 and 2 on some finite domains: Chain (Strens, 2000), DoubleLoop (Strens, 2000), Frozen Lake (FL) (Sutton and Barto, 2018), Cliff-Walking (Sutton and Barto, 2018), LavaLake (Leike et al., 2017), and IslandNavigation (Leike et al., 2017). All these environments require a balance of exploration and exploitation, and other than Chain and DoubleLoop, have some states that are low value and should be avoided. The full

experimental setup we use and descriptions of domains we consider are given in Appendix A. As we are in a discrete state space, we use as prior for a Dirichlet distribution with equal parameters $s = 0.5$ for all states in X and estimate the likelihood of the data using a categorical distribution, which results in a Dirichlet posterior with parameters $_0 + n_{x;a;x^0}$, where $n_{x;a;x^0}$ are the transition counts for each transition from (x, a) to x^0 in the collected data. We use a Normal distribution with mean 0 and variance 1 for the prior of the reward function.

With Alg. 1, we directly optimize CVaR of the policy performance according to the posterior as we collect more data and make updates. Alternatively, we could optimize the expected value with respect to the posterior. We denote this as "PG" for policy gradient, and "PG with CE" for policy gradient with Certainty Equivalent model. In the former, we take the average gradient step over samples of MDPs. In the latter, we take gradient steps with respect to the certainty equivalent, or MLE, model. We expect that the overall performance of optimizing the CVaR would be worse for environments where error in the model does not cause catastrophic drops in performance. We expect optimizing the risk would perform worse as it is more cautious with respect to different realizations of the true environment, potentially exploring the environment less.

The results for comparing the CVaR, PG, and PG with CE are shown in part (b) of Figure 2 (and 4, 5, 6, 7, 8 in Appendix). We can see that across all environments tested, the PG with CE model has the worst CVaR on the posterior, indicating that although it may have good performance in certain cases, it is not acting safely with respect to its uncertainty in the environment, as expected. In addition, as expected, for most environments the PG or PG with CE perform better than CVaR. However, in environments that require some degree of "safe" or conservative behaviour (i.e. LavaLake, IslandNavigation) CVaR outperforms the other methods.

We also evaluate Alg. 2. As the constraint here limits the space of policies, simply planning optimistically without a constraint may result in better overall performance on the true environment as more optimistic policies may violate the constraint during training. On the other hand, as before, if the environment has areas of low-probability but very low-value, in the early stages of training our agent may not have collected much or any data from these areas, and thus it will contain inaccuracies (due to the prior) and uncertainty about these areas. In this case, having the risk constraint over our belief should lower the chances of learning a policy during planning that would venture into these areas of the true environment as the policy is in general more cautious with respect to its belief.

First, we check the effect of adding a risk constraint to our existing PG planner. The results of this experiment are shown in part (d) of Figure 2 (and 4, 5, 6, 7, 8 in Appendix). In parts (a) and (c) of the same Figures, we show the results of adding a risk constraint to an optimistic planner.

For the PG variants, again we notice a marked improvement in the policy performance in more "risky" environments (FrozenLake, LavaLake, and IslandNavigation) for the constrained version, and similar performance to the unconstrained versions in the other environments. For the optimistic planner variants (parts (a) and (c) of the Figures), we find a similar pattern as the PG variants (part (d)), although we also see the unconstrained planners sometimes outperforming the constrained versions in both performance and CVaR. This could point to an issue with optimization, as optimization of the constrained problem requires us to first find the optimal CVaR (Alg. 2) and also to find constraint-feasible policies, which may be more difficult if the constraint is active.

Another observation is that in general, as the methods learn better policies, their CVaR also increases. This points to the idea that a good, high-performing policy naturally will not have low CVaR as catastrophic events are already minor. However, we also do not have any guarantees that the policy behaves safely with respect to our belief about the environment, including during learning. Being risk-aware may eliminate intermittent policies that may in expectation have good performance but also have low CVaR.

6. Related Works

Taking parameteric uncertainty into account in model-based methods has been extensively studied in the literature. For a full review of the literature in this field, up to the year 2015, we refer the reader to Chapter 6 of (Ghavamzadeh et al., 2015). We highlight the most relevant as well as more recent works. Delage and Mannor (2010) investigate the "percentile" criterion, which, in the context of this work, is the VaR (applied to the posterior distribution). For the case of a Dirichlet prior, Delage and Mannor (2010) show that an approximately optimal solution can be derived. The setting of Delage and Mannor (2010) is for a fixed dataset and the exploration of a policy as more data is collected is not considered. Additionally, our approach is based on PG.

Figure 2: IslandNavigation. Solid lines are Avg of 8 runs, shaded areas Std. Err. (Top) Env Returns, (Bottom) Posterior CVaR.

Chen and Bowling (2012) generalize the percentile criterion by replacing it with a measure over percentiles. For example, the CVaR can be obtained by using a specific measure over percentiles. Chen and Bowling (2012) also propose a family of percentile measures "k-of-N" that can be tuned to closely approximate the CVaR. This measure is equivalent to sampling in the following way: for a policy, sample N MDPs from the posterior, select the set of k MDPs with the worst performances $J(\pi; P_i); i \in \{1, \ldots, k\}$, and choose an MDP uniformly at random from this set. The authors show that the k-of-N measure can be solved efficiently and give convergence guarantees. We point out two key differences between Alg. 1 and the approach of Chen and Bowling (2012). We use PG instead of counterfactual regret minimization. This allows us to easily incorporate the risk-aware objective as a constraint (Section 4). The second difference is in the sampling procedure used to approximate, for example, the CVaR, as k-of-N could also be used in PG calculations. Comparing the way we estimate the CVaR (UPDATE RISK in Alg. 1) to the k-of-N sampling procedure, we note that for comparable k and N parameters, our approach has a factor k lower variance in estimating the PG than k-of-N since the latter uses a single sample to estimate the PG whereas in our approach, the PGs for around k samples would be averaged.

The problems we consider in this work are sometimes addressed under the formulation of Bayes-Adaptive MDP (BAMDP) (Duff, 2002), where the problem is cast into a belief-state MDP by augmenting the state with the posterior belief. The optimal policy in this case (i.e. the Bayes-optimal policy) makes the optimal tradeoff between exploration and exploitation with respect to its belief (Duff, 2002). In our work, we aim to explicitly prevent the agent from taking actions in likely-to-be unfavourable areas. In other words, we sacrifice Bayes-optimality in order to ensure the policies during learning are "safe" with respect to the parametric uncertainty. Sharma et al. (2019) consider the problem of trading off exploration and risk-sensitivity over model space in Bayesian RL. The formulation in this work is different from ours as they formulate the problem as a two-player zero-sum game between the agent and an adversary that tries to modify the agent's beliefs. They also use a different approach to solving the problem instead of PG.

7. Conclusions and Future work

We introduced two algorithms based on applying the notion of risk to address parameteric uncertainty in model-based RL. One was based on directly optimizing the CVaR using PG. The other was using the CVaR as a constraint, and acting optimistically in the environment to collect data and explore. We empirically evaluated these algorithms on some finite domains and saw promising results. As part of future work, we would like to extend this work and the empirical evaluations to continuous domains with neural network function approximators, and posterior approximations. PG methods have been applied successfully in these domains (Sutton et al., 1999; Schulman et al., 2015, 2017) and thus this is a natural next step for our approach. Note that since we do not exactly calculate a VaR/CVaR, but estimate them using a finite number of samples and with a given confidence level, there is an error introduced in the calculation of the subgradient. It is not known how this error affects the gradient step procedure and its convergence, and we leave analysis of this to future work. For the constrained algorithm 2, and important question is how to handle situations where the constraint is difficult to achieve. We also leave exploration of further optimistic objectives for future work.

8. Acknowledgements

AMF acknowledges the funding from the Canada CIFAR AI Chairs program, as well as the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grant program. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

References

Carlo Acerbi and Dirk Tasche. Expected shortfall: a natural coherent alternative to value at risk. Economic notes, 31 (2):379–388, 2002.

Giorgio Angelotti, Nicolas Drougard, and Caroline Ponzoni Carvalho Chanel. Exploitation vs caution: Risk-sensitive policies for of ine learning. arXiv preprint arXiv:2105.13431, 2021.

András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with Bellman-residual minimization based tted policy iteration and a single sample path. Machine Learning, 71:89–129, 2008.

Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. Mathematical nance, 9(3):203–228, 1999.

Karl Johan Astöm and Björn Wittenmark. Adaptive Control. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

Dimitri P. Bertsekas and John N. Tsitsiklis. Neuro-Dynamic Programming. Athena Scienti c, 1996.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax .

K Briggs and FM Ying. How to estimate quantiles easily and reliably. Mathematics Today, 2018(February), 2018.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. Proceedings of the 36th International Conference on Machine Learning (ICML), 2019.

Katherine Chen and Michael Bowling. Tractable objectives for robust policy optimization. Advances in Neural Information Processing Systems, 25, 2012.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. The Journal of Machine Learning Research, 18(1):6070–6120, 2017.

Erick Delage and Shie Mannor. Percentile optimization for markov decision processes with parameter uncertainty. Operations research, 58(1):203–213, 2010.

Pierluca D'Oro, Alberto Maria Metelli, Andrea Tirinzoni, Matteo Papini, and Marcello Restelli. Gradient-aware model-based policy search. Thirty-Fourth AAAI Conference on Arti cial Intelligence (AAAI), 2020.

Michael O'Gordon Duff. Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes. University of Massachusetts Amherst, 2002.

Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. Mismatched no more: Joint model-policy optimization for model-based rl. arXiv preprint arXiv:2110.02758, 2021.

Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized policy iteration with nonparametric function spaces. Journal of Machine Learning Research (JMLR), 17(139):1–66, 2016.

Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In Arti cial Intelligence and Statistics, pages 1486–1494. PMLR, 2017.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. Foundations and Trends® in Machine Learning, 8(5-6):359–483, 2015.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. Advances in Neural Information Processing Systems, 33:5541–5552, 2020.

László Györ , Michael Kohler, Adam Krzyzak, and Harro Walk. A Distribution-Free Theory of Nonparametric Regression. Springer Verlag, New York, 2002.

David Ha and Jürgen Schmidhuber. World models. arXiv preprint arXiv:1803.10122, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. arXiv preprint arXiv:1912.01603, 2019.

Garud N Iyengar. Robust dynamic programming. Mathematics of Operations Research, 30(2):257–280, 2005.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. Journal of Machine Learning Research (JMLR), 11:1563 – 1600, 2010.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. arXiv preprint arXiv:2002.04523, 2020.

Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. Ai safety gridworlds. arXiv preprint arXiv:1711.09883, 2017.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for tted value iteration. Journal of Machine Learning Research (JMLR), 9:815–857, 2008.

Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. Operations Research, 53(5):780–798, 2005.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In International conference on machine learning, pages 2701–2710. PMLR, 2017.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) ef cient reinforcement learning via posterior sampling. In Advances in Neural Information Processing Systems, volume 26, 2013.

R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. Journal of risk, 2:21–42, 2000.

R Tyrrell Rockafellar, Stan Uryasev, and Michael Zabarankin. Master funds in portfolio analysis with general deviation measures. Journal of Banking & Finance, 30(2):743–778, 2006.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. Nature, 588(7839):604–609, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In International conference on machine learning, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. Lectures on Stochastic Programming: Modeling and Theory. SIAM, 2009.

Apoorva Sharma, James Harrison, Matthew Tsao, and Marco Pavone. Robust and adaptive planning under model uncertainty. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 410–418, 2019.

Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer, 2008.

Malcolm Strens. A bayesian framework for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 943–950, 2000.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning (ICML)*, 1990.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010.

Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. In *Twenty-Ninth AAAI Conference on Arti cial Intelligence*, 2015.

Samuele Tosatto, Matteo Pirotta, Carlo D'Eramo, and Marcello Restelli. Boosted tted Q-iteration. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

Appendix A. Experimental Setup

A.1 Environment Descriptions

A.1.1 CHAIN

This is the environment de ned by Strens (2000) and shown in Fig. 3b. There are two actions and 5 states. There is also a 0:2 probability of the agent's intended action being changed to the other action. Once state 5 is reached, the agent stays there and receives 10 reward until it “slips” again with probability 0:2. The optimal policy is to always choose action a, even though sometimes this results in the action b being taken. This problem requires effective exploration to solve.

A.1.2 DOUBLELOOP

This environment is also de ned by Strens (2000) and shown in Fig. 3c. It involves two loops of length 5 joined at a single start state. Two actions are available and transitions are deterministic. Taking action a repeatedly causes traversal of the right loop, with a reward of 1 for every 5 actions taken. Taking action b repeatedly causes traversal of the left loop, with a reward of 2 for every 5 actions taken. This problem requires a dif cult compromise between exploration and exploitation.

A.1.3 CLIFFWALKING

This is the environment proposed by (Sutton and Barto, 2018), and illustrated in Fig. 3a. The Goal state is an absorbing state with reward 1. Each action has a 0.2 probability of “failing”, and taking the agent to a random adjacent state.

A.1.4 FROZEN LAKE (FL)

This is the Frozen Lake environment of OpenAI Gym Brockman et al. (2016) with modi ed rewards. In this environment, the agent moves in a 4 4 grid starting in the upper left corner towards an absorbing goal state that has a reward of 10 and is in the lower right corner of the grid. Along the way, the grid has some holes, which are also absorbing states with rewards of 1. The rest of the grid has rewards sampled uniformly from the interval (0;0:8) at the generation of the map, and the rewards are xed thereafter (deterministic). The grid is also “slippery” with probability 1=3 the agent will move in the intended direction with probability 1=3 and will move in either perpendicular direction with equal probability of 1=3 in both directions. The map is randomly generated at the start of each episode, ensuring there is a hole-free path from the start to the goal.

A.1.5 LAVA LAKE AND ISLANDNAVIGATION

These domains are taken from Leike et al. (2017) and shown in Figures 3d and 3e. They each have 4 actions and are episodic, resetting to a randomized starting point (from gures). The agent must reach a Goal state (G) upon which it receives a reward of 50 and the episode ends. Otherwise, a reward of 1 is incurred unless the agent lands in the Lava/Water, with a reward of 50 and termination of the episode. In the IslandNavigation domain additional information is provided to the agent through a “safety distance” that maps the current environment state to the agent's Manhattan distance to the closest water cell (Leike et al., 2017).

A.2 Experimental Details

The policy is represented using parameters = (00; : : : ; xa; : : : ; jXjjAj) for xa 2 (X A), and de ned as:

$$(a|x) = \frac{e^{xa}}{\sum_{a'} e^{xa'}}:$$

(a) Cliff-walking reproduced from Sutton and Barto (2018).

(b) Chain MDP reproduced from Strens (2000).

(c) Chain MDP reproduced from Strens (2000).

(d) LavaLake MDP reproduced from Leike et al. (2017)

(e) IslandNavigation MDP reproduced from Leike et al. (2017)

Figure 3: Illustrations for two of the MDPs used in the experiments.

Before policy updates begin, the posteriors are updated with data collected from a random policy for 10,000 environment steps.

The PG is calculated exactly on collected samples from the posterior, as we are in a tabular setting (i.e. we do not use samples from each model separately but rather solve each MDP exactly).

Hyperparameters, for all envs:

- quantile order : 0.1,
- upper quantile order : 0.1,
- signi cance level : 0.1,
- relative error tolerance$_{\text{rel}}$: 0.1,
- discount factor: 0.99,
- policy learning rate: 0.1 for unconstrained algorithms, 0.01 for constrained,
- learning rate: 0.01,
- Number of iterations of constrained optimization (outer loop, i.e. number of updates to penalty): 20,
- Number of samples taken for PG: 500, for risk-based methods the number could be adaptively chosen based on con dence intervals, but we found a xed number 500 was both suf cient for the given con dence interval and ef cient.
- Seeds are set in the numpy random number generator, the jax Bradbury et al. (2018) random number generator, and to seed the environments. It would be possible that two runs with the same seeds are not exactly identical however since there may be lower level sources of randomness we did not account for.

Appendix B. Additional Experiments

We present additional experiments in Figures 4, 5, 6, 7, 8.

(a) Comparison of Max Opt algorithm and Max Opt with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

(b) Comparison of PG algorithm using samples from the posterior, PG with the Certainty Equivalent or MLE model, and optimizing the CVaR directly. (Top) Average Rewards, (Bottom) CVaR on Posterior.

(c) Comparison of Upper CVaR algorithm and Upper CVaR with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

(d) Comparison of PG algorithm and PG with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

Figure 4: Performances on Chain env

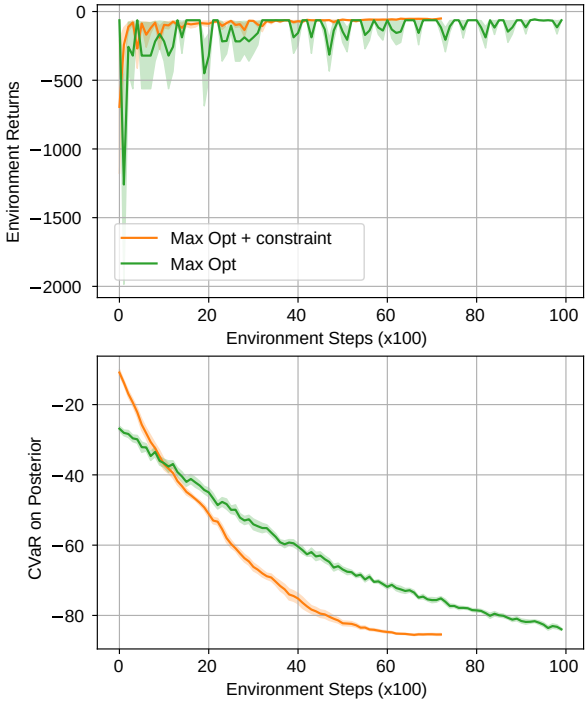
(a) Comparison of Max Opt algorithm and Max Opt with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

(b) Comparison of PG algorithm using samples from the posterior, PG with the Certainty Equivalent or MLE model, and optimizing the CVaR directly. (Top) Average Rewards, (Bottom) CVaR on Posterior.

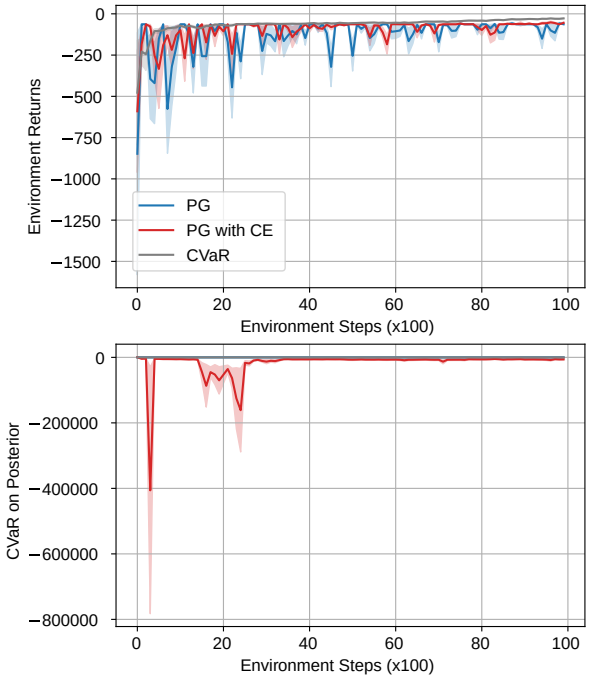
(c) Comparison of Upper CVaR algorithm and Upper CVaR with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

(d) Comparison of PG algorithm and PG with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

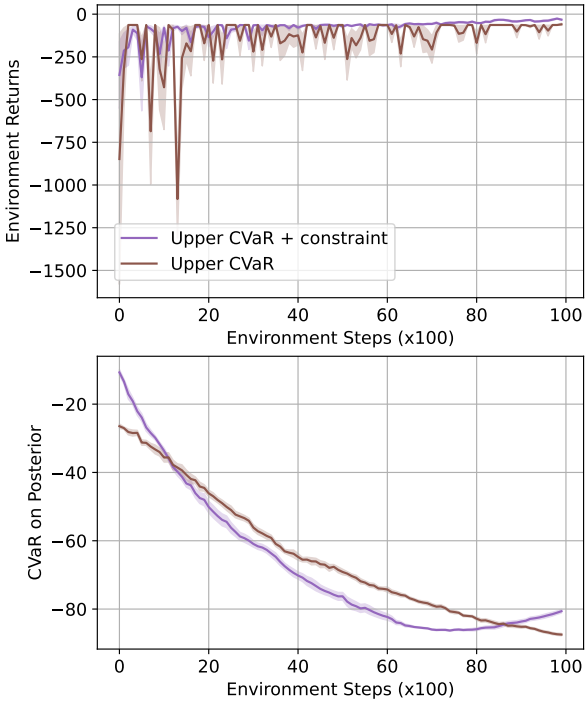
Figure 5: Performances on DoubleLoop env



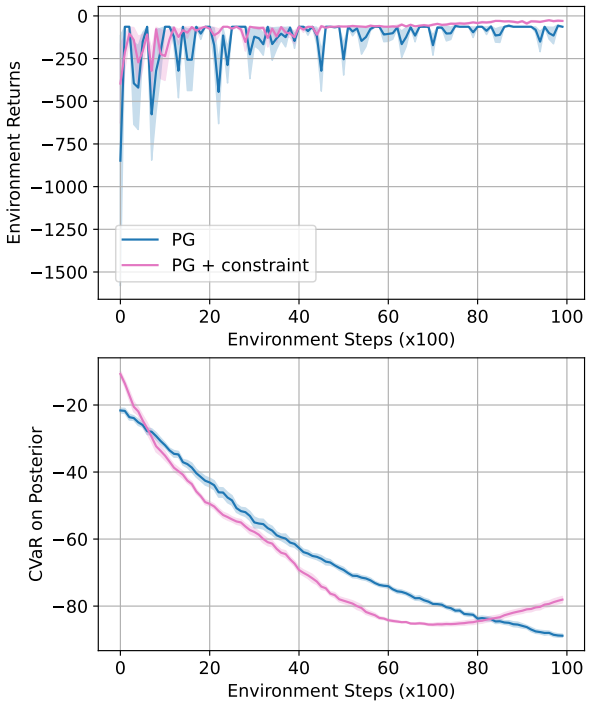
(a) Comparison of Max Opt algorithm and Max Opt with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.



(b) Comparison of PG algorithm using samples from the posterior, PG with the Certainty Equivalent or MLE model, and optimizing the CVaR directly. (Top) Average Rewards, (Bottom) CVaR on Posterior.

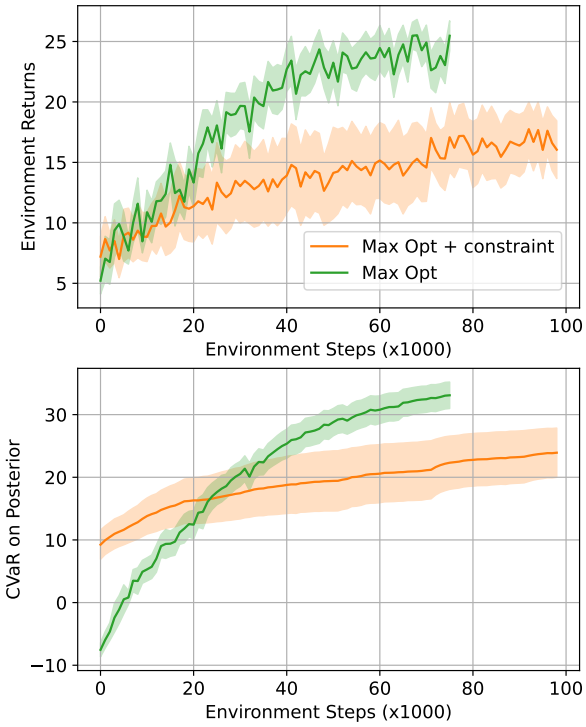


(c) Comparison of Upper CVaR algorithm and Upper CVaR with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

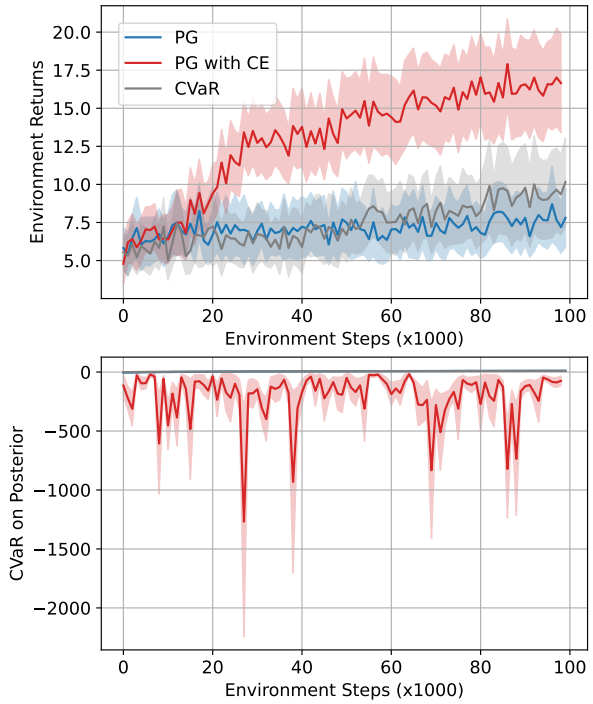


(d) Comparison of PG algorithm and PG with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

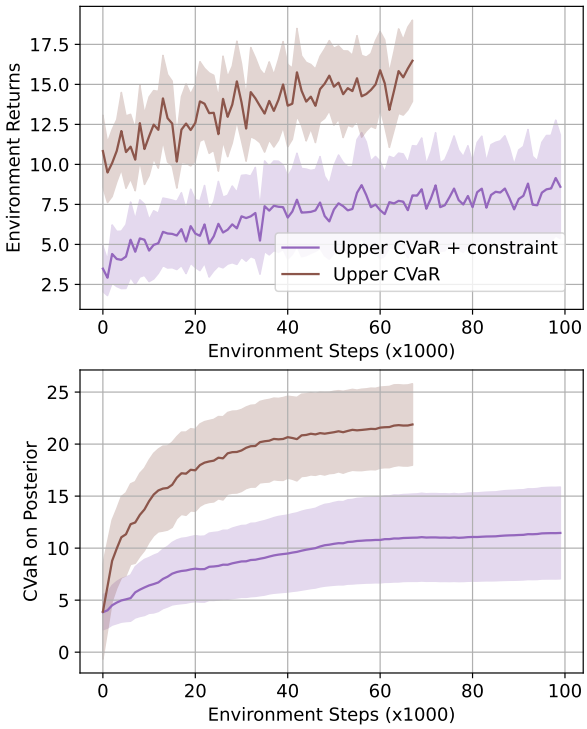
Figure 6: Performances on CliffWalking env



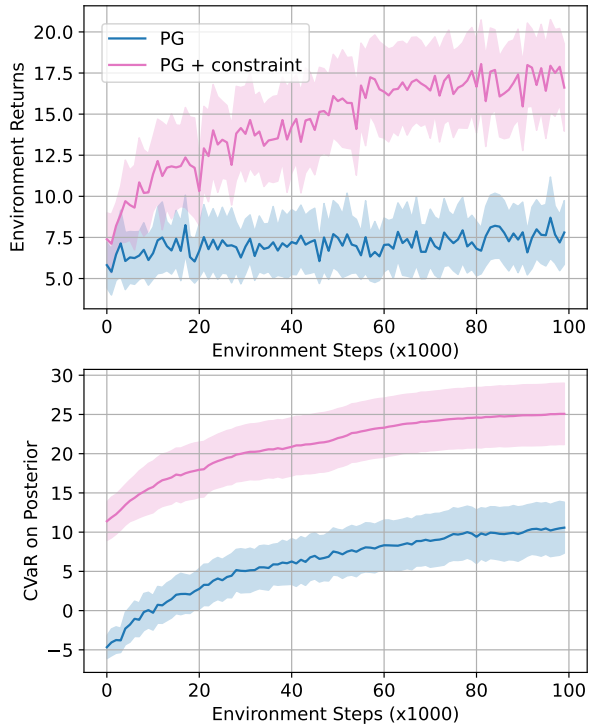
(a) Comparison of Max Opt algorithm and Max Opt with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.



(b) Comparison of PG algorithm using samples from the posterior, PG with the Certainty Equivalent or MLE model, and optimizing the CVaR directly. (Top) Average Rewards, (Bottom) CVaR on Posterior.

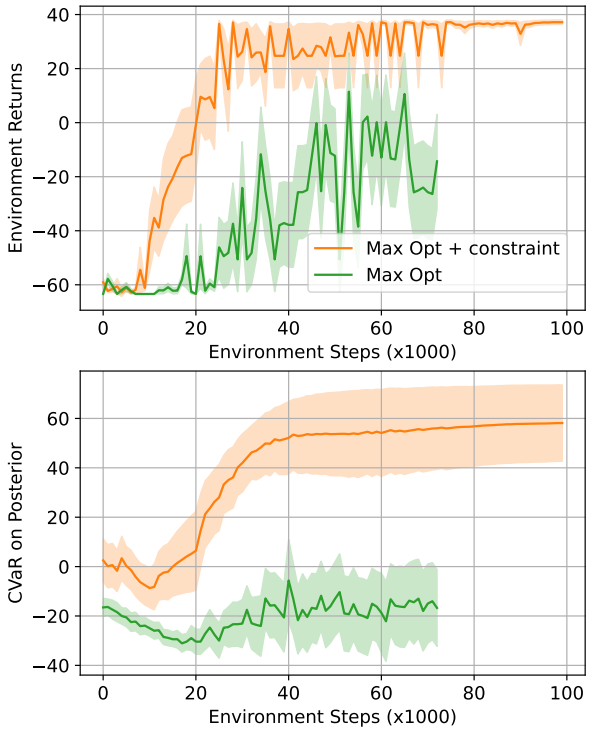


(c) Comparison of Upper CVaR algorithm and Upper CVaR with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.

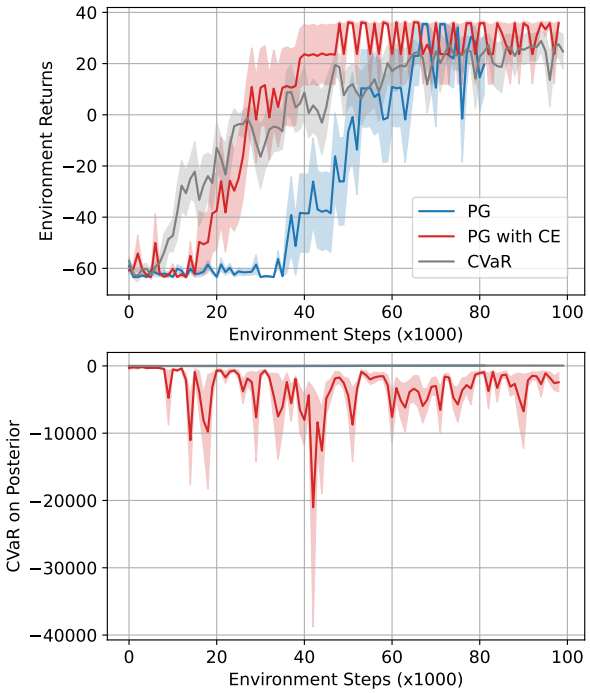


(d) Comparison of PG algorithm using samples from the posterior, PG with the Certainty Equivalent or MLE model, and optimizing the CVaR directly. (Top) Average Rewards, (Bottom) CVaR on Posterior.

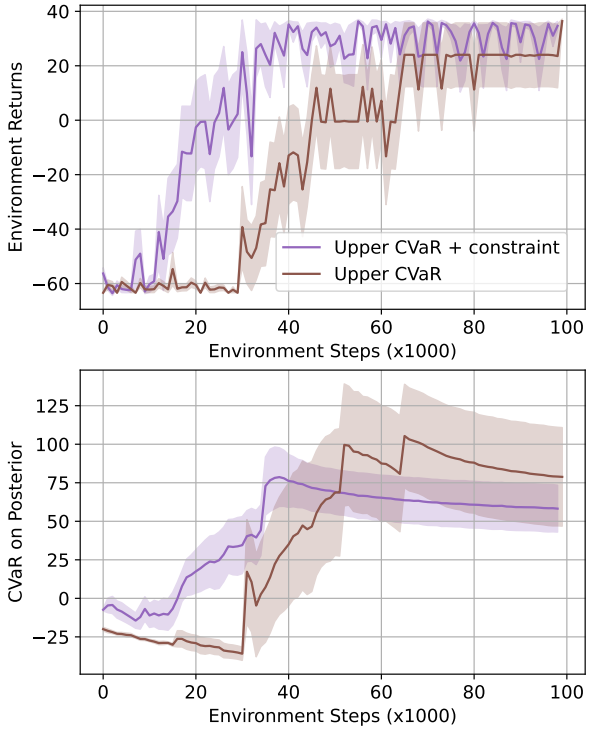
Figure 7: Performances on FrozenLake (FL) env



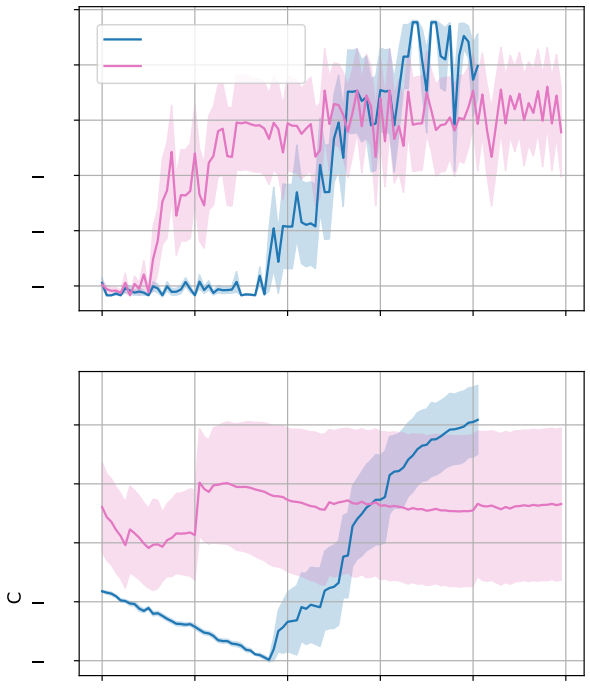
(a) Comparison of Max Opt algorithm and Max Opt with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.



(b) Comparison of PG algorithm using samples from the posterior, PG with the Certainty Equivalent or MLE model, and optimizing the CVaR directly. (Top) Average Rewards, (Bottom) CVaR on Posterior.



(c) Comparison of Upper CVaR algorithm and Upper CVaR with CVaR-constraint. (Top) Average Rewards, (Bottom) CVaR on Posterior.



(d) Comparison of PG algorithm using samples from the posterior, PG with the Certainty Equivalent or MLE model, and optimizing the CVaR directly. (Top) Average Rewards, (Bottom) CVaR on Posterior.

Figure 8: Performances on LavaLake