Risk-Aware Transfer in Reinforcement Learning using Successor Features

Michael Gimelfarb[*]
University of Toronto
mike.gimelfarb@mail.utoronto.ca

André Barreto
DeepMind
andrebarreto@google.com

Scott Sanner[*]
University of Toronto
ssanner@mie.utoronto.ca

Chi-Guhn Lee
University of Toronto
cglee@mie.utoronto.ca

Abstract

Sample efficiency and risk-awareness are central to the development of practical reinforcement learning (RL) for complex decision-making. The former can be addressed by transfer learning and the latter by optimizing some utility function of the return. However, the problem of transferring skills in a risk-aware manner is not well-understood. In this paper, we address the problem of risk-aware policy transfer between tasks in a common domain that differ only in their reward functions, in which risk is measured by the variance of reward streams. Our approach begins by extending the idea of generalized policy improvement to maximize entropic utilities, thus extending policy improvement via dynamic programming to sets of policies *and* levels of risk-aversion. Next, we extend the idea of successor features (SF), a value function representation that decouples the environment dynamics from the rewards, to capture the variance of returns. Our resulting risk-aware successor features (RaSF) integrate seamlessly within the RL framework, inherit the superior task generalization ability of SFs, and incorporate risk-awareness into the decision-making. Experiments on a discrete navigation domain and control of a simulated robotic arm demonstrate the ability of RaSFs to outperform alternative methods including SFs, when taking the risk of the learned policies into account.

1 Introduction

Reinforcement learning (RL) is a general framework for solving sequential decision-making problems, in which an agent interacts with an environment and receives continuous feedback in the form of rewards. However, many classical algorithms in RL do not explicitly address the need for *safety*, making them unreliable and difficult to deploy in some real-world applications [10]. One reason for this is the relative *sample inefficiency* of model-free RL algorithms, which often require millions of costly or dangerous interactions with the environment or fail to converge altogether [44, 46]. Transfer learning addresses these problems by incorporating prior knowledge or skills [23, 41]. Despite this, using the expected return as a measure of optimality could still lead to undesirable behavior such as excessive risk-taking, since low-probability catastrophic outcomes with negative reward and high variance could be underrepresented [29]. For this reason, risk-awareness is becoming an important aspect in the design of practical RL [14]. Thus, an ultimate goal of developing reliable systems should be to ensure that they are both sample efficient and risk-aware.

[*]Affiliate to Vector Institute, Toronto, Canada.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

	Transfers Skills	Exploits Task Structure	Risk-Sensitive
RL [7, 11, 20, 27, 28, 30, 36, 39, 45]	✗	✗	✓
Transfer [15, 17, 19, 25, 26, 38, 43]	✓	✗	✓
Successor Features [2–4, 8]	✓	✓	✗
RaSF (Ours)	✓	✓	✓

Table 1: Comparison of RaSF with relevant work in transfer learning and risk-aware RL.

We take a step in this direction by studying the problem of risk-aware policy transfer between tasks with different goals. A powerful way to tackle this problem in the risk-neutral setting is the *GPI/GPE* framework, of which *successor features* (SF) are a notable example [2]. Here, *generalized policy improvement* (GPI) provides a theoretical framework for transferring policies with monotone improvement guarantees, while *generalized policy evaluation* (GPE) facilitates the efficient evaluation of policies on novel tasks and is a key component in satisfying the assumptions of GPI in practice. Together, GPI/GPE provide strong transfer benefits in novel task instances even before any direct interaction with them has taken place, a phenomenon we call *task generalization*. The key to the superb generalization of GPI/GPE lies in their ability to directly exploit the structure of the task space, taking advantage of subtle differences and commonalities between task goals to transfer skills seamlessly in a *composable* manner. This property could be an effective way of tackling problems in *offline RL* [24], such as the transfer of skills learned in a simulator to a real-world environment. However in many cases, such as helicopter flight control [16], making one wrong decision could lead to catastrophic outcomes. Hence, being risk-aware could offer one way to avoid worst-case outcomes when transferring skills in real-world settings.

Contributions. We contribute a novel successor feature framework for transferring policies with the goal of maximizing the entropic utility of return in MDPs (Section 2.2). Intuitively, the entropic utility encourages agents to follow policies with predictable and controllable returns characterized by low variance, thus providing a natural way to incorporate risk-awareness. Furthermore, while our theoretical framework could be extended to other classes of utility functions, the entropic utility has many favorable mathematical properties [13, 22] that we exploit directly in this work to achieve *optimal* transfer (Lemma 1, Theorem 1 and 2). We also derive a form of risk-aware GPE based on the mean-variance approximation, in which the sufficient statistics of the return distribution can be computed directly (Section 3.3) or by leveraging recent developments in distributional RL [6]. Our resulting approach, which we call *Risk-Aware Successor Features* (RaSF), is able to exploit the task structure to achieve task generalization with respect to novel goal instances as well as levels of risk aversion, where emphasis is placed on avoiding high volatility of returns. Our approach is also complementary to other advances in successor features, including feature learning [3], universal approximation [8], exploration [21], and non-stationary reward preferences [4]. Empirical evaluations on discrete navigation and continuous robot control domains (Section 4) demonstrate the ability of RaSFs to better manage the trade-off between return and risk and avoid catastrophic outcomes, while providing excellent generalization on novel tasks in the same domain.

Related Work. The entropic and mean-variance objectives are popular ways of incorporating risk-awareness in RL [7, 11, 20, 27, 28, 30, 36, 39, 45]. However, transferring learned skills between tasks while taking risk into account is a difficult problem. One way to implement risk-aware transfer is to learn a critic [38] or teacher [43] that can guide an agent toward safer behaviors on future tasks. The risk-aware transfer of a policy from a simulator to a real-world setting has also been studied in the area of robotics [17]. Another approach for reusing policies is the *probabilistic policy reuse* of García and Fernández [15], but requires strong assumptions on the task space. *Hierarchical RL* (HRL) is another related approach that relies on hierarchical abstractions, enabling an agent to decompose tasks into a hierarchy of sub-tasks, and facilitating the transfer of temporally-extended skills from sub-tasks to the parent task. The *CISR* approach of Mankowitz et al. [26] is the first to investigate safety explicitly within HRL, followed up by work on *safe options* [18, 19, 25]. However, none of the existing work takes advantage of the compositional structure of task rewards to transfer skills while optimizing the variance-adjusted return, which is the problem we tackle in this paper (see Table 1).

2 Preliminaries

2.1 Markov Decision Process

Sequential decision-making in this paper follows the *Markov decision process* (MDP), defined as a four-tuple $\langle \mathcal{S}, \mathcal{A}, r, P \rangle$: $\mathcal{S}$ is a set of states; $\mathcal{A}$ is a finite set of actions; $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a bounded reward function, where $r(s, a, s')$ is the immediate reward received upon transitioning to state s' after taking action a in state s; and $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ is the transition function for state dynamics, where $P(s'|s, a)$ is the probability of transitioning to state s' immediately after taking action a in state s.

In the episodic MDP setting, decisions are made over a horizon $\mathcal{T} = \{0, 1, \dots T\}$ where $T \in \mathbb{N}$. We define a *stochastic Markov policy* as a mapping $\pi : \mathcal{S} \times \mathcal{T} \to \mathscr{P}(\mathcal{A})$, where $\mathscr{P}(\mathcal{A})$ denotes the set of all probability distributions over $\mathcal{A}$. Similarly, a *deterministic Markov policy* is a mapping $\pi : \mathcal{S} \times \mathcal{T} \to \mathcal{A}$. In the *risk-neutral* setting, the goal is to find a policy π that maximizes the expected sum of future rewards after initially taking action a in state s,

$$Q_h^\pi(s, a) = \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)} \left[\sum_{t=h}^{T} r(s_t, a_t, s_{t+1}) \,\middle|\, s_h = s,\, a_h = a,\, a_t \sim \pi_t(s_t) \right].$$

In this case, it is possible to show that a deterministic Markov policy π^* is optimal [33]. The theoretical framework in this paper also allows for time-dependent reward or transition functions.

2.2 Entropic Utility Maximization

We incorporate risk-awareness into the decision-making by maximizing the *entropic utility* U_β of the cumulative reward, defined for a fixed $\beta \in \mathbb{R}$ as

$$U_\beta[R] = \frac{1}{\beta} \log \mathbb{E}\left[e^{\beta R}\right], \tag{1}$$

for real-valued random variables R on a bounded support $\Omega \subset \mathbb{R}$. An important property of the entropic utility is the Taylor expansion $U_\beta[R] = \mathbb{E}[R] + \frac{\beta}{2}\mathrm{Var}[R] + O(\beta^2)$. Interpreting the risk as return variance, β can now be interpreted as the risk aversion of the agent: choosing $\beta < 0$ ($\beta > 0$) leads to *risk-averse* (*risk-seeking*) behavior, while $\beta = 0$ is *risk-neutral*, e.g. $U_0[R] = \mathbb{E}[R]$.

Specializing (1) to the MDP setting, the goal is to maximize

$$\mathcal{Q}_{h,\beta}^\pi(s, a) = U_\beta \left[\sum_{t=h}^{T} r(s_t, \pi_t(s_t), s_{t+1}) \right] \tag{2}$$

over all policies starting from $s_h = s$ and $a_h = a$. As in the risk-neutral setting, it is possible to show that a deterministic Markov policy is optimal [5]. Furthermore, $\mathcal{Q}_{h,\beta}^\pi$ can be computed iteratively through time using the *Bellman equation* [9, 32]:

$$\begin{aligned} \mathcal{Q}_{h,\beta}^\pi(s, a) &= U_\beta \left[r(s, a, s') + \mathcal{Q}_{h+1,\beta}^\pi(s', \pi_{h+1}(s')) \right] \\ &= \frac{1}{\beta} \log \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\exp\left\{ \beta \left(r(s, a, s') + \mathcal{Q}_{h+1,\beta}^\pi(s', \pi_{h+1}(s')) \right) \right\} \right], \end{aligned} \tag{3}$$

starting with $\mathcal{Q}_{T+1,\beta}^\pi(s, a) = 0$. In fact, (1) is the *only* utility function that has this equivalence, and other key properties (Lemma 1), while also satisfying *time consistency* that ensures the learned risk-aware behaviors remain consistent across time [22]. In this paper, we use (3) to establish a general GPI framework for risk-aware transfer learning with provable guarantees, and leverage approximations of (2) to learn portable policy representations.

In reinforcement learning, the Bellman equation is not applied directly since it suffers from the curse of dimensionality when $\mathcal{S}$ is high-dimensional or continuous, and since neither the dynamics nor the reward function are often known. Instead, the agent interacts with the environment using a stochastic exploration policy π^e, collects trajectories $\{(s_t, a_t, s_{t+1}, r_{t+1})\}_{t=0}^{T-1}$, and updates $\mathcal{Q}_{h,\beta}(s_t, a_t)$ via sample approximations $\hat{U}_\beta \approx U_\beta$ [36]. Our goal is to ameliorate the relative sample-inefficiency of RL through transfer learning that we aim to generalize to the risk-aware setting.

2.3 Transfer Learning

We now formalize the general *transfer learning* problem. Let $\mathcal{M}$ be the set of all MDPs with shared transition function P but different (bounded) reward functions. A fixed set of source tasks $M_1, \dots M_n \in \mathcal{M}$ is instantiated, and their corresponding optimal policies $\pi_1, \dots \pi_n$ are estimated. Our main goal is to transfer these resulting source policies to a new target task $M_{n+1} \in \mathcal{M}$, to obtain a policy π^*_{n+1} whose utility is better than one learned from scratch using only a fixed number of samples from M_{n+1}. We refer to this outcome as *positive transfer*.

As discussed earlier, a standard way to implement transfer learning is the GPI/GPE framework of Barreto et al. [2]. The core mechanism that enables positive transfer in the risk-neutral setting is called *generalized policy improvement* (GPI). Specifically, the set of source policies $\pi_1, \dots \pi_n$ are evaluated on the target task M_{n+1} to obtain corresponding values $Q^{\pi_1}_{n+1}, \dots Q^{\pi_n}_{n+1}$. Given a mechanism that can perform this policy evaluation step efficiently with some small error ε — namely successor features discussed and extended in Section 3.3 — an agent then selects actions in a greedy manner by following policy $\pi(s) \in \arg\max_a \max_{j=1\dots n} Q^{\pi_j}_{n+1}(s,a)$ in state s. The policy π corresponds to a strict policy improvement operator, and thus fulfills our requirements for positive transfer.

3 Risk-Aware Transfer Learning

An obvious challenge of applying GPI in the risk-aware setting is that transferring optimal risk-neutral source policies does not guarantee risk-aware optimality in the target task. A much stronger observation is that, even if the source policies π_j are risk-aware, performing the policy evaluation step in a risk-neutral way can still break the risk-awareness of GPI. This makes the extension of GPI to the risk-aware setting a non-trivial problem.

3.1 A Motivating Example

To see this, consider the MDP shown in Figure 1, which involves navigating from an initial state 'S' to a goal state 'G' in a grid with stochastic transitions. Traps of two types (X, Y) are placed in fixed cells, which upon entry terminate the episode with fixed costs, summarized compactly as pairs (c_1, c_2). We define two source tasks with costs $(20, 20)$ and $(0, 0)$ and a target task with costs $(20, 0)$. The optimal policies for $\beta = -0.1$ induce three different trajectories: safe (blue) and hazardous routes (red) for the source tasks, and a relatively safe route (green) for the target task. We also note that risk-awareness is prerequisite for this problem, since a risk-neutral agent prefers the hazardous path when optimizing any of the tasks.

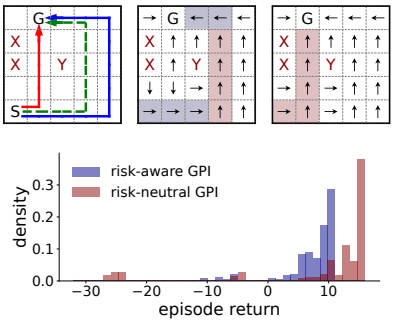

Figure 1: Comparing risk-aware and risk-neutral GPI. The risk-aware (for $\beta = -0.1$) GPI policy is shown in the top-middle plot, while the risk-neutral (for $\beta = 0$) GPI policy is shown in the top-right plot.

We compute the GPI policies π with risk-averse ($\beta = -0.1$) and risk-neutral ($\beta = 0$) policy evaluation, as shown in the middle and rightmost plots in the top row in Figure 1. The bottom plot shows the distribution of returns collected over 5,000 test runs by acting according to the two GPI policies. Risk-averse policy evaluation results in an optimal risk-averse GPI policy corresponding to the green trajectory, whereas risk-neutral policy evaluation does not, even though both source policies are optimal on their corresponding tasks. Interestingly, the risk-averse GPI policy is a non-trivial stitching of the two source policies.

3.2 Risk-Aware Generalized Policy Improvement

Motivated by this example, we conjecture that the risk-awareness of the GPI policy π is primarily dependent on the way in which the source policies are evaluated in target task instances. In this section, we describe theoretical results that generalize the concept of generalized policy iteration to the problem of maximizing the entropic utility of returns.

We begin by summarizing key properties necessary for establishing convergence of risk-aware GPI in the following lemma.

Lemma 1. *Let $\beta \in \mathbb{R}$ and X, Y be arbitrary random variables on Ω. Then:*

(1) (monotonicity) if $\mathbb{P}(X \geq Y) = 1$ then $U_\beta[X] \geq U_\beta[Y]$
(2) (cash invariance) $U_\beta[X + c] = U_\beta[X] + c$ for every $c \in \mathbb{R}$
(3) (convexity) if $\beta < 0$ ($\beta > 0$) then U_β is a concave (convex) function
(4) (non-expansion) for $f, g : \Omega \to \Omega$, it follows that

$$|U_\beta[f(X)] - U_\beta[g(X)]| \leq \sup_{P \in \mathscr{P}_X(\Omega)} \mathbb{E}_P |f(X) - g(X)|,$$

where $\mathscr{P}_X(\Omega)$ is the set of all probability distributions on Ω that are absolutely continuous w.r.t. the true distribution of X.

A proof is provided in Appendix B.2. Properties (1)-(3) characterize *concave utilities* [13], which intuitively can be seen as minimal requirements for rational decision-making: (1) a lottery that pays off more than another in every possible state of the world should always be preferred; (2) adding cash to a position should not increase its risk; and (3) the overall utility can be improved by diversifying across different risks. Property (4) is a derivative of the first three, and thus the theoretical results in this work could be extended to the broader class of iterated concave utilities [34].

We can now state the first main result of the paper.

Theorem 1 (GPI for Entropic Utility). *Let $\pi_1, \dots \pi_n$ be arbitrary deterministic Markov policies with approximate entropic utilities $\tilde{\mathcal{Q}}^{\pi_1}_{h,\beta}, \dots \tilde{\mathcal{Q}}^{\pi_n}_{h,\beta}$ when evaluated in an arbitrary task M, with error $|\tilde{\mathcal{Q}}^{\pi_i}_{h,\beta}(s,a) - \mathcal{Q}^{\pi_i}_{h,\beta}(s,a)| \leq \varepsilon$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $i = 1 \dots n$ and $h \in \mathcal{T}$. Define*

$$\pi_h(s) \in \arg\max_{a \in \mathcal{A}} \max_{i=1\dots n} \tilde{\mathcal{Q}}^{\pi_i}_{h,\beta}(s,a), \quad \forall s \in \mathcal{S}. \tag{4}$$

Then,

$$\mathcal{Q}^{\pi}_{h,\beta}(s,a) \geq \max_i \mathcal{Q}^{\pi_i}_{h,\beta}(s,a) - 2(T-h+1)\varepsilon, \quad h \leq T.$$

Thus, evaluating the risk of source policies using U_β provides monotone improvement guarantees for GPI, and thus satisfies our definition of positive transfer. Another significant property of the bound in Theorem 1, and the one in Theorem 2 below, is the linear separation between the optimal utility and the approximation error ε. Knowing how the optimality of π explicitly depends on ε, and how errors are propagated throughout the transfer learning process, is critical for developing reliable RL with predictable behavior, and highlights a key advantage of making GPI risk-aware. The additive relationship between the optimality and the approximation error in Theorem 1 further explains why SFs are robust to approximation errors. This becomes particularly advantageous in our setting, since estimating utilities U_β accurately with GPE is more complicated than the risk-neutral setting.

A stronger result for GPI can be derived when the source policies $\pi_1, \pi_2 \dots \pi_n$ are ε-optimal, and policy evaluation is once again performed using U_β. In this case, the optimality of GPI is determined by the similarity δ_r between the source and target task instances.

Theorem 2. *Let $\mathcal{Q}^{\pi_i^*}_{h,\beta}$ be the utilities of optimal Markov policies π_i^* from task M_i but evaluated in task M with reward function $r(s,a,s')$. Furthermore, let $\tilde{\mathcal{Q}}^{\pi_i^*}_{h,\beta}$ be such that $|\tilde{\mathcal{Q}}^{\pi_i^*}_{h,\beta}(s,a) - \mathcal{Q}^{\pi_i^*}_{h,\beta}(s,a)| \leq \varepsilon$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $h \in \mathcal{T}$ and $i = 1 \dots n$, and let π be the corresponding policy in (4). Finally, let $\delta_r = \min_{i=1\dots n} \sup_{s,a,s'} |r(s,a,s') - r_i(s,a,s')|$. Then,*

$$\left|\mathcal{Q}^{\pi}_{h,\beta}(s,a) - \mathcal{Q}^{*}_{h,\beta}(s,a)\right| \leq 2(T-h+1)(\delta_r + \varepsilon), \quad h \leq T.$$

These results are proved in Appendix B.2 for the episodic setting and in Appendix B.3 for the discounted setting. Also, please note that these bounds are tight (see, e.g. Nemecek and Parr [31] for the risk-neutral setting). Finally, while not required in this work, the above results could be extended to more general settings in risk-averse control [34], though practical implementation of GPE in these settings remains an open problem.

3.3 Risk-Aware Generalized Policy Evaluation

Following Barreto et al. [2], let $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \to \mathbb{R}^d$ be a bounded and task-independent feature map, and consider the following linear representation of rewards,

$$r(s, a, s') = \boldsymbol{\phi}(s, a, s')^\intercal \mathbf{w}, \quad \forall s, a, s',$$

where $\mathbf{w} \in \mathbb{R}^d$ is a task-dependent vector of reward parameters. The *risk-neutral* return becomes:

$$\begin{aligned} Q_h^\pi(s, a) &= \mathbb{E}_P \left[\sum_{t=h}^{T} \boldsymbol{\phi}(s_t, a_t, s_{t+1})^\intercal \mathbf{w} \,\middle|\, s_h = s,\, a_h = a,\, a_t \sim \pi_t(s_t) \right] \\ &= \mathbb{E}_P \left[\sum_{t=h}^{T} \boldsymbol{\phi}(s_t, a_t, s_{t+1}) \,\middle|\, s_h = s,\, a_h = a,\, a_t \sim \pi_t(s_t) \right]^\intercal \mathbf{w} = \boldsymbol{\psi}_h^\pi(s, a)^\intercal \mathbf{w}, \end{aligned} \tag{5}$$

where $\boldsymbol{\psi}_h^\pi(s, a)$ are the *successor features* (SFs) associated with the policy π. The linear dependence of the return on $\mathbf{w}$ allows for instantaneous policy evaluation in novel tasks with arbitrary reward preferences $\mathbf{w}$, making it a particular — and perhaps the canonical — instantiation of GPE. More critically, $\boldsymbol{\psi}_h^\pi$ can be seen as a *task-independent* and highly portable linear feature representation of policies, and it is the key to the generalization ability of SFs on novel task instances.

The concept of GPE can be generalized to incorporate entire distributions of the return. Repeating the above derivation for the entropic utility (2), we have:

$$\mathcal{Q}_{h,\beta}^\pi(s, a) = U_\beta \left[\sum_{t=h}^{T} r(s_t, \pi_t(s_t), s_{t+1}) \right] = U_\beta \left[\Psi_h^\pi(s, a)^\intercal \mathbf{w} \right], \tag{6}$$

corresponding to the random vector $\Psi_h^\pi(s, a) = \sum_{t=h}^{T} \boldsymbol{\phi}_t$ of unrealized feature returns at time h. Thus, we have transformed the problem of estimating the utility of returns into the problem of estimating the distribution of $\Psi_h^\pi(s, a)^\intercal \mathbf{w}$. The key question now is how to estimate this distribution for fast GPE.

A natural way to do this is by applying a second-order Taylor expansion for U_β, since it allows us to precompute and cache the necessary moments of the return distribution:

$$\begin{aligned} U_\beta \left[\Psi_h^\pi(s, a)^\intercal \mathbf{w} \right] &= \mathbb{E}_P[\Psi_h^\pi(s, a)^\intercal \mathbf{w}] + \frac{\beta}{2} \mathrm{Var}_P[\Psi_h^\pi(s, a)^\intercal \mathbf{w}] + O(\beta^2) \\ &\approx \boldsymbol{\psi}_h^\pi(s, a)^\intercal \mathbf{w} + \frac{\beta}{2} \mathbf{w}^\intercal \mathrm{Var}_P[\Psi_h^\pi(s, a)] \mathbf{w} = \tilde{\mathcal{Q}}_{h,\beta}^\pi(s, a), \end{aligned} \tag{7}$$

in which $\mathrm{Var}_P[\Psi_h^\pi(s, a)] = \Sigma_h^\pi(s, a)$ is interpreted as a covariance matrix for SFs. In the context of Theorems 1 and 2, the term ε encapsulates the errors in the approximation of $\boldsymbol{\psi}_h^\pi$ and Σ_h^π, plus the terms contained in $O(\beta^2)$ above. However, the main advantage of (7) is that, like (5), it is also analytic in $\mathbf{w}$ and allows for *instantaneous policy evaluation* with arbitrary reward preferences $\mathbf{w}$. Interestingly, (7) also allows instantaneous policy evaluation with respect to different choices of β, making it possible to revise or adapt the level of risk aversion on-demand. In all these cases, $\boldsymbol{\psi}_h^\pi$ and Σ_h^π provide task-independent and portable representations of policies while also accounting for exogenous risk. This is the key to preserving the task generalization ability of SFs in the risk-aware setting, and (7) can now be seen as a particular instantiation of GPE. We call this overall approach *Risk-aware Successor Features* (RaSF).

The simplest approaches for estimating Σ_h^π in the exact (e.g. tabular) Q-learning setting are based on dynamic programming [37, 40], which would allow the overall approach to be easily integrated into existing SF implementations. In particular, the covariance satisfies the Bellman equation

$$\Sigma_h^\pi(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\boldsymbol{\delta}_h \boldsymbol{\delta}_h^\intercal + \Sigma_{h+1}^\pi(s', \pi_{h+1}(s')) \,|\, s_h = s,\, a_h = a \right], \tag{8}$$

where $\boldsymbol{\delta}_h$ are the Bellman residuals of $\boldsymbol{\psi}_h^\pi(s, a)$. The approximation $\tilde{\boldsymbol{\psi}}_h^\pi$ is known to converge to the true value $\boldsymbol{\psi}_h^\pi$, and a similar result also holds for updating the covariance based on (8).

Theorem 3 (Convergence of Covariance). *Let $\| \cdot \|$ be a matrix-compatible norm, and suppose there exists $\varepsilon : \mathcal{S} \times \mathcal{A} \times \mathcal{T} \to [0, \infty)$ such that:*

1. $\|\tilde{\boldsymbol{\psi}}_h^\pi(s,a) - \boldsymbol{\psi}_h^\pi(s,a)\|^2 \leq \varepsilon_h(s,a)$

2. $\|\mathbb{E}_{s' \sim P(\cdot|s,a)}[\tilde{\boldsymbol{\delta}}_h(\tilde{\boldsymbol{\psi}}_h^\pi(s', \pi_{h+1}(s')) - \boldsymbol{\psi}_h^\pi(s', \pi_{h+1}(s')))^\intercal]\| \leq \varepsilon_h(s,a)$.

Then,

$$\left\|\Sigma_h^\pi(s,a) - \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\tilde{\boldsymbol{\delta}}_h \tilde{\boldsymbol{\delta}}_h^\intercal + \tilde{\Sigma}_{h+1}^\pi(s', \pi_{h+1}(s'))\right]\right\| \leq 3\varepsilon_h(s,a).$$

A proof can be found in Appendix B.4. Appendix A.1 describes how $\tilde{\boldsymbol{\psi}}_h^\pi$ and $\tilde{\Sigma}_h^\pi$ can be learned online from environment interactions, while Appendix A.4 discusses further generalizations of (7). There are, however, several limitations of estimating Σ_h^π in this way. First, obtaining accurate estimates of Σ_h^π requires accurate estimates of $\boldsymbol{\psi}_h^\pi$ (thus estimating one quantity on top of another), making this approach difficult to apply with deep function approximation. This claim is further substantiated by Theorem 3 and preliminary experiments. A second issue that occurs is *double sampling*, when the same transitions are used to update the mean and covariance, resulting in accumulation of bias in the latter [1, 47]. Our experiments on the reacher domain mitigate these issues by leveraging distributional RL to approximate (7), while maintaining computational efficiency.

4 Experiments

To evaluate the performance of RaSF, we revisit the benchmark domains in Barreto et al. [2], which have been slightly modified for learning and evaluating risk-aware behaviors. We defer all experimental details to Appendix C.

4.1 Four-Room

Domain Description. The first domain consists of a family of navigation tasks defined on a discrete 2-D space divided into four rooms, as illustrated on the left in Figure 2. The environment has additional objects that can be picked up by the agent by occupying their cells. These objects belong to one of three possible classes, drawn as different shapes in Figure 2, which determine their reward. The position of the objects remains fixed, but the rewards of their classes are reset every 20,000 transitions to random values sampled uniformly in $[-1, +1]$. To incorporate risk, traps are placed in fixed cells marked with X. For every time instant during which the agent occupies a trap cell, the trap activates spontaneously with a small probability, resulting in an immediate penalty and termination of the episode (we refer to this event as a failure). The goal is to maximize the total reward accumulated over 128 random task instances while minimizing the number of failures.

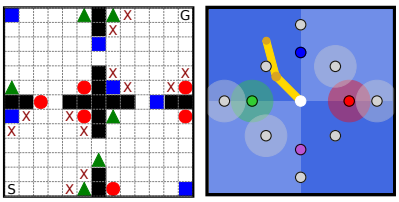

Figure 2: **Four-Room (Left):** the shapes of the objects represent their classes, 'S' is the start state, 'G' is the final goal state, and 'X' is a trap. **Reacher (Right):** colored and gray circles represent training and test targets, respectively, while shaded regions represent areas of high risk.

Baselines. In order to demonstrate the power of our approach in the absence of approximation errors, we define a simple instance of RaSFs in which $\boldsymbol{\psi}^{\pi_i}$ and Σ^{π_i} are learned exactly using lookup tables and dynamic programming (equation (8)). We also apply modest discounting of rewards to ensure that the Q-function converges, as is standard in RL and discussed further in Appendix A.2. The vector $\mathbf{w}$ is also learned using immediate reward feedback and exact, sparse state features $\boldsymbol{\phi}$ provided to the agent. Due to its similarity to standard Q-learning, we call this approach RaSFQL. To provide a challenging baseline for comparison, we implemented another policy reuse algorithm (PRQL) [12]. Further replacing the risk-neutral action selection mechanism of PRQL with *smart exploration* [16] allows PRQL to easily account for return volatility, and we refer to this baseline as RaPRQL.

Main Results. The performance of these algorithms is shown in Figure 3. The cumulative reward obtained by RaSFQL is generally lower than SFQL, as expected since a risk-averse agent should avoid the objects in the bottom-left and top-right rooms and forgo their associated rewards. Interestingly, the performance of RaSFQL far exceeds that of RaPRQL and even PRQL, suggesting that the benefits

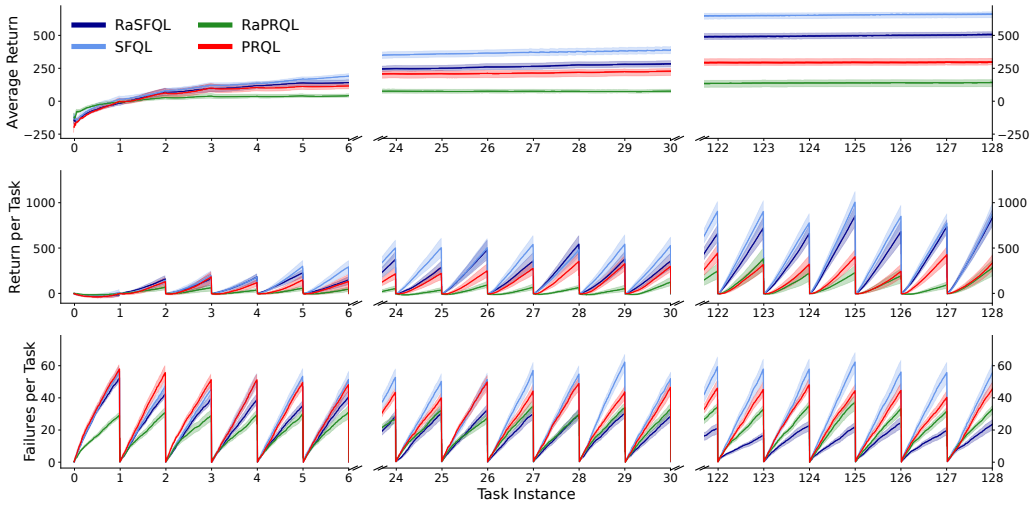

Figure 3: Average return, cumulative return and number of failures per task in the four-room domain, for $\beta = \omega = -2$. Shaded bands show one standard error over 30 independent runs.

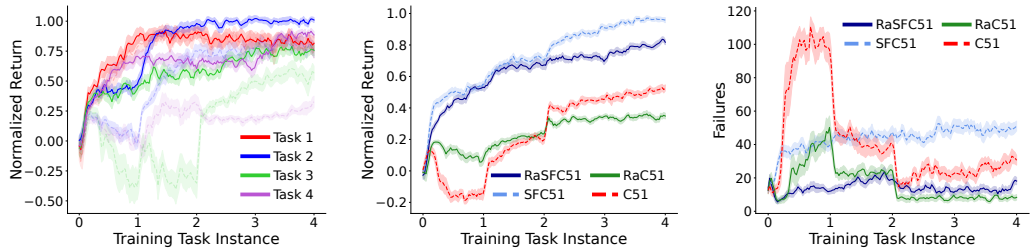

Figure 4: Performance on training and test tasks for the reacher domain, as a function of the number of experiences collected from the training tasks. **Left:** Normalized return on training tasks. Faded curves correspond to C51 performance. **Middle:** Normalized return averaged across all test tasks. **Right:** Total failures across all test tasks. Shaded bands show one standard error over 10 independent runs with different seeds.

of task generalization provided by GPI/GPE are quite strong. Furthermore, the number of failures observed by RaSFQL gradually decreases over the task instances, while the number of failures of SFQL slightly increases. This is consistent with Theorem 1 that guarantees monotone improvement in the *risk-adjusted* return of RaSFQL. On the other hand, while RaPRQL also learns to avoid risk, it fails slightly more often than RaSFQL. This suggests that the benefits of task generalization promised by GPI/GPE even allow risk-aware behaviors to emerge sooner than by using generic policy reuse methods that are unable to exploit the task structure, namely PRQL. This aspect becomes critical for minimizing failures when deploying a trained policy library on novel task instances in a real-world setting. Further analysis and ablation studies are provided in Appendix D.1.

4.2 Reacher

Domain Description. The second domain consists of a set of tasks based on the MuJoCo physics engine [42] that involve the maneuver of a robotic arm toward a fixed target location. As illustrated in the rightmost plot in Figure 2, the agent is only allowed to train on 4 tasks, whose target locations are indicated by colored circles, and must be able to perform well on 8 test tasks whose target locations are indicated by the grey circles. Furthermore, we incorporate two sources of reward volatility: (1) actions are perturbed by additive Gaussian noise; and (2) fixed regions around some of the target locations randomly incur negative rewards (failures), illustrated by faded circles in Figure 2. Please note that most of these high-volatility regions are centered on target locations of test tasks from which

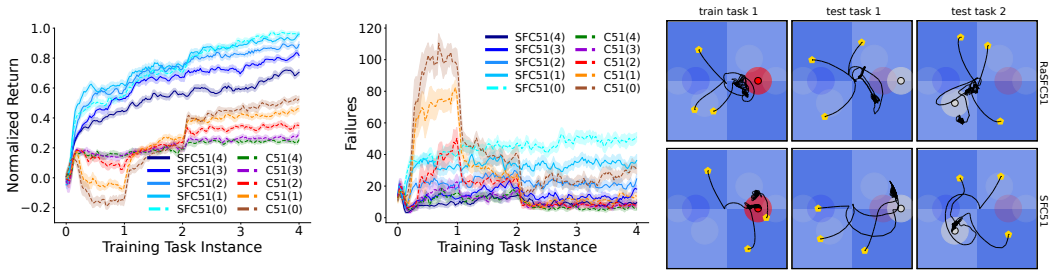

Figure 6: **Left:** Normalized average test return for the reacher domain, for different values of β (legend values indicate the negative values of β). **Middle:** Total failures across all test tasks for various values of β (legend values indicate the negative values of β). **Right:** Evolutions of the arm tip position during three successful rollouts of the reacher domain according to the GPI policy obtained after training with $\beta = -3$ (yellow pentagons indicate the initial states in each rollout). Only one training task and two test tasks are shown. The risk-averse agent learns to hover close to the goal while avoiding the high-volatility shaded regions.

the agent never learns directly. This stresses the agent's ability to avoid unforeseen dangers in the environment, in additional to performing well on previously unseen task instances.

Baselines. As discussed earlier, it is difficult to compute Σ^{π_i} directly using (8). A computationally tractable way to avoid these issues is to first approximate the density of $\Psi^{\pi_i}(s,a)$, and then extract the moments needed to compute (8). Specifically, we apply C51 [6] by modeling $\Psi_1^{\pi_i}(s,a), \dots \Psi_d^{\pi_i}(s,a)$ using histograms for each (s,a). However, Ψ^{π_i} are high-dimensional, so we avoid the curse of dimensionality by modeling the marginals $\Psi_j^{\pi_i}$ rather than their full joint distribution. This still turns out to be an effective way of detecting high-variance scenarios in the environment. The final architecture is illustrated in Figure 5, where the marginal distributions of Ψ^{π_i} are modeled as separate output "heads" with a shared state encoder. The rest of the training protocol is identical to the SFDQN of Barreto et al. [2], except that DQN is replaced by the C51 architecture above, as further detailed in Appendix A.3. The risk-averse and risk-neutral instances of this approach for modeling successor features are referred to as RaSFC51 and SFC51, respectively, while RaC51 and C51 replace successor features with *universal value functions* [35] for generalization across target locations.

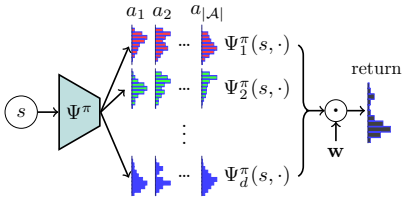

Figure 5: Architecture for $\Psi^\pi(s,a)$.

Main Results. The performance of the algorithms on the reacher domain is illustrated in Figure 4. We first observe that the performance of all four training policies for RaSFC51 improves almost immediately, obtaining returns that exceed 75% of the performance of a *fully trained* C51 agent, an observation that correlates highly with the original SFDQN. Interestingly, RaC51 is unable to achieve satisfactory performance on two out of the four training tasks, even after fully trained on all 4 tasks. Furthermore, the performance of RaSFC51 on the testing tasks far exceeds that of both RaC51 and C51, demonstrating the superior generalization ability of SFs in the risk-aware setting. Finally, the total number of failures across the test task instances is also considerably lower for RaSFC51 than it is for SFC51, and remains low during the entire training horizon. RaC51 and C51 fail frequently at the beginning of training, but less often later in training. While this may suggest that C51 is learning adequate risk-sensitive policies, it is mainly due to the fact that C51 is not able to generalize nearly as well as SFC51 for some of the test target locations, as elaborated in Appendix D.2.

Additional Analysis. We also conducted a sensitivity analysis for the risk-aversion parameter β for each of the algorithms, summarized in Figure 6. The performance of RaSFC51 and RaC51 decays gracefully as the level of risk-aversion β is increased in magnitude, but the performance of RaSFC51 uniformly outperforms RaC51 for *every* value of β tested. The middle plot demonstrates that the number of failures also decreases drastically as β is increased in magnitude, which stabilizies around $\beta = -4$. Furthermore, while C51 and RaC51 generally fail less often than SFC51 and RaSFC51, this

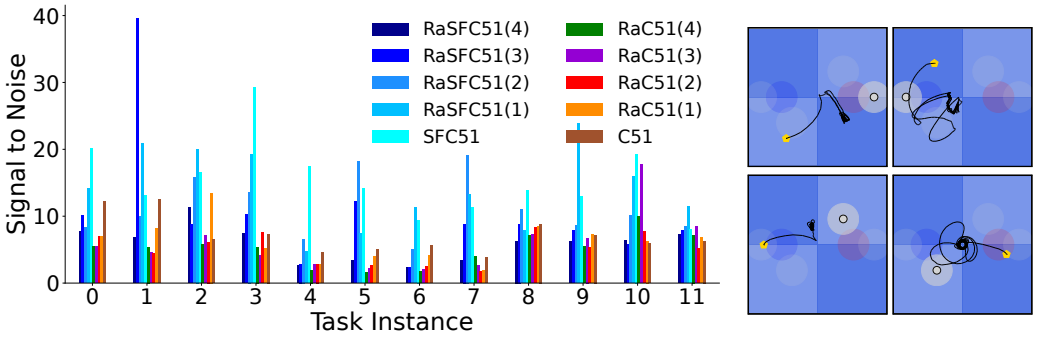

Figure 7: **Left:** Empirical signal-to-noise ratios of the GPI policies at the end of training. **Right:** Examples of worst-case behavior observed by RaSFC51 with $\beta = -3$, which depict the most likely sources of high return variability. While the agent learns to successfully avoid the risky regions of the domain, high return variance between rollouts can sometimes still be accumulated in other ways, such as failure to find a globally optimal policy (top left), inability to generalize the utility precisely on an unseen task for all states (bottom left), policy instability (top right) and divergence in the most extreme cases (bottom right) due to compounding of numerical errors.

is due to their inability to generalize their learned behavior to all novel target instances, as mentioned above. The final plot on the right shows that RaSFC51 indeed learns safer policies than SFC51 on both training and test instances. Specifically, RaSFC51 learns to hover as close to the goal as possible in most cases, while still avoiding the high-volatility shaded regions.

Failure Modes and Limitations. Finally, we compared the empirical signal-to-noise ratios[2] achieved by each baseline on each task using the returns of sampled trajectories at the end of training, and the results are summarized in Figure 7. RaSFC51 and SFC51 soundly outperform RaC51 and C51, and RaSFC51 generally obtains the highest ratios for medium values of $\beta \in \{-3, -2, -1\}$. However, RaSFC51 does not consistently outperform SFC51 on all domains, despite RaSFC51 avoiding the high-risk regions of the domain. We suspect that return variance is accumulated in other ways, some of which can be identified by examining the worst-case behaviors exhibited by the risk-aware GPI policy illustrated in the rightmost plots of Figure 7. However, we believe that these results are expected. Firstly, estimation of only the expected return in an off-policy setting with bootstrapping and deep function approximation is already a difficult problem [44], and addressing it is outside the scope of this work. Secondly, generalizing the empirical distribution learned on one set of tasks to a completely new set of *unseen* tasks – even within the same domain – is an even more challenging problem. We believe that advances in distributional RL that can learn more accurate distributions will strongly benefit our approach, as will methods suited for estimating the joint distributions of correlated returns efficiently.

5 Conclusion

We presented Risk-aware Successor Features (RaSFs) for realizing policy transfer between tasks with shared dynamics, with the goal of maximizing the entropic utility of return. We extended GPI to the risk-aware setting, providing monotone convergence and optimality guarantees, assuming that the utility of source policies can be evaluated. To facilitate policy evaluation, we also extended the notion of GPE to the risk-aware setting. Together, risk-aware GPI and GPE inherit the superior task generalization abilities of successor features, while learning to avoid dangerous high-volatility regions. More generally, incorporating risk and safety in sequential decision-making is a complex problem. The entropic utility objective does not capture tail risk nor other properties of the return distribution, which could be a challenging but powerful extension of successor features.

[2]This is computed as $\hat{\mu}/\hat{\sigma}$, where $\hat{\mu}$ is the estimated return and $\hat{\sigma}$ is the estimated standard deviation of the return accumulated across multiple rollouts. We use the signal-to-noise ratio because it is independent of β and allows consistent comparison across baselines.

Acknowledgements

The authors would like to thank Daniel Mankowitz for suggesting relevant research in the area of robust and risk-aware reinforcement learning and for providing insightful comments throughout the development of the paper.

Funding Transparency Statement

Michael Gimelfarb was funded by an Ontario Graduate Scholarship, a DiDi Graduate Student Award and a Vector Institute Postgraduate Affiliate Award, Scott Sanner was funded by an Ontario Early Researcher Award Grant, and Chi-Guhn Lee was funded by an NSERC Discovery Grant.

References

[1] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

[2] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, David Silver, and Hado P van Hasselt. Successor features for transfer in reinforcement learning. In *NeurIPS*, 2017.

[3] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *ICML*, pages 501–510, 2018.

[4] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *PNAS*, 117(48):30079–30087, 2020.

[5] Nicole Bäuerle and Ulrich Rieder. More risk-sensitive markov decision processes. *Mathematics of Operations Research*, 39(1):105–120, 2014.

[6] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *ICML*, pages 449–458, 2017.

[7] L Bisi, L Sabbioni, E Vittori, M Papini, and M Restelli. Risk-averse trust region optimization for reward-volatility reduction. In *IJCAI*, pages 4583–4589, 2020.

[8] Diana Borsa, Andre Barreto, John Quan, Daniel J Mankowitz, Hado van Hasselt, Remi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *ICLR*, 2018.

[9] Oscar Dowson, David P Morton, and Bernardo K Pagnoncelli. Multistage stochastic programs with the entropic risk measure, 2020.

[10] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *ICML Workshop RL4RealLife*, 2019.

[11] Yingjie Fei, Zhuoran Yang, Yudong Chen, Zhaoran Wang, and Qiaomin Xie. Risk-sensitive reinforcement learning: Near-optimal risk-sample tradeoff in regret. In *NeurIPS*, volume 33, 2020.

[12] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *AAMAS*, pages 720–727, 2006.

[13] Hans Föllmer and Alexander Schied. Convex measures of risk and trading constraints. *Finance and stochastics*, 6(4):429–447, 2002.

[14] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[15] Javier García and Fernando Fernández. Probabilistic policy reuse for safe reinforcement learning. *ACM Trans. Auton. Adapt. Syst.*, 13(3), March 2019. ISSN 1556-4665. doi: 10.1145/3310090.

[16] Clement Gehring and Doina Precup. Smart exploration in reinforcement learning using absolute temporal difference errors. In *AAMAS*, pages 1037–1044, 2013.

[17] David Held, Zoe McCarthy, Michael Zhang, Fred Shentu, and Pieter Abbeel. Probabilistically safe policy transfer. In *ICRA*, pages 5798–5805. IEEE, 2017.

[18] Takuya Hiraoka, Takahisa Imagawa, Tatsuya Mori, Takashi Onishi, and Yoshimasa Tsuruoka. Learning robust options by conditional value at risk optimization. In *NeurIPS*, pages 2619–2629, 2019.

[19] Arushi Jain, Khimya Khetarpal, and Doina Precup. Safe option-critic: learning safety in the option-critic architecture. *The Knowledge Engineering Review*, 36, 2021.

[20] Arushi Jain, Gandharv Patil, Ayush Jain, Khimya Khetarpal, and Doina Precup. Variance penalized on-policy and off-policy actor-critic. *arXiv preprint arXiv:2102.01985*, 2021.

[21] David Janz, Jiri Hron, Przemysław Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschiatschek. Successor uncertainties: exploration and uncertainty in temporal difference learning. In *NeurIPS*, 2019.

[22] Michael Kupper and Walter Schachermayer. Representation results for law invariant time consistent functions. *Mathematics and Financial Economics*, 2(3):189–210, 2009.

[23] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[24] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review. *and Perspectives on Open Problems*, 2020.

[25] Daniel Mankowitz, Timothy Mann, Pierre-Luc Bacon, Doina Precup, and Shie Mannor. Learning robust options. In *AAAI*, volume 32, 2018.

[26] Daniel J Mankowitz, Aviv Tamar, and Shie Mannor. Situational awareness by risk-conscious skills. *arXiv preprint arXiv:1610.02847*, 2016.

[27] Shie Mannor and John N Tsitsiklis. Algorithmic aspects of mean–variance optimization in markov decision processes. *European Journal of Operational Research*, 231(3):645–653, 2013.

[28] Hongzi Mao, Shaileshh Bojja Venkatakrishnan, Malte Schwarzkopf, and Mohammad Alizadeh. Variance reduction for reinforcement learning in input-driven environments. In *ICLR*, 2019.

[29] Teodor Mihai Moldovan and Pieter Abbeel. Risk aversion in markov decision processes via near optimal chernoff bounds. In *NeurIPS*, pages 3140–3148, 2012.

[30] David Nass, Boris Belousov, and Jan Peters. Entropic risk measure in policy search. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1101–1106. IEEE, 2019.

[31] Mark Nemecek and Ronald Parr. Policy caches with successor features. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 8025–8033. PMLR, 18–24 Jul 2021.

[32] Takayuki Osogami. Robustness and risk-sensitivity in markov decision processes. In *NeurIPS*, volume 25, pages 233–241, 2012.

[33] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[34] Andrzej Ruszczyński. Risk-averse dynamic programming for markov decision processes. *Mathematical programming*, 125(2):235–261, 2010.

[35] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *ICML*, pages 1312–1320. PMLR, 2015.

[36] Yun Shen, Michael J Tobia, Tobias Sommer, and Klaus Obermayer. Risk-sensitive reinforcement learning. *Neural computation*, 26(7):1298–1328, 2014.

[37] Craig Sherstan, Dylan R Ashley, Brendan Bennett, Kenny Young, Adam White, Martha White, and Richard S Sutton. Comparing direct and indirect temporal-difference methods for estimating the variance of the return. In *UAI*, 2018.

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

[39] Aviv Tamar, Dotan Di Castro, and Shie Mannor. Policy gradients with variance related risk criteria. In *ICML*, pages 1651–1658, 2012.

[40] Aviv Tamar, Dotan Di Castro, and Shie Mannor. Learning the variance of the reward-to-go. *The Journal of Machine Learning Research*, 17(1):361–396, 2016.

[41] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

[42] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[43] Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. Safe reinforcement learning via curriculum induction. In *NeurIPS*, volume 33, 2020.

[44] Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

[45] S Whiteson, S Zhang, and B Liu. Mean- variance policy iteration for risk- averse reinforcement learning. In *AAAI*. Association for the Advancement of Artificial Intelligence, 2021.

[46] Yang Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743, 2018.

[47] Yuhua Zhu and Lexing Ying. Borrowing from the future: An attempt to address double sampling. In *Mathematical and scientific machine learning*, pages 246–268. PMLR, 2020.

Risk-Aware Transfer in Reinforcement Learning using Successor Features

Supplementary Material

Abstract

This part of the paper discusses algorithmic details for the total reward and the discounted setting that were not included in the main paper due to space limitations. It includes proofs of all main theoretical claims, as well as all domain configurations and parameter settings that are required to reproduce the experiments. Finally, it includes additional experiments and ablation studies that had to be excluded from the main paper due to space limitations, and the NeurIPS paper checklist.

Contents

A Mathematical and Algorithmic Details

In this section, we outline the ways in which successor features and their covariance matrices can be learned in practical RL settings. Specifically, we discuss the details surrounding Bellman updates and the distributional RL framework, and also discuss the mean-variance approximation under more general assumptions in parameteric density estimation.

A.1 Mean-Variance Approximation for Episodic MDPs

Directly Computing Successor Features and Covariances. In the tabular setting, we compute estimates of the mean $\tilde{\boldsymbol{\psi}}_h^\pi(s,a)$ and covariance $\tilde{\Sigma}_h^\pi(s,a)$ by extending the analysis in Sherstan et al. [37] to the d-dimensional setting. For a transition $(s, a, \boldsymbol{\phi}, s', a')$, where $\boldsymbol{\phi} = \boldsymbol{\phi}(s,a,s')$, and learning rate $\alpha > 0$, the update for successor features is:

$$\begin{aligned} \tilde{\boldsymbol{\delta}}_h &= \boldsymbol{\phi} + \tilde{\boldsymbol{\psi}}_{h+1}^\pi(s',a') - \tilde{\boldsymbol{\psi}}_h^\pi(s,a), \\ \tilde{\boldsymbol{\psi}}_h^\pi(s,a) &= \tilde{\boldsymbol{\psi}}_h^\pi(s,a) + \alpha\tilde{\boldsymbol{\delta}}_h. \end{aligned} \tag{9}$$

By virtue of the covariance Bellman equation (8), a per-sample update of the covariance matrix can be computed by using the Bellman residuals $\tilde{\boldsymbol{\delta}}_h$ as a pseudo-reward:

$$\begin{aligned} \Delta_h &= \tilde{\boldsymbol{\delta}}_h\tilde{\boldsymbol{\delta}}_h^\intercal + \tilde{\Sigma}_{h+1}^\pi(s',a') - \tilde{\Sigma}_h^\pi(s,a), \\ \tilde{\Sigma}_h^\pi(s,a) &= \tilde{\Sigma}_h^\pi(s,a) + \bar{\alpha}\Delta_h. \end{aligned} \tag{10}$$

In practice, $\bar{\alpha}$ is usually set much smaller than α, e.g. $\bar{\alpha} = \rho\alpha$ for some positive $\rho \ll 1$.

Computing Reward Parameters. When the reward parameters $\mathbf{w}$ are unknown, they can be learned by solving a regression problem. Specifically, for known or estimated features $\boldsymbol{\phi}(s,a,s')$ and an observed reward r, the objective function to minimize is the *mean-squared error*,

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\left(r - \boldsymbol{\phi}(s,a,s')^\intercal\mathbf{w}\right)^2,$$

that can be minimized using *stochastic gradient descent* (SGD). Introducing a learning rate $\alpha_w > 0$, an update of SGD on a single transition $(s,a) \to s'$ is

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_w(r - \boldsymbol{\phi}(s,a,s')^\intercal\mathbf{w})\boldsymbol{\phi}(s,a,s').$$

Pseudocode. The general routine for performing risk-aware transfer learning in an online setting, which we call *Risk-Aware Successor Feature Q-Learning* (RaSFQL), can now be fully described. Pseudocode adapted for the total-reward episodic MDP setting is given as Algorithm 1. Please note that our approach closely follows the risk-neutral SFQL in Barreto et al. [2].

Both the discounted and total reward episodic settings are amenable to function approximation. However, as discussed in the main text, this "residual" method is usually not advisable as the approximation errors in the residuals $\tilde{\boldsymbol{\delta}}_h$ can dominate the environment uncertainty. While this could be useful for handling *epistemic* or model uncertainty in successor features [21], the intent of our work is to learn the *aleatory* or environment uncertainty. Therefore, a more precise method — based on the projected Bellman equation — will be introduced in Appendix A.3.

A.2 Mean-Variance Approximation for Discounted MDPs

Bellman Principle and Augmented MDP. The utility objective in the discounted infinite-horizon setting becomes

$$\mathcal{Q}_{\beta,\gamma}^\pi(s,a) = U_\beta\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t,s_{t+1})\right],$$

where $\gamma \in (0,1)$ is a discount factor for future rewards.

In the discounted setting, it is necessary to accumulate and keep track of the discounting over time. This can be implemented by augmenting the state space of the original MDP [5]. Specifically, define

Algorithm 1 RaSFQL with Mean-Variance Approximation

Requires $m, T, N_e \in \mathbb{N}$, $\varepsilon \in [0,1]$, $\alpha, \bar{\alpha}, \alpha_w > 0$, $\beta \in \mathbb{R}$, $\boldsymbol{\phi} \in \mathbb{R}^d$, $M_1, \dots M_m \in \mathcal{M}$
for $t = 1, 2 \dots m$ **do**
  \\ Initialize successor features and covariance for the current task
  if $t = 1$ **then** Initialize $\tilde{\boldsymbol{\psi}}^t, \tilde{\Sigma}^t$ to small random values **else** Initialize $\tilde{\boldsymbol{\psi}}^t, \tilde{\Sigma}^t$ to $\tilde{\boldsymbol{\psi}}^{t-1}, \tilde{\Sigma}^{t-1}$
  Initialize $\tilde{\mathbf{w}}_t$ to small random values
  \\ Commence training on task M_t
  for $n_e = 1, 2 \dots N_e$ **do**
    Initialize task M_t with initial state s
    for $h = 0, 1 \dots T$ **do**
      \\ Select the source task M_c using GPI
      $c \leftarrow \arg\max_j \max_b \{\tilde{\boldsymbol{\psi}}_h^j(s,b)^\intercal \tilde{\mathbf{w}}_t - \beta \tilde{\mathbf{w}}_t^\intercal \tilde{\Sigma}_h^j(s,b) \tilde{\mathbf{w}}_t\}$
      \\ Sample action from the epsilon-greedy policy based on π_c
      $\text{random\_a} \sim \text{Bernoulli}(\varepsilon)$
      if random_a **then** $a \sim \text{Uniform}(\mathcal{A})$ **else**
      $a \leftarrow \arg\max_b \{\tilde{\boldsymbol{\psi}}_h^c(s,b)^\intercal \tilde{\mathbf{w}}_t - \beta \tilde{\mathbf{w}}_t^\intercal \tilde{\Sigma}_h^c(s,b) \tilde{\mathbf{w}}_t\}$
      Take action a in M_t and observe r and s'
      \\ Update reward parameters for the current task
      $\tilde{\mathbf{w}}_t \leftarrow \tilde{\mathbf{w}}_t + \alpha_w (r - \boldsymbol{\phi}(s,a,s')^\intercal \tilde{\mathbf{w}}_t) \boldsymbol{\phi}(s,a,s')$
      \\ Update the successor features and covariance for the current task
      $a' \leftarrow \arg\max_b \max_j \{\tilde{\boldsymbol{\psi}}_h^j(s',b)^\intercal \tilde{\mathbf{w}}_t - \beta \tilde{\mathbf{w}}_t^\intercal \tilde{\Sigma}_h^j(s',b) \tilde{\mathbf{w}}_t\}$
      Update $\tilde{\boldsymbol{\psi}}_h^t, \tilde{\Sigma}_h^t$ on $(s, a, \boldsymbol{\phi}, s', a')$ using (9) and (10)
      \\ Update the successor features and covariance for task M_c
      if $c \neq t$ **then**
        $a' \leftarrow \arg\max_b \{\tilde{\boldsymbol{\psi}}_h^c(s',b)^\intercal \tilde{\mathbf{w}}_c - \beta \tilde{\mathbf{w}}_c^\intercal \tilde{\Sigma}_h^c(s',b) \tilde{\mathbf{w}}_c\}$
        Update $\tilde{\boldsymbol{\psi}}_h^c, \tilde{\Sigma}_h^c$ on $(s, a, \boldsymbol{\phi}, s', a')$ using (9) and (10)
      end if
      $s \leftarrow s'$
    end for
  end for
end for

$\mathcal{Z} = [0,1]$ and let $z \in \mathcal{Z}$ denote the state of discounting. For a given MDP $\langle \mathcal{S}, \mathcal{A}, r, P \rangle$, we define the augmented MDP $\langle \mathcal{S}', \mathcal{A}, r', P' \rangle$, with state space $\mathcal{S}' = \mathcal{S} \times \mathcal{Z}$, action space $\mathcal{A}$, reward function

$$r'((s,z), a, (s',z')) = z r(s,a,s'),$$

and dynamics

$$P'((s',z')|(s,z),a) = P(s'|s,a)\delta_{\gamma z}(z'),$$

where δ is the Dirac delta function. Applying this augmentation transformation to a set of MDPs with common transition function and common discount factor implies that the set of augmented MDPs will also have the same transition functions.

Moreover, the following Bellman equation can be derived for the augmented MDP [5]:

$$\begin{aligned} \mathcal{J}_\beta^\pi(s,a,z) &= U_\beta \left[z r(s,a,s') + \mathcal{J}_\beta^\pi(s', \pi(s', \gamma z), \gamma z) \right] \\ &= \frac{1}{\beta} \log \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\exp \left\{ \beta \left(z r(s,a,s') + \mathcal{J}_\beta^\pi(s', \pi(s', \gamma z), \gamma z) \right) \right\} \right]. \end{aligned} \tag{11}$$

Then, we can recover the original utility with $\mathcal{Q}_{\beta,\gamma}^\pi(s,a) = \mathcal{J}_\beta^\pi(s,a,1)$. Furthermore, the Bellman equation above converges to a unique fixed point, and so the search for optimal policies can be restricted to stationary Markov policies $\pi : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$.

However, learning general policies $\pi : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$ introduces additional difficulties in the function approximation setting. In this case, successor features and their covariance matrices would have to be functions of z. For a single transition, their corresponding updates would also require a sweep over all possible values of z, e.g. $z = 1, \gamma, \gamma^2, \dots$, and would be computationally demanding. On the other hand, restricting the search to stationary policies $\pi : \mathcal{S} \to \mathcal{A}$ alleviates this computational

burden, making the overall time and space complexity per update comparable to the risk-neutral SF representation, and also allows off-the-shelf RL algorithms to be used to learn successor features. This also facilitates more precise estimation of risk using the distributional framework discussed in the next section.

Fortunately, the restriction to z-independent source policies does not affect the validity of Theorem 1, since policy improvement was shown for *arbitrary* admissible policies. This implies that monotone policy improvement is guaranteed even for z-independent policies, provided that their utilities can be estimated. In the case of Theorem 2, the approximation error ε generally arises from two sources of additive error, namely that of restricting optimal policies π_i^* to z-independent optimal policies $\bar{\pi}_i^*$, and that of approximating utilities using function approximation, e.g.

$$\begin{aligned}
\varepsilon &= \left|\tilde{\mathcal{J}}_\beta^{\bar{\pi}_i^*}(s,a,1) - \mathcal{J}_\beta^{\pi_i^*}(s,a,1)\right| \\
&= \left|\tilde{\mathcal{J}}_\beta^{\bar{\pi}_i^*}(s,a,1) - \mathcal{J}_\beta^{\bar{\pi}_i^*}(s,a,1) + \mathcal{J}_\beta^{\bar{\pi}_i^*}(s,a,1) - \mathcal{J}_\beta^{\pi_i^*}(s,a,1)\right| \\
&\leq \left|\tilde{\mathcal{J}}_\beta^{\bar{\pi}_i^*}(s,a,1) - \mathcal{J}_\beta^{\bar{\pi}_i^*}(s,a,1)\right| + \left|\mathcal{J}_\beta^{\bar{\pi}_i^*}(s,a,1) - \mathcal{J}_\beta^{\pi_i^*}(s,a,1)\right| \\
&= \left\{\text{approximation error of } \mathcal{J}_\beta^{\bar{\pi}_i^*}\right\} + \left\{\text{absolute difference between utilities of } \bar{\pi}_i^* \text{ and } \pi_i^*\right\}.
\end{aligned}$$

The first source of error arises solely due to the method of function approximation, and can be reduced by using architectures whose training parameters and capacity are well-calibrated for each problem. The second source of error is in general irreducible, but whether it can be tolerated should be traded-off against the difficulty of learning z-dependent policies. In general, the learning of z-dependent policies tractably is a challenging problem, which we leave for future investigation.

Incorporating Moment Information into GPE in Discounted MDPs. We now apply the idea of generalized policy evaluation to discounted objectives. First, observe that for fixed $\pi : \mathcal{S} \to \mathcal{A}$:

$$\mathcal{J}_\beta^\pi(s,a,1) = U_\beta\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t), s_{t+1})\right] = U_\beta\left[\Psi^\pi(s,a)^\intercal \mathbf{w}\right], \tag{12}$$

corresponding to the random vector $\Psi^\pi(s,a) = \sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}_t$ of unrealized feature returns at time h. Thus, we have again transformed the problem of estimating the utility of rewards into the problem of estimating the moments of the random variable $\Psi^\pi(s,a)^\intercal \mathbf{w}$.

Next, computing the Taylor expansion of U_β:

$$\begin{aligned}
\mathcal{J}_\beta^\pi(s,a,1) &= \mathbb{E}_P[\Psi^\pi(s,a)^\intercal \mathbf{w}] + \frac{\beta}{2}\mathrm{Var}_P[\Psi^\pi(s,a)^\intercal \mathbf{w}] + O(\beta^2) \\
&\approx \boldsymbol{\psi}^\pi(s,a)^\intercal \mathbf{w} + \frac{\beta}{2}\mathbf{w}^\intercal \mathrm{Var}_P[\Psi^\pi(s,a)]\mathbf{w} = \tilde{\mathcal{J}}_\beta^\pi(s,a,1).
\end{aligned} \tag{13}$$

From a practical point of view, the mean-variance approximation in the discounted setting is identical to the episodic total-reward setting, with the exception that the successor features and covariance are discounted (and also time-independent). As in the undiscounted case, (13) induces an error of $O(\beta^2)$, but is now another instantiation of GPE. However, restricting the search to z-independent policies introduces additional approximation error that can also be absorbed into ε, as discussed previously. Crucially, the theoretical results proved for the discounted setting (Appendix B.3) will now also hold for z-independent stationary policies.

Bellman Updates for Covariance in Discounted MDPs. The covariance matrix satisfies the covariance Bellman equation

$$\Sigma_h^\pi(s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\boldsymbol{\delta}_h \boldsymbol{\delta}_h^\intercal + \gamma^2 \Sigma_{h+1}^\pi(s', \pi_{h+1}(s')) \,|\, s_h = s,\, a_h = a\right], \tag{14}$$

Similar to (9) and (10), in the discounted setting the successor features can be computed as [2]:

$$\begin{aligned}
\tilde{\boldsymbol{\delta}}_h &= \boldsymbol{\phi} + \gamma \tilde{\boldsymbol{\psi}}_{h+1}^\pi(s',a') - \tilde{\boldsymbol{\psi}}_h^\pi(s,a), \\
\tilde{\boldsymbol{\psi}}_h^\pi(s,a) &= \tilde{\boldsymbol{\psi}}_h^\pi(s,a) + \alpha \tilde{\boldsymbol{\delta}}_h.
\end{aligned} \tag{15}$$

Once again, the covariance matrix can be updated per sample following (14):

$$\begin{aligned}\Delta_h &= \tilde{\boldsymbol{\delta}}_h \tilde{\boldsymbol{\delta}}_h^\intercal + \gamma^2 \tilde{\Sigma}_{h+1}^\pi(s', a') - \tilde{\Sigma}_h^\pi(s, a), \\ \tilde{\Sigma}_h^\pi(s, a) &= \tilde{\Sigma}_h^\pi(s, a) + \bar{\alpha}\Delta_h.\end{aligned} \tag{16}$$

In the context of Algorithm 1, all calls to (9) and (10) would be replaced with (15) and (16), respectively.

The convergence of the covariance matrix in the discounted setting (14) is established in the following result that can be easily proved using the techniques in Appendix B.4 for the episodic setting.

Theorem 4 (Convergence of Covariance). *Let $\|\cdot\|$ be a matrix-compatible norm, and suppose there exists $\varepsilon : \mathcal{S} \times \mathcal{A} \times \mathcal{T} \to [0, \infty)$ such that*

1. $\|\tilde{\boldsymbol{\psi}}_h^\pi(s, a) - \boldsymbol{\psi}_h^\pi(s, a)\|^2 \leq \varepsilon_h(s, a)$
2. $\|\mathbb{E}_{s' \sim P(\cdot|s,a)}[\gamma \tilde{\boldsymbol{\delta}}_h (\tilde{\boldsymbol{\psi}}_h^\pi(s', \pi_{h+1}(s')) - \boldsymbol{\psi}_h^\pi(s', \pi_{h+1}(s')))^\intercal]\| \leq \varepsilon_h(s, a)$.

Then,

$$\left\| \Sigma_h^\pi(s, a) - \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\tilde{\boldsymbol{\delta}}_h \tilde{\boldsymbol{\delta}}_h^\intercal + \gamma^2 \tilde{\Sigma}_{h+1}^\pi(s', \pi_{h+1}(s')) \right] \right\| \leq 3\varepsilon_h(s, a).$$

Please note that this result is identical to Theorem 3, with the exception of the discount factor.

A.3 Histogram Representations for Successor Features

The theoretical framework for distributional RL is discussed in details in the relevant literature [6]. In this appendix, we discuss how this framework can be applied to learn distributions over successor features, and how to use these distributions to select actions in a risk-aware manner.

Learning Distributions over Successor Features. As discussed in the main text, the goal is to estimate the distribution of each component in the discounted infinite horizon setting

$$\Psi_i^\pi(s, a) = \sum_{t=0}^{\infty} \gamma^t \phi_i(s_t, a_t, s_{t+1}),$$

starting from $s_0 = s$, $a_0 = a$, where $a_t = \pi(s_t)$ is selected according to a policy π. Treating $\Psi_1^\pi, \dots \Psi_d^\pi$ as value functions, we are now able to apply distributional RL.

Specifically, suppose that each state feature component is bounded in a compact interval, e.g. $\phi_i(s, a, s') \in [\phi_i^{min}, \phi_i^{max}]$. Then, we may define corresponding bounds on $\Psi_i^\pi(s, a)$ by bounding the terms of its geometric series representation above:

$$\Psi_i^{min} = \frac{\phi_i^{min}}{1 - \gamma} \leq \Psi_i^\pi(s, a) \leq \frac{\phi_i^{max}}{1 - \gamma} = \Psi_i^{max}.$$

Now, we may model each $\Psi_h^\pi(s, a)$ by using a discrete distribution parameterized by $N \in \mathbb{N}$ and $[\Psi_i^{min}, \Psi_i^{max}]$, whose support is defined by a set of atoms

$$\mathcal{Z}_i = \left\{ z_{j,i} = \Psi_i^{min} + j\Delta z_i : 0 \leq j < N \right\}, \Delta z_i = \frac{\Psi_i^{max} - \Psi_i^{min}}{N - 1}, \forall i = 1, \dots d.$$

Finally, the atom probabilities for $z_{j,i}$ are given by a parameteric model $\theta_{j,i} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^N$, e.g.

$$Z_{\theta,i}(s, a) = z_{j,i} \quad \text{w.p. } p_{j,i}(s, a) = \frac{e^{\theta_{j,i}(s,a)}}{\sum_j e^{\theta_{j,i}(s,a)}}, \tag{17}$$

where the softmax layer ensures that probabilities are non-negative and sum to one.

In order to update $p_{j,i}$ on environment transitions (s, a, ϕ_i, s'), we project the Bellman updates for each i onto the support of $\mathcal{Z}_i$. To do this, given a sample (s, a, ϕ_i, s'), we compute the projected Bellman update, clipped to the interval $[\Psi_i^{min}, \Psi_i^{max}]$

$$\hat{\mathcal{T}}_i z_{j,i} = \text{clip}\left(\phi_i + \gamma z_{j,i}; [\Psi_i^{min}, \Psi_i^{max}]\right),$$

and then distribute its probability $p_{j,i}(s', \pi(s'))$ to the immediate neighbors of $\hat{\mathcal{T}}_i z_{j,i}$. Here, we again follow Bellemare et al. [6] and define the projected operator Φ with j-th component equal to

$$(\Phi \hat{\mathcal{T}}_i Z_{\theta,i}(s,a))_j = \sum_{k=0}^{N-1} \text{clip}\left(1 - \frac{\left|\hat{\mathcal{T}}_i z_{k,i} - z_{j,i}\right|}{\Delta z_i}; [0,1]\right) p_{k,i}(s', \pi(s')).$$

As standard in deep RL, we view the target distribution $p_{k,i}(s', \pi(s'))$ as parameterized by a set of frozen parameters θ'. Then, the loss function to optimize for the sample (s, a, ϕ_i, s') is given as the cross-entropy term

$$\mathcal{L}_i(\theta) = D_{KL}\left(\Phi \hat{\mathcal{T}}_i Z_{\theta',i}(s,a) \,\middle\|\, Z_{\theta,i}(s,a)\right),$$

that can be easily optimized using gradient descent.

Calculating Utilities. The calculation of (7) is a trivial matter given the distribution (17). In particular, we have:

$$\mathbb{E}[Z_{\theta,i}(s,a)^p] = \sum_{j=0}^{N-1} (z_{j,i})^p p_{j,i}(s,a), \quad p \in \mathbb{N}, \tag{18}$$

from which we can easily compute the variance

$$\text{Var}[\Psi_i^\pi(s,a)] = \text{Var}[Z_{\theta,i}(s,a)] = \text{E}[Z_{\theta,i}(s,a)^2] - \mathbb{E}[Z_{\theta,i}(s,a)]^2. \tag{19}$$

Recall that by the independence assumption, the cross-covariance terms are ignored in these calculations, and thus $\Sigma_i^\pi(s,a)$ is represented as a diagonal matrix with entries on the i-th diagonal term equal to $\text{Var}[\Psi_i^\pi(s,a)]$.

Another possibility is to compute the entropic utility U_β exactly. In particular, using the independence assumption of $\Psi_i^\pi(s,a)$ again, we have:

$$\begin{aligned} U_\beta[\Psi^\pi(s,a)^\intercal \mathbf{w}] &= \frac{1}{\beta} \log \mathbb{E}\left[e^{\beta \Psi^\pi(s,a)^\intercal \mathbf{w}}\right] \approx \sum_{i=1}^{d} \frac{1}{\beta} \log \mathbb{E}\left[e^{\beta \Psi_i^\pi(s,a) w_i}\right] \\ &= \sum_{i=1,\, w_i \neq 0}^{d} w_i \frac{1}{\beta w_i} \log \mathbb{E}\left[e^{(\beta w_i)\Psi_i^\pi(s,a)}\right] = \sum_{i=1}^{d} w_i U_{\beta w_i}[\Psi_i^\pi(s,a)], \end{aligned} \tag{20}$$

and can be seen as another risk-sensitive instantiation of GPE. Crucially, the utility terms in (20) can be calculated efficiently in the C51 framework using (17)

$$U_\beta[\Psi_i^\pi(s,a)] = \frac{1}{\beta} \log \mathbb{E}\left[e^{\beta Z_{\theta,i}(s,a)}\right] = \frac{1}{\beta} \log \sum_{j=0}^{N-1} e^{\beta z_{j,i}} p_{j,i}(s,a).$$

However, this quantity is difficult to compute numerically, since for negative β, the terms $e^{\beta z_{j,i}}$ often suffer from overflow at $z_{j,i}$ close to Ψ_i^{min}, and underflow for $z_{j,i}$ close to Ψ_i^{max}. This becomes considerably more problematic for β of larger magnitude, such as when risk-awareness is a priority, or for rewards $\mathbf{w}$ of larger magnitude. We also find that the log-sum-exp trick, a standard computational device used for calculations of this form, offers relatively little improvement. A similar issue has also been previously pointed out in other work using the entropic utility [52]. For this reason, we use the mean-variance approximation, which provides an excellent approximation to the entropic utility for various values of β, as we demonstrated experimentally, and without suffering from the aforementioned issues above.

Pseudocode. The approach described above can be applied to compute the distribution of successor features for every component $i = 1, \ldots d$ across all training task instances. This results in a new algorithm that we call SFC51. Generally, the training procedure of SFC51 is identical in structure to SFDQN in Barreto et al. [2], except the deterministic DQN update of successor features [55] is replaced by the distributional C51 update described above. Therefore, the overall training procedure is similar to Algorithm 1, but with a few subtle differences. First, instead of learning $\mathbf{w}$, it is provided to the agent as done in SFDQN. Second, every sample $(s, a, \boldsymbol{\phi}, s')$ collected from any training is

Algorithm 2 RaSFC51 with Mean-Variance Approximation

1: **Requires** $m, T, N, N_e \in \mathbb{N}$, $\varepsilon \in [0,1]$, $\beta, \phi_1^{min}, \phi_1^{max}, \dots \phi_d^{min}, \phi_d^{max} \in \mathbb{R}$, $\boldsymbol{\phi} \in \mathbb{R}^d$, $\gamma \in (0,1)$, $M_1, \dots M_m \in \mathcal{M}$ with $\mathbf{w}_1, \dots \mathbf{w}_m \in \mathbb{R}^d$
\\ Initialize atoms and their probability distributions
2: Initialize $\boldsymbol{\theta}^1(s,a), \dots \boldsymbol{\theta}^m(s,a)$ to random values
3: **for** $i = 1, 2 \dots d$ **do** $\Psi_i^{min} \leftarrow \frac{\phi_i^{min}}{1-\gamma}$, $\Psi_i^{max} \leftarrow \frac{\phi_i^{max}}{1-\gamma}$, $\Delta z_i \leftarrow \frac{\Psi_i^{max} - \Psi_i^{min}}{N-1}$
4: **for** $i = 1, 2 \dots d$ **do for** $j = 0, 1 \dots N-1$ **do** $z_{j,i} \leftarrow \Psi_i^{min} + j\Delta z_i$
\\ Main training loop
5: **for** $t = 1, 2 \dots m$ **do**
\\ Commence training on task M_t
6: **for** $n_e = 1, 2 \dots N_e$ **do**
7: Initialize task M_t with initial state s
8: **for** $h = 0, 1 \dots T$ **do**
\\ Extract sufficient statistics from $\boldsymbol{\theta}^t(s,\cdot)$ and select the source task M_c using GPI
9: **for** $j = 1, 2 \dots m$ **do** Compute $\tilde{\boldsymbol{\psi}}^j(s,\cdot)$, $\tilde{\Sigma}^j(s,\cdot)$ using $\boldsymbol{\theta}^j(s,\cdot)$ and (18) and (19)
10: $c \leftarrow \arg\max_j \max_b \{\tilde{\boldsymbol{\psi}}^j(s,b)^\intercal \mathbf{w}_t - \beta \mathbf{w}_t{}^\intercal \tilde{\Sigma}^j(s,b)\mathbf{w}_t\}$
\\ Sample action from the epsilon-greedy policy based on π_c
11: random_a $\sim$ Bernoulli(ε)
12: **if** random_a **then** $a \sim$ Uniform($\mathcal{A}$) **else**
$a \leftarrow \arg\max_b \{\tilde{\boldsymbol{\psi}}^c(s,b)^\intercal \mathbf{w}_t - \beta \mathbf{w}_t{}^\intercal \tilde{\Sigma}^c(s,b)\mathbf{w}_t\}$
13: Take action a in M_t and observe r and s'
\\ Update $\boldsymbol{\theta}^1(s,a), \dots \boldsymbol{\theta}^m(s,a)$
14: **for** $c = 1, 2 \dots m$ **do**
\\ Extract sufficient statistics from $\boldsymbol{\theta}^c(s',\cdot)$ and select action $a' = \pi_c(s')$
15: Compute $\tilde{\boldsymbol{\psi}}^c(s',\cdot)$, $\tilde{\Sigma}^c(s',\cdot)$ using $\boldsymbol{\theta}^c(s',\cdot)$ and (18) and (19)
16: $a' \leftarrow \arg\max_b \{\tilde{\boldsymbol{\psi}}^c(s',b)^\intercal \mathbf{w}_c - \beta \mathbf{w}_c{}^\intercal \tilde{\Sigma}^c(s',b)\mathbf{w}_c\}$
\\ Apply Categorical Algorithm to update $\boldsymbol{\theta}^c(s,a)$
17: **for** $j = 0, 1 \dots N-1$ **do for** $i = 1, \dots d$ **do** $m_{j,i} \leftarrow 0$
18: **for** $i = 1, 2 \dots d$ **do**
19: **for** $j = 0, 1 \dots N-1$ **do**
\\ Compute the projection of $\hat{\mathcal{T}}_i z_{j,i}$ onto the support $\mathcal{Z}_i$
20: $\hat{\mathcal{T}}_i z_{j,i} \leftarrow \text{clip}\left(\phi_i(s,a,s') + \gamma z_{j,i}; [\Psi_i^{min}, \Psi_i^{max}]\right)$
21: $b_{j,i} \leftarrow (\hat{\mathcal{T}}_i z_{j,i} - \Psi_i^{min})/\Delta z_i$
22: $l \leftarrow \lfloor b_{j,i} \rfloor$, $u \leftarrow \lceil b_{j,i} \rceil$
\\ Distribute probability of $\hat{\mathcal{T}}_i z_{j,i}$
23: $m_{l,i} \leftarrow m_{l,i} + p_{j,i}^c(s',a')(u - b_{j,i})$
24: $m_{u,i} \leftarrow m_{u,i} + p_{j,i}^c(s',a')(b_{j,i} - l)$
25: **end for**
26: **end for**
27: Backpropagate through $-\sum_{j,i} m_{j,i} \log p_{j,i}^c(s,a)$ to update $\boldsymbol{\theta}^c(s,a)$
28: **end for**
29: $s \leftarrow s'$
30: **end for**
31: **end for**
32: **end for**

used to update all successor feature distributions simultaneously, as also done in SFDQN. Finally, the utility of returns can be used to select actions, rather than the expected return as done in DQN. Applying this last modification to SFC51 leads our proposed algorithm, which we call *Risk-aware SFC51* (RaSFC51). Of course, SFC51 can be recovered by simply setting $\beta = 0$. A complete description of RaSFC51 with the mean-variance approximation is provided in Algorithm 2[3].

[3] In practice, the double for loop starting in lines 18 and 19 can be implemented efficiently by vectoring the computation of $m_{j,i}$, in languages that support vectorized arithmetic operations.

A.4 Possible Generalizations of the Mean-Variance Approximation

Cumulant-Generating Functions. The quantity $K_R(\beta) = \log \mathbb{E}[e^{\beta R}]$ in (1) is often referred to as the *cumulant-generating function*. The cumulant generating function admits the well-known Taylor expansion:

$$U_\beta[R] = \frac{1}{\beta} K_R(\beta) = \frac{1}{\beta} \sum_{n=1}^{\infty} \kappa_R(n) \frac{\beta^n}{n!} = \sum_{n=1}^{\infty} \kappa_R(n) \frac{\beta^{n-1}}{n!}, \tag{21}$$

where $\kappa_R(n)$ is the n-th *cumulant* of the random variable R [62]. The mean-variance approximation (7) then follows directly from (21) by ignoring all terms of order $n \geq 3$. Another way to look at the mean-variance approximation is that it is the result of applying a *Laplace approximation* to the return distribution prior to calculating its utility [48]. While it is also possible to approximate (21) using orders of n greater than 2, such approximations would no longer provide "instantaneous" GPE. In particular, cumulants are much harder to compute as functions of $\mathbf{w}$ for $n = 3$ [49], and no closed formulas are even known to us for $n \geq 4$.

Elliptical Distributions. The mean-variance approximation (7) results from making the distributional assumption $\Psi_h^\pi(s,a) \sim \mathcal{N}(\boldsymbol{\psi}_h^\pi(s,a), \Sigma_h^\pi(s,a))$. Since the normal distribution is a member of the class of elliptical distributions, a natural question to ask is whether GPE can apply to other members of this class of distributions as well.

Formally, a random variable X has an *elliptical distribution* on $\mathbb{R}^d$ if there exists $\boldsymbol{\mu} \in \mathbb{R}^d$, positive definite $\Sigma \in \mathbb{R}^{d \times d}$ and a positive-valued function $\xi : \mathbb{R} \to \mathbb{R}$, and the characteristic function of X has the form

$$\mathbb{E}[e^{i\mathbf{t}^\intercal X}] = e^{i\mathbf{t}^\intercal \boldsymbol{\mu}} \xi(\mathbf{t}^\intercal \Sigma \mathbf{t}), \quad \forall \mathbf{t} \in \mathbb{R}^d. \tag{22}$$

Equivalently, for any random variable with characteristic function (22), there exists a positive function $g_d : \mathbb{R} \to \mathbb{R}$ such that the density of X is

$$f_X(\mathbf{x}) \propto |\Sigma|^{-1/2} g_d\left((\mathbf{x} - \boldsymbol{\mu})^\intercal \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right).$$

In either case, we write $X \sim \mathcal{E}_d(\boldsymbol{\mu}, \Sigma, \xi)$. One advantage of this parameterization is that $\boldsymbol{\mu}$ corresponds exactly to the mean of X, e.g. $\mathbb{E}[X] = \boldsymbol{\mu}$. Furthermore, if the covariance of X exists, then it is equal to Σ up to a positive multiplicative constant, e.g. $\mathrm{Var}[X] = c\Sigma$ for some $c > 0$[4].

In order to connect this to the SF framework, we parameterize $\Psi_h^\pi(s,a) \sim \mathcal{E}_d(\boldsymbol{\psi}_h^\pi(s,a), \Sigma_h^\pi(s,a), \xi)$. Then, using the linearity property (6, 12), GPE evaluates the entropic utilities of the random variables $\Psi_h^\pi(s,a)^\intercal \mathbf{w}$. Fortunately, affine transforms of elliptically distributed random variables are univariate elliptically distributed [57].

Lemma 2. *Let $X \sim \mathcal{E}_d(\boldsymbol{\mu}, \Sigma, \xi)$ and $\mathbf{w} \in \mathbb{R}^d$. Then, $X^\intercal \mathbf{w} \sim \mathcal{E}_1(\boldsymbol{\mu}^\intercal \mathbf{w}, \mathbf{w}^\intercal \Sigma \mathbf{w}, \xi)$.*

Applying Lemma 2 and then substituting $t = -i\beta$, the entropic utility becomes

$$U_\beta[\Psi_h^\pi(s,a)^\intercal \mathbf{w}] = \boldsymbol{\psi}_h^\pi(s,a)^\intercal \mathbf{w} + \frac{1}{\beta} \log \xi\left(-\beta^2 \mathbf{w}^\intercal \Sigma_h^\pi(s,a) \mathbf{w}\right), \tag{23}$$

and is also a mean-variance approximation. However, unlike (7) in which $\frac{1}{\beta} \log \circ \xi$ is the identity mapping, (23) is allowed to depend *non-linearly* on the variance of returns. Table 2 illustrates these mappings for different distributional assumptions. For heavy-tailed distributions, ξ should increase super-linearly for sufficiently large return variances, and thus (23) will often be more sensitive to variance than (7). This phenomenon is clearly illustrated in Figure 8 by comparing the variance penalties of the normal and Laplace distributions. Another advantage of this generalization is that the methodologies for estimating successor features and their covariances (Appendix A.1 and A.3) can be directly applied to this more general setting. An ablation study comparing the Gaussian assumption and the Laplace assumption in a complex experiment is provided at the end of Appendix D.2.

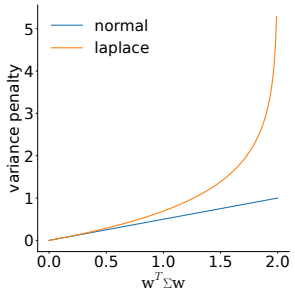

Figure 8: Comparing $\frac{1}{\beta} \log \circ \xi$ in (23) for normal and Laplace distributions, for $\beta = 1$.

[4] This implies that the Bellman updates (10) or (16) can still be used to learn Σ_h^π, but now the resulting estimates must be scaled by c when computing the utilities, if c is not one.

Name	Parameters	$\frac{1}{\beta}\log\circ\xi$
multivariate normal	$\boldsymbol{\mu}, \boldsymbol{\Sigma}$	$\frac{\beta}{2}\mathbf{w}^\intercal\Sigma\mathbf{w}$
multivariate Student	$\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu$	does not exist
multivariate Laplace	$\boldsymbol{\mu}, \boldsymbol{\Sigma}$	$-\frac{1}{\beta}\log(1-\frac{\beta^2}{2}\mathbf{w}^\intercal\Sigma\mathbf{w})$
multivariate logistic	$\boldsymbol{\mu}, \boldsymbol{\Sigma}$	$\frac{1}{\beta}\log \mathrm{B}(1-\beta\sqrt{\mathbf{w}^\intercal\Sigma\mathbf{w}}, 1+\beta\sqrt{\mathbf{w}^\intercal\Sigma\mathbf{w}})$

Table 2: Table of common elliptical distributions with corresponding variance penalties. Here B denotes the Beta function.

Skew-Elliptical Distributions. The elliptical distributions represent a well-known class of symmetric probability distributions, containing both heavy-tailed and light-tailed members as special cases. However, they cannot capture skew in the return distribution that often arises in strictly discounted MDPs. The class of *generalized skew-elliptical distributions* (GSE) can model skew by extending the characteristic function (22) to

$$\mathbb{E}[e^{i\mathbf{t}^\intercal X}] = e^{i\mathbf{t}^\intercal\boldsymbol{\mu}}\xi(\mathbf{t}^\intercal\Sigma\mathbf{t})k_d(\mathbf{t}), \quad \forall\mathbf{t}\in\mathbb{R}^d, \tag{24}$$

where $k_d : \mathbb{R}^d \to \mathbb{R}$ is some positive-valued function. In this case, we write $X \sim \mathcal{SE}_d(\boldsymbol{\mu}, \Sigma, \xi)$.

As for the elliptical distributions (Lemma 2), it is possible to show that GSE distributions are also closed under affine transforms [59].

Lemma 3. *Let $X \sim \mathcal{SE}_d(\boldsymbol{\mu}, \Sigma, \xi)$ and $\mathbf{w} \in \mathbb{R}^d$. Then, $X^\intercal\mathbf{w}$ is univariate GSE with characteristic function $\mathbb{E}[e^{itX^\intercal\mathbf{w}}] = e^{itX^\intercal\boldsymbol{\mu}}\xi\left(t^2\mathbf{w}^\intercal\Sigma\mathbf{w}\right)k(t;\mathbf{w},\Sigma)$ for some real-valued function k.*

By using Lemma 3 and the substitution $t = -i\beta$,

$$U_\beta[\Psi_h^\pi(s,a)^\intercal\mathbf{w}] = \boldsymbol{\psi}_h^\pi(s,a)^\intercal\mathbf{w} + \frac{1}{\beta}\log\xi\left(-\beta^2\mathbf{w}^\intercal\Sigma_h^\pi(s,a)\mathbf{w}\right) + \frac{1}{\beta}\log k(-i\beta;\mathbf{w},\Sigma_h^\pi(s,a)).$$

This new approximation generalizes (23) through the introduction of the term $\log k$, which intuitively captures the skew of the return distribution.

Mixtures Densities. One significant limitation of elliptical (and skew-elliptical) distributions to model returns is that they are unimodal, and can fail to capture multimodal risks in the environment. Consider the following *mixture of elliptical distributions* on $\mathbb{R}^d$:

$$\begin{aligned} I &\sim \mathrm{Categorical}_K(\boldsymbol{\pi}), \\ X \mid I = k &\sim \mathcal{E}_d(\boldsymbol{\mu}_k, \Sigma_k, \xi_k), \end{aligned}$$

where $\boldsymbol{\pi} \in \mathbb{R}^K$ satisfies $\pi_k \geq 0$ and $\sum_k \pi_k = 1$ and $k = 1, \dots K$ define the possible modes of the distribution. In other words, each component of the mixture is a member of an elliptical distribution. This model extends the standard Gaussian mixture model, and can approximate any continuous distribution to arbitrary accuracy provided K is chosen sufficiently large [58, 61]. In the context of risk-aware transfer (Theorem 1, Theorem 2) this means that the approximation error terms in ε associated with approximating U_β could in principle be driven to zero.

Applying Lemma 2 to each component, $X^\intercal\mathbf{w} \mid I = k \sim \mathcal{E}_1(\boldsymbol{\mu}_k{}^\intercal\mathbf{w}, \mathbf{w}^\intercal\Sigma_k\mathbf{w}, \xi_k)$, and so $X^\intercal\mathbf{w}$ is a mixture of univariate elliptical distributions. Now, (6, 12) can be computed using the law of total expectation,

$$U_\beta[\Psi_h^\pi(s,a)^\intercal\mathbf{w}] = \frac{1}{\beta}\log\sum_{k=1}^K \pi_k^\pi(s,a)e^{\beta\psi_k^\pi(s,a)^\intercal\mathbf{w}}\xi_k(-\beta^2\mathbf{w}^\intercal\Sigma_k^\pi(s,a)\mathbf{w}).$$

This expression does not simplify further unless $K = 1$ and is thus *not* a mean-variance approximation, although it *is* a generalization of (23). It can be computed numerically by using the log-sum-exp trick, and the parameters of its associated mixture density could be learned in the Bellman framework using expectation propagation [56].

B Proofs of Theoretical Results

In this section, we verify all the theoretical claims stated in the main paper for the episodic MDPs. For ease of exposition and due to space limitations, we also state and prove similar results for discounted MDPs in this section[5].

B.1 Proof of Lemma 1

Lemma 1. *Let $\beta \in \mathbb{R}$ and X, Y be arbitrary random variables on Ω. Then:*

A1 *(monotonicity) if $\mathbb{P}(X \geq Y) = 1$ then $U_\beta[X] \geq U_\beta[Y]$*
A2 *(cash invariance) $U_\beta[X + c] = U_\beta[X] + c$ for every $c \in \mathbb{R}$*
A3 *(convexity) if $\beta < 0$ ($\beta > 0$) then U_β is a concave (convex) function*
A4 *(non-expansion) for $f, g : \Omega \to \Omega$, it follows that*

$$|U_\beta[f(X)] - U_\beta[g(X)]| \leq \sup_{P \in \mathscr{P}_X(\Omega)} \mathbb{E}_P |f(X) - g(X)|,$$

where $\mathscr{P}_X(\Omega)$ is the set of all probability distributions on Ω that are absolutely continuous w.r.t. the true distribution of X.

Proof. The first three properties are derived in Föllmer and Schied [13]. As for the fourth property, we use **A3** and convex duality [13], [54] to write

$$U_\beta[R] = \sup_{P \in \mathscr{P}_R(\Omega)} \left\{ \mathbb{E}_P[R] - \frac{1}{\beta} D(P||P^*) \right\}, \tag{25}$$

where $\mathscr{P}_R(\Omega)$ is the set of all probability distributions on Ω absolutely continuous w.r.t. the true distribution P^* of R, D is the KL-divergence between P and P^*. Now, for $f, g : \Omega \to \Omega$, f and g are bounded and hence P-integrable for any $P \in \mathscr{P}_R(\Omega)$, and using (25):

$$\begin{aligned}
&|U_\beta[f(X)] - U_\beta[g(X)]| \\
&= \left| \sup_{P \in \mathscr{P}_X(\Omega)} \left\{ \mathbb{E}_P[f(X)] - \frac{1}{\beta} D(P||P^*) \right\} - \sup_{P \in \mathscr{P}_X(\Omega)} \left\{ \mathbb{E}_P[g(X)] - \frac{1}{\beta} D(P||P^*) \right\} \right| \\
&\leq \sup_{P \in \mathscr{P}_X(\Omega)} |\mathbb{E}_P[f(X)] - \mathbb{E}_P[g(X)]| \\
&\leq \sup_{P \in \mathscr{P}_X(\Omega)} \mathbb{E}_P |f(X) - g(X)|.
\end{aligned}$$

This completes the proof. □

B.2 Proofs of Theorem 1 and 2 for Episodic MDPs

Theorem 1. *Let $\pi_1, \dots \pi_n$ be arbitrary deterministic Markov policies with approximate entropic utilities $\tilde{\mathcal{Q}}^{\pi_1}_{h,\beta}, \dots \tilde{\mathcal{Q}}^{\pi_n}_{h,\beta}$ evaluated in an arbitrary task M, such that the errors satisfy $|\tilde{\mathcal{Q}}^{\pi_i}_{h,\beta}(s,a) - \mathcal{Q}^{\pi_i}_{h,\beta}(s,a)| \leq \varepsilon$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $i = 1 \dots n$ and $h \in \mathcal{T}$. Define*

$$\pi_h(s) \in \arg\max_{a \in \mathcal{A}} \max_{i=1\dots n} \tilde{\mathcal{Q}}^{\pi_i}_{h,\beta}(s,a), \quad \forall s \in \mathcal{S}.$$

Then,

$$\mathcal{Q}^{\pi}_{h,\beta}(s,a) \geq \max_i \mathcal{Q}^{\pi_i}_{h,\beta}(s,a) - 2(T - h + 1)\varepsilon, \quad h \leq T.$$

Proof. We have for all h that

$$|\max_i \mathcal{Q}^{\pi_i}_h(s,a) - \max_i \tilde{\mathcal{Q}}^{\pi_i}_h(s,a)| \leq \max_i |\mathcal{Q}^{\pi_i}_h(s,a) - \tilde{\mathcal{Q}}^{\pi_i}_h(s,a)| \leq \varepsilon$$

[5] While the analysis of GPI in the risk-neutral setting [2] follows Strehl and Littman [60], our analysis of risk-aware GPI is also inspired by Huang and Haskell [53].

We proceed by induction on h. Clearly, the desired result holds for $h = T + 1$ since $\mathcal{Q}^{\pi_i}_{T+1,\beta}(s,a) = \tilde{\mathcal{Q}}^{\pi_i}_{T+1,\beta}(s,a) = 0$ uniformly. Next, suppose that $\mathcal{Q}^{\pi}_{h+1,\beta}(s,a) \geq \max_i \mathcal{Q}^{\pi_i}_{h+1,\beta}(s,a) - 2\varepsilon(T-h)$ holds uniformly at time $h+1$. Using **A1** and **A2** of Lemma 1:

$$\begin{aligned}
\mathcal{Q}^{\pi}_{h,\beta}(s,a) &= U_\beta[r(s,a,s') + \mathcal{Q}^{\pi}_{h+1,\beta}(s', \pi_{h+1}(s'))] \\
&\geq U_\beta[r(s,a,s') + \max_i \mathcal{Q}^{\pi_i}_{h+1,\beta}(s', \pi_{h+1}(s')) - 2\varepsilon(T-h)] \\
&\geq U_\beta[r(s,a,s') + \max_i \tilde{\mathcal{Q}}^{\pi_i}_{h+1,\beta}(s', \pi_{h+1}(s')) - 2\varepsilon(T-h) - \varepsilon] \\
&= U_\beta[r(s,a,s') + \max_{a'} \max_i \tilde{\mathcal{Q}}^{\pi_i}_{h+1,\beta}(s', a') - 2\varepsilon(T-h) - \varepsilon] \\
&\geq U_\beta[r(s,a,s') + \max_{a'} \max_i \mathcal{Q}^{\pi_i}_{h+1,\beta}(s', a') - 2\varepsilon(T-h+1)] \\
&\geq U_\beta[r(s,a,s') + \max_i \max_{a'} \mathcal{Q}^{\pi_i}_{h+1,\beta}(s', a')] - 2\varepsilon(T-h+1) \\
&\geq U_\beta[r(s,a,s') + \max_i \mathcal{Q}^{\pi_i}_{h+1,\beta}(s', \pi_{i,h+1}(s'))] - 2\varepsilon(T-h+1) \\
&\geq U_\beta[r(s,a,s') + \mathcal{Q}^{\pi_i}_{h+1,\beta}(s', \pi_{i,h+1}(s'))] - 2\varepsilon(T-h+1) \\
&= \mathcal{Q}^{\pi_i}_{h,\beta}(s,a) - 2\varepsilon(T-h+1).
\end{aligned}$$

Since i is arbitrary, the proof is complete. □

Lemma 4. *Let $\mathcal{Q}^{ij}_h$ be the utility of policy π^*_i evaluated in task j at time h. Then for all i, j,*

$$\sup_{s,a} \left| \mathcal{Q}^{ii}_h(s,a) - \mathcal{Q}^{jj}_h(s,a) \right| \leq (T-h+1)\delta_{ij}.$$

Proof. Let $\Delta_{ij}(h) = \sup_{s,a} |\mathcal{Q}^{ii}_h(s,a) - \mathcal{Q}^{jj}_h(s,a)|$. Define $\mathscr{P}_{s,a}$ to be the set of probability distributions that are absolutely continuous with respect to $P(\cdot|s,a)$. Then, using **A4** from Lemma 1:

$$\begin{aligned}
&\Delta_{ij}(h) \\
&= \sup_{s,a} \left| \mathcal{Q}^{ii}_h(s,a) - \mathcal{Q}^{jj}_h(s,a) \right| \\
&= \sup_{s,a} \left| U_\beta[r_i(s,a,s') + \max_{a'} \mathcal{Q}^{ii}_{h+1}(s',a')] - U_\beta[r_j(s,a,s') + \max_{a'} \mathcal{Q}^{jj}_{h+1}(s',a')] \right| \\
&\leq \sup_{s,a} \sup_{P' \in \mathscr{P}_{s,a}} \mathbb{E}_{s' \sim P'(\cdot|s,a)} \left| r_i(s,a,s') - r_j(s,a,s') + \max_{a'} \mathcal{Q}^{ii}_{h+1}(s',a') - \max_{a'} \mathcal{Q}^{jj}_{h+1}(s',a') \right| \\
&\leq \sup_{s,a} \sup_{s'} \left| r_i(s,a,s') - r_j(s,a,s') + \max_{a'} \mathcal{Q}^{ii}_{h+1}(s',a') - \max_{a'} \mathcal{Q}^{jj}_{h+1}(s',a') \right| \\
&\leq \sup_{s,a,s'} |r_i(s,a,s') - r_j(s,a,s')| + \sup_{s,a,s'} \left| \max_{a'} \mathcal{Q}^{ii}_{h+1}(s',a') - \max_{a'} \mathcal{Q}^{jj}_{h+1}(s',a') \right| \\
&= \delta_{ij} + \sup_{s,a} \left| \mathcal{Q}^{ii}_{h+1}(s,a) - \mathcal{Q}^{jj}_{h+1}(s,a) \right| \\
&= \delta_{ij} + \Delta_{ij}(h+1).
\end{aligned}$$

Starting with $\Delta_{ij}(T+1) = 0$ and proceeding by backward induction, we have $\Delta_{ij}(h) \leq \delta_{ij}(T - h + 1)$ for all h. □

Lemma 5.

$$\sup_{s,a} \left| \mathcal{Q}^{jj}_h(s,a) - \mathcal{Q}^{ji}_h(s,a) \right| \leq (T-h+1)\delta_{ij}.$$

Proof. Define $\Gamma_{ij}(h) = \sup_{s,a} |\mathcal{Q}^{jj}_h(s,a) - \mathcal{Q}^{ji}_h(s,a)|$. Then using **A4** from Lemma 1:

$$\begin{aligned}
&\Gamma_{ij}(h) \\
&= \sup_{s,a} \left| \mathcal{Q}^{jj}_h(s,a) - \mathcal{Q}^{ji}_h(s,a) \right|
\end{aligned}$$

$$
\begin{aligned}
&= \sup_{s,a} \left| U_\beta[r_j(s,a,s') + \mathcal{Q}^{jj}_{h+1}(s', \pi^*_{j,h+1}(s'))] - U_\beta[r_i(s,a,s') + \mathcal{Q}^{ji}_{h+1}(s', \pi^*_{j,h+1}(s'))] \right| \\
&\leq \sup_{s,a} \sup_{P' \in \mathscr{P}_{s,a}} \mathbb{E}_{s' \sim P'(\cdot|s,a)} \left| r_i(s,a,s') - r_j(s,a,s') + \mathcal{Q}^{jj}_{h+1}(s', \pi^*_{j,h+1}(s')) - \mathcal{Q}^{ji}_{h+1}(s', \pi^*_{j,h+1}(s')) \right| \\
&\leq \sup_{s,a} \sup_{s'} \left| r_i(s,a,s') - r_j(s,a,s') + \mathcal{Q}^{jj}_{h+1}(s', \pi^*_{j,h+1}(s')) - \mathcal{Q}^{ji}_{h+1}(s', \pi^*_{j,h+1}(s')) \right| \\
&\leq \sup_{s,a,s'} |r_i(s,a,s') - r_j(s,a,s')| + \sup_{s,a,s'} \left| \mathcal{Q}^{jj}_{h+1}(s', \pi^*_{j,h+1}(s')) - \mathcal{Q}^{ji}_{h+1}(s', \pi^*_{j,h+1}(s')) \right| \\
&\leq \delta_{ij} + \sup_{s,a} \left| \mathcal{Q}^{jj}_{h+1}(s,a) - \mathcal{Q}^{ji}_{h+1}(s,a) \right| \\
&= \delta_{ij} + \Gamma_{ij}(h+1).
\end{aligned}
$$

Thus, $\Gamma_{ij}(h) \leq \delta_{ij}(T-h+1)$ as claimed. □

Theorem 2. *Let $\mathcal{Q}^{\pi^*_i}_{h,\beta}$ be the utilities of optimal Markov policies π^*_i evaluated in task M. Furthermore, let $\tilde{\mathcal{Q}}^{\pi^*_i}_{h,\beta}$ be such that $|\tilde{\mathcal{Q}}^{\pi^*_i}_{h,\beta}(s,a) - \mathcal{Q}^{\pi^*_i}_{h,\beta}(s,a)| < \varepsilon$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $h \in \mathcal{T}$ and $i = 1 \dots n$. Similarly, let π be the corresponding policy in (4). Finally, let $\delta_r = \min_{i=1\dots n} \sup_{s,a,s'} |r(s,a,s') - r_i(s,a,s')|$. Then,*

$$
\left| \mathcal{Q}^{\pi}_{h,\beta}(s,a) - \mathcal{Q}^{*}_{h,\beta}(s,a) \right| \leq 2(T-h+1)(\delta_r + \varepsilon), \quad h \leq T.
$$

Proof. Using Theorem 1 and the triangle inequality:

$$
\left| \mathcal{Q}^{\pi}_{h,\beta}(s,a) - \mathcal{Q}^{*}_{h,\beta}(s,a) \right| \leq \left| \mathcal{Q}^{\pi^*_j}_{h,\beta}(s,a) - \mathcal{Q}^{*}_{h,\beta}(s,a) \right| + 2(T-h+1)\varepsilon.
$$

The goal now is to bound the first term. By the triangle inequality and Lemma 4 and Lemma 5, $|\mathcal{Q}^{ii}_h(s,a) - \mathcal{Q}^{ji}_h(s,a)| \leq |\mathcal{Q}^{ii}_h(s,a) - \mathcal{Q}^{jj}_h(s,a)| + |\mathcal{Q}^{jj}_h(s,a) - \mathcal{Q}^{ji}_h(s,a)| = 2(T-h+1)\delta_{ij}$. Finally, designating j as source task j and i as target task, and substituting this bound into the first inequality above yields the desired result. □

B.3 Proofs of Theorem 1 and 2 for Discounted MDPs

Theorem 5. *Let $\pi_1, \dots \pi_n$ be arbitrary deterministic Markov policies with approximate entropic utilities $\tilde{\mathcal{J}}^{\pi_1}_\beta, \dots \tilde{\mathcal{J}}^{\pi_n}_\beta$ evaluated in an arbitrary task M, such that the errors satisfy $|\tilde{\mathcal{J}}^{\pi_i}_\beta(s,a,z) - \mathcal{J}^{\pi_i}_\beta(s,a,z)| \leq \varepsilon z$ for all s, a, z and i. Define*

$$
\pi(s,z) \in \arg\max_{a \in \mathcal{A}} \max_{i=1\dots n} \tilde{\mathcal{J}}^{\pi_i}_\beta(s,a,z), \quad \forall s \in \mathcal{S},\ z \in \mathcal{Z}. \tag{26}
$$

Then,

$$
\mathcal{J}^{\pi}_\beta(s,a,z) \geq \max_i \mathcal{J}^{\pi_i}_\beta(s,a,z) - \frac{2\varepsilon}{1-\gamma} z.
$$

Proof. Define $\mathcal{J}^{max}_\beta(s,a,z) = \max_i \mathcal{J}^{\pi_i}_\beta(s,a,z)$ and $\tilde{\mathcal{J}}^{max}_\beta(s,a,z) = \max_i \tilde{\mathcal{J}}^{\pi_i}_\beta(s,a,z)$. We have:

$$
|\mathcal{J}^{max}_\beta(s,a,z) - \tilde{\mathcal{J}}^{max}_\beta(s,a,z)| \leq \max_i |\mathcal{J}^{\pi_i}_\beta(s,a,z) - \tilde{\mathcal{J}}^{\pi_i}_\beta(s,a,z)| \leq \varepsilon z.
$$

Let T^{π}_β be the operator corresponding to (11). Then using **A1** and **A2** of Lemma 1 leads to:

$$
\begin{aligned}
T^{\pi}_\beta \tilde{\mathcal{J}}^{max}_\beta(s,a,z) &= U_\beta \left[z r(s,a,s') + \tilde{\mathcal{J}}^{max}_\beta(s', \pi(s', \gamma z), \gamma z) \right] \\
&= U_\beta \left[z r(s,a,s') + \max_{a'} \tilde{\mathcal{J}}^{max}_\beta(s', a', \gamma z) \right] \\
&\geq U_\beta \left[z r(s,a,s') + \max_{a'} \mathcal{J}^{max}_\beta(s', a', \gamma z) \right] - \gamma \varepsilon z \\
&\geq U_\beta \left[z r(s,a,s') + \mathcal{J}^{max}_\beta(s', \pi_i(s', \gamma z), \gamma z) \right] - \gamma \varepsilon z
\end{aligned}
$$

$$
\begin{aligned}
&\geq U_\beta\left[zr(s,a,s') + \mathcal{J}_\beta^{\pi_i}(s', \pi_i(s', \gamma z), \gamma z)\right] - \gamma\varepsilon z \\
&= T_\beta^{\pi_i}\mathcal{J}_\beta^{\pi_i}(s,a,z) - \gamma\varepsilon z \\
&= \mathcal{J}_\beta^{\pi_i}(s,a,z) - \gamma\varepsilon z \\
&\geq \max_i \mathcal{J}_\beta^{\pi_i}(s,a,z) - \gamma\varepsilon z \\
&\geq \tilde{\mathcal{J}}_\beta^{max}(s,a,z) - \varepsilon z - \gamma\varepsilon z
\end{aligned}
$$

Finally, using **A1** of Lemma 1 and the fact that T_β^π has a unique fixed point [5]:

$$
\begin{aligned}
\mathcal{J}_\beta^\pi(s,a,z) &= \lim_{k\to\infty}(T_\beta^\pi)^k\tilde{\mathcal{J}}_\beta^{max}(s,a,z) \geq \tilde{\mathcal{J}}_\beta^{max}(s,a,z) - (1+\gamma)\frac{\varepsilon}{1-\gamma}z \\
&\geq \mathcal{J}_\beta^{max}(s,a,z) - \frac{2\varepsilon}{1-\gamma}z,
\end{aligned}
$$

and is the desired result. □

Lemma 6. *Define $\mathcal{J}_j^i(s,a,z)$ be the utility of optimal policy π_i^* on the augmented MDP i when evaluated in the augmented MDP j. Furthermore, let $\delta_{ij} = \sup_{s,a,s'}|r_i(s,a,s') - r_j(s,a,s')|$. Then,*

$$
\sup_{s,a}\left|\mathcal{J}_i^i(s,a,z) - \mathcal{J}_j^j(s,a,z)\right| \leq \frac{\delta_{ij}}{1-\gamma}z.
$$

Proof. Define $\Delta_{ij}(z) = \sup_{s,a}\left|\mathcal{J}_i^i(s,a,z) - \mathcal{J}_j^j(s,a,z)\right|$. Let $\mathscr{P}_{s,a}$ be the set of probability distributions for the one-step transitions of the augmented MDP, e.g. $P((s',z')|(s,z),a)$ that are absolutely continuous w.r.t. the true distribution. Since $P((s',z')|(s,z),a) = P(s'|s,a)\delta_{z\gamma}(z')$, and $\delta_{z\gamma}(z')$ is absolutely continuous only w.r.t. itself, the set $\mathscr{P}_{s,a}$ consists of all products $P(s'|s,a)\delta_{z\gamma}(z')$, where $P(s'|s,a)$ is absolutely continuous w.r.t. the true dynamics of the original MDP.

Now, using **A4** from Lemma 1:

$$
\begin{aligned}
&\Delta_{ij}(z) \\
&= \sup_{s,a}\left|\mathcal{J}_i^i(s,a,z) - \mathcal{J}_j^j(s,a,z)\right| \\
&= \sup_{s,a}\left|U_\beta[zr_i(s,a,s') + \max_{a'}\mathcal{J}_i^i(s',a',\gamma z)] - U_\beta[zr_j(s,a,s') + \max_{a'}\mathcal{J}_j^j(s',a',\gamma z)]\right| \\
&\leq \sup_{s,a}\sup_{P\in\mathscr{P}_{s,a}}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left|zr_i(s,a,s') - zr_j(s,a,s') + \max_{a'}\mathcal{J}_i^i(s',a',\gamma z) - \max_{a'}\mathcal{J}_j^j(s',a',\gamma z)\right| \\
&\leq z\sup_{s,a,s'}|r_i(s,a,s') - r_j(s,a,s')| + \sup_{s,a,s'}\left|\max_{a'}\mathcal{J}_i^i(s',a',\gamma z) - \max_{a'}\mathcal{J}_j^j(s',a',\gamma z)\right| \\
&= z\delta_{ij} + \sup_{s,a}\left|\mathcal{J}_i^i(s,a,\gamma z) - \mathcal{J}_j^j(s,a,\gamma z)\right| \\
&= z\delta_{ij} + \Delta_{ij}(\gamma z).
\end{aligned}
$$

Repeating the above bounding procedure leads to:

$$
\begin{aligned}
\Delta_{ij}(z) &\leq z\delta_{ij} + \Delta_{ij}(\gamma z) \\
&\leq z\delta_{ij} + \gamma z\delta_{ij} + \Delta_{ij}(\gamma^2 z) \\
&\vdots \\
&\leq z\delta_{ij} + \gamma z\delta_{ij} + \gamma^2 z\delta_{ij} + \cdots = \frac{\delta_{ij}}{1-\gamma}z,
\end{aligned}
$$

and completes the proof. □

Lemma 7.

$$
\sup_{s,a}\left|\mathcal{J}_j^j(s,a,z) - \mathcal{J}_i^j(s,a,z)\right| \leq \frac{\delta_{ij}}{1-\gamma}z.
$$

Proof. Define $\Gamma_{ij}(z) = \sup_{s,a} |\mathcal{J}_j^j(s,a,z) - \mathcal{J}_i^j(s,a,z)|$. Then, using **A4** from Lemma 1 and the technique from Lemma 6:

$$
\begin{aligned}
&\Gamma_{ij}(z) \\
&= \sup_{s,a} \left| \mathcal{J}_j^j(s,a,z) - \mathcal{J}_i^j(s,a,z) \right| \\
&= \sup_{s,a} \left| U_\beta[z r_j(s,a,s') + \mathcal{J}_j^j(s', \pi_j^*(s',\gamma z), \gamma z)] - U_\beta[z r_i(s,a,s') + \mathcal{J}_i^j(s', \pi_j^*(s',\gamma z), \gamma z)] \right| \\
&\leq \sup_{s,a} \sup_{P \in \mathscr{P}_{s,a}} \mathbb{E}_{s' \sim P(\cdot|s,a)} \left| z r_i(s,a,s') - z r_j(s,a,s') + \mathcal{J}_j^j(s', \pi_j^*(s',\gamma z), \gamma z) - \mathcal{J}_i^j(s', \pi_j^*(s',\gamma z), \gamma z) \right| \\
&\leq z \sup_{s,a,s'} |r_i(s,a,s') - r_j(s,a,s')| + \sup_{s,a,s'} \left| \mathcal{J}_j^j(s', \pi_j^*(s',\gamma z), \gamma z) - \mathcal{J}_i^j(s', \pi_j^*(s',\gamma z), \gamma z) \right| \\
&\leq z\delta_{ij} + \sup_{s,a} \left| \mathcal{J}_j^j(s,a,\gamma z) - \mathcal{J}_i^j(s,a,\gamma z) \right| \\
&= z\delta_{ij} + \Gamma_{ij}(\gamma z) \\
&\leq \frac{\delta_{ij}}{1-\gamma} z.
\end{aligned}
$$

The proof is complete. □

Theorem 6. *Let $\mathcal{J}_\beta^{\pi_i^*}$ be the utilities of optimal Markov policies π_i^* evaluated in some task M. Furthermore, let $\tilde{\mathcal{J}}_\beta^{\pi_i^*}$ be such such that $|\tilde{\mathcal{J}}_\beta^{\pi_i^*}(s,a,z) - \mathcal{J}_\beta^{\pi_i^*}(s,a,z)| < \varepsilon z$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $z \in \mathcal{Z}$ and $i = 1 \dots n$, and π be the corresponding policy in (26). Finally, let $\delta_r = \min_{i=1\dots n} \sup_{s,a,s'} |r(s,a,s') - r_i(s,a,s')|$. Then,*

$$
\left| \mathcal{J}_\beta^\pi(s,a,z) - \mathcal{J}_\beta^*(s,a,z) \right| \leq \frac{2(\delta_r + \varepsilon)}{1-\gamma} z.
$$

Proof. Using Theorem 5:

$$
\begin{aligned}
\mathcal{J}_\beta^\pi(s,a,z) - \mathcal{J}_\beta^*(s,a,z) &= \mathcal{J}_\beta^\pi(s,a,z) - \mathcal{J}_\beta^{\pi_j^*}(s,a,z) + \mathcal{J}_\beta^{\pi_j^*}(s,a,z) - \mathcal{J}_\beta^*(s,a,z) \\
&\geq \frac{2\varepsilon}{1-\gamma} z + \mathcal{J}_\beta^{\pi_j^*}(s,a,z) - \mathcal{J}_\beta^*(s,a,z).
\end{aligned}
$$

The goal now is to bound the difference between the last two terms. Let $\mathcal{J}_j^i(s,a)$ be the entropic utility of the optimal policy π_i^* evaluated in the augmented MDP for task j. Then, by the triangle inequality, $|\mathcal{J}_i^i(s,a,z) - \mathcal{J}_i^j(s,a,z)| \leq |\mathcal{J}_i^i(s,a,z) - \mathcal{J}_j^j(s,a,z)| + |\mathcal{J}_j^j(s,a,z) - \mathcal{J}_i^j(s,a,z)|$. Applying Lemma 6 and Lemma 7, we have $|\mathcal{J}_i^i(s,a,z) - \mathcal{J}_i^j(s,a,z)| \leq \frac{2\delta_{ij}}{1-\gamma} z$, and the result follows. □

B.4 Proof of Theorem 3

The proofs depend on the following result adapted from Sherstan et al. [37].

Lemma 8. *Let X be a random vector in $\mathbb{R}^d$ that depends only on s_h, a_h, r_h and s_{h+1}. Then,*

$$
\mathbb{E}\left[X(\Psi_{h+1}^\pi(s', \pi_{h+1}(s')) - \boldsymbol{\psi}_{h+1}^\pi(s', \pi_{h+1}(s'))^\intercal \mid s_h = s,\, a_h = a \right] = 0.
$$

We first demonstrate that the Bellman equation (8) is correct for our problem.

Lemma 9.

$$
\Sigma_h^\pi(s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\boldsymbol{\delta}_h \boldsymbol{\delta}_h^\intercal + \Sigma_{h+1}^\pi(s', \pi_{h+1}(s')) \mid s_h = s,\, a_h = a \right].
$$

Proof. Let $\Psi_h^\pi(s,a) = \boldsymbol{\phi}_h + \boldsymbol{\phi}_{h+1} + \dots$ and define $\boldsymbol{\xi}_h^\pi(s,a) = \Psi_h^\pi(s,a) - \boldsymbol{\psi}_h^\pi(s,a)$. By definition of successor features, we have:

$$
\boldsymbol{\xi}_h^\pi(s,a) = \Psi_h^\pi(s,a) - \boldsymbol{\psi}_h^\pi(s,a)
$$

$$= \boldsymbol{\phi}_h + \boldsymbol{\psi}^\pi_{h+1}(s', \pi_{h+1}(s')) - \boldsymbol{\psi}^\pi_h(s,a) + (\Psi^\pi_{h+1}(s', \pi_{h+1}(s')) - \boldsymbol{\psi}^\pi_{h+1}(s', \pi_{h+1}(s')))$$
$$= \boldsymbol{\delta}_h + \boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s'))$$

By definition, the covariance is:

$$\begin{aligned}
\Sigma^\pi_h(s,a) &= \mathbb{E}\left[(\Psi^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a))(\Psi^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a))^\intercal \mid s_h = s,\, a_h = a\right] \\
&= \mathbb{E}\left[\boldsymbol{\xi}^\pi_h(s,a)\boldsymbol{\xi}^\pi_h(s,a)^\intercal \mid s_h = s,\, a_h = a\right] \\
&= \mathbb{E}\left[(\boldsymbol{\delta}_h + \boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s')))(\boldsymbol{\delta}_h + \boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s')))^\intercal \mid s_h = s,\, a_h = a\right] \\
&= \mathbb{E}\left[\boldsymbol{\delta}_h\boldsymbol{\delta}_h{}^\intercal + \boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s'))\boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s'))^\intercal \mid s_h = s,\, a_h = a\right] \\
&\quad + \mathbb{E}\left[\boldsymbol{\delta}_h\boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s'))^\intercal \mid s_h = s,\, a_h = a\right] \\
&\quad + \mathbb{E}\left[\boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s'))\boldsymbol{\delta}_h{}^\intercal \mid s_h = s,\, a_h = a\right] \\
&= \mathbb{E}\left[\boldsymbol{\delta}_h\boldsymbol{\delta}_h{}^\intercal + \boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s'))\boldsymbol{\xi}^\pi_{h+1}(s', \pi_{h+1}(s'))^\intercal \mid s_h = s,\, a_h = a\right] \\
&= \mathbb{E}\left[\boldsymbol{\delta}_h\boldsymbol{\delta}_h{}^\intercal + \Sigma^\pi_{h+1}(s', \pi_{h+1}(s')) \mid s_h = s,\, a_h = a\right],
\end{aligned}$$

where the second-last line follows from Lemma 8. □

Theorem 3. *Let $\|\cdot\|$ be a matrix-compatible norm, and suppose there exists $\varepsilon : \mathcal{S} \times \mathcal{A} \times \mathcal{T} \to [0, \infty)$ such that:*

1. $\|\tilde{\boldsymbol{\psi}}^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a)\|^2 \le \varepsilon_h(s,a)$
2. $\|\mathbb{E}_{s' \sim P(\cdot|s,a)}[\tilde{\boldsymbol{\delta}}_h(\tilde{\boldsymbol{\psi}}^\pi_h(s', \pi_{h+1}(s')) - \boldsymbol{\psi}^\pi_h(s', \pi_{h+1}(s')))^\intercal]\| \le \varepsilon_h(s,a)$.

Then,

$$\left\|\Sigma^\pi_h(s,a) - \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\tilde{\boldsymbol{\delta}}_h\tilde{\boldsymbol{\delta}}_h^\intercal + \tilde{\Sigma}^\pi_{h+1}(s', \pi_{h+1}(s'))\right]\right\| \le 3\varepsilon_h(s,a).$$

Proof. We start by decomposing the true covariance matrix:

$$\begin{aligned}
\Sigma^\pi_h(s,a) = \mathbb{E}\Big[&(\Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a) + \boldsymbol{\psi}^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a)) \\
&(\Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a) + \boldsymbol{\psi}^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a))^\intercal \mid s_h = s,\, a_h = a\Big] \\
= \mathbb{E}&\left[(\Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a))(\Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a))^\intercal \mid s_h = s,\, a_h = a\right] \\
&+ (\tilde{\boldsymbol{\psi}}^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a))(\tilde{\boldsymbol{\psi}}^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a))^\intercal \\
&+ 2\mathbb{E}\left[\Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a) \mid s_h = s,\, a_h = a\right](\tilde{\boldsymbol{\psi}}^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a))^\intercal \\
= \mathbb{E}&\left[(\Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a))(\Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a))^\intercal \mid s_h = s,\, a_h = a\right] \\
&- (\tilde{\boldsymbol{\psi}}^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a))(\tilde{\boldsymbol{\psi}}^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a))^\intercal
\end{aligned}$$

where in the last step we use the identity $\mathbb{E}\left[\Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a) \mid s_h = s,\, a_h = a\right] = \mathbb{E}\left[\Psi^\pi_h(s,a) - \boldsymbol{\psi}^\pi_h(s,a) \mid s_h = s,\, a_h = a\right] + \boldsymbol{\psi}^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a) = \boldsymbol{\psi}^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a)$. Now, we define $\tilde{\boldsymbol{\xi}}^\pi_h(s,a) = \Psi^\pi_h(s,a) - \tilde{\boldsymbol{\psi}}^\pi_h(s,a)$, then follow the derivations in Lemma 9 to write the first term above as:

$$\begin{aligned}
&\mathbb{E}\left[\tilde{\boldsymbol{\xi}}^\pi_h(s,a)\tilde{\boldsymbol{\xi}}^\pi_h(s,a)^\intercal \mid s_h = s,\, a_h = a\right] \\
&= \mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h\tilde{\boldsymbol{\delta}}_h^\intercal + \tilde{\boldsymbol{\xi}}^\pi_{h+1}(s', \pi_{h+1}(s'))\tilde{\boldsymbol{\xi}}^\pi_{h+1}(s', \pi_{h+1}(s'))^\intercal \mid s_h = s,\, a_h = a\right] \\
&\quad + \mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h\tilde{\boldsymbol{\xi}}^\pi_{h+1}(s', \pi_{h+1}(s'))^\intercal \mid s_h = s,\, a_h = a\right] + \mathbb{E}\left[\tilde{\boldsymbol{\xi}}^\pi_{h+1}(s', \pi_{h+1}(s'))\tilde{\boldsymbol{\delta}}_h^\intercal \mid s_h = s,\, a_h = a\right] \\
&= \mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h\tilde{\boldsymbol{\delta}}_h^\intercal + \tilde{\Sigma}^\pi_{h+1}(s', \pi_{h+1}(s')) \mid s_h = s,\, a_h = a\right] \\
&\quad + \mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h\tilde{\boldsymbol{\xi}}^\pi_{h+1}(s', \pi_{h+1}(s'))^\intercal \mid s_h = s,\, a_h = a\right] + \mathbb{E}\left[\tilde{\boldsymbol{\xi}}^\pi_{h+1}(s', \pi_{h+1}(s'))\tilde{\boldsymbol{\delta}}_h^\intercal \mid s_h = s,\, a_h = a\right].
\end{aligned}$$

Finally, we norm bound the desired difference as follows:

$$
\begin{aligned}
&\left\|\Sigma_h^\pi(s,a) - \mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h\tilde{\boldsymbol{\delta}}_h^\intercal + \tilde{\Sigma}_{h+1}^\pi(s',\pi_{h+1}(s')) \mid s_h = s,\, a_h = a\right]\right\| \\
&\leq 2\left\|\mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h\tilde{\boldsymbol{\xi}}_{h+1}^\pi(s',\pi_{h+1}(s'))^\intercal \mid s_h = s,\, a_h = a\right]\right\| \\
&\quad + \left\|(\tilde{\boldsymbol{\psi}}_h^\pi(s,a) - \boldsymbol{\psi}_h^\pi(s,a))(\tilde{\boldsymbol{\psi}}_h^\pi(s,a) - \boldsymbol{\psi}_h^\pi(s,a))^\intercal\right\| \\
&\leq 2\left\|\mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h(\Psi_{h+1}^\pi(s',\pi_{h+1}(s')) - \boldsymbol{\psi}_{h+1}^\pi(s',\pi_{h+1}(s')))^\intercal \mid s_h = s,\, a_h = a\right]\right\| \\
&\quad + 2\left\|\mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h(\boldsymbol{\psi}_{h+1}^\pi(s',\pi_{h+1}(s')) - \tilde{\boldsymbol{\psi}}_{h+1}^\pi(s',\pi_{h+1}(s')))^\intercal \mid s_h = s,\, a_h = a\right]\right\| \\
&\quad + \left\|(\tilde{\boldsymbol{\psi}}_h^\pi(s,a) - \boldsymbol{\psi}_h^\pi(s,a))(\tilde{\boldsymbol{\psi}}_h^\pi(s,a) - \boldsymbol{\psi}_h^\pi(s,a))^\intercal\right\| \\
&\leq 2\left\|\mathbb{E}\left[\tilde{\boldsymbol{\delta}}_h(\boldsymbol{\psi}_{h+1}^\pi(s',\pi_{h+1}(s')) - \tilde{\boldsymbol{\psi}}_{h+1}^\pi(s',\pi_{h+1}(s')))^\intercal \mid s_h = s,\, a_h = a\right]\right\| \\
&\quad + \left\|\tilde{\boldsymbol{\psi}}_h^\pi(s,a) - \boldsymbol{\psi}_h^\pi(s,a))\right\|^2 \\
&\leq 2\varepsilon_h(s,a) + \varepsilon_h(s,a) = 3\varepsilon_h(s,a).
\end{aligned}
$$

This is the desired result. □

C Experiment Details

In this section, we describe the setup of the domains discussed in the main paper in greater detail. We also provide detailed descriptions of baseline algorithms, as well as all hyper-parameters used and how they were selected.

C.1 Motivating Example

Domain Configuration. The motivating example is a 5-by-5 grid-world domain with discrete states and discrete actions described by the four possible directions of movement into an adjacent cell. The environment is made stochastic by introducing random action noise as follows. Desired actions are taken only with probability 0.8, while the remaining time a (uniformly) random action is taken. Furthermore, transitions that would take the agent outside of the boundaries of the grid leave the agent in its current position. The cost structure is defined as follows. The goal state is terminal and provides a reward of $+20$. Each time step incurs a fixed penalty of -1, on top of any other rewards or costs incurred.

Learning Source Policies and Utilities. To recover the properties of risk-aware and risk-neutral GPI claimed in the main text, we first learn the source policies π_1 and π_2 and their utilities $\mathcal{Q}_\beta^{\pi_1}$ and $\mathcal{Q}_\beta^{\pi_2}$ using a variant of the classic value iteration algorithm adapted to maximize the entropic utility (see Algorithm 3). We consider the non-discounted setting ($\gamma = 1$), and iterate until an absolute error less than $\varepsilon_{exit} = 10^{-12}$ is achieved between two consecutive iterations[6]. The two source policies are then recovered by acting greedily with respect to the learned utilities.

Transfer Learning. In order to implement GPI, we evaluate these two resulting policies on the target task by adapting the iterative procedure in Algorithm 3 for *policy evaluation*. Essentially, line 10 of the algorithm is replaced by $r(s,a,s') + \gamma\mathcal{Q}(s',\pi(s'))$ for $\pi \in \{\pi_1,\pi_2\}$. We repeat this procedure twice to produce two sets of value functions: a set $\{\mathcal{Q}_0^{\pi_1}, \mathcal{Q}_0^{\pi_2}\}$ for $\beta = 0$[7] and a set $\{\mathcal{Q}_{-0.1}^{\pi_1}, \mathcal{Q}_{-0.1}^{\pi_2}\}$ for $\beta = -0.1$. The two GPI policies are then defined as $\pi_\beta(s) \in \arg\max_a \max_{i=1,2} \mathcal{Q}_\beta^{\pi_i}(s,a)$ for $\beta \in \{0, -0.1\}$. Finally, we generate the histogram of returns by simulating episodes of length $T = 35$, in which actions are selected from π_β, and computing the cumulative reward obtained on each episode.

[6] Please note that convergence of value iteration is guaranteed due to the existence of absorbing states and because the underlying MDP is ergodic.

[7] For $\beta = 0$, Algorithm 3 reduces to standard value iteration.

Algorithm 3 Value Iteration for Entropic Utility Maximization

Requires $\varepsilon_{exit} > 0$, $\gamma \in [0, 1]$, $\beta \in \mathbb{R}$, $\langle \mathcal{S}, \mathcal{A}, r, P \rangle \in \mathcal{M}$
for $s \in \mathcal{S}$, $a \in \mathcal{A}$ **do** $\mathcal{Q}(s, a) \leftarrow 0$
for $n = 1, 2 \ldots \infty$ **do**
    \\ Update $\mathcal{Q}(s, a)$ for all state-action pairs
    for $s \in \mathcal{S}$, $a \in \mathcal{A}$ **do**
        \\ Perform one iteration of (11) with the greedy policy derived from $\mathcal{Q}$
        $\mathcal{Q}'(s, a) \leftarrow 0$
        for $s' \in \mathcal{S}$ **do**
            if s' is terminal **then**
                $\text{target} \leftarrow r(s, a, s')$
            else
                $\text{target} \leftarrow r(s, a, s') + \gamma \max_b \mathcal{Q}(s', b)$
            end if
            $\mathcal{Q}'(s, a) \leftarrow \mathcal{Q}'(s, a) + P(s'|s, a)\, e^{\beta \times \text{target}}$
        end for
        $\mathcal{Q}'(s, a) \leftarrow \frac{1}{\beta} \log \mathcal{Q}'(s, a)$
    end for
    \\ Check for convergence in utility values
    $\varepsilon \leftarrow \max_{s,a} |\mathcal{Q}'(s, a) - \mathcal{Q}(s, a)|$
    if $\varepsilon < \varepsilon_{exit}$ **then return** $\mathcal{Q}'$
    \\ If not converged, then continue with value iteration
    for $s \in \mathcal{S}$, $a \in \mathcal{A}$ **do** $\mathcal{Q}(s, a) \leftarrow \mathcal{Q}'(s, a)$
end for

C.2 Four-Room

State and Action Spaces. The four-room domain consists of a family of discrete-state discrete-action MDPs $\mathcal{M}$ defined as follows. The world is defined as a set of discrete cells arranged in a grid of dimensions 13-by-13, such that at each time instant, the agent occupies a specific cell with some x- and y-coordinates $(p_x, p_y) \in \{0, \ldots 12\}^2$. As the agent explores the space, it can collect objects belonging to one of 3 possible classes. While the initial positions of these objects remains fixed throughout the experiment, their existence is determined by whether or not they have already been collected by the agent in a given episode (the same object cannot be picked up multiple times in a given episode). In our configuration, there are 6 instances of objects belonging to each class, for a total of $n_o = 18$ collectible objects. Therefore, the state space $\mathcal{S} = \{0, 1\}^{n_o} \times \{0, \ldots 12\}^2$ consists of the concatenation of the agent's current position (p_x, p_y) and a set of binary variables indicating whether or not each object has already been picked up by the agent. All objects are reset at the beginning of each episode. Actions are defined as $\mathcal{A} = \{\text{left}, \text{up}, \text{right}, \text{down}\}$ that move the agent to an adjacent cell in the corresponding direction. In the case that the destination cell lies outside the grid, then the agent remains in the current cell at the next time instant.

Reward Function and Risk. The goal cell 'G' provides a fixed reward of $+1$ and immediately terminates the episode upon entry. The reward r_c associated with each object class $c \in \{1, 2, 3\}$ is reset every time a new task begins, and is sampled from a uniform distribution on $[-1, +1]$. Occupying a trap cell that triggers at a particular time instant defines a failure, and is communicated to the agent by incurring a penalty of -2 and immediately terminating the episode. However, occupying a trap cell does not automatically guarantee a failure. Instead, a failure is only triggered with probability 0.05 independently at every time instant during which the agent occupies a trap cell. This additional reward stochasticity can be implemented without breaking the existing successor feature framework by introducing a fictitious terminal state s_f to indicate failure, which is reached at random when in cells marked 'X'. This state augmentation induces a modified MDP with a deterministic reward of -2 on arrival to state s_f, whose associated transitions are stochastic in nature. Crucially, this state augmentation transformation applies uniformly to all task instances, and thus does not break our assumptions about $\mathcal{M}$. We use a discount factor of $\gamma = 0.95$.

Features and Linear Reward Parameterization. Exact state features $\boldsymbol{\phi}(s, a, s')$ are provided directly to the agent. Specifically, we define $\phi_c(s, a, s')$ for every class of objects $c \in \{1, 2, 3\}$ to take the value 1 if the agent occupies a cell with an object of class c in state s' and 0 otherwise. Similarly, we define $\phi_g(s, a, s')$ to take the value 1 if s' corresponds to the goal cell and 0 otherwise. Unlike Barreto et al. [2], the four-room domain also contains an additional failure state with non-zero reward, as described above, and this must also be incorporated into the SF representation. This can be done by defining $\phi_f(s, a, s')$ that takes the value 1 if s' corresponds to the state s_f and 0 otherwise[8]. The state features $\boldsymbol{\phi} \in \mathbb{R}^5$ are then the concatenation of ϕ_c, ϕ_g and ϕ_f. These features are sparse, but can represent the reward functions of all possible task instances in $\mathcal{M}$ exactly. Finally, we define $\mathbf{w}_c = r_c$, $\mathbf{w}_g = 1$ and $\mathbf{w}_f = -2$, and it is now clear that $r(s, a, s') = \boldsymbol{\phi}(s, a, s')^\intercal \mathbf{w}$ holds.

Hyper-Parameters. Each time a new task is created, a new $\tilde{\boldsymbol{\psi}}^\pi$ and $\tilde{\Sigma}^\pi$ are created. The training loop of RaSFQL then proceeds according to Algorithm 1. We set $\alpha = 0.5$ and $\varepsilon = 0.12$, based on preliminary experiments for Q-learning. We also set $\bar{\alpha} = 0.1$ for learning $\tilde{\Sigma}^\pi$ and $\alpha_w = 0.5$ for learning $\mathbf{w}$ with gradient descent. Rollouts are limited to $T = 200$ steps for all algorithms.

Baseline. The baseline used for comparison is the *probabilistic policy reuse* framework of Fernández and Veloso [12] (PRQL), here adapted for learning risk-sensitive behaviors. In order to do this, we incorporate the *smart exploration* strategy of Gehring and Precup [16]. This strategy is fundamentally similar to our mean-variance approach, since it also incorporates second-moment or reward-variance information into action selection in a similar way. The controllability bonus $C^\pi(s, a)$ in each state-action pair is learned using a Q-learning approach by using the negative of the absolute Bellman residuals $-|\delta|$ as pseudo-rewards, and learned in parallel to the Q-values in practical implementations. The penalty for $C(s, a)$ is denoted as ω, and is fundamentally similar to β used by SFQL. The resulting algorithm, which we call RaPRQL, is described in Algorithm 4.

Baseline Hyper-Parameters. Similar to RaSFQL, every time a new task is created, a new Q^π and C^π are created for RaPRQL for learning new policies. We set $\alpha = 0.5$ for fair comparison with RaSFQL, and $\rho = 0.1$ based on the original implementation [16]. The performance is highly sensitive to the parameters η and τ used by PRQL. To select these two hyper-parameters, we follow Barreto et al. [2] and run a grid search for $\eta \in \{0.1, 0.3, 0.5\}$ and $\tau \in \{1, 10, 100\}$, selecting the combination of η and τ that resulted in the highest cumulative return over 128 task instances. This validation experiment is repeated for every value of ω.

C.3 Reacher

State and Action Spaces. The state space $\mathcal{S} \subset \mathbb{R}^4$ consists of the angles and angular velocities of the robotic arm's two joints. The two-dimensional action space $\mathcal{A} \subset [-1, +1]^2$ is discretized using 3 values per dimension, corresponding to maximum positive $(+1)$ and negative (-1) and zero torque for each actuator, resulting in a total of 9 possible actions. At the beginning of each episode, the angle of the central joint is sampled from a uniform distribution on $[-\pi, +\pi]$, while the angle of the outer joint is sampled from a uniform distribution on $[-\pi/2, +\pi/2]$, and the angular velocities are initialized to zero. Furthermore, state transitions are made stochastic by adding zero-mean Gaussian noise to actions with standard deviation 0.03, and then clipping the actions to $[-1, +1]$.

Reward Function and Risk. The reward received at each time step is $1 - 4\delta$, where δ is the Euclidean distance between the target position and the tip of the robotic arm. We define 12 target locations, of which 4 are used for training and the remaining 8 for testing. Furthermore, circular regions of radius $\delta_f = 0.06$ are placed around 6 of the 12 target locations (2 training and 4 testing) in which failures occur spontaneously with probability $p_f = 0.035$. Once a failure occurs, a cost of $c_f = 3$ is incurred and the episode continues without termination. This implies that the expected

[8] It is not practical to redefine the task space with the augmented state s_f in an actual implementation. Instead, we simulate this by providing the state features $\boldsymbol{\phi}$ with a binary variable indicating failure. This does not change the SF implementation, since the occurrence of a failure event can be deduced using the done flag (indicating arrival in a terminal state) and the state s'.

Algorithm 4 RaPRQL with Smart Exploration

Requires $m, T, N_e \in \mathbb{N}$, $\varepsilon, \eta \in [0,1]$, $\alpha, \rho, \tau > 0$, $\omega \in \mathbb{R}$, $M_1, \dots M_m \in \mathcal{M}$
for $t = 1, 2 \dots m$ **do**
  Initialize $Q^t(s,a), C^t(s,a)$ to small random or zero values
  for $k = 1, 2 \dots t$ **do** $\text{score}_k \leftarrow 0$, $\text{used}_k \leftarrow 0$
  $c \leftarrow t$
  $R \leftarrow 0$
  \\ Commence training on task M_t
  for $n_e = 1, 2 \dots N_e$ **do**
    Initialize M_t with initial state s
    for $h = 0, 1 \dots T$ **do**
      \\ Select actions according to Q-values plus controllability bonus
      if $c \neq t$ **then** use_prev_policy $\sim$ Bernoulli(η) **else** use_prev_policy $\leftarrow$ false
      if use_prev_policy **then**
        \\ Action is selected from π_c, the source policy being used
        $a \leftarrow \arg\max_b \{Q_h^c(s,b) + \omega C_h^c(s,b)\}$
      else
        \\ Action is selected from π_t, the policy being learned
        random_a $\sim$ Bernoulli(ε)
        if random_a **then** $a \sim$ Uniform$(\mathcal{A})$ **else** $a \leftarrow \arg\max_b \{Q_h^t(s,b) + \omega C_h^t(s,b)\}$
      end if
      Take action a in M_t and observe r and s'
      \\ Update the Q-values for the current task
      $\delta_h \leftarrow r + \max_b Q_{h+1}^t(s',b) - Q_h^t(s,a)$
      $Q_h^t(s,a) \leftarrow Q_h^t(s,a) + \alpha\delta_h$
      \\ Update the controllability bonus for the current task
      $C_h^t(s,a) \leftarrow C_h^t(s,a) + \alpha\rho(-|\delta_h| - C_h^t(s,a))$
      $R \leftarrow R + r$
      $s \leftarrow s'$
    end for
    \\ Update average return obtained by following policy π_c
    $\text{score}_c \leftarrow \frac{\text{score}_c \times \text{used}_c + R}{\text{used}_c + 1}$
    \\ Sample a new source policy
    for $k = 1, 2 \dots t$ **do** $p_k \leftarrow \frac{e^{\tau \times \text{score}_k}}{\sum_j e^{\tau \times \text{score}_j}}$
    $c \sim$ Multinomial$(p_1, p_2, \dots p_t)$
    $\text{used}_c \leftarrow \text{used}_c + 1$
    $R \leftarrow 0$
  end for
end for

reward, as a function of the distance δ, is[9]

$$R(\delta) = \begin{cases} 1 - 4\delta & \text{if } \delta > \delta_f \\ 1 - 4\delta - c_f \times p_f & \text{if } \delta \leq \delta_f. \end{cases}$$

Therefore, a rational[10] risk-neutral agent would prefer to enter inside the failure region if it holds that $1 - c_f \times p_f \geq 1 - 4\delta_f$, or in other words if

$$c_f \times p_f \leq 4\delta_f.$$

Clearly, given our choice of values for c_f, p_f and δ_f, the above condition holds in our setting. Setting up the reward structure and risk in this way makes it possible to control the trade-off between risk and reward, and thus the anticipated behavior of the agents, in a principled way. We also apply discounting of future reward using $\gamma = 0.9$.

[9] The reasoning here has simplified some of the aspects of the environment, ignoring the effects of multiple risk regions that could alter the trajectories, limited-length episodes and discounting.

[10] Of course, a rational agent would want to keep the tip as close to the target location as possible, and so would want $\delta = 0$.

Features and Linear Reward Parameterization. The state features are vectors $\boldsymbol{\phi}(s, a, s') \in \mathbb{R}^{13}$, in which the first 12 components consist of $1 - 4\delta_g$, where δ_g are the Euclidean distances to each of the goal locations g. The last component takes the value 1 if a failure event occurs and 0 otherwise. As done in the four-room experiment, state features are provided to the agent. However, target goal locations $\mathbf{w} \in \mathbb{R}^{13}$ are not learned in this instance, but provided directly to the agent as well. Specifically, we set $\mathbf{w}_g = 1$ for the goal with index g and $\mathbf{w}_{13} = c_f = -3$, and set all other elements to zero. This recovers the correct reward function $r(s, a, s')$ for all task instances as described above.

Hyper-Parameters and Learning Architectures. The overall training and testing procedures closely mimic Barreto et al. [2]. The successor features $\boldsymbol{\psi}^\pi$ and their distribution $\boldsymbol{\Psi}^\pi$ are represented as multi-layer perceptrons (MLP) with two hidden layers of size 256 and $\tanh$ non-linearities. The SFC51 and RaSFC51 architectures are generally identical and require output layers of dimensions $\mathbb{R}^{9\times51\times13}$, with a $\mathrm{softmax}$ activation function applied with respect to the second dimension. Similarly, C51 and RaC51 also require output layers but of dimensions $\mathbb{R}^{9\times51}$ and $\mathrm{softmax}$ applied with respect to the second dimension. For SFDQN, the output of the network is linear with dimensions $\mathbb{R}^{9\times13}$. We also use target networks for both SFC51/RaSFC51 and C51/RaC51, which are updated every $1{,}000$ transitions by copying weights from the learning networks. These target networks are only used for computing the bootstrapped return estimates. For SFC51, RaSFC51 and SFDQN, separate MLPs are used to learn each policy. To allow C51 and RaC51 to generalize across target locations, we apply *universal value function approximation* [35] and incorporate the target position into the state. This makes C51 essentially identical to the DQN baseline in Barreto et al. [2], except that DQN is replaced by C51. For C51-based agents, recall that the range of possible values of $\boldsymbol{\phi}$ must also be specified. For SFC51 and RaSFC51, we use $\phi^d_{min} = -1$ and $\phi^d_{max} = 1$ for $d = 1, 2 \ldots 12$ and use $\phi^{13}_{min} = 0$ and $\phi^{13}_{max} = 1$, which corresponds to a relatively tight bound for state features described in the previous paragraph. For C51 and RaC51, we set the bounds to $V_{min} = -30$ and $V_{max} = 10$, which corresponds to a tight bound for the discounted return. These intervals are discretized into $N = 51$ atoms for learning histograms, as recommended in the original paper [6].

Training and Testing Procedures. Agents are trained on all 4 training task instances sequentially one at a time, for $200{,}000$ time steps per task using an epsilon-greedy policy with $\varepsilon = 0.1$. Analogous to Barreto et al. [2], every sample is used to train all 4 policies simultaneously for SFC51, RaSFC51 and SFDQN. A randomized replay buffer of infinite capacity stores all previously-observed transitions $(s, a, \boldsymbol{\phi}, s')$ from all 4 training tasks, to avoid "catastrophic forgetting" of previously learned task instances. Each update of the network is based on a mini-batch of size 32 sampled uniformly from the replay buffer, and uses the Adam optimizer with a learning rate of 10^{-3}. Please note that these parameters, and those in the previous paragraph, are generally identical to those used in Barreto et al. [2]. Testing follows an epsilon-greedy policy with $\varepsilon = 0.03$ and greedy actions are selected according to risk-aware GPI, e.g. $a^* \in \arg\max_a \max_{i\in\{1,\ldots4\}} \{\tilde{\boldsymbol{\psi}}^{\pi_i}(s, a)^\intercal \mathbf{w}_j + \beta \mathbf{w}_j{}^\intercal \tilde{\Sigma}^{\pi_i}(s, a)\mathbf{w}_j\}$. Recall that test rewards $\mathbf{w}_j$ are provided to the agent. We set the episode length to $T = 500$ time steps for training and testing. All visualizations are based on estimating the test return at regular intervals of $5{,}000$ time steps, calculated as the average performance of 5 independent rollouts.

Normalization of Returns. Since the performance varies for different target locations, Barreto et al. [2] applies a normalization procedure to compare the performance between tasks in a fair manner. We apply the same procedure, by first training a standard C51 agent from scratch on each training and test task 10 times, and recording the average performance at the beginning and end of training, $\bar{G}_b$ and $\bar{G}_a$, respectively. The normalized return illustrated in all figures is then calculated as $G_n = (G - \bar{G}_b)/(\bar{G}_a - \bar{G}_b)$.

Physics Simulator. The physics simulator used for the reacher domain is provided by the open-source `pybullet` and `pybullet-gym` packages [50, 51]. We adapted the Python environment in the latter package to handle multiple target goal locations as required in our problem setting. Please note that this package is released under the MIT license.

C.4 Additional Details for Reproducibility

Reproducing Four-Room. The four-room experiment was run on an Alienware m17 R3, whose software and hardware specifications are provided in Table 3. Please note that while this machine has a GPU and tensorflow installed, neither were used in this experiment.

Component	Description	Quantity
Operating System	Windows 10 Home	
Python	3.8.5 (Anaconda)	
tensorflow	2.3.1	
System Memory	32 GB	
Hard Disk	953.9GB	1
CPU	Intel i7-10875H @ 2.30GHz (turbo-boost @ 5.1GHz)	1
GPU	Nvidia RTX 2080 Super 8GB	1

Table 3: Software and hardware configuration used to run all experiments for the four-room domain.

Reproducing Reacher. The reacher experiment was run on a Lenovo ThinkStation P920 workstation, whose software and hardware specifications are described in Table 4.

Component	Description	Quantity
Operating System	Ubuntu 18.04	
Python	3.8.5	
tensorflow	2.4.0	
System Memory	187 GB	
Hard Disk	953.9GB	5
CPU	Intel Xeon Gold 6234 @ 3.30GHz (turbo-boost @ 4GHz)	32
GPU	Nvidia Quadro RTX 8000 48GB	2

Table 4: Software and hardware configuration used to run all experiments for the reacher domain.

Other Factors. Please note that seeds were not fixed during the experiment but generated in each trial using Python's default seed generation algorithm. This allows us to average the performance of all algorithms over different seed values and initializations. No internal modifications to the Python environment nor to any of its installed packages were made. No effort to overclock the machines' CPUs or GPUs beyond their factory settings were made in order to decrease the overall computation time (see below).

Computation Time. The majority of the computation time in running the experiment was allocated to the reacher domain, partially because of the size of the network architectures required to learn meaningful policies (2 hidden layers consisting of 256 neurons), and the number of samples required to draw meaningful conclusions for all baselines. The computation time is considerably greater for RaSFC51 (around 28-36 hours per trial) than it is for RaC51 (around 6-8 hours per trial), which is expected since the former must train 4 neural networks while the latter must train only one. This could potentially lead to negative environmental impacts if the model is to be deployed on complex problems in real-world settings. At the same time, the potential speed-ups demonstrated by RaSFC51 as compared to RaC51 could reduce the *overall* training time considerably and offset the total energy requirement of learning policies with a satisfactory variance-adjusted return. Parallelization of the training loop could also be beneficial and provide significant time and cost savings in practice.

D Additional Ablation Studies and Plots

In this section, we include the full details and results of the ablation studies described in the main text, and additional analysis that had to be left out of the main paper due to space limitations.

D.1 Four-Room

Effect of Varying β. We can study the effect of β on the return performance and risk-sensitivity of the learned behaviors by repeating the four-room experiment (Appendix C.2) for various values of β. In particular, we trained RaSFQL for $\beta \in \{0, -1/2, -1, -2, -4\}$ (ω for RaPRQL), and recorded

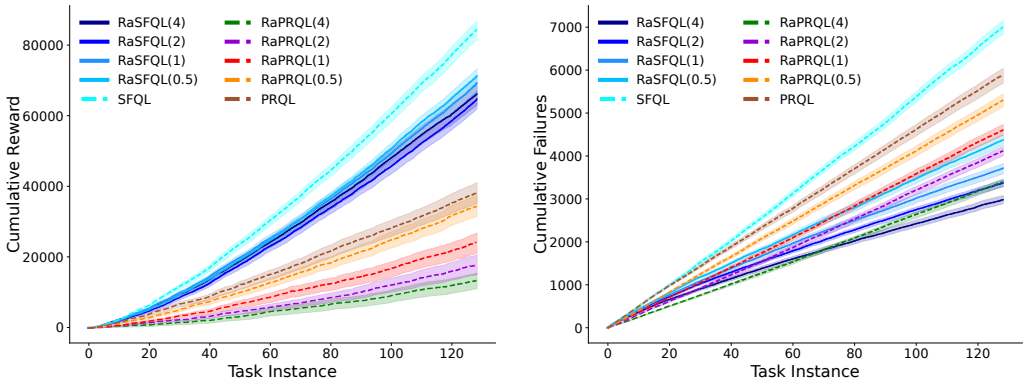

Figure 9: **Left:** cumulative reward collected across all training tasks in the four-room domain, for various values of β for RaSFQL (ω for RaPRQL). **Right:** cumulative number of failures across all training tasks in the four-room domain, for various values of β, ω. Please note that legend entries in parentheses indicate the negative values of β and ω. Shaded error bars indicate one standard error over 30 independent runs of each algorithm.

the cumulative reward and number of failures across all 128 training task instances. The results of these experiments are summarized in Figure 9. We see that the performance of RaSFQL degrades gracefully as β decreases (a relative drop in cumulative reward of approximately 25% is observed when β is decreased from 0 to -4), while the corresponding degradation for RaPRQL is considerably more pronounced (a relative drop in cumulative reward of roughly 75% is observed for an identical change in ω). Meanwhile, the number of cumulative failures of RaSFQL is generally lower than RaPRQL for every pair of identical values of β and ω. In fact, for $\beta \in \{-1, -2, -4\}$, the cumulative numbers of failures are increasing at sub-linear rates, which implies that risk-avoidance behavior is becoming more prominent as the number of training task instances increases.

Examination of Learned Behaviors. In order to better understand the kind of risk-averse behaviors being learned, we instantiated 27 novel test task instances by enumerating $\mathbf{w}_i \in \{-1, 0, 1\}$ for every object class $i = 1, 2, 3$. We then tested the performance of the GPI policy obtained from the training procedure described in the main paper, by simulating 100 rollouts following the epsilon-greedy policy with $\varepsilon = 0.1$ on each of the test tasks. Please note that no training was ever performed on the test tasks. The state visitation counts across all 100 trajectories were computed for every task instance and arranged in a 3D-lattice as indicated in Figure 10. We repeated this procedure twice: once for RaSFQL with $\beta = -2$ and once for SFQL. Interestingly, RaSFQL and SFQL learn behaviors that are similar to each other when looking at the same task, but each of them exploits different regions of the state space depending on the reward. However, RaSFQL almost always learns to avoid the dangerous objects in the bottom-left and top-right rooms, whereas SFQL does not necessarily do so. This discrepancy is most evident, for instance, when $\mathbf{w}_2 = 1$ and for $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) = (1, -1, -1)$ and $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) = (1, -1, 0)$.

D.2 Reacher

Benefit of Distributional RL for Learning Successor Features. The left plot in Figure 11 illustrates the normalized test return, averaged across all test tasks, for SFC51 and the original SFDQN implementation of Barreto et al. [2]. Both agents are risk-neutral in this comparison. It is likely that learning the full distribution of SF returns provides additional stability of the Bellman backups in stochastic domains, and thus allows SFs to inherit the advantages of distributional RL for maximizing expected return [6].

Effect of Varying β. Unlike the four-room domain, we saw that the number of failures of RaSFC51 was modestly *greater* than RaC51 for the same values of β. In order to better understand how efficiently the trade-off between risk and reward is handled by these two algorithms, we decided to compute an alternative measure of return by dividing the normalized return by the total number

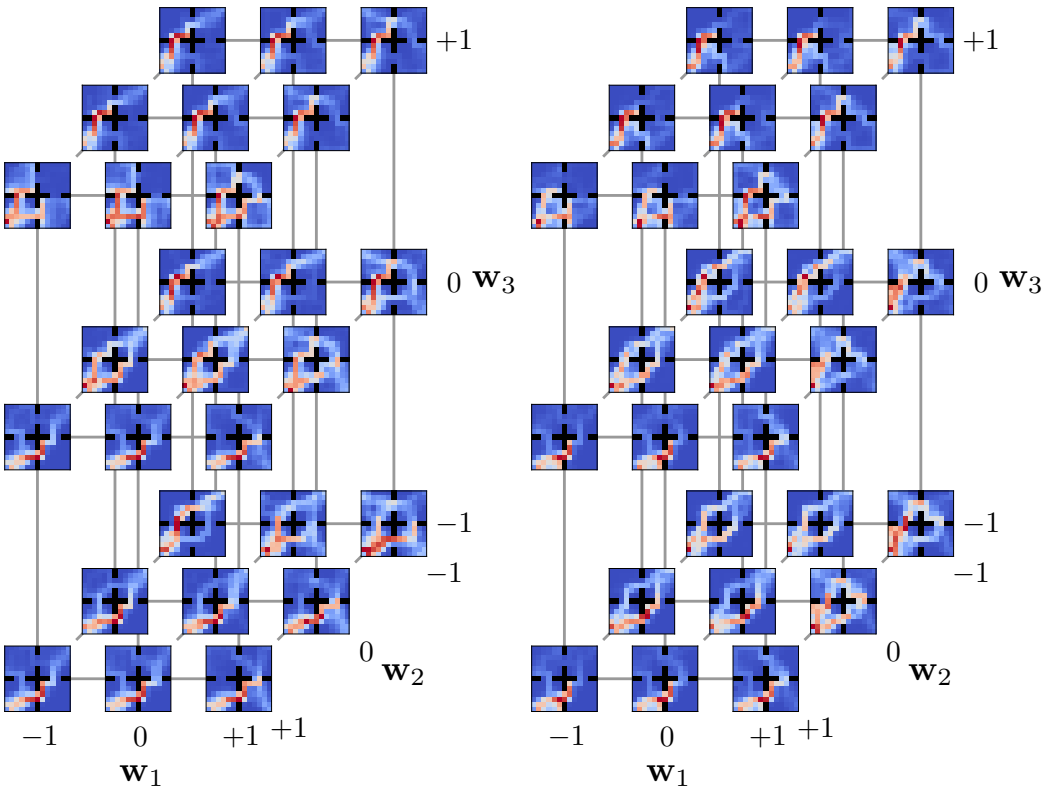

Figure 10: Visitation counts over 100 rollouts from behavior/training policies (epsilon-greedy with $\varepsilon = 0.12$) derived from GPI after training on all 128 task instances. The behavior policies are illustrated on 27 novel task instances in which the reward $\mathbf{w}$ varies, e.g. $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \{-1, 0, 1\}$. **Left:** Behavior policies derived from GPI for RaSFQL with $\beta = -2$. **Right:** Behavior policies derived from GPI for standard SFQL. Visitation counts are averaged over 30 independent runs for each algorithm.

of failures. Intuitively, this quantity provides an estimate of the expected reward collected between between successive failure events. The right plot contained in Figure 11, which compares this quantity for RaSFC51 and RaC51 for different values of β, shows that RaSFC51 is actually much more efficient at managing the trade-off between risk and reward for larger-magnitude values of β. This is not surprising, given that RaSFC51 can obtain much high return than RaC51 for a comparable number of failures, when $\beta = -3$ and $\beta = -4$. In fact, for $\beta = -4$, the number of failures of RaSFC51 and RaC51 become equivalent as both methods learn sufficiently conservative policies. Even in this case, successor features combined with GPI allow RaSFC51 to generalize much better on novel tasks than RaC51.

Examination of Learned Behaviors. As suggested in the main text, one possible conjecture is that RaSFC51 learns to correctly solve the test tasks, requiring the robotic arm to hover closer to the edge of the risky areas, while RaC51 does not. The presence of environment stochasticity, errors in function approximation, and the stochasticity of the epsilon-greedy policy used during testing could exacerbate this. Comparing rollouts of successfully-learned behavior produced by RaSFC51 and RaC51 in training tasks in Figure 12, and testing tasks in Figure 13, confirms that RaSFC51 is much better at task generalization than RaC51. Here, RaSFC51 learns to hover right at the boundaries of the high-variance regions, preferring not to enter them whenever possible. On the other hand, risk-neutral SFC51 is completely unaware of the risky areas, focusing exclusively on minimizing the distance to the target location, but is able to successfully locate all targets. RaC51 demonstrates similar risk-aware behaviors as RaSFC51, but cannot reliably locate the target on some of the test task instances.

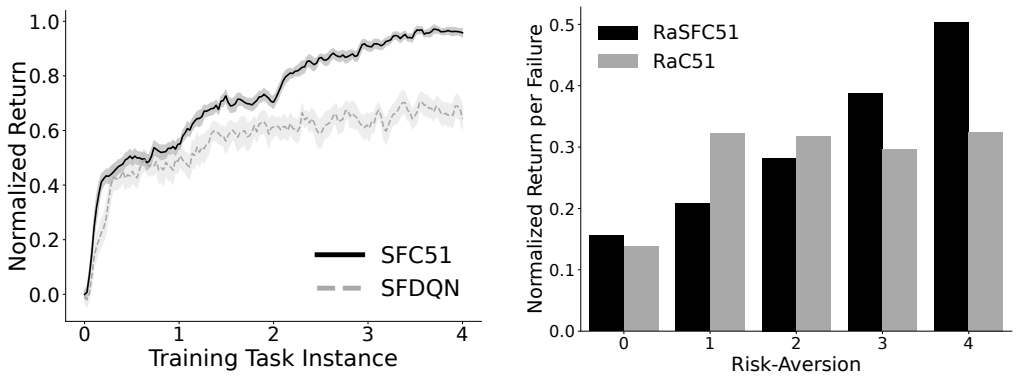

Figure 11: Additional ablation studies for the reacher domain, that were left out of the main paper due to space limitations. **Left:** Normalized average test return for the reacher domain, showing the improvement obtained by replacing DQN by C51 as a function approximator for SFs. **Right:** In order to assess the trade-off between return and possibility of failure, we divide the normalized return, averaged across all test tasks, by the total number of failures for each value of β. The resulting measure is compared between RaSFC51 and RaC51. The x-axis indicates the negative values of β.

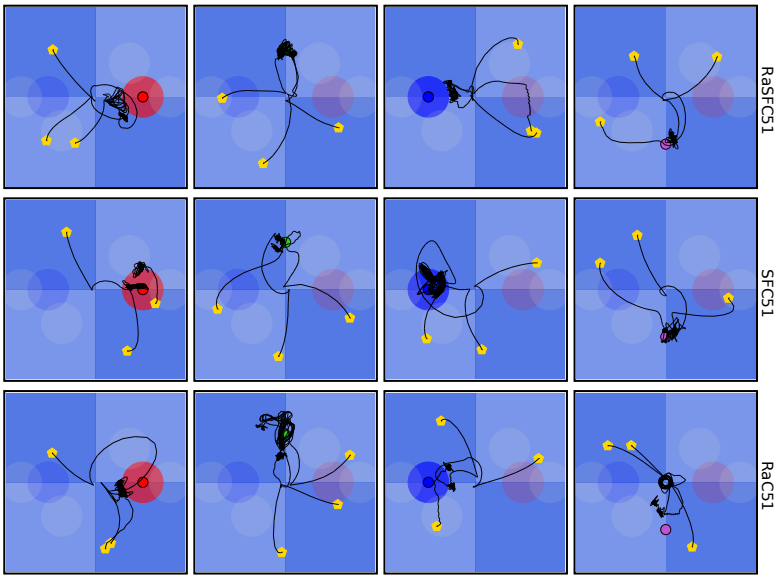

Figure 12: Evolutions of the robotic arm tip position in three successful rollouts of the reacher domain according to the GPI policy obtained after training on all 4 tasks. Here, all 4 training tasks are shown.

Examination of Learned Covariance. A similar conclusion can also be drawn by observing the heat-maps of the learned mean-variance objectives in Figure 15. For SFC51, these objectives take the highest values precisely at the target locations, whereas for RaSFC51 these take the highest values slightly away from the targets in regions of low volatility. This is expected as the utility of hovering very close to a target location centered in a risky region should be lower than hovering outside the risky region, for a sufficiently risk-averse agent. Moreover, the first 4 rows correspond to training task values and the last 8 correspond to test task values. Because a similar pattern described above can also be observed in test tasks, the ability of SFs to generalize expected return estimates to novel task instances also extends to higher-order sufficient statistics, namely the variance of return. Finally, the aggregated plots located in the top half in Figure 16 show that RaSFC51 learns the return variance correctly after having trained on all 4 task instances. On the other hand, the SFDQN architecture that learns the covariance using the residual method (8) is unable to learn the variance correctly, likely due to the propagation of errors and overestimation bias in $\tilde{\boldsymbol{\psi}}^{\pi_i}(s,a)$ as discussed in the main paper.

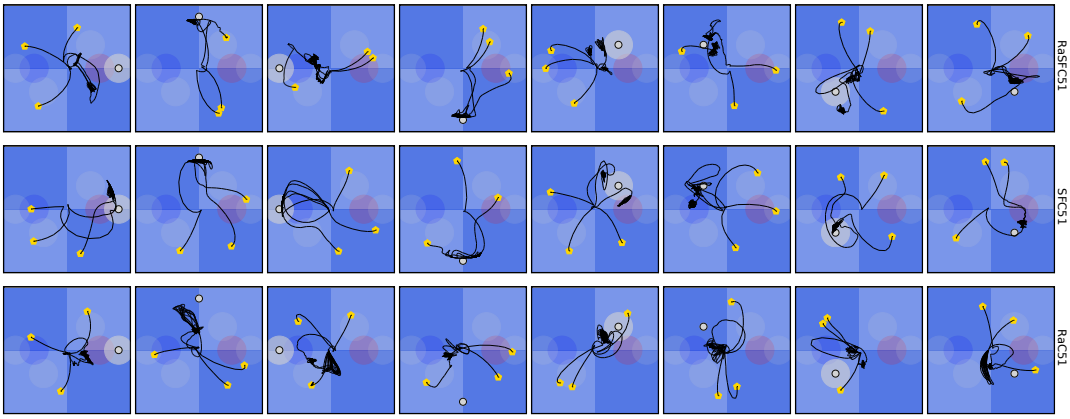

Figure 13: Evolutions of the robotic arm tip position in three successful rollouts of the reacher domain according to the GPI policy obtained after training on all 4 tasks. Here, all 8 test tasks are shown.

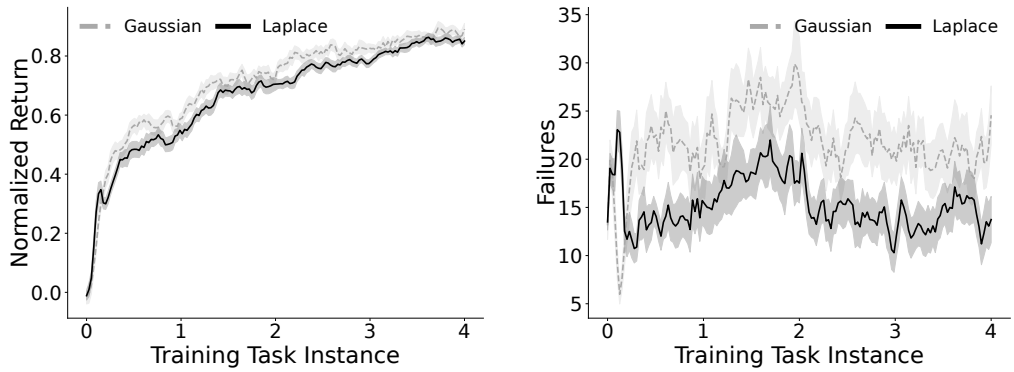

Figure 14: Ablation study for varying the distributional assumption used in the mean-variance approximation in the reacher domain. Gaussian implies the usual mean-variance approximation with the assumption of normally-distributed returns, while Laplace assumes a Laplace distribution for returns. Both approximations use $\beta = -2$. **Left:** Normalized average test return across all test tasks. **Right:** Total failures across all test tasks.

Effect of Varying the Distributional Assumption of Returns. A final ablation study we conducted was to assess the stability of the task generalization of risk-aware SFs under different distributional assumptions of the cumulative reward. As discussed in Section A.4, the mean-variance approximation used in this work results from making a Gaussian assumption on the cumulative reward distribution, which may not always be closely-aligned with reality. To assess the flexibility of our approach, we evaluate the performance of RaSFC51 when the distribution of cumulative reward is assumed to follow a Laplace approximation (see Table 2 for details). We repeat the reacher experiment with this modification and report the results (test return and test failures) in Figure 14. As we can see, the normalized test return under both distributional assumptions is essentially unchanged. However, the number of test failures is reduced to almost half its original amount by simply replacing the normal distribution with the Laplace distribution for modeling returns. By inspecting and comparing the penalties of these two distributional assumptions in Figure 8, this result is expected, since the Laplace distribution is better-suited for modeling heavy-tailed risks and thus penalizes the variance more than the normal distribution. This further suggests that the assumptions used for modeling the distributions of returns in GPE can have a profound effect on the effectiveness of transfer and the risk-sensitivity of the learned policies. Thus, these distributional assumptions can – and perhaps should – be treated as important hyper-parameters or design parameters in future experiments. Designing and characterizing suitable distributional assumptions for different classes of problems could be an interesting topic for further investigation.

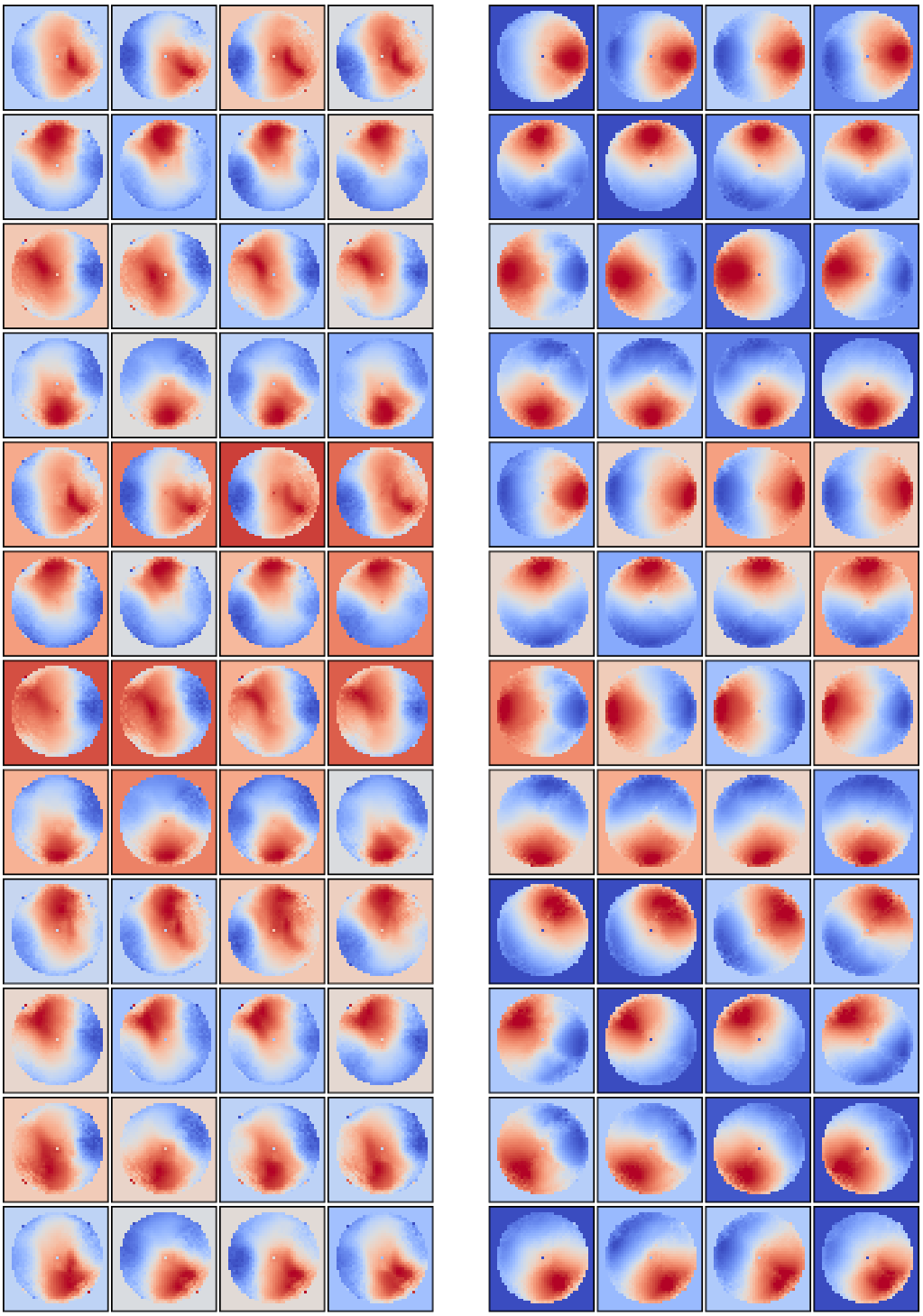

Figure 15: Each plot located in column i and row j illustrates the value of the mean-variance objective $\boldsymbol{\psi}^{\pi_i}(s,a)^\intercal \mathbf{w}_j - \mathbf{w}_j{}^\intercal \Sigma^{\pi_i}(s,a)\mathbf{w}_j$ as a function of the robotic arm tip position in (x,y) coordinates for the reacher domain, after training each agent on all 4 tasks. In other words, the first 4 rows illustrate the value functions learned on the training task instances, while the last 8 rows illustrate the value functions learned on the test tasks. **Left:** mean-variance objective computed by RaSFC51 with $\beta = -3$. **Right:** mean-variance objective computed by SFC51.

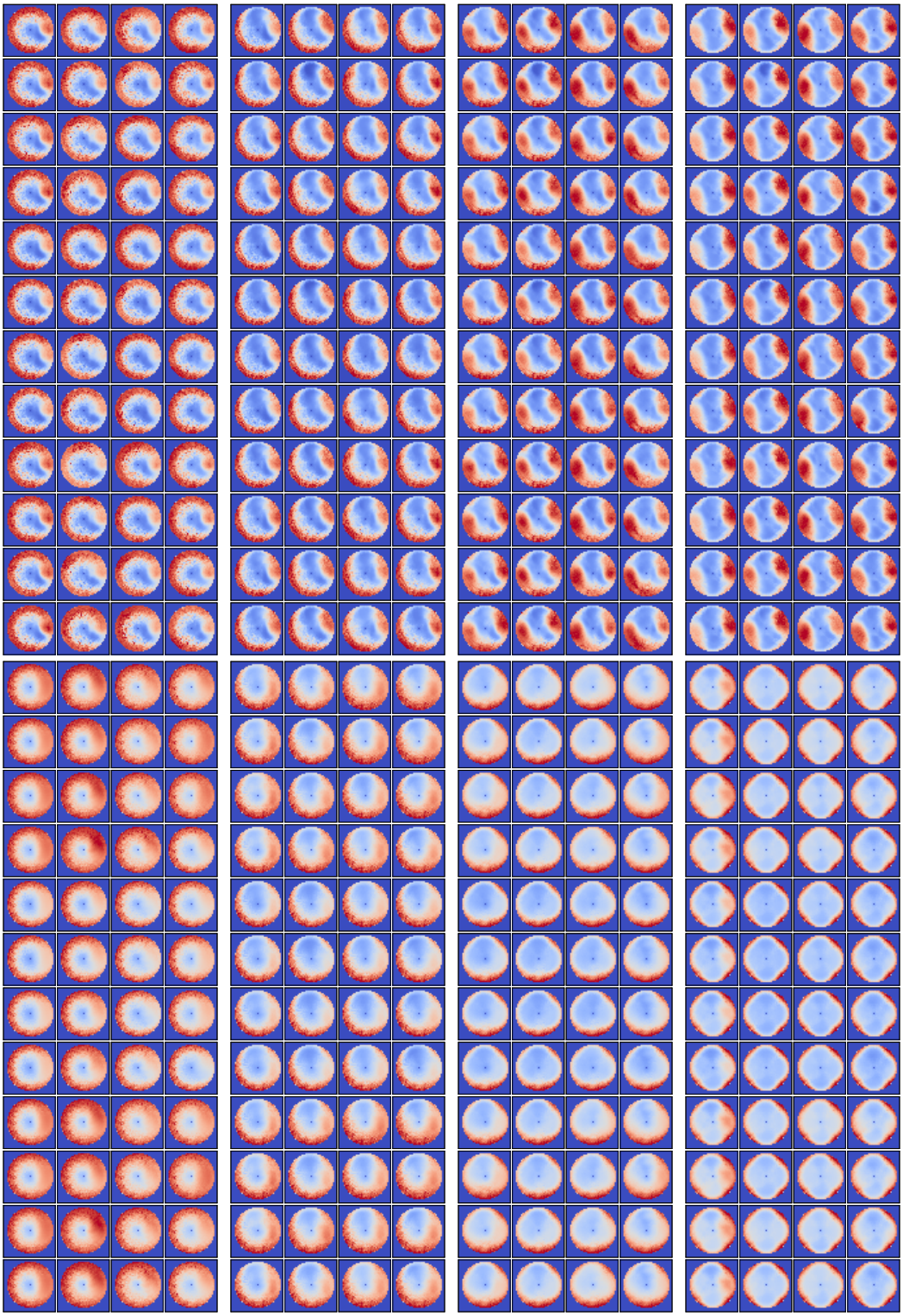

Figure 16: Each plot located in column i and row j illustrates the value of the variance $\mathbf{w}_j{}^\intercal \Sigma^{\pi_i}(s,a)\mathbf{w}_j$ as a function of the robotic arm tip position in (x,y) coordinates for the reacher domain, after training on 1, 2, 3 and 4 source tasks (respectively, left to right). **Top:** variance computed by RaSFC51 with $\beta = -3$. **Bottom:** variance computed by SFDQN using (8).

E Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The abstract and contributions paragraph in the introduction clearly state the paper's novel contributions. We discuss that the theoretical results could generalize to other concave utility functions. Our experimental design closely matches the one in the original paper on successor features, and we also mention that our approach can take advantage of other improvements in successor features. Thus, we expect our empirical results to follow through for domains with similar structure and complexity to those introduced in subsequent literature on successor features.
   (b) Did you describe the limitations of your work? [Yes] The conclusion summarizes several limitations associated with assumptions that are introduced when made in the paper. Specifically, we describe how the error term ε propagates as a result of relying on the mean-variance approximation in GPE in Section 3.3. We also point out the independence assumption used in the SFC51 and RaSFC51 architectures in Section 4. Finally, in the conclusion we mention that our approach is specific to the entropic utility objective, that is largely based on estimating risk using the variance of returns.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We do not foresee any direct societal impacts of this work. However, we do briefly discuss computation time and the need to train large models in Appendix C.4.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All assumptions are stated at the beginning of each theoretical result.
   (b) Did you include complete proofs of all theoretical results? [Yes] All complete proofs are provided in Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Instructions for replicating the domains and how models are created and trained are provided in detail in Appendix C. A zip file of our code is also provided.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All hyper-parameters and how they were selected, along with other key details regarding training, are provided in Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All plots clearly show error bars, which represent one standard error around the mean.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Appendix C.4 describes the computing infrastructures used, including both hardware and software, to help reproduce the experiment. We also include a paragraph describing the computation time to run a trial of the reacher domain, which represented an overwhelming majority of the overall computation time.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Our implementation of the reacher domain is based on the pybullet-gym package as described in Appendix C.3.
   (b) Did you mention the license of the assets? [Yes] We mentioned that the package above is MIT licensed.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is provided as a zip file with the submission.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We do not use data.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We do not use data.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

[48] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[49] Kris Boudt, Dries Cornilly, and Tim Verdonck. A coskewness shrinkage approach for estimating the skewness of linear combinations of random variables. *Journal of Financial Econometrics*, 18(1):1–23, 2020.

[50] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[51] Benjamin Ellenberger. Pybullet gymperium. `https://github.com/benelot/pybullet-gym`, 2018–2019.

[52] Abhijit A Gosavi, Sajal K Das, and Susan L Murray. Beyond exponential utility functions: A variance-adjusted approach for risk-averse reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 1–8. IEEE, 2014.

[53] Wenjie Huang and William B Haskell. Stochastic approximation for risk-aware markov decision processes. *IEEE Transactions on Automatic Control*, 2020.

[54] Fabio Maccheroni, Massimo Marinacci, and Aldo Rustichini. Ambiguity aversion, robustness, and the variational representation of preferences. *Econometrica*, 74(6):1447–1498, 2006.

[55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[56] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Parametric return density estimation for reinforcement learning. In *UAI*, pages 368–375, 2010.

[57] Joel Owen and Ramon Rabinovitch. On the class of elliptical distributions and their applications to the theory of portfolio choice. *The Journal of Finance*, 38(3):745–752, 1983.

[58] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[59] Tomer Shushi. Generalized skew-elliptical distributions are closed under affine transformations. *Statistics & Probability Letters*, 134:1–4, 2018. ISSN 0167-7152. doi: https://doi.org/10.1016/j.spl.2017.10.012.

[60] Alexander L Strehl and Michael L Littman. A theoretical analysis of model-based interval estimation. In *ICML*, pages 856–863, 2005.

[61] D Michael Titterington, Adrian FM Smith, and UE Makov. *Statistical analysis of finite mixture distributions*, volume 198. John Wiley & Sons Incorporated, 1985.

[62] Eric W. Weisstein. Cumulant. from mathworld—a wolfram web resource. URL `https://mathworld.wolfram.com/Cumulant.html`. Last visited on 13/4/2021.