Mean-Variance Policy Iteration for Risk-Averse Reinforcement Learning

Shangtong Zhang *
University of Oxford

Bo Liu
Auburn University

Shimon Whiteson
University of Oxford

Abstract

We present a mean-variance policy iteration (MVPI) framework for risk-averse control in a discounted infinite horizon MDP. MVPI enjoys *great flexibility* in that any policy evaluation method and risk-neutral control method can be dropped in for risk-averse control off the shelf, in both on- and off-policy settings. This flexibility reduces the gap between risk-neutral control and risk-averse control and is achieved by working on a novel augmented MDP directly. We propose risk-averse TD3 as an example instantiating MVPI, which outperforms vanilla TD3 and many previous risk-averse control methods in challenging Mujoco robot simulation tasks under a risk-aware performance metric. This risk-averse TD3 is the first to introduce deterministic policies and off-policy learning into risk-averse reinforcement learning, both of which are key to the performance boost we show in Mujoco domains.

1 Introduction

One fundamental task in reinforcement learning (RL, Sutton and Barto 2018) is control, in which we seek a policy that maximizes certain performance metrics. In risk-neutral RL, the performance metric is usually the expectation of some random variable, for example, the expected total (discounted or undiscounted) reward (Puterman, 2014; Sutton and Barto, 2018). We, however, sometimes want to minimize certain risk measures of that random variable while maximizing its expectation. For example, a portfolio manager usually wants to reduce the risk of a portfolio while maximizing its return. Risk-averse RL is a framework for studying such problems.

Although many real-world applications can potentially benefit from risk-averse RL, e.g., pricing (Wang, 2000), healthcare (Parker, 2009), portfolio management (Lai et al., 2011), autonomous driving (Matthaeia et al., 2015), and robotics (Majumdar and Pavone, 2020), the development of risk-averse RL largely falls behind risk-neutral RL. Risk-neutral RL methods have enjoyed superhuman performance in many domains, e.g., Go (Silver et al., 2016), protein design (Senior et al., 2018), DoTA (OpenAI, 2018), and StarCraft II (Vinyals et al., 2019), while no human-level performance has been reported for risk-averse RL methods in real-world applications. Risk-neutral RL methods have enjoyed stable off-policy learning (Watkins and Dayan, 1992; Maei, 2011; Fujimoto et al., 2018; Haarnoja et al., 2018), while state-of-the-art risk-averse RL methods, e.g., Xie et al. (2018); Bisi et al. (2019), still require on-policy samples. Risk-neutral RL methods have exploited deep neural network function approximators and distributed training (Mnih et al., 2016; Espeholt et al., 2018), while tabular and linear methods still dominate the experiments of risk-averse RL literature (Tamar et al., 2012; Prashanth and Ghavamzadeh, 2013; Xie et al., 2018; Chow et al., 2018). Such a big gap between risk-averse RL and risk-neutral RL gives rise to a natural question: *can we design a meta algorithm that can easily leverage recent advances in risk-neutral RL for risk-averse RL?* In this paper, we give an affirmative answer via the mean-variance policy iteration (MVPI) framework.

*Correspondence to `shangtong.zhang@cs.ox.ac.uk`

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2020.

Although many risk measures have been used in risk-averse RL, in this paper, we mainly focus on variance (Sobel, 1982; Mannor and Tsitsiklis, 2011; Tamar et al., 2012; Prashanth and Ghavamzadeh, 2013; Xie et al., 2018) given its advantages in interpretability and computation (Markowitz and Todd, 2000; Li and Ng, 2000). Such an RL paradigm is usually referred to as mean-variance RL, and previous mean-variance RL methods usually consider the variance of the total reward random variable (Tamar et al., 2012; Prashanth and Ghavamzadeh, 2013; Xie et al., 2018). Recently, Bisi et al. (2019) propose a reward-volatility risk measure that considers the variance of a per-step reward random variable, which bounds the variance of the total reward from above, indicating that minimizing the variance of the per-step reward implicitly minimizes the variance of the total reward. Bisi et al. (2019) also show that the variance of the per-step reward can better capture the short-term risk than the variance of the total reward and usually leads to smoother trajectories.

In this paper, we further argue that optimizing the variance of the per-step reward as a proxy for the variance of the total reward is easier than optimizing the variance of the total reward directly, and therefore develop MVPI under the per-step reward perspective. MVPI enjoys great flexibility in that *any policy evaluation method and risk-neutral control method can be dropped in for risk-averse control off the shelf, in both on- and off-policy settings*. Key to the flexibility of MVPI is that it works on an augmented MDP directly, which we make possible by introducing the Fenchel duality and block cyclic coordinate ascent to solve a policy-dependent reward issue (Papini et al., 2018). This issue refers to a requirement to solve an MDP whose reward function depends on the policy being followed, i.e., the reward function of this MDP is nonstationary. Consequently, standard tools from the MDP literature are not applicable. We propose risk-averse TD3 as an example instantiating MVPI, which outperforms vanilla TD3 (Fujimoto et al., 2018) and many previous mean-variance RL methods (Tamar et al., 2012; Prashanth and Ghavamzadeh, 2013; Xie et al., 2018; Bisi et al., 2019) in challenging Mujoco robot simulation tasks in terms of a risk-aware performance metric. To the best of our knowledge, we are the first to benchmark mean-variance RL methods in Mujoco domains, a widely used benchmark for robotic-oriented RL research, and the first to bring off-policy learning and deterministic policies into mean-variance RL.

2 Mean-Variance RL

We consider an infinite horizon MDP with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a bounded reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a transition kernel $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, an initial distribution $\mu_0 : \mathcal{S} \to [0, 1]$, and a discount factor $\gamma \in [0, 1]$. The initial state S_0 is sampled from μ_0. At time step t, an agent takes an action A_t according to $\pi(\cdot|S_t)$, where $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the policy followed by the agent. The agent then gets a reward $R_{t+1} \doteq r(S_t, A_t)$ and proceeds to the next state S_{t+1} according to $p(\cdot|S_t, A_t)$. In this paper, we consider a deterministic reward setting for the ease of presentation, following Chow (2017); Xie et al. (2018). The return at time step t is defined as $G_t \doteq \sum_{i=0}^{\infty} \gamma^i r(S_{t+i}, A_{t+i})$. When $\gamma < 1$, G_t is always well defined. When $\gamma = 1$, to ensure G_t remains well defined, it is usually assumed that all polices are proper (Bertsekas and Tsitsiklis, 1996), i.e., for any policy π, the chain induced by π has some absorbing states, one of which the agent will eventually go to with probability 1. Furthermore, the rewards are always 0 thereafter. For any $\gamma \in [0, 1]$, G_0 is the random variable indicating the total reward, and we use its expectation $J(\pi) \doteq \mathbb{E}_{\mu_0, p, \pi}[G_0]$, as our primary performance metric. In particular, when $\gamma = 1$, we can express G_0 as $G_0 = \sum_{t=0}^{T-1} r(S_t, A_t)$, where T is a random variable indicating the first time the agent goes to an absorbing state. For any $\gamma \in [0, 1]$, the state value function and the state-action value function are defined as $v_\pi(s) \doteq \mathbb{E}[G_t|S_t = s]$ and $q_\pi(s, a) \doteq \mathbb{E}[G_t|S_t = s, A_t = a]$ respectively.

Total Reward Perspective. Previous mean-variance RL methods (Prashanth and Ghavamzadeh, 2013; Tamar et al., 2012; Xie et al., 2018) usually consider the variance of the total reward. Namely, they consider the following problem:

$$\max_\theta \mathbb{E}[G_0] \quad \text{subject to} \quad \mathbb{V}(G_0) \leq \xi, \tag{1}$$

where $\mathbb{V}(\cdot)$ indicates the variance of a random variable, ξ indicates the user's tolerance for variance, and π is parameterized by θ. In particular, Prashanth and Ghavamzadeh (2013) consider the setting $\gamma < 1$ and convert (1) into an unconstrained saddle-point problem: $\max_\lambda \min_\theta L_1(\theta, \lambda) \doteq -\mathbb{E}[G_0] + \lambda(\mathbb{V}(G_0) - \xi)$, where λ is the dual variable. Prashanth and Ghavamzadeh (2013) use stochastic gradient descent to find the saddle-point of $L_1(\theta, \lambda)$. To estimate $\nabla_{\theta,\lambda} L_1(\theta, \lambda)$, they propose two simultaneous perturbation methods: simultaneous perturbation stochastic approximation and smoothed functional (Bhatnagar et al., 2013), yielding a three-timescale algorithm. Empirical success is observed in a simple traffic control MDP. Tamar et al. (2012) consider the setting $\gamma = 1$. Instead of using the saddle-point formulation in Prashanth and Ghavamzadeh (2013), they consider the following unconstrained problem: $\max_\theta L_2(\theta) \doteq \mathbb{E}[G_0] - \lambda g(\mathbb{V}(G_0) - \xi)$, where $\lambda > 0$ is a hyperparameter to be tuned and $g(\cdot)$ is a penalty function, which they define as $g(x) \doteq (\max\{0, x\})^2$. The analytical expression of $\nabla_\theta L_2(\theta)$ they provide involves a term $\mathbb{E}[G_0]\nabla_\theta \mathbb{E}[G_0]$. To estimate this term, Tamar et al. (2012) consider a two-timescale algorithm and keep running estimates for $\mathbb{E}[G_0]$ and $\mathbb{V}[G_0]$ in a faster timescale, yielding an episodic algorithm. Empirical success is observed in a simple portfolio management MDP. Xie et al. (2018) consider the setting $\gamma = 1$ and set the penalty function $g(\cdot)$ in Tamar et al. (2012) to the identity function. With the Fenchel duality $x^2 = \max_y(2xy - y^2)$, they transform the original problem into $\max_{\theta,y} L_3(\theta, y) \doteq 2y(\mathbb{E}[G_0] + \frac{1}{2\lambda}) - y^2 - \mathbb{E}[G_0^2]$, where y is the dual variable. Xie et al. (2018) then propose a solver based on stochastic coordinate ascent, yielding an episodic algorithm.

Per-Step Reward Perspective. Recently Bisi et al. (2019) propose a reward-volatility risk measure, which is the variance of a per-step reward random variable R. In the setting $\gamma < 1$, it is well known that the expected total discounted reward can be expressed as $J(\pi) = \frac{1}{1-\gamma}\sum_{s,a} d_\pi(s, a)r(s, a)$, where $d_\pi(s, a)$ is the normalized discounted state-action distribution: $d_\pi(s, a) \doteq (1 - \gamma)\sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s, A_t = a|\mu_0, \pi, p)$. We now define the per-step reward random variable R, a discrete random variable taking values in the image of r, by defining its probability mass function as $p(R = x) = \sum_{s,a} d_\pi(s, a)\mathbb{I}_{r(s,a)=x}$, where $\mathbb{I}$ is the indicator function. It follows that $\mathbb{E}[R] = (1 - \gamma)J(\pi)$. Bisi et al. (2019) argue that $\mathbb{V}(R)$ can better capture short-term risk than $\mathbb{V}(G_0)$ and optimizing $\mathbb{V}(R)$ usually leads to smoother trajectories than optimizing $\mathbb{V}(G_0)$, among other advantages of this risk measure. Bisi et al. (2019), therefore, consider the following objective:

$$J_\lambda(\pi) \doteq \mathbb{E}[R] - \lambda\mathbb{V}(R). \tag{2}$$

Bisi et al. (2019) show that $J_\lambda(\pi) = \mathbb{E}[R - \lambda(R - \mathbb{E}[R])^2]$, i.e., to optimize the risk-aware objective $J_\lambda(\pi)$ is to optimize the canonical risk-neutral objective of a new MDP, which is the same as the original MDP except that the new reward function is

$$r'(s, a) \doteq r(s, a) - \lambda\big(r(s, a) - (1 - \gamma)J(\pi)\big)^2.$$

Unfortunately, this new reward function r' depends on the policy π due to the occurrence of $J(\pi)$, implying the reward function is actually nonstationary. By contrast, in canonical RL settings (e.g., Puterman (2014); Sutton and Barto (2018)), the reward function is assumed to be stationary. We refer to this problem as the *policy-dependent-reward* issue. Due to this issue, the rich classical MDP toolbox cannot be applied to this new MDP easily, and the approach of Bisi et al. (2019) *does not and cannot* work on this new MDP directly.

Bisi et al. (2019) instead work on the objective Eq (2) directly *without* resorting to the augmented MDP. They propose to optimize a performance lower bound of $J_\lambda(\pi)$ by extending the performance difference theorem (Theorem 1 in Schulman et al. (2015)) from the risk-neutral objective $J(\pi)$ to the risk-aware objective $J_\lambda(\pi)$, yielding the Trust Region Volatility Optimization (TRVO) algorithm, which is similar to Trust Region Policy Optimization (Schulman et al., 2015).

Importantly, Bisi et al. (2019) show that $\mathbb{V}(G_0) \leq \frac{\mathbb{V}(R)}{(1-\gamma)^2}$, indicating that minimizing the variance of R implicitly minimizes the variance of G_0. We, therefore, can optimize $\mathbb{V}(R)$ as a proxy (upper bound) for optimizing $\mathbb{V}(G_0)$. In this paper, we argue that $\mathbb{V}(R)$ is easier to optimize than $\mathbb{V}(G_0)$. The methods of Tamar et al. (2012); Xie et al. (2018) optimizing $\mathbb{V}(G_0)$ involve terms like $(\mathbb{E}[G_0])^2$ and $\mathbb{E}[G_0^2]$, which lead to terms like $G_0^2 \sum_{t=0}^{T-1} \nabla_\theta \log \pi(A_t|S_t)$ in their update rules, yielding large variance. In particular, it is computationally prohibitive to further expand G_0^2 explicitly to apply variance reduction techniques like baselines (Williams, 1992). By contrast, we show in the next section that by considering $\mathbb{V}(R)$, MVPI involves only $r(s, a)^2$, which is easier to deal with than G_0^2.

3 Mean-Variance Policy Iteration

Although in many problems our goal is to maximize the expected total undiscounted reward, practitioners often find that optimizing the discounted objective ($\gamma < 1$) as a proxy for the undiscounted objective ($\gamma = 1$) is better than optimizing the undiscounted objective directly, especially when deep neural networks are used as function approximators (Mnih et al., 2015; Lillicrap et al., 2015; Espeholt et al., 2018; Xu et al., 2018; Van Seijen et al., 2019). We, therefore, focus on the discounted setting in the paper, which allows us to consider optimizing the variance of the per-step reward as a proxy (upper bound) for optimizing the variance of the total reward.

To address the policy-dependent reward issue, we use the Fenchel duality to rewrite $J_\lambda(\pi)$ as

$$J_\lambda(\pi) = \mathbb{E}[R] - \lambda\mathbb{E}[R^2] + \lambda(\mathbb{E}[R])^2 = \mathbb{E}[R] - \lambda\mathbb{E}[R^2] + \lambda \max_y \big(2\mathbb{E}[R]y - y^2\big), \tag{3}$$

yielding the following problem:

$$\max_{\pi,y} J_\lambda(\pi, y) \doteq \textstyle\sum_{s,a} d_\pi(s,a)\big(r(s,a) - \lambda r(s,a)^2 + 2\lambda r(s,a)y\big) - \lambda y^2. \tag{4}$$

We then propose a *block cyclic coordinate ascent* (BCCA, Luenberger and Ye 1984; Tseng 2001; Saha and Tewari 2010, 2013; Wright 2015) framework to solve (4), which updates y and π alternatively as shown in Algorithm 1. At the k-th iteration, we first fix π_k and update y_{k+1} (Step 1). As $J_\lambda(\pi_k, y)$ is

Algorithm 1: Mean-Variance Policy Iteration (MVPI)

for k = *1, …* **do**

  Step 1: $y_{k+1} \doteq (1-\gamma)J(\pi_k)$ `// The exact solution for` $\arg\max_y J_\lambda(\pi_k, y)$

  Step 2:

  $\pi_{k+1} \doteq \arg\max_\pi \Big(\sum_{s,a} d_\pi(s,a)\big(r(s,a) - \lambda r(s,a)^2 + 2\lambda r(s,a)y_{k+1}\big) - \lambda y_{k+1}^2 \Big)$

end

quadratic in y, y_{k+1} can be computed analytically as $y_{k+1} = \sum_{s,a} d_{\pi_k}(s,a) r(s,a) = (1-\gamma)J(\pi_k)$, i.e., all we need in this step is $J(\pi_k)$, which is exactly the performance metric of the policy π_k. We, therefore, refer to Step 1 as *policy evaluation*. We then fix y_{k+1} and update π_{k+1} (Step 2). Remarkably, Step 2 can be reduced to the following problem:

$$\pi_{k+1} = \arg\max_\pi \textstyle\sum_{s,a} d_\pi(s,a)\hat{r}(s,a;y_{k+1}),$$

where $\hat{r}(s,a;y) \doteq r(s,a) - \lambda r(s,a)^2 + 2\lambda r(s,a)y$. In other words, to compute π_{k+1}, we need to solve a new MDP, which is the same as the original MDP except that the reward function is $\hat{r}$ instead of r. This new reward function $\hat{r}$ does not depend on the policy π, avoiding the policy-dependent-reward issue of Bisi et al. (2019). In this step, a new policy π_{k+1} is computed. An intuitive conjecture is that this step is a *policy improvement* step, and we confirm this with the following proposition:

Proposition 1. *(Monotonic Policy Improvement)* $\forall k, J_\lambda(\pi_{k+1}) \geq J_\lambda(\pi_k)$.

Though the monotonic improvement w.r.t. the objective $J_\lambda(\pi, y)$ in Eq (4) follows directly from standard BCCA theories, Theorem 1 provides the monotonic improvement w.r.t. the objective $J_\lambda(\pi)$ in Eq (3). The proof is provided in the appendix. Given Theorem 1, we can now consider the whole BCCA framework in Algorithm 1 as a policy iteration framework, which we call *mean-variance policy iteration* (MVPI). Let $\{\pi_\theta : \theta \in \Theta\}$ be the function class for policy optimization, we have

Assumption 1. $\{\theta \in \Theta, y \in \mathbb{R} \mid J_\lambda(\theta, y) \geq J_\lambda(\theta_0)\}$ *is compact, where* θ_0 *is the initial parameters.*

Assumption 2. $\sup_{\theta\in\Theta} \max\{||\frac{\partial \log \pi_\theta(a|s)}{\partial\theta_i \partial\theta_j}||, ||\nabla_\theta \log \pi_\theta(a|s)||\} < \infty$, Θ *is open and bounded.*

Proposition 2. *(Convergence of MVPI with function approximation) Under Assumptions 1 & 2, let*

$$y_{k+1} \doteq \arg\max_y J_\lambda(\theta_k, y), \quad \theta_{k+1} \doteq \arg\max_{\theta\in\Theta} J_\lambda(\theta, y_{k+1}), \quad k = 0, 1, \dots$$

then $J_\lambda(\theta_{k+1}) \geq J_\lambda(\theta_k)$, $\{J_\lambda(\theta_k)\}_{k=1,\dots}$ *converges, and* $\liminf_k ||\nabla_\theta J_\lambda(\theta_k)|| = 0$.

Remark 1. *Assumption 1 is standard in BCCA literature (e.g., Theorem 4.1 in Tseng (2001)). Assumption 2 is standard in policy optimization literature (e.g., Assumption 4.1 in Papini et al. (2018)). Convergence in the form of* $\liminf$ *also appears in other literature (e.g., Luenberger and Ye (1984); Tseng (2001); Konda (2002); Zhang et al. (2020c)).*

The proof is provided in the appendix. MVPI enjoys great flexibility in that any policy evaluation method and risk-neutral control method can be dropped in off the shelf, which makes it possible to leverage all the advances in risk-neutral RL. MVPI differs from the standard policy iteration (PI, e.g., see Bertsekas and Tsitsiklis (1996); Puterman (2014); Sutton and Barto (2018)) in two key ways: **(1)** policy evaluation in MVPI requires only a scalar performance metric, while standard policy evaluation involves computing the value of all states. **(2)** policy improvement in MVPI considers an augmented reward $\hat{r}$, which is different at each iteration, while standard policy improvement always considers the original reward. Standard PI can be used to solve the policy improvement step in MVPI.

Average Reward Setting: So far we have considered the total reward as the primary performance metric for mean-variance RL. We now show that MVPI can also be used when we consider the average reward as the primary performance metric. Assuming the chain induced by π is ergodic and letting $\bar{d}_\pi(s)$ be its stationary distribution, Filar et al. (1989); Prashanth and Ghavamzadeh (2013) consider the *long-run variance* risk measure $\Lambda(\pi) \doteq \sum_{s,a} \bar{d}_\pi(s,a)\big(r(s,a) - \bar{J}(\pi)\big)^2$ for the average reward setting, where $\bar{d}_\pi(s,a) \doteq \bar{d}_\pi(s)\pi(a|s)$ and $\bar{J}(\pi) = \sum_{s,a} \bar{d}_\pi(s,a) r(s,a)$ is the average reward. We now define a risk-aware objective

$$\bar{J}_\lambda(\pi) \doteq \bar{J}(\pi) - \lambda\Lambda(\pi) = \max_y \sum_{s,a} \bar{d}_\pi(s,a)\hat{r}(s,a;y) - \lambda y^2, \tag{5}$$

where we have used the Fenchel duality and BCCA can take over to derive MVPI for the average reward setting as Algorithm 1. It is not a coincidence that the only difference between (4) and (5) is the difference between d_π and $\bar{d}_\pi$. The root cause is that the total discounted reward of an MDP is always equivalent to the average reward of an artificial MDP (up to a constant multiplier), whose transition kernel is $\tilde{p}(s'|s,a) = \gamma p(s'|s,a) + (1-\gamma)\mu_0(s')$ (e.g., see Section 2.4 in Konda (2002) for details).

Off-Policy Learning: Off-policy learning has played a key role in improving data efficiency (Lin, 1992; Mnih et al., 2015) and exploration (Osband et al., 2016, 2018) in risk-neutral control algorithms. Previous mean-variance RL methods, however, consider only the on-policy setting and cannot be easily made off-policy. For example, it is not clear whether perturbation methods for estimating gradients (Prashanth and Ghavamzadeh, 2013) can be used off-policy. To reweight terms like $G_0^2 \sum_{t=0}^{T-1} \nabla_\theta \log \pi(A_t|S_t)$ from Tamar et al. (2012); Xie et al. (2018) in the off-policy setting, we would need to compute the product of importance sampling ratios $\Pi_{i=0}^{T-1} \frac{\pi(a_i|s_i)}{\mu(a_i|s_i)}$, where μ is the behavior policy. This product usually suffers from high variance (Precup et al., 2001; Liu et al., 2018) and requires knowing the behavior policy μ, both of which are practical obstacles in real applications. By contrast, as MVPI works on an augmented MDP directly, any risk-neutral off-policy learning technique can be used for risk-averse off-policy control directly. In this paper, we consider MVPI in both on-line and off-line off-policy settings.

On-line setting. In the on-line off-policy setting, an agent interacts with the environment following a behavior policy μ to collect transitions, which are stored into a replay buffer (Lin, 1992) for future reuse. Mujoco robot simulation tasks (Brockman et al., 2016) are common benchmarks for this paradigm (Lillicrap et al., 2015; Haarnoja et al., 2018), and TD3 is a leading algorithm in Mujoco tasks (Achiam, 2018). TD3 is a risk-neutral control algorithm, reducing the over-estimation bias (Hasselt, 2010) of DDPG (Lillicrap et al., 2015), which is a neural network implementation of the deterministic policy gradient theorem (Silver et al., 2014). Given the empirical success of TD3, we propose MVPI-TD3 for risk-averse control in this setting. In the policy evaluation step of MVPI-TD3, we set y_{k+1} to the average of the recent K rewards, where K is a hyperparameter to be tuned and we have assumed the policy changes slowly. Theoretically, we should use a weighted average as $d_\pi(s,a)$ is a discounted distribution. Though implementing this weighted average is straightforward, practitioners usually ignore discounting for state visitation in policy gradient methods to improve sample efficiency (Mnih et al., 2016; Schulman et al., 2015, 2017; Bacon et al., 2017). Hence, we do not use the weighted average in MVPI-TD3. In the policy improvement step of MVPI-TD3, we sample a mini-batch of transitions from the replay buffer and perform one TD3 gradient update. The pseudocode of MVPI-TD3 is provided in the appendix.

Off-line setting. In the off-line off-policy setting, we are presented with a batch of transitions $\{s_i, a_i, r_i, s_i'\}_{i=1,\dots,K}$ and want to learn a good target policy π for control solely from this batch of transitions. Sometimes those transitions are generated by following a known behavior policy μ. But more commonly, those transitions are generated from multiple unknown behavior policies, which we refer to as the behavior-agnostic off-policy setting (Nachum et al., 2019a). Namely, the state-action pairs (s_i, a_i) are distributed according to some unknown distribution d, which may result from multiple unknown behavior policies. The successor state s_i' is distributed according to $p(\cdot|s_i, a_i)$

	$\Delta_{\text{J}}^{\text{TRVO}}$	$\Delta_{\text{mean}}^{\text{TRVO}}$	$\Delta_{\text{variance}}^{\text{TRVO}}$	$\Delta_{\text{SR}}^{\text{TRVO}}$	$\Delta_{\text{J}}^{\text{MVPI}}$	$\Delta_{\text{mean}}^{\text{MVPI}}$	$\Delta_{\text{variance}}^{\text{MVPI}}$	$\Delta_{\text{SR}}^{\text{MVPI}}$
InvertedP.	-1107%	-3%	NaN [2]	NaN	0%	0%	NaN	NaN
InvertedD.P.	-1915%	-27%	1867%	-84%	82%	-40%	-81%	38%
HalfCheetah	82%	-84%	-83%	-63%	86%	-53%	-85%	20%
Walker2d	36%	-61%	-36%	-51%	97%	-47%	-97%	193%
Swimmer	-5%	0%	188%	-41%	-5%	0%	151%	-37%
Hopper	26%	-31%	-26%	-19%	84%	-6%	-84%	133%
Reacher	-42%	-7%	86%	22%	2%	5%	2%	6%
Ant	98%	-84%	-98%	0%	98%	-59%	-98%	173%

Table 1: Normalized statistics of TRVO and MVPI-TD3. MVPI is shorthand for MVPI-TD3 in this table. For algo $\in \{\text{MVPI-TD3}, \text{TRVO}, \text{TD3}\}$, we compute the risk-aware performance metric as $\text{J}_{\text{algo}} \doteq \text{mean}_{\text{algo}} - \lambda \text{variance}_{\text{algo}}$ with $\lambda = 1$, where $\text{mean}_{\text{algo}}$ and $\text{variance}_{\text{algo}}$ are mean and variance of the 100 evaluation episodic returns. Then we compute the normalized statistics as $\Delta_{\text{J}}^{\text{algo}} \doteq \frac{\text{J}_{\text{algo}} - \text{J}_{\text{TD3}}}{|\text{J}_{\text{TD3}}|}$, $\Delta_{\text{mean}}^{\text{algo}} \doteq \frac{\text{mean}_{\text{algo}} - \text{mean}_{\text{TD3}}}{|\text{mean}_{\text{TD3}}|}$, $\Delta_{\text{variance}}^{\text{algo}} \doteq \frac{\text{variance}_{\text{algo}} - \text{variance}_{\text{TD3}}}{|\text{variance}_{\text{TD3}}|}$, $\text{SR}_{\text{algo}} \doteq \frac{\text{mean}_{\text{algo}}}{\sqrt{\text{variance}_{\text{algo}}}}$, $\Delta_{\text{SR}}^{\text{algo}} \doteq \frac{\text{SR}_{\text{algo}} - \text{SR}_{\text{TD3}}}{|\text{SR}_{\text{TD3}}|}$ Both MVPI-TD3 and TRVO are trained with $\lambda = 1$. All J_{algo} are averaged over 10 independent runs.

and $r_i = r(s, a)$. The degree of off-policyness in this setting is usually larger than the on-line off-policy setting.

In the off-line off-policy setting, the policy evaluation step in MVPI becomes the standard *off-policy evaluation* problem (OPE, Thomas et al. (2015); Thomas and Brunskill (2016); Jiang and Li (2016); Liu et al. (2018)), where we want to estimate a scalar performance metric of a policy with off-line samples. One promising approach to OPE is *density ratio learning*, where we use function approximation to learn the density ratio $\frac{d_\pi(s,a)}{d(s,a)}$ directly, which we then use to reweight $r(s, a)$. All off-policy evaluation algorithms can be integrated into MVPI in a plug-and-play manner. In the off-line off-policy setting, the policy improvement step in MVPI becomes the standard *off-policy policy optimization* problem, where we can reweight the canonical on-policy actor-critic (Sutton et al., 2000; Konda, 2002) with the density ratio as in Liu et al. (2019) to achieve off-policy policy optimization. Algorithm 2 in the appendix provides an example of Off-line MVPI.

In the on-line off-policy learning setting, the behavior policy and the target policy are usually closely correlated (e.g., in MVPI-TD3), we, therefore, do not need to learn the density ratio. In the off-line off-policy learning setting, the dataset may come from behavior policies that are arbitrarily different from the target policy. We, therefore, resort to density ratios to account for this discrepancy. Density ratio learning itself is an active research area and is out of scope of this paper. See Hallak and Mannor (2017); Liu et al. (2018); Gelada and Bellemare (2019); Nachum et al. (2019a); Zhang et al. (2020a,b); Mousavi et al. (2020) for more details about density ratio learning.

4 Experiments

On-line learning setting. In many real-world robot applications, e.g., in a warehouse, it is crucial that the robots' performance be consistent. In such cases, risk-averse RL is an appealing option to train robots. Motivated by this, we benchmark MVPI-TD3 on eight Mujoco robot manipulation tasks from OpenAI gym. As we are not aware of any other off-policy mean-variance RL method, we use several recent on-policy mean-variance RL method as baselines, namely, the methods of Tamar et al. (2012); Prashanth and Ghavamzadeh (2013), MVP (Xie et al., 2018), and TRVO (Bisi et al., 2019). The methods of Tamar et al. (2012); Prashanth and Ghavamzadeh (2013) and MVP are not designed for deep RL settings. To make the comparison fair, we improve those baselines with parallelized actors to stabilize the training of neural networks as in Mnih et al. (2016).[3] TRVO is essentially MVPI with TRPO for the policy improvement. We, therefore, implement TRVO as MVPI with Proximal Policy Optimization (PPO, Schulman et al. 2017) to improve its performance. We also use the vanilla risk-neutral TD3 as a baseline. We use two-hidden-layer neural networks for function approximation.

[2]This is due to 0 in denominator. Both the policy from TD3 and the environment are deterministic. So the variance of the TD3 evaluation episodic returns is 0.

[3]They are on-policy algorithms so we cannot use experience replay.

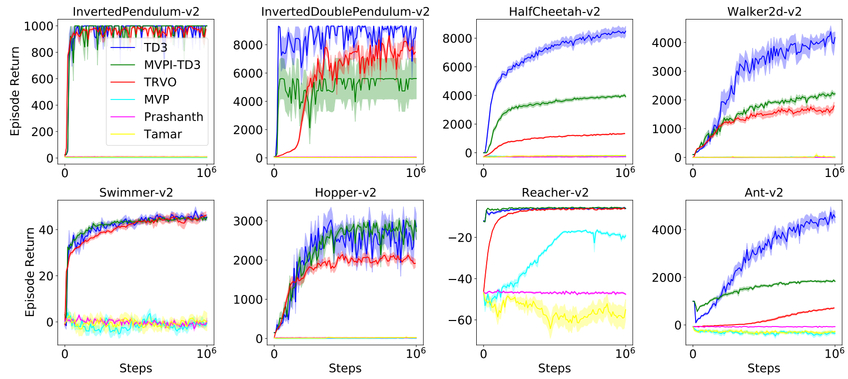
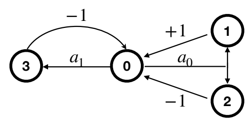


Figure 1: Training progress of MVPI-TD3 and baseline algorithms. Curves are averaged over 10 independent runs with shaded regions indicating standard errors.

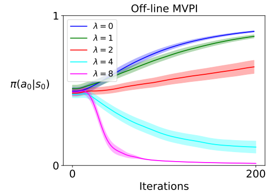
We run each algorithm for 10^6 steps and evaluate the algorithm every 10^4 steps for 20 episodes. We report the mean of those 20 episodic returns against the training steps in Figure 1. The curves are generated by setting $\lambda = 1$. More details are provided in the appendix. The results show that MVPI-TD3 outperforms all risk-averse baselines in all tested domains (in terms of both final episodic return and learning speed), with only one exception, `InvertedDoublePendulum`, where TRVO outperforms MVPI-TD3. Moreover, the curves of the methods from the total reward perspective are always flat in all domains with only one exception that MVP achieves a reasonable performance in `Reacher`, though exhaustive hyperparameter tuning is conducted, including λ and ξ. Those flat curves suggest that perturbation-based gradient estimation in Prashanth and Ghavamzadeh (2013) may not work well with neural networks, and the $G_0^2 \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t)$ term in Tamar et al. (2012) and MVP may suffer from high variance, yielding instability. By contrast, the two algorithms from the per-step reward perspective (MVPI-TD3 and TRVO) do learn a reasonable policy, which experimentally supports our argument that the variance of the per-step reward is easier to optimize than the variance of the total reward.

As shown in Figure 1, the vanilla risk-neutral TD3 outperforms all risk-averse algorithms (in terms of episodic return). This is expected as it is in general hard for a risk-averse algorithm to outperform its risk-neutral counterpart in terms of a risk-neutral performance metric. We now compare TD3, MVPI-TD3 and TRVO in terms of a risk-aware performance metric. To this end, we test the agent at the end of training for an extra 100 episodes to compute a risk-aware performance metric. We report the normalized statistics in Table 1. The results show that MVPI-TD3 outperforms TD3 in 6 out of 8 tasks in terms of the risk-aware performance metric. Moreover, MVPI-TD3 outperforms TRVO in 6 out of 8 tasks. We also compare the algorithms in terms of the sharp ratio (SR, Sharpe 1966). Although none of the algorithms optimizes SR directly, MVPI-TD3 outperforms both TD3 and TRVO in 6 out 8 tasks in terms of SR. This performance boost of MVPI-TD3 over TRVO indeed results from the performance boost of TD3 over PPO, and it is the flexibility of MVPI that makes this off-the-shelf application TD3 in risk-averse RL possible. We also provide versions of Figure 1 and Table 1 with $\lambda = 0.5$ and $\lambda = 2$ in the appendix. The relative performance is the same as $\lambda = 1$.

Off-line learning setting. We consider an infinite horizon MDP (Figure 2). Two actions a_0 and a_1 are available at s_0, and we have $p(s_3|s_0, a_1) = 1, p(s_1|s_0, a_0) = p(s_2|s_0, a_0) = 0.5$. The discount factor is $\gamma = 0.7$ and the agent is initialized at s_0. We consider the objective $J_\lambda(\pi)$ in Eq (3). If $\lambda = 0$, the optimal policy is to choose a_0. If λ is large enough, the optimal policy is to choose a_1. We consider the behavior-agnostic off-policy setting, where the sampling distribution d satisfies $d(s_0, a_0) = d(s_0, a_1) = d(s_1) = d(s_2) = d(s_3) = 0.2$. This sampling distribution may result from multiple unknown behavior policies. Although the representation is tabular, we use a softmax policy. So the problem we consider is *nonlinear* and *nonconvex*. As we are not aware of any other behavior-agnostic off-policy risk-averse RL method, we benchmark only Off-line MVPI (Algorithm 2). Details are provided in the appendix. We report the probability of selecting a_0 against training iterations. As shown in Figure 2, $\pi(a_0|s_0)$ decreases as λ increases, indicating Off-line MVPI copes well with different risk levels. The main challenge in Off-line MVPI rests on learning the density ratio. Scaling



(a)



(b)

Figure 2: (a) A tabular MDP. (b) The training progress of Off-line MVPI. Curves are averaged over 30 independent runs with shaded regions indicating standard errors.

up density ratio learning algorithms reliably to more challenging domains like Mujoco is out of the scope of this paper.

5 Related Work

Both MVPI and Bisi et al. (2019) consider the per-step reward perspective for mean-variance RL. In this work, we mainly use the variance of the per-step reward as a proxy (upper bound) for optimizing the variance of the total reward. Though TRVO in Bisi et al. (2019) is the same as instantiating MVPI with TRPO, the derivation is dramatically different. In particular, it is not clear whether the performance-lower-bound-based derivation for TRVO can be adopted to deterministic policies, off-policy learning, or other policy optimization paradigms, and this is not explored in Bisi et al. (2019). By contrast, MVPI is compatible with *any* existing risk-neural policy optimization technique. Furthermore, MVPI works for both the total discounted reward setting and the average reward setting. It is not clear how the performance lower bound in Bisi et al. (2019), which plays a central role in TRVO, can be adapted to the average reward setting. All the advantages of MVPI over TRVO result from addressing the policy-dependent-reward issue in Bisi et al. (2019). While the application of Fenchel duality in RL is not new, previously it has been used only to address double sampling issues (e.g., Liu et al. (2015); Dai et al. (2017); Xie et al. (2018); Nachum et al. (2019a)). By contrast, we use Fenchel duality together with BCAA to address the policy-dependent-reward issue in Bisi et al. (2019) and derive a policy iteration framework that appears to be novel to the RL community. Besides variance, value at risk (VaR, Chow et al. 2018), conditional value at risk (CVaR, Chow and Ghavamzadeh 2014; Tamar et al. 2015; Chow et al. 2018), sharp ratio (Tamar et al., 2012), and exponential utility (Howard and Matheson, 1972; Borkar, 2002) are also used for risk-averse RL. In particular, it is straightforward to consider exponential utility for the per-step reward, which, however, suffers from the same problems as the exponential utility for the total reward, e.g., it overflows easily (Gosavi et al., 2014).

6 Conclusion

In this paper, we propose MVPI for risk-averse RL. MVPI enjoys great flexibility such that any policy evaluation method and risk-neutral control method can be dropped in for risk-averse control off the shelf, in both on- and off-policy settings. This flexibility dramatically reduces the gap between risk-neutral control and risk-averse control. To the best of our knowledge, MVPI is the first empirical success of risk-averse RL in Mujoco robot simulation domains, and is also the first success of off-policy risk-averse RL and risk-averse RL with deterministic polices. Deterministic policies play an important role in reducing the variance of a policy (Silver et al., 2014). Off-policy learning is important for improving data efficiency (Mnih et al., 2015) and exploration (Osband et al., 2018). Incorporating those two elements in risk-averse RL appears novel and is key to the observed performance improvement.

Possibilities for future work include considering other risk measures (e.g., VaR and CVaR) of the per-step reward random variable, integrating more advanced off-policy policy optimization techniques (e.g., Nachum et al. 2019b) in off-policy MVPI, optimizing λ with meta-gradients (Xu et al., 2018), analyzing the sample complexity of MVPI, and developing theory for approximate MVPI.

Acknowledgments and Disclosure of Funding

SZ is generously funded by the Engineering and Physical Sciences Research Council (EPSRC). This project has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713). The experiments were made possible by a generous equipment grant from NVIDIA. BL's research is funded by the National Science Foundation (NSF) under grant NSF IIS1910794 and an Amazon Research Award.

References

Achiam, J. (2018). Spinning up in deep reinforcement learning.

Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Bertsekas, D. (1995). *Nonlinear Programming*. Athena Scientific.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific Belmont, MA.

Bhatnagar, S., Prasad, H., and Prashanth, L. (2013). *Stochastic Recursive Algorithms for Optimization*. Springer London.

Bisi, L., Sabbioni, L., Vittori, E., Papini, M., and Restelli, M. (2019). Risk-averse trust region optimization for reward-volatility reduction. *arXiv preprint arXiv:1912.03193*.

Borkar, V. S. (2002). Q-learning for risk-sensitive control. *Mathematics of operations research*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Chow, Y. (2017). *Risk-Sensitive and Data-Driven Sequential Decision Making*. PhD thesis, Stanford University.

Chow, Y. and Ghavamzadeh, M. (2014). Algorithms for cvar optimization in mdps. In *Advances in Neural Information Processing Systems*.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. (2018). Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*.

Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., and Song, L. (2017). Sbeed: Convergent reinforcement learning with nonlinear function approximation. *arXiv preprint arXiv:1712.10285*.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. (2017). Openai baselines. `https://github.com/openai/baselines`.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. (2018). Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*.

Filar, J. A., Kallenberg, L. C., and Lee, H.-M. (1989). Variance-penalized markov decision processes. *Mathematics of Operations Research*.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*.

Gelada, C. and Bellemare, M. G. (2019). Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

Gosavi, A. A., Das, S. K., and Murray, S. L. (2014). Beyond exponential utility functions: A variance-adjusted approach for risk-averse reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

Hallak, A. and Mannor, S. (2017). Consistent on-line off-policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning*.

Hasselt, H. V. (2010). Double q-learning. In *Advances in neural information processing systems*.

Howard, R. A. and Matheson, J. E. (1972). Risk-sensitive markov decision processes. *Management Science*.

Jiang, N. and Li, L. (2016). Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*.

Konda, V. R. (2002). *Actor-critic algorithms*. PhD thesis, Massachusetts Institute of Technology.

Lai, T. L., Xing, H., Chen, Z., et al. (2011). Mean-variance portfolio optimization when means and covariances are unknown. *The Annals of Applied Statistics*.

Li, D. and Ng, W.-L. (2000). Optimal dynamic portfolio selection: Multiperiod mean-variance formulation. *Mathematical finance*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*.

Liu, B., Liu, J., Ghavamzadeh, M., Mahadevan, S., and Petrik, M. (2015). Finite-sample analysis of proximal gradient td algorithms. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*.

Liu, Q., Li, L., Tang, Z., and Zhou, D. (2018). Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. (2019). Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*.

Luenberger, D. G. and Ye, Y. (1984). *Linear and nonlinear programming (3rd Edition)*. Springer.

Maei, H. R. (2011). *Gradient temporal-difference learning algorithms*. PhD thesis, University of Alberta.

Majumdar, A. and Pavone, M. (2020). How should a robot assess risk? towards an axiomatic theory of risk in robotics. In *Robotics Research*. Springer.

Mannor, S. and Tsitsiklis, J. (2011). Mean-variance optimization in markov decision processes. *arXiv preprint arXiv:1104.5601*.

Markowitz, H. M. and Todd, G. P. (2000). *Mean-variance analysis in portfolio choice and capital markets*. John Wiley & Sons.

Matthaeia, R., Reschkaa, A., Riekena, J., Dierkesa, F., Ulbricha, S., Winkleb, T., and Maurera, M. (2015). Autonomous driving: Technical, legal and social aspects.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*.

Mousavi, A., Li, L., Liu, Q., and Zhou, D. (2020). Black-box off-policy estimation for infinite-horizon reinforcement learning. In *International Conference on Learning Representations*.

Nachum, O., Chow, Y., Dai, B., and Li, L. (2019a). Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *arXiv preprint arXiv:1906.04733*.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. (2019b). Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*.

OpenAI (2018). Openai five. `https://openai.com/five/`.

Osband, I., Aslanides, J., and Cassirer, A. (2018). Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*.

Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., and Restelli, M. (2018). Stochastic variance-reduced policy gradient. *arXiv preprint arXiv:1806.05618*.

Parker, D. (2009). Managing risk in healthcare: understanding your safety culture using the manchester patient safety framework (mapsaf). *Journal of nursing management*.

Prashanth, L. and Ghavamzadeh, M. (2013). Actor-critic algorithms for risk-sensitive mdps. In *Advances in neural information processing systems*.

Precup, D., Sutton, R. S., and Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. In *Proceedings of the 18th International Conference on Machine Learning*.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Saha, A. and Tewari, A. (2010). On the finite time convergence of cyclic coordinate descent methods. *arXiv preprint arXiv:1005.2146*.

Saha, A. and Tewari, A. (2013). On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*.

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

Senior, A., Jumper, J., and Hassabis, D. (2018). Alphafold: Using ai for scientific discovery. *DeepMind. Recuperado de: https://deepmind. com/blog/alphafold*.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of business*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*.

Sobel, M. J. (1982). The variance of discounted markov decision processes. *Journal of Applied Probability*.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction (2nd Edition)*. MIT press.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*.

Tamar, A., Di Castro, D., and Mannor, S. (2012). Policy gradients with variance related risk criteria. *arXiv preprint arXiv:1206.6404*.

Tamar, A., Glassner, Y., and Mannor, S. (2015). Optimizing the cvar via sampling. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

Thomas, P. and Brunskill, E. (2016). Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148.

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. (2015). High-confidence off-policy evaluation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494.

Van Seijen, H., Fatemi, M., and Tavakoli, A. (2019). Using a logarithmic mapping to enable lower discount factors in reinforcement learning. In *Advances in Neural Information Processing Systems*.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*.

Wang, S. S. (2000). A class of distortion operators for pricing financial and insurance risks. *Journal of risk and insurance*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine Learning*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34.

Xie, T., Liu, B., Xu, Y., Ghavamzadeh, M., Chow, Y., Lyu, D., and Yoon, D. (2018). A block coordinate ascent algorithm for mean-variance optimization. In *Advances in Neural Information Processing Systems*.

Xu, Z., van Hasselt, H. P., and Silver, D. (2018). Meta-gradient reinforcement learning. In *Advances in neural information processing systems*.

Zhang, R., Dai, B., Li, L., and Schuurmans, D. (2020a). Gendice: Generalized offline estimation of stationary values. In *International Conference on Learning Representations*.

Zhang, S., Liu, B., and Whiteson, S. (2020b). Gradientdice: Rethinking generalized offline estimation of stationary values. In *Proceedings of the 37th International Conference on Machine Learning*.

Zhang, S., Liu, B., Yao, H., and Whiteson, S. (2020c). Provably convergent two-timescale off-policy actor-critic with function approximation. In *Proceedings of the 37th International Conference on Machine Learning*.

A Proofs

A.1 Proof of Proposition 1

Proof.

$$
\begin{aligned}
&J_\lambda(\pi_{k+1}) \\
=&\sum_{s,a} d_{\pi_{k+1}}(s,a)(r(s,a) - \lambda r(s,a)^2) + \lambda \max_y (2\sum_{s,a} d_{\pi_{k+1}}(s,a) r(s,a) y - y^2) \\
\geq &\sum_{s,a} d_{\pi_{k+1}}(s,a)(r(s,a) - \lambda r(s,a)^2) + \lambda (2\sum_{s,a} d_{\pi_{k+1}}(s,a) r(s,a) y_{k+1} - y_{k+1}^2) \\
=&\sum_{s,a} d_{\pi_{k+1}}(s,a)\big(r(s,a) - \lambda r(s,a)^2 + 2\lambda r(s,a) y_{k+1}\big) - \lambda y_{k+1}^2 \\
\geq &\sum_{s,a} d_{\pi_{k}}(s,a)\big(r(s,a) - \lambda r(s,a)^2 + 2\lambda r(s,a) y_{k+1}\big) - \lambda y_{k+1}^2 \\
& \qquad\qquad\qquad\qquad\qquad\qquad \text{(By definition, } \pi_{k+1} \text{ is the maximizer.)} \\
=&\sum_{s,a} d_{\pi_{k}}(s,a)(r(s,a) - \lambda r(s,a)^2) + \lambda (2\sum_{s,a} d_{\pi_{k}}(s,a) r(s,a) y_{k+1} - y_{k+1}^2) \\
=&\sum_{s,a} d_{\pi_{k}}(s,a)(r(s,a) - \lambda r(s,a)^2) + \lambda \max_y (2\sum_{s,a} d_{\pi_{k}}(s,a) r(s,a) y - y^2) \\
& \qquad\qquad\qquad\qquad \text{(By definition, } y_{k+1} \text{ is the maximizer of the quadratic.)} \\
=&J_\lambda(\pi_k)
\end{aligned}
$$

□

A.2 Proof of Proposition 2

Lemma 1. *Under Assumption 2, $\nabla_\theta J_\lambda(\theta)$ is Lipschitz continuous in θ.*

Proof. By definition,

$$\nabla J_\lambda(\theta) = \nabla \mathbb{E}[R] - \nabla \lambda \mathbb{E}[R^2] + 2\lambda \mathbb{E}[R] \nabla \mathbb{E}[R].$$

The policy gradient theorem (Sutton et al., 2000) and the boundedness of $\nabla \log \pi_\theta(a|s)$ imply that $\nabla \mathbb{E}[R]$ is bounded. So $\mathbb{E}[R]$ is Lipschitz continuous. Lemma B.2 in Papini et al. (2018) shows that the Hessian of $\mathbb{E}[R]$ is bounded. So $\nabla \mathbb{E}[R]$ is Lipschitz continuous. So does $\nabla \mathbb{E}[R^2]$. Together with the boundedness of $\mathbb{E}[R]$, it is easy to see $\nabla J_\lambda(\theta)$ is Lipschitz continuous. □

We now prove Theorem 2.

Proof. Under Assumption 1, Theorem 4.1(c) in Tseng (2001) shows that the limit of any convergent subsequence $\{(\theta_k, y_k)\}_{k \in \mathcal{K}}$, referred to as $(\theta_\mathcal{K}, y_\mathcal{K})$, satisfies $\nabla_\theta J_\lambda(\theta_\mathcal{K}, y_\mathcal{K}) = 0$ and $\nabla_y J_\lambda(\theta_\mathcal{K}, y_\mathcal{K}) = 0$. In particular, that Theorem 4.1(c) is developed for general block coordinate ascent algorithms with M blocks. Our MVPI is a special case with two blocks (i.e., θ and y). With only two blocks, the conclusion of Theorem 4.1(c) follows immediately from Eq (7) and Eq (8) in Tseng (2001), without involving the assumption that the maximizers of the $M-2$ blocks are unique.

As $J_\lambda(\theta, y)$ is quadratic in y, $\nabla_y J_\lambda(\theta_\mathcal{K}, y_\mathcal{K}) = 0$ implies $y_\mathcal{K} = \arg\max_y J_\lambda(\theta_\mathcal{K}, y) = (1-\gamma)J(\theta_\mathcal{K})$. Recall the Fenchel duality $x^2 = \max_z f(x,z)$, where $f(x,z) \doteq 2xz - z^2$. Applying Danskin's theorem (Proposition B.25 in Bertsekas (1995)) to Fenchel duality yields

$$\frac{\partial x^2}{\partial x} = \frac{\partial f(x, \arg\max_z f(x,z))}{\partial x}. \tag{6}$$

Note Danskin's theorem shows that we can treat $\arg\max_z f(x, z)$ as a constant independent of x when computing the gradients in the RHS of Eq (6). Applying Danskin's theorem in the Fenchel duality used in Eq (3) yields

$$\nabla_\theta J_\lambda(\theta_\mathcal{K}) = \nabla_\theta J_\lambda(\theta_\mathcal{K}, y_\mathcal{K}) = 0. \tag{7}$$

Eq (7) can also be easily verified without invoking Danskin's theorem by expanding the gradients explicitly. Eq (7) indicates that the subsequence $\{\theta_k\}_{k\in\mathcal{K}}$ converges to a stationary point of $J_\lambda(\theta)$.

Theorem 1 establishes the monotonic policy improvement when we search over all possible policies (The $\arg\max$ of Step 2 in Algorithm 1 is taken over all possible policies). Fortunately, the proof of Theorem 1 can also be used (up to a change of notation) to establish that

$$J_\lambda(\theta_{k+1}) \geq J_\lambda(\theta_k). \tag{8}$$

In other words, the monotonic policy improvement also holds when we search over Θ. Eq (8) and the fact that $J_\lambda(\theta)$ is bounded from above imply that $\{J_\lambda(\theta_k)\}_{k=1,\dots}$ converges to some J_*.

Let $\Theta_0 \doteq \{\theta \in \Theta \mid J_\lambda(\theta) \geq J_\lambda(\theta_0)\}$. We first show Θ_0 is compact. Let $\{\theta^i\}_{i=1,\dots}$ be any convergent sequence in Θ_0 and θ^∞ be its limit. We define $y^i \doteq \arg\max_y J_\lambda(\theta^i, y) = (1-\gamma)J(\theta^i)$ for $i = 1, \dots, \infty$. The proof of Lemma 1 shows $J(\theta)$ is Lipschitz continuous in θ, indicating $\{\theta^i, y^i\}$ converges to $\{\theta^\infty, y^\infty\}$. As $J_\lambda(\theta^i, y^i) = J_\lambda(\theta^i) \geq J_\lambda(\theta_0)$, Assumption 1 implies $J_\lambda(\theta^\infty, y^\infty) \geq J_\lambda(\theta_0)$, i.e., $J_\lambda(\theta^\infty) \geq J_\lambda(\theta_0)$, $\theta^\infty \in \Theta_0$. So Θ_0 is compact. As $\{\theta_k\}$ is contained in Θ_0, there must exist a convergent subsequence, indicating

$$\liminf_k ||\nabla_\theta J_\lambda(\theta_k)|| = 0.$$

□

B Experiment Details

The pseudocode of MVPI-TD3 and our TRVO (MVPI-PPO) are provide in Algorithms 3 and 4 respectively.

Algorithm 2: Off-line MVPI

Input: A batch of transitions $\{s_i, a_i, r_i, s'_i\}_{i=1,\dots,K}$ and a learning rate α
while *True* **do**
  Learn the density ratio $\rho_\pi(s, a) \doteq \frac{d_\pi(s,a)}{d(s,a)}$ with $\{s_i, a_i, r_i, s'_i\}_{i=1,\dots,K}$
  `// For example, use GradientDICE (Zhang et al., 2020b)`
  $y \leftarrow \frac{1}{K}\sum_{i=1}^K \rho_\pi(s_i, a_i) r_i$
  for $i = 1, \dots, K$ **do**
    $\hat{r}_i \leftarrow r_i - \lambda r_i^2 + 2\lambda r_i y$
    $a'_i \sim \pi(\cdot|s'_i)$
  end
  Learn $q_\pi(s, a)$ with $\{s_i, a_i, \hat{r}_i, s'_i, a'_i\}_{i=1,\dots,K}$
  `// For example, use TD(0) (Sutton, 1988) in` $\mathcal{S}\times\mathcal{A}$
  $\theta \leftarrow \theta + \alpha\rho_\pi(s_i, a_i)\nabla_\theta \log\pi(a_i|s_i) q_\pi(s_i, a_i)$, where i is randomly selected
end

Task Selection: We use eight Mujoco tasks from Open AI gym [4](Brockman et al., 2016) and implement the tabular MDP in Figure 2a by ourselves.

Function Parameterization: For MVPI-TD3 and TD3, we use the same network architecture as Fujimoto et al. (2018). For TRVO (MVPI-PPO), the methods of Tamar et al. (2012); Prashanth and Ghavamzadeh (2013), and MVP, we use the same network architecture as Schulman et al. (2017).

Hyperparameter Tuning: For MVPI-TD3 and TD3, we use the same hyperparameters as Fujimoto et al. (2018). In particular, for MVPI-TD3, we set $K = 10^4$. For TRVO (MVPI-PPO), we use

[4] `https://gym.openai.com/`

Algorithm 3: MVPI-TD3

Input:
θ, ψ: parameters for the deterministic policy π and the value function q_π
K: number of recent rewards for estimating the policy performance
λ: weight of the variance penalty

Initialize the replay buffer $\mathcal{M}$
Initialize S_0
for $t = 0, \dots,$ **do**
    $A_t \leftarrow \pi(S_t) + \mathcal{N}(0, \sigma^2)$
    Execute A_t, get R_{t+1}, S_{t+1}
    Store $(S_t, A_t, R_{t+1}, S_{t+1})$ into $\mathcal{M}$
    $y \leftarrow \frac{1}{K} \sum_{i=t-K+2}^{t+1} R_t$
    Sample a mini-batch $\{s_i, a_i, r_i, s'_i\}_{i=1,\dots,N}$ from $\mathcal{M}$
    for $i = 1, \dots, N$ **do**
        $\hat{r}_i \leftarrow r_i - \lambda r_i^2 + 2\lambda r_i y$
    end
    Use TD3 with $\{s_i, a_i, \hat{r}_i, s'_i\}_{i=1,\dots,N}$ to optimize θ and ψ
    $t \leftarrow t + 1$
end

Algorithm 4: MVPI-PPO

Input:
θ, ψ: parameters for the policy π and the value function v_π
K, λ: rollout length and weight for variance

while *True* **do**
    Empty a buffer $\mathcal{M}$
    Run π for K steps in the environment, storing $\{s_i, a_i, r_i, s_{i+1}\}_{i=1,\dots,K}$ into $\mathcal{M}$
    $y \leftarrow \frac{1}{K} \sum_{i=1}^{K} r_i$
    for $i = 1, \dots, K$ **do**
        $\hat{r}_i \leftarrow r_i - \lambda r_i^2 + 2\lambda r_i y$
    end
    Use PPO with $\{s_i, a_i, \hat{r}_i, s_{i+1}\}_{i=1,\dots,K}$ to optimize θ and ψ
end

the same hyperparameters as Schulman et al. (2017). We implement the methods of Prashanth and Ghavamzadeh (2013); Tamar et al. (2012) and MVP with multiple parallelized actors like A2C in Dhariwal et al. (2017) and inherit the common hyperparameters from Dhariwal et al. (2017).

Hyperparameters of Prashanth and Ghavamzadeh (2013): To increase stability, we treat λ as a hyperparameter instead of a variable. Consequently, ξ does not matter. We tune λ from $\{0.5, 1, 2\}$. We set the perturbation β in Prashanth and Ghavamzadeh (2013) to 10^{-4}. We use 16 parallelized actors. The initial learning rate of the RMSprop optimizer is 7×10^{-5}, tuned from $\{7 \times 10^{-5}, 7 \times 10^{-4}, 7 \times 10^{-3}\}$. We also test the Adam optimizer, which performs the same as the RMSprop optimizer. We use policy entropy as a regularization term, whose weight is 0.01. The discount factor is 0.99. We clip the gradient by norm with a threshold 0.5.

Hyperparameters of Tamar et al. (2012): We tune λ from $\{0.5, 1, 2\}$. We use $\xi = 50$, tuned from $\{1, 10, 50, 100\}$. We set the initial learning rate of the RMSprop optimizer to 7×10^{-4}, tuned from $\{7 \times 10^{-5}, 7 \times 10^{-4}, 7 \times 10^{-3}\}$. We also test the Adam optimizer, which performs the same as the RMSprop optimizer. The learning rates for the running estimates of $\mathbb{E}[G_0]$ and $\mathbb{V}(G_0)$ is 100 times of the initial learning rate of the RMSprop optimizer. We use 16 parallelized actors. We use policy entropy as a regularization term, whose weight is 0.01. We clip the gradient by norm with a threshold 0.5.

Hyperparameters of Xie et al. (2018): We tune λ from $\{0.5, 1, 2\}$. We set the initial learning rate of the RMSprop optimizer to 7×10^{-4}, tuned from $\{7 \times 10^{-5}, 7 \times 10^{-4}, 7 \times 10^{-3}\}$. We also test the Adam optimizer, which performs the same as the RMSprop optimizer. We use 16 parallelized actors. We use policy entropy as a regularization term, whose weight is 0.01. We clip the gradient by norm with a threshold 0.5.

Computing Infrastructure: We conduct our experiments on an Nvidia DGX-1 with PyTorch, though no GPU is used.

In our off-line off-policy experiments, we set K to 10^3 and use tabular representation for ρ_π, q_π. For π, we use a softmax policy with tabular logits.

C Other Experimental Results

We report the empirical results with $\lambda = 0.5$ and $\lambda = 2$ in Figure 3, Table 2, Figure 4, and Table 3.

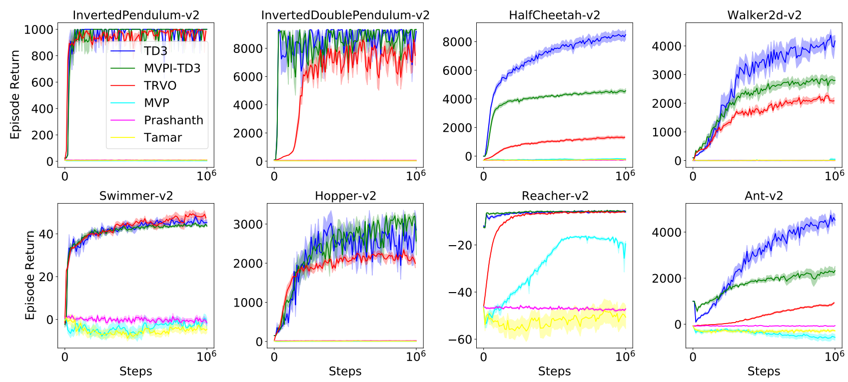


Figure 3: Figure 1 with $\lambda = 0.5$.

	Δ_J^{TRVO}	Δ_{mean}^{TRVO}	$\Delta_{variance}^{TRVO}$	Δ_{SR}^{TRVO}	Δ_J^{MVPI}	Δ_{mean}^{MVPI}	$\Delta_{variance}^{MVPI}$	Δ_{SR}^{MVPI}
InvertedP.	-581%	-3%	NaN	NaN	0%	0%	NaN	NaN
InvertedD.P.	-1407%	-24%	1337%	-80%	-10%	0%	10%	-4%
HalfCheetah	83%	-84%	-83%	-62%	66%	-46%	-65%	-8%
Walker2d	-44%	-51%	42%	-59%	91%	-34%	-90%	107%
Swimmer	0%	4%	264%	-46%	-4%	-4%	4%	-6%
Hopper	-19%	-28%	18%	-34%	74%	-10%	-73%	74%
Reacher	-26%	-4%	80%	23%	3%	5%	2%	6%
Ant	93%	-81%	-92%	-30%	88%	-49%	-88%	44%

Table 2: Table 1 with $\lambda = 0.5$.

	Δ_J^{TRVO}	Δ_{mean}^{TRVO}	$\Delta_{variance}^{TRVO}$	Δ_{SR}^{TRVO}	Δ_J^{MVPI}	Δ_{mean}^{MVPI}	$\Delta_{variance}^{MVPI}$	Δ_{SR}^{MVPI}
InvertedP.	-1719%	-2%	NaN	NaN	-9%	-9%	NaN	NaN
InvertedD.P.	-1707%	-31%	1686%	-84%	78%	-28%	-77%	50%
HalfCheetah	92%	-88%	-92%	-57%	96%	-69%	-96%	60%
Walker2d	78%	-71%	-78%	-39%	98%	-63%	-98%	141%
Swimmer	-80%	-15%	1151%	-76%	-24%	-11%	231%	-51%
Hopper	73%	-49%	-73%	-2%	78%	-20%	-77%	69%
Reacher	-34%	1%	55%	21%	4%	3%	-5%	0%
Ant	97%	-95%	-97%	-72%	99%	-69%	-99%	221%

Table 3: Table 1 with $\lambda = 2$.

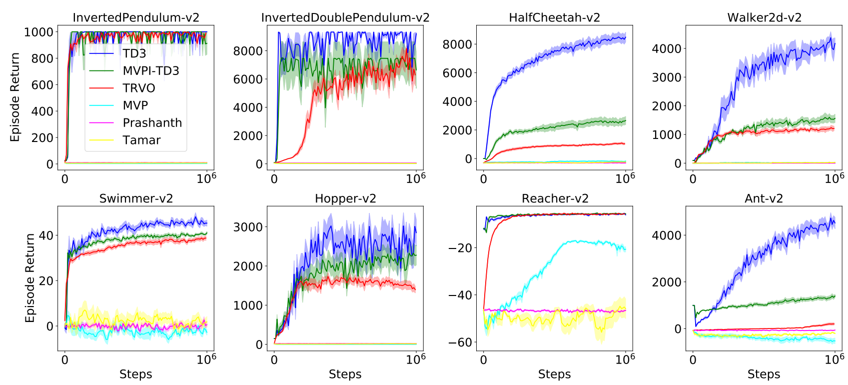


Figure 4: Figure 1 with $\lambda = 2$.