Reinforcement Learning with Perturbed Rewards

Jingkang Wang
University of Toronto & Vector Institute
Toronto, Canada
wangjk@cs.toronto.edu

Yang Liu
University of California, Santa Cruz
California, USA
yangliu@ucsc.edu

Bo Li
University of Illinois, Urbana–Champaign
Illinois, USA
lbo@illinois.edu

Abstract

Recent studies have shown that reinforcement learning (RL) models are vulnerable in various noisy scenarios. For instance, the observed reward channel is often subject to noise in practice (e.g., when rewards are collected through sensors), and is therefore not credible. In addition, for applications such as robotics, a deep reinforcement learning (DRL) algorithm can be manipulated to produce arbitrary errors by receiving corrupted rewards. In this paper, we consider noisy RL problems with *perturbed rewards*, which can be approximated with a confusion matrix. We develop a robust RL framework that enables agents to learn in noisy environments where only perturbed rewards are observed. Our solution framework builds on existing RL/DRL algorithms and firstly addresses the biased noisy reward setting without any assumptions on the true distribution (e.g., zero-mean Gaussian noise as made in previous works). The core ideas of our solution include estimating a reward confusion matrix and defining a set of unbiased surrogate rewards. We prove the convergence and sample complexity of our approach. Extensive experiments on different DRL platforms show that trained policies based on our estimated surrogate reward can achieve higher expected rewards, and converge faster than existing baselines. For instance, the state-of-the-art PPO algorithm is able to obtain 84.6% and 80.8% improvements on *average score* for five Atari games, with error rates as 10% and 30% respectively.

Introduction

Designing a suitable reward function plays a critical role in building reinforcement learning models for real-world applications. Ideally, one would want to customize reward functions to achieve application-specific goals (Hadfield-Menell et al. 2017). In practice, however, it is difficult to design a reward function that produces credible rewards in the presence of noise. This is because the output from any reward function is subject to multiple kinds of randomness:

- *Inherent Noise*. For instance, sensors on a robot will be affected by physical conditions such as temperature and lighting, and therefore will report back noisy observed rewards.
- *Application-Specific Noise*. In machine teaching tasks (Loftin et al. 2014), when an RL agent receives feedback/instructions, different human instructors might provide drastically different feedback that leads to biased rewards for machine.
- *Adversarial Noise*. Huang et al. have shown that by adding adversarial perturbation to each frame of the game, they can mislead pre-trained RL policies arbitrarily.

Assuming an arbitrary noise model makes solving this noisy RL problem extremely challenging. Instead, we focus on a specific noisy reward model which we call *perturbed rewards*, where the observed rewards by RL agents are learnable. The perturbed rewards are generated via a confusion matrix that flips the true reward to another one according to a certain distribution. This is not a very restrictive setting (Everitt et al. 2017) to start with, even considering that the noise could be adversarial: For instance, adversaries can manipulate sensors via reversing the reward value.

In this paper, we develop an unbiased reward estimator aided robust framework that enables an RL agent to learn in a noisy environment with observing only perturbed rewards. The main challenge is that the observed rewards are likely to be biased, and in RL or DRL the accumulated errors could amplify the reward estimation error over time. To the best of our knowledge, this is the first work addressing robust RL in the biased rewards setting (existing work need to assume the unbiased noise distribution). We do not require any assumption on the knowledge of true reward distribution or adversarial strategies, other than the fact that the generation of noises follows a reward confusion matrix. We address the issue of estimating the reward confusion matrices by proposing an efficient and flexible estimation module for settings with deterministic rewards.

Everitt et al. provided preliminary studies for this noisy reward problem and gave some general negative results. The authors proved a *No Free Lunch* theorem, which is, without any assumption about what the reward corruption is, all agents can be misled. Our results do not contradict with the results therein, as we consider a stochastic noise generation model (that leads to a set of perturbed rewards).

We analyze the convergence and sample complexity for the policy trained using our proposed method based on sur-

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

rogate rewards, using Q-Learning as an example. We then conduct extensive experiments on OpenAI Gym (Brockman et al. 2016) and show that the proposed reward robust RL method achieves comparable performance with the policy trained using the true rewards. In some cases, our method even achieves higher cumulative reward - this is surprising to us at first, but we conjecture that the inserted noise together with our noise-removal unbiased estimator add another layer of exploration, which proves to be beneficial in some settings.

Our contributions are summarized as follows: (1) We formulate and generalize the idea of defining a simple but effective unbiased estimator for true rewards under reinforcement learning setting. The proposed estimator helps guarantee the convergence to the optimal policy even when the RL agents only have noisy observations of the rewards. (2) We analyze the convergence to the optimal policy and the finite sample complexity of our reward-robust RL methods, using Q-Learning as the example. (3) Extensive experiments on OpenAI Gym show that our proposed algorithms perform robustly even at high noise rates.

Related Work

Robust Reinforcement Learning It is known that RL algorithms are vulnerable in noisy environments (Irpan 2018). Recent studies (Huang et al. 2017; Kos and Song 2017; Lin et al. 2017) show that learned RL policies can be easily misled with small perturbations in observations. The presence of noise is very common in real-world environments, especially in robotics-relevant applications (Deisenroth, Rasmussen, and Fox 2011; Loftin et al. 2014). Consequently, robust RL algorithms have been widely studied, aiming to train a robust policy that is capable of withstanding perturbed observations (Teh et al. 2017; Pinto et al. 2017; Gu, Jia, and Choset 2018) or transferring to unseen environments (Rajeswaran et al. 2016; Fu, Luo, and Levine 2017). However, these algorithms mainly focus on noisy vision observations, instead of observed rewards. Some early works (Moreno et al. 2006; Strens 2000; Romoff et al. 2018) on noisy reward RL rely on the knowledge of unbiased noise distribution, which limits their applicability to more general biased rewards settings. A couple of recent works (Lim, Xu, and Mannor 2016; Roy, Xu, and Pokutta 2017) have looked into a parallel question of training robust RL algorithms with uncertainty in models.

Learning with Noisy Data Learning appropriately with biased data has received quite a bit of attention in recent machine learning studies (Natarajan et al. 2013; Scott et al. 2013; Scott 2015; Sukhbaatar and Fergus 2014; van Rooyen and Williamson 2015; Menon et al. 2015). The idea of this line of works is to define unbiased surrogate loss functions to recover the true loss using the knowledge of the noise. Our work is the first to formally establish this extension both theoretically and empirically. Our quantitative understandings will provide practical insights when implementing reinforcement learning algorithms in noisy environments.

Problem Formulation and Preliminaries

In this section, we define our problem of learning from perturbed rewards in reinforcement learning. Throughout this paper, we will use *perturbed reward* and *noisy reward* interchangeably, considering that the noise could come from both intentional perturbation and natural randomness. In what follows, we formulate our Markov Decision Process (MDP) and reinforcement learning (RL) problem with perturbed rewards.

Reinforcement Learning: The Noise-Free Setting

Our RL agent interacts with an unknown environment and attempts to maximize the total of its collected reward. The environment is formalized as a Markov Decision Process (MDP), denoting as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$. At each time t, the agent in state $s_t \in \mathcal{S}$ takes an action $a_t \in \mathcal{A}$, which returns a reward $r(s_t, a_t, s_{t+1}) \in \mathcal{R}$ (which we will also shorthand as r_t) [1], and leads to the next state $s_{t+1} \in \mathcal{S}$ according to a transition probability kernel $\mathcal{P}$. $\mathcal{P}$ encodes the probability $\mathbb{P}_a(s_t, s_{t+1})$, and commonly is unknown to the agent. The agent's goal is to learn the optimal policy, a conditional distribution $\pi(a|s)$ that maximizes the state's value function. The value function calculates the cumulative reward the agent is expected to receive given it would follow the current policy π after observing the current state s_t: $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right]$, where $0 \leq \gamma \leq 1$ is a discount factor ($\gamma = 1$ indicates an undiscounted MDP setting (Schwartz 1993; Sobel 1994; Kakade 2003)). Intuitively, the agent evaluates how preferable each state is, given the current policy. From the Bellman Equation, the optimal value function is given by $V^*(s) = \max_{a \in \mathcal{A}} \sum_{s_{t+1} \in \mathcal{S}} \mathbb{P}_a(s_t, s_{t+1}) \left[r_t + \gamma V^*(s_{t+1}) \right]$. It is a standard practice for RL algorithms to learn a state-action value function, also called the Q-function. Q-function denotes the expected cumulative reward if agent chooses a in the current state and follows π thereafter: $Q^\pi(s, a) = \mathbb{E}_\pi \left[r(s_t, a_t, s_{t+1}) + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a \right]$.

Perturbed Reward in RL

In many practical settings, the RL agent does not observe the reward feedback perfectly. We consider the following MDP with perturbed reward, denoting as $\tilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, C, \mathcal{P}, \gamma \rangle$[2]: instead of observing $r_t \in \mathcal{R}$ at each time t directly (following his action), our RL agent only observes a perturbed version of r_t, denoting as $\tilde{r}_t \in \tilde{\mathcal{R}}$. For most of our presentations, we focus on the cases where $\mathcal{R}$, $\tilde{\mathcal{R}}$ are finite sets; but our results generalize to the continuous reward settings with discretization techinques.

The generation of $\tilde{r}$ follows a certain function $C : \mathcal{S} \times \mathcal{R} \rightarrow \tilde{\mathcal{R}}$. To let our presentation stay focused, we consider the following state-independent flipping error rates model: if the rewards are binary (consider r_+ and r_-), $\tilde{r}(s_t, a_t, s_{t+1})$

[1] We do not restrict the reward to deterministic in general, except for when we need to estimate the noises in the perturbed reward (Section 3.3).

[2] The MDP with perturbed reward can equivalently be defined as a tuple $\tilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \tilde{\mathcal{R}}, \mathcal{P}, \gamma \rangle$, with the perturbation function C implicitly defined as the difference between $\mathcal{R}$ and $\tilde{\mathcal{R}}$.

($\tilde{r}_t$) can be characterized by the following noise rate parameters e_+, e_-: $e_+ = \mathbb{P}(\tilde{r}(s_t, a_t, s_{t+1}) = r_- | r(s_t, a_t, s_{t+1}) = r_+), e_- = \mathbb{P}(\tilde{r}(s_t, a_t, s_{t+1}) = r_+ | r(s_t, a_t, s_{t+1}) = r_-)$. When the signal levels are beyond binary, suppose there are M outcomes in total, denoting as $[R_0, R_1, \cdots, R_{M-1}]$. $\tilde{r}_t$ will be generated according to the following confusion matrix $\mathbf{C}_{M\times M}$ where each entry $c_{j,k}$ indicates the flipping probability for generating a perturbed outcome: $c_{j,k} = \mathbb{P}(\tilde{r}_t = R_k | r_t = R_j)$. Again we'd like to note that we focus on settings with finite reward levels for most of our paper, but we provide discussions later on how to handle continuous rewards.

In the paper, we also generalize our solution to the case without knowing the noise rates (i.e., the reward confusion matrices) for settings in which the rewards for each (state, action) pair is deterministic, which is different from the assumption of knowing them as adopted in many supervised learning works (Natarajan et al. 2013). Instead we will estimate the confusion matrices in our framework.

Learning with Perturbed Rewards

In this section, we first introduce an unbiased estimator for binary rewards in our reinforcement learning setting when the error rates are known. This idea is inspired by (Natarajan et al. 2013), but we will extend the method to the multi-outcome, as well as the continuous reward settings.

Unbiased Estimator for True Reward

With the knowledge of noise rates (reward confusion matrices), we are able to establish an unbiased approximation of the true reward in a similar way as done in (Natarajan et al. 2013). We will call such a constructed unbiased reward as a *surrogate reward*. To give an intuition, we start with replicating the results for binary reward $\mathcal{R} = \{r_-, r_+\}$ in our RL setting:

Lemma 1. *Let r be bounded. Then, if we define,*

$$\hat{r}(s_t, a_t, s_{t+1}) := \begin{cases} \frac{(1-e_-)\cdot r_+ - e_+ \cdot r_-}{1-e_+-e_-} & (\tilde{r}(s_t, a_t, s_{t+1}) = r_+) \\ \frac{(1-e_+)\cdot r_- - e_- \cdot r_+}{1-e_+-e_-} & (\tilde{r}(s_t, a_t, s_{t+1}) = r_-) \end{cases} \tag{1}$$

we have for any $r(s_t, a_t, s_{t+1})$, $\mathbb{E}_{\tilde{r}|r}[\hat{r}(s_t, a_t, s_{t+1})] = r(s_t, a_t, s_{t+1})$.

In the standard supervised learning setting, the above property guarantees convergence - as more training data are collected, the empirical surrogate risk converges to its expectation, which is the same as the expectation of the true risk (due to unbiased estimators). This is also the intuition why we would like to replace the reward terms with surrogate rewards in our RL algorithms.

The above idea can be generalized to the multi-outcome setting in a fairly straight-forward way. Define $\hat{\mathbf{R}} := [\hat{r}(\tilde{r} = R_0), \hat{r}(\tilde{r} = R_1), ..., \hat{r}(\tilde{r} = R_{M-1})]$, where $\hat{r}(\tilde{r} = R_k)$ denotes the value of the surrogate reward when the observed reward is R_k. Let $\mathbf{R} = [R_0; R_1; \cdots; R_{M-1}]$ be the bounded reward matrix with M values. We have the following results:

Lemma 2. *Suppose $\mathbf{C}_{M\times M}$ is invertible. With defining:*

$$\hat{\mathbf{R}} = \mathbf{C}^{-1} \cdot \mathbf{R} \tag{2}$$

we have for any $r(s_t, a_t, s_{t+1})$, $\mathbb{E}_{\tilde{r}|r}[\hat{r}(s_t, a_t, s_{t+1})] = r(s_t, a_t, s_{t+1})$.

Continuous reward When the reward signal is continuous, we discretize it into M intervals, and view each interval as a reward level, with its value approximated by its middle point. With increasing M, this quantization error can be made arbitrarily small. Our method is then the same as the solution for the multi-outcome setting, except for replacing rewards with discretized ones. Note that the finer-degree quantization we take, the smaller the quantization error - but we would suffer from learning a bigger reward confusion matrix. This is a trade-off question that can be addressed empirically.

So far we have assumed knowing the confusion matrices and haven't restricted our solution to any specific setting, but we will address this additional estimation issue focusing on determinisitc reward settings, and present our complete algorithm therein.

Convergence and Sample Complexity: Q-Learning

We now analyze the convergence and sample complexity of our surrogate reward based RL algorithms (with assuming knowing $\mathbf{C}$), taking Q-Learning as an example.

Convergence guarantee First, the convergence guarantee is stated in the following theorem:

Theorem 1. *Given a finite MDP, denoting as $\hat{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{R}}, \mathcal{P}, \gamma \rangle$, the Q-learning algorithm with surrogate rewards, given by the update rule,*

$$Q_{t+1}(s_t, a_t) = (1-\alpha_t)Q(s_t, a_t) + \alpha_t \left[\hat{r}_t + \gamma \max_{b\in\mathcal{A}} Q(s_{t+1}, b) \right], \tag{3}$$

converges w.p.1 to the optimal Q-function as long as $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$.

Note that the term on the right hand of Eqn. (3) includes surrogate reward $\hat{r}$ estimated using Eqn. (1) and Eqn. (2). Theorem 1 states that that agents will converge to the optimal policy *w.p.1* when replacing the rewards with surrogate rewards, despite of the noises in the observed rewards. This result is not surprising - though the surrogate rewards introduce larger variance, we are grateful of their unbiasedness, which grants us the convergence. In other words, the addition of the perturbed reward does not affect the convergence guarantees of Q-Learning with surrogate rewards.

Sample complexity To establish our sample complexity results, we first introduce a *generative model* following previous literature (Kearns and Singh 1998; 2000; Kearns, Mansour, and Ng 1999). This is a practical MDP setting to simplify the analysis.

Definition 1. *A generative model $G(\mathcal{M})$ for an MDP $\mathcal{M}$ is a sampling model which takes a state-action pair (s_t, a_t) as input, and outputs the corresponding reward $r(s_t, a_t)$ and the next state s_{t+1} randomly with the probability of $\mathbb{P}_a(s_t, s_{t+1})$, i.e., $s_{t+1} \sim \mathbb{P}(\cdot|s, a)$.*

Exact value iteration is impractical if the agents follow the generative models above exactly (Kakade 2003). Consequently, we introduce a *phased Q-Learning* which is similar to the ones presented in (Kakade 2003; Kearns and Singh 1998) for the convenience of proving our sample complexity results. We briefly outline *phased Q-Learning* as follows - the complete description can be found in Appendix A (Algorithm 2).

Definition 2. *Phased Q-Learning algorithm takes m samples per phase by calling generative model $G(\mathcal{M})$. It uses the collected m samples to estimate the transition probability $\mathcal{P}$ and then update the estimated value function per phase. Calling generative model $G(\hat{\mathcal{M}})$ means that surrogate rewards $\hat{r}$ are returned and used to update the value function.*

The sample complexity of *Phased Q-Learning* is given as follows:

Theorem 2. *(Upper Bound) Let $r \in [0, R_{\max}]$ be bounded reward, $\mathbf{C}$ be an invertible reward confusion matrix with $\det(\mathbf{C})$ denoting its determinant. For an appropriate choice of m, the Phased Q-Learning algorithm calls the generative model $G(\hat{\mathcal{M}})$ $O\left(\frac{|\mathcal{S}||\mathcal{A}|T}{\epsilon^2(1-\gamma)^2\det(\mathbf{C})^2}\log\frac{|\mathcal{S}||\mathcal{A}|T}{\delta}\right)$ times in T epochs, and returns a policy such that for all state $s \in \mathcal{S}$, $|V_\pi(s) - V^*(s)| \le \epsilon$, $\epsilon > 0$, w.p. $\ge 1 - \delta$, $0 < \delta < 1$.*

Theorem 2 states that, to guarantee the convergence to the optimal policy, the number of samples needed is no more than $O(1/\det(\mathbf{C})^2)$ times of the one needed when the RL agent observes true rewards perfectly. This additional constant is the price we pay for the noise presented in our learning environment. When the noise level is high, we expect to see a much higher $1/\det(\mathbf{C})^2$; otherwise when we are in a low-noise regime , Q-Learning can be very efficient with surrogate reward (Kearns and Singh 2000). Note that Theorem 2 gives the upper bound in discounted MDP setting; for undiscounted setting ($\gamma = 1$), the upper bound is at the order of $O\left(\frac{|\mathcal{S}||\mathcal{A}|T^3}{\epsilon^2\det(\mathbf{C})^2}\log\frac{|\mathcal{S}||\mathcal{A}|T}{\delta}\right)$. This result is not surprising, as the phased Q-Learning helps smooth out the noise in rewards in consecutive steps. We will experimentally test how the bias removal step performs without explicit phases.

While the surrogate reward guarantees the unbiasedness, we sacrifice the variance at each of our learning steps, and this in turn delays the convergence (as also evidenced in the sample complexity bound). It can be verified that the variance of surrogate reward is bounded when $\mathbf{C}$ is invertible, and it is always higher than the variance of true reward. This is summarized in the following theorem:

Theorem 3. *Let $r \in [0, R_{\max}]$ be bounded reward and confusion matrix $\mathbf{C}$ is invertible. Then, the variance of surrogate reward $\hat{r}$ is bounded as follows: $\mathbf{Var}(r) \le \mathbf{Var}(\hat{r}) \le \frac{M^2}{\det(\mathbf{C})^2} \cdot R^2_{\max}$.*

To give an intuition of the bound, when we have binary reward, the variance for surrogate reward bounds as follows: $\mathbf{Var}(r) \le \mathbf{Var}(\hat{r}) \le \frac{4R^2_{\max}}{(1-e_+-e_-)^2}$. As $e_- + e_+ \to 1$, the variance becomes unbounded and the proposed estimator is no longer effective, nor will it be well-defined.

Variance reduction In practice, there is a trade-off question between bias and variance by tuning a linear combination of $\mathbf{R}$ and $\hat{\mathbf{R}}$, *i.e.*, $\mathbf{R}_{proxy} = \eta\mathbf{R} + (1-\eta)\hat{\mathbf{R}}$, via choosing an appropriate $\eta \in [0, 1]$. Other variance reduction techniques in RL with noisy environment, for instance (Romoff et al. 2018), can be combined with our proposed bias removal technique too. We test them in the experiment section.

Estimation of Confusion Matrices

In previous solutions, we have assumed the knowledge of reward confusion matrices, in order to compute the surrogate reward. This knowledge is often not available in practice. Estimating these confusion matrices is challenging without knowing any ground truth reward information; but we would like to remark that efficient algorithms have been developed to estimate the confusion matrices in supervised learning settings (Bekker and Goldberger 2016; Liu and Liu 2017; Khetan, Lipton, and Anandkumar 2017; Hendrycks et al. 2018). The idea in these algorithms is to dynamically refine the error rates based on aggregated rewards. Note this approach is not different from the inference methods in aggregating crowdsourcing labels, as referred in the literature (Dawid and Skene 1979; Karger, Oh, and Shah 2011; Liu, Peng, and Ihler 2012). We adapt this idea to our reinforcement learning setting, which is detailed as follows.

The estimation procedure is only for the case with deterministic reward, but not for stochastic rewards. The reason is that we will use repeated observations to refine an estimated ground truth reward, which will be leveraged to estimate the confusion matrix. With uncertainty in the true reward, it is not possible to distinguish a clean case with true reward $\mathbf{C} \cdot \mathbf{R}$ from the perturbed reward case with true reward $\mathbf{R}$ and added noise by confusion matrix $\mathbf{C}$.

At each training step, the RL agent collects the noisy reward and the current *state-action* pair. Then, for each pair in $\mathcal{S} \times \mathcal{A}$, the agent predicts the true reward based on accumulated historical observations of reward for the corresponding *state-action* pair via, e.g., averaging (majority voting). Finally, with the predicted true reward and the accuracy (error rate) for each state-action pair, the estimated reward confusion matrices $\tilde{\mathbf{C}}$ are given by

$$\bar{r}(s, a) = \underset{R_i \in \mathcal{R}}{\arg\max}\ \#[\tilde{r}(s, a) = R_i], \tag{4}$$

$$\tilde{c}_{i,j} = \frac{\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \#\left[\tilde{r}(s, a) = R_j|\bar{r}(s, a) = R_i\right]}{\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \#[\bar{r}(s, a) = R_i]}, \tag{5}$$

where in above $\#\,[\cdot]$ denotes the number of state-action pair that satisfies the condition $[\cdot]$ in the set of observed rewards $\tilde{R}(s, a)$ (see Algorithm 1); $\bar{r}(s, a)$ and $\tilde{r}(s, a)$ denote predicted true rewards (using majority voting) and observed rewards when the state-action pair is (s, a). We break potential ties in Eqn. (4) equally likely. The above procedure of updating $\tilde{c}_{i,j}$ continues indefinitely as more observation

Algorithm 1 Reward Robust RL (sketch)

```
Input: $\tilde{\mathcal{M}}$, $\tilde{R}(s, a)$, $\eta$
Output: $Q(s, a)$, $\pi(s)$
Initialize value function $Q(s, a)$ arbitrarily.
while $Q$ is not converged do
    Initialize state $s \in \mathcal{S}$, observed reward set $\tilde{R}(s, a)$
    Set confusion matrix $\tilde{\mathbf{C}}$ as identity matrix $\mathbf{I}$
    while $s$ is not terminal do
        Choose $a$ from $s$ using policy derived from $Q$
        Take action $a$, observe $s'$ and noisy reward $\tilde{r}$
        if collecting enough $\tilde{r}$ for all $\mathcal{S} \times \mathcal{A}$ pairs then
            Get predicted true reward $\bar{r}$ using majority voting
            Re-estimate $\tilde{\mathbf{C}}$ based on $\tilde{r}$ and $\bar{r}$ (using Eqn. 5)
        end if
        Obtain surrogate reward $\dot{r}$ ($\hat{\mathbf{R}} = (1 - \eta) \cdot \mathbf{R} + \eta \cdot \tilde{\mathbf{C}}^{-1}\mathbf{R}$)
        Update $Q$ using surrogate reward
        $s \leftarrow s'$
    end while
end while
return $Q(s, a)$ and $\pi(s)$
```

arrives. Our final definition of surrogate reward replaces a known reward confusion $\mathbf{C}$ in Eqn. (2) with our estimated one $\tilde{\mathbf{C}}$. We denote this estimated surrogate reward as $\dot{r}$.

We present (*Reward Robust RL*) in Algorithm 1Note that the algorithm is rather generic, and we can plug in any exisitng RL algorithm into our reward robust one, with only changes in replacing the rewards with our estimated surrogate rewards.

Experimental Results

In this section, we conduct extensive experiments to evaluate the noisy reward robust RL mechanism with different games, under various noise settings[3].

Experimental Setup

Environments and RL Algorithms To fully test the performance under different environments, we evaluate the proposed robust reward RL method on two classic control games (CartPole, Pendulum) and seven Atari 2600 games (AirRaid, Alien, Carnival, MsPacman, Pong, Phoenix, Seaquest), which encompass a large variety of environments, as well as rewards. Specifically, the rewards could be unary (CartPole), binary (most of Atari games), multivariate (Pong) and even continuous (Pendulum). A set of state-of-the-art RL algorithms are experimented with, while training under different amounts of noise (see Appendix B for more details).

Reward Post-Processing For each game and RL algorithm, we test the performance for learning with true rewards, noisy rewards and surrogate rewards. Both symmetric and asymmetric noise settings with different noise levels are tested. For symmetric noise, the confusion matrices are symmetric. As for asymmetric noise, two types of random noise are tested: 1) *rand-one*, each reward level can only be perturbed into another reward; 2) *rand-all*, each reward could be perturbed to any other reward, via adding a random noise matrix. To measure the amount of noise *w.r.t* confusion matrices, we define the weight of noise ω in Appendix B. The larger ω is, the higher the noise rates are.

Table 1: Average scores of various RL algorithms on CartPole and Pendulum with noisy rewards ($\tilde{r}$) and surrogate rewards under known ($\hat{r}$) or estimated ($\dot{r}$) noise rates. Note that the results for last two algorithms DDPG (rand-one) & NAF (rand-all) are on Pendulum, but the others are on CartPole.

Noise Rate	Reward	Q-Learn	CEM	SARSA	DQN	DDQN	DDPG	NAF
$\omega = 0.1$	$\tilde{r}$	170.0	98.1	165.2	187.2	**187.8**	-1.03	-4.48
	$\hat{r}$	165.8	**108.9**	**173.6**	**200.0**	181.4	**-0.87**	**-0.89**
	$\dot{r}$	**181.9**	99.3	171.5	**200.0**	185.6	-0.90	-1.13
$\omega = 0.3$	$\tilde{r}$	134.9	28.8	144.4	173.4	168.6	-1.23	-4.52
	$\hat{r}$	149.3	**85.9**	152.4	175.3	198.7	**-1.03**	**-1.15**
	$\dot{r}$	**161.1**	**82.2**	**159.6**	**186.7**	**200.0**	-1.05	-1.36
$\omega = 0.7$	$\tilde{r}$	56.6	19.2	12.6	17.2	11.8	-8.76	-7.35
	$\hat{r}$	**177.6**	**87.1**	151.4	185.8	**195.2**	**-1.09**	**-2.26**
	$\dot{r}$	172.1	83.0	**174.4**	**189.3**	191.3	–	–

Table 2: Average scores of PPO on five selected games with noisy rewards ($\tilde{r}$) and surrogate rewards under known ($\hat{r}$) or estimated ($\dot{r}$) noise rates.

Noise Rate	Reward	Lift (↑)	Alien	Carnival	Phoenix	MsPacman	Seaquest
$\omega = 0.1$	$\tilde{r}$	–	1835.1	1239.3	4609.0	1709.1	849.2
	$\hat{r}$	**70.4%↑**	1737.0	3966.8	**7586.4**	**2547.3**	1610.6
	$\dot{r}$	**84.6%↑**	**2844.1**	**5515.0**	5668.8	2294.5	**2333.9**
$\omega = 0.3$	$\tilde{r}$	–	538.2	919.9	2600.3	1109.6	408.7
	$\hat{r}$	**119.8%↑**	**1668.6**	**4220.1**	**4171.6**	1470.3	**727.8**
	$\dot{r}$	**80.8%↑**	1542.9	4094.3	2589.1	**1591.2**	262.4
$\omega = 0.7$	$\tilde{r}$	–	495.2	380.3	126.5	491.6	0.0
	$\hat{r}$	**757.4%↑**	**1805.9**	4088.9	**4970.4**	1447.8	**492.5**
	$\dot{r}$	**648.9%↑**	1618.0	**4529.2**	2792.1	**1916.7**	328.5

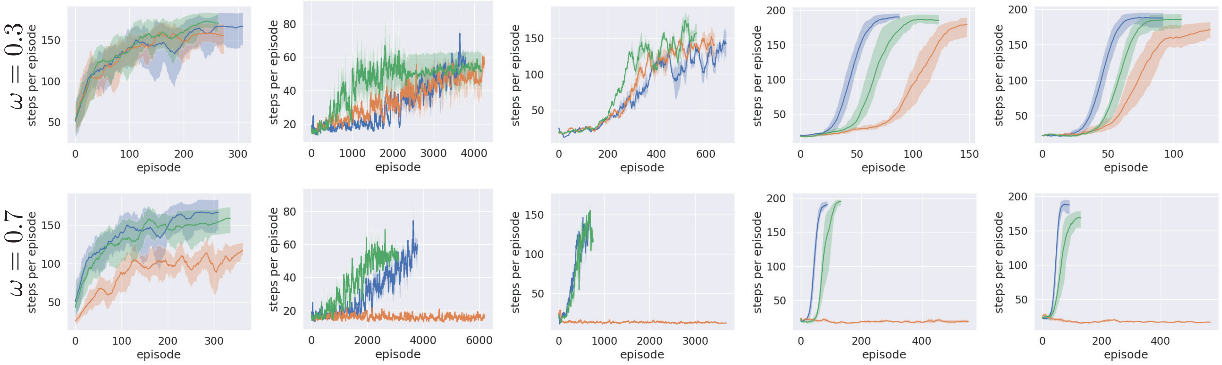
Robustness Evaluation

CartPole The goal in *CartPole* is to prevent the pole from falling by controlling the cart's direction and velocity. The reward is $+1$ for every step taken, including the termination step. When the cart or pole deviates too much or the episode length is longer than 200, the episode terminates. Due to the unary reward $\{+1\}$ in CartPole, a corrupted reward -1 is added as the unexpected error ($e_- = 0$). As a result, the reward space $\mathcal{R}$ is extended to $\{+1, -1\}$. Five algorithms Q-Learning (Watkins and Dayan 1992), CEM (Szita and Lörincz 2006), SARSA (Sutton and Barto 1998), DQN (van Hasselt, Guez, and Silver 2016) and DDQN (Wang et al. 2016) are evaluated.

In Figure 1, we show that our estimator successfully produces meaningful surrogate rewards that adapt the underlying RL algorithms to the noisy settings, without any assumption of the true distribution of rewards. With the noise rate increasing (from 0.1 to 0.9), the models with noisy rewards converge slower due to larger biases. However, we observe that the models always converge to the best score 200 with the help of surrogate rewards.

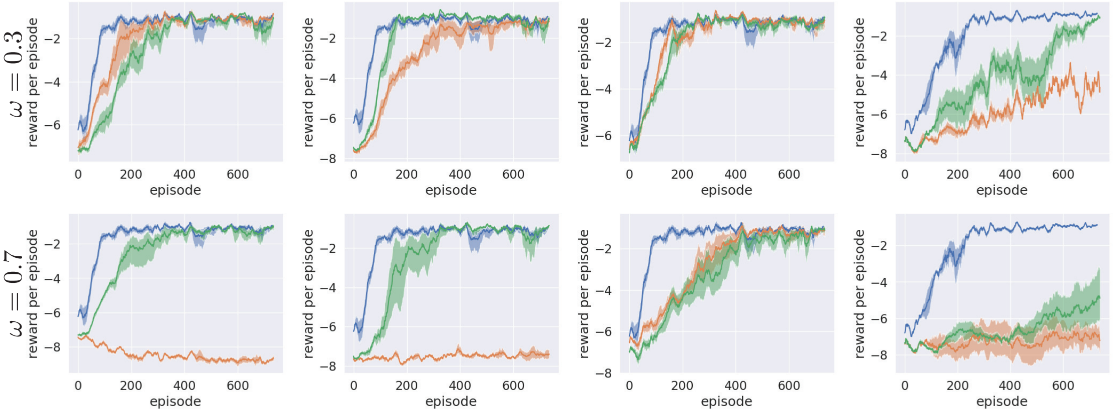
In some circumstances (slight noise - see Figure 1b, 1c),

[3] Due to the page limit, we leave all the proofs, detailed experimental settings, supplementary results and more discussions in the appendix. It is online available: https://arxiv.org/abs/1810.01032



(a) Q-Learning (b) CEM (c) SARSA (d) DQN (e) DDQN

Figure 1: Learning curves from five RL algorithms on CartPole game with true rewards (r) (blue), noisy rewards ($\tilde{r}$) (orange) and estimated surrogate rewards ($\dot{r}$) ($\eta = 1$) (green). Note that $\mathbf{C}$ are unknown to the agents and each experiment is repeated 10 times with different random seeds. We plotted 10% to 90% percentile area with its mean highlighted. Full results are in Appendix D (Figure 6).



(a) DDPG (symmetric) (b) DDPG (rand-one) (c) DDPG (rand-all) (d) NAF (rand-all)

Figure 2: Learning curves from DDPG and NAF on Pendulum game with true rewards (r) (blue), noisy rewards ($\tilde{r}$) (orange) and surrogate rewards ($\hat{r}$) ($\eta = 1$) (green). Both symmetric and asymmetric noise are conduced in the experiments and each experiment is repeated 3 times with different random seeds. Full results are in Appendix D (Figure 9).

the surrogate rewards even lead to faster convergence. This points out an interesting observation: learning with surrogate reward sometimes even outperforms the case with observing the true reward. We conjecture that the way of adding noise and then removing the bias (or moderate noise) introduces implicit exploration. This may also imply why some algorithms with estimated confusion matrices $\tilde{\mathbf{C}}$ leads to better results than with known $\mathbf{C}$ in some cases (Table 1).

Pendulum The goal in *Pendulum* is to keep a frictionless pendulum standing up. Different from the CartPole setting, the rewards in pendulum are continuous: $r \in (-16.28, 0.0]$. The closer the reward is to zero, the better performance the model achieves. For simplicity, we firstly discretized $(-17, 0]$ into 17 intervals: $(-17, -16], (-16, -15], \cdots, (-1, 0]$, with its value approximated using its maximum point. After the quantization step, the surrogate rewards can be estimated using multi-outcome extensions.

We experiment two popular algorithms, DDPG (Lillicrap et al. 2015) and NAF (Gu et al. 2016) in this game. In Figure 2, both algorithms perform well with surrogate rewards under different amounts of noise. In most cases, the biases were corrected in the long-term, even when the amount of noise is extensive (e.g., $\omega = 0.7$). The quantitative scores on CartPole and Pendulum are given in Table 1, where the scores are averaged based on the last 30 episodes. Our reward robust method is able to achieve good scores consistently.

Atari We validate our algorithm on seven Atari 2600 games using the state-of-the-art algorithm PPO (Schulman et al. 2017). The games are chosen to cover a variety of environments. The rewards in the Atari games are clipped into $\{-1, 0, 1\}$. We leave the detailed settings to Appendix B.

Results for PPO on Pong-v4 in symmetric noise setting

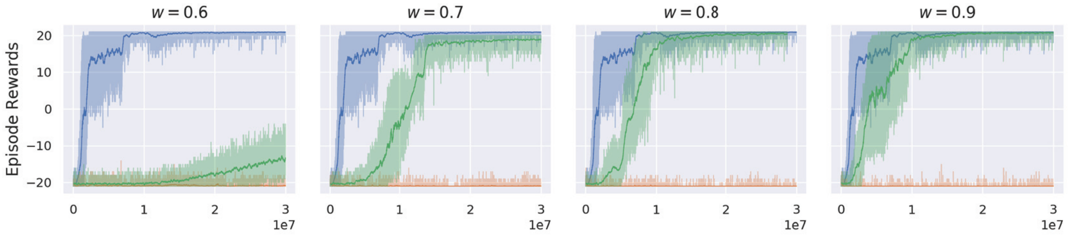
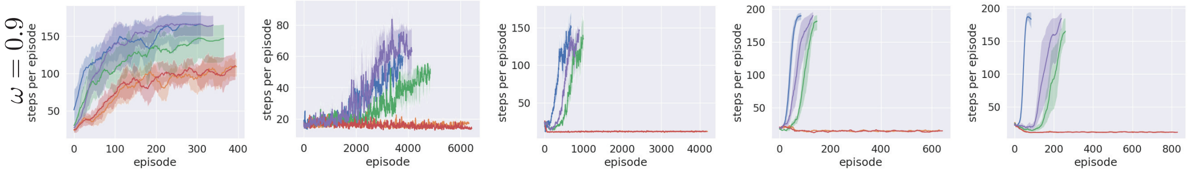


Figure 3: Learning curves from PPO on Pong-v4 game with true rewards (r) (blue), noisy rewards ($\tilde{r}$) (orange) and surrogate rewards ($\eta = 1$) ($\hat{r}$) (green). The noise rate ω increases from 0.6 to 0.9, with a step of 0.1.



(a) Q-Learning (b) CEM (c) SARSA (d) DQN (e) DDQN

Figure 4: Learning curves from five *reward robust* RL algorithms on CartPole game with true rewards (r) (blue), noisy rewards ($\tilde{r}$) (orange), sample-mean noisy rewards (red), estimated surrogate rewards ($\dot{r}$) (green) and sample-mean estimated surrogate rewards (purple).

are presented in Figure 3. More results on other Atari games and noise settings are given in Appendix D. Similar to previous results, our surrogate estimator performs consistently well and helps PPO converge to the optimal policy. Table 2 shows the average scores of PPO on five selected Atari games with different amounts of noise (symmetric & asymmetric). In particular, when the noise rates $e_+ = e_- > 0.3$, agents with surrogate rewards obtain significant amounts of improvements in average scores. For the cases with unknown $\mathbf{C}$ ($\dot{r}$ in Table 2), due to the large state-space (image-input) in confusion matrix estimation, we embed and consider the adjacent frames within a batch as the same state and set the memory size for states as 1,000.

Compatible with Variance Reduction Techniques

As illustrated in Theorem 3, our surrogate rewards introduce larger variance while conducting unbiased estimation, which are likely to decrease the stability of RL algorithms. Apart from the linear combination idea (a linear trade-off), some variance reduction techniques in statistics (e.g., correlated sampling) can also be applied to our method. Specially, Romoff et al. proposed to use a reward estimator to compensate for stochastic corrupted-reward signals. It is worthy to notice that their method is designed for variance reduction under zero-mean noises, which is no longer efficacious in more general *perturbed-reward* setting. However, it is potential to integrate their method with our *robust-reward* RL framework because surrogate rewards provide unbiasedness guarantee.

To verify this idea, we repeated the experiments of *Cartpole* but included variance reduction step for estimated surrogate rewards. Following Romoff et al., we adopted sample mean as a simple approximator during the training and set sequence length as 100. As shown in Figure 4, the models with only variance reduction technique (red lines) suffer from huge regrets, and in general do not converge to the optimal policies. Nevertheless, the variance reduction step helps surrogate rewards (purple lines) to achieve faster convergence or better performance in multiple cases. More quantitative results are provided in Appendix D (Table 4) which show that our surrogate reward benefits from variance reduction techniques (“ours + VRT”), especially when the noise rate is high.

Conclusions

Improving the robustness of RL in the settings with perturbed and noisy rewards is important given the fact that such noises are common when exploring a real-world scenario, such as sensor errors. In addition, in adversarial environments, perturbed reward could be leveraged Different robust RL algorithms have been proposed but they either only focus on the noisy observations or need strong assumption on the unbiased noise distribution for observed rewards. In this paper, we propose the first simple yet effective RL framework for dealing with biased noisy rewards. The convergence guarantee and finite sample complexity of Q-Learning (or its variant) with estimated surrogate rewards are provided. To validate the effectiveness of our approach, extensive experiments are conducted on OpenAI Gym, showing that surrogate rewards successfully rescue models from misleading rewards even at high noise rates. We believe this work will further shed light on exploring robust RL approaches under different noisy rewards observations in real-world environments.

Acknowledgement

This work was supported by National Science Foundation award CCF-1910100 and DARPA award ASED-00009970.

References

Bekker, A. J., and Goldberger, J. 2016. Training deep neural-networks based on unreliable labels. In *ICASSP*, 2682–2686.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym.

Dawid, A. P., and Skene, A. M. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics* 20–28.

Deisenroth, M. P.; Rasmussen, C. E.; and Fox, D. 2011. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems*.

Everitt, T.; Krakovna, V.; Orseau, L.; and Legg, S. 2017. Reinforcement learning with a corrupted reward channel. In *IJCAI*, 4705–4713.

Fu, J.; Luo, K.; and Levine, S. 2017. Learning robust rewards with adversarial inverse reinforcement learning. *CoRR* abs/1710.11248.

Gu, S.; Lillicrap, T. P.; Sutskever, I.; and Levine, S. 2016. Continuous deep q-learning with model-based acceleration. In *ICML*, volume 48, 2829–2838.

Gu, Z.; Jia, Z.; and Choset, H. 2018. Adversary a3c for robust reinforcement learning.

Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S. J.; and Dragan, A. 2017. Inverse reward design. In *NIPS*, 6765–6774.

Hendrycks, D.; Mazeika, M.; Wilson, D.; and Gimpel, K. 2018. Using trusted data to train deep networks on labels corrupted by severe noise. *CoRR* abs/1802.05300.

Huang, S.; Papernot, N.; Goodfellow, I.; Duan, Y.; and Abbeel, P. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*.

Irpan, A. 2018. Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html.

Kakade, S. M. 2003. *On the Sample Complexity of Reinforcement Learning*. Ph.D. Dissertation, University of London.

Karger, D. R.; Oh, S.; and Shah, D. 2011. Iterative learning for reliable crowdsourcing systems. In *NIPS*, 1953–1961.

Kearns, M. J., and Singh, S. P. 1998. Finite-sample convergence rates for q-learning and indirect algorithms. In *NIPS*, 996–1002.

Kearns, M. J., and Singh, S. P. 2000. Bias-variance error bounds for temporal difference updates. In *COLT*, 142–147.

Kearns, M. J.; Mansour, Y.; and Ng, A. Y. 1999. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI*, 1324–1231.

Khetan, A.; Lipton, Z. C.; and Anandkumar, A. 2017. Learning from noisy singly-labeled data. *CoRR* abs/1712.04577.

Kos, J., and Song, D. 2017. Delving into adversarial attacks on deep policies. *CoRR* abs/1705.06452.

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. *CoRR* abs/1509.02971.

Lim, S. H.; Xu, H.; and Mannor, S. 2016. Reinforcement learning in robust markov decision processes. *Math. Oper. Res.* 41(4):1325–1353.

Lin, Y.; Hong, Z.; Liao, Y.; Shih, M.; Liu, M.; and Sun, M. 2017. Tactics of adversarial attack on deep reinforcement learning agents. In *IJCAI*, 3756–3762.

Liu, Y., and Liu, M. 2017. An online learning approach to improving the quality of crowd-sourcing. *IEEE/ACM Transactions on Networking* 25(4):2166–2179.

Liu, Q.; Peng, J.; and Ihler, A. T. 2012. Variational inference for crowdsourcing. In *NIPS*, 701–709.

Loftin, R. T.; Peng, B.; MacGlashan, J.; Littman, M. L.; Taylor, M. E.; Huang, J.; and Roberts, D. L. 2014. Learning something from nothing: Leveraging implicit human feedback strategies. In *RO-MAN*, 607–612. IEEE.

Menon, A.; Van Rooyen, B.; Ong, C. S.; and Williamson, B. 2015. Learning from corrupted binary labels via class-probability estimation. In *ICML*, 125–134.

Moreno, A.; Martín, J. D.; Soria, E.; Magdalena, R.; and Martínez, M. 2006. Noisy reinforcements in reinforcement learning: some case studies based on gridworlds. In *WSEAS*, 296–300.

Natarajan, N.; Dhillon, I. S.; Ravikumar, P. K.; and Tewari, A. 2013. Learning with noisy labels. In *Advances in neural information processing systems*, 1196–1204.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *ICML*, volume 70, 2817–2826.

Rajeswaran, A.; Ghotra, S.; Levine, S.; and Ravindran, B. 2016. Epopt: Learning robust neural network policies using model ensembles. *CoRR* abs/1610.01283.

Romoff, J.; Piché, A.; Henderson, P.; François-Lavet, V.; and Pineau, J. 2018. Reward estimation for variance reduction in deep reinforcement learning. *CoRR* abs/1805.03359.

Roy, A.; Xu, H.; and Pokutta, S. 2017. Reinforcement learning under model mismatch. *CoRR* abs/1706.04711.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *CoRR* abs/1707.06347.

Schwartz, A. 1993. A reinforcement learning method for maximizing undiscounted rewards. In *ICML*, 298–305.

Scott, C.; Blanchard, G.; Handy, G.; Pozzi, S.; and Flaska, M. 2013. Classification with asymmetric label noise: Consistency and maximal denoising. In *COLT*, 489–511.

Scott, C. 2015. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *AISTATS*.

Sobel, M. J. 1994. Mean-variance tradeoffs in an undiscounted MDP. *Operations Research* 42(1):175–183.

Strens, M. J. A. 2000. A bayesian framework for reinforcement learning. In *ICML*, 943–950.

Sukhbaatar, S., and Fergus, R. 2014. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080* 2(3):4.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning.

Szita, I., and Lörincz, A. 2006. Learning tetris using the noisy cross-entropy method. *Neural Computation* 18(12):2936–2941.

Teh, Y. W.; Bapst, V.; Czarnecki, W. M.; Quan, J.; Kirkpatrick, J.; Hadsell, R.; Heess, N.; and Pascanu, R. 2017. Distral: Robust multitask reinforcement learning. In *NIPS*, 4499–4509.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *AAAI*, 2094–2100.

van Rooyen, B., and Williamson, R. C. 2015. Learning in the presence of corruption. *arXiv preprint arXiv:1504.00091*.

Wang, Z.; Schaul, T.; Hessel, M.; van Hasselt, H.; Lanctot, M.; and de Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *ICML*, volume 48, 1995–2003.

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. In *Machine Learning*, 279–292.