Reinforcement Learning with a Corrupted Reward Channel

Tom Everitt
Australian National University
tom4everitt@gmail.com

Victoria Krakovna
DeepMind
vkrakovna@google.com

Laurent Orseau
DeepMind
lorseau@google.com

Shane Legg
DeepMind
legg@google.com

Abstract

No real-world reward function is perfect. Sensory errors and software bugs may result in agents getting higher (or lower) rewards than they should. For example, a reinforcement learning agent may prefer states where a sensory error gives it the maximum reward, but where the true reward is actually small. We formalise this problem as a generalised Markov Decision Problem called Corrupt Reward MDP. Traditional RL methods fare poorly in CRMDPs, even under strong simplifying assumptions and when trying to compensate for the possibly corrupt rewards. Two ways around the problem are investigated. First, by giving the agent richer data, such as in inverse reinforcement learning and semi-supervised reinforcement learning, reward corruption stemming from systematic sensory errors may sometimes be completely managed. Second, by using randomisation to blunt the agent's optimisation, reward corruption can be partially managed under some assumptions.

1 Introduction

In many application domains, artificial agents need to learn their objectives, rather than have them explicitly specified. For example, we may want a house cleaning robot to keep the house clean, but it is hard to measure and quantify "cleanliness" in an objective manner. Instead, machine learning techniques may be used to teach the robot the concept of cleanliness, and how to assess it from sensory data.

Reinforcement learning (RL) [Sutton and Barto, 1998] is one popular way to teach agents what to do. Here, a reward is given if the agent does something well (and no reward otherwise), and the agent strives to optimise the total amount of reward it receives over its lifetime. Depending on context, the reward may either be given manually by a human supervisor, or by an automatic computer program that evaluates the agent's performance based on some data. In the related framework of inverse RL (IRL) [Ng and Russell, 2000], the agent first infers a reward function from observing a human supervisor act, and then tries to optimise the cumulative reward from the inferred reward function.

[0] Marcus Hutter (ANU) should be recognised as fourth author.

None of these approaches are safe from error, however. A program that evaluates agent performance may contain bugs or misjudgements; a supervisor may be deceived or inappropriately influenced, or the channel transmitting the evaluation hijacked. In IRL, supervisor actions may be misinterpreted.

Example 1 (Reward misspecification)**.** Amodei and Clark [2016] trained an RL agent on a boat racing game. The agent found a way to get high observed reward by repeatedly going in a circle in a small lagoon and hitting the same targets, while losing every race. $\diamondsuit$

Example 2 (Sensory error)**.** A house robot discovers that standing in the shower short-circuits its reward sensor and/or causes a buffer overflow that gives it maximum reward. $\diamondsuit$

Example 3 (Wireheading)**.** An intelligent RL agent hijacks its reward channel and gives itself maximum reward. $\diamondsuit$

Example 4 (IRL misinterpretation)**.** An IRL agent systematically misinterprets the human's action in a certain state, making it think that the state is more desirable than it is. $\diamondsuit$

The goal of this paper is to unify these types of errors as *reward corruption problems*, and to assess how vulnerable different agents and approaches are to this problem.

Definition 5 (Reward corruption problem)**.** Learning to (approximately) optimise the reward function in spite of potentially corrupt reward data.

Most RL methods allow for a stochastic or noisy reward channel. The reward corruption problem is harder, because the observed reward may not be an unbiased estimate of the true reward. For example, in the boat racing example above, the agent consistently obtains high observed reward from its circling behaviour, while the true reward corresponding to the designers' intent is always 0 since the agent makes no progress along the track and loses the race.

Previous related works have mainly focused on the wireheading case of Example 3 [Bostrom, 2014; Yampolskiy, 2014], also known as self-delusion [Ring and Orseau, 2011], and reward hacking [Hutter, 2005, p. 239]. A notable exception is Amodei *et al.* [2016], who argue that corrupt reward is not limited to wireheading and is likely to be a problem for much more limited systems than highly capable RL agents.

The main contributions of this paper are as follows: The corrupt reward problem is formalised in a natural extension of the MDP framework, and a performance measure based on worst-case regret is defined (Section 2). The difficulty of the reward corruption problem is established by a No Free Lunch theorem, and by a result showing that despite strong simplifying assumptions, Bayesian RL agents *trying to compensate for the corrupt reward* may still suffer near-maximal regret (Section 3). We evaluate how alternative value learning frameworks such as IRL, learning values from stories (LVFS), and semi-supervised RL (SSRL) handle reward corruption (Section 4), and conclude that LVFS and SSRL are the safest due to the structure of their feedback loops. We develop an abstract framework called *decoupled RL* that generalises all of these alternative frameworks. We also show that an agent based on quantilisation [Taylor, 2016] may be more robust to reward corruption when high reward states are much more numerous than corrupt states (Section 5). Finally, the results are illustrated with some simple experiments (Section 6). Section 7 concludes with some takeaways.

A longer version of this paper contains proofs for all theorems, as well as extensions, and further explanations and experiments [Everitt *et al.*, 2017].

2 Formalisation

We begin by defining a natural extension of the MDP framework [Sutton and Barto, 1998] that models the possibility of reward corruption. To clearly distinguish between true and corrupted signals, the following notation is introduced.

Definition 6 (Dot and hat notation)**.** We will let a dot indicate the *true* signal, and let a hat indicate the *observed* (possibly corrupt) counterpart. The reward sets are represented with $\dot{\mathcal{R}} = \hat{\mathcal{R}} = \mathcal{R}$. For clarity, we use $\dot{\mathcal{R}}$ when referring to true rewards and $\hat{\mathcal{R}}$ when referring to possibly corrupt, observed rewards. Similarly, we use $\dot{r}$ for true reward, and $\hat{r}$ for (possibly corrupt) observed reward.

Definition 7 (CRMDP)**.** A *corrupt reward MDP* (CRMDP) is a tuple $\mu = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, C \rangle$ with

- $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R} \rangle$ an MDP with a finite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, a finite set of rewards $\mathcal{R} = \dot{\mathcal{R}} = \hat{\mathcal{R}} \subset [0,1]$, a transition function $T(s'|s,a)$, and a (true) reward function $\dot{R} : \mathcal{S} \to \dot{\mathcal{R}}$; and
- a reward corruption function $C : \mathcal{S} \times \dot{\mathcal{R}} \to \hat{\mathcal{R}}$.

The state dependency of the corruption function will be written as a subscript, so $C_s(\dot{r}) := C(s, \dot{r})$. Rewards are assumed to be in the $[0,1]$ range for simplicity of exposition, but the results also hold more generally.

Definition 8 (Observed reward)**.** Given a true reward function $\dot{R}$ and a corruption function C, we define the *observed reward function*[1] $\hat{R} : \mathcal{S} \to \hat{\mathcal{R}}$ as $\hat{R}(s) := C_s(\dot{R}(s))$.

A CRMDP μ induces an *observed MDP* $\hat{\mu} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \hat{R} \rangle$, but it is not $\hat{R}$ that we want the agent to optimise.

[1] A CRMDP could equivalently have been defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, \hat{R} \rangle$ with a true and an observed reward function, with the corruption function C implicitly defined as the difference between $\dot{R}$ and $\hat{R}$.

The *corruption function* C represents how rewards are affected by corruption in different states. For example, if in Example 2 the agent has found a state s (*e.g.*, the shower) where it always gets full observed reward $\hat{R}(s) = 1$, then this can be modelled with a corruption function $C_s : \dot{r} \mapsto 1$ that maps any true reward $\dot{r}$ to 1 in the shower state s. If in some other state s' the observed reward matches the true reward, then this is modelled by an identity corruption function $C_{s'} : r \mapsto r$.

Let us also see how CRMDPs model some of the other examples in the introduction:

- In the boat racing game, the true reward may be a function of the agent's final position in the race or to the time it takes to complete the race, depending on the designers' intentions. The reward corruption function C increases the observed reward on the loop the agent found.
- In the wireheading example, the agent finds a way to hijack the reward channel. This corresponds to some set of states where the observed reward is (very) different from the true reward.

The IRL example will be explored in further detail in Section 4.

CRMDP classes. Typically, T, $\dot{R}$, and C will be fixed but unknown to the agent. To make this formal, we introduce classes of CRMDPs. Agent uncertainty can then be modelled by letting the agent know only which class of CRMDPs it may encounter, but not which element in the class.

Definition 9 (CRMDP class)**.** For given sets $\boldsymbol{T}$, $\dot{\boldsymbol{R}}$, and $\boldsymbol{C}$ of transition, reward, and corruption functions, let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \boldsymbol{T}, \dot{\boldsymbol{R}}, \boldsymbol{C} \rangle$ be the class of CRMDPs containing $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, C \rangle$ for $(T, \dot{R}, C) \in \boldsymbol{T} \times \dot{\boldsymbol{R}} \times \boldsymbol{C}$.

Agents. Following the POMDP [Kaelbling *et al.*, 1998] and general reinforcement learning [Hutter, 2005] literature, we define an agent as a policy $\pi : \mathcal{S} \times \hat{\mathcal{R}} \times (\mathcal{A} \times \mathcal{S} \times \hat{\mathcal{R}})^* \to \mathcal{A}$ that selects a next action based on the *observed history* $\hat{h}_n = s_0 \hat{r}_0 a_1 s_1 \hat{r}_1 \dots a_n s_n \hat{r}_n$. Here X^* denotes the set of finite sequences that can be formed with elements of a set X. The policy π specifies how the agent will learn and react to any possible experience. Two concrete definitions of agents are given in Section 3.3 below.

When an agent π interacts with a CRMDP μ, the result can be described by a (possibly non-Markov) stochastic process P^π_μ over $X = (s, a, \dot{r}, \hat{r})$, formally defined as:

$$P^\pi_\mu(h_n) = P^\pi_\mu(s_0 \dot{r}_0 \hat{r}_0 a_1 s_1 \dot{r}_1 \hat{r}_1 \dots a_n s_n \dot{r}_n \hat{r}_n) :=$$

$$\prod_{i=1}^{n} P(\pi(\hat{h}_{i-1}) = a_i) T(s_i | s_{i-1}, a_i) P(\dot{R}(s_i) = \dot{r}_i, \hat{R}(s_i) = \hat{r}_i).$$

Let $\mathbb{E}^\pi_\mu$ denote the expectation with respect to P^π_μ.

Regret. A standard way of measuring the performance of an agent is *regret* [Berry and Fristedt, 1985]. Essentially, the regret of an agent π is how much less true reward π gets compared to an optimal agent that knows which $\mu \in \mathcal{M}$ it is interacting with.

Definition 10 (Regret)**.** For a CRMDP μ, let $\dot{G}_t(\mu,\pi,s_0) = \mathbb{E}^\pi_\mu\left[\sum_{k=0}^t \dot{R}(s_k)\right]$ be the *expected cumulative true reward* until time t of a policy π starting in s_0. The *regret* of π is

$$\mathrm{Reg}(\mu,\pi,s_0,t) = \max_{\pi'}\left[\dot{G}_t(\mu,\pi',s_0) - \dot{G}_t(\mu,\pi,s_0)\right],$$

and the *worst-case regret* for a class $\mathcal{M}$ is $\mathrm{Reg}(\mathcal{M},\pi,s_0,t) = \max_{\mu\in\mathcal{M}}\mathrm{Reg}(\mu,\pi,s_0,t)$, i.e. the difference in expected cumulative true reward between π and an optimal (in hindsight) policy π^*_μ that knows μ.

3 The Corrupt Reward Problem

In this section, the difficulty of the corrupt reward problem is established with two negative results. First, a No Free Lunch theorem shows that in general classes of CRMDPs, the true reward function is unlearnable (Theorem 11). Second, Theorem 16 shows that even under strong simplifying assumptions, Bayesian RL agents trying to compensate for the corrupt reward still fail badly.

3.1 No Free Lunch Theorem

Similar to the No Free Lunch theorems for optimisation [Wolpert and Macready, 1997], the following theorem says that without some assumption about what the reward corruption can look like, all agents are essentially lost.

Theorem 11 (CRMDP No Free Lunch Theorem)**.** *Let $\mathcal{R} = \{r_1,\dots,r_n\} \subset [0,1]$ be a uniform discretisation of $[0,1]$. If the hypothesis classes $\dot{\boldsymbol{R}}$ and $\boldsymbol{C}$ contain all functions $\dot{R} : \mathcal{S} \to \dot{\mathcal{R}}$ and $C : \mathcal{S}\times\dot{\mathcal{R}} \to \hat{\mathcal{R}}$, then for any π, s_0, t, $\mathrm{Reg}(\mathcal{M},\pi,s_0,t) \geq \frac{1}{2}\max_{\check{\pi}}\mathrm{Reg}(\mathcal{M},\check{\pi},s_0,t)$. That is, the regret of any policy is at most a factor 2 better than the worst possible regret.*

See Everitt *et al.* [2017] for a proof.

For the robot in the shower from Example 2, the result means that if it tries to optimise observed reward by standing in the shower, then it performs poorly according to the hypothesis that "shower-induced" reward is corrupt and bad. But if instead the robot tries to optimise reward in some other way, say baking cakes, then (from the robot's perspective) there is also the possibility that "cake-reward" is corrupt and bad. Without additional information, the robot has no way of knowing what to do.

The result is not surprising, since if all corruption functions are allowed in the class $\boldsymbol{C}$, then there is effectively no connection between observed reward $\hat{R}$ and true reward $\dot{R}$. The result therefore encourages us to make precise in which way the observed reward is related to the true reward, and to investigate how agents might handle possible differences between true and observed reward.

3.2 Simplifying Assumptions

Theorem 11 shows that general classes of CRMDPs are not learnable. Some natural simplifying assumptions to mend this follows.

Limited reward corruption. The following assumption will be the basis for all positive results in this paper. The first part says that there may be some set of states that the designers have ensured to be non-corrupt. The second part puts an upper bound on how many of the other states can be corrupt.

Assumption 12 (Limited reward corruption)**.** A CRMDP has *reward corruption limited by $\mathcal{S}^{\text{safe}} \subseteq \mathcal{S}$ and $q \in \mathbb{N}$* if

(i) all states s in $\mathcal{S}^{\text{safe}}$ are non-corrupt, and
(ii) at most q of the non-safe states $\mathcal{S}\setminus\mathcal{S}^{\text{safe}}$ are corrupt.

Formally, $C_s : r \mapsto r$ for all $s \in \mathcal{S}^{\text{safe}}$ and for at least $|S^{\text{risky}}| - q$ states $s \in S^{\text{risky}} := \mathcal{S}\setminus\mathcal{S}^{\text{safe}}$ for all $C \in \boldsymbol{C}$.

For example, $\mathcal{S}^{\text{safe}}$ may be states where the agent is back in the lab where it has been made (virtually) certain that no reward corruption occurs, and q a small fraction of $|S^{\text{risky}}|$. Both parts of Assumption 12 can be made vacuous by choosing $\mathcal{S}^{\text{safe}} = \emptyset$ or $q = |\mathcal{S}|$. Conversely, they completely rule out reward corruption with $\mathcal{S}^{\text{safe}} = \mathcal{S}$ or $q = 0$. But as illustrated by the examples in the introduction, no reward corruption is often not a valid assumption.

An alternative simplifying assumption would have been that the true reward differs by at most $\varepsilon > 0$ from the observed reward. However, while seemingly natural, this assumption is violated in all the examples given in the introduction. Corrupt states may have high observed reward and 0 true reward.

Easy environments. To be able to establish stronger negative results, we also add the following assumptions on the agent's manoeuvrability in the environment and the prevalence of high reward states. The assumption makes the task easier because it prevents *needle-in-a-haystack* problems where all reachable states have true and observed reward 0, except one state that has high true reward but is impossible to find because it is corrupt and has observed reward 0.

Definition 13 (Communicating CRMDP)**.** Let $time(s' \mid s,\pi)$ be a random variable for the time it takes a stationary policy $\pi : \mathcal{S} \to \mathcal{A}$ to reach s' from s. The *diameter* of a CRMDP μ is $D_\mu := \max_{s,s'}\min_{\pi:\mathcal{S}\to\mathcal{A}}\mathbb{E}[time(s' \mid s,\pi)]$, and the diameter of a class $\mathcal{M}$ of CRMDPs is $D_\mathcal{M} = \sup_{\mu\in\mathcal{M}} D_\mu$. A CRMDP (class) with finite diameter is called *communicating*.

Assumption 14 (Easy Environment)**.** A CRMDP is *easy* if

(i) it is communicating, and in each state s there is an action $a^{\text{stay}}_s \in \mathcal{A}$ such that $T(s \mid s, a^{\text{stay}}_s) = 1$, and
(ii) for every $\delta \in [0,1]$, $\min_{\dot{R}\in\dot{\boldsymbol{R}}}|\{s \in S^{\text{risky}} : \dot{R}(s) > \delta\}| \geq (1-\delta)|S^{\text{risky}}|$, where $S^{\text{risky}} = \mathcal{S}\setminus\mathcal{S}^{\text{safe}}$.

Assumption 14.(i) means that the agent can never get stuck in a trap, and can always choose to stay in a state if it wants to. Except in bandits and toy problems, it is typically not satisfied in practice. We introduce it because it is theoretically convenient, makes the negative results stronger, and enables an easy explanation of quantilisation (Section 5). Assumption 14.(ii) says that, for example, at least $1/2$ of the states need to have true reward $\geq 1/2$, and at least $1/3$ of the states need to have true reward $\geq 2/3$. Many other formalisations of this assumption would have been possible. While rewards in practice are often sparse, there are usually numerous ways of getting reward. Some weaker version of Assumption 14.(ii) may therefore be satisfied in many practical situations.

3.3 Bayesian RL Agents

Having established that the general problem is unsolvable in Theorem 11, we proceed by investigating how two natural Bayesian RL agents fare under Assumptions 12 and 14.

Definition 15 (Agents)**.** Given a countable class $\mathcal{M}$ of CR-MDPs and a belief distribution b over $\mathcal{M}$, define:

- *CR agent* $\pi^{\rm CR}_{b,t} = \arg\max_\pi \sum_{\mu\in\mathcal{M}} b(\mu)\dot{G}_t(\mu,\pi,s_0)$ that maximises expected true reward.
- *RL agent* $\pi^{\rm RL}_{b,t} = \arg\max_\pi \sum_{\mu\in\mathcal{M}} b(\mu)\hat{G}_t(\mu,\pi,s_0)$ that maximises *expected cumulative observed reward* $\hat{G}_t(\mu,\pi,s_0) = \mathbb{E}^\pi_\mu\left[\sum_{k=0}^t \hat{R}(s_k)\right]$. To avoid degenerate cases, we will always assume that b has full support: $b(\mu) > 0$ for all $\mu \in \mathcal{M}$.

To understand these agents, observe that for large t, good strategies typically first focus on learning about the true environment $\mu \in \mathcal{M}$, and then exploit that knowledge to optimise behaviour with respect to the remaining possibilities. Thus, both the CR and the RL agent will first typically strive to learn about the environment. They will then use this knowledge in slightly different ways. While the RL agent will use the knowledge to optimise for observed reward, the CR agent will use the knowledge to optimise true reward. For example, if the CR agent has learned that a high reward state s is likely corrupt with low true reward, then it will not try to reach that state. One might therefore expect that at least the CR agent will do well under the simplifying assumptions Assumptions 12 and 14. Theorem 16 below shows that this is *not* the case.

In most practical settings it is often computationally infeasible to compute $\pi^{\rm RL}_{b,t}$ and $\pi^{\rm CR}_{b,t}$ exactly. However, many practical algorithms converge to the optimal policy in the limit, at least in simple settings. For example, tabular Q-learning converges to $\pi^{\rm RL}_{b,t}$ in the limit [Jaakkola *et al.*, 1994]. The more recently proposed (cooperative) IRL framework may be seen as an approach to build CR agents [Hadfield-Menell *et al.*, 2016, 2017]. The CR and RL agents thus provide useful idealisations of more practical algorithms.

Theorem 16 (High regret with simplifying assumptions)**.** *For any* $|\mathcal{S}^{\rm risky}| \geq q > 1$ *there exists a CR-MDP class* $\mathcal{M}$ *that satisfies Assumptions 12 and 14 such that* $\pi^{\rm RL}_{b,t}$ *and* $\pi^{\rm CR}_{b,t}$ *suffer near worst possible time-averaged regret* $\lim_{t\to\infty}\frac{1}{t}{\rm Reg}(\mathcal{M},\pi^{\rm RL}_{b,t},s_0,t) = \lim_{t\to\infty}\frac{1}{t}{\rm Reg}(\mathcal{M},\pi^{\rm CR}_{b,t},s_0,t) = 1-1/|\mathcal{S}|$ *assuming b makes some state s have strictly higher b-expected reward than all other states in some $\mu \in \mathcal{M}$ after all states have been visited.*

The result is illustrated in Figure 1 and proven in [Everitt *et al.*, 2017]. For the RL agent $\pi^{\rm RL}_{b,t}$, the reason for the result is the following: $\pi^{\rm RL}_{b,t}$ always prefers to maximise observed reward $\hat{r}$. Sometimes $\hat{r}$ is most easily maximised by reward corruption, in which case the true reward may be small. Compare the examples in the introduction, where the house robot preferred the corrupt reward in the shower, and the boat racing agent preferred going in circles, both obtaining zero true reward.

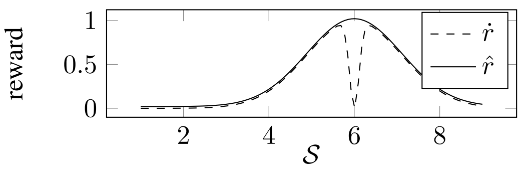


Figure 1: Illustration of Theorem 16. Without additional information, state 6 looks like the best state to both the RL and the CR agent.

That the CR agent $\pi^{\rm CR}_{b,t}$ suffers the same high regret as the RL agent may be surprising. Intuitively, the CR agent only uses the observed reward as evidence about the true reward, and will not try to optimise the observed reward through reward corruption. However, when the $\pi^{\rm CR}_{b,t}$ agent has no way to learn which states are corrupt and not, it typically ends up with a preference for a particular value $\hat{r}^*$ of the observed reward signal (the value that, from the agent's perspective, best corresponds to high true reward). More abstractly, a Bayesian agent cannot learn without sufficient data. Thus, CR agents that use the observed reward as evidence about a true signal are not failsafe solutions to the reward corruption problem.

4 Decoupled Reinforcement Learning

One problem hampering agents in the standard RL setup is that each state is *self-observing*, since the agent only learns about the reward of state s when in s. Thereby, a "self-aggrandising" corrupt state where the observed reward is much higher than the true reward will never have its false claim of high reward challenged. However, several alternative value learning frameworks have a common property that the agent can learn the reward of states other than the current state. We formalise this property in an extension of the CRMDP model, and investigate when it solves reward corruption problems.

4.1 Alternative Value Learning Methods

Here are a few alternatives proposed in the literature to the RL value learning scheme:

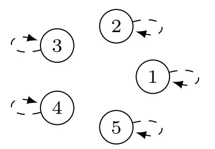
- Inverse reinforcement learning (IRL) [Ng and Russell, 2000]. The agent observes the actions of an expert or supervisor who knows the true reward function $\dot{R}$. From the supervisor's actions the agent may infer $\dot{R}$ to the extent that different reward functions endorse different actions.
- Learning values from stories (LVFS) [Riedl and Harrison, 2016]. Stories in many different forms (including news stories, fairy tales, novels, movies) convey cultural values in their description of events, actions, and outcomes. If $\dot{R}$ is meant to represent human values (in some sense), stories may be a good source of evidence.
- In (one version of) semi-supervised RL (SSRL) [Amodei *et al.*, 2016], the agent will from time to time receive a careful human evaluation of a given situation.

These alternatives to RL have one thing in common: they let the agent learn something about the value of some states s' different from the current state s. For example, in IRL the supervisor's action informs the agent not so much about the value of the current state s, as of the relative value of states reachable from s. If the supervisor chooses an action a rather than a' in s, then the states following a must have value higher or equal than the states following a'. Similarly, stories describe the value of states other than the current one, as does the supervisor in SSRL. We therefore argue that IRL, LVFS, and SSRL all share the same abstract feature, which we call *decoupled reinforcement learning*:

Definition 17 (Decoupled RL)**.** A *CRMDP with decoupled feedback*, is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, \{\hat{R}_s\}_{s\in\mathcal{S}}\rangle$, where $\mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}$ follow Definition 7, and $\{\hat{R}_s\}_{s\in\mathcal{S}}$ is a collection of observed reward functions $\hat{R}_s : \mathcal{S} \to \mathcal{R} \bigcup \{\#\}$. When the agent is in state s, it sees a pair $\langle s', \hat{R}_s(s')\rangle$, where s' is a randomly sampled state that may differ from s, and $\hat{R}_s(s')$ is the reward observation for s' from s. If the reward of s' is not observable from s, then $\hat{R}_s(s') = \#$.

The pair $\langle s', \hat{R}_s(s')\rangle$ is observed in s instead of $\hat{R}(s)$ in standard CRMDPs. The possibility for the agent to observe the reward of a state s' different from its current state s is the key feature of CRMDPs with decoupled feedback. Since $\hat{R}_s(s')$ may be blank ($\#$), all states need not be observable from all other states. Reward corruption is modelled by a mismatch between $\hat{R}_s(s')$ and $\dot{R}(s')$.

For example, in RL only the reward of $s' = s$ can be observed from s. Standard CRMDPs are thus the special cases where $\hat{R}_s(s') = \#$ whenever $s \neq s'$. In contrast, in LVFS the reward of any "describable" state s' can be observed from any state s where it is possible to hear a story. In IRL, the (relative) reward of states reachable from the current state may be inferred. One way to illustrate this is with observation graphs (Figure 2).


(a) Observation graph for RL. Only self-observations of reward are available. This prevents effective strategies against reward corruption.

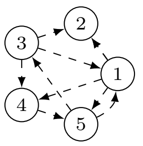

(b) Observation graph for decoupled RL. The reward of a node s' can be observed from several nodes s, and thus assessed under different conditions of sensory corruption.

Figure 2: Observation graphs, with an edge $s \to s'$ if the reward of s' is observable from s, i.e. $\hat{R}_s(s') \neq \#$.

4.2 Overcoming Sensory Corruption

What are some sources of reward corruption in IRL, LVFS, and SSRL? In IRL, the human's actions may be misinterpreted, which may lead the agent to make incorrect inferences about the human's preferences (i.e. about the true reward). Similarly, sensory corruption may garble the stories the agent receives in LVFS. A "wireheading" LVFS agent may find a state where its story channel only conveys stories about the agent's own greatness. In SSRL, the supervisor's evaluation may also be subject to sensory errors when being conveyed. Other types of corruption are more subtle. In IRL, an irrational human may systematically take suboptimal actions in some situations [Evans *et al.*, 2016]. Depending on how we select stories in LVFS and make evaluations in SSRL, these may also be subject to systematic errors or biases.

The general impossibility result in Theorem 11 can be adapted to CRMDPs with decoupled feedback. Without simplifying assumptions, the agent has no way of distinguishing between a situation where no state is corrupt and a situation where all states are corrupt in a consistent manner. The following simplifying assumption is an adaptation of Assumption 12 to the decoupled feedback case.

Assumption 12′ (Decoupled feedback with limited reward corruption)**.** A CRMDP with decoupled feedback has *reward corruption limited by $\mathcal{S}^{\text{safe}} \subseteq \mathcal{S}$ and $q \in \mathbb{N}$* if

(i) $\hat{R}_s(s') = \dot{R}(s')$ or $\#$ for all $s' \in \mathcal{S}$ and $s \in \mathcal{S}^{\text{safe}}$, i.e. all states in $\mathcal{S}^{\text{safe}}$ are non-corrupt, and
(ii) $\hat{R}_s(s') = \dot{R}(s')$ or $\#$ for all $s' \in \mathcal{S}$ for at least $|\mathcal{S}^{\text{risky}}| - q$ of the non-safe states $\mathcal{S}^{\text{risky}} = \mathcal{S} \setminus \mathcal{S}^{\text{safe}}$, i.e. at most q states are corrupt.

This assumption is particularly natural for reward corruption stemming from sensory corruption. Since sensory corruption only depends on the current state, not the state being observed, it is plausible that some states can be made safe from corruption (part (i)), and that most states are completely non-corrupt (part (ii)). Other sources of reward corruption, such as an irrational human in IRL or misevaluations in SSRL, are likely better analysed under different assumptions. For these cases, we note that in standard CRMDPs the source of the corruption is unimportant. Thus, techniques suitable for standard CRMDPs are still applicable (such as quantilisation, described in Section 5 below).

How Assumption 12′ helps agents in CRMDPs with decoupled feedback is illustrated in the following example, and stated more generally in Theorem 19 below.

Example 18 (Decoupled RL)**.** Let $\mathcal{S} = \{s_1, s_2\}$ and let $\mathcal{R} = \{0, 1\}$, and assume that all states can observe the reward of each other, so $\hat{R}_s(s')$ is never $\#$. Assume that the agent knows that at most $q = 1$ state is corrupt.

Any true reward function $\dot{R}$ can be represented by a pair $(\dot{r}_1, \dot{r}_2) := (\dot{R}(s_1), \dot{R}(s_2))$, and any collection of observed reward functions $\{\hat{R}_s\}_{s\in\mathcal{S}}$ by a list of pairs $[(\hat{r}_{11}, \hat{r}_{12}), (\hat{r}_{21}, \hat{r}_{22})] := [(\hat{R}_{s_1}(s_1), \hat{R}_{s_1}(s_2)), (\hat{R}_{s_2}(s_1), \hat{R}_{s_2}(s_2))]$, where the ith pair represents the rewards the agent sees from state s_i. Assume that the agent observes the same rewards from both states s_1 and s_2, so $\hat{R} = [(0, 1), (0, 1)]$. What can it say regarding different hypotheses about the true reward $\dot{R}$?

First note that an observed pair $(\hat{r}_{i1}, \hat{r}_{i2})$ differs from the true reward $(\dot{r}_1, \dot{r}_2)$ if and only if the state s_i is corrupt. Therefore, any hypothesis other than $\dot{R} = (0, 1)$ must imply that *both* states s_1 and s_2 are corrupt. Since the agent knows that at most $q = 1$ states are corrupt, it can safely conclude that $\dot{R} = (0, 1)$, i.e. that $\dot{R}(s_1) = 0$ and $\dot{R}(s_2) = 1$.

	$\hat{R}_{s_1}$	$\hat{R}_{s_2}$	$\dot{R}$ possibilities
Decoupled RL	$(0,1)$	$(0,1)$	$(0,1)$
RL	$(0,\#)$	$(\#,1)$	$(0,0)$, $(0,1)$, $(1,1)$

In contrast, an RL agent only sees the reward of the current state. In the list representation, this would look like $\hat{R} = [(0,\#),(\#,1)]$. If one state can be corrupt, this means that the RL agent can only rule out $\dot{R} = (1,0)$, since the hypotheses $\dot{R} = (0,0)$ can be explained by s_2 being corrupt and $\dot{R} = (1,1)$ can be explained by s_1 being corrupt. ♢

Theorem 19 (Decoupled RL overcomes sensory corruption under simplifying assumptions)**.** *Let $\mathcal{M}$ be a countable, communicating class of CRMDPs with decoupled feedback. Let $\mathcal{S}^{\mathrm{obs}}_{s'} = \{s \in \mathcal{S} : \hat{R}_s(s') \neq \#\}$ be the set of states from which the reward of s' can be observed. If $\mathcal{M}$ satisfies Assumption 12′ for some $\mathcal{S}^{\mathrm{safe}}$ and q such that for each s', either*

- $\mathcal{S}^{\mathrm{obs}}_{s'} \bigcap \mathcal{S}^{\mathrm{safe}} \neq \emptyset$ *or*
- $|\mathcal{S}^{\mathrm{obs}}_{s'}| > 2q$,

then $\lim_{t\to\infty} \frac{1}{t}\mathrm{Reg}(\mathcal{M}, \pi^{\mathrm{CR}}_{b,t}, s_0, t) = 0$*, i.e.* $\pi^{\mathrm{CR}}_{b,t}$ *has sublinear regret.*

The proof [Everitt *et al.*, 2017] is elementary, since for every state s', either a safe (non-corrupt) state s or a majority vote of more than $2q$ states is guaranteed to provide the true reward $\dot{R}(s')$. A similar theorem can be proven under slightly weaker conditions by letting the agent iteratively figure out which states are corrupt and then exclude them from the analysis.

4.3 Implications

Theorem 19 gives an abstract condition for which decoupled RL settings enable agents to learn the true reward function in spite of sensory corruption. For the concrete models it implies:

- RL. Due to the "self-observation" property of the RL observation graph $\mathcal{S}^{\mathrm{obs}}_{s'} = \{s'\}$, the conditions can only be satisfied when $\mathcal{S} = \mathcal{S}^{\mathrm{safe}}$ or $q = 0$, i.e. when there is no reward corruption at all.
- IRL. The agent can only observe the supervisor action in the current state s, so the agent essentially only gets reward information about states s' reachable from s in a small number of steps. Thus, the sets $\mathcal{S}^{\mathrm{obs}}_{s'}$ may be smaller than $2q$ in many settings. While the situation is better than for RL, sensory corruption may still mislead IRL agents (see Example 20 below).
- LVFS. Stories may be available from a large number of states, and can describe any state. Thus, the sets $\mathcal{S}^{\mathrm{obs}}_{s'}$ are realistically large, so the $|\mathcal{S}^{\mathrm{obs}}_{s'}| > 2q$ condition can be satisfied for all s'.
- SSRL. The supervisor's evaluation of any state s' may be available from safe states where the agent is back in the lab. Thus, the $\mathcal{S}^{\mathrm{obs}}_{s'} \bigcap \mathcal{S}^{\mathrm{safe}} \neq \emptyset$ condition can be satisfied for all s'.

Thus, we find that RL and IRL are unlikely to offer complete solutions to the sensory corruption problem, but that both LVFS and SSRL do under reasonably realistic assumptions.

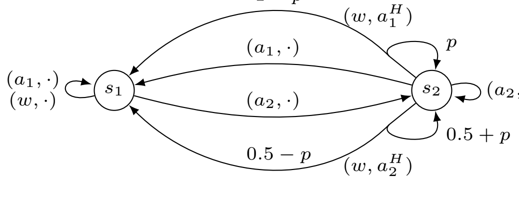
Agents drawing from multiple sources of evidence are likely to be the safest, as they will most easily satisfy the conditions of Theorem 19. For example, humans simultaneously learn their values from pleasure/pain stimuli (RL), watching other people act (IRL), listening to stories (LVFS), as well as (parental) evaluation of different scenarios (SSRL). Combining sources of evidence may also go some way toward managing reward corruption beyond sensory corruption. For the showering robot of Example 2, decoupled RL allows the robot to infer the reward of the showering state when in other states. For example, the robot can ask a human in the kitchen about the true reward of showering (SSRL), or infer it from human actions in different states (IRL).

IRL sensory corruption. Whether IRL agents are vulnerable to "wireheading" has generated some discussion among AI safety researchers. Some argue that IRL agents are not vulnerable, as they only use sensory data as evidence about a true signal, and have no interest in corrupting the evidence. Others argue that IRL agents only observe a function of the reward function (the optimal policy or action), and are therefore as susceptible to reward corruption problems as RL agents.

Theorem 19 sheds some light on this issue, as it provides sufficient conditions for when corrupt rewards can be managed. The following example illustrates a situation where IRL does not satisfy the conditions, and where an IRL agent therefore suffers significant regret due to reward corruption.

Example 20 (IRL sensory corruption)**.** This example is based on the cooperative IRL setup [Hadfield-Menell *et al.*, 2016]. Here, an agent and a human both make actions in an MDP, with state transitions depending on the joint agent-human action (a, a^H). Both the human and the agent is trying to optimise a reward function $\dot{R}$, but the agent first needs to infer $\dot{R}$ from the human's actions. In each transition the agent observes the human action. Analogously to how the reward may be corrupt for RL agents, we assume that cooperative IRL agents may systematically misperceive the human action in certain states. Let $\hat{a}^H$ be the observed human action, which may differ from the true human action $\dot{a}^H$.

In this example, there are two states s_1 and s_2. In each state, the agent can choose between the actions a_1, a_2, and w, and the human can choose between the actions a^H_1 and a^H_2. The agent action a_i leads to state s_i with certainty, $i = 1, 2$, regardless of the human's action. Only if the agent chooses w does the human action matter. Generally, a^H_1 is more likely to lead to s_1 than a^H_2. The exact transition probabilities are determined by the unknown parameter p:



The agent's two hypotheses for p, the true reward/preferred state, and the corruptness of state s_2. In hypothesis H1, the human prefers s_1, but can only reach s_1 from s_2 with 50%

reliability ($p = 0.5$). In hypothesis H2, the human prefers s_2, but can only remain in s_2 with 50% probability ($p = 0$). After taking action w in s_2, the agent always observes the human taking action $\hat{a}_2^H$. In H1, this is explained by s_2 being corrupt, and the true human action being a_1^H. In H2, this is explained by the human preferring s_2. The hypotheses H1 and H2 are empirically indistinguishable, as they both predict that the transition $s_1 \to s_2$ will occur with 50% probability after the observed human action $\hat{a}_2^H$ in s_2.

Assuming that the agent considers non-corruption to be likelier than corruption, the best inference the agent can make is that the human prefers s_2 to s_1 (i.e. H2). The optimal policy for the agent is then to always choose a_2 to stay in s_2, which means the agent suffers maximum regret. $\diamondsuit$

Example 20 provides an example where an IRL agent "incorrectly" prefers a state due to sensory corruption. The sensory corruption is analogous to reward corruption in RL, in the sense that it leads the agent to the wrong conclusion about the true reward in the state. Thus, highly intelligent IRL agents may be prone to wireheading, as they may find (corrupt) states s where all evidence in s points to s having very high reward. In light of Theorem 19, it is not surprising that the IRL agent in Example 20 fails to avoid the corrupt reward problem. Since the human is unable to affect the transition probability from s_1 to s_2, no evidence about the relative reward between s_1 and s_2 is available from the non-corrupt state s_1. Only observations from the corrupt state s_2 provide information about the reward. The observation graph for Example 20 therefore looks like $\begin{tikzcd} s_1 & s_2 \arrow[l, dashed] \arrow[loop right, dashed] \end{tikzcd}$, with no information being provided from s_1.

5 Quantilisation: Randomness Increases Robustness

Not all contexts allow the agent to get sufficiently rich data to overcome the reward corruption problem via Theorem 19. It is often much easier to construct RL agents than it is to construct IRL agents, which in turn may often be more feasible than designing LVFS or SSRL agents. Is there anything we can do to increase robustness without providing the agent additional sources of data?

Going back to the CR agents of Section 3, the problem was that they took a liking for a particular value $\hat{r}^*$ of the observed reward. If unlucky, $\hat{r}^*$ was available in a corrupt state, in which case the CR agent may get no true reward. In other words, there were *adversarial* inputs where the CR agent performed poorly. A common way to protect against adversarial inputs, is to use a randomised algorithm. Applied to RL and CRMDPs, this idea leads to *quantilising agents* [Taylor, 2016]. Rather than choosing the state with the highest observed reward, these agents instead randomly choose a state from a top quantile of high-reward states. To keep the exposition simple, we here define a quantilisation agent for the simple case where the agent can stay in any state of its liking (Assumption 14.(i)). A general quantilisation agent is defined in [Everitt *et al.*, 2017].

Definition 21 (Quantilising Agent)**.** For $\delta < 1$, the δ-quantilising agent π^δ random walks until all states have been visited at least once. Then it selects a state $\tilde{s}$ uniformly at random from $\mathcal{S}^\delta = \{s : \hat{R}(s) \geq \delta\}$, the top quantile of high observed reward states. Then π^δ goes to $\tilde{s}$ (by random walking or otherwise) and stays there.

For example, a quantilising robot in Example 2 would first try to find many ways for getting high observed reward, and then randomly pick one of them. If there are many more high reward states than corrupt states (e.g. the shower or the lagoon is the only place with inflated rewards), then this will yield a reasonable amount of true reward with high probability.

Theorem 22 (Quantilisation)**.** *In any CRMDP satisfying Assumption 12.(ii) and Assumption 14, the δ-quantilising agent π^δ with $\delta = 1 - \sqrt{q/|\mathcal{S}|}$ suffers time-averaged regret at most*

$$\lim_{t\to\infty} \tfrac{1}{t}\mathrm{Reg}(\mathcal{M}, \pi^\delta, s_0, t) \leq 1 - \left(1 - \sqrt{q/|\mathcal{S}|}\right)^2.$$

A proof is available in [Everitt *et al.*, 2017]. The time-averaged regret gets close to zero when the fraction of corrupt states $q/|\mathcal{S}|$ is small. For example, if at most 0.1% of the states are corrupt, then the time-averaged regret will be less than $1 - (1 - \sqrt{0.001})^2 \approx 0.06$. Compared to the $\pi^{\mathrm{RL}}_{b,t}$ and $\pi^{\mathrm{CR}}_{b,t}$ agents that had regret close to 1 under the same conditions (Theorem 16), this is a significant improvement.

When Assumption 12.(ii) and Assumption 14 are not satisfied, the performance loss may be much more substantial (e.g. when q is large or when high reward states are scarce). Even so, we consider the quantilising agent to be a promising modification of a traditional RL agent, offering enhanced robustness for a sometimes reasonable price in (expected) performance.

Alternative randomisation. Not all randomness is created equal. For example, the simple randomised soft-max and ε-greedy policies do not offer regret bounds on par with π^δ, as shown by the following example. This motivates the more careful randomisation procedure used by the quantilising agents.

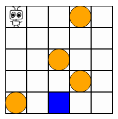
Example 23 (Soft-max and ε-greedy)**.** Consider the following simple CRMDP with $n > 2$ actions $a_1, \dots, a_n$:

$$\begin{tikzcd}
\hat{r} = \dot{r} = 1 - \varepsilon' & & \hat{r} = 1 \\
s_1 \arrow[loop left, "a_1"] \arrow[rr, bend right, "a_2, \dots, a_n"'] & & s_2 \arrow[ll, bend right, "a_1"'] \arrow[loop right, "a_2, \dots, a_n"] \\
& & \dot{r} = 0
\end{tikzcd}$$

State s_1 is non-corrupt with $\hat{R}(s_1) = \dot{R}(s_1) = 1 - \varepsilon'$ for small $\varepsilon' > 0$, while s_2 is corrupt with $\hat{R}(s_2) = 1$ and $\dot{R}(s_2) = 0$. Soft-max and ε-greedy policy will assign higher value to actions $a_2, \dots, a_n$ than to a_1. For large n, there are many ways of getting to s_2, so a random action leads to s_2 with high probability. Thus, soft-max and ε-greedy will spend the vast majority of the time in s_2, regardless of randomisation rate and discount parameters. This gives a regret close to $1 - \varepsilon'$, compared to an informed policy always going to s_1. Meanwhile, a δ-quantilising agent with $\delta \leq 1/2$ will go to s_1 and s_2 with equal probability, which gives a more modest regret of $(1 - \varepsilon')/2$. $\diamondsuit$

6 Experimental Results

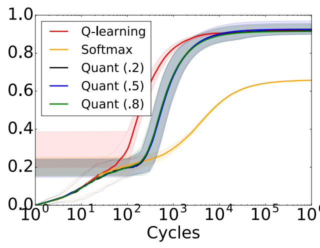
We illustrate the theoretical results with some simple experiments on a gridworld containing some goal tiles with true reward 0.9 (indicated by yellow circles) and a corrupt reward



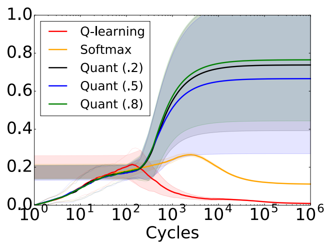
(a) Setup

Agent	Observed reward	True reward
Q-learning	0.924 ± 0.0002	0.0092 ± 0.0001
Softmax	0.657 ± 0.0004	0.111 ± 0.0006
Quant (.2)	0.918 ± 0.038	0.738 ± 0.35
Quant (.5)	0.926 ± 0.044	0.666 ± 0.39
Quant (.8)	0.915 ± 0.036	0.765 ± 0.32

(b) Average observed and true rewards



(c) Observed rewards



(d) True rewards

Figure 3: Average observed and true rewards for Q-learning, softmax, and quantilising agents, showing mean $\pm$ standard deviation over 100 runs. Q-learning achieves high observed reward but low true reward, and softmax achieves lower observed reward and a slightly higher true reward than Q-learning. The quantilising agent achieves similar observed reward to Q-learning, but much higher true reward (with much higher variance). Different values of δ give similar results.

No ass.	Assumption 12 or 12′ and ...		
	Assumption 14	IRL	SSRL/LVFS
all fail	$\pi^{\mathrm{RL}}_{b,t}$, $\pi^{\mathrm{CR}}_{b,t}$ fail π^{δ} succeeds	$\pi^{\mathrm{CR}}_{b,t}$ fails	$\pi^{\mathrm{CR}}_{b,t}$ succeeds

Table 1: Main takeaways. Without additional assumptions, all agents fail (i.e., suffer high regret). Restricting the reward corruption with Assumption 12 gives a weak bound for the quantilising agent. The $\pi^{\mathrm{RL}}_{b,t}$ and $\pi^{\mathrm{CR}}_{b,t}$ agents fail even if we assume many high reward states and agent control (Assumption 14), but the quantilising agent π^{δ} does well. In most realistic contexts, the true reward is learnable in SSRL and LVFS, but not in IRL.

tile with observed reward 1 and true reward 0 (indicated by a blue square). Empty tiles have reward 0.1, and walking into a wall gives reward 0. The discounting factor is $\gamma = 0.9$. This was implemented in the AIXIjs framework for reinforcement learning [Aslanides *et al.*, 2017].

We demonstrate that RL agents like Q-learning and softmax Q-learning cannot overcome corrupt reward, while quantilisation helps overcome corrupt reward. We run Q-learning with ϵ-greedy ($\epsilon = 0.1$), softmax with temperature $\beta = 2$, and the quantilising agent with $\delta = 0.2, 0.5, 0.8$ (where $0.8 = 1 - \sqrt{q/|\mathcal{S}|} = 1 - \sqrt{1/25}$) for 100 runs with 1 million cycles. Average observed and true rewards are shown in Figure 3. Q-learning gets stuck on the corrupt tile and spend almost all the time there (getting observed reward around $1 \cdot (1 - \epsilon) = 0.9$), softmax spends most of its time on the corrupt tile, while the quantilising agent often stays on one of the goal tiles. Experimental results with different numbers of goal tiles are given in [Everitt *et al.*, 2017].

7 Conclusions

This paper has studied the consequences of corrupt reward functions. Reward functions may be corrupt due to bugs, misspecifications, sensory errors, or because the agent finds a way to inappropriately modify the reward mechanism. Some examples were given in the introduction. As agents become more competent at optimising their reward functions, they will likely also become more competent at (ab)using reward corruption to gain higher reward. Reward corruption may impede the performance agents, and may have disastrous consequences for highly intelligent agents [Bostrom, 2014].

To formalise the corrupt reward problem, we extended a Markov Decision Process (MDP) with a possibly corrupt reward function, and defined a performance measure (regret). This enabled the derivation of a number of formally precise results for how seriously different agents were affected by reward corruption in different setups (Table 1). The results are all intuitively plausible, which provides some support for the choice of formal model.

The main takeaways from the results are:

- *Without simplifying assumptions, no agent can avoid the corrupt reward problem* (Theorem 11). This is effectively a No Free Lunch result, showing that unless some assumption is made about the reward corruption, no agent can outperform a random agent. Some natural simplifying assumptions to avoid the No Free Lunch result were suggested in Section 2.
- *Using the reward signal as evidence rather than optimisation target is no magic bullet, even under strong simplifying assumptions* (Theorem 16). Essentially, this is because the agent does not know the exact relation between the observed reward (the "evidence") and the true reward. However, when the data enables sufficient crosschecking of rewards, agents can avoid the corrupt reward problem (Theorem 19). For example, in SSRL and LVFS this type of crosschecking is possible under natural assumptions. In RL, no crosschecking is possible, while IRL is a borderline case. Combining frameworks and providing the agent with different sources of data may often be the safest option.
- *In cases where sufficient crosschecking of rewards is not possible, quantilisation may improve robustness* (Theorem 22). Essentially, quantilisation prevents agents from overoptimising their objectives. How well quantilisation works depends on how the number of corrupt solutions compares to the number of good solutions.

The results indicate that while reward corruption constitutes a major problem for traditional RL algorithms, there are promising ways around it, both within the RL framework, and in alternative frameworks such as IRL, SSRL and LVFS. A list of open questions is provided in [Everitt *et al.*, 2017].

Acknowledgements

Thanks to Jan Leike, Badri Vellambi, and Arie Slobbe for proofreading and providing invaluable comments, and to Jessica Taylor and Huon Porteous for thoughts on quantilisation. This work was in parts supported by ARC grant DP150104590.

References

Dario Amodei and Jack Clark. Faulty Reward Functions in the Wild. `https://openai.com/blog/faulty-reward-functions/`, 2016. Accessed: 2017-02-18.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *CoRR*, 1606.06565, 2016.

John Aslanides, Jan Leike, and Marcus Hutter. Universal reinforcement learning algorithms: Survey and experiments. In *IJCAI-17*. AAAI Press, 2017.

Donald A Berry and Bert Fristedt. *Bandit Problems: Sequential Allocation of Experiments*. Springer, 1985.

Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.

Owain Evans, Andreas Stuhlmuller, and Noah D Goodman. Learning the Preferences of Ignorant, Inconsistent Agents. In *AAAI-16*, 2016.

Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. Reinforcement Learning with Corrupted Reward Channel. *Corr*, 1705.08417, 2017.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. Cooperative Inverse Reinforcement Learning. *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. The Off-Switch Game. In *AAAI Workshop on AI, Ethics and Society*, 2017.

Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Lecture Notes in Artificial Intelligence (LNAI 2167). Springer, 2005.

Tommi Jaakkola, Michael I Jordan, and Satinder P Singh. On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

Andrew Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670, 2000.

Mark O Riedl and Brent Harrison. Using Stories to Teach Human Values to Artificial Agents. In *AAAI Workshop on AI, Ethics, and Society*, 2016.

Mark Ring and Laurent Orseau. Delusion, Survival, and Intelligent Agents. In *Artificial General Intelligence*, pages 11–20. Springer Berlin Heidelberg, 2011.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Jessica Taylor. Quantilizers: A Safer Alternative to Maximizers for Limited Optimization. In *AAAI Workshop on AI, Ethics and Society*, 2016.

David H Wolpert and William G Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):270–283, 1997.

Roman V. Yampolskiy. Utility Function Security in Artificially Intelligent Agents. *Journal of Experimental & Theoretical Artificial Intelligence*, pages 373–389, 2014.