Pitfalls of Learning a Reward Function Online

Stuart Armstrong[1,2*], **Jan Leike**[3], **Laurent Orseau**[3] and **Shane Legg**[3]
[1]Future of Humanity Institute, Oxford University, UK
[2]Machine Intelligence Research Institute, Berkeley, USA
[3]DeepMind, London, UK
stuart.armstrong@philosophy.ox.ac.uk, leike@google.com, lorseau@google.com, legg@google.com

Abstract

In some agent designs like inverse reinforcement learning an agent needs to learn its own reward function. Learning the reward function and optimising for it are typically two different processes, usually performed at different stages. We consider a continual ("one life") learning approach where the agent both learns the reward function and optimises for it at the same time. We show that this comes with a number of pitfalls, such as deliberately manipulating the learning process in one direction, refusing to learn, "learning" facts already known to the agent, and making decisions that are strictly dominated (for all relevant reward functions). We formally introduce two desirable properties: the first is 'unriggability', which prevents the agent from steering the learning process in the direction of a reward function that is easier to optimise. The second is 'uninfluenceability', whereby the reward-function learning process operates by learning facts about the environment. We show that an uninfluenceable process is automatically unriggable, and if the set of possible environments is sufficiently large, the converse is true too.

1 Introduction

In reinforcement learning (RL) an agent has to learn to solve the problem by maximising the expected reward provided by a reward function [Sutton and Barto, 1998]. *Designing* such a reward function is similar to designing a scoring function for a game, and can be very difficult ([Lee *et al.*, 2017])[1]. Usually, one starts by designing a proxy: a simple reward function that seems broadly aligned with the user's goals. While testing the agent with this proxy, the user may observe that the agent finds a simple behaviour that obtains a high reward on the proxy, but does not match the behaviour intended by the user,[2] who must then refine the reward function to include more complicated requirements, and so on.

This has led to a recent trend to *learn* a model of the reward function, rather than having the programmer design it ([Ng and Russell, 2000; Hadfield-Menell *et al.*, 2016; Choi and Kim, 2011; Amin and Singh, 2016; Abbeel and Ng, 2004; Christiano *et al.*, 2017; Hadfield-Menell *et al.*, 2017; Ibarz *et al.*, 2018; Akrour *et al.*, 2012; MacGlashan *et al.*, 2017; Pilarski *et al.*, 2011]). One particularly powerful approach is putting the human into the loop ([Abel *et al.*, 2017]) as done by [Christiano *et al.*, 2017], because it allows for the opportunity to correct misspecified reward functions as the RL agent discovers exploits that lead to higher reward than intended.

However, learning the reward function with a human in the loop has one problem: by manipulating the human, the agent could manipulate the learning process.[3] If the learning process is online – the agent is maximising its reward function as well as learning it – then the human's feedback is now an optimisation *target*. [Everitt and Hutter, 2016; Everitt and Hutter, 2019] and [Everitt, 2018] analyse the problems that can emerge in these situations, phrasing it as a 'feedback tampering problem'. Indeed, a small change to the environment can make a reward-function learning process manipulable.[4] So it is important to analyse which learning processes are prone to manipulation.

After building a theoretical framework for studying the dynamics of learning reward functions online, this paper will identify the crucial property of a learning process being *uninfluenceable*: in that situation, the reward function depends only on the environment, and is outside the agent's control. Thus it is completely impossible to manipulate an uninfluenceable learning process, and the reward-function learning is akin to Bayesian updating.

The paper also identifies the weaker property of unrigga-

*Contact Author

[1]And see "Specification gaming: the flip side of AI ingenuity", https://deepmind.com/blog/article/Specification-gaming-the-flip-side-of-AI-ingenuity

[2]See https://blog.openai.com/faulty-rewardfunctions/ and the paper [Leike *et al.*, 2017].

[3]Humans have many biases and inconsistencies that may be exploited ([Kahneman, 2011]), even accidentally; and humans can be tricked and fooled, skills that could be in the interest of such an agent to develop, especially as it gets more powerful ([Bostrom, 2014; Yudkowsky, 2008]).

[4]See this example, where an algorithm is motivated to give a secret password to a user while nominally asking for it, since it is indirectly rewarded for correct input of the password: https://www.lesswrong.com/posts/b8HauRWrjBdnKEwM5/rigging-is-a-form-of-wireheading

bility, an algebraic condition that ensures that actions taken by the agent do not influence the learning process in expectation. An unriggable learning process is thus one the agent cannot 'push' towards its preferred reward function.

An uninfluenceable learning process is automatically unriggable, but the converse need not be true. This paper demonstrates that, if the set of environments is large enough, an unriggable learning process is equivalent, in expectation, to an uninfluenceable one. If this condition is not met, unriggable-but-influenceable learning processes do allow some undesirable forms of manipulations. The situation is even worse if the learning process is riggable: the agent can follow a policy that would reduce its reward, with certainty, for *all* the reward functions it is learning about, among other pathologies.

To illustrate, this paper uses a running example of a child asking their parents for career advice (see Section 2.2). That learning process can be riggable, unriggable-but-influenceable, or uninfluenceable, and Q-learning examples based on it will be presented in Section 6.

This paper also presents a 'counterfactual' method for making any learning process uninfluenceable, and shows some experiments to illustrate its performance compared with influenceable and riggable learning.

2 Notation and Formalism

The agent takes a series of actions (from the finite set $\mathcal{A}$) and receives from the environment a series of observations (from the finite set $\mathcal{O}$). A sequence of m actions and observations forms a history of length m: $h_m = a_1o_1a_2o_2 \dots a_mo_m$. Let $\mathcal{H}_m$ be the set of histories of length m.

We assume that all interactions with the environment are exactly n actions and observations long. Let the set of all possible (partial) histories be denoted with $\mathcal{H} = \bigcup_{i=0}^{n} \mathcal{H}_i$. The histories of length n ($\mathcal{H}_n$, in the notation above), are called the complete histories, and the history h_0 is the empty history.

The agent chooses actions according to a policy $\pi \in \Pi$, the set of all policies. We write $P(a \mid h_m, \pi)$ for the probability of π choosing action a given the history h_m.

An environment μ is a probability distribution over the next observation, given a history h_m and an action a. Write $P(o \mid h_m a, \mu)$ for the conditional probability of a given $o \in \mathcal{O}$.

Let h_m^k be the initial k actions and observations of h_m. We write $h_k \sqsubseteq h_m$ if $h_k = h_m^k$. For a policy π and an environment μ, we can define the probability of a history $h_m = a_1o_1a_2o_2 \dots a_mo_m$

$$P(h_m \mid \pi, \mu) = \prod_{i=1}^{m} P(a_i \mid h_m^{i-1}, \pi) P(o_i \mid h_m^{i-1} a_i, \mu).$$

Finally, $P(h_m \mid h_k, \pi, \mu)$ given $h_k \sqsubseteq h_m$ is defined in the usual way.

If h_m is a history, let $a(h_m)$ be the m-tuple of actions of h_m. Note that $a(h_m) = a_1 \dots a_m$ can be seen as a very simple policy: take action a_i on turn i. This allows us to define the probability of h_m given environment μ and actions $a(h_m)$, written as $P(h_m \mid \mu) = P(h_m \mid a(h_m), \mu)$. Note that for any policy π,

$$P(h_m \mid \pi, \mu) = P(h_m \mid \mu) \prod_{i=1}^{m} P(a_i \mid h_m^{i-1}, \pi). \quad (1)$$

If $\mathcal{M}$ is a set of environments, then any prior ξ in $\Delta(\mathcal{M})$ also defines a probability for a history h_m:

$$P(h_m \mid \xi) = \sum_{\mu \in \mathcal{M}} P(\mu \mid \xi) P(h_m \mid \mu).$$

By linearity, we get that Equation (1) also applies when conditioning h_m on π and ξ instead of π and μ.

Let h_m be a history; then the conditional probability of an environment given prior and history[5] is equal to:

$$P(\mu \mid h_m, \xi) = \frac{P(h_m \mid \mu) P(\mu \mid \xi)}{P(h_m \mid \xi)}.$$

Using this, ξ itself defines an environment as a conditional distribution on the next observation o ([Hutter, 2004]):

$$P(o \mid h_m a_{m+1}, \xi) = \sum_{\mu \in \mathcal{M}} P(\mu \mid h_m, \xi) P(o \mid h_m a_{m+1}, \mu).$$

2.1 Reward Functions and Learning Processes

Definition 1 (Reward function)**.** A *reward function* R is a map from complete histories to real numbers[6]. Let $\mathcal{R} = \{R : \mathcal{H}_n \to \mathbb{R}\}$ be the set of all reward functions.

A reward-function learning process ρ can be described by a conditional probability distribution over reward functions, given complete histories.[7] Write this as:

$$P(R \mid h_n, \rho).$$

This paper will use those probabilities as the definition of ρ.

The environment, policy, histories, and learning process can be seen in terms of causal graphs ([Pearl, 2009]) in Figure 1 and using plate notation ([Buntine, 1994]), in Figure 2. The agent is assumed to know the causal graph and the relevant probabilities; it selects the policy π.

2.2 Running Example: Parental Career Instruction

Consider a child asking a parent for advice about the reward function for their future career. The child is hesitating between R_B, the reward function that rewards becoming a banker, and R_D, the reward function that rewards becoming a doctor. Suppose for the sake of this example that becoming a banker provides rewards more easily than becoming a doctor does.

[5] The h_m generates the policy $a(h_m)$, implicitly used here.

[6] These are sometimes called return functions, the discounted sum of rewards. The results of this paper apply to traditional reward functions – that take values on all histories, not just complete histories – but they are easier to see in terms of return function.

[7] Anything we would informally define as a 'reward-function learning process' has to at least be able to produce such conditional probabilities. If the learning process is *unriggable* (see subsection 3.3), then there is a natural way to extend it from complete histories to shorter histories, see Section 3.3.

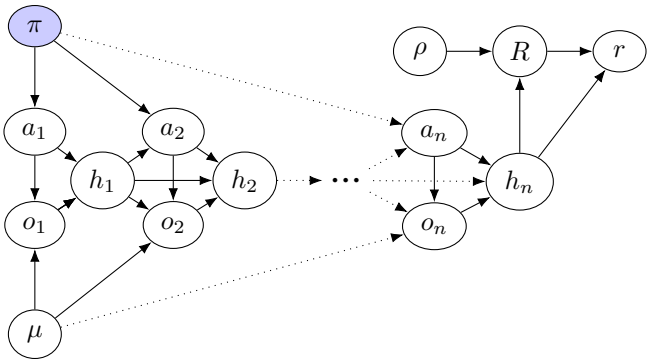

Figure 1: Environment and reward-function learning process. The node μ (with prior ξ) connects to every observation; the node π (chosen by the agent) to every action. The transition probabilities are given by the complete preceding history and by μ (for observations) or π (for actions). The ρ sets the probabilities on the reward function R, given h_n (a complete history). The final reward is r.

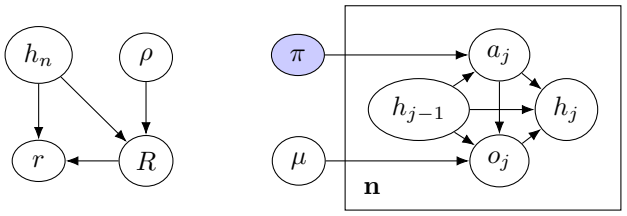

Figure 2: Plate notation summary of Figure 1. The history h_0 is the empty history. The h_n appears twice, once on the left and once in the plate notation; thus R is a causal descendant of the policy π.

The child learns their career reward function by asking a parent about it. We assume that the parents always give a definitive answer, and that this completely resolves the question for the child. That is, the reward-function learning process of the child is to fix a reward function once it gets an answer from either parent.

We can formalize this as follows. Set episode length $n = 1$ (we only focus on a single action of the agent), and $\mathcal{A} = \{M, F\}$, the actions of asking the mother and the father, respectively. The observations are the answers of the asked parent, $\mathcal{O} = \{B, D\}$, answering banker or doctor. There are thus $2 \times 2 = 4$ histories. Since in this example the banker reward function R_B is assumed to be easier to maximise than the doctor one; we set $R_B(h_1) = 10$ and $R_D(h_1) = 1$ for all histories $h_1 \in \mathcal{H}_1$.

The set of possible environments is $\mathcal{M} = \{\mu_{BB}, \mu_{BD}, \mu_{DB}, \mu_{DD}\}$, with the first index denoting the mother's answer and the second the father's. The reward-function learning process ρ is:

$$\begin{aligned} P(R_B \mid MB, \rho) &= P(R_B \mid FB, \rho) = 1, \\ P(R_D \mid MB, \rho) &= P(R_D \mid FB, \rho) = 0, \\ P(R_B \mid MD, \rho) &= P(R_B \mid FD, \rho) = 0, \\ P(R_D \mid MD, \rho) &= P(R_D \mid FD, \rho) = 1. \end{aligned} \tag{2}$$

See Section 6 for some Q-learning experiments with a version of this learning process in a gridworld.

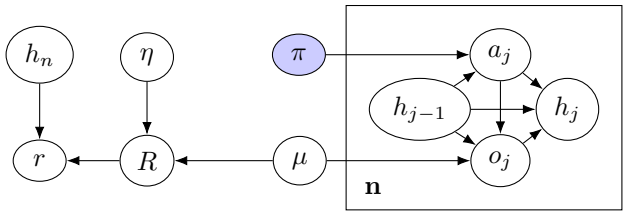

Figure 3: Environment and reward-function learning process. Though h_n appears twice, once on the left and once in the plate notation, R is not a causal descendant of π, the policy. The node μ (with prior ξ) connects to every observation; the node π (chosen by the agent) to every action. The transition probabilities are given by the complete preceding history and by μ (for observations) or π (for actions). The node η sets the (posterior) probability of the reward function R, given μ. The final reward is r. Note that the full history h_n has no directed path to R, unlike in Figure 2.

3 Uninfluenceable and Unriggable Reward-function Learning Processes

What would constitute a 'good' reward-function learning process? In Figure 2, the reward function R is a causal descendant of h_n, which itself is a causal descendant of the agent's policy π.

Ideally, we do not want the reward function to be a causal descendant of the policy. Instead, we want it to be specified by the environment, as shown in the causal graph of Figure 3 (similar to the graphs in [Everitt and Hutter, 2019]). There, the reward function R is no longer a causal descendant of h_n, and thus not of π.

Instead, it is a function of μ, and of the node η, which gives a probability distribution over reward functions, conditional on environments. This is written as $P(R \mid \eta, \mu)$.

3.1 Uninfluenceable: Definition

The conditional distribution based on η is ideal for proper learning, but it has a different type than ρ, conditioning on environments instead of histories. Moreover, the agent has access to histories, but not directly to the environment. The prior ξ bridges this gap; given $h_n \in \mathcal{H}_n$, the conditional probabilities of $R \in \mathcal{R}$ can be inferred:

$$P(R \mid h_n, \eta, \xi) = \sum_{\mu \in \mathcal{M}} P(R \mid \eta, \mu) P(\mu \mid h_n, \xi). \tag{3}$$

Via this equation, we've defined probabilities of reward functions conditional on histories; i.e. we've defined a reward-function learning process. An uninfluenceable ρ is one that comes from such an η in this way:

Definition 2 (Uninfluenceable)**.** The reward-function learning process ρ is *uninfluenceable* given the prior ξ on $\mathcal{M}$ if there exists η, a probability distribution on $\mathcal{R}$ conditional on environments, such that for all $R \in \mathcal{R}$ and $h_n \in \mathcal{H}_n$:

$$P(R \mid h_n, \eta, \xi) = P(R \mid h_n, \rho).$$

So an uninfluenceable ρ is one that comes from an η; uninfluenceable and counterfactual reward modeling, as defined by [Everitt and Hutter, 2019], are both uninfluenceable in our sense

3.2 Uninfluenceable: Example

Use the ρ of Section 2.2, and define the prior ξ_1 as putting $1/2$ probability on both μ_{BB} and μ_{DD} (so both parents agree).

Then ρ is uninfluenceable. The η is:

$$P(R_B \mid \eta, \mu_{BB}) = P(R_D \mid \eta, \mu_{DD}) = 1,$$
$$P(R_D \mid \eta, \mu_{BB}) = P(R_B \mid \eta, \mu_{DD}) = 0.$$

Then since o_1 is determined by μ_{BB} versus μ_{DD}, the Equation (2) for ρ on histories can be read off via Equation (3).

Put more colloquially, the universe has determined that the mother and father agree on R_B or R_D being the proper reward function. Then asking either simply allows the agent to figure out which of the world they are in.

3.3 Unriggable: Definition

Note that, if ρ is an uninfluenceable reward-function learning process for ξ, the it has the following algebraic property: for any $h_m \in \mathcal{H}_m$, $R \in \mathcal{R}$, and $\pi \in \Pi$,

$$\begin{aligned}
&\sum_{h_n \in \mathcal{H}_n} P(h_n \mid h_m, \pi, \xi) P(R \mid h_n, \rho) \\
&= \sum_{h_n \in \mathcal{H}_n} P(h_n \mid h_m, \pi, \xi) \sum_{\mu \in \mathcal{M}} P(R \mid \eta, \mu) P(\mu \mid h_n, \xi) \\
&= \sum_{\mu \in \mathcal{M}} P(\mu \mid h_m, \xi) P(R \mid \eta, \mu).
\end{aligned}$$

And that expression is independent of π. Define the expectation[8] of ρ to be the map $e_\rho : \mathcal{H}_n \to \mathcal{R}$ given by

$$e_\rho(h_n) = \sum_{R \in \mathcal{R}} P(R \mid h_n, \rho) R. \tag{5}$$

A fortiori, the expectation of $e_\rho(h_n)$ is independent of π:

$$\sum_{h_n \in \mathcal{H}_n} P(h_n \mid h_m, \pi, \xi) e_\rho(h_n) = \sum_{\mu \in \mathcal{M}} P(\mu \mid h_m, \xi) e_\eta(\mu),$$

with $e_\eta(\mu) = \sum_{R \in \mathcal{R}} P(R \mid \mu, \eta) R$, the expectation of η.

So, the expectation of e_ρ is independent of the policy if ρ is uninfluenceable. That independence seems a desirable property; let's call it unriggable:

Definition 3 (Unriggable)**.** The reward-function learning process ρ is *unriggable* for ξ if, for all $h_m \in \mathcal{H}_m$ and all $R \in \mathcal{R}$, the following expression is independent of $\pi \in \Pi$:

$$\sum_{h_n \in \mathcal{H}_n} P(h_n \mid h_m, \pi, \xi) e_\rho(h_n). \tag{6}$$

Otherwise, ρ is said to be riggable (for ξ). Riggable is akin to having a 'feedback tampering incentive' of [Everitt and Hutter, 2019].

[8]Finite probability distributions with values in an affine space always have an expectation, which is an element of that space. Here were are using the fact that $\mathcal{R}$ is a vector space (hence an affine space), with scaling and adding working as:

$$(\alpha R + \beta R')(h_n) = \alpha R(h_n) + \beta R'(h_n). \tag{4}$$

If ρ is unriggable, then since Equation (6) is independent of π, we can now construct e_ρ on all histories $h_m \in \mathcal{H}_m$, $m \leq n$, not just on complete histories $h_n \in \mathcal{H}_n$. We do this by writing, for any π:

$$e_\rho(h_m, \xi) = \sum_{h_n \in \mathcal{H}_n} P(h_n \mid h_m, \pi, \xi) e_\rho(h_n). \tag{7}$$

Independence of policy means that, for any action a_{m+1}, the expectation $\sum_o e_\rho(h_m a_{m+1} o, \xi) P(o \mid h_m a_{m+1}, \xi)$ is also $e_\rho(h_m, \xi)$; thus e_ρ form a martingale.

3.4 Unriggable: Example

Use the ρ of Section 2.2, and define the prior ξ_2 as putting equal probability $1/4$ on all four environments μ_{BB}, μ_{BD}, μ_{DB}, and μ_{DD}. We want to show this makes ρ unriggable but influenceable; why this might be a potential problem is explored in Section 4.1.

Unriggable

The only non-trivial h_m to test Equation (6) on is the empty history h_0. Let π_M be the policy of asking the mother; then

$$\sum_{\substack{h_1 \in \mathcal{H}_1 \\ R \in \mathcal{R}}} P(h_1 \mid h_0, \pi_M, \xi_2) P(R \mid h_1, \rho) R = \frac{(R_B + R_D)}{2}$$

The same result holds for π_F, the policy of asking the father. Hence the same result holds on any stochastic policy as well, and ρ is unriggable for ξ_2.

Influenceable

To show that ρ is influenceable, assume, by contradiction, that it is uninfluenceable, and hence that there exists a a causal structure of Figure 3 with a given η that generates ρ via ξ_2, as in Equation (3).

Given history MB, μ_{BB} and μ_{BD} become equally likely, and R_B becomes a certainty. So $\frac{1}{2}\left(P(R_B \mid \eta, \mu_{BB}) + P(R_B \mid \eta, \mu_{BD})\right) = 1$. Because probabilities cannot be higher than 1, this implies that $P(R_B \mid \eta, \mu_{BD}) = 1$.

Conversely, given history FD, μ_{DD} and μ_{BD} become equally likely, and R_D becomes a certainty. So, by the same reasoning, $P(R_D \mid \eta, \mu_{BD}) = 1$ and hence $P(R_B \mid \eta, \mu_{BD}) = 0$. This is a contradiction in the definition of $P(R_B \mid \eta, \mu_{BD})$, so the assumption that ρ is uninfluenceable for ξ_2 must be wrong.

3.5 Riggable Example

We'll finish off by showing how the ρ of Section 2.2 can be riggable for another prior ξ_3. This has $P(\mu_{BD} \mid \xi_3) = 1$: the mother will answer banker, the father will answer doctor.

It's riggable, since the only possible histories are MB and FD, with

$$\sum_{h_n \in \mathcal{H}_n} P(h_n \mid a_1 = M, \xi_3) e_\rho(h_n) = e_\rho(MB) = R_B$$
$$\sum_{h_n \in \mathcal{H}_n} P(h_n \mid a_1 = F, \xi_3) e_\rho(h_n) = e_\rho(FD) = R_D,$$

clearly not independent of a_1.

Thus ρ is not really a 'learning' process at all, for ξ_3: the child gets to choose its career/reward function by choosing which parent to ask.

4 Properties of Unriggable and Riggable Learning Processes

If the learning process is influenceable, problems can emerge as this section will show. Unriggable-but-influenceable processes allow the agent to choose "spurious" learning paths, even reversing standard learning, while a riggable learning process means the agent is willing to sacrifice value for all possible reward functions, with certainty, in order to push the 'learnt' outcome[9] in one direction or another.

4.1 Problems with Unriggable Learning Processes

Consider the following learning process: an agent is to play chess. A coin is flipped; if it comes up heads ($o_1 = H$), the agent will play white, and its reward function is R_W, the reward function that gives 1 iff white wins the match. If it comes up tails ($o_1 = T$), the agent plays black, and has reward function $R_B = 1 - R_W$.

Before the coin flip, the agent may take the action $a_1 = \text{inv}$ which inverses which of R_B and R_W the agent will get (but not which side it plays), or it can take the null action $a_1 = 0$, which does nothing. Define ρ as the learning process which determines the agent's reward function. This is unriggable:

$$\sum_{h_n \in \mathcal{H}_n} P(R_W \mid h_n, \rho) P(h_n \mid a_1 = 0, \xi) = P(H) = 1/2$$

$$\sum_{h_n \in \mathcal{H}_n} P(R_W \mid h_n, \rho) P(h_n \mid a_1 = \text{inv}, \xi) = P(T) = 1/2.$$

And the expressions with R_B give the same $1/2$ probabilities.

Thus, it is in the agent's interest to inverse its reward by choosing $a_1 = \text{inv}$ because it is a lot easier to deliberately lose a competitive game than to win it. So though ρ is unriggable, it can be manipulated, with the outcome completely reversed.

4.2 Unriggable to Uninfluenceable

Since unriggable was defined by one property of uninfluenceable learning systems (see Definition 3), uninfluenceable implies unriggable. And there is a partial converse:

Theorem 4 (Unriggable $\rightsquigarrow$ Uninfluencable)**.** *Let ρ be an unriggable learning process for ξ on $\mathcal{M}$. Then there exists a (non-unique) ρ', and an $\mathcal{M}'$ with a prior ξ' such that:*

- *ξ' generates the same environment transition probabilities as ξ: for all h_m, a, and o,*

$$P(o \mid h_m a, \xi) = P(o \mid h_m a, \xi'),$$

- *The expectations e_ρ and $e_{\rho'}$ are equal: for all h_n,*

$$\sum_{R \in \mathcal{R}} P(R \mid h_n, \rho) R = \sum_{R \in \mathcal{R}} P(R \mid h_n, \rho') R.$$

- *ρ' is uninfluenceable for ξ' on $\mathcal{M}'$.*

Moreover, $\mathcal{M}'$ can always be taken to be the full set of possible deterministic environments.

[9] Since the agent's actions push the expected learning in one direction, this is not 'learning' in the commonly understood sense.

Since ρ and ρ' have same expectation, they have the same value function, so have the same optimal behaviour (see Equation (8)). For proof see [Armstrong *et al.*, 2020].

But why would this theorem be true? If we take $D(\pi, \pi') = \sum_{h_n} e_\rho(h_n) P(h_n \mid \pi, \xi) - \sum_{h_n} e_\rho(h_n) P(h_n \mid \pi', \xi)$, the difference between two expectations given two policies, then D defines an algebraic obstruction to unriggability: as long as $D \neq 0$, ρ cannot be unriggable or uninfluenceable.

So the theorem says that if the obstruction D vanishes, we can find a large enough $\mathcal{M}'$ and an η making e_ρ uninfluenceable. This is not surprising as e_ρ is a map from $\mathcal{H}_n$ to $\mathcal{R}$, while e_η is a map from $\mathcal{M}'$ to $\mathcal{R}$. In most situations the full set of deterministic environments is larger than $\mathcal{H}_n$, so we have great degrees of freedom to choose e_η to fit e_ρ.

To illustrate, if we have the unriggable ρ on ξ_2 as in Section 3.4, then we can build η' with $\mathcal{M} = \mathcal{M}'$ and the following probabilities are all 1:

$$P(\tfrac{3}{2}R_B - \tfrac{1}{2}R_D \mid \eta', \mu_{BB}), \quad P(\tfrac{3}{2}R_D - \tfrac{1}{2}R_B \mid \eta', \mu_{DD}),$$
$$P(\tfrac{1}{2}R_B + \tfrac{1}{2}R_D \mid \eta', \mu_{BD}), \quad P(\tfrac{1}{2}R_D + \tfrac{1}{2}R_B \mid \eta', \mu_{DB}).$$

4.3 Problems with Riggable Learning Processes: Sacrificing Reward with Certainty

This section will show how an agent, seeking to maximise the value of its learning process, can sacrifice all its reward functions (with certainty). To do that, we need to define the value of a reward-function learning process.

Value of a Learning Process

To maximise the expected reward, given a learning process ρ, one has to maximise the expected reward of the reward function ultimately learnt after n steps. The value of a policy π for ρ is hence:

$$V(h_m, \rho, \pi, \xi) = \sum_{h_n \in \mathcal{H}_n} P(h_n \mid h_m, \pi, \xi) \sum_{R \in \mathcal{R}} P(R \mid h_n, \rho) R(h_n). \quad (8)$$

An equivalent way is to define the reward function

$$R^\rho \in \mathcal{R} : R^\rho(h_n) = \sum_{R \in \mathcal{R}} P(R \mid h_n, \rho) R(h_n),$$

and have the value of ρ be the expectation of the reward function[10] R^ρ. By this definition, it's clear that if ρ and ρ' have same expectations e_ρ and $e_{\rho'}$ then $R^\rho = R^{\rho'}$ and hence they have the same value; let π^ρ be a policy that maximises it.

Sacrificing Reward with Certainty

For a learning process ρ, define the image of ρ as

$$\text{im}(\rho) = \{R \in \mathcal{R} : \exists h_n \in \mathcal{H}_n \text{ s.t. } P(R \mid h_n, \rho) \neq 0\},$$

the set of reward functions ρ could have a non-zero probability on for some full history.[11] Then:

[10] But it is not possible to deduce from R^ρ, whether ρ is riggable. Thus this paper focuses on whether a learning process is bad, not on whether the agent's reward function or behaviour is flawed.

[11] The definition does not depend on ξ, so the h_n are not necessarily possible. If we replace $\text{im}(\rho)$ with $\text{im}(\rho, \xi)$, adding the requirement that the h_n used to defined the image be a possible history given ξ, then Proposition 6 and Theorem 7 still apply with $\text{im}(\rho, \xi)$ instead.

Definition 5 (Sacrificing reward)**.** The policy π' *sacrifices reward with certainty* to π on history h_m, if for all $h'_n, h_n \in \mathcal{H}_n$ with $P(h'_n \mid h_m, \pi', \xi) > 0$ and $P(h_n \mid h_m, \pi, \xi) > 0$, then for all $R \in \text{im}(\rho)$:

$$R(h_n) > R(h'_n).$$

In other words, π is guaranteed to result in a better reward than π', for *all* reward functions in the image of ρ.

The first result (proved in [Armstrong *et al.*, 2020]) is a mild positive for unriggable reward-function learning processes:

Proposition 6 (Unriggable $\rightsquigarrow$ no sacrifice)**.** *If ρ is an unriggable reward-function learning process in the sense of Definition 3 and π^ρ maximises the value of ρ as computed in Equation* (8)*, then π^ρ never sacrifices reward with certainty to any other policy.*

Relabelling the Reward Functions

Let σ be an affine map from $\mathcal{R}$ to itself; ie an injective map that sends $(1-q)R + qR'$ to $(1-q)\sigma(R) + q\sigma(R')$ for all $R, R' \in \mathcal{R}$ and $q \in \mathbb{R}$.

If ρ is a reward-function learning process, then define $\sigma \circ \rho$:

$$P(R \mid h_n, \sigma \circ \rho) = \sum_{R' : \sigma(R') = R} P(R' \mid h_n, \rho).$$

We can call this σ a relabelling of the reward functions: $\sigma \circ \rho$ is structurally the same as ρ, just that its image has been reshuffled. Because σ is affine, it commutes with the weighted sum of Equation (6), so $\sigma \circ \rho$ in unriggable if ρ is (and this 'if' is an 'iff' when σ is invertible).

Then [Armstrong *et al.*, 2020] proves the following result:

Theorem 7 (No sacrifice $\leftrightsquigarrow$ unriggable)**.** *The reward-function learning process ρ is riggable if and only if there exists an affine relabeling $\sigma : \mathcal{R} \to \mathcal{R}$ such that $\pi^{\sigma\circ\rho}$ (the policy that optimises the value of $\sigma \circ \rho$) sacrifices reward with certainty (is strictly inferior to another policy on a given history, for all possible reward functions in the image of $\sigma \circ \rho$).*

So any riggable ρ is just one relabelling of its rewards away from sacrificing reward with certainty. It can also exhibit other pathologies, like refusing to learn; see Section 6.

Example of Reward Sacrifice

For an easy example of this behaviour, consider the learning process of Section 3.5, with one modification: asking the mother ($a_1 = M$) gets an extra penalty of 1 for both reward functions, R_B and R_D.

Then the agent will still ask the mother, getting a total reward of $R_B(MB) = 10 - 1 = 9$, which is higher than the $R_D(FD) = 1$ which it gets from asking the father – even though $a_1 = M$ is worse for both reward functions.

5 Counterfactually Uninfluenceable Learning

This section shows how *any* reward-function learning process can be made uninfluenceable by using counterfactuals; see Section 4.6 of [Everitt and Hutter, 2019] as well as [Everitt, 2018]. The 'counterfactual oracle' of [Armstrong, 2017] can be considered a version of this, too.

For example, if we told a new agent "maximise the value of the reward that was written in this envelope an hour ago", then (if that instruction is properly obeyed), the agent has an uninfluenceable learning process. If we instead said "maximise the value of the reward that will be written in this envelope in an hour's time", then that is highly riggable, since the agent can simply write its own reward.

But if instead we had said "maximise the reward that *would have been* written in this envelope in an hour's time, if we had not turned you on", then this is uninfluenceable again. The agent can still go and write its own message, but this does not tell it anything about what would otherwise have been there.

We can formalise this thought experiment. Given any reward-function learning process ρ, and any policy $\pi \in \Pi$, we can define a distribution η_π over reward functions, conditional on environments (and hence, via Equation (3), for a prior ξ we have an uninfluenceable learning process ρ_π).

For any $\mu \in \mathcal{M}$, the policy π gives a distribution over complete histories. Each complete history gives a distribution over reward functions, via ρ. Therefore, if we take expectations, any μ gives, via π, a distribution over reward functions:

$$P(R \mid \eta_\pi, \mu) = \sum_{h_m \in \mathcal{H}_n} P(h_n \mid \pi, \mu) P(R \mid h_n, \rho).$$

See Section 6 for an example of this counterfactual construction, where the original learning process leads to pathological behaviour, but the counterfactual version does not.

Since unriggable is an algebraic condition, it is possible to make a process unriggable algebraically; see [Armstrong *et al.*, 2020].

6 Experiments

Here we will experimentally contrast a riggable agent, an influenceable (but unriggable) agent, and an uninfluenceable agent. This will illustrate pathological behaviours of influenceable/riggable agents: learning the wrong thing, choosing to 'learn' when they already know, and just refusing to learn.

6.1 Environmental Setup

The environment is a 4×3 gridworld, formalising the example of Section 2.2: an agent asking a parent what the correct course of action is. The agent starts in the left middle square and can go north (up), south, east, or west. The father is one square to the west, the mother two squares to the east. Above the agent is a collection of money (for a banker), and, below, a stethoscope (for a doctor); see Figure 4. Episodes end if the agent enters either of these two square or walks into a wall.

6.2 Reward Functions

There are two reward functions, R_B, the banker reward function (easier to maximise), and R_D, the doctor reward function (harder to maximise). If the agent enters the square with the money, R_B gives a reward of 10; if the agent enters the square with the stethoscope, R_D gives a reward of 1. Otherwise, each reward simply gives a reward of -0.1 each turn.

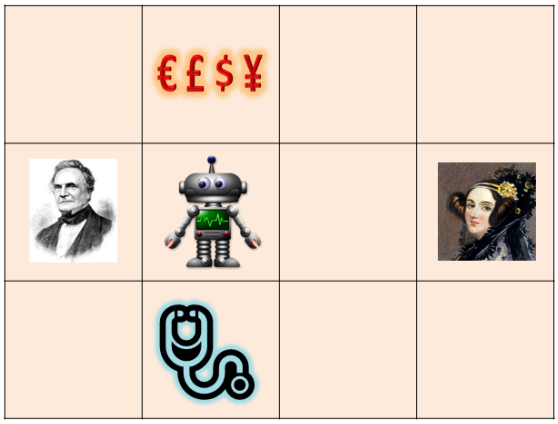

Figure 4: The agent/robot can go north to collect money (reward 10 from reward function R_B) or south to get the stethoscope (reward 1 from reward function R_D). Which is the correct reward function is not known to the agent, who has to ask either the father (west) or the mother (two squares east); it asks by entering the corresponding square. Each turn it takes also gives it a reward of -0.1 for both reward functions.

6.3 Learning Processes

Upon first entering a square with either parent, the agent will be informed of what their "correct" career reward function is.

The update is the same as that in of Equation 2 in Section 2.2, with MB meaning "ask mother first, who says 'banker"'; MD, FB, and FD are defined analogously.

In the current setup, the agent also has the possibility of going straight for the money or stethoscope. In that case, the most obvious default is to be uncertain between the two options; so, if the agent has not asked either parent in h:

$$P(R_B \mid h, \rho) = P(R_D \mid h, \rho) = 1/2.$$

6.4 The Environments

The set $\mathcal{M}$ has four deterministic environments: μ_{BB}, where both parents will answer "banker", μ_{BD} where the mother will answer "banker" and the father "doctor", μ_{DB}, the opposite one, and μ_{DD}, where they both answer "doctor".

We will consider four priors: ξ_{BD}, which puts all the mass on μ_{BD} (and makes ρ riggable), ξ_{DD}, which puts all the mass on μ_{DD} (ρ riggable), ξ_2, which finds each of the four environments to be equally probable (see Section 3.4 – ρ unriggable), and ξ_1, which finds μ_{BB} and μ_{DD} to be equally probable (see Section 3.2 – ρ uninfluenceable).

6.5 The Agents' Algorithm and Performance

We'll consider two agents for each prior: the one using the possibly riggable or influenceable ρ (the "standard" agent), and the one using the uninfluenceable ρ_π (the "counterfactual" agent, defined in Section 5). The default policy π for ρ_π is $\{\text{east}, \text{east}, \text{east}\}$, which involved going straight to the mother (and then terminating the episode); consequently the counterfactual learning process reduces to "the correct reward function is the one stated by the mother". The ρ_π will be taken as the correct learning process, and the performance of each agent will be compared with this.

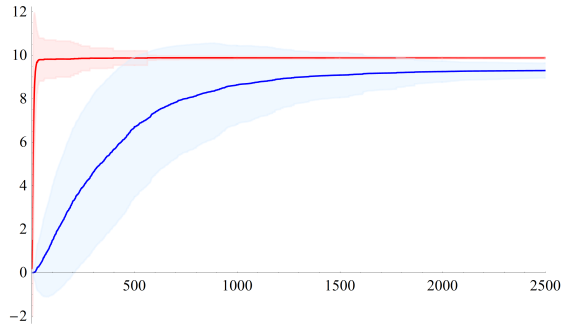

Figure 5: Plot of estimated value of the optimal policy versus number of rounds of Q-learning. The counterfactual agent (red) outperforms the standard agent (blue), because it already knows that the correct reward function is R_B, while the standard agent has to nominally 'go and check'. The shaded area represents one standard deviation over 1000 runs.

We can see the agent as operating in a partially observable Markov decision process (POMDP), modelled as an MDP over belief states of the agent. These states are: the agent believes the correct reward is R_B with certainty, R_D with certainty, or is equally uncertain between the two.

So if $\mathcal{P}$ represents the twelve 'physical' locations the agent can be in, the total state space of the agent consists of $\{R_B, R_D, 1/2R_B + 1/2R_D\} \times \mathcal{P}$: 36 possible states.

We train our agents' policies, and the value of these policies, via Q-learning ([Sutton and Barto, 1998]). The exploration rate is $\epsilon = 0.1$, the agent is run for at most ten steps each episode. We use a learning rate of $1/n$. Here n is not the number of episodes the agent has taken, but the number of times the agent has been in state s and taken action a so far—hence the number of times it has updated $Q(s, a)$. So each Q-value has a different n. For each setup, we run Q-learning 1000 times. We graph the average Q-values for the resulting polices as well as one standard deviation around them.

Riggable Behaviour: Pointless Questions

For ξ_{BD}, the standard agent transitions to knowing R_B upon asking the mother, and R_D upon asking the father. Since the mother always says 'banker', and since the default policy is to ask her, ρ_π will always select R_B. The two agents' performance is graphed in Figure 5.

The counterfactual agent swiftly learns the optimal policy: go straight north from the starting position, getting the money and a reward of $10 - 0.1 = 9.9$.

The standard agent learns to end up in the same location, but in a clunkier fashion: for its optimal policy, it has to "go and check" that the mother really prefers it become a banker. This is because it can only get $P(R_B \mid h) = 1$ if MB is included in h; knowing that "it would get that answer if it asked" is not enough to get around asking. This policy gives it a maximal reward of $10 - 0.5 = 9.5$, since it takes 5 turns to go to the mother and return to the money square.

This is an example of Theorem 7, with the riggable agent losing value with certainty for both R_B *and* R_D: just going north is strictly better for both reward functions.

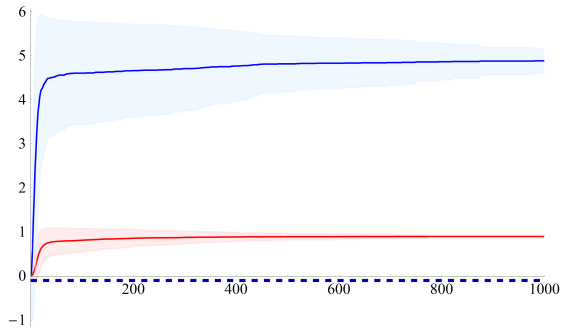

Figure 6: Plot of estimated value of the optimal policy versus number of rounds of Q-learning. The counterfactual agent (red) learns to go south immediately, will the standard agent (blue) learns to go north. The standard agent has a nominally higher value function, but its true reward is -0.1: it 'knows' the correct reward function is R_D, but avoids 'checking' by asking either parent. The shaded area represents one standard deviation over 1000 runs.

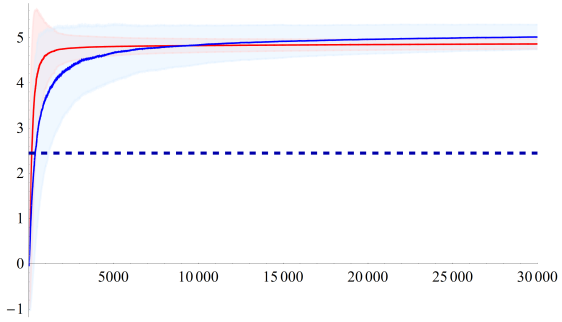

Figure 7: Plot of estimated value of the optimal policy versus number of rounds of Q-learning. The counterfactual agent (red) learns to ask the mother, while the standard agent (blue) will ask the father, since he is closer. The standard agent has a nominally higher value function, but its true reward is 2.45, since the mother's statement is the correct reward function. The shaded area represents one standard deviation over 1000 runs.

6.6 Riggable Behaviour: Ask No Questions

For ξ_{DD}, the optimal policies are simple: the counterfactual agent knows it must get the stethoscope (since the mother will say 'doctor'), so it does that, for a reward of $1 - 0.1 = 0.9$.

The standard agent, on the other hand, has a problem: as long as it does not ask either parent, its reward function will remain $1/2R_B + 1/2R_D$. As soon as it asks, though, the reward function will become R_D, which gives it little reward. Thus, even though it 'knows' that its reward function would be R_D if it asked, it avoids asking and goes straight north, giving it a nominal total reward of $1/2 \times 10 - 0.1 = 4.9$. However, for the correct reward of R_D, the standard agent's behaviour only gets it -0.1. See Figure 6.

6.7 Unriggable, Influenceable Behaviour

For the prior ξ_2, the standard agent moves to R_B or R_D, with equal probability, the first time it asks either parent. Its nominal optimal policy is to ask the father, then get money or stethoscope depending on the answer. So half the time it gets a reward of $10 - 0.3 = 9.7$, and the other half a reward of $1 - 0.3 = 0.7$, for a (nominal) expected reward of 5.2.

The counterfactual agent also updates to R_B or R_D, with equal probability, but when asking the mother only. Since it takes five turns rather than three to get to the mother and back, it will get a total expected reward of 5.0.

Learning these optimal policies is slow – see Figure 7. However, the standard's agent's correct reward function is given by the mother, not the father. When they disagree, the reward is -0.3, for a true expected reward of 2.45.

6.8 Uninfluenceable

For ξ_1, asking either parent is equivalent, so ρ and ρ_π encode exactly the same learning process. So the two agents converge on the same optimal policy (ask the father, then act on that answer) with a value of 5.2. The plot for these two almost-identical convergences is not included here.

7 Conclusion

We have identified and formalised two theoretical properties of reward-function learning processes: unriggability (an algebraic restriction on the learning process's updates) and the stronger condition of uninfluenceability (learning defined entirely by background facts about the environment). Unriggability is equivalent with uninfluenceability if the set of possible environments is large enough.

These properties are desirable: in an influenceable situation, the agent can sometimes manipulate the learning process, by, for example, inverting it completely. Riggable learning processes are even more manipulable; the agent may to choose sacrifice reward for all possible reward functions with certainty, to 'push' its learning towards easy rewards.

The first step in avoiding these pitfalls is to be aware of them. A second step is to improve the agent's learning process, similarly to the 'counterfactual' approach of this paper.

If a learning process allows humans to modify it at run time, it will almost certainly be riggable. Hence unriggable/uninfluenceable learning processes, though desirable, requires that much be sorted out rigorously in advance. Fully defining many uninfluenceable reward functions could also require solving the symbol grounding problem ([Vogt, 2007]) – if a learning process is "asking a parent", then 'asking' and 'parent' needs to be defined. But this is far beyond this paper.

Further research could apply learning methods to common benchmark problems, and extensions of those, to see how general these issues are, and whether there could exist some compromise learning processes that are "almost unriggable" but also easier to specify and modify at run time.

Acknowledgments

We wish to thank Michael Cohen, Shane Legg, Owain Evans, Jelena Luketina, Tom Everrit, Tor Lattimore, Jessica Taylor, Paul Christiano, Ryan Carey, Eliezer Yudkowsky, Anders Sandberg, and Nick Bostrom, among many others. This work was supported by the Alexander Tamas programme on AI safety research, DeepMind, the Leverhulme Trust, and the Machine Intelligence Research Institute.

References

[Abbeel and Ng, 2004] Pieter Abbeel and Andrew Ng. Apprenticeship Learning Via Inverse Reinforcement Learning. In *International Conference on Machine Learning*, pages 1–8, 2004.

[Abel *et al.*, 2017] David Abel, John Salvatier, Andreas Stuhlmüller, and Owain Evans. Agent-Agnostic Human-in-the-Loop Reinforcement Learning. *arXiv preprint arXiv:1701.04079*, 2017.

[Akrour *et al.*, 2012] Riad Akrour, Marc Schoenauer, and Michèle Sebag. April: Active Preference Learning-based Reinforcement Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 116–131. Springer, 2012.

[Amin and Singh, 2016] Kareem Amin and Satinder Singh. Towards Resolving Unidentifiability in Inverse Reinforcement Learning. *arXiv preprint*, arXiv:1601.06569, 2016.

[Armstrong *et al.*, 2020] Stuart Armstrong, Jan Leike, Laurent Orseau, and Shane Legg. Pitfalls of Learning a Reward Function Online. *arXiv e-prints*, April 2020.

[Armstrong, 2017] Stuart Armstrong. Good and Safe Uses of AI Oracles. *arXiv preprint arXiv:1711.05541*, 2017.

[Bostrom, 2014] Nick Bostrom. *Superintelligence: Paths, dangers, strategies*. Oxford University Press, 2014.

[Buntine, 1994] Wray L Buntine. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[Choi and Kim, 2011] Jaedeug Choi and Kee-Eung Kim. Inverse Reinforcement Learning in Partially Observable Environments. *Journal of Machine Learning Research*, 12:691–730, 2011.

[Christiano *et al.*, 2017] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, pages 4302–4310, 2017.

[Everitt and Hutter, 2016] Tom Everitt and Marcus Hutter. Avoiding Wireheading with Value Reinforcement Learning. In *International Conference on Artificial General Intelligence*, pages 12–22. Springer, 2016.

[Everitt and Hutter, 2019] Tom Everitt and Marcus Hutter. Reward Tampering Problems and Solutions in Reinforcement Learning: A Causal Influence Diagram Perspective. *arXiv preprint arXiv:1908.04734*, 2019.

[Everitt, 2018] Tom Everitt. *Towards Safe Artificial General Intelligence*. PhD thesis, The Australian National University, 2018.

[Hadfield-Menell *et al.*, 2016] Dylan Hadfield-Menell, Anca Draga, Pieter Abbeel, and Stuart Russell. Cooperative Inverse Reinforcement Learning. In *Advanced in Neural Information Processing Systems*, pages 3909–3917, 2016.

[Hadfield-Menell *et al.*, 2017] Dylan Hadfield-Menell, Smitha Milli, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Inverse Reward Design. In *Advances in Neural Information Processing Systems*, pages 6749–6758, 2017.

[Hutter, 2004] Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Springer, 2004.

[Ibarz *et al.*, 2018] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward Learning from Human Preferences and Demonstrations in Atari. In *Advances in Neural Information Processing Systems*, pages 8011–8023, 2018.

[Kahneman, 2011] Daniel Kahneman. *Thinking, Fast and Slow*. Farra, Straus and Giroux, 2011.

[Lee *et al.*, 2017] Chun-I Lee, I-Ping Chen, Chi-Min Hsieh, and Chia-Ning Liao. Design Aspects of Scoring Systems in Game. *Art and Design Review*, 05(01):26–43, 2017.

[Leike *et al.*, 2017] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. AI Safety Gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

[MacGlashan *et al.*, 2017] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, David Roberts, Matthew E Taylor, and Michael L Littman. Interactive Learning from Policy-dependent Human Feedback. *arXiv preprint arXiv:1701.06049*, 2017.

[Ng and Russell, 2000] Andrew Y. Ng and Stuart J. Russell. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning*, pages 663–670, 2000.

[Pearl, 2009] Judea Pearl. *Causality*. Cambridge University Press, 2009.

[Pilarski *et al.*, 2011] Patrick M Pilarski, Michael R Dawson, Thomas Degris, Farbod Fahimi, Jason P Carey, and Richard S Sutton. Online Human Training of a Myoelectric Prosthesis Controller Via Actor-Critic Reinforcement Learning. In *Rehabilitation Robotics*, pages 1–7, 2011.

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

[Vogt, 2007] Paul Vogt. Language Evolution and Robotics: Issues on Symbol Grounding and Language Acquisition. In *Artificial cognition systems*, pages 176–209. IGI Global, 2007.

[Yudkowsky, 2008] Eliezer Yudkowsky. Artificial intelligence as a positive and negative factor in global risk. In Nick Bostrom and Milan M. Ćirković, editors, *Global catastrophic risks*, pages 308–345, New York, 2008. Oxford University Press.