Assisted Robust Reward Design

Jerry Zhi-Yang He HZYJERRY@BERKELEY.EDU
Department of Computer Science
University of California, Berkeley
Berkeley, CA 94305, USA

Anca D. Dragan ANCA@BERKELEY.EDU
Department of Computer Science
University of California, Berkeley
Berkeley, CA 94305, USA

Editor: Workshop on Real World Experiment Design and Active Learning at ICML 2020

Abstract

Real-world AI systems often need complex reward functions. When we define the problem the AI needs to solve, we pretend that an engineer specifies this complex reward exactly and it is set in stone from then on. In practice, however, reward design is an *iterative* process: the engineer designs a reward, eventually encounters an environment where the reward incentivizes the wrong behavior, revises the reward, and repeats until convergence. What would it mean to rethink AI to formally account for this iterative nature of reward design? We propose that the AI system needs to not take the specified reward for granted, but rather have *uncertainty* about what the reward is, and account for the future iterations as *future evidence* it will receive. We contribute an AI-assisted reward design method that speeds up the design process by taking control over this future evidence: rather than letting the designer eventually encounter environments that require revising the reward, the system actively exposes the designer to the environments that have the most potential to narrow down what the reward should be. We test this method in an autonomous driving task, and find that it more quickly improves the car's behavior in held-out environments, and iteratively proposes environments that are "edge cases" for the current reward.

Keywords: Reward Design, Active Learning, Safety

1. Introduction

The job of an AI agent is to maximize its cumulative reward. From the perspective of the agent, it is as if the reward function fell from the sky: it is just there, it is something embedded in the very problem definition. In reality, there is a lot happening behind the scenes, where a human designer has to actually specify this reward. This is almost always an iterative process, where the reward specification evolves over time as the designer tests the agent in more and more environments.

Take, for instance, autonomous cars. They have to correctly evaluate and balance safety, efficiency, comfort and abiding by the law. An engineer might start by looking at some representative environments, and specifying a reward function that leads to the behavior they want in each of them. But working well in this set of environments is not enough — the reward function has to incentivize the right behavior in *any* environment the car will

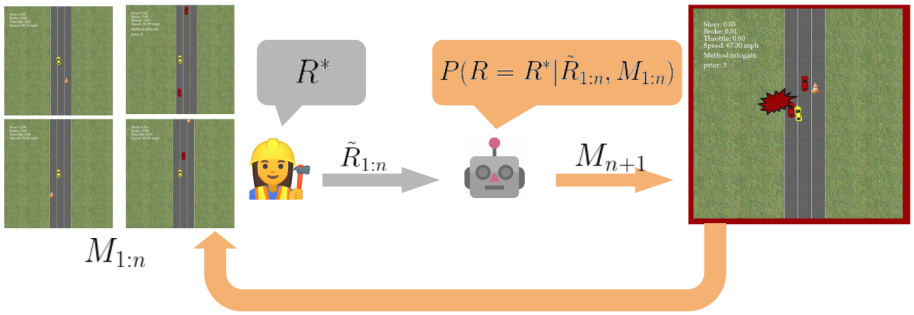

Figure 1: Assisted Reward Design Process. The designer specifies a proxy reward on a set of environments for highway merging. Our algorithm takes the current design, and queries the designer with a new environment.

encounter in its lifetime[1]. The engineer will test the car in further environments, maybe in simulation, or maybe by test-driving in the real world. Almost inevitably at some point, optimizing for the same reward will lead to some undesirable behavior. The engineer will then *revise* the initial reward, and repeat the process.

An AI system should account for this *iterative* nature of reward design. We argue that apart from simply maximizing the specified reward, the AI should also help the engineer figure out what the reward is — we refer to as *Assisted Reward Design*.

To that end, we use a model in which the AI has *uncertainty* about the reward function — it is no longer set in stone as in the current formulations. Every revision of the reward by the designer does *not* define the reward function fully, but is instead an *evidence* about it. The AI's ultimate goal is to do well with respect to the true reward at deployment time.

How does this formulation lead to design assistance? A naive agent would passively estimate the reward from observations so far. An assistive agent, in contrast, accounts for future iterations of the reward as *future observations*. The agent can then act to make these observations more informative, by exposing the designer to environments where revisions are most likely needed. Rather than letting the designer eventually encounter edge-case environments, the agent actively proposes them and asks for reward revisions.

We test our framework on an autonomous driving task in simulation with ground truth rewards, and in a case study with end users. We find that it leads to improvements in behavior on difficult test scenarios more quickly than the passive baseline, and that the environments it produces are ones in which the current reward estimate fails.

Overall, we are excited to contribute a framework in which AI systems do not just optimize what we design for them, but actively help us design better and more robust reward functions. Our work fits broadly in reward learning (Ng and Russell, 2000; Abbeel and Ng, 2004; Ramachandran and Amir, 2007; Bajcsy et al., 2017; Ziebart et al., 2008; Ratliff et al., 2006; Levine and Koltun, 2012), and is related to methods that actively query for demonstrations (Lopes et al., 2009; Brown et al., 2019), comparisons (Christiano et al., 2017; Sadigh et al., 2017), advice (Odom and Natarajan, 2015), or scalar feedback (Reddy et al., 2019). Our focus is on expert users that specify reward functions. Our finding is that by keeping a belief over reward functions, agents can identify edge-cases where the currently probable rewards contradict, and help the expert iterate.

1. We assume that everything else such as world model, planning algorithms, etc. work out well for the agent. In this paper we focus only on the reward.

2. Assisted Reward Design

2.1 Problem Setup

Let an "environment" M be an MDP without the reward function. We assume access to a large set of such environments at development time, $\mathcal{M}_{\text{devel}}$, which designer cannot exhaustively test. This can be from a several-million mile driving dataset, a parameterized distribution of synthetic environments, or the output from a generative model of the world. The agent is deployed in a potentially different $\mathcal{M}_{\text{deploy}}$, which we do *not* have access to. [2] We further assume access to a space of reward functions parametrized by $w \in W$ — these can be linear weights on pre-defined features, or weights in a neural network that maps raw input to scalar reward. We denote by w^* the parameters of the desired reward function, which would induce the desired behavior in $\mathcal{M}_{\text{devel}}$ and $\mathcal{M}_{\text{deploy}}$. We denote by $\xi_{w,M} = \arg\max_\xi R_w(\xi; M)$ the optimal trajectory (or policy, but for the purpose of this work we consider deterministic MDPs with set initial states) induced by w. Therefore, our assumption is that there exists a w^* such that $\xi_{w^*,M}$ is the desired behavior for any $M \in \mathcal{M}_{\text{devel}}$ and any $M \in \mathcal{M}_{\text{deploy}}$. We do *not* have access to w^*.

We first detail how unassisted reward design works, then define the assisted reward design problem and introduce our method.

2.2 The Process of Unassisted Reward Design.

In one-shot design, the designer takes a subset of environments $\mathcal{M}_0 \subseteq \mathcal{M}_{\text{devel}}$, assume it is representative of $\mathcal{M}_{\text{deploy}}$, design a $\tilde{w}_0$ for that $\mathcal{M}_0$, and deploy the system with $\tilde{w}_0$. Eventually they encounter during their testing or deployment a new M' on which optimizing $\tilde{w}_0$ does not lead to desirable behavior. Then, they would augment their set to $\mathcal{M}_1 = \mathcal{M}_0 \cup \{M'\}$, and re-design: $\mathcal{M}_0 \to \tilde{w}_0 \to \mathcal{M}_1 = \mathcal{M}_0 \cup \{M'\} \to \tilde{w}_1 \to ...$ Implicitly, at every step along the way, the AI agent treats the current $\tilde{w}_i$ as equivalent to w^*.

2.3 The Assisted Reward Design Problem.

In the Assisted Reward Design problem, the AI agent no longer treats proxy reward $\tilde{w}$ as equivalent to w^* — rather, it is a proxy observation of w^* based on the current $\mathcal{M}_i$. The agent interprets $\tilde{w}$ as evidence about w^*, and use all past evidence to obtain a belief of w^*. The key step is to be able to account for (and thus influence) the future evidence in order to get a more successful reward estimate. We hereby formulate the assisted reward design as a POMDP problem with state uncertainty.

State, actions, observations, transitions. The notion of a "state" is a set $\mathcal{M}$ of environments along with the hidden state w^*. The agent (and designer) starts at state $(\mathcal{M}_0, w^*)$ (either chosen by the designer, or simply the empty set). We can transition from a state to the next by "acting", i.e. adding one more environment M_i to consider, $\mathcal{M}_{i+1} = \mathcal{M}_i \cup M_i$. Every time we enter a state $(\mathcal{M}, w^*)$, we receive an observation $\tilde{w}$ about w^*.

Observation Model. We adopt the approximately optimal designer model from previous work on Inverse Reward Design (Hadfield-Menell et al., 2017), which maps the true reward

2. Our method works best when can over-parametrize environments to induce a vast set $\mathcal{M}_{\text{devel}}$ that includes everything we might see at deployment time in $\mathcal{M}_{\text{deploy}}$ — otherwise, if $\mathcal{M}_{\text{deploy}}$ is allowed to be drastically different, this still leaves the robot exposed to potential failures after deployment.

w^* and designer's training environment set $\mathcal{M}$ to a distribution over the proxy reward $\tilde{w}$.

$$P_{\text{design}}(\tilde{w}|w^*, \mathcal{M}) \propto \exp\left(\sum_{k=1}^{|\mathcal{M}|} \beta\big[R_{w^*}(\xi_{\tilde{w}_i, M_k})\big]\right) \tag{1}$$

Objective. The objective of assisted reward design is to achieve high reward at deployment time. What the robot controls is a sequence of actions, i.e. a sequence of environments it can propose. These lead to observations about w^*. At deployment time, the robot uses its belief to generate optimal trajectories, and cumulates reward according to w^*:

$$\min_{M_1,..,M_T} \mathop{\mathbb{E}}_{M\sim\mathcal{M}_{\text{deploy}}} \max_{\xi} \mathop{\mathbb{E}}_{w\sim b(M_1,..M_T)} [R_{w^*}(\xi, M)] \tag{2}$$

where $b(M_1, .., M_T)$ is the robot's belief based on its state $\mathcal{M}_T$.

Belief Distribution. At iteration i, we can compute a belief distribution w^* based on past obervations $P_i(w = w^*|\tilde{w}_i, \mathcal{M}_i)$ using Eq. (1) and Bayes Rule:

$$P_i(w = w^*|\tilde{w}_i, \mathcal{M}_i) \propto P(w) \prod_{k=1}^{|\mathcal{M}|} \frac{\exp\big[\beta R_w(\xi_{\tilde{w}_i, M_k})\big]}{\tilde{Z}_k(w)},\ \tilde{Z}_k(w) = \int_{\tilde{w}} \exp\big[\beta R_w(\xi_{\tilde{w}_i, M_k})\big] d\tilde{w} \tag{3}$$

where $P(w)$ is the prior and $\tilde{Z}$ is the normalizing constant. This posterior distribution $P_i(w = w^*)$ gives us uncertainty over true reard w^*. We next introduce how the AI agent can take actions to influence of this uncertainty.

2.4 Approximate Solution via Information Gain.

We use a proxy objective: disambiguate the reward as much as possible; ideally, if we identify w^* exactly, the robot attains 0 regret at deployment. Thus, we can employ a greedy strategy by selecting the action (M) that leads to the most information gain:

$$f(M) = \mathbb{H}[w|(\mathcal{M}_i, w_i)] - \mathbb{E}_{\tilde{w}_{i+1}\sim P(w^*)} \mathbb{H}[w|(\mathcal{M}_{i+1}, \tilde{w}_{i+1})] \tag{4}$$

Here $\mathbb{H}[w]$ is the entropy of posterior distribution of $P(w = w^*)$ and $\mathbb{H}[w|(\mathcal{M}_{i+1}, \tilde{w}_{i+1})]$ is the conditional entropy after the new observation. This heuristic is common in active learning (Houlsby et al., 2011; Gal et al., 2017) and robot active exploration (Burgard et al., 1997). The resulting query environments have high probability of narrowing down our uncertainty over w^*: for instance, an environment where the autonomous vehicle is uncertain whether it should overtake the nearby car or not. In practice to limit the AI's action space, we compute Eq. (4) on a candidate set $\mathcal{M}_{\text{cand}} \subseteq \mathcal{M}_{\text{devel}}$ where $|\mathcal{M}_{\text{cand}}| \ll |\mathcal{M}_{\text{devel}}|$.

3. Experiments on Highway Merging for Autonomous Vehicles

We evaluate Assisted Reward Design on (1) a simulation experiment where we assume that there exists a ground truth reward function and (2) a human case study where the subjective reward fuctions do not have explicit forms. We compare our information-based acquisition function with a random baseline and a domain-specific metric.

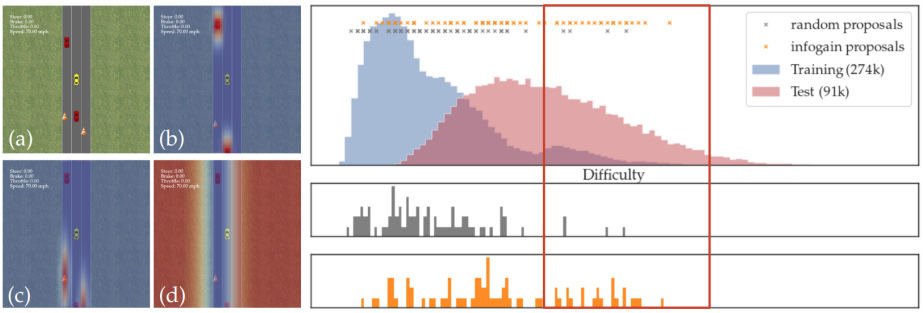

Figure 2: Left: (a) A vehicle (yellow) needs to merge to the left lane facing other human-driving vehicles (red) and traffic cones. (b)(c)(d) highlight three environment features. Right: distribution of $\mathcal{M}_{\text{devel}}$ (blue) and $\mathcal{M}_{\text{deploy}}$ (pink) based on the difficulty metric. The test distribution noticeably contains a higher concentration of environments with dense interactions.

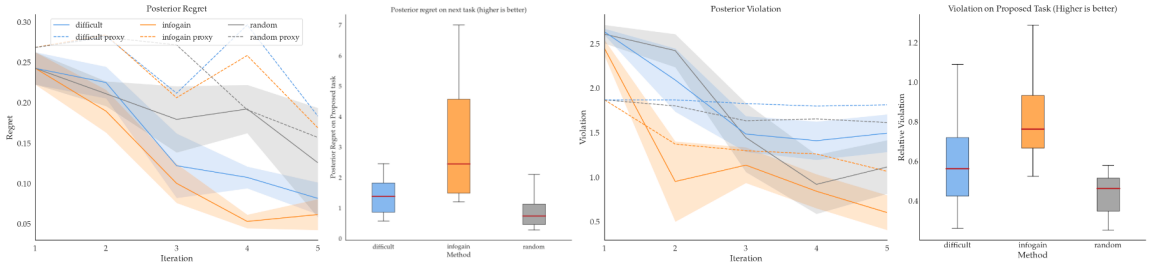

Figure 3: Left: simulation experiment, Right: case study. We visualize the regret compared to true reward under different acquisition functions in (a), violations in (c) and the efficacy of the proposed environment in (b)(d). The "box" denotes 5th to 95th percentile

Driving Environment. The goal of the car illustrated in Fig. 2 is to merge to the left lane on the 3-lane highway, where there 2 constant-speed human vehicles and 2 traffic cones. The autonomous car needs to decide whether to overtake or to abide by safety and slow down. Our cost function consists of 11 features, including distances, speed, control effort, etc. While real world systems with perception can often contain much larger feature sets, we find that it's challenging enough to hand-design robust reward functions on our 11 features.

Environment Distribution Merging tends to be more difficult in crowded environments. We can use a difficulty metric to describe crowdedness: $\sum_i \frac{1}{d_{\text{human vehicle }_i}} + \sum_j \frac{1}{d_{\text{traffic cone}_j}}$. Fig. 2 visualizes $\mathcal{M}_{\text{devel}}$ and $\mathcal{M}_{\text{deploy}}$ under this metric. Note that both $\mathcal{M}_{\text{devel}}$ and $\mathcal{M}_{\text{deploy}}$ have a "long-tail" distribution of events, with $\mathcal{M}_{\text{deploy}}$ being more difficult by design. An ideal acquisition function to identify the rare events that are informative for reward design.

3.1 Simulation Experiment

A good assisted reward design system should help the reward designer quickly recover high-quality reward functions. To evaluate this, we specify one set of w^* as the ground truth reward, and simulate the designer using w^* and Eq. (1). Our active method based on Maximum Information outperforms random baseline (grey) and hand-crafted domain-specific difficulty metric (blue) in Fig. 3.

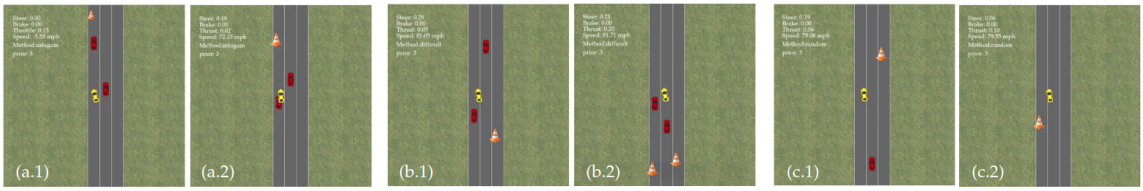

Figure 4: Visualization of the top 2 proposed environments based on each acquisition function.

Edge-Case Nature of Proposed Environments To study why our acquisition performs better than the others, we measure the quality of the proposed environments using:

$$r(M_{\text{next}}) = \frac{\mathbb{E}_{\tilde{w}\sim P(w=w^*)}\left[\text{Regret}(\tilde{w}; w^*, M_{\text{next}})\right]}{\mathbb{E}_{\tilde{w}\sim P(w=w^*)}\mathbb{E}_{\mathcal{M}\sim M_{\text{deploy}}}\left[\text{Regret}(\tilde{w}; w^*, M)\right]} \tag{5}$$

This computes the ratio of regret on next environment versus average regret on test environments. The higher $r(M_{\text{next}})$ is, the more the next environment uncovers the overall regret of the current posterior. As visualized in Fig. 3, Maximum Information proposes environments of highest value. This suggests that the to find the most useful environment for Assisted Reward Design, simply relying on heuristic environment ranking is not enough. It is much more effective to utilize all proxy rewards and their induced uncertainty over w^*.

3.2 Case Study with Human Subjects

We evaluate our algorithm on the autonomous driving task where we use it to design a reward function that matches our internal desired criteria. We answer the queries stemming from the our algorithm, as well as the baselines. To remove our bias, we blind ourselves to which algorithm produced the query.

Results Since we cannot explicitly write down the reward function and compute posterior regret as in Section 3.1, we introduce a set of violation criteria (collisions, driving offtrack, stopping, driving over speed, etc) as "unit tests" on the behavior. [3] Maximal Information outperforms the other acquisition functions as shown in Fig. 3.

Edge-Case Nature of Proposed Environments We measure the quality of the proposed environments at each iteration similar to Eq. (5). We use $\text{Violate}(\tilde{w}; M_{\text{next}})$ instead of $\text{Regret}(\tilde{w}; w^*, M_{\text{next}})$. As visualized in Fig. 2, Maximum Information outperforms the heuristic difficulty metric in finding environments that contains more violations and leads to overall better reward designs.

Qualitative Analysis We random sample 3 initial environments and query designer proxies $\tilde{w}$ on each of them. We then select the top 2 proposed environments and visualize the trajectory of MAP estimate w_{MAP} in them. Results are shown in Fig. 4. Notice that interestingly, Maximum Information proposed environments where the current MAP estimate fails. Though we do not explicitly optimize for finding failure cases, these environments help reward designers effective narrow down on the true reward. In comparison, heuristic difficulty finds environments that do not induce failure in the resulting trajectories.

3. Although we could incorporate these constraints in the "reward" function, we keep the reward features oblivious to them to demonstrate that Assisted Reward Design can help us improve design quality.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. Learning robot objectives from physical human interaction. *Conference on Robot Learning (CoRL)*, 2017.

Daniel S. Brown, Yuchen Cui, and Scott Niekum. Risk-aware active inverse reinforcement learning. *CoRR*, abs/1901.02161, 2019. URL `http://arxiv.org/abs/1901.02161`.

Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Active mobile robot localization. Citeseer, 1997.

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 4299–4307, 2017.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org, 2017.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in neural information processing systems*, pages 6765–6774, 2017.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*, 2012.

Manuel Lopes, Francisco Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 31–46. Springer, 2009.

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

Phillip Odom and Sriraam Natarajan. Active advice seeking for inverse reinforcement learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

Siddharth Reddy, Anca D. Dragan, Sergey Levine, Shane Legg, and Jan Leike. Learning human objectives by evaluating hypothetical behavior, 2019.

Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.