Cooperative Reinforcement Learning

Lawrence Chan *

December 11, 2017

Abstract

We propose a new framework for studying value alignment problems, Cooperative Reinforcement Learning (CRL), which improves upon existing frameworks by relaxing assumptions about full introspective access to the human's reward function. A CRL game is a cooperative, partial-information game with two agents, human and robot; both are rewarded according to a reward function unknown to both, and only the human observes the rewards. We show that solving CRL problems can be reduced to solving a POMDP, then propose general CRL game solvers. We demonstrate that our solutions to the CRL problems allow the robot to outperform non-intervention in cases where the human is suboptimal, even when the robot receives no reward signal. In doing so, our solutions learn behaviors such as helping an underexploring human explore or accounting for noise in the human policy.

1 Introduction

Autonomous agents are becoming more capable of optimizing the reward functions given to them in a wide variety of domains, from Go [1] to translation [2]. However, specifying reward functions so that we get what we want can be difficult, and misstating our objectives can lead to disaster, especially as these agents become increasingly good at optimizing for their given rewards. For example, Amodei et al. [3] give the example of a domestic robot rewarded for moving a box; when deployed, this robot knocks over a vase that happens to be in its way. This problem is that of *value alignment* - getting an autonomous agent to optimize for what we care about. A solution to this problem will both have short term implications for the design of usable AI systems such as autonomous vehicles and robotic assistants, as well as long term implications for powerful AI and its relationship to humanity [4].

The field of *Inverse Reinforcement Learning* (IRL) [5, 10] seeks to solve a related problem: given a set of behaviors undertaken by an agent, deduce the reward signal that the agent was trying to maximize. We can hope that applying solutions to the IRL problem to humans will allow robots to learn an appropriately robust objective function. In cases where we can demonstrate the desired behavior directly, we can use imitation learning to train a robot policy to perform well [7], though this approach does not scale to achieving better-than-human behavior.

Unfortunately, existing IRL algorithms tend to require the assumption that the human behavior is near optimal at optimizing for the human's utility function. This is undesirable in two ways. First, this precludes a variety of useful teaching behaviors. For example, when teaching a robot to drive a car, watching a human drive is an inefficient way to learn how to drive. The human should perhaps deliberately demonstrate how each of the controls affects the car's behavior and what to do during a hurricane alert; the robot should perhaps try driving slowly and ask for human feedback. Though none of this behavior can be naturally phrased in the IRL framework,

*This senior thesis was supervised by Professor Val Tannen.

the *Cooperative Inverse Reinforcement Learning* (CIRL) framework [8] is able to capture these behaviors.

However, both the traditional IRL and CIRL frameworks fail to naturally capture another important part of the human-robot interaction - namely, that the human does not necessarily know their reward function ahead of time, a point well emphasized by the literature on human decision making [9]. This motivates us to propose a new framework for discussing human-robot interaction and value alignment problems.

1.1 Our Contribution

We propose a novel framework for discussing human-robot interaction and value alignment problems - *Cooperative Reinforcement Learning* (CRL) - where the human is not considered to have introspective access to their reward function. In addition, we propose several algorithms for solving the robot's part of the problem for problems in the framework, and reduce a general version of the problem to a partially observable Markov decision process (POMDP). We then propose several solution methods for solving CRL games from the robot's perspective. Finally, we use one of our solution methods to study the robot policies in a simple human-robot interaction games. Our learned policies demonstrate interesting behavior such as querying the human in addition to outperforming non-intervention for sub-optimal human policies despite never observing the reward signal.

2 Prior Work

In addition to the cooperative inverse reinforcement learning (CIRL) framework which this model extends, the model proposed in this paper is related to a wide variety of existing work, which we can divide into several categories: inverse reinforcement learning, reinforcement learning via human preference elicitation, and principal agent models and mechanism design.

Inverse Reinforcement Learning Algorithms for *inverse reinforcement learning*, or IRL, seek to infer the reward function of the behavior of an agent, which is assumed to be (approximately) optimal [10, 6]. Of particular interest is the wide variety of expert policies assumed in the literature, from the optimal expected reward-maximizing behavior [10, 6] to "Boltzman rational", where actions are selected proportionally to the exponent of their value [11, 12]. In addition, several algorithms for IRL also place a prior on reward functions, either directly [11] or indirectly [12]. However, the IRL algorithms known to the author all assume that the agent has (at least approximate) introspective access to their reward function, which may not be the case for humans exploring novel domains. Our framework naturally accounts for this situation.

Reinforcement Learning via Preference Elicitation Another related area of research is reinforcement learning from human ratings or rankings, on which a substantial amount of work has been done [13, 14, 15, 16]. Of particular interest is Christiano et al.'s work extending existing approaches to modern problems in deep reinforcement learning [17]. In their work, they elicit preferences from humans in terms of comparisons between two agent trajectories, and demonstrate that the results of these comparisons can be used to train neural network policies to accomplish complicated tasks. This can be seen as an instance of our framework where the human actions consist of choices between two trajectories.

Principal-Agent Models and Mechanism Design Of course, the problem of value alignment is not unique to artificial agents. Principal-agent problems in business and economics [] can be seen as the human analogue of value-alignment. More generally, the field of mechanism

design studies the problem of creating incentives to align interests of different agents and improve welfare.
However, as Hadfield-Menell et al note, there exist significant distinctions between the models studied in principal-agent or mechanism design problems, and the task of value alignment [8]. Notably, unlike in the case of human agents with different pre-existing desires, there need not be an intrinsic value mismatch between the goals of an artificial intelligence and the goals of its designers [4]. The task of value alignment is not so much to subordinate autonomous agents with arbitrary goals to human interests, but to build agents who share our goals.

3 Cooperative Reinforcement Learning

A Cooperative Reinforcement Learning (CRL) game is a two-player Markov game with identical payoffs between a human $\mathbf{H}$ and a robot $\mathbf{R}$. The game is described by a tuple

$$M \;=\; \langle \mathcal{S}, \{\mathcal{A}^{\mathbf{H}}, \mathcal{A}^{\mathbf{R}}\}, T(\cdot|\cdot,\cdot,\cdot), \{\Theta, R(\cdot,\cdot,\cdot;\cdot,\cdot)\}, P_0(\cdot,\cdot), \gamma \rangle,$$

where:

- $\mathcal{S}$ is a set of world states: $s \in \mathcal{S}$.
- $\mathcal{A}^{\mathbf{H}}$ is a set of actions for $\mathbf{H}$: $a^{\mathbf{H}} \in \mathcal{A}^{\mathbf{H}}$.
- $\mathcal{A}^{\mathbf{R}}$ is a set of actions for $\mathbf{R}$: $a^{\mathbf{R}} \in \mathcal{A}^{\mathbf{R}}$.
- $T(\cdot|\cdot,\cdot,\cdot)$ is a conditional distribution on the next world state, given previous state and action for both agents: $T(s'|s, a^{\mathbf{H}}, a^{\mathbf{R}})$.
- Θ is a set of possible static reward parameters, observed by neither $\mathbf{H}$ nor $\mathbf{R}$.
- $R(\cdot,\cdot,\cdot;\cdot)$ is a parameterized reward function (possibly stochastic) that maps world states, joint actions, and reward parameters to real numbers. $R : \mathcal{S} \times \mathcal{A}^{\mathbf{H}} \times \mathcal{A}^{\mathbf{R}} \times \Theta \to \mathbb{R}$.
- $P_0(\cdot,\cdot)$ is a distribution over the initial state, represented as tuples: $P_0(s_0, \theta)$
- γ is a discount factor: $\gamma \in [0, 1]$.

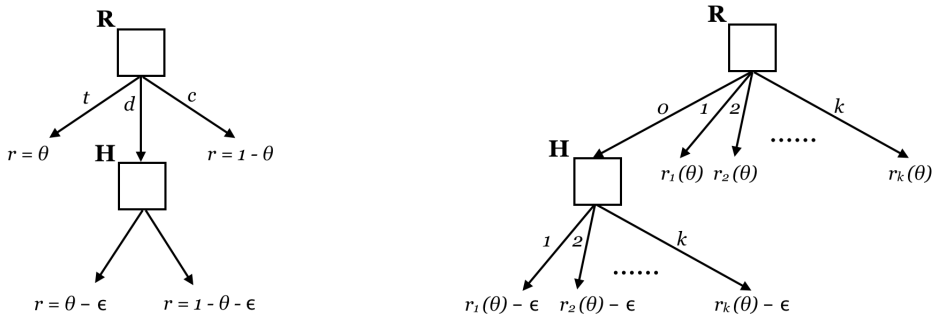
As before, the structure of the game is known to both $\mathbf{H}$ and $\mathbf{R}$. At the beginning of the game, a tuple $(s_0, \theta) \in \mathcal{S} \times \Theta$ is sampled from P_0. Unlike the CIRL game, in the CRL game $\mathbf{H}$ is not aware of the reward parameters. After both $\mathbf{H}$ and $\mathbf{R}$ observe the current state s_t and their actions $a^{\mathbf{H}}$, $a^{\mathbf{R}}$, both actors receive reward $R_t = R(s_t, a^{\mathbf{H}}, a^{\mathbf{R}}; \theta)$ and observe each other's actions. In addition, H observes R_t. Finally, a state s_{t+1} is sampled from the transition distribution $T(s'|s, a^{\mathbf{H}}, a^{\mathbf{R}})$, and the process repeats.

The objects of interest in this formulation are the policies $\pi^{\mathbf{H}}$ and $\pi^{\mathbf{R}}$. In general, these policies can be arbitrary functions of their observation histories: $\pi^{\mathbf{H}} : \left[\mathcal{A}^{\mathbf{H}} \times \mathcal{A}^{\mathbf{R}} \times \mathcal{S}\right]^* \to \mathcal{A}^{\mathbf{H}}$, $\pi^{\mathbf{R}} : \left[\mathcal{A}^{\mathbf{H}} \times \mathcal{A}^{\mathbf{R}} \times \mathcal{S}\right]^* \to \mathcal{A}^{\mathbf{R}}$. We would like to find a policies that maximize the expected sum of discounted rewards under the initial distribution P_0 of reward parameters and world states.

Note that this formulation is extremely broad: for example, it can capture games where the human and robot act sequentially. The main difference between a CRL game and a CIRL game [8] is that $\mathbf{H}$ does not necessarily the true value of the reward parameters when $|\theta| > 1$, but instead must learn the reward function through experience, perhaps using methods from the reinforcement-learning literature such as Sarsa or Q-learning. CIRL games can also be thought of as special cases of CRL games where the first reward signal to the human consists of a specification of θ.

3.1 Examples

We offer several examples of CRL games, which illustrate some interesting properties of the CRL framework.



(a) The tea or coffee game

(b) The collaborative bandit game

Figure 1: The basic structure of the tea or coffee game and its generalization, the collaborative bandit game. Square nodes indicate choice nodes for the human **H** or the robot **R**. In each round of the tea or coffee game, the robot **R** can either choose to make tea (t) or coffee (c), or it can defer the choice to the human **H**. In each unit of the collaborative bandit game, the **R** can either choose to pull one of the K arms in the bandit, or defer the choice of arm to the human **R**. The choices are made repeatedly over some number of rounds, and the task is to maximize the expected sum of discounted rewards.

Example 1 (Tea or Coffee) At the start of every day, the human **H** would like to consume a single caffeinated beverage, either coffee or tea. Before the human wakes up, the robot can prepare one of the two caffeinated beverages or choose not to, if the robot does not, then the human will prepare one of the two beverages. The human has an unknown but fixed preference toward either coffee or tea, represented by $\Theta = \{0, 1\}$. The human and the robot share a uniform prior over Θ. The human gets θ reward for consuming tea and $1 - \theta$ reward for consuming coffee, which she will observe after consuming either the tea or coffee. In addition, the act of preparing coffee or tea costs the human $\epsilon = 0.1$ points of reward, but if the tea or coffee was already prepared by the robot **R**, then no such penalty is incurred.

If the human knew the true value of θ in the above example, then she can maximize reward by preparing tea when $\theta = 1$ and coffee when $\theta = 0$. The best policy for the robot would then be wait one day to see what beverage the human chooses, then make that beverage every day after that.

However, if the robot performs this policy when the human does not know the true value of θ a priori, then this policy will lead to an expected reward of 0.5 a day, lower than the 0.9 that the human can achieve if the robot always takes no action. Instead, the optimal policy for the robot would be to perpare a drink for a human - to help the human to learn the value of θ - and then let the human prepare a drink so the human can signal the value to the robot. This policy has expected reward 1 a day for each day after the second.

Interestingly, **R** can help **H** perform better even if the cost ϵ of making beverages for the human is 0, if **H** is not following the optimal (reward-maximizing) policy, as the following example demonstrates:

Example 2 (Underexploring Tea or Coffee). Suppose that the human and robot are playing the game from example 1 except that the cost of making beverages for the human $\epsilon = 0$, and that the human **H** is a Q-learner with initial value -1 and discount factor $\lambda < 1$ that follows a greedy policy for choosing tea or coffee. That is, after taking **H** makes tea with probability

1 if **H**'s Q-value for tea is higher, and vice versa in the case where her Q-value for coffee is higher (in the case where her Q-values are equal, she chooses one at random). No matter what beverage she samples randomly in the first timestep, she will end up always make that beverage afterwards - so half the time her average reward per episode is 0! However, if the **R** makes both beverages, **H** will "learn" which beverage she likes better, and thus have average reward per episode approaching 1.

Remark 1. Both of the examples above are quite unrealistic, but they illustrate key points about the formulation: that *the optimal policy when interacting with a human without perfect knowledge about their own preferences can different from one where the human has perfect knowledge about preferences*, and that *the optimal policy of the robot depends on our model of human behavior.*

Notice that the environment above is essentially a collaborative multi-armed bandit problem. This motivates the following example, which generalizes the environment:

Example 3 (Collaborative Bandits) The human **H** and the robot **R** are playing a multi-armed bandit with K arms. The distribution over the parameter space Θ of the bandit is known to both **H** and the **R**, as is the reward distribution $r_k(\theta)$ given the parameters of the Bandit θ and that the k th arm is pulled, though neither know the exact value of θ at the start of the game. We suppose that at each timestep, the robot first takes an action choose either to pull one of the arms or to allow the human the choose an arm to pull. If the robot allows the human to pull the arm, then it observes which of the arms the human chose, otherwise, it does not. To incentivize the robot to act, there is a small cost $\epsilon \geq 0$ to querying the human.

We note that this example is extremely general, and can be seen as including problems such as helping a customer order dishes at a restaurant, assisting a office worker in learning a new piece of software, and collaborating with a human controller in choosing routes for an autonomous vehicle.

3.2 Reduction to POMDP

As in the CIRL case, we can consider the problem from both the human and the robot's perspectives [8], which allows us to reduce the problem to a general decentralized POMDP (Dec-POMDP)[18]. (Note that the reduction proposed used to reduce CIRL games to a coordination-POMDP does not work, as the robot must maintain a belief state over the human's belief state over the rewards.) However, as solving Dec-POMDPs is a NEXP-hard problem [18], studying the optimal policies in this setting would require a doubly-exponential time algorithm. Instead, we can choose to restrict our attention to human policies with fixed memory:

Theorem 1. Let M be a CRL game. If we restrict the space of human policies $\pi^{\mathbf{H}}$ to those the form $\pi^{\mathbf{H}} : \mathcal{S} \times \mathcal{A}^{\mathbf{R}} \times \mathcal{A}^{\mathbf{H}} \times \Phi \times img(R) \to \mathcal{A}^{\mathbf{H}} \times \Phi$, where Φ represents the internal memory of the human, and with initial memory distribution $P_1(\cdot)$, then we can reduce the CRL game to a (single-actor) POMDP M_C of with state space S_C of size $|\mathcal{S}| \cdot |\Theta| \cdot |\Phi|$, such that for any policy pair in M, there is a policy in M_C that achieves the same sum of discounted rewards. Likewise, for any policy in M_C, there exists a pair of policies in M that achieves the same sum of discounted rewards.

Proof. Let M_C be the POMDP where the single actor is a coordinator **C**, where states are tuples of world state, reward parameters, and **H** 's internal memory: $(s, \theta, \phi) \in \mathcal{S}_C = \mathcal{S} \times \Theta \times \Phi$. The first two components of the initial state tuple are sampled from P_0, while **H**'s internal memory is sampled from P_1. **C**'s actions are tuples $(\delta^{\mathbf{H}}, a^{\mathbf{R}})$, where $\delta^{\mathbf{H}}$ is a decision rule for **H** that maps

its internal state (ϕ) and most recently observed reward to an action and internal state update: $\delta^{\mathbf{H}} : \Phi \times R \to \mathcal{A}^{\mathbf{H}} \times \Phi$ and $a^{\mathbf{R}}$ is an action for $\mathbf{R}$. As in M, observations for $\mathbf{C}$ in M_C consist of tuples of world state and human and robot actions: $(s, a^{\mathbf{H}}, a^{\mathbf{R}}) \in \mathcal{S} \times \mathcal{A}^{\mathbf{H}} \times \mathcal{A}^{\mathbf{R}}$. Transitions in M_C correspond to transitions and updates of $\mathbf{H}$'s belief state in M after $\mathbf{H}$ takes action (and updates their belief state according to) $\delta^{\mathbf{H}}(\Phi)$ and $\mathbf{R}$ takes action $a^{\mathbf{R}}$.

Note that $\mathbf{R}$ has no private observation, and so for any policy $\pi^{\mathbf{R}}$ $\mathbf{R}$ could choose to follow, $\mathbf{C}$ can match it by simulating $\pi^{\mathbf{R}}$ and outputting the corresponding action. Similarly, for any policy $\pi^{\mathbf{H}}$ that $\mathbf{H}$ can implement, $\mathbf{C}$ can match it by simulating $\pi^{\mathbf{H}}$ and outputting a representation of the decision rule implemented by $\pi^{\mathbf{H}}$. To see that there is a $\mathbf{C}$ policy that can reproduce the behavior of any $\pi^{\mathbf{H}}$, we can simply choose the decision rule:

$$\delta^{H}(\phi, r) = \pi^{\mathbf{H}}(s, a^{\mathbf{H}}, a^{\mathbf{R}}, r, \phi)$$

to produce the same behavior.

Likewise, since $\mathbf{C}$ only observes common observations, any coordinator strategy can be implemented by $\mathbf{H}$ and $\mathbf{R}$ each simulating $\mathbf{C}$ and directly executing the appropriate decision rules and actions. □

Remark 1. Though in general the size of Φ may be very large (up to $|\mathcal{A}^{\mathbf{H}}|^t \cdot |\mathcal{A}^{\mathbf{R}}|^t$, the number of possible histories of robot and human actions, for a game with t timesteps), in practice it may be appropriate to assume that the amount of memory available to a human (and thus the state space of the POMDP) will be much smaller than this.

Corollary 1. Let M be a CIRL game. Then if we restrict the space of human policies to the form in theorem 1, there exists optimal policies $(\pi^{\mathbf{H}^*}, \pi^{\mathbf{R}^*})$ that depend only on the current state, last actions, last observed reward, and $\mathbf{R}$'s beliefs:

$$\pi^{\mathbf{H}^*} : \mathcal{S} \times \mathcal{A}^{\mathbf{H}} \times \mathcal{A}^{\mathbf{R}} \times img(R) \times \Phi \to \mathcal{A}^{\mathbf{H}} \times \Phi, \qquad \pi^{\mathbf{R}^*} : \mathcal{S} \times \Delta_\theta \times \Delta_\phi \to \mathcal{A}^{\mathbf{R}}.$$

Proof. The optimal policy in a POMDP depends only on the belief state [19]. As $\mathbf{C}$'s belief state in M_C is uniquely determined by $\mathbf{R}$'s belief state over $\Theta \times \Phi$, the corollary follows immediately from theorem 1. □

In practice, it is more interesting to consider the problem solely from the perspective of the robot given fixed policies from human (or a distribution over a fixed set of policies), as in [8], as it would be unrealistic to model the human as optimally coordinating with an optimal robot to maximize their reward. This allows us to construct a natural reduction to a POMDP with a much smaller action space:

Theorem 2. For a given CRL game M with a fixed human policy $\pi^{\mathbf{H}} : \mathcal{S} \times \mathcal{A}^{\mathbf{H}} \times \mathcal{A}^{\mathbf{R}} \times img(R) \times \Phi \to \mathcal{A}^{\mathbf{H}} \times \Phi$ and initial human state ϕ sampled from $P_1(\cdot)$ (both of which are known to $\mathbf{R}$), the task of finding the optimal robot policy can be reduced to solving a POMDP $M_{\mathbf{R}}$ with state space of size $|\mathcal{S}| \cdot |\Theta| \cdot |\Phi|$ and action space $\mathcal{A}^{\mathbf{R}}$ such that for each robot policy in M, there exists a policy in $M_{\mathbf{R}}$ that achieves an equal sum of discounted rewards, and vice versa.

Proof. Let $M_{\mathbf{R}}$ be the POMDP with the single actor $\mathbf{R}$, where states are tuples of world state, $\mathbf{H}$'s last observed reward, and $\mathbf{R}$'s internal memory: $(s, \theta, \phi) \in \mathcal{S}_{\mathbf{R}} = \mathcal{S} \times \Theta \times \Phi$. As before, first two component of the initial state tuple is sampled from P_0, while the second component is sampled from P_1.The action space in $M_{\mathbf{R}}$ is simply $\mathcal{A}^{\mathbf{R}}$, while the transitions in $M_{\mathbf{R}}$ simply correspond to transitions in M after $\mathbf{H}$ takes the action and performs the belief updates determined by $\pi^{\mathbf{H}}(s, a^{\mathbf{H}}, a^{\mathbf{R}}, R(s, a^{\mathbf{H}}, a^{\mathbf{R}}; \Theta), \phi)$ and $\mathbf{R}$ takes the action $a^{\mathbf{R}}$.

Note that since $\mathbf{R}$ has the same observations in either M or $M_{\mathbf{R}}$, $\mathbf{R}$ can simply use the same policy in either environment. As the $\mathbf{H}$'s decision rule and update behavior is fixed, the M and $M_{\mathbf{R}}$ have the same human state updates and transitions. So the POMDP $M_{\mathbf{R}}$ is a reduction of the problem faced by $\mathbf{R}$ in M when the human policy is fixed and known. □

4 CRL Solvers

We propose several solvers for the CRL problem from the perspective of a robot, when the human policy is known[1].

POMDP Solvers As the task from the perspective of the robot can be reduced to solving a POMDP, we can solve CRL games by performing this reduction and then applying a POMDP solver. We used a modified version of Silver and Veness's Partially Observable Monte-Carlo Planning (POMCP) to serve as a solver, as it is able achieve high performance on very large state spaces and requires only a black box simulator. [20]. Our version of POMCP does not receive and update on the reward signal, but is otherwise identical to the original version.

Feedforward Policy Network Alternatively, in cases where a sufficient statistic of the robot's belief state is easy to calculate, we can compute this statistic and use the sufficient statistic and the world state as input to a feedforward neural network policy. The policy network can then be trained using gradient-based policy optimization methods such as Trust Region Policy Optimization [21] and Proximal Policy Optimization [22]. We will employ this technique even in cases where we do not have a sufficient statistic for the robot's belief state, this technique may still lead to good policies when we can compute a fairly informative statistic for the belief state.

Recurrent Policy Network Instead of computing a sufficient statistic manually and feeding this into a policy, we can instead train a recurrent policy that takes as input $(s, a^{\mathbf{H}}, a^{\mathbf{R}})$ tuples of observations and maintains its own state. We can also optimize this policy using the policy-optimization methods mentioned in the feedforward case [23]. This has the advantage of being more general and working even in cases where no easily calculated sufficient statistic is available, but makes the training of the network more difficult, as it must learn both to maintain a representation of a belief state and how to act on this belief state.

5 Experiments

We consider two experimental domains: the proof-of-concept Tea or Coffee as in examples 1 and 2, and the general collaborative bandit environment of example 3. Unless otherwise noted, all settings involve 15 rounds and have discount rate equal to 1. The penalty for querying the human is 0.1 per round for Tea or Coffeeand 0 in the collaborative bandit environment.

Our environments were built in the OpenAI Gym and rllab frameworks [24, 25], and we implemented our neural network policies in Tensorflow [26]. Our recurrent policy was implemented by a gated recurrent unit (GRU) network with 256 hidden states [27], while our feedforward network with three hidden layers of 256, 256, and 32 hidden units respectively. For those policies, we used Proximal Policy Optimization (PPO) [22], with a batch size of 10000 for the proof of concept environment and a batch size of 250000 for the collaborative bandit environment, to estimate the policy gradient. We trained the policies using Adam [28]. Our input to the feedforward policy network consisted of the number of each action (either to make a beverage

[1]Code for these solvers, and our experiments are available at https://github.com/Chanlaw/crl

or pull an arm) taken by the human, as well as the number of each action taken by the network. For POMCP, we used 500 state particles for the Tea or Coffee problem and 2000 state particles for the collaborative bandit problem.

5.1 Tea or Coffee Environment

In the Tea or Coffee domain, we consider the cases where the human is an (individually) optimal agent, an underexploring agent, and a noisily-rational agent. We describe each in turn:

- The optimal human agent is an agent that both correctly identifies which drink is their preferred drink after one observation, and there after only prepares their preferred drink.
- The underexploring agent starts with pessimistic estimates of the value of both the drinks, and so will stick to whatever drink they first encounter unless they are forced to drink both drinks once each.
- The noisily rational agent does identify which drink they prefer after one drink, but only makes their preferred drink with 80% probability.

We implemented all three of these agents as ε-greedy agents [29]: the optimal agent can be implemented as a greedy agent with initial drink values 0.5; the underexploring agent can be implemented as a greedy agent with initial drink values -1; and the noisily rational agent is implemented as a ε-greedy agent with initial drink values 0.5 and $\varepsilon = 0.2$.

5.2 Collaborative Bandit Environment

In the collaborative bandits domain, we considered Bernoulli multi-armed bandits with four arms, where each arm issue a reward of one with probability θ and zero with probability $1 - \theta$. In this case, we have that $\Theta = [0, 1]^4$. In addition, we suppose that the reward probabilities for the four arms are distributed independently and uniformly in $[0, 1]$.

Note that this setup allows for more efficient inference, as the posterior for reward parameter θ_k for an arm is simply the Beta distribution $\beta(s, f)$, where s is the number of times the reward was one for that arm plus one, and f is equal to the number of times the reward was zero for that arm plus one [30]. This allows us to consider a wider variety of human agents. In addition, we were able to explicitly compute the value of this fifteen-timestep bandit problem under the optimal policy using labelled real-time dynamic programming [31]: **10.25**.

We also consider several different human policies: an ε-greedy agent with $\varepsilon = 0.1$, a win-stay-lose-shift agent, a Knowledge Gradient agent, a Thompson-Sampling agent, and a Gittins-Index Policy:

- The ε-greedy agent maintains the empirical means of rewards from each of the arms, and chooses the arm with highest reward with probability $1 - \varepsilon$ and a random arm with probability ε.
- The win-stay-lose-shift (WSLS) agent pulls the same arm as the one pulled last round if the arm returned reward one and chooses another arm randomly otherwise.
- The Knowledge Gradient (KG) agent computes the "knowledge gradient" v_k^{KG} for each arm, defined to be the difference in the expected per-timestep reward of the greedy policy one in the following timestep, given the agent pulls arm k. It then chooses the action that maximizes the value

  $$E[\theta_k] + (T - t - 1)v_k^{KG}$$

  where $(T - t - 1)$ represents the number of timesteps remaining. We can think of this agent as choosing the arm that will lead to the highest expected rewards, given the (faulty) assumption the agent follows a greedy policy starting from the next round.

- The Thompson-Sampling (TS) agent [32] samples a parameter from the posterior distribution for each of the four arms, and chooses the arm with the highest associated reward parameter.
- The Gittins Index (GI) policy [33] computes the Gittins Indices for each of the arms with some discount rate, then chooses the arm with the highest index. This method has been shown to give the Bayes-optimal solution in the discounted infinite-horizon case. We follow the approximations described by Chakravorty and Mahajan in [34], and choose a discount rate that performs best empirically.

5.3 Results

We are explored three questions: Can our solution methods learn sensible policies in the Tea or Coffee environment? Can these methods offer good solutions in the Collaborative Bandit Environment? And to what extent are the solutions robust to different models of the human?

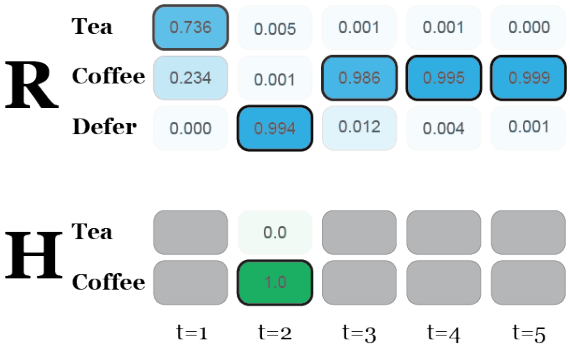
Both POMCP and our recurrent policy network were able to solve the Tea or Coffee problem without much trouble in all three cases, achieving near-optimal results. Interestingly, we were not able to get the feedforward policy to adequately solve the problem. The policies found by POMCP and through policy optimization with a recurrent policy were similar, and some interesting examples are offered and discussed in figure 2. Notably, the response to each of the human human seems to involve performing some initial exploration for the human, allowing the human to reveal their preferences, and then always making the human's preferred drink. For the optimal policy, the robot performs one round of exploration and one round of preference elicitation; for the underexploring policy, the robot performs two rounds of exploration and one round of elicitation instead; and for the nosily rational human policy, the robot performs one round of exploration and two or more rounds of elicitation.

For the collaborative Bandit environment, we were not able to achieve good results with POMCP, perhaps due to the much larger Θ. For example, though our recurrent policy was able to assist four of the human policies, it became apparent early on that POMCP was only able to improve on the WSLS policy, and in fact lead to worse performance on the other four policies. As a result, we focused most of our experimentation on the learned recurrent policy. We report the average performance of each of the human policies, with and without the assistance of the recurrent network, in Table 1. Interestingly, the best performing Gittins index policy actually matches the performance of the optimal policy found via dynamic programming without any assistance, while the knowledge-gradient policy is able to achieve near-optimal performance with the help of the appropriately trained recurrent policy.

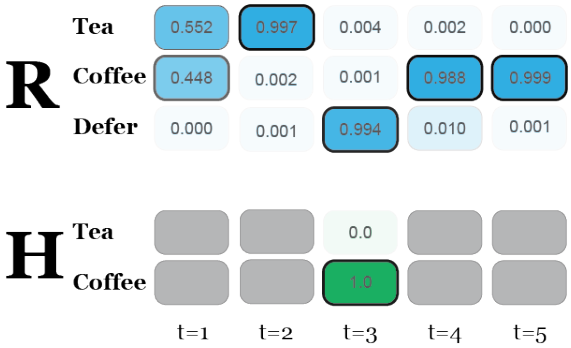
As the bandit environment also more actions than the Tea or Coffee environment, and more human policies are considered, we will not provide example trajectories. However, we also noted interesting behaviors in our learned policies; for example, when trained against the Win-Stay-Lose-Shift policy, we found that our agent learned to use the agent's reactions as a direct proxy for the reward signal. When we trained an agent against the Gittins Index policy, we found that

	ε-**greedy**	**WSLS**	**KG**	**TS**	**GI** ($\gamma = .9$)	**Optimal**
No intervention	9.19 ± 3.51	8.48 ± 4.19	10.17 ± 3.35	9.24 ± 3.04	10.25 ± 3.26	10.25 ± 3.27
Recurrent Policy	$\mathbf{9.42 \pm 3.53}$	$\mathbf{9.94 \pm 3.43}$	$\mathbf{10.25 \pm 3.36}$	$\mathbf{9.38 \pm 3.30}$	10.25 ± 3.28	N/A

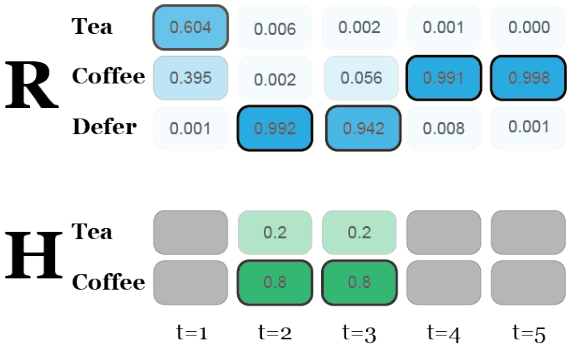
Table 1: Mean and standard deviation of cumulative rewards for different **H** policies for the collaborative bandit problem, with and without intervention from a trained recurrent policy. All results are evaluated over 100,000 games.



(a) The optimal response to the optimal human policy.



(b) Assisting an underexploring policy.



(c) Accounting for noise against a noisily rational policy.

Figure 2: Visualizations of three trajectories taken by a learned recurrent policy against each of the three human models we consider for the TeaOrCoffee game. Actions available to **R** are colored in blue, while actions available to **H** are colored in green, with actions that the agents are more likely to take in darker colors. The numbers represents the probability the policy will take the action at the given timestep. The action actually taken has a bolded border.

In each of these examples, $\theta = 0$, that is, the human has a preference for Coffee. We show only the first five timesteps of the fifteen-round interaction between **H** and **R**, as the behavior of our learned **R** on the remaining ten rounds are quite similar to its behavior on $t = 4$ and $t = 5$.

In (a), the robot is interacting with the optimal human policy. In this case, it has learned to offer the optimal response of arbitrarily choosing one of the two beverages and making them, allowing the human to reveal their preference, then making the human's preferred beverage from then on.

In (b), the robot interacts with an underexploring human policy. Here, it has learned to force the human to explore, creating both of the beverages before allowing the human to make a beverage.

In (c), the robot interacts with a nosily rational policy. Because of the noise, the robot waits for two timesteps before taking over from the human.

	Trained Against			
Tested Against	ε-**greedy**	**WSLS**	**KG**	**TS**
ε-**greedy**	$\mathbf{9.42 \pm 3.53}$	9.21 ± 3.70	9.18 ± 3.61	8.99 ± 3.94
WSLS	9.45 ± 3.43	$\mathbf{9.94 \pm 3.43}$	9.57 ± 3.49	$\mathbf{9.95 \pm 3.36}$
KG	10.13 ± 3.23	10.17 ± 3.36	$\mathbf{10.25 \pm 3.28}$	10.11 ± 3.30
TS	9.33 ± 3.07	9.36 ± 3.28	9.26 ± 3.09	$\mathbf{9.38 \pm 3.30}$
GI	10.08 ± 3.24	9.96 ± 3.44	10.23 ± 3.26	10.25 ± 3.26

Table 2: The mean and standard deviation of recurrent policies for $\mathbf{R}$ trained against the (suboptimal) human policy and tested against another human policy, for the collaborative bandit problem. All policies are evaluated over $100,000$ games. Note that in general, the Knowledge Gradient and Gittins Index policies perform decently well regardless of what recurrent policy assists it, but performance can be much worse when the robot is trained against one suboptimal policy and tested against another.

it learned the policy of almost always not intervening.

We also considered the problem of robustness to human model misspecification in table 2, where we plot the performance of each of our five human policies when assisted by recurrent policy trained against the four suboptimal human policies. (We omit the recurrent policy trained against the Gittins Index policy, which learns to never intervene.) Here, we that note that unsurprisingly, the best performing recurrent policies are generally the ones trained against the correct model, though in more than half of the cases, the resulting human-recurrent policy pair still outperforms the human policy with no intervention.

6 Conclusion and Future Work

As artificial agents become increasingly adept at optimizing for their objective functions, it becomes more important to develop frameworks and algorithms that allow us to construct agents whose reward signal is more precisely aligned with human interests. In this work, we extend existing frameworks to naturally capture the problem that humans may not know their own reward function a priori. By framing problems in this framework, we are both able to study sub-problems in value alignment and also create algorithms that allow for better collaboration with humans, even without access to the reward signal.

One potential area of work that we find exciting is the design of policy optimization algorithms that take advantage of the structure of CRL games. For example, we find it plausible that recent work in reinforcement learning with sparse rewards [35, 36] can be extended to more efficiently solve CRL games with fewer training examples or more robustness.

7 Acknowledgements

The author would like to thank Professor Val Tannen for supervising this senior thesis. In addition, we would like to thank Dylan Hadfield-Menell and Professor Anca Dragan for the inspiration for the project, and for help in understanding the CIRL framework. Finally, we would like to thank Summer Yue and Chris Painter for helpful comments.

References

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[2] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[4] Nick Bostrom. *Superintelligence: Paths, dangers, strategies*. OUP Oxford, 2014.

[5] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103. ACM, 1998.

[6] 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.

[7] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[8] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3909–3917. Curran Associates, Inc., 2016.

[9] Jonathan Baron. *Thinking and deciding*. Cambridge University Press, 2000.

[10] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[11] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51(61801):1–4, 2007.

[12] 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.

[13] Riad Akrour, Marc Schoenauer, and Michele Sebag. Preference-based policy learning. *Machine learning and knowledge discovery in databases*, pages 12–27, 2011.

[14] 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.

[15] Christian Wirth and Johannes Fürnkranz. Epmc: Every visit preference monte carlo for reinforcement learning. In *Asian Conference on Machine Learning*, pages 483–497, 2013.

[16] Layla El Asri, Bilal Piot, Matthieu Geist, Romain Laroche, and Olivier Pietquin. Score-based inverse reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 457–465. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[17] Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

[18] Daniel S Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 32–37. Morgan Kaufmann Publishers Inc., 2000.

[19] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

[20] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[21] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[24] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[25] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[26] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[27] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[28] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[30] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*, volume 2. CRC press Boca Raton, FL, 2014.

[31] Blai Bonet and Hector Geffner. Labeled rtdp: Improving the convergence of real-time dynamic programming. In *ICAPS*, volume 3, pages 12–21, 2003.

[32] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[33] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

[34] Jhelum Chakravorty and Aditya Mahajan. Multi-armed bandits, gittins index, and its calculation. *Methods and Applications of Statistics in Clinical Trials: Planning, Analysis, and Inferential Methods, Volume 2*, pages 416–435, 2014.

[35] Chelsea Finn, Tianhe Yu, Justin Fu, Pieter Abbeel, and Sergey Levine. Generalizing skills with semi-supervised reinforcement learning. *arXiv preprint arXiv:1612.00429*, 2016.

[36] Paulo Rauber, Filipe Mutz, and Juergen Schmidhuber. Hindsight policy gradients. *arXiv preprint arXiv:1711.06006*, 2017.