Machine Teaching for Inverse Reinforcement Learning: Algorithms and Applications

Daniel S. Brown and **Scott Niekum**
Department of Computer Science
University of Texas at Austin
{dsbrown,sniekum}@cs.utexas.edu

Abstract

Inverse reinforcement learning (IRL) infers a reward function from demonstrations, allowing for policy improvement and generalization. However, despite much recent interest in IRL, little work has been done to understand the minimum set of demonstrations needed to teach a specific sequential decision-making task. We formalize the problem of finding maximally informative demonstrations for IRL as a machine teaching problem where the goal is to find the minimum number of demonstrations needed to specify the reward equivalence class of the demonstrator. We extend previous work on algorithmic teaching for sequential decision-making tasks by showing a reduction to the set cover problem which enables an efficient approximation algorithm for determining the set of maximally-informative demonstrations. We apply our proposed machine teaching algorithm to two novel applications: providing a lower bound on the number of queries needed to learn a policy using active IRL and developing a novel IRL algorithm that can learn more efficiently from informative demonstrations than a standard IRL approach.

1 Introduction

As robots and digital personal assistants become more prevalent, there is growing interest in developing algorithms that allow everyday users to program or adapt these intelligent systems to accomplish sequential decision-making tasks, such as performing household chores, or carrying on a meaningful conversation. A common way to teach sequential decision-making tasks is through Learning from Demonstration (LfD) (Argall et al. 2009), in which the goal is to learn a policy from demonstrations of desired behavior. More specifically, Inverse Reinforcement Learning (IRL) (Ng and Russell 2000; Arora and Doshi 2018) is a form of LfD that aims to infer the reward function that motivated the demonstrator's behavior, allowing for reinforcement learning (Sutton and Barto 1998) and generalization to unseen states. Despite much interest in IRL, there is not a clear, agreed-upon definition of optimality in IRL, namely, the size of the minimal set of demonstrations needed to teach a sequential decision-making task.

There are many compelling reasons to study optimal teaching for IRL: (1) it gives insights into the intrinsic difficulty of teaching certain sequential decision-making tasks; (2) it provides a lower bound on the number of samples needed by active IRL algorithms (Lopes, Melo, and Montesano 2009; Brown, Cui, and Niekum 2018); (3) optimal teaching can be used to design algorithms that better leverage highly informative demonstrations which do not follow the i.i.d. assumption made by many IRL algorithms; (4) studying optimal teaching can help humans better teach robots through demonstration (Cakmak and Lopes 2012) and help robots better communicate their intentions (Huang et al. 2017); and (5) optimal teaching can give insight into how to design (Mei and Zhu 2015) and defend against (Alfeld, Zhu, and Barford 2017) demonstration poisoning attacks in order to design IRL algorithms that are robust to poor or malicious demonstrations.

We formulate the problem of optimal teaching for sequential decision making tasks using the recently popularized *machine teaching* framework (Zhu 2015). The machine teaching problem is the inverse of the machine learning problem. In machine teaching, the goal is to select the optimal training set that minimizes teaching cost, often defined as the size of the training data set, and the loss or teaching risk between the model learned by the student and the learning target. While machine teaching has been applied to regression and classification (Zhu 2015; Liu and Zhu 2016), little work has addressed machine teaching for sequential decision-making tasks such as learning from demonstration via IRL.

The contributions of this paper are fourfold: (1) a formal definition of machine teaching for IRL, (2) an efficient algorithm to compute optimal teaching demonstrations for IRL, (3) an application of machine teaching to find the lower bound on the number of queries needed to learn a task using active IRL, and (4) a novel Bayesian IRL algorithm that learns more efficiently from informative demonstrations than a standard IRL approach by leveraging the non-i.i.d. nature of highly informative demonstrations from a teacher.

2 Related work

Determining the minimum number of demonstrations needed to teach a task falls under the fields of Algorithmic Teaching (Goldman and Kearns 1995; Balbach and Zeugmann 2009) and Machine Teaching (Zhu 2015; Zhu et al. 2018). However, almost all previous work has been limited to optimal teaching for classification and regression tasks. The work of Singla et al. (2014) bears a strong resemblance to our work: they

use submodularity to find an efficient approximation algorithm for an optimal teaching problem that has a set-cover reduction; however, their approach is designed for binary classification rather than sequential decision making.

Cakmak and Lopes (2012) examined the problem of giving maximally informative demonstrations to teach a sequential decision-making task; however, as we discuss in Section 5, their algorithm often underestimates the minimum number of demonstrations needed to teach a task. Other related approaches examine how a robot can give informative demonstrations to a human (Huang et al. 2017), or formalize optimal teaching as a cooperative two-player Markov game (Hadfield-Menell et al. 2016); however, neither approach addresses the machine teaching problem of finding the minimum number of demonstrations needed to teach a task.

Our proposed machine teaching algorithm leverages the notion of behavioral equivalence classes over reward functions to achieve an efficient approximation algorithm. Zhang et al. (2009) also use behavioral equivalence classes over reward functions as part of their solution to a policy teaching problem, in which the goal is to induce a desired policy by modifying the intrinsic reward of an agent through incentives. Rathnasabapathy et al. (2006) and Zeng et al. (2012) use equivalence classes over agent behaviors when solving the problem of interacting with multiple agents in a POMDP.

There is a large body of work on using active learning for IRL (Lopes, Melo, and Montesano 2009; Cohn, Durfee, and Singh 2011; Cui and Niekum 2017; Sadigh et al. 2017; Brown, Cui, and Niekum 2018). Our goal of finding a minimal set of demonstrations to teach an IRL agent is related to one of the goals of active learning: reducing the number of examples needed to learn a concept (Settles 2012). In active learning, the agent requests labeled examples to search for the correct hypothesis. Optimal teaching is usually more sample efficient than active learning since the teacher gets to pick maximally informative examples to teach the target concept to the learner (Zhu et al. 2018). Thus, a solution to the machine teaching problem for IRL provides a method for finding the lower bound on the number of queries needed to learn a policy when using active IRL.

In the field of Cognitive Science, researchers have investigated Bayesian models of informative human teaching and the inferences human students make when they know they are being taught (Shafto and Goodman 2008). Ho et al. (Ho et al. 2016) showed that humans give different demonstrations when performing a sequential decision making task, depending on whether they are teaching or simply doing the task. While studies have shown that standard IRL algorithms can benefit from informative demonstrations (Cakmak and Lopes 2012; Ho et al. 2016), to the best of our knowledge, no IRL algorithms exist that can explicitly leverage the informative nature of such demonstrations. In Section 7.2 we propose a novel IRL algorithm that can learn more efficiently from informative demonstrations than a standard Bayesian IRL approach that assumes demonstrations are drawn i.i.d. from the demonstrators policy. Research in computational learning theory has shown a dramatic reduction in the number of teaching examples needed to teach anticipatory learners who know they are being taught by a teacher (Doliwa et al. 2014; Gao et al. 2017), but has not addressed sequential decision making tasks. To the best of our knowledge, our work is the first to demonstrate the advantages of an anticipatory IRL algorithm which can leverage the non-i.i.d. nature of highly informative demonstrations from a teacher.

3 Problem formalism

3.1 Markov decision processes

We model the environment as a Markov decision process (MDP), $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma, S_0 \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition function, $R : \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and S_0 is the initial state distribution. A policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is a mapping from states to a probability distribution over actions. We assume that a stochastic optimal policy gives equal probability to all optimal actions. The value of executing policy π starting at state $s \in S$ is defined as

$$V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s]. \tag{1}$$

The Q-value of a state-action pair (s, a) is defined as

$$Q^{\pi}(s, a) = R(s) + \gamma \mathbb{E}_{s' \sim T(\cdot|s,a)}[V^{\pi}(s')] \tag{2}$$

and we denote the optimal Q-value function as $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$.

As is common in the literature (Ziebart et al. 2008; Sadigh et al. 2016; Pirotta and Restelli 2016; Barreto et al. 2017), we assume that the reward function can be expressed as a linear combination of features, $\phi : \mathcal{S} \mapsto \mathbb{R}^k$, so that $R(s) = \mathbf{w}^T \phi(s)$ where $\mathbf{w} \in \mathbb{R}^k$ is the vector of feature weights. This assumption is not restrictive as these features can be nonlinear functions of the state variables. We can write the expected discounted return of a policy as

$$\rho(\pi) \; = \; \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathbf{w}^T \phi(s_t) \mid \pi] \; = \; \mathbf{w}^T \mu_{\pi}, \tag{3}$$

where $\mu_{\pi} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi]$.

3.2 Machine teaching

The machine teaching problem (Zhu 2015) is to select the optimal training set D^* that minimizes the teaching cost, often defined as the size of the data set, and the teaching risk which represents the teacher's dissatisfaction with the model learned by the student. We focus on the constrained form of machine teaching (Zhu et al. 2018) defined as

$$\min_{D} \quad \text{TeachingCost}(D) \tag{4}$$

$$s.t. \quad \text{TeachingRisk}(\hat{\theta}) \leq \epsilon \tag{5}$$

$$\hat{\theta} = \text{MachineLearning}(D) \tag{6}$$

where D is the training set to be optimized, $\hat{\theta}$ is the model the student learns under D, and $\epsilon \geq 0$ determines how much the model learned by the student can differ from the learning target of the teacher.

3.3 Problem definition

We now formulate the optimal teaching problem for IRL as a machine teaching problem. We assume that the expert teacher operates under a ground-truth reward, R^*, and is able to demonstrate state-action pairs (s, a) by executing the corresponding optimal policy π^*. A naive formulation of the machine teaching problem for IRL would be to find the minimal set of demonstrations, $\mathcal{D}$, that enables an IRL agent to learn R^* within some ϵ teaching risk. However, IRL is ill-posed (Ng and Russell 2000)—there are an infinite number of reward functions that explain any optimal policy. Instead, we focus on determining the minimal set of demonstrations that enable a learner to find a reward function that results in an optimal policy with performance similar to the performance of the teacher's policy under R^*. Specifically, we define the policy loss of an estimated weight vector $\hat{\mathbf{w}}$ compared with the true weight vector $\mathbf{w}^*$ as

$$\text{Loss}(\mathbf{w}^*, \hat{\mathbf{w}}) = \mathbf{w}^{*T}(\mu_{\pi^*} - \mu_{\hat{\pi}}), \tag{7}$$

where π^* is the optimal policy under $\mathbf{w}^*$ and $\hat{\pi}$ is the optimal policy under $\hat{\mathbf{w}}$. Equation (7) gives the difference in expected return between the teacher's policy π^* and the expected return of the learner's policy, when both are evaluated under the teacher's reward function $R^* = \mathbf{w}^{*T}\phi(s)$. We can now formalize the machine teaching problem for IRL.

Machine teaching problem for IRL: Given an MDP, $\mathcal{M}$, and the teacher's reward function, $R^* = \mathbf{w}^{*T}\phi(s)$, find the set of demonstrations, $\mathcal{D}$, that minimizes the following optimization problem:

$$\min_{\mathcal{D}} \quad \text{TeachingCost}(\mathcal{D}) \tag{8}$$

$$s.t. \quad \text{Loss}(\mathbf{w}^*, \hat{\mathbf{w}}) \leq \epsilon \tag{9}$$

$$\hat{\mathbf{w}} = \text{IRL}(\mathcal{D}) \tag{10}$$

where $\mathcal{D}$ is the set of demonstrations, and $\hat{\mathbf{w}}$ is the reward recovered by the learner using Inverse Reinforcement Learning (IRL). This formalism covers both exact teaching ($\epsilon = 0$) and approximate teaching ($\epsilon > 0$). In this work we define

$$\text{TeachingCost}(\mathcal{D}) = |\mathcal{D}| \tag{11}$$

where, $|\mathcal{D}|$ can denote either the number of (s, a) pairs in $\mathcal{D}$ or the number of trajectories in $\mathcal{D}$; however, our proposed approach can be easily extended to problems with different teaching costs, e.g., where some demonstrations may be more expensive or dangerous for the teacher.

3.4 Discussion

Like most machine teaching problems (Zhu et al. 2018), the machine teaching problem for IRL is a difficult optimization problem. A brute-force approach would require searching over the power set of all possible demonstrations. This search is intractable due to the size of the power set and the need to solve an IRL problem for each candidate set of demonstrations. One of our contributions is an efficient algorithm for solving the machine teaching problem for IRL that only requires solving a single policy evaluation problem to find the expected feature counts of π^* and then running a greedy set-cover approximation algorithm.

Before discussing our proposed approach in detail, we first introduce the notion of a behavioral equivalence class which is a key component of our approximation algorithm. We will also provide an overview and analysis of the work of Cakmak and Lopes (2012) which provides the baseline and motivation for our approach.

4 Behavioral Equivalence Classes

The *behavioral equivalence class* (BEC) of a policy π is defined as the set of reward functions under which π is optimal:

$$\text{BEC}(\pi) = \{\mathbf{w} \in \mathbb{R}^k \mid \pi \text{ optimal w.r.t. } R(s) = \mathbf{w}^T\phi(s)\}. \tag{12}$$

In this section we briefly discuss how to calculate the behavioral equivalence class for both a policy and for a set of demonstrations from a policy. Given an MDP with either finite or continuous states and with a reward function represented as a linear combination of features, Ng and Russell (2000) derived the behavioral equivalence class (BEC) for a policy. We summarize their result as follows:

Theorem 1. *(Ng and Russell 2000) Given an MDP, BEC(π) is given by the following intersection of half-spaces:*

$$\mathbf{w}^T(\mu_\pi^{(s,a)} - \mu_\pi^{(s,b)}) \geq 0, \tag{13}$$

$$\forall a \in \arg\max_{a' \in \mathcal{A}} Q^*(s, a'), b \in \mathcal{A}, s \in \mathcal{S}, \tag{14}$$

where $\mathbf{w} \in \mathbb{R}^k$ are the reward function weights and

$$\mu_\pi^{(s,a)} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi, s_0 = s, a_0 = a], \tag{15}$$

is the vector of expected feature counts that result from taking action a in state s and following π thereafter.

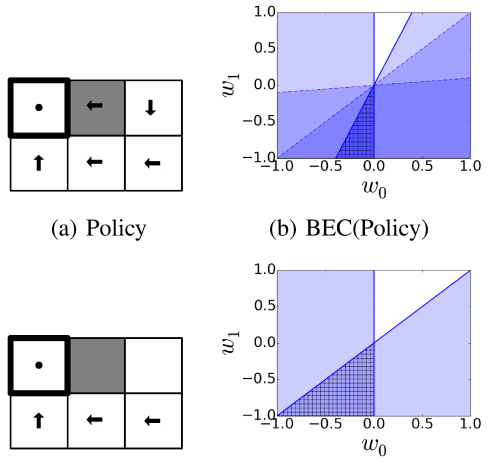
We can similarly define the BEC for a set of demonstrations $\mathcal{D}$ from a policy π:

Corollary 1. *BEC($\mathcal{D}|\pi$) is given by the following intersection of half-spaces:*

$$\mathbf{w}^T(\mu_\pi^{(s,a)} - \mu_\pi^{(s,b)}) \geq 0, \ \forall (s, a) \in \mathcal{D}, b \in \mathcal{A}. \tag{16}$$

All proofs can be found in the appendix.

Example: Consider the grid world shown in Figure 1(a), with four actions available in each state and deterministic transitions. We computed the BEC using a featurized reward function $R(s) = \mathbf{w}^T\phi(s)$, where $\mathbf{w} = (w_0, w_1)$ with w_0 indicating the reward weight for a "white" cell and w_1 indicating the reward weight for the "grey" cell (see appendix for full details). The resulting half-space constraints are shown in Figure 1(b). The intersection of these half-spaces exactly describes the set of rewards that make the policy shown in Figure 1(a) optimal: both white and grey cells have negative reward and the weight for the grey feature is low enough that the optimal policy avoids the shaded cell when starting from the top right cell.



(a) Policy (b) BEC(Policy)

(c) Demonstration (d) BEC(Demonstration)

Figure 1: Behavioral equivalence classes (BEC) for a grid world with 6 states. The upper left state is a terminal state. Each state has 4 available actions and the reward function is a linear combination of two binary features that indicate whether the color of the cell is white or grey. (a) The optimal policy. (b) The resulting intersection of half-spaces (shaded region) that defines all weight vectors such that the policy shown in (a) is optimal. (c) A demonstration from the optimal policy in (a) that is not maximally informative. (d) The intersection of half-spaces (shaded region) resulting from the demonstration shown in (c).

Figures 1(c) and 1(d) show the BEC of a demonstration. The demonstration shows that both feature weights are non-positive and that w_1 is no better than w_0 (otherwise the demonstration would have gone through the grey cell); however, the demonstration leaves open the possibility that all feature weights are equal. However, if the demonstration had started in the top right cell, the BEC of the demonstration would be identical to the BEC of the optimal policy. This highlights the fact that some demonstrations from an optimal policy are more informative than others. An efficient algorithm for finding maximally informative demonstrations using the BEC of the teachers policy is one of the contributions of this paper.

5 Uncertainty Volume Minimization

We now give an overview of the algorithmic teaching approach proposed by Cakmak and Lopes (2012) which motivates our work. The main insight that Cakmak and Lopes use is that of Corollary 1: if an optimal demonstration contains (s, a), then an IRL algorithm can infer that

$$Q^*(s,a) \geq Q^*(s,b), \forall b \in \mathcal{A} \tag{17}$$

$$\Leftrightarrow \quad \mathbf{w}^T(\mu_{\pi^*}^{(s,a)} - \mu_{\pi^*}^{(s,b)}) \geq 0, \forall b \in \mathcal{A}. \tag{18}$$

Given a candidate demonstration set $\mathcal{D}$, Cakmak and Lopes use the intersection of the corresponding half-spaces (as defined in Corollary 1) as a representation of the learner's uncertainty over the true reward function. They use a Monte Carlo estimate of the volume of this cone as a measure of the learner's uncertainty, and seek demonstrations that minimize the uncertainty, $G(\mathcal{D})$, over the true reward function, where

$$G(\mathcal{D}) = \frac{1}{N}\sum_{j=1}^{N} \delta(x_j \in C(\mathcal{D})), \tag{19}$$

δ is an indicator function, $C(\mathcal{D})$ is the intersection of half-spaces given in Corollary 1, and the volume is estimated by drawing N random points x_j from $[-1, 1]^k$. The set of maximally informative demonstrations is chosen greedily by iteratively selecting the trajectory that maximally decreases $G(\mathcal{D})$. This process repeats until $G(\mathcal{D})$ falls below a user defined threshold ϵ. We refer to this algorithm as the Uncertainty Volume Minimization (UVM) algorithm.

In the UVM algorithm, trajectories are added until $G(\mathcal{D})$ is below some user-provided threshold ϵ; however, this does not solve the machine teaching problem for IRL presented in Section 3.3. This is because it only ensures that the estimated uncertainty volume $G(\mathcal{D})$ is less than ϵ, not that the policy loss (Eq. (7)) is less than ϵ. In order to guarantee that the policy loss is below a desireable level, this threshold must be carefully tuned for every MDP. If ϵ is too low, then the algorithm will never terminate. Alternatively, if ϵ is too high, then not enough demonstrations will be selected to teach an appropriate reward function and the policy loss may be large, depending on the reward function selected by the IRL algorithm. In our experiments we remove the need for parameter tuning by stopping the UVM algorithm if it cannot add another demonstration that decreases $G(\mathcal{D})$.

Another limitation of the UVM algorithm is that of volume estimation: exact volume estimation is $\#\mathcal{P}$-hard (Valiant 1979; Simonovits 2003) and straightforward Monte Carlo estimation is known to fail in high-dimensions (Simonovits 2003). Additionally, if there are two (or more) actions that are both optimal in state s, and those two actions (a and b) are demonstrated, this will result in the following constraints:

$$w^T(\mu_\pi^{(s,a)} - \mu_\pi^{(s,b)}) \geq 0 \text{ and } w^T(\mu_\pi^{(s,b)} - \mu_\pi^{(s,a)}) \geq 0$$
$$\Rightarrow w^T(\mu_\pi^{(s,b)} - \mu_\pi^{(s,a)}) = 0 \tag{20}$$

This is problematic because any strict subspace of $\mathbb{R}^k$ has measure zero, resulting in an uncertainty volume of zero. Thus, the UVM algorithm will terminate with zero uncertainty if two optimal actions are ever demonstrated from the same state, even if this leaves an entire $(k-1)$ dimensional subspace of uncertainty over the reward function. Note that when π^* is a stochastic optimal policy this behavior will occur once any trajectory that contains a state with more than one optimal action is chosen—the best trajectory to select next will always be one that visits a previously demonstrated state and chooses an alternative optimal action, resulting in zero uncertainty volume.

One possible solution would be to use more sophisticated sampling techniques such as hit-and-run sampling (Smith 1984). However, these methods still assume there are points on the interior of the sampling region and can be computationally intensive, as they require running long Markov chains to ensure good mixing. Another possible remedy is

to only use deterministic policies for teaching. We instead propose a novel approach based on a set cover equivalence which removes the need to estimate volumes and works for both deterministic and stochastic teacher policies.

6 Set Cover Machine Teaching for IRL

Our proposed algorithm seeks to remedy the problems with the UVM algorithm identified in the previous section in order to find an efficient approximation to the machine teaching problem proposed in Section 3.3.

Our first insight is the following:

Proposition 1. *Consider an optimal policy π^* for reward $R^*(s) = \mathbf{w}^{*T}\phi(s)$. Given any weight vector $\mathbf{w} \in$ BEC(π^*), if $R(s) = \mathbf{w}^T\phi(s)$ is not constant for all states in $\mathcal{S}$, then Loss($\mathbf{w}^*$, $\mathbf{w}$) = 0.*

This proposition says that if we have a non-degenerate weight vector in the behavioral equivalence class for a policy π^*, then we incur zero policy loss by using $\mathbf{w}$ rather than $\mathbf{w}^*$ for performing policy optimization. This follows directly from Equation (12). Thus, to ensure that the policy loss constraint, $\text{Loss}(\mathbf{w}^*, \hat{\mathbf{w}}) \leq \epsilon$, holds in the machine teaching problem, we can focus on finding a demonstration set $\mathcal{D}$ such that the weight vector, $\hat{\mathbf{w}}$, learned through IRL is in BEC(π^*).

Note that Proposition 1 also assumes that the IRL agent being taught will not find a degenerate reward function if a non-degenerate solution exists. This property is true of all standard IRL methods (Gao et al. 2012; Arora and Doshi 2018). While it is possible that an IRL algorithm may return a constant reward function (e.g., $R(s) = 0, \forall s \in \mathcal{S}$) if $\mathcal{D} = \emptyset$, the only way for $\mathcal{D} = \emptyset$ to be the optimal solution for machine teaching is if the resulting loss is less than ϵ, i.e.,

$$\text{Loss}(\mathbf{w}^*, \hat{\mathbf{w}}) = \mathbf{w}^{*T}(\mu_{\pi^*} - \mu_{\hat{\pi}}) \leq \epsilon. \tag{21}$$

For $\epsilon = 0$, this will be false since a constant reward function will almost surely lead to an optimal policy $\hat{\pi}$ which does not match the feature counts of the teacher's policy, π^*.

Our second insight is based on the fact that the behavioral equivalence class for π^* is an intersection of half-spaces (Theorem 1). Rather than give demonstrations until the uncertainty volume, $G(\mathcal{D})$, is less than some arbitrary value, demonstrations should be chosen specifically to define BEC(π^*). Thus, to obtain a feasible solution to the machine teaching problem for IRL we need to select a demonstration set such that the corresponding intersection of half-spaces, BEC($\mathcal{D}|\pi^*$) is equal to BEC(π^*).

Our final insight is to formulate an efficient approximation algorithm for the machine teaching problem for IRL through a reduction to the set cover problem. This allows us to avoid the difficult volume estimation problem required by the UVM algorithm and focus instead on a well known discrete optimization problem. From Section 4 we know that the behavioral equivalence class of both a policy and a demonstration are both characterized by intersections of half-spaces, and each demonstration from π^* produces an intersection of half-spaces which contains BEC(π^*). Thus, the machine teaching problem for IRL (Section 3.3) with $\epsilon = 0$ is an instance of the set cover problem: we have a set of half-spaces defining BEC(π^*), each possible trajectory from π^* covers zero or more of the half-spaces that define BEC(π^*), and we wish to find the smallest set of demonstrations, $\mathcal{D}$, such that BEC($\mathcal{D}|\pi^*$) = BEC(π^*).

One potential issue is that, as seen in Figure 1, many half-space constraints will be non-binding and we are only interested in covering the non-redundant half-space constraints that minimally define BEC(π^*). To address this, we use linear programming to efficiently remove redundant half-spaces constraints (Paulraj and Sumathi 2010) before running our set cover algorithm (see appendix for details).

Note that this approach allows us to solve the machine teaching IRL problem without needing to repeatedly solve RL or IRL problems. The only full RL problem that needs to be solved is to obtain π^* from $\mathbf{w}^*$. After π^* is obtained, we can efficiently solve for the feature expectations $\mu_{\pi^*}^{(s,a)}$ by solving the following equation

$$\mu_{\pi^*}^{(s,a)} = \phi(s) + \gamma \mathbb{E}_{s'|a}[\mu_{\pi^*}^{(s')}] \tag{22}$$

where $\mu_{\pi^*}^{(s)} = \phi(s) + \gamma \mathbb{E}_{s'|\pi^*(s)}[\mu_{\pi^*}^{(s')}]$. These equations satisfy a Bellman equation and can be solved for efficiently. The values $\mu_{\pi^*}^{(s,a)}$ and $\mu_{\pi^*}^{(s)}$ are often called successor features (Dayan 1993) in reinforcement learning and recent work has shown that they can be efficiently computed for model-free problems with continuous state-spaces (Barreto et al. 2017).

We summarize our approach as follows: Given π^*, the optimal policy under the teachers reward function $\mathbf{w}^*$, (1) Solve for the successor features $\mu_{\pi^*}^{(s,a)}$, (2) Find the half-space constraints for BEC(π^*) using Theorem 1, (3) Find the minimal representation of BEC(π^*) using linear programming, (4) Generate candidate demonstrations under π^* from each starting state and calculate their corresponding half-space unit normal vectors using Corollary 1, and (5) Greedily cover all half-spaces in BEC(π^*) by sequentially picking the candidate demonstration that covers the most uncovered half-spaces.

We call this algorithm Set Cover Optimal Teaching (SCOT) and give pseudo-code in Algorithm 1. In the pseudo-code we use $\hat{\mathbf{N}}[\cdot]$ to denote the set of unit normal vectors for a given intersection of half-spaces and $\setminus$ to denote set subtraction. To generate candidate demonstration trajectories we perform m rollouts of π^* for each start state. If π^* is deterministic, then $m = 1$ is sufficient.

Whereas UVM finds demonstrations that successively slice off volume from the uncertainty region, SCOT directly estimates the minimal set of demonstrations that exactly constrain BEC(π^*). This removes both the need to calculate high-dimensional volumes and the need to determine an appropriate stopping threshold. SCOT also has the following desirable properties:

Proposition 2. *The Set Cover Optimal Teaching (SCOT) algorithm always terminates.*

Theorem 2. *Under the assumption of error-free demonstrations, SCOT is a $(1 - 1/e)$-approximation to the Machine Teaching Problem for IRL (Section 3.3) for the following learning algorithms: Bayesian IRL (Ramachandran and Amir 2007; Choi and Kim 2011), Policy Matching (Neu and Szepesvári 2007), and Maximum Likelihood IRL (Babes et al. 2011; Lopes, Melo, and Montesano 2009).*

Algorithm 1 Set Cover Optimal Teaching (SCOT)

Require: MDP $\mathcal{M}$ with set of possible initial states S_0 and reward function $R^*(s) = \mathbf{w}^{*T}\phi(s)$.

// Compute the behavioral equivalence class of π^*
Compute optimal policy π^* for $\mathcal{M}$ and feature expectations $\mu_{\pi^*}^{(s,a)}$.
Use Theorem 1 to compute BEC(π^*).
$U \leftarrow \hat{\mathbf{N}}[\text{BEC}(\pi^*)]$.
Remove redundant half-space constraints from U.
// Compute candidate demonstration trajectories
$\mathcal{T} = \emptyset$
for all $s_0 \in S_0$ **do**
  for $i = 1, \ldots, m$ **do**
    Generate trajectory $\tau = (s_0, a_0, \ldots, s_{H-1}, a_{H-1})$ by starting at s_0 and following π^* for H steps.
    $\mathcal{T} = \mathcal{T} \cup \tau$
    Use Corollary 1 to calculate BEC($\tau|\pi^*$)
  end for
end for
// Solve set cover using greedy approximation
$\mathcal{D} \leftarrow \emptyset$, $C \leftarrow \emptyset$
while $|U \setminus C| \neq 0$ **do**
  $\tau_{\text{greedy}} = \arg\max_{\tau \in \mathcal{T}} \left|\hat{\mathbf{N}}[\text{BEC}(\tau|\pi^*)] \cap U \setminus C\right|$
  $\mathcal{D} = \mathcal{D} \cup \tau_{\text{greedy}}$
  $C = C \cup \hat{\mathbf{N}}[\text{BEC}(\tau|\pi^*)]$
end while
return $\mathcal{D}$

6.1 Algorithm comparison

To compare the performance of SCOT and UVM, we ran an experiment on random 9x9 grid worlds with 8-dimensional binary features per cell. We computed maximally informative demonstration sets with SCOT and UVM using trajectories consisting of single state-action pairs. We measured the performance loss for each algorithm by running IRL to find the maximum likelihood reward function given the demonstrations, and then calculating both the policy loss and the percentage of states where the resulting policy took a suboptimal action under the true reward. Table 2 shows that the UVM algorithm underestimates the size of the optimal teaching set of demonstrations, due to the difficulty of estimating volumes as discussed earlier, resulting in high performance loss. We tried sampling more points, but found that this only slightly improved performance loss while significantly increasing run-time. Compared to UVM, SCOT successfully finds demonstrations that lead to the correct policy, with orders of magnitude less computation.

To further explore the sensitivity of UVM to the number of features, we ran a test on a fixed size grid world with varying numbers of features. We used a deterministic teaching policy to ameliorate the problems with volume computation discussed in Section 5. We found that SCOT is robust for high-dimensional feature spaces, whereas UVM consistently underestimates the minimum number of demonstrations needed when there are 10 or more features, even when teaching a deterministic policy (see appendix for full details).

7 Applications of Machine Teaching for IRL

We now discuss some novel applications of machine teaching for IRL. One immediate application of SCOT is that it allows the first rigorous and efficiently computable definition of intrinsic teaching difficulty (teaching dimension) for IRL benchmarks. In the following section we demonstrate how SCOT can be used to benchmark active IRL algorithms by providing a lower bound on sample complexity. Finally, we demonstrate that SCOT can be incorporated into Bayesian IRL to allow more efficient use of informative demonstrations through counter-factual reasoning.

7.1 Bounding sample complexity for active IRL

Our first application is to provide a lower bound on the sample complexity of learning a reward function via active queries (Lopes, Melo, and Montesano 2009; Cui and Niekum 2017; Brown, Cui, and Niekum 2018). To the best of our knowledge, no one has tried to benchmark existing algorithms against optimal queries, due to the combinatorial explosion of possible queries. Note that SCOT requires knowledge of the optimal policy, so it cannot be used directly as an active learning algorithm. Instead, we use SCOT as a tractable approximation to the optimal sequence of queries for active IRL. SCOT generates a sequence of maximally informative demonstrations via the set cover approximation. Thus, we can treat the sequence of demonstrations found by SCOT as an approximation of the best sequence of active queries to ask an oracle when performing active IRL.

We evaluated three active query strategies from the literature: *Max Entropy*, a strategy proposed by Lopes et al. (Lopes, Melo, and Montesano 2009) that queries the state with the highest action entropy, *Max Infogain*, a strategy proposed by Cui at al. (Cui and Niekum 2017) that selects the trajectory with the largest expected change in the posterior $P(R|D)$, and *Max VaR*, a recently proposed risk-aware active IRL strategy (Brown, Cui, and Niekum 2018) that utilizes probabilistic performance bounds for IRL (Brown and Niekum 2018) to query for the optimal action at the state where the maximum likelihood action given the current demonstrations has the highest 0.95-Value-at-Risk (95th-percentile policy loss over the posterior) (Jorion 1997). We compare these algorithms against random queries and against the maximally informative sequence of queries found using SCOT.

We ran an experiment on 100 random 10x10 grid worlds with 10-dimensional binary features. Figure 2 shows the performance loss for each active IRL algorithm. Each iteration corresponds to a single state query and a corresponding optimal trajectory from that state. After adding each new trajectory to $\mathcal{D}$, the MAP reward function is found using Bayesian IRL (Ramachandran and Amir 2007), and the corresponding optimal policy is compared against the optimal policy under the true reward function.

The results in Figure 2 show that all active IRL approaches perform similarly for early queries, but that Max Entropy ends up performing no better than random as the number of queries increases. This result matches the findings of prior work which showed that active entropy queries perform similarly to random queries for complex domains (Lopes, Melo,

	Avg. number of (s, a) pairs	Avg. policy loss	Avg. % incorrect actions	Avg. time (s)
UVM (10^5)	5.150	1.539	31.420	567.961
UVM (10^6)	6.650	1.076	19.568	1620.578
UVM (10^7)	8.450	0.555	18.642	10291.365
SCOT	17.160	0.001	0.667	0.965

Table 1: Comparison of Uncertainty Volume Minimization (UVM) and Set Cover Optimal Teaching (SCOT) averaged across 20 random 9x9 grid worlds with 8-dimensional features. UVM(x) was run using x Monte Carlo samples. UVM underestimates the number of (s, a) pairs needed to teach π^*.

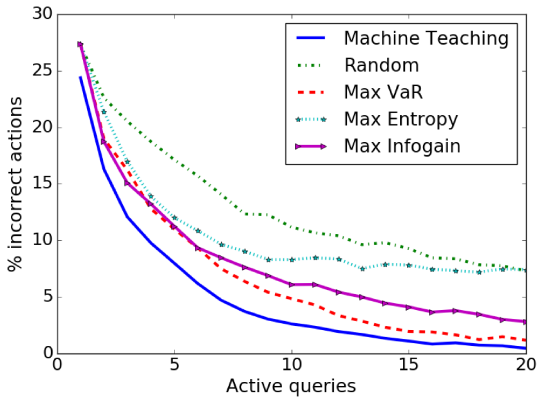


Figure 2: Performance of active IRL algorithms compared to an approximately optimal machine teaching benchmark. Results are averaged over 100 random 10x10 grid worlds.

and Montesano 2009). Max VaR and Max Infogain perform better than Max Entropy for later queries. By benchmarking the against SCOT we see that Max VaR queries are a good approximation of maximally informative queries.

7.2 Using optimal teaching to improve IRL

We next use machine teaching as a novel way to improve IRL when demonstrations are known to be informative. Human teachers are known to give highly informative, non i.i.d. demonstrations when teaching (Ho et al. 2016; Shafto and Goodman 2008). For example, when giving demonstrations, human teachers do not randomly sample from the optimal policy, potentially giving the same (or highly similar) demonstration twice. However, existing IRL approaches usually assume demonstrations are i.i.d. (Ramachandran and Amir 2007; Ziebart et al. 2008; Babes et al. 2011; Fu, Luo, and Levine 2017). We propose an algorithm called Bayesian Information-Optimal IRL (BIO-IRL) that adds a notion of demonstrator informativeness to Bayesian IRL (BIRL) (Ramachandran and Amir 2007). Our insight is that if a learner knows it is receiving demonstrations from a teacher, then the learner should search for a reward function that makes the demonstrations look both optimal and informative.

BIO-IRL algorithm Our proposed algorithm leverages the assumption of an expert teacher: demonstrations not only follow π^*, but are also highly informative. We use the following likelihood for BIO-IRL:

$$P(\mathcal{D}|R) \propto P_{\text{info}}(\mathcal{D}|R) \cdot \prod_{(s,a)\in\mathcal{D}} P((s,a)|R) \tag{23}$$

where $P((s,a)|R)$ is the standard BIRL softmax likelihood that computes the probability of taking action a in state s under R and $P_{\text{info}}(\mathcal{D}|R)$ measures how informative the entire demonstration set $\mathcal{D}$ appears under R.

We compute $P_{\text{info}}(\mathcal{D}|R)$ as follows. Given a demonstration set $\mathcal{D}$ and a hypothesis reward function R, we first compute the information gap, infoGap$(\mathcal{D}, R)$, which uses behavioral equivalence classes to compare the relative informativeness of $\mathcal{D}$ under R, with the informativeness of the maximally informative teaching set $\mathcal{D}^*$ under R (see Algorithm 2). We estimate informativeness by computing the angular similarity between half-space normal vectors (see appendix for details). BIO-IRL uses the absolute difference between angular similarities to counterfactually reason about the gap in informativeness between the actual demonstrations and an equally sized set of demonstrations designed to teach R. Given the actual demonstration $\mathcal{D}$ and the machine teaching demonstration $\mathcal{D}^*$ under R, we let

$$P_{\text{info}}(\mathcal{D}|R) \propto \exp(-\lambda \cdot \text{infoGap}(\mathcal{D}, R)) \tag{24}$$

where $\lambda \geq 0$ is a hyperparameter modeling the confidence that the demonstrations are informative. If $\lambda = 0$, then BIO-IRL is equivalent to standard BIRL.

The main computational bottlenecks in Algorithm 2 are finding the optimal policy for R^*, and computing the expected feature counts for state-action pairs that are used to calculate the various BEC constraints. Computing the optimal policy π^* is already required for BIRL. Computing the expected feature counts for the behavioral equivalence classes is equivalent to performing a policy evaluation step which is computationally cheaper than fully solving for π^* (Barreto et al. 2017). By caching BEC(π^*) for each reward function it is possible to save significant computation time during MCMC by reusing the cached BEC if a proposal R satisfies the vectorized version of Theorem 1 (see (Choi and Kim 2011) and Corollary 2 in the appendix).

Experiments Consider the Markov chain in Figure 3. If an information-optimal demonstrator gives the single demonstration shown on the left, then BIO-IRL assumes that both the orange and black features are equally preferable over white. This is because, given different preferences, an informative teacher would have demonstrated a different trajectory. For

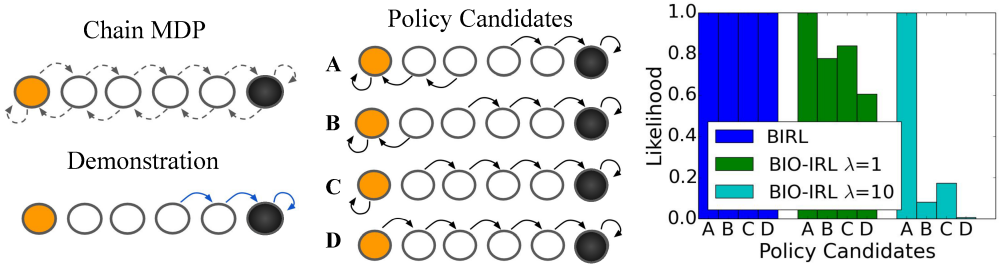


Figure 3: Simple Markov chain with three features (orange, white, black) and two actions available in each state. Left: a single demonstrated trajectory. Center: all policies that are consistent with the demonstration. Right: likelihoods for BIRL and BIO-IRL with λ = 1 and 10. BIRL gives all rewards that lead to any of the policy candidates a likelihood of 1.0 since it only reasons about the optimality of the demonstration under a hypothesis reward. BIO-IRL reasons about both the optimality of the demonstration and the informativeness of the demonstration and gives highest likelihood to reward functions that induce policy A.

Algorithm 2 infoGap($\mathcal{D}, R$)

Calculate π^* under R.
Calculate BEC(π^*), BEC($\mathcal{D}|\pi^*$)
$\mathcal{D}^* \leftarrow$ **SCOT**(π^*)
$m \leftarrow$ number of trajectories in $\mathcal{D}$
$\mathcal{D}^*_{1:m} \leftarrow$ first m trajectories in $\mathcal{D}^*$
Calculate BEC($\mathcal{D}^*|\pi^*$)
infoDemo $\leftarrow$ **angSim**($\hat{\mathbf{N}}$[BEC($\mathcal{D}|\pi^*$)], $\hat{\mathbf{N}}$[BEC(π^*)])
infoOpt $\leftarrow$ **angSim**($\hat{\mathbf{N}}$[BEC($\mathcal{D}^*_{1:m}|\pi^*$)], $\hat{\mathbf{N}}$[BEC(π^*)])
return **abs**(infoDemo - infoCounterfact)

example, if the black feature was always preferred over white and orange, a maximally informative demonstration would have demonstrated a trajectory that started in the leftmost state and moved right until it reached the rightmost state.

Because the likelihood function for standard BIRL only measures how well the demonstration matches the optimal policy for a given reward function, BIRL assigns equal likelihoods to all reward functions that result in one of the policy candidates shown in the center of the Figure 3. The bar graph in Figure 3 shows the likelihood of each policy candidate under BIRL and BIO-IRL with different λ parameters. Rather than assigning equal likelihoods, BIO-IRL puts higher likelihood on Policy A, the policy that makes the demonstration appear both optimal and informative. Changing the likelihood function to reflect informative demonstrations results in a tighter posterior distribution, which is beneficial when reasoning about safety in IRL (Brown and Niekum 2018).

We also evaluated BIO-IRL in a ball sorting task shown in Figure 4(a). In this task, balls start in one of 25 evenly spaced starting conditions and need to be moved into one of four bins located on the corners of the table. Demonstrations are given by selecting an initial state for the ball and moving the ball until it is in one of the bins. Actions are discretized to the four cardinal directions along the table top and the reward is a linear combination of five indicator features representing whether the ball is in one of the four bins or on the table. We generated demonstrations from 50 random rewards with $\gamma = 0.95$. This provides a wide variety of preferences over bin placement depending on the initial distance of the ball to the different bins. We used SCOT to generate informative demonstrations which were given sequentially to BIRL and BIO-IRL. Figure 4 shows that BIO-IRL is able leverage informative, non-i.i.d. demonstrations to learn more efficiently than BIRL (see appendix for details).

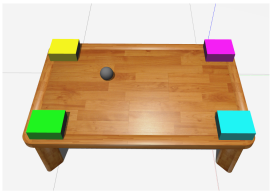
8 Summary and Future Work

We formalized the problem of optimally teaching an IRL agent as a machine teaching problem and proposed an efficient approximation algorithm, SCOT, to solve the machine teaching problem for IRL. Through a set-cover reduction we avoid sampling and use submodularity to achieve an efficient approximation algorithm with theoretical guarantees that the learned reward and policy are correct. Our proposed approach enables an efficient and robust algorithm for selecting maximally informative demonstrations that shows several orders of magnitude improvement in computation time over prior work and scales better to higher-dimensional problems.

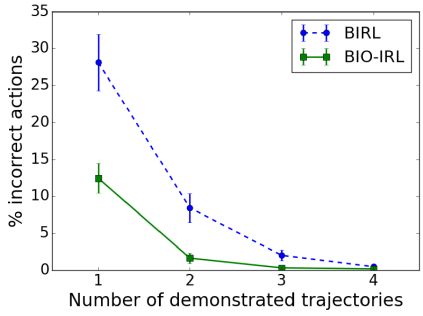
For our first application of machine teaching for IRL we examined using SCOT to approximate lower bounds on sample complexity for active IRL algorithms. Benchmarking active IRL against an approximately optimal query strategy shows that a recent risk-sensitive IRL approach (Brown, Cui, and Niekum 2018) is approaching the machine teaching lower bound on sample complexity for grid navigation tasks.

Our second application of machine teaching demonstrated that an agent that knows it is receiving informative demonstrations can learn more efficiently than a standard Bayesian IRL approach. When humans teach each other we typically do not randomly pick examples to show, rather expert teachers cherry-pick pick highly informative demonstrations that highlight important details and guide the learner away from common pitfalls. However, most IRL algorithms assume that demonstrations are sampled i.i.d. from the demonstrator's policy. We proposed BIO-IRL as a way to use machine teaching to relax i.i.d. assumptions and correct for demonstration bias when learning from an informative teacher.

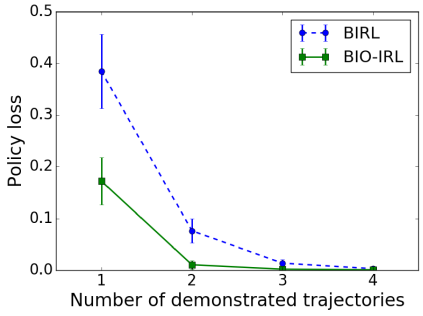
One area of future work is applying SCOT to continuous state-spaces. Expected feature counts can be efficiently computed even if the model is unknown and the state-space is



(a)



(b)



(c)

Figure 4: (a) Ball sorting task. The ball starts at one of 36 positions on the table and must be moved into one of four bins depending on the demonstrator's preferences. 0-1 action losses (b) and policy value losses for MAP policies found by BIO-IRL and BIRL when receiving informative demonstrations. Error bars are 95% confidence intervals around the mean from 50 trials.

continuous (Barreto et al. 2017). Additionally, behavioral equivalence classes can be approximated for continuous state spaces by sampling a representative set of starting states. Given an approximation of the BEC for continuous spaces, SCOT is still guaranteed to terminate and retain efficiency (see appendix for details). Future work also includes using SCOT to approximate the theoretical lower bound on sample complexity for more complicated active learning domains, benchmarking other active learning approaches, and using our proposed machine teaching framework to rank common LfD benchmarks according to their inherent teaching difficulty. Future work should also investigate methods for estimating demonstrator informativeness as well as methods to detect and correct for other demonstrator biases in order to learn more efficiently from non-i.i.d. demonstrations.

There are many other interesting possible applications of machine teaching to IRL. One potential application is to use machine teaching to study the robustness of IRL algorithms to poor or malicious demonstrations by studying optimal demonstration set attacks and defenses (Mei and Zhu 2015; Alfeld, Zhu, and Barford 2017). Machine teaching for sequential decision making tasks also has applications in ad hoc teamwork (Stone et al. 2010) and explainable AI (Gunning 2017). In ad hoc teamwork, one or more robots or agents may need to efficiently provide information about their intentions without having a reliable or agreed upon communication protocol. Agents could use our proposed machine teaching algorithm to devise maximally informative trajectories to convey intent to other agents. Similarly, in the context of explainable AI, a robot or machine may need to convey its intention or objectives to a human (Huang et al. 2017). Simply showing a few maximally informative examples can be a simple yet powerful way to convey intention.

Acknowledgments

This work has taken place in the Personal Autonomous Robotics Lab (PeARL) at The University of Texas at Austin. PeARL research is supported in part by the NSF (IIS-1724157, IIS-1638107, IIS-1617639, IIS-1749204) and ONR (N00014-18-2243).

References

Alfeld, S.; Zhu, X.; and Barford, P. 2017. Explicit defense actions against test-set attacks. In *AAAI*, 1274–1280.

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Arora, S., and Doshi, P. 2018. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*.

Babes, M.; Marivate, V.; Subramanian, K.; and Littman, M. L. 2011. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 897–904.

Balbach, F. J., and Zeugmann, T. 2009. Recent developments in algorithmic teaching. In *International Conference on Language and Automata Theory and Applications*, 1–18.

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; van Hasselt, H. P.; and Silver, D. 2017. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, 4055–4065.

Brown, D. S., and Niekum, S. 2018. Efficient Probabilistic Performance Bounds for Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*.

Brown, D. S.; Cui, Y.; and Niekum, S. 2018. Risk-aware active inverse reinforcement learning. In *Proceedings of the 2nd Annual Conference on Robot Learning (CoRL)*.

Cakmak, M., and Lopes, M. 2012. Algorithmic and human teaching of sequential decision tasks. In *AAAI*.

Choi, J., and Kim, K.-E. 2011. Map inference for bayesian inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, 1989–1997.

Cohn, R.; Durfee, E.; and Singh, S. 2011. Comparing action-query strategies in semi-autonomous agents. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, 1287–1288.

Cui, Y., and Niekum, S. 2017. Active learning from critiques via bayesian inverse reinforcement learning. In *Robotics:*

Science and Systems Workshop on Mathematical Models, Algorithms, and Human-Robot Interaction.

Dayan, P. 1993. Improving generalization for temporal difference learning: The successor representation. *Neural Computation* 5(4):613–624.

Doliwa, T.; Fan, G.; Simon, H. U.; and Zilles, S. 2014. Recursive teaching dimension, vc-dimension and sample compression. *The Journal of Machine Learning Research* 15(1):3107–3131.

Fu, J.; Luo, K.; and Levine, S. 2017. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*.

Gao, Y.; Peters, J.; Tsourdos, A.; Zhifei, S.; and Meng Joo, E. 2012. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics* 5(3):293–311.

Gao, Z.; Ries, C.; Simon, H. U.; and Zilles, S. 2017. Preference-based teaching. *Journal of Machine Learning Research* 18(31):1–32.

Goldman, S. A., and Kearns, M. J. 1995. On the complexity of teaching. *Journal of Computer and System Sciences* 50(1):20–31.

Gunning, D. 2017. Explainable artificial intelligence (xai).

Hadfield-Menell, D.; Russell, S. J.; Abbeel, P.; and Dragan, A. 2016. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems 29*. 3909–3917.

Ho, M. K.; Littman, M.; MacGlashan, J.; Cushman, F.; and Austerweil, J. L. 2016. Showing versus doing: Teaching by demonstration. In *Advances In Neural Information Processing Systems*, 3027–3035.

Huang, S. H.; Held, D.; Abbeel, P.; and Dragan, A. D. 2017. Enabling robots to communicate their objectives. In *Robotics: Science and Systems*.

Jorion, P. 1997. *Value at risk*. McGraw-Hill, New York.

Liu, J., and Zhu, X. 2016. The teaching dimension of linear learners. *Journal of Machine Learning Research* 17(162):1–25.

Lopes, M.; Melo, F.; and Montesano, L. 2009. Active learning for reward estimation in inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 31–46. Springer.

Mei, S., and Zhu, X. 2015. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, 2871–2877.

Melo, F. S.; Lopes, M.; and Ferreira, R. 2010. Analysis of inverse reinforcement learning with perturbed demonstrations. In *ECAI*, 349–354.

Michini, B.; Walsh, T. J.; Agha-Mohammadi, A.-A.; and How, J. P. 2015. Bayesian nonparametric reward learning from demonstration. *IEEE Transactions on Robotics* 31(2):369–386.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming* 14(1):265–294.

Neu, G., and Szepesvári, C. 2007. Apprenticeship learning using inverse reinforcement learing and gradient methods. In *Proc. of 23rd Conference Annual Conference on Uncertainty in Artificial Intelligence*, 295–302.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for inverse reinforcement learning. In *ICML*, 663–670.

Paulraj, S., and Sumathi, P. 2010. A comparative study of redundant constraints identification methods in linear programming problems. *Mathematical Problems in Engineering*.

Pirotta, M., and Restelli, M. 2016. Inverse reinforcement learning through policy gradient minimization. In *AAAI*.

Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, 2586–2591.

Rathnasabapathy, B.; Doshi, P.; and Gmytrasiewicz, P. 2006. Exact solutions of interactive pomdps using behavioral equivalence. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 1025–1032.

Sadigh, D.; Sastry, S. S.; Seshia, S. A.; and Dragan, A. 2016. Information gathering actions over human internal state. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 66–73.

Sadigh, D.; Dragan, A. D.; Sastry, S. S.; and Seshia, S. A. 2017. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*.

Settles, B. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6(1):1–114.

Shafto, P., and Goodman, N. 2008. Teaching games: Statistical sampling assumptions for learning in pedagogical situations. In *Proceedings of the 30th annual conference of the Cognitive Science Society*, 1632–1637.

Simonovits, M. 2003. How to compute the volume in high dimension? *Mathematical programming* 97(1):337–374.

Singla, A.; Bogunovic, I.; Bartók, G.; Karbasi, A.; and Krause, A. 2014. Near-optimally teaching the crowd to classify. In *ICML*, 154–162.

Smith, R. L. 1984. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research* 32(6):1296–1308.

Stone, P.; Kaminka, G. A.; Kraus, S.; Rosenschein, J. S.; et al. 2010. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Valiant, L. G. 1979. The complexity of computing the permanent. *Theoretical computer science* 8(2):189–201.

Wolsey, L. A. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica* 2(4):385–393.

Zeng, Y., and Doshi, P. 2012. Exploiting model equivalences for solving interactive dynamic influence diagrams. *Journal of Artificial Intelligence Research* 43:211–255.

Zhang, H.; Parkes, D. C.; and Chen, Y. 2009. Policy teaching through reward function learning. In *Proceedings of the 10th ACM conference on Electronic commerce*, 295–304. ACM.

Zhu, X.; Singla, A.; Zilles, S.; and Rafferty, A. N. 2018. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*.

Zhu, X. 2015. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, 4083–4087.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*, 1433–1438.

A Behavioral equivalence classes

Theorem 1. *(Ng and Russell 2000) Given an MDP, BEC(π) is given by the following intersection of halfspaces:*

$$\mathbf{w}^T(\mu_\pi^{(s,a)} - \mu_\pi^{(s,b)}) \geq 0, \tag{25}$$

$$\forall a \in \arg\max_{a' \in \mathcal{A}} Q^*(s,a'), b \in \mathcal{A}, s \in \mathcal{S} \tag{26}$$

$\mathbf{w} \in \mathbb{R}^k$ are the reward function weights, $\mu_\pi^{(s,a)} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi, s_0 = s, a_0 = a]$, is the vector of expected feature counts from taking action a in state s and acting optimally thereafter.

Proof. In every state s we can assume that there is one or more optimal actions a. For each optimal action $a \in \arg\max_{a' \in \mathcal{A}} Q^*(s,a')$, we then have by definition that

$$Q^*(s,a) \geq Q^*(s,b),\ \forall b \in A \tag{27}$$

Rewriting this in terms of expected discounted feature counts we have

$$w^T \mu_\pi^{(s,a)} \geq w^T \mu_\pi^{(s,b)},\ \forall b \in A \tag{28}$$

Thus, the behavioral equivalence class is the intersection of the following half-spaces

$$w^T(\mu_\pi^{(s,a)} - \mu_\pi^{(s,b)}) \geq 0, \tag{29}$$

$$\forall a \in \arg\max_{a' \in \mathcal{A}} Q^*(s,a'), b \in \mathcal{A}, s \in \mathcal{S}. \tag{30}$$

□

We can define the BEC for a set of demonstrations $\mathcal{D}$ from a policy π similarly:

Corollary 1. *BEC($\mathcal{D}|\pi$), is given by the following intersection of halfspaces:*

$$\mathbf{w}^T(\mu_\pi(s,a) - \mu_\pi(s,b)) \geq 0,\ \forall (s,a) \in \mathcal{D}, b \in \mathcal{A}. \tag{31}$$

Proof. The proof follows from the proof of Theorem 1 by only considering half-spaces corresponding to optimal (s,a) pairs in the demonstration. □

B Example

Given an MDP with finite states and actions, we can calculate BEC(π) via the following result, proved by Ng and Russell (Ng and Russell 2000), which is equivalent to Theorem 1.

Corollary 2. *(Ng and Russell 2000) Given a finite state space $\mathcal{S}$ with a finite number of actions $\mathcal{A}$, policy π is optimal if and only if reward function R satisfies*

$$(\mathbf{T}_\pi - \mathbf{T_a})(\mathbf{I} - \gamma \mathbf{T}_\pi)^{-1}\mathbf{R} \geq 0,\ \forall a \in \mathcal{A} \tag{32}$$

where $\mathbf{T_a}$ is the transition matrix associated with always taking action a, $\mathbf{T}_\pi$ is the transition matrix associated with policy π, and $\mathbf{R}$ is the column vector of rewards for each state $s \in \mathcal{S}$.

Proof. See (Ng and Russell 2000). □

Consider the grid world shown in Figure 1(a) (see the main text) with four actions (up, down, left, right) available in each state and deterministic transitions. Actions that would leave the grid boundary (such as taking the up action from the states in the top row) result in a self-transition. We computed the BEC region defined by Theorem 2:

$$(\mathbf{T}_\pi - \mathbf{T_a})(\mathbf{I} - \gamma \mathbf{T}_\pi)^{-1}\mathbf{\Phi w} \geq 0$$

for $a \in \{up, down, left, right\}$, setting $\gamma = 0.9$ and using a featurized reward function $R(s) = w^T\phi(s)$, where $w = (w_0, w_1)$ is the feature weight vector with w_0 indicating the reward weight for a "white" cell and w_1 indicating the reward weight for a "shaded" cell. We can express the vector of state rewards as $\mathbf{R} = \Phi\mathbf{w}$, where

$$\Phi = [\phi(s_0)^T, \phi(s_1)^T, \phi(s_2)^T, \phi(s_3)^T, \phi(s_4)^T, \phi(s_5)^T]$$

and $\phi(s_i) = (1,0)$ for $i \in \{0,2,3,4,5\}$ and $\phi(s_1) = (0,1)$, are the feature vectors for each state numbered left to right top to bottom.

The computation results in the following non-redundant constraints that fully define BEC(π) for π given in Figure 1:

$$2.539w_0 - w_1 \geq 0,\ \ -w_0 \geq 0. \tag{33}$$

These constraints exactly describe the set of rewards that make the policy shown in Figure 1(a) optimal. This can be seen by noting that the constraints ensure that all feature weights are non-positive, because a positive weights would cause the optimal policy to avoid early termination to accumulate as much reward as possible. We also have the constraint that if we start in state 3, it is better to move down and around the shaded state then to go directly to the terminal state, this means

$$w_0 + \gamma w_0 + \gamma^2 w_0 + \gamma^3 w_0 + \gamma^4 w_0 \geq w_0 + \gamma w_1 + \gamma^2 w_0 \tag{34}$$

$$\Leftrightarrow (1 + \gamma^2 + \gamma^3)w_0 \geq w_1 \tag{35}$$

which gives us the second constraint using $\gamma = 0.9$. It is straightforward to complete similar inequalities for all states to check that $0 \geq w_0$ and $2.539w_0 \geq w_1$ are the only non-redundant constraints.

Computing the intersection of halfspaces corresponding to the demonstration gives the following convex cone

$$-w_1 \geq 0,\ \ w_1 - w_2 \geq 0. \tag{36}$$

Note that the second constraint on the difference between the two feature weights is looser than the BEC region for

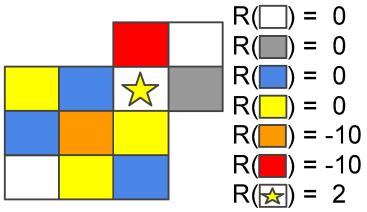


Figure 5: Grid MDP with actions up, down, left, and right. The cell with a star is a terminal state. All other states are possible starting states.

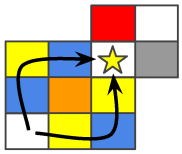


Figure 6: Demonstrations found by the UVM algorithm (Cakmak and Lopes 2012) that result in a false conclusion of zero uncertainty over the demonstrator's true reward. This false certainty comes despite not receiving any evidence about the relative rewards of the red and gray states.

the entire optimal policy. This is because the demonstration only shows that both features are non-positive (making the terminal a goal) and that w_2 is no better than w_1 (otherwise the demonstration would have gone through the shaded region). The demonstration leaves open the possibility that all feature weights are equal. We also note that if the demonstration had started in the top right cell, the BEC region of the demonstration would equal the BEC region of the optimal policy.

C Uncertainty Volume Minimization Algorithm

Pseudo-code for the UVM algorithm is shown in Algorithm 3.

Consider the MDP shown in Figure 5. When the UVM algorithm is run on this task the algorithm exits the while-loop reporting that the uncertainty has gone to zero and returns the demonstration set shown in Figure 6. This is clearly not an optimal demonstration since the starting state in the upper right has never been demonstrated so the agent has no idea what it should do from that state.

This highlights a problem that the UVM algorithm has with estimating volumes. If there are ever two actions that are optimal in a given state s, and those two actions (call them a and b) are demonstrated, then we will have the following halfspace constraints:

$$w^T(\mu_\pi^{(s,a)} - \mu_\pi^{(s,b)}) \geq 0 \tag{37}$$

$$w^T(\mu_\pi^{(s,b)} - \mu_\pi^{(s,a)}) \geq 0 \tag{38}$$

Thus, we have $w^T(\mu_\pi^{(s,b)} - \mu_\pi^{(s,a)}) = 0$.

This is problematic since any strict subspace of $\mathbb{R}^k$ has measure zero, resulting in an uncertainty volume of zero. Thus, for any optimal policy where there exists a state with two or more optimal actions, the UVM algorithm will terminate with zero uncertainty if these two optimal actions are added to $\mathcal{D}$. This is true, even if this leaves an entire $(k-1)$ dimensional subspace of uncertainty over the reward.

Algorithm 3 Uncertainty Volume Minimization

Require: Set of possible initial states S_0
Require: Feature weights $\mathbf{w}$ of the optimal reward function
Initialize $\mathcal{D} \leftarrow \emptyset$
Compute optimal policy π^* based on $\mathbf{w}$
repeat
  $\zeta_{\text{best}} \leftarrow$ **null**
  for all $s_0 \in S_0$ **do**
    Generate K trajectories from s_0 following π^*
    for $j \in [1, K]$ **do**
      if $G(\mathcal{D} \cup \zeta_j) < G(\mathcal{D} \cup \zeta_{\text{best}})$ and $\zeta_j \notin \mathcal{D}$ **then**
        $\zeta_{\text{best}} \leftarrow \zeta_j$
      end if
    end for
  end for
  $\mathcal{D} \leftarrow \mathcal{D} \cup \zeta_{\text{best}}$
until ζ_{best} **is null**
return Demonstration set $\mathcal{D}$

D Removing redundant constraints

A redundant constraint is one that can be removed without changing the BEC region. We can find redundant constraints efficiently using linear programming. To check if a constraint $a^T x \leq b$ is binding we can remove that constraint and solve the linear program with $\max_x a^T x$ as the objective. If the optimal solution is still constrained to be less than or equal to b even when the constraint is removed, then the constraint can be removed. However, if the optimal value is greater than b then the constraint is non-redundant. Thus, all redundant constraints can be removed by making one pass through the constraints, where each constraint is immediately removed if redundant. We optimize this approach by first normalizing each constraint and removing duplicates and any trivial, all zero constraints.

E Set-cover algorithm termination

Proposition 2. *The set-cover machine teaching algorithm for IRL always terminates.*

Proof. To prove that our algorithm always terminates, consider the polyhedral cone $C_f = \{x \in \mathbb{R}^n \mid A_f x \geq 0\}$ that represents BEC(π^*). Each demonstrated state action pair, (s, a) defines a set of half spaces

$$w^T(\mu(s, \pi^*(s)) - \mu(s, a)) \geq 0, \ \forall a \in \mathcal{A}. \tag{39}$$

When the intersection of the halfspaces for every $(s, a) \in \mathcal{D}$ is equal to BEC(π^*), the algorithm terminates an returns $\mathcal{D}$. For discrete domains, BEC(π^*) is simply the intersection of a finite number of half-spaces from every optimal (s, a) pair, so once every optimal (s, a) has been demonstrated, the algorithm is guaranteed to terminate. In practice, BEC(π^*) can

be fully defined by only a subset of the possible demonstrations. Thus, our machine teaching algorithm seeks to select the minimum number of demonstrations that cover all of the rows of A_f.

For continuous domains, we cannot fully enumerate every optimal (s, a)-pair. However, it is possible to approximate BEC(π^*) by sampling optimal rollouts from the state space. We then solve the constraint set-cover problem using these same sampled rollouts, so we are again guaranteed to terminate once all demonstrations are chosen, and will likely terminate after only selecting a small subset of the sampled demonstrations. □

Proposition 3. *The set-cover machine teaching algorithm is a $(1 - 1/e)$-approximation to the minimum number of demonstrations needed to fully define BEC(π^*).*

Proof. This result follows from the submodularity of the set cover problem (Wolsey 1982; Nemhauser, Wolsey, and Fisher 1978). □

F Optimality of set cover algorithm

We now prove the condition under which our proposed algorithm is a (1-1/e)-approximation of the solution to the Machine Teaching Problem for IRL.

Both the UVM and SCOT algorithms focus on teaching halfspaces to an IRL algorithm to define the behavioral equivalence region, BEC(π^*). Thus, they assume that when the IRL algorithm receives state-action pair (s, a) from the demonstrator, the IRL algorithm will enforce the constraint that $Q^*(s, a) \geq Q^*(s, b), \forall b \in \mathcal{A}$. We call this assumption the *halfspace assumption*.

Definition 1. *The halfspace assumption is that $Q^*(s, a) \geq Q^*(s, b), \forall b \in \mathcal{A}, (s, a) \in \mathcal{D}$.*

We now prove that, under the assumption of error-free demonstrations, three common IRL algorithms make the halfspace assumption: Bayesian IRL (Ramachandran and Amir 2007; Choi and Kim 2011), Policy Matching (Neu and Szepesvári 2007), and Maximum Likelihood IRL (Lopes, Melo, and Montesano 2009; Babes et al. 2011).

Lemma 1. *Under the assumption of error-free demonstrations, Bayesian IRL (Ramachandran and Amir 2007; Choi and Kim 2011) makes the halfspace assumption.*

Proof. Bayesian IRL uses likelihood

$$P_{\text{opt}}(\mathcal{D}|R) = \prod_{(s,a)\in\mathcal{D}} \frac{e^{\alpha Q^*(s,a)}}{\sum_{b\in A} e^{\alpha Q^*(s,b)}} \tag{40}$$

where Q^* is the optimal Q-function under reward function R and $\alpha \in [0, \infty)$ represents the confidence that the demonstrations come from π^*. As $\alpha \to \infty$, Bayesian IRL assume error-free demonstrations and we have

$$\lim_{\alpha\to\infty} P_{\text{opt}}(\mathcal{D}|R) = 0 \iff \exists b \in \mathcal{A}, \text{ s.t. } Q^*(s, a) < Q^*(s, b). \tag{41}$$

Thus, Bayesian IRL only gives positive likelihood to reward functions R, if $Q^*(s, a) \geq Q^*(s, b)\, \forall b \in \mathcal{A}, (s, a) \in \mathcal{D}$. □

Corollary 3. *Under the assumption of error-free demonstrations, Policy Matching (Neu and Szepesvári 2007) makes the optimal teaching assumption.*

Proof. Melo et al. (Melo, Lopes, and Ferreira 2010) proved that Bayesian IRL (Ramachandran and Amir 2007) and Policy Matching (Neu and Szepesvári 2007) share the same reward solution space. Thus, the lemma follows from the previous proof. □

Corollary 4. *Under the assumption of error-free demonstrations, Maximum Likelihood IRL (Lopes, Melo, and Montesano 2009; Babes et al. 2011) makes the optimal teaching assumption.*

Proof. Maximum Likelihood IRL uses the same likelihood function as Bayesian IRL, thus the result follows from the previous lemma. □

We can now prove the following Theorem:

Theorem 2. *Under the assumption of error-free demonstrations, SCOT is a $(1 - 1/e)$-approximation to the Machine Teaching Problem for IRL (Section 3.3) for the following learning algorithms:*

- *Bayesian Inverse Reinforcement Learning (Ramachandran and Amir 2007; Choi and Kim 2011)*
- *Policy Matching (Neu and Szepesvári 2007)*
- *Maximum Likelihood Inverse Reinforcement Learning (Lopes, Melo, and Montesano 2009; Babes et al. 2011)*

Proof. Given an IRL algorithm that makes the halfspace assumption, by Proposition 3 SCOT will find a set of demonstrations that are a (1-1/e)-approximation of the maximally informative demonstration set. Thus, by Lemma 1, in the limit as $\alpha \to \infty$, the SCOT machine teaching algorithm is a (1-1/e)-approximation to the optimal demonstration set for Bayesian IRL. Similarly, by corollaries 3 and 4, SCOT is a (1-1/e)-approximation to the optimal demonstration set for Policy Matching and Maximum Likelihood IRL. □

G Algorithm comparison full results

We compared the SCOT algorithm with the UVM algorithm of Cakmak and Lopes (Cakmak and Lopes 2012). We also report here a comparison against SCOT without removing redundancies and against random selected demonstrations from the optimal policy.

We ran an experiment on random 9x9 grid worlds with 8 binary indicator features per cell with one feature active per cell and $\gamma = 0.95$. For this experiment the demonstrations were single state-action pairs. We measured the 0-1 policy loss (Michini et al. 2015) for each demonstration set by computing the percentage of states where the resulting policy took a suboptimal action under the true reward. The policy was found by first finding the maximum likelihood reward function (Lopes, Melo, and Montesano 2009; Babes et al. 2011), by using BIRL (Ramachandran and Amir 2007) with a uniform prior and $\alpha = 100$. We ran the MCMC chain for 10,000 steps using $\alpha = 100$ and step size of 0.005.

Table 2: Comparison of different optimal teaching algorithms across random 9x9 grid worlds with 8-dimensional binary features. Algorithms compared are Uncertainty Volume Minimization (UVM), Set Cover Optimal Teaching (SCOT) with and without redundancies, and random sampling from the optimal policy. UVM(x) was run using x Monte Carlo samples for volume estimation. Results show the average number of state-action pairs in the demonstration set $\mathcal{D}$, the average number of suboptimal actions when performing IRL using $\mathcal{D}$ learned policy compared to optimal, and the average run time of the optimal teaching algorithm in seconds. All results are averaged over 20 replicates.

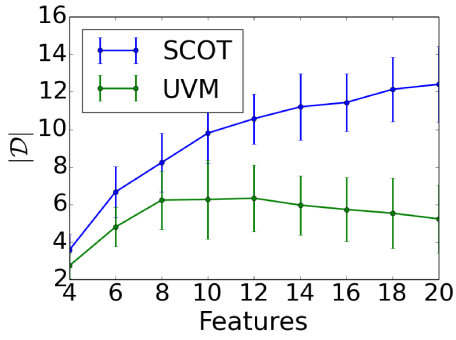
	Ave. (s, a) pairs	Ave. policy loss	Ave. % incorrect actions	Ave. time (s)
UVM (10^4)	3.850	1.722	44.074	247.944
UVM (10^5)	5.150	1.539	31.420	567.961
UVM (10^6)	6.650	1.076	19.568	1620.578
UVM (10^7)	8.450	0.555	18.642	10291.365
SCOT (redundant)	66.740	0.001	0.617	12.407
SCOT	17.160	0.001	0.667	0.965
Random	17.700	0.015	10.123	0.000

Given the maximum likelihood reward function, the corresponding policy was then found using value iteration. The results are shown in Table 2.

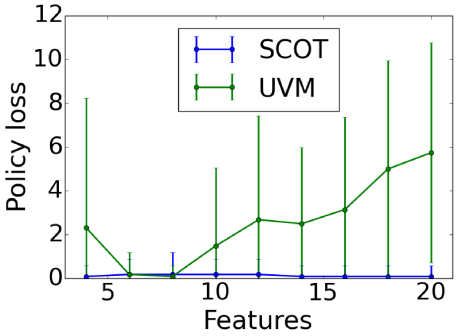
We found that the UVM algorithm usually underestimates the size of the optimal teaching set of demonstrations, due to the difficulty of estimating volumes as discussed earlier, resulting in high 0-1 loss. We tried sampling more points, but found that this only slightly improved 0-1 loss while significantly increasing run-time. Compared to UVM, our results show that SCOT is more accurate and more efficient—it can successfully find good demonstrations that lead to IRL learning the correct reward and corresponding policy, with orders of magnitude less computation time. We also found that removing redundant halfspaces is important; keeping redundant constraints results in slightly lower average performance loss, due to the randomness in MCMC, but finds demonstration sets that are much larger. SCOT removes redundant half-space constraints which results in solutions with fewer state-action pairs and a faster run-time, since the set-cover problem is substantially reduced in size.

We also compared against randomly sampling 20 state-action pairs from the optimal policy. The results show that SCOT is able to find informative demonstrations that significantly reduce the number of (s, a) pairs to teach a policy, compared to sampling i.i.d. from the policy.

To further explore the sensitivity of UVM to the number of features, we ran a test on a fixed 6x6 size grid world with varying numbers of features. We wanted to see if using UVM and SCOT to teach a deterministic policy would avoid the early stopping problem for UVM by avoiding the problem of having multiple optimal actions from the same starting position. We also investigated using longer demonstrations with a horizon of 6. We found that the SCOT algorithm picks more demonstrations as the number of features increases, however, the UVM algorithm cannot reliably estimate volumes for higher-dimensional spaces. The results are shown in Figure 7. The number of state-action pairs in the demonstrations are shown to plateau and even slightly decrease for UVM. Thus, UVM underestimates the number of required demonstrations to teach an optimal policy for high-dimensional features while still requiring nearly than three orders of magnitude more computation time.



(a)



(b)

Figure 7: SCOT algorithm is robust to increasing numbers of features, whereas UVM is not robust since it relies on good volume estimates. Results averaged over 30 runs on a 6x6 grid world. UVM uses 1,000,000 samples for volume estimation.

H Active learning experiment parameters

We generated 10x10 random grid worlds where each state was assigned a random one-hot 10-dimensional feature vector we set the discount factor to $\gamma = 0.95$. We ran BIRL with

a uniform prior to obtain the MAP reward function given demonstrations for each active IRL algorithm. Each active query resulted in an optimal trajectory of length 20 demonstrated from the optimal policy. We ran the MCMC chain for 10,000 steps using $\alpha = 100$ and step size of 0.005.

I BIO-IRL algorithm specifics

We calculate angular similarity as follows: We take all the normal vectors from BEC($\mathcal{D}|\pi^*$) and do a greedy matching to BEC(π^*). Once we match the first vector in BEC($\mathcal{D}|\pi^*$) with the closest vector in BEC(R), we remove the best match from BEC(π^*) and continue with the next vector in BEC($\mathcal{D}|\pi^*$). When there are no remaining half-spaces in BEC($\mathcal{D}|\pi^*$) The algorithm returns the cumulative sum of half-space similarities divided by the number of half-spaces in BEC(π^*).

Because our normal vectors can have positive and negative elements, we define the similarity between two vectors $\mathbf{x}$ and $\mathbf{y}$ as

$$similarity(\mathbf{x}, \mathbf{y}) = 1 - \cos^{-1}(\mathbf{x} \cdot \mathbf{y})/\pi. \qquad (42)$$

I.1 Markov chain BIO-IRL experiment

We used $\alpha = 100$ as the softmax temperature parameter for BIRL and BIO-IRL.

I.2 Ball sorting BIO-IRL experiment

We discretized the table top into a 6x6 grid of positions, all of which are potential starting states. The four discrete actions move the ball along the table top in the four cardinal directions. The demonstrations consisted of optimal trajectories found using Value Iteration of length 10. BIRL and BIO-IRL both used the following parameters: $\alpha = 100$, MCMC chain length=1000, MCMC step size = 0.05. BIO-IRL used $\lambda = 100$.