




# Interactively Teaching an Inverse Reinforcement Learner with Limited Feedback

Rustam Zayanov<sup>1</sup><sup>a</sup>, Francisco S. Melo<sup>1</sup><sup>b</sup> and Manuel Lopes<sup>1</sup><sup>c</sup>

<sup>1</sup>*INESC-ID & Instituto Superior Técnico, Universidade de Lisboa*

*rustam.zayanov@tecnico.ulisboa.pt, fmelo@inesc-id.pt, manuel.lopes@tecnico.ulisboa.pt*

**Keywords:** Sequential Decision Processes, Inverse Reinforcement Learning, Machine Teaching, Interactive Teaching and Learning

**Abstract:** We study the problem of teaching via demonstrations in sequential decision-making tasks. In particular, we focus on the situation when the teacher has no access to the learner’s model and policy, and the feedback from the learner is limited to trajectories that start from states selected by the teacher. The necessity to select the starting states and infer the learner’s policy creates an opportunity for using the methods of inverse reinforcement learning and active learning by the teacher. In this work, we formalize the teaching process with limited feedback and propose an algorithm that solves this teaching problem. The algorithm uses a modified version of the active value-at-risk method to select the starting states, a modified maximum causal entropy algorithm to infer the policy, and the difficulty score ratio method to choose the teaching demonstrations. We test the algorithm in a synthetic car driving environment and conclude that the proposed algorithm is an effective solution when the learner’s feedback is limited.

## 1 INTRODUCTION


Machine Teaching (MT) is a computer science field that formally studies a learning process from a teacher’s point of view. The teacher’s goal is to teach a target concept to a learner by demonstrating an optimal (often the shortest) sequence of examples. MT has the potential to be applied to a wide range of practical problems (Zhu, 2015; Zhu et al., 2018), such as: developing better Intelligent Tutoring Systems for automated teaching for humans, developing smarter learning algorithms for robots, determining the teachability of various concept classes, testing the validity of human cognitive models, and cybersecurity.


One promising application domain of MT is the automated teaching of sequential decision skills to human learners, such as piloting an airplane or performing a surgical operation. In this domain, MT can be combined with the theory of Inverse Reinforcement Learning (IRL) (Ng et al., 2000; Abbeel and Ng, 2004), also known as Inverse Optimal Control. IRL formally studies algorithms for inferring an agent’s goal based on its observed behavior in a sequential decision setting. Assuming that a learner will use a spe-


cific IRL algorithm to process the teacher’s demonstrations, the teacher could pick an optimal demonstration sequence for that algorithm.

Most MT algorithms assume that the teacher knows the learner’s model, that is, the learner’s algorithm of processing demonstrations and converting them into knowledge about the target concept. In the case of human cognition, formalizing and verifying such learner models is still an open research question. The scarcity of such models poses a challenge to the application of MT to automated human teaching. One way of alleviating the necessity of a fully defined learner model is to develop MT algorithms that make fewer assumptions about the learner. In the sequential decision-making domain, (Kamalaruban et al., 2019) and (Yengera et al., 2021) have proposed teaching algorithms that admit some level of uncertainty about the learner model. In particular, their teaching algorithms assume that the learner’s behavior (policy) is maximizing some reward function, but it is unknown how the learner updates that reward function given the teacher’s demonstrations. To cope with this uncertainty, the teacher is allowed to observe the learner’s behavior during the teaching process and infer the learner’s policy from the observed trajectories, thus making the process iterative and interactive.

Both works assume that the teacher can period-

<sup>a</sup> <https://orcid.org/0009-0006-5301-6382>

<sup>b</sup> <https://orcid.org/0000-0001-5705-7372>

<sup>c</sup> <https://orcid.org/0000-0002-6238-8974>

ically observe *many* learner’s trajectories from *every* *initial state* and thus estimate the learner’s policy with high precision. Unfortunately, the need to produce many trajectories from every initial state may be unfeasible in real-life scenarios. In our present work, we address a more realistic scenario in which the feedback from the learner is limited to just *one trajectory* per each iteration of the teaching process. The limit on the learner’s feedback poses a challenge for the teacher in reliably estimating the learner’s policy, which, in turn, may diminish the usefulness of the teacher’s demonstrations. Thus, our research question is: **What are the effective ways of teaching an inverse reinforcement learner when the learner’s policy and update algorithm are unknown, and the learner’s feedback is limited?**

The teacher’s ability to precisely estimate the learner’s policy greatly depends on the informativeness of the received trajectories. We consider two scenarios: an unfavorable scenario when the teacher has no influence on what trajectories it will receive, and a more favorable scenario when the teacher can choose the states from which the learner will generate trajectories. The necessity to select the starting states creates an opportunity for the teacher to use methods of Active Learning (AL) (Settles, 2009). In the context of sequential decision-making, AL considers situations when a learner has to infer an expert’s reward and can interactively choose the states from which the expert’s demonstrations should start (Lopes et al., 2009).

The contribution of our work is two-fold. Firstly, we propose a new framework that formalizes interactive teaching when the learner’s feedback is limited. Secondly, we propose an algorithm for teaching with limited feedback. The algorithm performs three steps per every teaching iteration: selection of a query state (AL problem), inference of the current learner’s policy (IRL problem), and selection of a teaching demonstration (MT problem). The algorithm uses a modified version of the Active-VaR (Brown et al., 2018) method for choosing query states, a modified version of the Maximum Causal Entropy (MCE) (Ziebart et al., 2013) method for inferring the learner’s policy, and the difficulty score ratio (DSR) (Yengera et al., 2021) method for selecting the teaching demonstration. We test the algorithm in a synthetic car driving environment and conclude that it is a viable solution when the learner’s feedback is limited<sup>1</sup>.

<sup>1</sup>The implementation of the algorithms is available at <https://github.com/rzayanov/irl-teaching-limited-feedback>

## 2 RELATED WORK

(Liu et al., 2017) explore the problem of MT with unlimited feedback in the domain of supervised learning when the teacher and the learner represent the target concept as a linear model. They consider a teacher that does not know the feature representation and the parameter of the learner. For this scenario, they introduce an interaction protocol with unlimited learner feedback, where the teacher can query the learner at every step by sending all possible examples and receiving all learner’s output labels. (Liu et al., 2018) continue this work and explore teaching with limited feedback in the same supervised learning setting. Similarly to our work, the teacher can not request all learner’s labels at every step but instead has to choose which examples to query using an AL method.

(Melo et al., 2018) explore how interaction can help when the teacher has wrong assumptions about the learner. The authors focus on the problem of teaching the learners that aim to estimate the mean of a Gaussian distribution given scalar examples. When the teacher knows the correct learner model, the teaching goal is achieved after showing one example. When it has wrong assumptions, and no interaction is allowed, the learner approaches the correct mean only asymptotically. When interaction is allowed, the teacher can query the learner at any time, and the learner responds with the value of its current estimate perturbed by noise. They show that this kind of interaction significantly boosts teaching progress.

(Cakmak and Lopes, 2012) and (Brown and Niekum, 2019) propose non-interactive MT algorithms for sequential decision-making tasks. Both algorithms produce a minimal set of demonstrations that is sufficient to reliably infer the reward function. Both algorithms are agnostic of the learner model and don’t specify the order of demonstrations, which might be crucial for teaching performance if the learner is not capable of processing the whole set at once. The algorithm of (Cakmak and Lopes, 2012) is based on the assumptions that the reward is a linear combination of *state features* and that the teacher will provide enough demonstrations for the learner to estimate the teacher’s expected *feature counts* reliably. With these assumptions, each demonstrated state-action pair induces a half-space constraint on the reward weight vector. Assuming that the learner weights are bounded, it is possible to estimate the volume of the subspace defined by any set of such constraints. A smaller volume means less uncertainty regarding the true weight vector. Thus, demonstrations that minimize the subspace volume are preferred. The authors propose a non-interactive algo-

rithm for choosing the demonstration set: at every step, the teacher will pick a demonstration that minimizes the resulting subspace volume. (Brown and Niekum, 2019) propose an improved non-interactive MT algorithm called Set Cover Optimal Teaching (SCOT). They first define a policy’s *behavioral equivalence class* (BEC) as a set of reward weights under which that policy is optimal. A BEC of a demonstration given a policy is the intersection of half-spaces formed by all state-action pairs present in such demonstration. The authors propose finding the smallest set of demonstrations whose BEC is equal to the BEC of the optimal policy. Finding such a set is a set-cover problem. The proposed algorithm is based on generating  $m$  demonstrations from each starting state and using a greedy method of picking candidates.

(Kamalaruban et al., 2019) and (Yengera et al., 2021) propose interactive MT algorithms for sequential decision-making tasks when the learner can process only one demonstration at a time, but the feedback from the learner is unlimited or has a high limit. (Kamalaruban et al., 2019) first consider an omniscient teacher whose goal is to steer the learner toward the optimal weight parameter and find an effective teaching algorithm. Next, they consider a less informative teacher that can not observe the learner’s policy and has no information about the learner’s feature representations and the update algorithm. Instead of directly observing the current learner’s policy  $\pi_i^L$ , the teacher can periodically request the learner to generate  $k$  trajectories from *every initial state*, thus estimating  $\pi_i^L$ . The limitation of this approach is that the necessity to produce trajectories from every initial state may be hard to implement in practice when  $k$  is high or the number of initial states is high. (Yengera et al., 2021) further explore the problem of teaching with unlimited feedback and propose the difficulty score ratio (DSR) algorithm. They introduce the notion of a *difficulty score* of a trajectory given a policy, which is proportional to its conditional likelihood given that policy, and propose a teaching algorithm that selects a trajectory that maximizes the ratio of difficulty scores of the learner’s policy and the target policy. To the best of our knowledge, the problem of teaching with limited feedback in the domain of sequential decision-making tasks has not yet been addressed in the literature.

### 3 PROBLEM FORMALISM

The underlying task to be solved by an agent is formally represented as a Markov Decision Process

(MDP) denoted as  $M = (\mathcal{S}, \mathcal{A}, \mathbb{T}, \mathbb{P}_0, \gamma, R^*)$ , where  $\mathcal{S}$  is the set of states,  $\mathcal{A}$  is the set of actions,  $\mathbb{T}(S' | s, a)$  is the state transition probability upon taking action  $a$  in state  $s$ ,  $\mathbb{P}_0(S)$  is the initial state distribution,  $\gamma$  is the discount factor, and  $R^* : \mathcal{S} \rightarrow \mathbb{R}$  is the reward function to be learned.

A stationary *policy* is a mapping  $\pi$  that maps each state  $s \in \mathcal{S}$  into a probability distribution  $\pi(\cdot | s)$  over  $\mathcal{A}$ . A policy can be executed in  $M$ , which will produce a sequence of state-action pairs called *trajectory*. For any trajectory  $\xi = \{s_0, a_0, \dots, s_T, a_T\}$ , we will denote its  $i$ -th state and action as  $s_i^\xi$  and  $a_i^\xi$ , respectively. Given a policy  $\pi$ , the *state-value function*  $V^\pi(s)$ , the *expected policy value*  $\mathcal{V}^\pi$ , and the *Q-value function*  $Q^\pi(s, a)$  are defined as follows respectively:

$$V^\pi(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R(S_t) \mid \pi, \mathbb{T}, S_0 = s \right] \quad (1)$$

$$\mathcal{V}^\pi = \mathbb{E}_{S \sim \mathbb{P}_0} [V^\pi(S)] \quad (2)$$

$$Q^\pi(s, a) = R(s) + \gamma \mathbb{E}_{S' \sim \mathbb{T}(\cdot | s, a)} [V^\pi(S')] \quad (3)$$

A policy  $\pi^*$  is considered *optimal* if it has the highest state-values for every state. For any MDP, at least one optimal policy exists, which can be obtained via the policy iteration method (Sutton and Barto, 2018).

## 4 FRAMEWORK FOR TEACHING WITH LIMITED FEEDBACK

In this section, we present our contributions: the framework for teaching with limited feedback and an algorithm for solving the problem of teaching with limited feedback.

We consider two entities that can execute policies on  $M$ : a *teacher* with complete access to  $M$  and a *learner* that can access all elements of  $M$  except the reward function, which we denote as  $M \setminus R^*$ . The teacher and the learner can interact with each other iteratively, with every iteration consisting of five steps described in Algorithm 1. In the first step, the teacher chooses a *query state*  $s_i^q$  and asks the learner to generate a trajectory starting from  $s_i^q$ . We assume that the query states can only be selected from the set of initial states, i.e.,  $s_i^q \in \mathcal{S}_0 = \{s : \mathbb{P}_0(s) > 0\}$ . In the second step, the learner generates a *trajectory*  $\xi_i^L$  by executing its policy starting from  $s_i^q$  and sends it back to the teacher. In the third step, the teacher uses the learner’s trajectory to update its estimate of the learner’s current reward  $\hat{R}_i^L$  and policy  $\hat{\pi}_i^L$ . In the fourth step, the teacher demonstrates the optimal behavior by generating a trajectory  $\xi_i^T$ , which we call a *demonstration*, and sending it to the learner. In the last step, the

learner learns from the demonstration to update its reward  $R_i^L$  and policy  $\pi_i^L$ . The teaching process is terminated when the *teaching goal* is achieved, in the sense defined below.

Algorithm 1: Framework for teaching with limited feedback

- 
- 1: **for**  $i = 1, \dots, \infty$  **do**
  - 2: Teacher sends a *query state*  $s_i^q$  and requests a *trajectory* starting from it
  - 3: Learner generates and sends a *trajectory*  $\xi_i^L$
  - 4: Teacher updates its estimate of the learner’s reward  $\hat{R}_i^L$  and policy  $\hat{\pi}_i^L$
  - 5: Teacher generates and sends a *demonstration*  $\xi_i^T$
  - 6: Learner updates its reward  $R_i^L$  and policy  $\pi_i^L$
  - 7: Stop if the teaching goal is achieved
  - 8: **end for**
- 

We consider the problem described above from the perspective of a teacher that has limited knowledge about the learner. We consider the following set of assumptions:

- **Access to state features:** Both the teacher and the learner can observe the same  $d$  numerical *features* associated with every state, formalized as a mapping  $\phi : \mathcal{S} \rightarrow \mathbb{R}^d$ . The (discounted) *feature counts* are defined for a trajectory  $\xi$  or for a policy  $\pi$  and a state  $s$  as follows:

$$\mu(\xi) = \sum_t \gamma^t \phi(s_t) \quad (4)$$

$$\mu(\pi, s) = \mathbb{E} \left[ \sum_t \gamma^t \phi(S_t) \mid \pi, S_0 = s \right] \quad (5)$$

- **Rationality:** At every iteration, the learner maintains some reward mapping  $R_i^L$  and derives a stationary policy  $\pi_i^L$  that is appropriate for  $R_i^L$ , which it uses to generate trajectories. The exact method of deriving  $\pi_i^L$  from  $R_i^L$  is unknown to the teacher.
- **Reward as a function of features:** As it is common in the IRL literature, the learner represents the reward as a linear function of state features:  $R_i^L(s) = \langle \theta_i, \phi(s) \rangle$ , where the vector  $\theta_i$  is called the *feature weights*. Furthermore, we assume that the true reward  $R^*$  can be expressed as a function of these features, i.e.,  $\exists \theta^* \text{ s.t. } \forall s, R^*(s) = \langle \theta^*, \phi(s) \rangle$ .
- **Learning from demonstrations:** Upon receiving a demonstration  $\xi_i^T$ , the learner uses it to update its parameter  $\theta_{i+1}$  and thus its reward  $R_{i+1}^L$ . The exact method of updating  $\theta_{i+1}$  from  $\xi_i^T$  is unknown to the teacher.

There are different ways of evaluating the teacher’s performance. In general, some notion of nu-

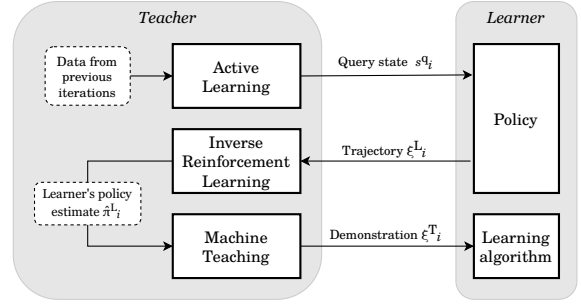


Figure 1: The teaching algorithm can be divided into three modules, each solving AL, IRL, or MT problem at every iteration. This diagram shows the inputs and outputs of the modules.

merical *loss*  $L_i$  is defined for every step (also called the teaching risk), and the teacher’s goal is related to the progression of that loss. Similarly to the previous works, we will use a common definition of the loss as the *expected value difference* (EVD):  $L_i = \mathcal{V}^{\pi^*} - \mathcal{V}^{\hat{\pi}_i^L}$  (Abbeel and Ng, 2004; Ziebart, 2010), when evaluated against the real reward  $R^*$ , and define the teaching goal as achieving a certain loss threshold  $\epsilon$  in the lowest number of iterations.

Since the teacher has no access to  $\pi_i^L$ , it has to infer it from the trajectories received during teaching, which corresponds to the problem of Inverse Reinforcement Learning. To infer  $\pi_i^L$  effectively, the teacher has to pick the query states with the highest potential of yielding an informative learner trajectory, which corresponds to the problem of Active Learning. Finally, to achieve the ultimate goal of improving the learner’s policy value, the teacher must select the most informative demonstrations to send, which corresponds to the problem of Machine Teaching. Since a teaching algorithm has to solve these three problems sequentially, it can be divided into three “modules”, each module solving one problem. Figure 1 shows the inputs and outputs of these modules.

We propose a concrete implementation of such a teaching algorithm, which we call Teaching with Limited Feedback (TLimF). It is formally described in Algorithm 2. In the AL module, it uses the Interactive-Value-at-Risk (VaR) algorithm, which is a version of the Active-VaR algorithm (Brown and Niekum, 2019) that we adapted to teaching with limited feedback. In the IRL module, TLimF uses the Interactive-MCE algorithm, which is our adapted version of the MCE-IRL algorithm (Ziebart et al., 2013). Finally, in the MT module, it uses the DSR algorithm (Yengera et al., 2021). We describe all these three algorithms below.

Algorithm 2: Teaching with Limited Feedback (TLimF)

---

```

1: for  $i = 1, \dots, \infty$  do
2: 

---


3:  $s_i^q = \text{Interactive-VaR}(\xi_1^L, \dots, \xi_i^L, \hat{\pi}_{i-1}^L)$   $\triangleright$  AL
   step
4: Send  $s_i^q$  to the learner, receive  $\xi_i^L$ 
5: 

---


6:  $\hat{\pi}_i^L = \text{Interactive-MCE}(\xi_i^L, \hat{\theta}_{i-1}^L)$   $\triangleright$  IRL step
7: 

---


8:  $\xi_i^T = \text{DSR}(\hat{\pi}_i^L)$   $\triangleright$  MT step
9: Send  $\xi_i^T$  to the learner
10: 

---


11: Stop if  $\mathcal{V}^{\pi^*} - \mathcal{V}^{\pi_i^L} < \epsilon$ 
12: end for

```

---

## 4.1 Interactive-MCE

Our algorithm for the IRL module, Interactive-MCE, is based on the MCE-IRL algorithm proposed by (Ziebart et al., 2013). The original algorithm searches for a solution in the class MCE policies,

$$\pi^\theta(a | s) = \exp[\beta Q^{\text{soft}}(s, a) - \beta V^{\text{soft}}(s)], \quad (6)$$

$$Q^{\text{soft}}(s, a) = \langle \theta, \phi(s) \rangle + \gamma \mathbb{E}_{S \sim \mathbb{T}(\cdot | s, a)}[V^{\text{soft}}(S)], \quad (7)$$

$$V^{\text{soft}}(s) = \frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp[\beta Q^{\text{soft}}(s, a')], \quad (8)$$

where  $\beta$  is the entropy factor. For any  $\theta$ , the corresponding MCE policy can be found with the soft-value iteration method (Ziebart, 2010). The MCE-IRL algorithm looks for a parameter  $\theta$  and a policy  $\pi^\theta$  that has the highest likelihood of producing the observed set of trajectories  $\Xi$ , which is a convex problem when the reward is linear. It can be solved with the gradient ascent method, with the gradient equal to

$$\nabla L(\theta) = \frac{1}{|\Xi|} \sum_{\xi} (\mu(\xi) - \mu(\pi^\theta, s_0^\xi)). \quad (9)$$

The original MCE-IRL algorithm assumes that all the available trajectories were generated by a constant policy that is based on a constant reward function. However, in our situation, the trajectories received from the learner are generated by different policies based on different rewards since the learner is assumed to update its reward function after receiving every teacher demonstration. Thus, using all trajectories simultaneously with the MCE-IRL algorithm might infer a reward that is very different from the actual learner’s reward.

We propose a sequential version of this algorithm. At every interaction step  $i$ , this algorithm starts with the previously inferred weights  $\hat{\theta}_{i-1}$  and applies the

MCE gradient ascent with only the new trajectory  $\xi_i$  as the evidence. Unlike a similar algorithm used by the MCE learner in (Kamalaruban et al., 2019), which performs only one MCE iteration per each new trajectory, our variant performs the gradient ascent for many iterations to better utilize the knowledge contained in the trajectories. If the learner’s trajectories are short, this method might overfit to the actions observed in the latest trajectory. To avoid that, the older trajectories could be included in the gradient update, possibly with lower weight, or the feedback might have to be increased to a higher number of trajectories per iteration. Interactive-MCE is formally described in Algorithm 3.

Algorithm 3: Interactive-MCE

---

```

Require: trajectory  $\xi_i$ , previous or initial estimate
            $\hat{\theta}_{i-1}$ 
1:  $s_0 = \text{First-State}(\xi_i)$ 
2:  $\hat{\theta}_i = \hat{\theta}_{i-1}$ 
3:  $\hat{\pi}_i = \text{Soft-Value-Iter}(\hat{\theta}_i)$ 
4: for  $n = 1, \dots, N$  do
5:    $\hat{\theta}_i = \hat{\theta}_i + \eta_n (\mu^{\xi_i} - \mu^{\hat{\pi}_i, s_0})$ 
6:    $\hat{\pi}_i = \text{Soft-Value-Iter}(\hat{\theta}_i)$ 
7: end for
8: return  $\hat{\theta}_i, \hat{\pi}_i$ 

```

---

## 4.2 Interactive-VaR

Our algorithm for the AL module, Interactive-VaR, is based on the Active-VaR algorithm proposed by (Brown and Niekum, 2019). The original algorithm assumes that the MDP is deterministic, the reward weights lie on an L1-norm unit sphere, and the expert is following a constant parametrized *softmax* policy,

$$\pi^\theta(a | s) = \frac{\exp[cQ^\theta(s, a)]}{\sum_{a' \in \mathcal{A}} \exp[cQ^\theta(s, a')]}, \quad (10)$$

where  $c$  is a known confidence factor and  $Q^\theta$  are the Q-values of an optimal policy for  $\theta$ . For any reward weights  $\theta$  on the L1-norm unit sphere, the probability of observing the given set of trajectories  $\Xi$  is

$$\mathbb{P}(\Xi | \theta) = \frac{1}{Z} \exp \left[ \sum_{\xi \in \Xi} \sum_t c Q^\theta(s_t^\xi, a_t^\xi) \right], \quad (11)$$

where  $Z$  is a normalizing constant. If the apriori distribution of  $\theta$  is unknown, the probability of the given weights  $\theta$  generating the observed trajectories is

$$\mathbb{P}(\theta | \Xi) = \frac{1}{Z} \mathbb{P}(\Xi | \theta). \quad (12)$$

For any policy  $\pi$ , weights  $\theta$  and starting state  $s$ , the expected value difference (EVD) of  $\pi$  is defined as

$$\text{EVD}(\theta | \pi, s) = \mathcal{V}^\pi(s) - \mathcal{V}^{\pi^\theta}(s). \quad (13)$$

The Active-VaR method proposes to choose the next query state  $s_i^q$  by finding the state that has the maximum VaR of EVD of the previously inferred policy  $\hat{\pi}_{i-1}$ :

$$s_i^q = \arg \max_{s \in \mathcal{S}_0} \text{VaR}[\text{EVD}(\theta | \hat{\pi}_{i-1}, s)] \quad (14)$$

The original Active-VaR algorithm is not well-suited for the problem in question because the observations were generated by different learner policies, each corresponding to a different reward. One way of addressing this problem is to give less weight to the older observations when computing the likelihood of any  $\theta$ :

$$\mathbb{P}(\theta | \xi_1^L, \dots, \xi_k^L) \approx \frac{1}{Z} \prod_{i=1}^k \mathbb{P}(\xi_i^L | \theta)^{\lambda_i} \quad (15)$$

$$\lambda_k = 1 \quad (16)$$

$$\lim_{i \rightarrow -\infty} \lambda_i = 0 \quad (17)$$

In particular, it is possible to consider only the last  $n$  observations,

$$\mathbb{P}(\theta | \xi_1^L, \dots, \xi_k^L) \approx \frac{1}{Z} \prod_{i=(k-n)^+}^k \mathbb{P}(\xi_i^L | \theta) \quad (18)$$

or to have the weight decay exponentially,

$$\mathbb{P}(\theta | \xi_1^L, \dots, \xi_k^L) \approx \frac{1}{Z} \prod_{i=1}^k \mathbb{P}(\xi_i^L | \theta)^{\lambda^{k-i}}, \lambda < 1. \quad (19)$$

An additional advantage of the exponential decay is computational speed because after receiving a new trajectory, it is possible to compute the updated likelihoods by reusing the likelihoods computed in the previous iteration:

$$\prod_{i=1}^k \mathbb{P}(\xi_i^L | \theta)^{\lambda^{k-i}} = P_k = P_{k-1}^\lambda \mathbb{P}(\xi_k^L | \theta). \quad (20)$$

To avoid using several policy classes within the compound teaching algorithm, we assume that the learner follows an MCE policy instead of a softmax policy. Given that, firstly, we use the soft Q-values of the MCE policy to calculate the demonstration probabilities:

$$\mathbb{P}(\xi | \theta) = \frac{1}{Z} \exp \left[ \sum_t Q^{\text{soft}}(s_t, a_t) \right] \quad (21)$$

Secondly, for calculating VaR, we use the difference of the expected soft values:  $\text{Soft-EVD}(\theta | \pi, s) =$

$V^{\text{soft}, \pi}(s) - V^{\text{soft}, \pi^\theta}(s)$ . Finally, we replace the assumption about the known softmax confidence factor  $c$  with a similar assumption about the known MCE entropy factor  $\beta$ .

Interactive-MCE is formally described in Algorithm 4.

Algorithm 4: Interactive Value-at-Risk (Interactive-VaR)

---

**Require:** previous trajectories  $\xi_1, \dots, \xi_i$ , previous or initial estimate  $\hat{\pi}_{i-1}^L$

- 1: **if**  $i = 1$  **then**
- 2:     Pick a random initial state  $s_i^q \in \mathcal{S}_0$
- 3: **else**
- 4:     Sample reward weights  $\Theta$
- 5:      $s_i^q = \arg \max_{s \in \mathcal{S}_0} \text{VaR}[\text{Soft-EVD}(\Theta | \hat{\pi}_{i-1}^L, s)]$
- 6: **end if**
- 7: **return**  $s_i^q$

---

### 4.3 Difficulty Score Ratio

For deterministic MDPs, the *difficulty score* of a demonstration  $\xi$  w.r.t. a policy  $\pi$  is defined as

$$\Psi(\xi) = \frac{1}{\prod_t \pi(a_t | s_t)} \quad (22)$$

The DSR algorithm selects the next teacher's demonstration  $\xi_i^T$  by iterating over a pool of candidate trajectories  $\Xi$  and finding the trajectory with the maximum difficulty score ratio. DSR is formally described in Algorithm 5.

Algorithm 5: Difficulty Score Ratio (DSR)

---

**Require:** Policy estimate  $\hat{\pi}_i^L$

- 1: **for**  $\xi$  in candidate pool  $\Xi$  **do**
- 2:      $\hat{\Psi}_i^L(\xi) = \prod_t \hat{\pi}_i^L(a_t^\xi | s_t^\xi)^{-1}$
- 3:      $\Psi^T(\xi) = \prod_t \pi^*(a_t^\xi | s_t^\xi)^{-1}$
- 4: **end for**
- 5:  $\xi_i^T = \arg \max_{\xi \in \Xi} \frac{\hat{\Psi}_i^L(\xi)}{\Psi^T(\xi)}$
- 6: **return**  $\xi_i^T$

---

## 5 EXPERIMENTAL EVALUATION

We tested our teaching algorithm in the synthetic car driving environment proposed by (Kamalaruban et al., 2019). The environment consists of 40 isolated roads, each road having two lanes. The agent represents a car that is driving along one of the roads. The

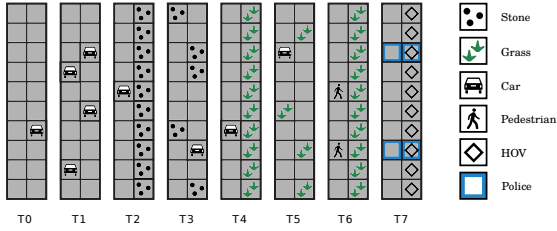


Figure 2: Examples of each road type of the car environment. Each  $2 \times 10$  grid represents a road. The agent starts at the bottom left corner of a randomly selected road. After the agent has advanced for 10 steps upwards along the road, the MDP is terminated.

road is selected randomly at the start of the decision process, and the process terminates when the agent has reached the end of the road. There are eight road types, with five roads of each type. The road types, which we refer to as T0-T7, represent various driving conditions:

- T0 roads are mostly empty and have a few other cars.
- T1 roads are more congested and have many other cars.
- T2 roads have stones on the right lane, which should be avoided.
- T3 roads have cars and stones placed randomly.
- T4 roads have grass on the right lane, which should be avoided.
- T5 roads have cars and grass placed randomly.
- T6 roads have grass on the right lane and pedestrians placed randomly, both of which should be avoided.
- T7 roads have a high-occupancy vehicle (HOV) lane on the right and police at certain locations. Driving on a HOV lane is preferred, whereas the police is neutral.

Each road is represented as a  $2 \times 10$  grid. We assume without loss of generality that only the agent is moving, other objects being static. Roads of the same type differ in the placement of the random objects. Figure 2 demonstrates example roads of all types.

The agent has three actions at every state: left, right, and stay. Choosing left moves the agent to the left lane if it was on the right lane, otherwise moves it to a random lane. Choosing right yields a symmetrical transition. Choosing stay keeps the agent on the same lane. Regardless of the chosen action, the agent always advances along the road. The environment has 40 possible initial states, each corresponding to the bottom left corner of every road. After advancing along the road for ten steps, the MDP is terminated. We assume  $\gamma = 0.99$ .

Table 1: True feature weights

Feature	Weight
stone	-1
grass	-0.5
car	-5
pedestrian	-10
HOV	+1
police	0
car-in-front	-2
ped-in-front	-5

For every state, eight binary features are observable. Six of them represent the environment objects: stone, grass, car, pedestrian, HOV, and police. The last two indicate whether there’s a car in the next cell or a pedestrian. We consider the reward to be a linear function of these binary features, with reward weights specified in Table 1.

## 5.1 CrossEnt-BC Learner

We use a linear variant of the Cross-Entropy Behavioral Cloning (CrossEnt-BC) learner proposed by (Yengera et al., 2021) for our experiments. This learner follows a parametrized softmax policy,

$$\pi_i^{\text{CE}}(a | s) = \frac{\exp[H_i(s, a)]}{\sum_{a' \in \mathcal{A}} \exp[H_i(s, a')]}, \quad (23)$$

where  $H_i$  is a parametric scoring function that depends on a parameter  $\theta_i^L$  and a constant feature mapping,

$$\phi^{\text{CE}}(s, a) = \mathbb{E}_{S' \sim \mathbb{T}(\cdot | s, a)}[\phi(S')], \quad (24)$$

and is defined as  $H_i(s, a) = \langle \theta_i^L, \phi^{\text{CE}}(s, a) \rangle$ . The likelihood of any demonstration  $\xi$  and its gradient are defined respectively as

$$L(\theta_i^L) = \log \mathbb{P}(\xi | \theta_i^L), \quad (25)$$

$$\nabla L(\theta_i^L) = \sum_t \left( \phi^{\text{CE}}(s_t^\xi, a_t^\xi) - \mathbb{E}_{a \sim \pi_i^{\text{CE}}(\cdot | s_t^\xi)} \left[ \phi^{\text{CE}}(s_t^\xi, a) \right] \right). \quad (26)$$

This learner starts with random initial weights  $\theta_i^L$ , every element being uniformly sampled from  $(-10, 10)$ . Upon receiving a new demonstration  $\xi_i^T$  from the teacher, the learner performs a projected gradient ascent,

$$\theta_{i+1}^L = \text{Proj}_{\Theta} [\theta_i^L + \eta \nabla L(\theta_i^L)], \quad (27)$$

where  $\eta = 0.34$  and  $\Theta$  is a hyperball centered at zero with a radius of 100.

Table 2: Tested algorithms

<i>Name</i>	<i>AL</i>	<i>IRL</i>	<i>MT</i>
RANDOM	-	-	Random
NOAL	Random	Interactive-MCE	DSR
UNMOD	Active-VaR	MCE-IRL	DSR
TLIMF	Interactive-VaR	Interactive-MCE	DSR
TUNLIMF	Not needed	Not needed	DSR

## 5.2 Teaching Algorithms

We compared the following algorithms, also presented in Table 2:

- RANDOM teacher does not infer  $\theta_i^L$  and selects demonstrations by choosing a random initial state and generating an optimal demonstration from that state. This algorithm was originally proposed in (Kamalaruban et al., 2019) and serves as the worst-case baseline.
- NOAL teacher selects query states randomly but uses MCE to infer the learner reward and DSR to select demonstrations. We included this algorithm as the second worst-case baseline to verify whether the usage of an AL algorithm by other teachers can boost the teaching process.
- UNMOD uses unmodified Active-VaR to select query states, followed by unmodified MCE-IRL and DSR. We included this algorithm to verify whether the changes that we introduced in Interactive-VaR and Interactive-MCE affect the performance.
- TLIMF uses Interactive-VaR to select query states, followed by Interactive-MCE and DSR.
- TUNLIMF teacher knows the exact learner’s policy at every step and therefore does not need AL and IRL modules. It uses the DSR algorithm to select demonstrations. This algorithm was originally proposed in (Yengera et al., 2021) and serves as the best-case baseline.

We did not include non-interactive algorithms in the experiment, because it was shown in (Yengera et al., 2021) that the state-of-the-art non-interactive MT algorithm, Set Cover Optimal Teaching, did not perform better than RANDOM in this environment. We also did not include the Black-Box (BBox) algorithm of (Kamalaruban et al., 2019) in the comparison, because it was shown in (Yengera et al., 2021) that TUNLIMF has similar performance to BBox and can be considered an improvement over it.

The Interactive-VaR algorithm samples reward weights from the L1-norm sphere with a radius equal to 24, which is the L1-norm of the true feature

weights. The VaR is computed on 5,000 uniformly sampled weights on the sphere<sup>2</sup>. For computing the posterior likelihood of  $\theta$ , the demonstrations are weighted exponentially with  $\lambda = 0.4$ . The EVD between the two policies is computed using soft policy values. The  $\alpha$  factor of VaR was set to 0.95. The MCE algorithms use 100 iterations of the gradient ascent. The DSR algorithm selects demonstrations from a constant pool that consists of 10 randomly sampled trajectories per road.

## 5.3 Analysis of the Teacher’s Performance

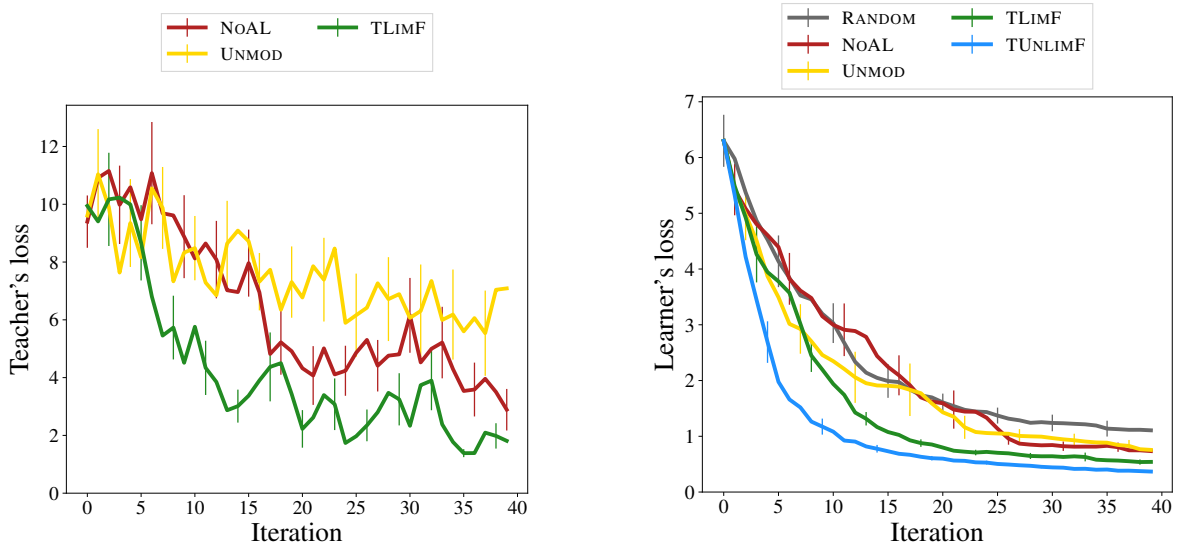
We conducted the experiment 16 times with different random seeds, which affected the random placement of objects on the roads and the random initial weights of the learners, and averaged the results of 16 experiments.

Figure 3a displays the ability of the teaching algorithms to accurately estimate the current learner’s policy. For every iteration step, it shows the loss of the teacher’s inferred policy  $\hat{\pi}_i^L$  w.r.t. the actual learner’s policy  $\pi_i^L$ . The thick lines represent the average of 16 experiments, and the thin vertical lines measure the standard error. As we can see, at any iteration, TLIMF is able to estimate the learner’s policy more reliably than NOAL, which means that using an AL algorithm is crucial for effectively estimating the learner’s policy. We can also see that UNMOD performs considerably worse than NOAL, which means that unmodified Active-VaR and MCE-IRL algorithms are not suitable for teaching with limited feedback. The teacher’s performance in estimating the learner’s policy is an intermediate result that affects the overall teaching performance, which is discussed next.

Figure 3b and table 3 display the effectiveness of the teacher’s effort in teaching the learner. For every iteration step, figure 3b shows the loss of the learner’s policy  $\pi_i^L$  w.r.t. the optimal policy  $\pi^*$ , averaged over 16 experiments. Table 3 shows how many iterations, on average, the teachers need before reaching various

<sup>2</sup>Increasing the sample size or sampling with MCMC yields similar results.





(a) Teacher's inferred policy loss.

(b) Learner's policy loss.

Figure 3: Teaching results are measured as (a) the loss of the teacher's inferred policy and (b) the loss of the actual learner's policy. The thick lines represent the averages of 16 experiments. The thin vertical lines measure the standard error. TLIMF demonstrates the lowest losses during most of the process, with NOAL and UNMOD significantly lagging behind.

loss thresholds. As we can see, TLIMF does not attain the performance of the upper baseline, TUNLIMF, but it performs considerably better than other teaching algorithms: its loss is consistently lower starting from the seventh iteration, and it needs considerably fewer iterations to reach the presented loss thresholds. This implies that *for teaching with limited feedback, the best performance is achieved when the teacher is using specialized AL and IRL algorithms to select query states and infer the learner's policy*. The NOAL teacher performs worse than TLIMF but better than the lower baseline, RANDOM: its loss is considerably lower starting from the 25th iteration, and it needs fewer iterations to reach the loss thresholds. This implies that *teaching without AL is still better than selecting demonstrations randomly*. Finally, the performance of the teacher with unmodified AL and IRL algorithms, UNMOD, is high during the first six iterations, but it gradually worsens during the teaching process and falls below the performance of NOAL. It also shows significantly low performance at reaching the loss thresholds, needing more iterations than the random teacher, which implies that modifying the algorithms was necessary for good performance.

Table 3: Iterations needed to reach a loss threshold  $\epsilon$ 

Teacher	$\epsilon = 2$	$\epsilon = 1$	$\epsilon = 0.5$
RANDOM	16	36	96
NOAL	13	25	49
UNMOD	31	54	102
TLIMF	8	19	46
TUNLIMF	5	11	27

## 6 SUMMARY AND FUTURE WORK

We have proposed a teacher-learner interaction framework in which the feedback from the learner is limited to just one trajectory per teaching iteration. Such a framework is closer to real-life situations and more challenging when compared with the frameworks used in previous works. In this framework, the teacher has to solve AL, IRL, and MT problems sequentially at every teaching iteration. We have proposed a teaching algorithm that consists of three modules, each dedicated to solving one of these three sub-problems. This algorithm uses a modified MCE-IRL algorithm for solving the IRL sub-problem, a modified Active-VaR algorithm for solving the AL problem, and the DSR algorithm for solving the MT problem. We have tested the algorithm on a synthetic car-driving environment and compared it with the existing algorithms and the worst-case baseline. We have concluded that the new algorithm is effective at solving the teaching problem.

In future work, it would be interesting to study such a teacher-learner interaction in more complex environments. For example, an environment could have more states and a non-linear reward function possibly represented as a neural network. Another question yet to be addressed is the convergence guarantees of the proposed algorithms. It is also interesting to check whether the MT module of the algorithm could be improved by considering the uncertainty of the estimated learner policy. Another possible direction of research is finding more sophisticated ways of weighing older trajectories of the learner. E.g., if the environment consists of several isolated regions and any feature is confined to a certain region, then sending a teaching demonstration in one region might not change the learner’s behavior in others, therefore the previous learner’s trajectories from other regions might not need to be weighed down.

## ACKNOWLEDGEMENTS

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020 (INESC-ID multi-annual funding) and the RELEvaNT project, with reference PTDC/CCI-COM/5060/2021.

Rustam Zayanov would also like to thank Open Philanthropy for their scholarship, which facilitated his dedicated involvement in this project.

## REFERENCES

- Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1.
- Brown, D. S., Cui, Y., and Niekum, S. (2018). Risk-aware active inverse reinforcement learning. In *Conference on Robot Learning*, pages 362–372. PMLR.
- Brown, D. S. and Niekum, S. (2019). Machine teaching for inverse reinforcement learning: Algorithms and applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7749–7758.
- Cakmak, M. and Lopes, M. (2012). Algorithmic and human teaching of sequential decision tasks. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.
- Kamalaruban, P., Devidze, R., Cevher, V., and Singla, A. (2019). Interactive teaching algorithms for inverse reinforcement learning. *arXiv preprint arXiv:1905.11867*.
- Liu, W., Dai, B., Humayun, A., Tay, C., Yu, C., Smith, L. B., Rehg, J. M., and Song, L. (2017). Iterative machine teaching. In *International Conference on Machine Learning*, pages 2149–2158. PMLR.
- Liu, W., Dai, B., Li, X., Liu, Z., Rehg, J., and Song, L. (2018). Towards black-box iterative machine teaching. In *International Conference on Machine Learning*, pages 3141–3149. PMLR.
- 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*, pages 31–46. Springer.
- Melo, F. S., Guerra, C., and Lopes, M. (2018). Interactive optimal teaching with unknown learners. In *IJCAI*, pages 2567–2573.
- Ng, A. Y., Russell, S., et al. (2000). Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2.
- Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Yengera, G., Devidze, R., Kamalaruban, P., and Singla, A. (2021). Curriculum design for teaching via demonstrations: Theory and applications. *Advances in Neural Information Processing Systems*, 34:10496–10509.
- Zhu, X. (2015). Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.
- Zhu, X., Singla, A., Zilles, S., and Rafferty, A. N. (2018). An overview of machine teaching. *arXiv preprint arXiv:1801.05927*.
- Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.
- Ziebart, B. D., Bagnell, J. A., and Dey, A. K. (2013). The principle of maximum causal entropy for estimating interacting processes. *IEEE Transactions on Information Theory*, 59(4):1966–1980.