TÉCNICO LISBOA

Interactively Teaching an Inverse Reinforcement Learner with Limited Feedback

Rustam Zayanov

Thesis to obtain the Master of Science Degree in

Data Science and Engineering

Supervisors:
Prof. Francisco António Chaves Saraiva de Melo
Prof. Manuel Fernando Cabido Peres Lopes

Examination Committee

Chairperson: Prof. Maria do Rosário De Oliveira Silva
Supervisor: Prof. Francisco António Chaves Saraiva de Melo
Member of the Committee: Prof. Pedro Alexandre Simões dos Santos

December 2022

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to express my gratitude to my supervisors, Professor Francisco Melo and Professor Manuel Lopes, for their valuable feedback and guidance. I would also like to thank Open Philanthropy for providing me with a scholarship to work on this thesis.

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. We propose two teaching algorithms that both use the principle of maximum causal entropy to infer the policy. The first algorithm uses an adapted version of the active value-at-risk method to select the starting states. The second algorithm queries the learner with the starting states of the previous demonstrations. Both algorithms use the difficulty score ratio method to choose the teaching demonstrations. We test both algorithms in a synthetic car driving environment and conclude that both are viable solutions when the learner's feedback is limited.

Keywords

Sequential Decision Processes, Inverse Reinforcement Learning, Machine Teaching, Interactive Teaching and Learning

Resumo

Estudamos o problema do ensino por demonstrações em tarefas de tomada de decisão sequencial. Em particular, abordamos a situação em que o professor não tem acesso ao modelo e política do aluno, e o *feedback* do aluno é limitado a trajetórias que começam a partir de estados selecionados pelo professor. A necessidade de selecionar os estados iniciais e inferir a política do aluno cria uma oportunidade de usar métodos de aprendizagem por reforço inverso e aprendizagem ativa pelo professor. Propomos dois algoritmos de ensino que utilizam o princípio da entropia causal máxima para inferir a política. O primeiro algoritmo usa uma versão adaptada do método *active value-at-risk* para selecionar os estados iniciais. O segundo algoritmo inquire o aluno usando os estados iniciais das demonstrações anteriores. Ambos os algoritmos utilizam o método de *difficulty score ratio* para escolher as demonstrações de ensino. Testamos ambos os algoritmos num ambiente sintético de condução e concluímos que ambos são soluções viáveis quando o *feedback* do aluno é limitado.

Palavras Chave

Processos de Decisão Sequenciais, Aprendizagem por Reforço Inverso, Ensino Automático, Ensino e Aprendizagem Interativos

Contents

List of Figures

List of Tables

Acronyms

AL active learning

BEC behavioral equivalence class

CrossEnt-BC cross-entropy behavioral cloning

DBAL demonstration-based active learning

DBAL-T DBAL-based teaching

DSR difficulty score ratio

EVD expected value difference

HOV high-occupancy vehicle

IRL inverse reinforcement learning

MCE maximum causal entropy

MCE-IRL maximum causal entropy inverse reinforcement learning

MCMC Markov chain Monte Carlo

MDP Markov decision process

MT machine teaching

RndAL random active learning

VaR Value-at-Risk

VaR-T VaR-based teaching

1

Introduction

Contents

Machine teaching (MT) 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 [24, 25], 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) [1, 17]. 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 specific IRL algorithm to process the teacher's demonstrations, the teacher could pick an optimal demonstration sequence for that algorithm, i.e., the shortest sequence that is sufficient for the learner to infer the goal of the task.

For MT algorithms to be usable, the teacher usually needs to know the learner 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. When the teacher makes very few or no assumptions about the learner, such a learner model is often called a "black-box."

For example, in the sequential decision-making domain, Kamalaruban et al. [10] and Yengera et al. [23] 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 trajectories during the teaching process, thus making the process iterative and interactive. Both works assume that the teacher can periodically observe many learner's trajectories from *every initial state* and thus estimate the learner's policy with high precision and with this extract information about the learner's learning process. Unfortunately, the need to produce trajectories from every initial state could impose an excessive load on the learner in real-life scenarios. In the case of a human learner, this could require the learner to take a broad range of tests before receiving a new lesson.

In our present work, we address a more realistic scenario in which the feedback from the learner is limited to just one trajectory at every iteration of the teaching process. Such limited feedback, however,

poses a challenge in reliably estimating the learner's policy.

Thus, our research question is the following: What are the effective ways of teaching an inverse reinforcement learner when the feedback from the learner is limited?

The teacher's ability to correctly estimate the policy greatly depends on the informativeness of the trajectory received. Given this limitation, we consider a teacher that can query the learner for trajectories that start from specific states chosen by the teacher. This ability to choose initial states is similar to a real-life examination process when a human examiner can choose the next question based on the student's previous answers. The necessity to select the starting states and infer the learner's policy creates an opportunity to use methods of IRL and active learning (AL) by the teacher. AL, when studied in the context of sequential decision-making [14], considers the situation when a learner has to infer an expert's reward from demonstrated trajectories and can interactively choose the states from which the demonstrations should start.

1.1 Contributions

Firstly, we propose a new interactive teaching framework that formalizes the interaction between the teacher and the learner when the learner's feedback is limited. It is based on the framework of Kamalaruban et al. [10] and extends it with a teacher's capacity of sending query states to the learner and receiving trajectories that start from those states.

Then we propose two teaching algorithms for the new framework. Both algorithms consist of three steps: selection of the query state with an AL method, inference of the current learner's policy with an IRL method, and selection of the best teaching demonstration with an MT method. While both algorithms use the maximum causal entropy (MCE) [28] method for inferring the learner's policy and the *difficulty score ratio* (DSR) method Yengera et al. [23] for selecting the demonstration, they differ in how they choose the query states. The first algorithm, Interactive Value-at-Risk (VaR), is an adapted version of the risk-minimizing AL algorithm proposed by Brown et al. [3], which we modify to make it suitable for the new framework. The second algorithm, demonstration-based active learning (DBAL), is a simpler algorithm that chooses query states equal to the first state of the previous teacher's demonstration.

Finally, we test both algorithms in a synthetic car driving environment and conclude that both are viable solutions when the learner's feedback is limited. We test two cases: in the first case, the teacher does not know the initial policy of the learner, and in the second, the teacher knows the exact initial policy.

1.2 Document structure

In the next chapter, we provide the theoretical background on sequential decision-making, IRL, MT, and AL.

In Chapter 3, we review existing literature and describe some of the previously proposed algorithms for solving the MT, IRL, and AL problems in isolation.

In Chapter 4, we formalize the interaction between the teacher and the learner and propose two teaching algorithms to solve the problem of interactive teaching with limited feedback.

In Chapter 5, we compare the two new algorithms with the best and worst baselines in a synthetic car environment and analyze the results.

Finally, in Chapter 6, we summarize the results and suggest some directions for future work.

2

Background

Contents

2.1 Sequential decision-making

The underlying task to be solved by the agent is formally represented as a Markov decision process (MDP) denoted as $M = (\mathcal{S}, \mathcal{A}, \mathbb{T}, \mathbb{P}_0, \gamma, R^\star)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathbb{T}(s' \mid s, a)$ is the state transition probability upon taking action a in state s, $\mathbb{P}_0(s)$ is the initial state distribution, γ is the discount factor, and $R^\star : \mathcal{S} \to \mathbb{R}$ is the reward function to be learned.

At the start of the process (time step $t = 0$), the agent observes the surrounding environment in a state s_0 which is drawn randomly according to the initial state distribution $\mathbb{P}_0(s \in \mathcal{S})$. At every given time step t, the agent observes the environment at state s_t and can take any action $a_t \in \mathcal{A}$. Upon taking that action, the agent receives a reward $r_t = R^\star(s_t)$, and the environment transitions to a new state s_{t+1}, which is drawn randomly, according to the transition probability $\mathbb{T}(\cdot \mid s_t, a_t)$. After the transition is complete, the agent can choose its next action. The process continues indefinitely, but if M has absorbing states (i.e. those that always transition to themselves), then such states can be called *terminal states*, and the process can be terminated once such a state is reached. The agent's goal is to take a sequence of actions that maximize the total discounted expected reward,

$$\mathbb{E}_{S_0 \sim \mathbb{P}_0} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \mid \mathbb{T} \right],$$

where $\mathbb{E}(\cdot)$ denotes an expected value of a random variable. The Markov decision process is a decision process that satisfies the Markov property: at each time step t, the reward r_t and the new state s_{t+1} depend only on the current state s_t and action a_t and do not depend on any other previous states or actions.

An MDP is called *deterministic* if all transition probabilities are deterministic:

$$\forall s, a : \exists s' : \mathbb{T}(s' \mid s, a) = 1$$

A stationary *policy* is a mapping π that maps each state $s \in \mathcal{S}$ into a probability distribution $\pi(\cdot \mid s)$ over $\mathcal{A}$. A policy can be executed in M, which will produce a sequence of state-action pairs called *trajectory*. For any trajectory $\xi = \{s_0, a_0, \ldots, s_T, a_T\}$, we will denote its first state as s_0^ξ. Given a policy π, 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:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \mid \pi, \mathbb{T}, S_0 = s \right] \\ \mathcal{V}^\pi &= \mathbb{E}_{S \sim \mathbb{P}_0}[V^\pi(S)] \\ Q^\pi(s, a) &= R(s) + \gamma \mathbb{E}_{S \sim \mathbb{T}(\cdot \mid s, a)}[V^\pi(S)] \end{aligned}$$

Algorithm 1 Policy iteration

Require: MDP M

$V(s) = 0, \forall s$
while $V(\cdot)$ not converged **do**
    $Q(s,a) = R(s) + \gamma \mathbb{E}_{S \sim \mathbb{T}(\cdot|s,a)}[V(S)], \forall s, a$
    $V(s) = \max_a Q(s,a), \forall s$
end while
$\pi(a \mid s) = \frac{\mathbb{I}(a \in \arg\max_{a'} Q(s,a'))}{|\arg\max_{a'} Q(s,a')|}$

A policy $\pi^\star$ 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 policy iteration [20], described in Algorithm 1. This algorithm starts by initializing all state-values to zero and iteratively updates Q-values and state-values until state-values converge. Finally, for every state, it assigns an equal non-zero policy probability to all actions that have the highest Q-value for that state.

It is common in the literature to assume that the environment also has d numerical *features* associated with every state, formalized as a mapping $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^d$, and that the reward can be expressed as a function of those features. For example, it is common to assume that the reward can be expressed as the linear combination of the state features, $R^\star(s) = \langle \boldsymbol{\theta}, \boldsymbol{\phi}(s) \rangle$, where $\boldsymbol{\theta}$ is called the *reward weight vector* or simply *reward weights*. Since $\boldsymbol{\theta}$ fully defines $R^\star$, the terms "reward" and "reward weights" can be used interchangeably. For a given trajectory ξ or for a set of trajectories Ξ, the discounted *feature count vector* is defined as follows respectively:

$$\boldsymbol{\mu}(\xi) = \sum_t \gamma^t \boldsymbol{\phi}(s_t)$$
$$\boldsymbol{\mu}(\Xi) = \frac{1}{|\Xi|} \sum_\xi \boldsymbol{\mu}(\xi)$$

For a given policy π and a state s, the discounted expected feature counts are defined as follows:

$$\boldsymbol{\mu}(\pi, s) = \mathbb{E}\left[\sum_t \gamma^t \boldsymbol{\phi}(S_t) \mid \pi, \mathbb{T}, S_0 = s\right]$$
$$\boldsymbol{\mu}(\pi) = \mathbb{E}_{S \sim \mathbb{P}_0}[\boldsymbol{\mu}(\pi, S)]$$

2.2 Inverse Reinforcement Learning

Many real-life tasks can be expressed as an MDP, but for some tasks, expressing a reward function may be difficult. Instead, it might be easier to obtain many near-optimal trajectories. The task of recovering the reward function from trajectories corresponds to the problem of inverse reinforcement learning.

Formally, IRL considers two entities that have access to the environment: an *expert* that has full

access to M, and a *learner* that can access all elements of M except for the reward function, denoted as $M \setminus R$. The learner has access to a set Ξ of optimal or near-optimal trajectories generated by the expert, also called *demonstrations*. The learner's task is to find a reward function that could "explain" the given demonstrations.

If the demonstrations are known to be generated by an optimal policy, then the goal of IRL is to find a reward function whose optimal policy has a non-zero chance of generating all trajectories in Ξ. This problem is ill-posed because there may exist many reward functions that satisfy this condition, including an all-zero reward [17]. Several ways of solving this ambiguity were proposed. For example, one way is to find a reward whose optimal policy maximizes the difference between Q-values of the observed actions and the rest [17],

$$\hat{R}^\star = \arg\max_{R \in \mathcal{R}} \sum_{\xi} \sum_{t} \left(Q^{\pi^\star_R}(s_t, a_t) - \max_{a \in A \setminus a_t} Q^{\pi^\star_R}(s_t, a) \right)$$

where $\mathcal{R}$ is a set of possible rewards, assumed to be known to the learner beforehand.

However, it is often likely that the demonstrations were not generated by an optimal policy of some reward function. In this case, the common approach is to assume that the expert policy belongs to a certain class of policies and to find the reward that yields the most likely policy in that class. We describe some of these approaches in Chapter 3.

2.3 Active Learning

Active learning considers the situations when the learner can choose the data to learn from. It is useful when the training data is hard to generate or transmit to the learner. For example, in the case of classification tasks, after having received n labeled examples, the learner could choose which unlabeled examples should be labeled next and added to the training set.

In the case of AL for IRL, instead of having a fixed set of demonstrations Ξ to learn from, the learner could interactively query the expert for trajectories that start from states selected by the learner. The interaction between the learner and the expert is described in Algorithm 2. At every iteration, the learner sends a query state s_i^{q} to the expert, receives a demonstration ξ_i that starts from the query state, and uses the demonstration to update its reward estimation $\hat{R}^\star_i$. The interaction continues until the *learning goal* is achieved. The formulation of the learning goal depends on the AL algorithm in use. For example, the stopping condition of the Active-VaR algorithm (section 3.3) depends on the learner's estimation of the worst-case mismatch between the true policy and the inferred policy.

We describe some AL algorithms in Chapter 3, and a more detailed survey of AL algorithms is presented in [19].

Algorithm 2 Active Learning in Inverse Reinforcement Learning

for $i = 1, \ldots, \infty$ **do**
    Learner sends a *query state* s_i^{q} and requests a trajectory starting from it
    Expert generates and sends a *demonstration* ξ_i to the learner
    Learner updates its reward estimate $\hat{R}_i^\star$
    Stop if the learning goal is achieved
end for

Algorithm 3 Interactive teaching for Inverse Reinforcement Learning

for $i = 1, \ldots, T$ **do**
    Teacher observes an estimate of the learner's current policy $\hat{\pi}_i^L$
    Teacher generates and sends a *demonstration* ξ_i to the learner
    Learner updates its reward estimate $\hat{R}_i^\star$
end for

2.4 Machine Teaching

Machine teaching studies the teaching process from the teacher's point of view. The teacher's goal is to find an optimal (often the shortest) sequence of examples or demonstrations that is sufficient for teaching the target concept to the given learner [7]. If the learner's learning algorithm is fully known to the teacher, no interaction between the teacher and the learner is necessary, and the optimal teaching set can be computed and sent to the learner in one batch. When the learner's model is not fully known [5], the teacher is allowed to interactively receive information from the learner, which it could use to select the next examples.

Interactive teaching for IRL tasks was first studied by Kamalaruban et al. [10]. Their proposed mode of interaction can be found in Algorithm 3. At every iteration, the teacher observes the estimate of the learner's current policy $\hat{\pi}_i^L$ and sends a demonstration ξ_i to the learner, and the learner uses the demonstration to update its reward estimation $\hat{R}_i^\star$.

We describe some MT algorithms in Chapter 3, and a more detailed overview of MT literature can be found in [25].

3

Related work

Contents

In this chapter, we describe the IRL, AL, MT methods that can be used in solving the MT problem with limited feedback. We also describe works that consider uncertainty about the learner in contexts other than interactive teaching for IRL, such as classification tasks and non-interactive teaching for IRL

3.1 Inverse Reinforcement Learning

The IRL problem was first introduced by Ng et al. [17]. In the original problem setting, the entities are called the *expert* and the *learner*, with the expert merely being the source of demonstrations, and the learner's goal being to infer the expert's reward, R^E, given the policy π^E or demonstrations Ξ. The authors define a "meaningful" $\hat{R}^E$ as the one which maximizes the difference between Q-values of the observed actions and the rest. The resulting maximization problem is linearly solvable. They also consider the special case when R^E is a linear combination of state features, which is also linearly solvable. Finally, they consider the situation when π^E is not available directly, but the learner can request expert trajectories from any state. The authors have laid the theoretical foundations for IRL, but their approach had limited application in practice since the real-life expert trajectories can often be non-optimal.

Abbeel and Ng [1] further explore the situation when only demonstrations are available, and R^E is a linear combination of features. They assume that the expected feature counts can be reliably estimated from Ξ, and the reward weights have a known boundary. With these assumptions, they measure the loss of the learner as the distance of the expected feature counts: $\left\|\boldsymbol{\mu}(\Xi) - \boldsymbol{\mu}(\pi^L)\right\|$. They propose the max-margin and the projection methods that do not necessarily recover the true reward but return a set of policies that either have to be examined by a human expert or "mixed" together to attain a performance similar to the optimal policy.

Ramachandran and Amir [18] study the situation when the expert demonstrations are nearly-optimal. In particular, they assume that the demonstrations were generated by a *softmax* policy, which assigns higher probabilities to the actions that have the highest Q-values of an optimal policy. Given such an assumption, the probability of generating a set of demonstrations given a reward function is well-defined. By applying the Bayes rule, it is possible to express the probability of a reward function given a set of demonstrations. Authors propose solving the IRL problem by finding the mean or the median of the resulting distribution over the space of reward functions. Calculating the exact a posteriori probabilities is computationally expensive, so authors propose obtaining a sample of reward functions via the Markov chain Monte Carlo (MCMC) method. They also propose `PolicyWalk`, an effective method of finding Q-values for a given reward function based on Q-values of a nearby reward function. The Bayesian approach to IRL serves as a basis for a number of AL algorithms for IRL [3, 14], including the Active VaR algorithm (section 3.3), whose modified version we use for selecting query states in one of our teaching

algorithms.

3.2 Entropy-based IRL

Ziebart et al. [27] were the first to introduce the entropy-based IRL formulation. The authors focus on recovering the reward function from sub-optimal demonstrations, for which the previous IRL methods do not work well. Similarly to [1], the reward is assumed to be a linear function of features, and the learner's goal is to find a policy that matches the feature counts.

The entropy-based approach is further researched in [26, 28], where the authors propose solving IRL with the maximum causal entropy (MCE-IRL) method.

The MCE-IRL approach relies on the notion of a *policy entropy*. Given a state s or a trajectory ξ, it is defined as follows respectively:

$$H^{\pi}(s) = -\sum_{a \in \mathcal{A}} \pi(a \mid s) \log \pi(a \mid s)$$
$$H^{\pi}(\xi) = \sum_{t} \gamma^{t} H^{\pi}(s_t)$$

For a linear reward, MCE-IRL only considers policies whose expected feature counts are equal to the feature counts of the demonstration: $\boldsymbol{\mu}(\pi, s_0^{\xi}) = \boldsymbol{\mu}(\xi)$, where s_0^{ξ} is the first state of ξ. Many policies may satisfy this condition, and some may express additional preferences not supported by the evidence in ξ. To avoid making such unjustified preferences, the MCE principle prescribes choosing a policy that has the maximum entropy on ξ:

$$\pi = \arg\max_{\pi'} H^{\pi'}(\xi)$$

If π is the solution to this optimization problem, then, under mild technical conditions, there exist some feature weights $\boldsymbol{\theta}$, such that the *Soft Bellman equations* ([26], Algorithm 9.1) hold:

$$Q^{\text{soft}}(s, a) = \langle \boldsymbol{\theta}, \boldsymbol{\phi}(s) \rangle + \gamma \mathbb{E}_{S \sim \mathbb{T}(\cdot \mid s, a)}[V^{\text{soft}}(S)]$$
$$V^{\text{soft}}(s) = \frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp[\beta Q^{\text{soft}}(s, a')]$$
$$\pi(a \mid s) = \exp[\beta Q^{\text{soft}}(s, a) - \beta V^{\text{soft}}(s)]$$

where β is the entropy factor that specifies the magnitude of $\boldsymbol{\theta}$, often implied to be equal to 1. For any given feature weights $\boldsymbol{\theta}$, the policy that takes this form is called an MCE policy, and it is possible to compute it via the soft-value iteration method (Algorithm 6).

Maximizing the entropy implies maximizing the log-likelihood of the observed trajectory, i.e., $L(\boldsymbol{\theta}) =$

$\log \mathbb{P}(\xi \mid \boldsymbol{\theta})$. This optimization problem is convex in the space of feature weights $\boldsymbol{\theta}$. Therefore, the solution can be found with convex optimization techniques, e.g., the gradient ascent method. The gradient at any point $\boldsymbol{\theta}$ is the difference of the feature counts: $\nabla L(\boldsymbol{\theta}) = \boldsymbol{\mu}(\xi) - \boldsymbol{\mu}(\pi^{\boldsymbol{\theta}}, s_0^{\xi})$, where $\pi^{\boldsymbol{\theta}}$ is the MCE policy for $\boldsymbol{\theta}$.

In our present work, we propose teaching algorithms that use a modified version of the MCE-IRL approach to infer the learner's policy.

Levine et al. [11] explores MCE-IRL solutions for nonlinear reward functions.

Jacq et al. [8] consider the situation when a learner A is observing trajectories generated by another learner B. In this situation, the true reward is unknown for both learners, and the learner B gradually improves its MCE policy as it collects more data about the environment, sequentially adopting a chain of policies $\pi_1^B, \pi_2^B, \ldots, \pi_i^B$. The authors show that if the learner A can access two such policies, then it can restore the true reward function up to a shaping. If the learner A can only access demonstrations instead of policies, the authors propose estimating the corresponding policies with MCE-IRL and using them for estimating the true reward.

3.3 Active IRL

The problem of active learning in sequential decision settings was introduced by Lopes et al. [14]. In their formulation, the learner assumes that the expert's demonstrations were generated by a softmax policy. The learner uses the Bayesian approach to solve the IRL problem [18], which defines the posterior reward probability $\mathbb{P}(R \mid \Xi)$ through a prior probability $\mathbb{P}(R)$ and the likelihood of observing the demonstrations, $\mathbb{P}(\Xi \mid R)$. The learner can interactively ask the teacher for actions that would be chosen at specific states. The goal of the learner is to sequentially query the states in such a way that the uncertainty about $\mathbb{P}(R \mid \Xi)$ is reduced as quickly as possible. They propose two algorithms for this problem, one based on Monte Carlo sampling of the reward space and the other based on gradient ascent. In the problem of teaching with limited feedback, there is a possibility for the teacher to select query states with a modified version of either of these two algorithms, which is a possible direction for further research.

Brown et al. [3] approached the problem from the perspective of risk minimization and proposed the Active Value-at-Risk (Active-VaR) method.

Like most IRL research, this algorithm assumes that all given trajectories are generated by a constant policy of an expert. It additionally assumes that the MDP is deterministic, the reward weights in question lie on an L1-norm unit sphere, and the agent is following a *softmax* policy with a known confidence factor c. For any reward function R, the corresponding softmax policy π^{SM} is defined as

$$\pi^{\mathsf{SM}}(a \mid s) = \frac{\exp[cQ^{\star}(s,a)]}{\sum_{a' \in \mathcal{A}} \exp[cQ^{\star}(s,a')]},$$

where $Q^\star$ are the Q-values of an optimal policy for R.

For any reward weights $\boldsymbol{\theta}$ on the L1-norm unit sphere, the probability of observing the given set of trajectories Ξ is

$$\mathbb{P}(\Xi \mid \boldsymbol{\theta}) = \prod_{\xi} \prod_{t} \pi^{\mathsf{SM}}(a_t \mid s_t) = \frac{1}{Z} \exp[\sum_{\xi} \sum_{t} cQ(s_t, a_t)],$$

where Z is a normalizing constant.

The algorithm is based on the theory of Bayesian IRL [18]. By applying the Bayes rule, the probability of the given weights $\boldsymbol{\theta}$ generating the observed trajectories is

$$\mathbb{P}(\boldsymbol{\theta} \mid \Xi) = \frac{1}{Z'} \mathbb{P}(\boldsymbol{\theta}) \mathbb{P}(\Xi \mid \boldsymbol{\theta}),$$

where $\mathbb{P}(\boldsymbol{\theta})$ represents the apriori assumptions about the reward function. In the absence of prior knowledge, according to the maximum entropy principle, $\mathbb{P}(\boldsymbol{\theta})$ is assumed to be uniform.

For any fixed policy π, weights $\boldsymbol{\theta}$ and starting state s, the *policy loss* of π is defined as the expected value difference, i.e.,

$$L = \mathrm{EVD}(\boldsymbol{\theta} \mid \pi, s) = \mathcal{V}^{\pi}(s) - \mathcal{V}^{\pi^{\boldsymbol{\theta}}}(s),$$

where $\pi^{\boldsymbol{\theta}}$ is an optimal policy for $\boldsymbol{\theta}$. Since $\boldsymbol{\theta}$ is unknown and treated as a random variable, the expected value difference (EVD) is random too and has a certain posterior probability distribution on the L1-norm unit sphere.

The VaR [9] of a random loss value Z is a risk metric that indicates the minimum loss that would occur if the outcome was particularly unfavorable. Formally it is defined as

$$\mathrm{VaR}_{\alpha}(Z) = \inf\{z : \mathbb{P}(Z \leq z) \geq \alpha\},$$

where $\alpha \in (0, 1)$ measures sensitivity to the risk and is usually high (e.g. 0.95). The Active-VaR method proposes to choose the next query by finding the state that has the maximum VaR of EVD:

$$s_i^{\mathsf{q}} = \arg\max_{s \in \mathcal{S}_0} \mathrm{VaR}[\mathrm{EVD}(\boldsymbol{\theta} \mid \hat{\pi}_{i-1}, s)]$$

Intuitively, this algorithm chooses the starting states for which the worst possible policy losses are particularly high, thus reducing the overall worst-case difference in the performance of the inferred policy.

In our present work, we propose a teaching algorithm that uses a modified version of this method for selecting query states that the teacher will send to the learner.

3.4 Machine Teaching

The concept of MT was originally proposed by Goldman and Kearns [7] in what is now called the *teaching set* formulation. In this formulation, X is a limited set of *instances* and $H = \{h : X \to \{0, 1\}\}$ is a limited set of binary *concepts* that the learner initially has in consideration. The teacher has to teach the learner a *target concept* $h^\star \in H$ by sending a sequence of *examples* $E' = \{(x, h^\star(x)), x \in E \subseteq X\}$. When the learner receives such a set, it will discard all concepts that are inconsistent with E'. A set $E^\star \subset X$ is called a *teaching set* for $(h^\star, H)$ if $h^\star$ is the only concept in H that is left after discarding all incompatible concepts. An *optimal teacher* is then the one that provides the smallest teaching set. The authors note that the problem of finding the optimal teaching set is NP-hard and equivalent to the minimum set cover problem. The authors have laid the theoretical foundations for MT, however, the teaching set formulation of many real-life problems can often be intractable.

Devidze et al. [6] study how the uncertainty about the learner model affects the performance of the teacher. They consider the problem of binary classification and a probabilistic learner that maintains a score $Q(h)$ for each hypothesis h and picks hypotheses randomly according to their relative weight. The authors analyze the effects of teacher's imperfect assumptions about the learner's model. They find that the effectiveness of teaching is more sensitive to some aspects of the learner's model (e.g. the learning rate) than to others (e.g. the initial hypothesis scores).

3.5 Interactive Machine Teaching

Liu et al. [12] explore the interaction between a teacher and a learner in a supervised learning setting when both entities represent the target concept as a linear model. The goal of the teacher is to give the most informative examples to the learner so that the learner's parameter can converge to the target value as quickly as possible. They first consider an *omniscient* teacher that knows everything about the learner and conclude that in many situations, an effective teaching strategy should start with easy examples and end with difficult ones. They also consider a less informative 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. Their work provides a foundation for further research in interactive MT, but the reliance on the unlimited feedback from the learner makes the application of their methods limited in practice.

Liu et al. [13] further explore the scenario when the teacher's knowledge about the learner is limited. Similarly to the previous work, the teacher knows the learner's model (e.g. loss function) and the optimization algorithm (including the learning rate) but does not know the feature representation or the parameter. 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. [16] explore the situation when the teacher has wrong assumptions about the learner. Interaction between the teacher and the learner is used to mitigate those wrong assumptions. The authors focus on 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.

Dasgupta et al. [5] explore interactive teaching in the classical *teaching set* formulation. They consider a *black-box* learner, whose set of concepts is unknown to the teacher, and find that without interaction having a teacher is not better than having random examples. Then they explore the effects of having interaction and propose an algorithm that builds a good approximation of the optimal teaching set.

3.6 Non-interactive teaching for IRL learners

MT was first applied to sequential tasks by Cakmak and Lopes [4]. They make common assumptions that the reward is linear and that the teacher will provide enough demonstrations for the learner to estimate the teacher's expected feature counts reliably. With these assumptions, every demonstrated state-action pair induces a half-space constraint on the true reward weight vector:

$$\langle \boldsymbol{\theta}^\star, \boldsymbol{\mu}^{\pi^\star}(s,a) \rangle \geq \langle \boldsymbol{\theta}^\star, \boldsymbol{\mu}^{\pi^\star}(s,a') \rangle, \forall a'$$

where $\boldsymbol{\mu}^\pi(s,a)$ is the vector of expected feature counts when the agent starts at the state s and its first action is a. 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 an algorithm for choosing the demonstration set: at every step, the teacher will pick a demonstration that minimizes the resulting subspace volume. This approach has several limitations, e.g., if there is a state that has two optimal actions, then having a demonstration with both state-action pairs will immediately result in a subspace of zero volume.

Brown and Niekum [2] address the limitations found in the method of Cakmak and Lopes [4]. They define a policy's *behavioral equivalence class* (BEC) as a set of reward weights under which that policy is optimal. Similarly to [4], they assume that the learner has enough demonstrations to estimate expected feature counts reliably. A BEC of a demonstration given a policy is the intersection of half-spaces formed

by all state-action pairs present in such demonstration. They note that if $\boldsymbol{\theta}^L \in \mathrm{BEC}(\pi^\star)$ and $\langle \boldsymbol{\theta}^L, \boldsymbol{\phi}(s) \rangle$ is not constant for all s, then $\mathcal{V}^{\pi^L} = \mathcal{V}^{\pi^\star}$. Given the fact that many popular IRL algorithms produce feature weights within the BEC of the 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 authors propose an algorithm based on generating m demonstrations from each starting state and a greedy method of picking candidates. Both this algorithm and the algorithm of Cakmak and Lopes [4] assume that the demonstrations are optimal, which limits their applicability in practice.

Melo and Lopes [15] explore the setting when the teacher has to teach several IRL learners that differ in the problem they solve or their learning model. They define the teaching loss as a mismatch between the learner's optimal policies for the true reward and the reward inferred from the demonstration, $\mathbb{I}(\pi^L(R^\star) \neq \pi^L(R^\Xi))$, presented as a hard constraint. The teacher knows the learner model and uses the set-cover algorithm of Brown and Niekum [2] to build the demonstration set. They show that if the same demonstration has to be given to all learners, the teaching problem may not always have a solution. The authors verify this observation for the cases when learners differ in transition probabilities, discount factors, and reward representations. They find a condition necessary for two learners to be teachable from a single demonstration: $\pi^{L_A}(R^\star) = \pi^{L_B}(R^\star)$. Finally, they provide two class teaching algorithms, one for exact teaching, which involves providing individual demonstrations, and one for approximate teaching, which does not include the individual teaching phase.

Troussard et al. [21] explore the situation when an MCE learner receives demonstrations from the teacher in batches. They consider a learner that uses soft-value-iteration to find an MCE policy given current weights and updates the weights with the gradient ascent algorithm. The teacher's goal is to steer the learner toward the optimal weight parameter. At each step, the teacher, without knowing the learner's state, builds a minibatch of demonstrations Ξ' that maximizes the minibatch reward $\langle \boldsymbol{\theta}^\star, \boldsymbol{\mu}^{\Xi'} \rangle$. The authors assume that the teacher can only choose demonstrations from a defined demonstration pool and analyze how the order of demonstrations affects the learning performance. Since the MCE learner's update algorithm uses the gradient of a likelihood of a demonstration under the current policy, they propose ordering demonstrations by the likelihood. The likelihood of a trajectory under an MCE policy is proportional to its reward. Thus, the demonstrations can be ordered by their reward.

3.7 Interactive teaching for IRL learners

Kamalaruban et al. [10] consider the interaction between a teacher and an IRL learner. The authors consider a learner model that uses MCE policies and performs a single step of the MCE gradient ascent upon receiving every new demonstration. They first consider an omniscient teacher whose goal is to steer the learner toward the optimal weight parameter and find an effective teaching algorithm. To pick

the new demonstration, the teacher solves an optimization problem that relies on the knowledge of the learner's policy and update 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 π_i^L, the teacher can periodically request the learner to generate k trajectories from every initial state, thus estimating π_i^L.

Yengera et al. [23] further explore the ideas proposed in [10] and propose the difficulty score ratio (DSR) algorithm.

For deterministic MDPs, the *difficulty score* of a demonstration ξ w.r.t. a policy π is defined as

$$\Psi(\xi) = \frac{1}{\prod_t \pi(a_t \mid s_t)}$$

Given the teacher's policy $\pi^\star$ and an estimate of the current learner's policy $\hat{\pi}^L$, it is possible to compute the difficulty score ratio:

$$\frac{\hat{\Psi}_i^L(\xi)}{\Psi^T(\xi)} = \prod_t \frac{\pi^\star(a_t \mid s_t)}{\hat{\pi}_i^L(a_t \mid s_t)}$$

The DSR algorithm selects the next teacher's demonstration ξ_i^T by iterating over a pool of candidate trajectories Ξ and finding the trajectory with the maximum difficulty score ratio:

$$\xi_i^T = \arg\max_{\xi \in \Xi} \frac{\hat{\Psi}_i^L(\xi)}{\Psi^T(\xi)}$$

This algorithm was proven effective in the case of an MCE learner and a cross-entropy behavioral cloning (CrossEnt-BC) learner [23].

Both this approach and the approach of Kamalaruban et al. [10] assume that the teacher can periodically observe many learner's trajectories from every initial state and thus estimate the learner's policy with high precision, which limits their application in practice. In our present work, we propose teaching algorithms that receive limited feedback from the learner and use the DSR method to choose the teacher's demonstrations.

4

Teaching with limited feedback

Contents

In this chapter, we first provide the formal definition of the problem of interactive teaching with limited feedback. Then, in sections 4.2 and 4.3, we describe how the MCE-IRL and Active-VaR algorithms can be modified to be compatible with the interactive teaching process. We call these modified versions Interactive-MCE and Interactive-VaR, respectively. Then, in section 4.4, we propose a teaching algorithm that combines Interactive-VaR, Interactive-MCE, and DSR into a compound algorithm which we call *VaR-based teaching* (VaR-T).

In section 4.5, we propose a simple AL algorithm that was specifically designed for teaching with limited feedback. Unlike Interactive-VaR, which does not take into consideration the full context of teaching with limited feedback, the new algorithm, *demonstration-based active learning* (DBAL), uses the knowledge about the previous teacher's demonstrations to select the next query states. In section 4.6, we propose a teaching algorithm (DBAL-T) that is based on it.

4.1 Problem formalism

We consider two entities that can execute policies in MDP 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 $M \setminus R^\star$. Since the teacher has access to the reward function, it also knows the optimal policy $\pi^\star$ for the environment, which it will use to generate the teaching demonstrations.

The teacher and the learner can interact with each other iteratively, each iteration consisting of five steps described in Algorithm 4. 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., $s_i^{\mathsf{q}} \in \mathcal{S}_0 = \{s : \mathbb{P}_0(s) > 0\}$. In the second step, the learner generates a *trajectory* ξ_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 and policy. In the fourth step, the teacher demonstrates the optimal behavior by generating a trajectory ξ_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 and policy. The teaching process is terminated when the *teaching goal* is achieved, in the sense defined below.

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 $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^d$.
- **Rationality**: At every iteration, the learner maintains some reward mapping R_i^L and derives a stationary policy π_i^L that is optimal for R_i^L, which it uses to generate trajectories. The exact method of deriving π_i^L from R_i^L is unknown to the teacher.

Algorithm 4 Interactive teaching for Inverse Reinforcement Learning with limited feedback

1: **for** $i = 1, \ldots, \infty$ **do**
2: Teacher sends a *query state* s_i^{q} and requests a *trajectory* starting from it
3: Learner generates and sends a *trajectory* ξ_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* ξ_i^T
6: Learner updates its reward R_i^L and policy π_i^L
7: Stop if the teaching goal is achieved
8: **end for**

- **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: $R_i^L(s) = \langle \boldsymbol{\theta}_i, \boldsymbol{\phi}(s) \rangle$, where the vector $\boldsymbol{\theta}_i$ is called the *feature weights*. Furthermore, we assume that the true reward $R^\star$ can be expressed as a function of these features, i.e., there is a weight vector $\boldsymbol{\theta}^\star \in \mathbb{R}^d$ such that $R^\star(s) = \langle \boldsymbol{\theta}^\star, \boldsymbol{\phi}(s) \rangle$ for all $s \in \mathcal{S}$.

- **Learning from demonstrations**: Upon receiving a demonstration ξ_i^T, the learner uses it to update its parameter $\boldsymbol{\theta}_{i+1}$ and thus its reward R_{i+1}^L. The exact method of updating $\boldsymbol{\theta}_{i+1}$ from ξ_i^T is unknown to the teacher.

There are different ways of evaluating the teacher's performance. In general, some notion of numerical *loss* 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): $L_i = \mathcal{V}^{\pi^\star} - \mathcal{V}^{\pi_i^L}$ [1, 26], when evaluated against the real reward $R^\star$, and define the teaching goal as achieving a certain loss threshold ε in the lowest number of demonstrations.

Since the teacher has no access to π_i^L, it has to infer it from the trajectories received during teaching, which corresponds to the problem of inverse reinforcement learning. To infer π_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 on every iteration, it can be divided into three "modules", each solving one problem. Figure 4.1 shows the inputs and outputs of these modules. Every module has access to the previously generated data (e.g., policy estimates, learner trajectories, demonstrations) as the input. The AL module produces a query state s_i^{q}, which is sent to the learner. The IRL module takes the learner's trajectory ξ_i^L and produces the estimate of the learner's policy $\hat{\pi}_i^L$. The MT module takes the policy estimate and produces the demonstration ξ_i^T, which is sent to the learner. Each module is discussed in the subsequent sections.

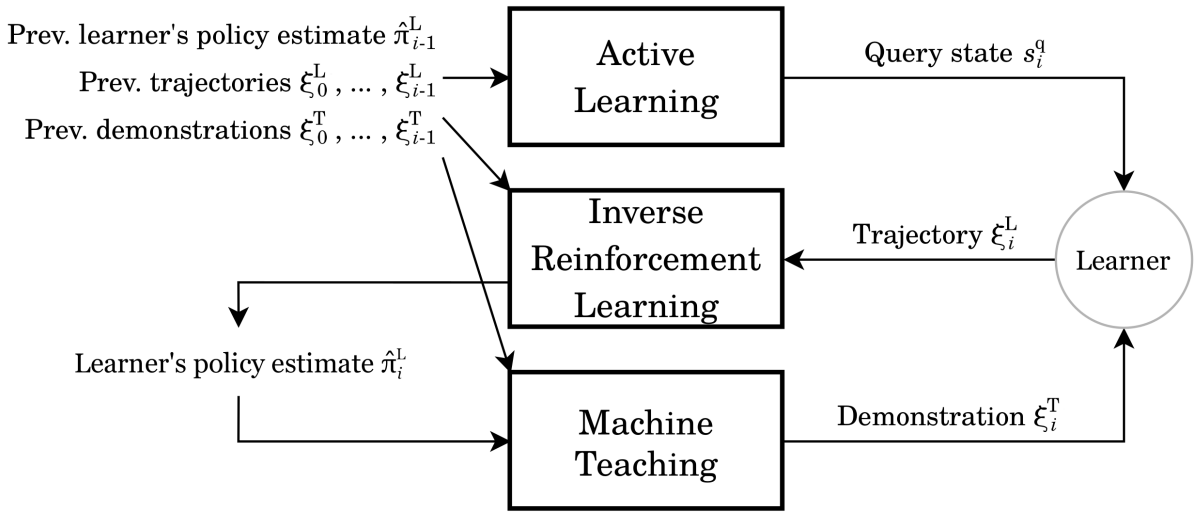


Figure 4.1: The division of the teaching algorithm into three modules, each solving AL, IRL, or MT problem at every iteration. This diagram shows the inputs and outputs of the modules.

4.2 Interactive-MCE

Most of the IRL literature assumes that all the available trajectories were generated by a constant policy that is based on a constant reward function. In the example of MCE-IRL, if a set Ξ of trajectories is available, the whole set can be used simultaneously to compute the gradient:

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

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 slightly modified version of the sequential MCE-IRL algorithm of Kamalaruban et al. [10]. At every interaction step i, this algorithm starts with the previously inferred weights $\hat{\boldsymbol{\theta}}_{i-1}$ and applies the MCE gradient ascent with only the new trajectory ξ_i as the evidence. If some features are not visited in ξ_i, the sequential MCE will preserve the old values of the corresponding weights in $\hat{\boldsymbol{\theta}}_i$, which might have been calculated in the previous iterations. Unlike the original algorithm, 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.

Interactive-MCE is formally described in Algorithm 5. The algorithm starts by computing the feature counts of the trajectory and setting the previously inferred weights as the current weight vector. It then computes the MCE policy for the current weight vector by using the soft-value iteration method [26]

Algorithm 5 Interactive-MCE

Require: trajectory ξ_i, previous or initial estimate $\hat{\boldsymbol{\theta}}_{i-1}$

$s_0 = \text{First-State}(\xi_i)$
$\boldsymbol{\mu}^{\xi_i} = \text{Feature-Counts}(\xi_i)$
$\hat{\boldsymbol{\theta}}_i = \hat{\boldsymbol{\theta}}_{i-1}$
$\hat{\pi}_i = \text{Soft-Value-Iter}(\hat{\boldsymbol{\theta}}_i)$
for $n = 1, \ldots, N$ **do**
    $\boldsymbol{\mu}^{\hat{\pi}_i, s_0} = \text{Feature-Counts}(\hat{\pi}_i, s_0)$
    $\hat{\boldsymbol{\theta}}_i = \hat{\boldsymbol{\theta}}_i + \eta_n(\boldsymbol{\mu}^{\xi_i} - \boldsymbol{\mu}^{\hat{\pi}_i, s_0})$
    $\hat{\pi}_i = \text{Soft-Value-Iter}(\hat{\boldsymbol{\theta}}_i)$
end for

Algorithm 6 Soft-value iteration

Require: weight vector $\boldsymbol{\theta}$

$V^{\text{soft}}(s) = 0, \forall s$
while $V^{\text{soft}}(\cdot)$ not converged **do**
    $Q^{\text{soft}}(s, a) = \langle \boldsymbol{\theta}, \boldsymbol{\phi}(s) \rangle + \gamma \mathbb{E}_{S \sim \mathbb{T}(\cdot \mid s, a)}[V^{\text{soft}}(S)], \forall s, a$
    $V^{\text{soft}}(s) = \frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp[\beta Q^{\text{soft}}(s, a')], \forall s$
end while
$\pi(a \mid s) = \exp[\beta Q^{\text{soft}}(s, a) - \beta V^{\text{soft}}(s)]$

(described further below). Then the algorithm performs the gradient ascent for N steps. At every step of the gradient ascent, it computes the expected feature counts of the current policy in an MDP that always starts with the first state of the trajectory. It then updates the current weight vector and recalculates the current MCE policy. Finally, it outputs the last MCE policy as the result.

The soft-value iteration method of finding the MCE policy for the given reward weights is formally described in Algorithm 6. It is similar to the classical value iteration method [20]. It operates by iteratively updating the estimates of the soft state-values and the soft Q-values until the soft state-values converge. After that, it calculates and outputs the resulting MCE policy.

4.3 Interactive-VaR

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. A simple way of addressing this problem is to give less weight to the older observations when computing the likelihood

of any $\boldsymbol{\theta}$:

$$\mathbb{P}(\boldsymbol{\theta} \mid \xi_1^L, \ldots, \xi_k^L) \approx \frac{1}{Z} \prod_{i=1}^{k} \mathbb{P}(\xi_i^L \mid \boldsymbol{\theta})^{\boldsymbol{\lambda}_{k-i}}$$

$$\boldsymbol{\lambda}_0 = 1$$

$$\lim_{i \to \infty} \boldsymbol{\lambda}_i = 0$$

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

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

or to have $\boldsymbol{\lambda}_i$ decay exponentially as $\boldsymbol{\lambda}_i = \lambda^i$, $\lambda < 1$.

We also introduce several modifications to the original algorithm to better adapt it to the assumption that the learner uses an MCE policy instead of a softmax policy. Firstly, we use the soft Q-values of the MCE policy to calculate the demonstration probabilities:

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

Secondly, for calculating VaR, we use the difference of the expected soft values: $\mathrm{EVD}(\boldsymbol{\theta}|\pi, s) = V^{\text{soft},\pi}(s) - V^{\text{soft},\pi^{\boldsymbol{\theta}}}(s)$. Finally, we replace the assumption about the known softmax confidence factor c with a similar assumption about the known MCE entropy factor β.

4.4 VaR-based Teaching

Our first teaching algorithm, VaR-T, sequentially executes Interactive-VaR, Interactive-MCE, and DSR at every iteration. It is formally described in Algorithm 7. On the first iteration, the AL module of the compound teaching algorithm selects a random query state. In the subsequent iterations, it uniformly samples a set of candidate rewards on an L1-norm sphere, computes the conditional probability of each reward, computes the soft EVD for every reward and every initial state, computes VaR for every initial state, sends the state with the highest VaR to the learner, and receives the learner's trajectory. The IRL module executes the Interactive-MCE algorithm (Algorithm 5) and produces the estimate of the learner's policy. The MT algorithm iterates over a set of candidate demonstrations and sends the demonstration with the highest difficulty score ratio to the learner.

Algorithm 7 VaR-based Teaching (VaR-T)

for $i = 1, \ldots, \infty$ **do**

---

  if $i = 1$ **then** ▷ AL step

    Pick a random initial state $s_1^{\mathsf{q}} \in \mathcal{S}_0$

  else

    Sample reward weights Θ

    $s_i^{\mathsf{q}} = \arg\max_{s \in \mathcal{S}_0} \mathrm{VaR}[\text{Soft-EVD}(\Theta \mid \hat{\pi}_{i-1}^L, s)]$

  end if

  Send s_i^{q} to the learner and receive ξ_i^L

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

---

  Compute difficulty scores $\hat{\Psi}_i^L(\xi)$ and $\Psi^T(\xi)$ for every ξ in the candidate pool Ξ ▷ MT step

  $\xi_i^T = \arg\max_{\xi \in \Xi} \frac{\hat{\Psi}_i^L(\xi)}{\Psi^T(\xi)}$

  Send ξ_i^T to the learner

  Stop if $\mathcal{V}^{\pi^\star} - \mathcal{V}^{\pi_i^L} < \varepsilon$

end for

4.5 Demonstration-based Active Learning

The demonstration-based active learning (DBAL) algorithm is a simple alternative to Interactive-VaR. The idea of this algorithm is to simply choose the next query equal to the first state of the previous demonstration:

$$s_i^{\mathsf{q}} = \begin{cases} s_0^{\xi_{i-1}^T}, & \text{if } i > 1 \\ \text{random } s \in \mathcal{S}_0, & \text{otherwise} \end{cases}$$

The rationale for this approach is as follows: When the learner receives a demonstration ξ_{i-1}^T, it is reasonable to assume that it will update the weights of the features visited in ξ_{i-1}^T. Thus, querying its behavior from the same initial state will likely produce a trajectory containing the most information about the changed weights.

This algorithm corresponds to the real-life situation when a teacher gives tests to the students after every class, every test focusing on the material just learned.

4.6 DBAL-based Teaching

Our second teaching algorithm, DBAL-T, sequentially executes DBAL, Interactive-MCE, and DSR. It is formally described in Algorithm 8. It is similar to the VaR-T algorithm (Algorithm 7) with the only difference in the way of selecting the query states. Instead of finding the state with the highest VaR, it selects the first state of the previous teacher's demonstration.

Algorithm 8 DBAL-based Teaching (DBAL-T)

for $i = 1, \ldots, \infty$ **do**

---

  if $i = 1$ **then** ▷ AL step

    Pick a random initial state $s_1^{\mathsf{q}} \in \mathcal{S}_0$

  else

    $s_i^{\mathsf{q}} = \text{First-State}(\xi_{i-1}^T)$

  end if

  Send s_i^{q} to the learner and receive ξ_i^L

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

---

  $\xi_i^T = \arg\max_\xi \frac{\hat{\Psi}_i^L(\xi)}{\Psi^T(\xi)}$ ▷ MT step

  Send ξ_i^T to the learner

  Stop if $\mathcal{V}^{\pi^\star} - \mathcal{V}^{\pi_i^L} < \varepsilon$

end for

When the teacher uses DBAL for selecting queries, the three algorithms create a circular dependency: ξ_{i+1}^T depends on $\hat{\pi}_i^L$, which depends on ξ_i^L, which depends on s_{i-1}^{q}, which itself depends on ξ_{i-2}^T. This results in the algorithm being strongly dependent on the teacher's initial assumption about the learner's reward weights $\hat{\boldsymbol{\theta}}_0^L$. For example, if the teacher assumes $\hat{\boldsymbol{\theta}}_0^L = \boldsymbol{\theta}^\star$, then the following scenario may happen:

- The teacher sends a random s_1^{q} to the learner
- The teacher receives ξ_1^L and calculates a reward estimation $\hat{\boldsymbol{\theta}}_1^L$. It will be different from $\boldsymbol{\theta}^\star$ only in the features encountered in ξ_1^L.
- The teacher sends a demonstration ξ_1^T that is likely to visit the same features as ξ_1^L.
- The learner learns from ξ_1^T.
- The teacher sends s_2^{q} which is the first state of ξ_1^T.
- The teacher receives ξ_2^L with the same set of features and remains in the loop of updating the same feature set.

This problem can be avoided if the teacher starts with a *pessimistic assumption* about the initial learner weights, $\hat{\boldsymbol{\theta}}_0^L = -\boldsymbol{\theta}^\star$.

One practical advantage of DBAL over Interactive-VaR is computational speed. The performance of Interactive-VaR depends on the size of MDP and the reward sample size, whereas DBAL constantly requires only one read operation.

5

Experimental evaluation

Contents

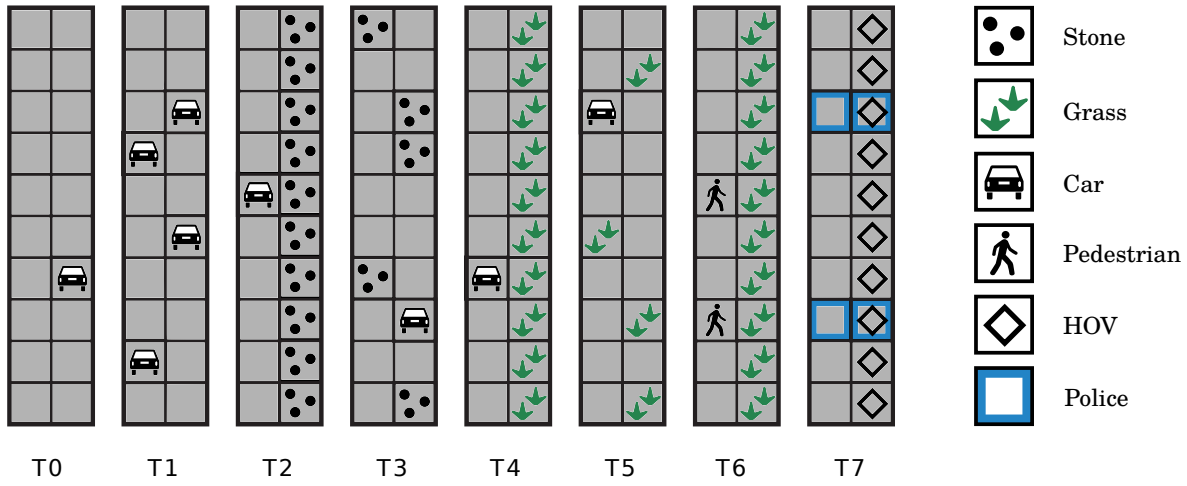


Figure 5.1: Examples of each road type of the car environment. Each 2×10 grid represents a road. The agent starts at the bottom left corner of a randomly selected road. After the agent has advanced for 10 steps upwards along the road, the MDP is terminated.

We tested our teaching algorithm in the synthetic car driving environment proposed by Kamalaruban et al. [10].

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 road is selected randomly at the start of each run, and the run 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 represent various driving tasks, which we will refer to as T0-T7:

- 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.

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.

Table 5.1: 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

Each road is represented as a 2×10 grid. 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 $\gamma = 0.99$.

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 5.1. In particular, the agent is severely penalized for driving into a pedestrian (-10) or having one immediately in front of it (-5). It is also moderately penalized for driving into another car (-5) or having one in front of it (-2). The agent is slightly penalized for driving over stones (-1) or grass (-0.5). Driving over police has no reward, and driving over HOV is preferred (+1).

Figure 5.1 demonstrates example roads of all types. In the figure, each 2×10 grid represents a road. The icons in the grid cells indicate the features that are present in those cells.

We chose this environment for our tests because the fact that the roads are completely separated from each other makes the teaching and learning algorithms especially sensitive to the selection of initial states and demonstrations. Unlike in the more connected environments, where the agent may have a chance to visit many features from any initial state, the features that the agent can visit in this environment in a single run are strictly determined by the initial state of the run. This severely limits the overall informativeness of trajectories and puts more importance on the process of selection of teacher's query states and demonstrations.

Table 5.2: Tested algorithms

Name	*AL*	*IRL*	*MT*
RANDOM	-	-	Random
RNDAL	Random	Interactive-MCE	DSR
VAR-T	Interactive-VaR	Interactive-MCE	DSR
DBAL-T	DBAL	Interactive-MCE	DSR
UNLIM	-	-	DSR

5.1 MCE learner

We use a linear variant of the MCE learner proposed by Kamalaruban et al. [10] for our experiments. This learner starts with random initial weights $\boldsymbol{\theta}_1^L$, every element being uniformly sampled from $(-10, 10)$ and corresponding to one state feature. For every weight vector $\boldsymbol{\theta}_i^L$, it finds an MCE policy π_i^L with the soft-value iteration (algorithm 6). Upon receiving a new demonstration ξ_i^T from the teacher, the learner updates its weights by executing one iteration of the MCE gradient ascent (algorithm 5), with a constant learning rate $\eta = 0.34$.

5.2 Teaching algorithms

We compared the following algorithms, also presented in Table 5.2:

- RANDOM teacher does not infer $\boldsymbol{\theta}_i^L$ and selects demonstrations completely randomly. This algorithm was originally proposed in [10] and serves as the worst-case baseline.
- RNDAL teacher selects query states randomly but uses MCE (section 4.2) to infer the learner reward and DSR (section 3.7) to select demonstrations. We included this algorithm as the second worst-case baseline to see whether using an AL algorithm can boost the teaching process.
- VAR-T (section 4.4) uses Interactive-VaR to select query states, followed by Interactive-MCE and DSR.
- DBAL-T (section 4.6) uses DBAL to select query states, followed by Interactive-MCE and DSR.
- UNLIM teacher knows the exact learner's policy at every step and uses the DSR algorithm to select demonstrations. This algorithm was originally proposed in [23] and serves as the best-case baseline.

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[1]. For computing the posterior likelihood of $\boldsymbol{\theta}$, the demonstrations are weighted exponentially with $\lambda = 0.4$. The EVD between the two policies is computed using soft policy values. The α factor of VaR was set to 0.95.

The Interactive-MCE algorithm uses 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 teacher's performance

We have tested the algorithms with 16 randomly initialized environments which differ in the placement of random road objects.

Figure 5.2a 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 π_i^L. The thick lines represent the averages of 16 runs, and the thin vertical lines measure the span of two standard deviations. As we can see, at any iteration, DBAL-T is able to estimate the learner's policy more reliably, while VAR-T performs worse but significantly better than RNDAL . The teacher's performance in estimating the learner's policy is an intermediate result that affects the overall teaching performance, which is discussed next.

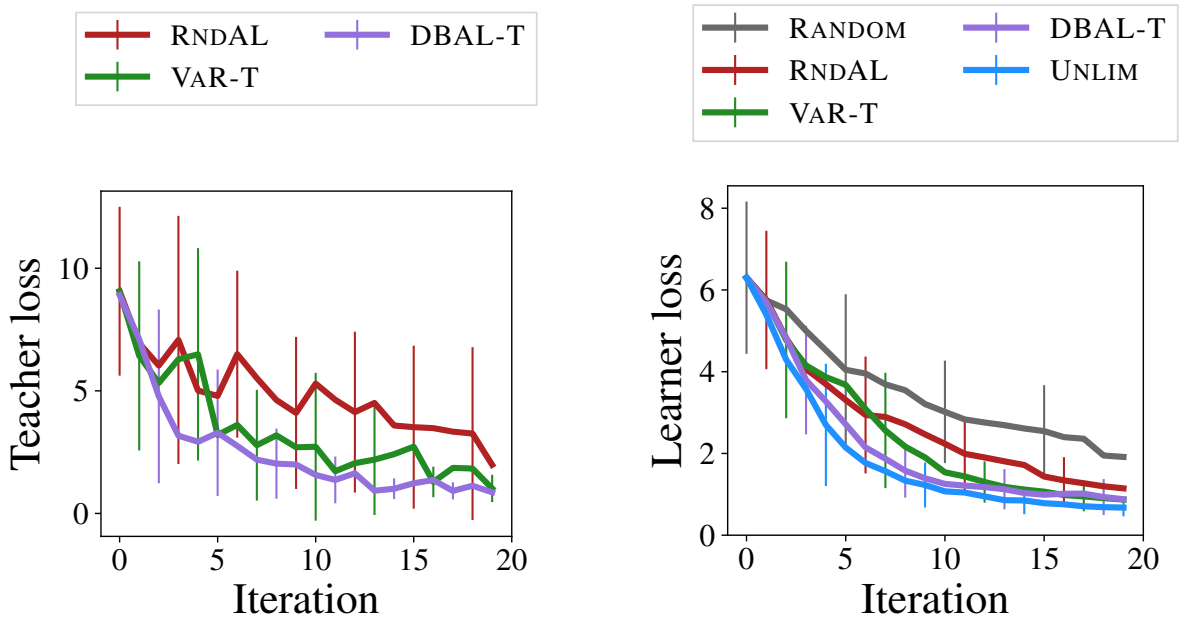
Figure 5.2b displays the effectiveness of the teacher's effort in teaching the learner. For every iteration step, it shows the loss of the learner's policy π_i^L w.r.t. the optimal policy $\pi^\star$, averaged over 16 runs. The thin vertical lines measure the span of two standard deviations across 16 runs.

As we can see, for any loss threshold higher than one, DBAL-T and VAR-T would require almost the same number of teaching iterations as UNLIM , which implies that **teaching with limited feedback can be performed as effectively as teaching with unlimited feedback**.

The performance of RNDAL significantly lags behind, requiring almost twice as many iterations to reach any threshold higher than one, which implies that selecting query states with an AL algorithm significantly improves the effectiveness of teaching. However, it's significantly more effective than RANDOM and requires almost twice fewer iterations to reach any threshold higher than two, which means that a teaching algorithm that receives random learner's trajectories performs significantly better than a random teaching algorithm.

Since some algorithms have similar performance and fall within each other's standard deviation span, we have performed Welch's t-test [22] for every iteration and every pair of algorithms to see which pairs of algorithms have a statistically significant difference in performance. Welch's t-test is used to test the hypothesis that two populations have equal means. We consider the performance of two teaching

[1] Increasing the sample size or sampling with MCMC yields similar results.



(a) Teacher's inferred policy loss.

(b) Learner's policy loss.

Figure 5.2: Teaching results 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 runs. The thin vertical lines measure the span of two standard deviations. DBAL-T demonstrates the lowest losses during the whole process, followed by VAR-T and RNDAL .

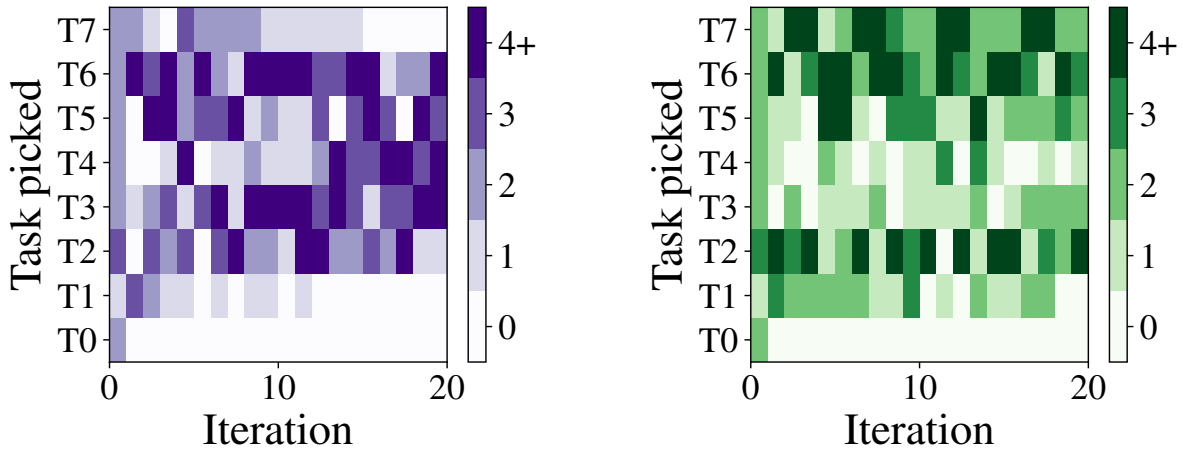
Table 5.3: Percentage of teaching iterations (out of 20) at which a pair of teaching algorithms has a significant difference in performance, as measured by Welch's t-test.

	RNDAL	VAR-T	DBAL-T	UNLIM
RANDOM	45	65	80	85
RNDAL		30	45	75
VAR-T			0	60
DBAL-T				5

algorithms to be significantly different in a given iteration if Welch's t-test returns a p-value lower than 0.05, thus rejecting the hypothesis of equal means. For every pair of algorithms, the table 5.3 shows the percentage of iterations (out of 20) in which the performance is significantly different. The obtained values support the results stated above: DBAL-T is similar in performance to UNLIM and VAR-T , whereas the performance of RANDOM and RNDAL is significantly different from any other algorithm.

5.4 Curriculum analysis

Figure 5.3 shows the frequency of query tasks chosen by the DBAL-T and VAR-T across 16 runs. The color in every cell indicates in how many runs (out of 16) the given algorithm has selected the given task in the given iteration. As we can see, VAR-T chooses T2 (stones on the right lane) much more



(a) DBAL-T **(b)** VAR-T

Figure 5.3: Frequencies of tasks associated with query states selected by (a) DBAL-T and (b) VAR-T . At every iteration, the frequencies are counted across 16 independent runs. DBAL-T prefers querying T2-T6. VAR-T prefers querying T2, T6, and T7.

frequently than T3 (mixture of stones and cars), even though T3 has more features. The query states chosen by DBAL-T are equal to the demonstrations chosen in the previous steps (discussed below).

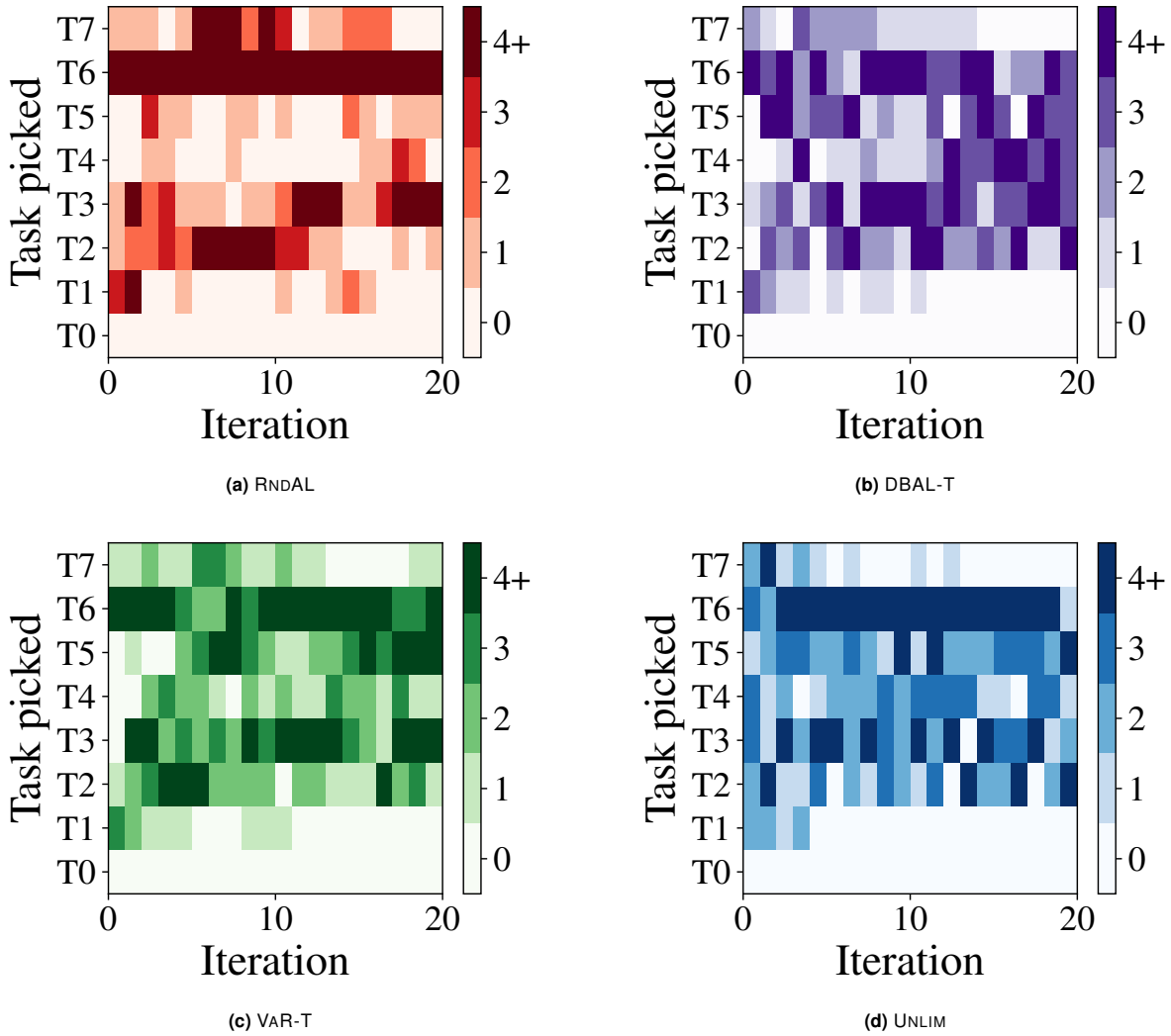
Figure 5.4 shows the frequency of demonstration tasks chosen by the teachers. The color in every cell indicates in how many runs (out of 16) the given algorithm has selected the given task in the given iteration. As we can see, all teachers choose T6 and T3 much more frequently than other tasks. We assume this happens because T6 and T3 together cover six MDP features, which yields higher difficulty scores on average.

5.5 Prepared teacher

In the original experiment, we assumed that the teacher had no information about the learner's reward at the beginning of the teaching process. In reality, however, it is often helpful for the teacher to *prepare* by estimating the learner's initial knowledge beforehand. This can be done by assigning some tests in advance or analyzing the tests the student took in the past. We model this scenario by assuming that the initial teacher's assumption is equal to the real learner's initial weights: $\hat{\boldsymbol{\theta}}_0^L = \boldsymbol{\theta}_0^L$.

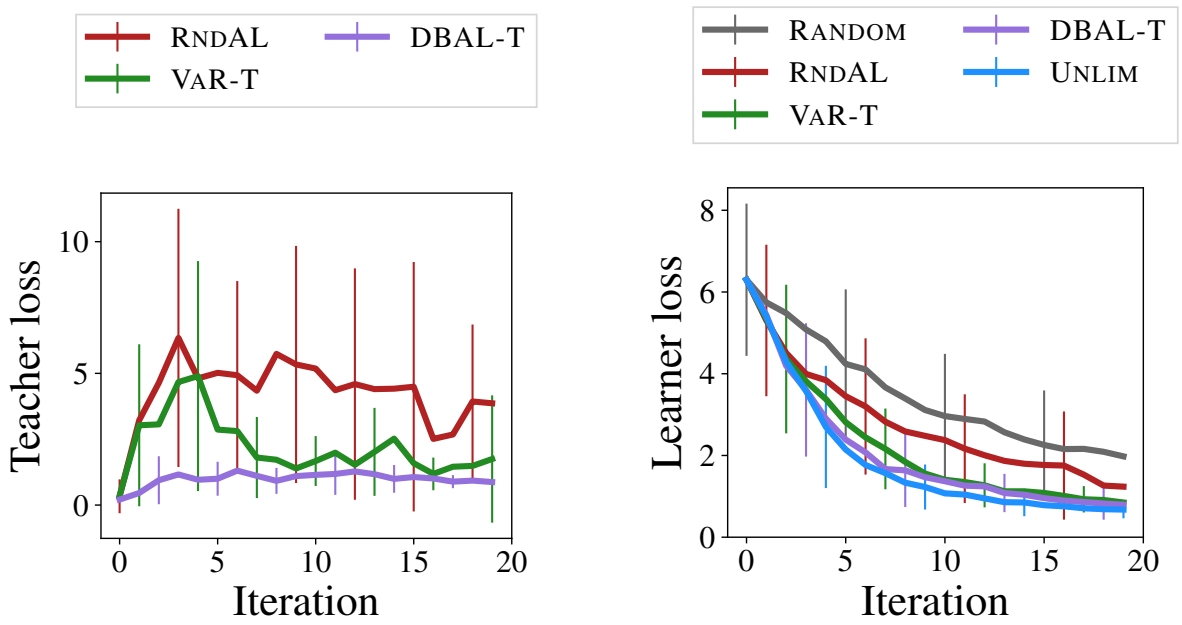
The teacher still does not know the learner's update algorithm, so after k iterations, the teacher's estimate $\hat{\boldsymbol{\theta}}_k^L$ may diverge from $\boldsymbol{\theta}_k^L$. Figure 5.5a shows how strongly different algorithms diverge in the process of teaching. As expected, DBAL-T has the best estimates on average, followed by VAR-T and RNDAL .

The corresponding teaching progress (figure 5.5b) demonstrates results similar to the case of the unprepared teacher, with DBAL-T and VAR-T requiring almost the same number of iterations as UNLIM



(a) RNDAL **(b)** DBAL-T **(c)** VAR-T **(d)** UNLIM

Figure 5.4: Frequencies of tasks associated with demonstrations selected by (a) RNDAL , (b) DBAL-T , (c) VAR-T , and (d) UNLIM . At every iteration, the frequencies are counted across 16 independent runs. The most frequent tasks overall are T3 and T6.



(a) Teacher's inferred policy loss.

(b) Learner's policy loss.

Figure 5.5: Teaching results when the teachers know the starting weights of the learners. DBAL-T demonstrates the lowest losses during the whole process, followed by VAR-T and RNDAL .

Table 5.4: Percentage of teaching iterations (out of 20) at which a pair of prepared teaching algorithms has a significant difference in performance, as measured by Welch's t-test.

	RNDAL	VAR-T	DBAL-T	UNLIM
RANDOM	0	80	85	85
RNDAL		20	60	75
VAR-T			0	10
DBAL-T				0

to reach any threshold higher than one, and RNDAL requiring almost twice as many.

Table 5.4 shows Welch's t-test results for the case of prepared teachers. As we can see, DBAL-T is similar in performance to UNLIM and VAR-T , whereas the performance of RANDOM and RNDAL is significantly different from the other three algorithms.

5.6 Evaluation of the Interactive-VaR algorithm

We have performed an additional experiment to test whether our modified version of the Active-VaR algorithm, Interactive-VaR, performs well when the trajectories that it receives are generated by a constant policy. The experiment was performed in the same car driving environment, but with a random reward weight vector and two different entities. The first entity, the *expert*, knows the true reward and uses an MCE policy to generate demonstrations. The second entity, the *active learner*, uses a learning algorithm

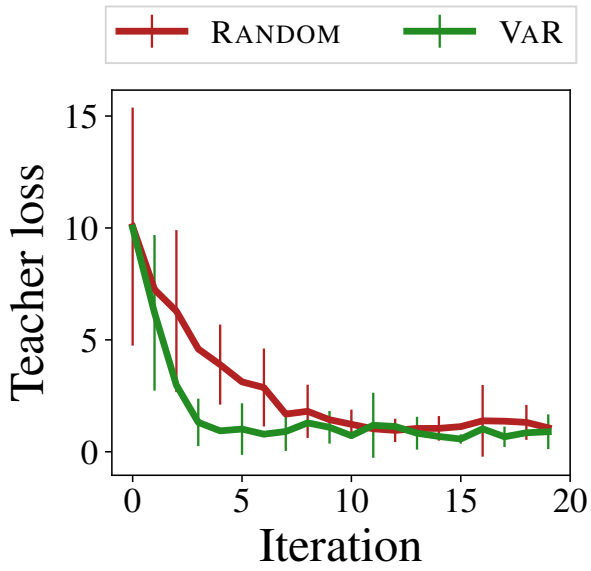


Figure 5.6: Active learning results measured as the loss of the learner's inferred policy. The thick lines represent the averages of 16 runs. The thin vertical lines measure the span of two standard deviations. Interactive-VaR-based learner demonstrates significantly lower losses.

that is a combination of an AL algorithm for selecting the query states and an IRL algorithm for inferring the expert's policy from the demonstrations. Each element of the true reward weight vector is uniformly sampled from $(-10, 10)$.

We have compared two active learners. The first learner uses Interactive-VaR to select the query states, and the second learner chooses query states randomly. Both learners use Interactive-MCE for inferring the expert's policy from the expert's trajectories. Figure 5.6 shows the performance of the two learners, measured as the EVD of the expert's and the learner's policies, averaged over 16 runs. As we can see, the VaR-based learner improves its estimation significantly faster than the learner that picks query states randomly.

6

Summary and future work

Contents

We have proposed a new teacher-learner interaction framework for teaching sequential decision-making tasks to a black-box learner. Unlike previous frameworks, the feedback from the learner is limited to just one trajectory per teaching iteration, which brings it closer to real-life situations and is more challenging for the teacher. In this framework, the teacher has to solve AL, IRL, and MT problems sequentially at every teaching iteration.

We have proposed two teaching algorithms that address this additional challenge of limited feedback. Our proposed algorithms consist of three modules, each dedicated to solving one of the three sub-problems. We have proposed a modified MCE-IRL algorithm for solving the IRL sub-problem, and two algorithms for solving the AL problem: a modified Active-VaR algorithm and a new algorithm that leverages the knowledge of the previous teacher's demonstrations. We have used the difficulty score ratio algorithm for solving the MT problem.

We have tested the teaching algorithms on a synthetic car-driving environment and compared them with existing algorithms and the worst-case baseline. We have concluded that the algorithms are effective at solving the teaching problem, with the simple DBAL-T algorithm being computationally cheaper and more effective than the VaR-based algorithm.

6.1 Future work

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's policy. Additionally, it would be interesting to test the algorithms with other types of learners.

Bibliography

[1] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[2] D. S. Brown and S. Niekum, "Machine teaching for inverse reinforcement learning: Algorithms and applications," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7749–7758.

[3] D. S. Brown, Y. Cui, and S. Niekum, "Risk-aware active inverse reinforcement learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 362–372.

[4] M. Cakmak and M. Lopes, "Algorithmic and human teaching of sequential decision tasks," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[5] S. Dasgupta, D. Hsu, S. Poulis, and X. Zhu, "Teaching a black-box learner," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1547–1555.

[6] R. Devidze, F. Mansouri, L. Haug, Y. Chen, and A. Singla, "Understanding the power and limitations of teaching with imperfect knowledge," *arXiv preprint arXiv:2003.09712*, 2020.

[7] S. A. Goldman and M. J. Kearns, "On the complexity of teaching," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 20–31, 1995.

[8] A. Jacq, M. Geist, A. Paiva, and O. Pietquin, "Learning from a learner," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2990–2999.

[9] P. Jorion, *Value at risk: the new benchmark for controlling market risk*. Irwin Professional Pub., 1997.

[10] P. Kamalaruban, R. Devidze, V. Cevher, and A. Singla, "Interactive teaching algorithms for inverse reinforcement learning," *arXiv preprint arXiv:1905.11867*, 2019.

[11] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," *Advances in neural information processing systems*, vol. 24, 2011.

[12] W. Liu, B. Dai, A. Humayun, C. Tay, C. Yu, L. B. Smith, J. M. Rehg, and L. Song, "Iterative machine teaching," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2149–2158.

[13] W. Liu, B. Dai, X. Li, Z. Liu, J. Rehg, and L. Song, "Towards black-box iterative machine teaching," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3141–3149.

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

[15] F. S. Melo and M. Lopes, "Teaching multiple inverse reinforcement learners," *Frontiers in Artificial Intelligence*, p. 27, 2021.

[16] F. S. Melo, C. Guerra, and M. Lopes, "Interactive optimal teaching with unknown learners." in *IJCAI*, 2018, pp. 2567–2573.

[17] A. Y. Ng, S. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[18] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning." in *IJCAI*, vol. 7, 2007, pp. 2586–2591.

[19] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[21] M. Troussard, E. Pignat, P. Kamalaruban, S. Calinon, and V. Cevher, "Interaction-limited inverse reinforcement learning," *arXiv preprint arXiv:2007.00425*, 2020.

[22] B. L. Welch, "The generalization of 'student's'problem when several different population varlances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.

[23] G. Yengera, R. Devidze, P. Kamalaruban, and A. Singla, "Curriculum design for teaching via demonstrations: Theory and applications," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 496–10 509, 2021.

[24] X. Zhu, "Machine teaching: An inverse problem to machine learning and an approach toward optimal education," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

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

[26] B. D. Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

[27] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[28] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "The principle of maximum causal entropy for estimating interacting processes," *IEEE Transactions on Information Theory*, vol. 59, no. 4, pp. 1966–1980, 2013.