Curriculum Design for Teaching via Demonstrations: Theory and Applications

Gaurav Yengera[1,2] gyengera@mpi-sws.org
Rati Devidze[1] rdevidze@mpi-sws.org
Parameswaran Kamalaruban[3] kparameswaran@turing.ac.uk
Adish Singla[1] adishs@mpi-sws.org

[1]Max Planck Institute for Software Systems (MPI-SWS), Saarbrucken, Germany
[2]Saarland University, Saarland Informatics Campus (SIC), Saarbrucken, Germany
[3]The Alan Turing Institute, London, UK

Abstract

We consider the problem of teaching via demonstrations in sequential decision-making settings. In particular, we study how to design a personalized curriculum over demonstrations to speed up the learner's convergence. We provide a unified curriculum strategy for two popular learner models: Maximum Causal Entropy Inverse Reinforcement Learning (MaxEnt-IRL) and Cross-Entropy Behavioral Cloning (CrossEnt-BC). Our unified strategy induces a ranking over demonstrations based on a notion of difficulty scores computed w.r.t. the teacher's optimal policy and the learner's current policy. Compared to the state of the art, our strategy doesn't require access to the learner's internal dynamics and still enjoys similar convergence guarantees under mild technical conditions. Furthermore, we adapt our curriculum strategy to the setting where no teacher agent is present using task-specific difficulty scores. Experiments on a synthetic car driving environment and navigation-based environments demonstrate the effectiveness of our curriculum strategy.

1 Introduction

Imitation learning is a paradigm in which a learner acquires a new set of skills by imitating a teacher's behavior. The importance of imitation learning is realized in real-world applications where the desired behavior cannot be explicitly defined but can be demonstrated easily. These applications include the settings involving both human-to-machine interaction [1–4], and human-to-human interaction [5, 6]. The two most popular approaches to imitation learning are Behavioral Cloning (BC) [7] and Inverse Reinforcement Learning (IRL) [8]. BC algorithms aim to directly match the behavior of the teacher using supervised learning methods. IRL algorithms operate in a two-step approach: first, a reward function explaining the teacher's behavior is inferred; then, the learner adopts a policy corresponding to the inferred reward.

In the literature, imitation learning has been extensively studied from the learner's point of view to design efficient learning algorithms [9–15]. However, much less work is done from the teacher's point of view to reduce the number of demonstrations required to achieve the learning objective. In this paper, we focus on the problem of Teaching via Demonstrations (TvD), where a helpful teacher assists the imitation learner in converging quickly by designing a personalized curriculum [16–20]. Despite a substantial amount of work on curriculum design for reinforcement learning agents [21–27], curriculum design for imitation learning agents is much less investigated.

Prior work on curriculum design for IRL learners has focused on two concrete settings: non-interactive and interactive. In the non-interactive setting [17, 18], the teacher provides a near-optimal set of demonstrations as a single batch. These curriculum strategies do not incorporate

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

any feedback from the learner, hence unable to adapt the teaching to the learner's progress. In the interactive setting [28], the teacher can leverage the learner's progress to adaptively choose the next demonstrations to accelerate the learning process. However, the existing state-of-the-art work [28] has proposed interactive curriculum algorithms that are based on learning dynamics of a specific IRL learner model (i.e., the learner's gradient update rule); see further discussion in Section 1.1. In contrast, we focus on designing an interactive curriculum algorithm with theoretical guarantees that is agnostic to the learner's dynamics. This will enable the algorithm to be applicable for a broad range of learner models, and in practical settings where the learner's internal model is unknown (such as tutoring systems with human learners). A detailed comparison between our curriculum algorithm and the prior state-of-the-art algorithms from [18, 28] is presented in Section 1.1.

Our approach is motivated by works on curriculum design for supervised learning and reinforcement learning algorithms that use a ranking over the training examples using a difficulty score [29–35]. In particular, our work is inspired by theoretical results on curriculum learning for linear regression models [32]. We define difficulty scores for any demonstration based on the teacher's optimal policy and the learner's current policy. We then study the differential effect of the difficulty scores on the learning progress for two popular imitation learners: Maximum Causal Entropy Inverse Reinforcement Learning (MaxEnt-IRL) [10] and Cross-Entropy loss-based Behavioral Cloning (CrossEnt-BC) [36]. Our main contributions are as follows:[1]

1. Our analysis for both MaxEnt-IRL and CrossEnt-BC learners leads to a unified curriculum strategy, i.e., a preference ranking over demonstrations. This ranking is obtained based on the ratio between the demonstration's likelihood under the teacher's optimal policy and the learner's current policy. Experiments on a synthetic car driving environment validate our curriculum strategy.
2. For the MaxEnt-IRL learner, we prove that our curriculum strategy achieves a linear convergence rate (under certain mild technical conditions), notably without requiring access to the learner's dynamics.
3. We adapt our curriculum strategy to the learner-centric setting where a teacher agent is not present through the use of task-specific difficulty scores. As a proof of concept, we show that our strategy accelerates the learning process in synthetic navigation-based environments.

1.1 Comparison to Existing Approaches on Curriculum Design for Imitation Learning

In the non-interactive setting, [18] have proposed a batch teaching algorithm (SCOT) by showing that the teaching problem can be formulated as a set cover problem. In contrast, our algorithm is interactive in nature and hence, can leverage the learner's progress (see experimental results in Section 5).

In the interactive setting, [28] have proposed the Omniscient algorithm (OMN) based on the iterative machine teaching (IMT) framework [37]. Their algorithm obtains strong convergence guarantees for the MaxEnt-IRL learner model; however, requires *exact* knowledge of the learner's dynamics (i.e, the learner's update rule). Our algorithm on the other hand is agnostic to the learner's dynamics and is applicable to a broader family of learner models (see Sections 4 and 5).

Also for the interactive setting, [28] have proposed the Blackbox algorithm (BBOX) as a heuristic to apply the OMN algorithm when the learner's dynamics are unknown—this makes the BBOX algorithm more widely applicable than OMN. However, this heuristic algorithm is still based on the gradient functional form of the linear MaxEnt-IRL learner model (see Footnote 2), and does not provide any convergence guarantees. In contrast, our algorithm is derived independent of any specific learner model and we provide a theoretical analysis of our algorithm for different learner models (see Theorems 1, 2, and 3). Another crucial difference is that the BBOX algorithm requires access to the true reward function of the environment, which precludes it from being applied to learner-centric settings where no teacher agent is present. In comparison, our algorithm is applicable to learner-centric settings (see experimental results in Section 6).

1.2 Additional Related Work on Curriculum Design and Teaching

Curriculum design. Curriculum design for supervised learning settings has been extensively studied in the literature. Early works present the idea of designing a curriculum comprising of tasks with increasing difficulty to train a machine learning model [29–31]. However, these approaches require

[1] Github repo: https://github.com/adishs/neurips2021_curriculum-teaching-demonstrations_code.

task-specific knowledge for designing heuristic difficulty measures. Recent works have tackled the problem of automating curriculum design [38, 39]. There is also an increasing interest in theoretically analyzing the impact of a curriculum (ordering) of training tasks on the convergence of supervised learner models [32, 40, 41]. In particular, our work builds on the idea of difficulty scores of the training examples studied in [32].

The existing results on curriculum design for sequential decision-making settings are mostly empirical in nature. Similar to the supervised learning settings, the focus on curriculum design for reinforcement learning settings has been shifted from hand-crafted approaches [34, 35] to automatic methods [21–23, 25]. We refer the reader to a recent survey [42] on curriculum design for reinforcement learning. The curriculum learning paradigm has also been studied in psychology literature [43–47]. One key aspect in these works has been to design algorithms that account for the pedagogical intentions of a teacher, which often aims to explicitly demonstrate specific skills rather than just provide an optimal demonstration for a task. We see our work as complementary to these.

Machine teaching. The algorithmic teaching problem considers the interaction between a teacher and a learner where the teacher's objective is to find an optimal training sequence to steer the learner towards a desired goal [37, 48–50]. Most of the work in machine teaching for supervised learning settings is on batch teaching where the teacher provides a batch of teaching examples at once without any adaptation. The question of how a teacher should adaptively select teaching examples for a learner has been addressed recently in supervised learning settings [51–56].

Furthermore, [16–18, 28, 57, 58] have studied algorithmic teaching for sequential decision-making tasks. In particular, [17, 18] have proposed batch teaching algorithms for an IRL agent, where the teacher decides the entire set of demonstrations to provide to the learner before any interaction. These teaching algorithms do not leverage any feedback from the learner. In contrast, as discussed in Section 1.1, [28] have proposed interactive teaching algorithms (namely OMN and BBOX) for an IRL agent, where the teacher takes into account how the learner progresses. The works of [57, 58] are complementary to ours and study algorithmic teaching when the learner has a differet worldview than the teacher or has its own specific preferences.

2 Formal Problem Setup

Here, we formalize our problem setting which is based on prior work on sequential teaching [28, 37].

Environment. We consider an environment defined as a Markov Decision Process (MDP) $M := \langle S; A; T; \gamma; P_0; R^E \rangle$, where the state and action spaces are denoted by S and A, respectively. $T : S \times S \times A \to [0; 1]$ is the transition dynamics, γ is the discounting factor, and $P_0 : S \to [0; 1]$ is an initial distribution over states S. A policy $\pi : S \times A \to [0; 1]$ is a mapping from a state to a probability distribution over actions. The underlying reward function is given by $R^E : S \times A \to \mathbb{R}$.

Teacher-learner interaction. We consider a setting with two agents: a teacher and a sequential learner. The teacher has access to the full MDP M and has a *target policy* π^E (e.g., a near-optimal policy w.r.t. R^E). The learner knows the MDP M but not the reward function R^E, i.e., has only access to $M \setminus R^E$. The teacher's goal is to provide an informative sequence of demonstrations to teach the policy π^E to the learner. Here, a teacher's demonstration $\xi = \{(s, a)\}_{t=0,1,\ldots}$ is obtained by first choosing an initial state $s_0 \in S$ (where $P_0(s_0) > 0$) and then choosing a trajectory, sequence of state-action pairs, obtained by executing the policy π^E in the MDP M. The interaction between the teacher and the learner is formally described in Algorithm 1. For simplicity, we assume that the teacher directly observes the learner's policy π^L_t at any time t. In practice, the teacher could approximately infer the policy π^L_t by probing the learner and using Monte Carlo methods.

Generic learner model. Here, we describe a generic learner update rule for Algorithm 1. Let $\mathbb{R}^d$ be a parameter space. The learner searches for a policy in the following parameterized policy space: $\Pi_\Theta := \{\pi_\theta : S \times A \to [0; 1]; \theta \in \Theta\}$, where $\Theta \subseteq \mathbb{R}^d$. For the policy search, the learner sequentially minimizes a loss function ℓ that depends on the policy parameter θ and the demonstration $\xi = \{(s, a)\}$ provided by the teacher. More concretely, we consider $\ell(\theta; \xi) := -\log \mathrm{P}(\xi \mid \theta)$, where $\mathrm{P}(\xi \mid \theta) = P_0(s_0) \prod_t \pi_\theta(a \mid s) T(s_{+1} \mid s, a)$ is the likelihood (probability) of the demon-

Algorithm 1 Teacher-Learner Interaction

1: **Initialization:** Initial knowledge of learner θ^L_1.
2: **for** $t = 1, 2, \ldots$ **do**
3: Teacher observes the learner's current policy θ^L_t.
4: Teacher provides demonstration ξ_t to the learner.
5: Learner updates its policy to θ^L_{t+1} using ξ_t.

stration under policy in the MDP M. At time t, upon receiving a demonstration ξ_t provided by the teacher, the learner performs the following online projected gradient descent update: $\theta_{t+1} \leftarrow \text{Proj}_\Theta[\theta_t - \eta_t g_t]$, where η_t is the learning rate, and $g_t = [\nabla_\theta \ell(\xi_t, \theta)]_{\theta=\theta_t}$. Note that the parameter θ_1 reflects the initial knowledge of the learner. Given the learner's current parameter θ_t at time t, the learner's policy is defined as $\pi^L_t := \pi_{\theta_t}$.

Teaching objective. For any policy , the value (total expected reward) of in the MDP M is defined as $V^\pi := \sum_{s,a} \sum_{t=0}^{\infty} P(S_t = s \mid \pi, M)\, \pi(a \mid s)\, R^E(s, a)$, where $P(S_t = s \mid \pi, M)$ denotes the probability of visiting the state s after steps by following the policy . Let π^L denote the learner's final policy at the end of teaching. The performance of the policy π^L (w.r.t. R^E) in M can be evaluated via $V^E - V^{\pi^L}$ [9, 59]. The teaching objective is to ensure that the learner's final policy ϵ-*approximates* the teacher's policy, i.e., $V^E - V^{\pi^L} \leq \epsilon$. The teacher aims to provide an optimized sequence of demonstrations $\{\xi_t\}_{t=1,2,\ldots}$ to the learner to achieve the teaching objective. The teacher's performance is then measured by the number of demonstrations required to achieve this objective. Based on existing work [28, 37], we assume that $\exists \beta > 0$ such that $\theta^E = \beta$ (we refer to as the *target teaching parameter*). Similar to [28], we assume that a smoothness condition holds in the policy parameter space: $|V^\theta - V^{\theta'}| \leq O(\|\theta - \theta'\|)\ \forall\ \theta, \theta' \in$. Then, the teaching objective in terms of V convergence can be reduced to the convergence in the parameter space, i.e., we can focus on the quantity $\|\theta^* - \theta_t\|$.

3 Curriculum Design using Difficulty Scores

In this section, we introduce our curriculum strategy which is based on the concept of *difficulty scores* and is agnostic to the dynamics of the learner.

Difficulty scores. We begin by assigning a difficulty score $d(\xi)$ for any demonstration w.r.t. a parameterized policy in the MDP M. Inspired by difficulty scores for supervised learning algorithms [32], we consider a difficulty score which is directly proportional to the loss function ℓ, i.e., $d(\xi) \propto g(\ell(\xi, \theta))$, for a monotonically increasing function g. Setting $g(\cdot) = \exp(\cdot)$ leads to $d(\xi) = \prod_{(s,a)\in\xi} \frac{1}{\pi(a \mid s)}$ for MDPs with deterministic transition dynamics. Based on this insight, we define the following difficulty score which we use throughout our work.

Definition 1. *The difficulty score of a demonstration w.r.t. the policy in the MDP M is given by* $d(\xi) := \prod_{(s,a)\in\xi} \frac{1}{\pi(a \mid s)}$.

Intuitively, the difficulty score of a demonstration w.r.t. an agent's policy is inversely proportional to the preference of the agent to follow the demonstration. Demonstrations with a higher likelihood under the agent's policy (higher preference) have a lower difficulty score and vice versa. With the above definition, the difficulty scores for any demonstration w.r.t. the teacher's and learner's policies (at any time t) are respectively given by $d^E(\xi) := d_\theta(\xi)$ and $d^L_t(\xi) := d_{\theta_t}(\xi)$.

General curriculum strategy. Our curriculum strategy picks the next demonstration ξ_t to provide to the learner based on a preference ranking induced by the teacher's and learner's difficulty scores. The difficulty score of a demonstration w.r.t. the teacher and learner (at any time t) is denoted by d^E and d^L_t respectively. Specifically, our curriculum strategy is given by:

$$\xi_t \leftarrow \arg\max_{\xi} \frac{d^L_t(\xi)}{d^E(\xi)}. \tag{1}$$

Teacher-centric and learner-centric settings. In the teacher-centric setting formalized in Section 2, our curriculum strategy utilizes the difficulty scores induced by the learner's current policy π^L_t and the teacher's policy π^E. From Eq. (1) and Definition 1, we obtain the following teacher-centric curriculum strategy: $\xi_t \in \arg\max_{\xi} \prod_{(s,a)\in\xi} \frac{\pi^E(a|s)}{\pi^L_t(a|s)}$.

Additionally, we also consider the learner-centric setting where a teacher agent is not present and the target policy π^E is unknown. Here, the learner can benefit from designing a self-curriculum (i.e., automatically ordering demonstrations) based on its current policy π^L_t. We adapt our curriculum strategy to this setting by utilizing task-specific domain knowledge to define the teacher's difficulty score $\Psi^E(\xi)$ for any demonstration ξ. From Eq. (1), given the learner's current policy π^L_t and the teacher's difficulty score $\Psi^E(\xi)$, the learner-centric curriculum strategy is given as follows: $\xi_t \in \arg\max_{\xi} \frac{1}{\Psi^E(\xi)} \prod_{(s,a)\in\xi} \frac{1}{\pi^L_t(a|s)}$.

Note that our curriculum strategy only requires access to the learner's and teacher's policies (π^L_t and π^E) and does not depend on the learner's internal dynamics (i.e, its update rule as mentioned in Section 2). This makes our approach more widely applicable to practical applications where it is often possible to infer an agent's policy, but the internal update rule is unkown.

4 Theoretical Analysis of Our Curriculum Strategy

In this section, we present the theoretical analysis of our curriculum strategy for two popular learner models, namely, MaxEnt-IRL and CrossEnt-BC. For our analysis, we consider the teacher-centric setting as introduced in Section 2. Our curriculum strategy obtains a preference ranking over the demonstrations to provide to the learner based on the difficulty scores (see Definition 1). To this end, we analyze the relationship between the difficulty scores (w.r.t. the teacher and the learner) of the provided demonstration and the teaching objective (convergence towards the target teaching parameter θ^*) during each sequential update step of the learner.

Given two difficulty values $\Psi^E, \Psi^L \in \mathbb{R}$, we define the feasible set of demonstrations at time t as $\mathcal{D}_t(\Psi^E, \Psi^L) := \{\xi : \Psi^E(\xi) = \Psi^E \text{ and } \Psi^L_t(\xi) = \Psi^L\}$. This set contains all demonstrations for which the teacher's difficulty score $\Psi^E(\xi)$ is equal to the value Ψ^E, and the learner's difficulty score $\Psi^L(\xi)$ is equal to the value Ψ^L. Let $\rho_t(\Psi^E, \Psi^L)$ denote the expected convergence rate of the teaching objective at time t, given difficulty values Ψ^E and Ψ^L:

$$\rho_t(\Psi^E, \Psi^L) := \mathbb{E}_{\xi \mid \Psi^E, \Psi^L}[\|\theta^* - \theta_t\|^2 - \|\theta^* - \theta_{t+1}(\xi)\|^2]; \tag{2}$$

where the expectation is w.r.t. the uniform distribution over the set $\mathcal{D}_t(\Psi^E, \Psi^L)$. Below, we analyse the differential effect of Ψ^E and Ψ^L on $\rho_t(\Psi^E, \Psi^L)$, i.e., the effect of picking demonstrations with higher or lower difficulty scores on the learning progress.

4.1 Analysis for MaxEnt-IRL Learner

Here, we consider the popular MaxEnt-IRL learner model [10, 59, 60] in an MDP M with deterministic transition dynamics, i.e., $T : S \times S \times A \to \{0, 1\}$. The MaxEnt-IRL learner model uses a parametric reward function $R : S \times A \to \mathbb{R}$ where $\theta \in \mathbb{R}^d$ is a parameter. The reward function R also depends on a feature mapping $\phi : S \times A \to \mathbb{R}^{d'}$ which encodes each state-action pair (s, a) by a feature vector $\phi(s, a) \in \mathbb{R}^{d'}$. For our theoretical analysis, we consider R with a linear form, i.e., $R_\theta(s, a) := \langle \theta, \phi(s, a) \rangle$ and $d = d'$. In our experiments, we go beyond these simplifications and consider environments with stochastic transition dynamics and non-linear reward functions.

Under the MaxEnt-IRL learner model, the parametric policy takes the following soft-Bellman form: $\pi_\theta(a|s) = \exp(Q_\theta(s, a) - V_\theta(s))$, where $V_\theta(s) = \log \sum_a \exp Q_\theta(s, a)$ and $Q_\theta(s, a) = R_\theta(s, a) + \sum_{s'} T(s'|s, a) V_\theta(s')$. For any given θ, the corresponding policy π_θ can be efficiently computed via the Soft-Value-Iteration procedure with reward R_θ (see [59, Algorithm. 9.1]). For the above setting and a given parameter θ, the probability distribution $\mathbb{P}(\xi \mid \theta)$ over the demonstration ξ takes the closed-form $\mathbb{P}(\xi \mid \theta) = \frac{\exp(\langle \theta, \phi_\xi \rangle)}{Z(\theta)}$, where $\phi_\xi := \sum_{t=0} \phi(s_t, a_t)$ and $Z(\theta)$ is a normalization factor. Then, at time t, the gradient of the MaxEnt-IRL learner is given by $g_t = \phi_{\xi_t} - \phi_{\tilde{\xi}_t}$, where

$:= \sum_{s,a} \sum_{t=0}^{1} \gamma^t P(s_t = s \mid \pi, M) \pi(a \mid s)$ (s,a) is the feature expectation vector of policy π. We note that our curriculum strategy in Eq. (1) is not using knowledge of g_t.

For the MaxEnt-IRL learner, we obtain the following theorem, which shows the differential effect of the difficulty scores (w.r.t. the teacher and the learner) on the expected rate of convergence of the teaching objective $\Delta_t(\theta^E, \theta^L)$. We note that [32] obtained similar results for linear regression learner models in the supervised learning setting.

Theorem 1. *Assume that α_t is sufficiently small for all t s.t. $\alpha_t \|g_t\|^2 \leq 2\langle \theta^E - \theta^L, g_t \rangle$, where g_t is the gradient of the MaxEnt-IRL learner. Then, for the MaxEnt-IRL learner, the expected convergence rate of the teaching objective $\Delta_t(\theta^E, \theta^L)$ is:*

monotonically decreasing with value θ^E, i.e., $\frac{\partial \Delta_t}{\partial \theta^E} < 0$, and

monotonically increasing with value θ^L, i.e., $\frac{\partial \Delta_t}{\partial \theta^L} > 0$.

Theorem 1 suggests that choosing demonstrations with lower difficulty score w.r.t. the teacher's policy and higher difficulty score w.r.t. the learner's policy would lead to faster convergence. Our curriculum strategy in Eq. (1) induces a preference ranking over demonstrations that aligns with these insights of Theorem 1. Furthermore, the following theorem states that the particular form of combining the two difficulty scores used in curriculum strategy, Eq. (1), achieves linear convergence to the teaching objective. This is similar to the state-of-the-art OMN algorithm based on the IMT framework for sequential learners [28, 37]. Importantly, unlike the OMN algorithm, our curriculum strategy does not rely on specifics of the learner model when selecting demonstrations.

Theorem 2. *Consider Algorithm 1 with the MaxEnt-IRL learner and our curriculum strategy in Eq. (1). Then, the teaching objective $\|\theta^E - \theta_t\|$ is achieved in $t = O(\log \frac{1}{\epsilon})$ iterations.*

In the above theorem, the constant terms suppressed by the $O(\,)$ notation depend on the learning rate of the learner (α_t), the distance between the learner's initial parameter/knowledge and the target teaching parameter ($\|\theta^E - \theta_1\|$), and the *richness* of the set of demonstrations obtained by executing the policy π^E in the MDP M. The *richness* notion is formally discussed in the Appendix.

4.2 Analysis for CrossEnt-BC Learner

Next, we consider the CrossEnt-BC learner model [15, 36]. In this case, the learner's parametric policy takes the following softmax form: $\pi(a|s) = \frac{\exp(H_\theta(s,a))}{\sum_{a'} \exp(H_\theta(s,a'))}$, where $H_\theta : S \times A \to \mathbb{R}$ is a parametric scoring function that depends on the parameter $\theta \in \mathbb{R}^d$ and a feature mapping $\phi : S \times A \to \mathbb{R}^{d'}$. For our theoretical analysis, we consider a linear scoring function H_θ of the form $H_\theta(s,a) := \langle \theta, \phi(s,a) \rangle$ (with $d = d'$). Then, at time step t, the gradient g_t of the CrossEnt-BC learner is given by: $g_t = \sum_{\tau=0}^{1} \mathbb{E}_{a \sim \pi_t(\cdot|s^\tau)} [\phi(s^\tau, a) - \phi(s^\tau, a^\tau)]$. In the experiments, we also consider non-linear scoring functions parameterized by neural networks.

Similar to Theorem 1, we obtain the following theorem for the CrossEnt-BC learner, which also justifies our curriculum strategy in Eq. (1).

Theorem 3. *Assume that α_t is sufficiently small for all t s.t. $\alpha_t \|g_t\|^2 \leq 2\langle \theta^E - \theta^L, g_t \rangle$, where g_t is the gradient of the CrossEnt-BC learner. Then, for the CrossEnt-BC learner, the expected convergence rate of the teaching objective $\Delta_t(\theta^E, \theta^L)$, after first-order approximation, is:*

monotonically decreasing with θ^E, i.e., $\frac{\partial \Delta_t}{\partial \theta^E} < 0$, and

monotonically increasing with θ^L, i.e., $\frac{\partial \Delta_t}{\partial \theta^L} > 0$.

We note that the proof of Theorem 3 relies on the first-order Taylor approximation of the term $\log \sum_{a'} \exp(H_\theta(s^\tau, a'))$ around θ_t (detailed in the Appendix). Due to this approximation, it is more challenging to obtain a convergence result analogous to Theorem 2.

5 Experimental Evaluation: Teacher-Centric Setting

Inspired by the works of [28, 61, 62], we evaluate the performance of our curriculum strategy, Eq. (1), in a synthetic car driving environment on MaxEnt-IRL and CrossEnt-BC learners. In particular, we consider the environment of [28] and the teacher-centric setting of Section 2.

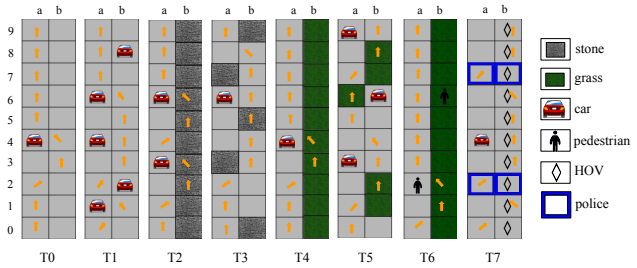

Figure 1: Car environment with 8 different types of tasks. Arrows represent the path taken by the teacher's policy.

$\phi^E(s)$	w
stone	-1
grass	-0.5
car	-5
ped	-10
car-front	-2
ped-front	-5
HOV	+1
police	0

Figure 2: Environment features $\phi^E(s)$ and reward weights w.

Car driving setup. Fig. 1 illustrates a synthetic car driving environment consisting of 8 different types of tasks, denoted as T0, T1, ⋯, T7. Each type is associated with different driving skills. For instance, T0 corresponds to a basic setup representing a traffic-free highway. T1 represents a crowded highway. T2 has stones on the right lane, whereas T3 has a mix of both cars and stones. Similarly, T4 has grass on the right lane, and T5 has a mix of both grass and cars. T6 and T7 introduce more complex features such as pedestrians, police, and HOV (high occupancy vehicles). The agent starts navigating from an initial state at the bottom of the left lane of each task, and the goal is to reach the top of a lane while avoiding cars, stones, and other obstacles. The agent's action space is given by $A = \{\text{left}, \text{straight}, \text{right}\}$. Action left steers the agent to the left of the current lane. If the agent is already in the leftmost lane when taking action left, then the lane is randomly chosen with uniform probability. We define similar stochastic dynamics for taking action right; action straight means no change in the lane. Irrespective of the action taken, the agent always moves forward.

Environment MDP. Based on the above setup, we define the environment MDP, M_{car}, consisting of 8 types of tasks, namely T0–T7, and 5 tasks of each type. Every location in the environment is associated with a state. Each task is of length 10 and width 2, leading to a state space of size 5 8 20. We consider an action-independent reward function $R^{E}{}_{\text{car}}$ that is dependent on an underlying feature vector ^{E} (see Fig. 2). The feature vector of a state s, denoted by $^{E}(s)$, is a binary vector encoding the presence or absence of an object at the state. In this work we have two types of features: features indicating the type of the current cell as stone, grass, car, ped, police, and HOV, as well as features providing some look-ahead information such as whether there is a car or pedestrian in the immediate front cell (denoted as car-front and ped-front). Now we explain the reward function $R^{E}{}_{\text{car}}$. For states in tasks of type T0-T6, the reward is given by $w;\ ^{E}(s)$ (see Fig. 2). Essentially there are different penalties (i.e., negative rewards) for colliding with specific obstacles such as stone and car. For states in tasks of type T7, there is a reward of value +1 for driving on HOV, however, if police is present while driving on HOV, a reward value of 5 is obtained. Overall, this results in the reward function $R^{E}{}_{\text{car}}$ being nonlinear.

5.1 Teaching Algorithms

Here, we introduce the teaching algorithms considered in our experiments. The teacher's near-optimal policy ^{E} is obtained via policy iteration [63]. The teacher selects demonstrations to provide to the learner using its teaching algorithm. We compare the performance of our proposed CUR teacher, which implements our strategy in Eq. (1), with the following baselines:

CUR-T: A variant of our CUR teacher that samples demonstrations based on the difficulty score ^{E} alone, and sets $^{L}_{t}$ to constant.

CUR-L: A similar variant of our CUR teacher that samples demonstrations based on the difficulty score $^{L}_{t}$ alone, and sets ^{E} to constant.

AGN: an agnostic teacher that picks demonstrations based on random ordering [28, 32].

OMN: The omniscient teacher is a state-of-the-art algorithm [28, 37], which is applicable only to MaxEnt-IRL learners. OMN requires complete knowledge of the parameter, the learner's current parameter $_{t}$, and the learner's gradients $_{t}g_{t}$. Based on this knowledge, the teacher picks demonstrations to directly steer the learner towards, i.e., by minimizing k ($_{t}$ $_{t}g_{t}$)k.

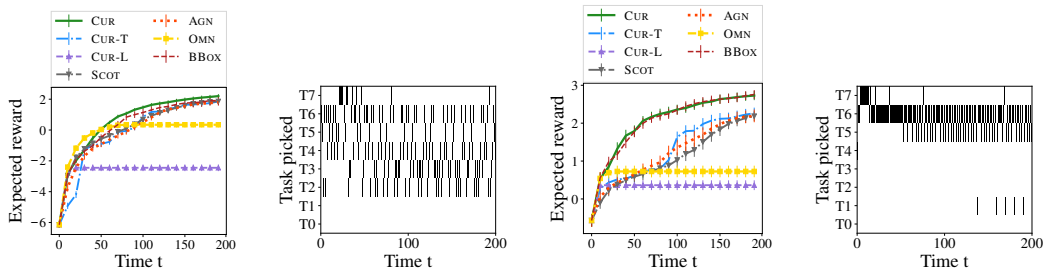

Figure 3: Learning curves and curriculum visualization for MaxEnt-IRL learners (with varying initial knowledge) trained on the car driving environment: (a) reward convergence plot and (b) curriculum generated by the CUR teacher for the learner with initial knowledge of T0; (c) reward convergence plot and (d) curriculum generated by the CUR teacher for the learner with initial knowledge of T0–T3.

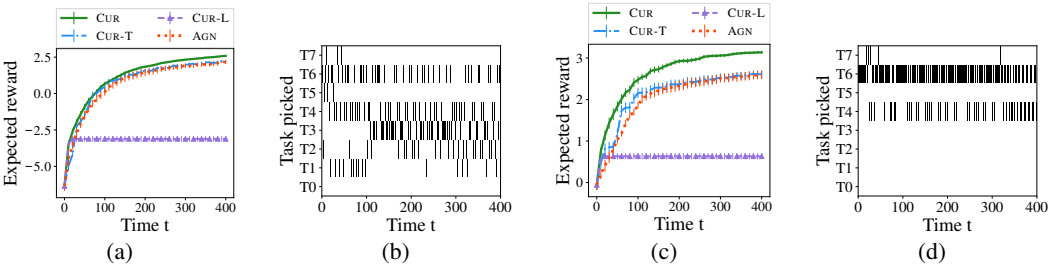

Figure 4: Learning curves and curriculum visualization for CrossEnt-BC learners (with varying initial knowledge) trained on the car driving environment: (a) reward convergence plot and (b) curriculum generated by the CUR teacher for the learner with initial knowledge of T0; (c) reward convergence plot and (d) curriculum generated by the CUR teacher for the learner with initial knowledge of T0–T3.

BBOX: The blackbox teacher [28] is designed based on the functional form of gradients for the linear MaxEnt-IRL learner model.[2] Specifically, the teacher picks a demonstration which maximizes $j \sum_{s^0;a^0} f \; {}^{L}_{t} (s^0;a^0) \quad (s^0;a^0) g R^E (s^0;a^0) j$, where denotes state visitation frequency vectors. The BBOX teacher does not require access to or the learner's current parameter $_t$; however, it requires access to the true reward function R^E.

SCOT: The SCOT teacher [18] aims to find the smallest set of demonstrations required to teach an optimal reward function to the MaxEnt-IRLlearner. The teacher uses a set cover algorithm to precompute the entire curriculum as a batch, prior to training. In our implementation, after having provided the entire batch, the teacher continues providing demonstrations selected at random.

5.2 Learner Models

Next, we describe the MaxEnt-IRL and CrossEnt-BC learner models. For the MaxEnt-IRL learner, we evaluate all the above-mentioned teaching algorithms that include state-of-the-art baselines; for the CrossEnt-BC learner, we evaluate CUR, CUR-T, CUR-L, and AGN algorithms.

MaxEnt-IRL learner. For alignment with the prior state-of-the-art work on teaching sequential MaxEnt-IRL learners [28], we perform *teaching over states* in our experiments. More concretely, at time t the teacher picks a state s_t (where $P_0(s_t) > 0$) and provides all demonstrations starting from s_t to the learner given by $_{s_t} = \quad = \quad s \; ; a \quad$ s.t. $s_0 = s_t$. The gradient g_t of the MaxEnt-IRL learner is then given by $g_t = \quad _t ; s_t \quad \Xi_{s_t}$, where (i) $\Xi_{s_t} := \frac{1}{j \Xi_{s_t} j} \quad 2 \Xi_{s_t}$, and (ii) $; s_t$ is the feature expectation vector of policy with starting state set to s_t (see Section 4.1). Based on [28], we consider the learner's feature mapping as $(s;a) = \; ^E (s)$ and the learner uses a non-linear parametric reward function $R^L (s;a) = h \; _{1:d^0};\quad (s;a) i + h \; _{d^0+1:2d^0};\quad (s;a) i^2$ where d^0 is the

[2]The BBOX teacher's objective is derived under the assumptions that the reward function can be linearly parameterized as $w \; ; \; ^E (s)$ and gradients g_t are based on the linear MaxEnt-IRL learner model. Under these assumptions, the teacher's objective can be equivalently written as $j h w \; ; g_t i j$.

dimension of $(s; a)$. As explained in [28], a linear reward representation cannot capture the optimal behaviour for M_{car}. We consider learners with varying levels of initial knowledge, i.e., the learner is trained on a subset of tasks before the teaching process starts. In this setting, for our curriculum strategy in Eq. (1) the difficulty score of a set of demonstrations associated with a starting state s is computed as the mean difficulty score of individual demonstrations in the set.

CrossEnt-BC learner. We consider the CrossEnt-BC learner model of Section 4.2 as our second learner model. The learner's feature mapping is given by $(s; a) = \mathbb{E}_{s' \sim T(\cdot|s;a)}[\ ^{E}(s')]$. A quadratic parametric form is selected for the scoring function, i.e., $H(s; a) = h_{1:d^0}; (s; a)i + h_{d^0+1:2d^0}; (s; a)i^2$, where d^0 is the dimension of $(s; a)$. We consider learners with varying initial knowledge and perform teaching over states similar to the MaxEnt-IRL learner.

5.3 Experimental results

Figs. 3a, 3c and 4a, 4c show the convergence of the total expected reward for the MaxEnt-IRL and CrossEnt-BC learners respectively, averaged over 10 runs. The CUR teacher outperforms OMN despite not requiring information about the learner's dynamics. For non-linear parametric reward functions, the MaxEnt-IRL learner no longer solves a convex optimization problem. As a result, forcing the learner to converge to a fixed parameter doesn't necessarily perform well, as seen by the poor performance of the OMN teacher in Fig. 3c. The CUR teacher is competitive with the BBOX teacher. Unlike our CUR teacher, the BBOX teacher does require exact access to the true reward function, R^E. The CUR teacher consistently outperforms the AGN and SCOT teachers, as well as both the CUR-T and CUR-L variants.

Figs. 3b, 3d and 4b, 4d visualize the curriculum generated by the CUR teacher for the MaxEnt-IRL and CrossEnt-BC learners respectively. Here, the curriculum refers to the type of task, T0–T7, associated with the demonstrations provided by the teacher to the learner at time step t. For both types of learners we see that at the beginning of training, the teacher focuses on tasks which teach skills the learner is yet to master. For example, in Fig. 4d, the teacher picks tasks T4, T6, and T7, which teaches the learner to avoid grass, pedestrians, and to navigate through police and HOV. We also notice that the CUR teacher can identify degradation in performance on previously mastered tasks, e.g., task T1 in Fig. 3d, and corrects for this by picking them again later during training.

Additional results under limited observability. In the above experiments, we consider the learner's policy to be fully observable by the teacher at every time step. Here, we study the performance of our CUR teacher under the limited observability setting, similar to [28], where the learner's policy needs to be estimated by probing the learner. The probing process is formally characterized by two parameters, $(B; k)$, where the learner's policy is probed after every B time steps and each probing step corresponds to querying the learner's policy π^L_t a total of k times from each state $s \in S$ in the MDP. The learner's policy, $\pi^L_t(a|s)\ \forall a; s$, is then approximated based on the fraction of the k queries in which the learner performed action a from state s. In between every B time steps that the learner is probed, the CUR teacher does not update its estimate of the learner's policy. We note that the $(B = 1; k \neq 1)$ setting corresponds to full observability of the learner. Fig. 5 depicts the performance of the CUR teacher for different values of $(B; k)$. Even under limited observability, the CUR teacher's performance is competitive with the full observability setting. The performance of $(B = 40; k \neq 1)$ is even slightly better at certain time steps during later stages of training compared to $(B = 1; k \neq 1)$, which is possibly due to the strategy of greedily picking demonstrations not being necessarily optimal. Also, for the limited observability setting it can be interesting to explore approaches that alleviate the need to query the full policy of the learner [64, 65].

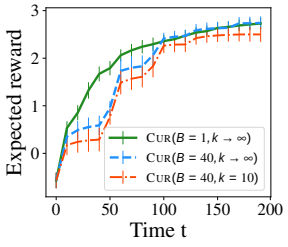

Figure 5: Learning curves for the MaxEnt-IRL learner under limited observability.

6 Experimental Evaluation: Learner-Centric Setting

In this section, we evaluate our curriculum strategy in a learner-centric setting, i.e., no teacher agent is present, and the teacher's difficulty $^{E}(\ )$ is expressed by a task-specific difficulty score (see Section 3). We evaluate our approach for training a multi-task neural policy to solve discrete optimization problems. Here, we provide an overview of the results with additional details in the Appendix.

Figure 6: Illustration of a shortest path navigation task (left) and convergence curves (right).

Figure 7: Illustration of a travelling salesman navigation task (left) and convergence curves (right).

Experiment setup. We begin by describing the synthetic navigation-based environments considered in our experiments. Our first navigation environment comprises of tasks based on the shortest path problem [66]. We represent each task with a grid-world (see Fig. 6a) containing goal cells (depicted by stars), and cells with muds and bombs (shown in brown and red respectively). The agent aims to navigate to the closest goal cell while avoiding muds and bombs. Our second navigation environment comprises of tasks inspired by the travelling salesman problem (TSP) [67, 68] (see Fig. 7a). Again we represent each task with a grid-world, where the agent's goal is to find the shortest tour which visits all goals and returns to its initial location (see Fig. 7a). Orange arrows in Figs. 6a and 7a depict the optimal path for the agent. In our experimental setup, we begin by creating a pool of tasks and split them into training and test sets. The curriculum algorithms order the training tasks during the training phase based on their strategy to speed up the learner's progress. The aim of the learner is to learn a multi-task neural policy that can generalize to new unseen tasks in the test set.

Curriculum algorithms. We compare the performance of four different curriculum algorithms: (i) the CUR algorithm picks tasks from the training set using Eq. (1) where the numerator is $|\xi|$ and the denominator ^{E} is defined by a task-specific difficulty score (detailed in the Appendix); (ii) the CUR-L algorithm picks tasks from the training set using Eq. (1) where the numerator is $|\xi|$ and the denominator is set to 1; (iii) the CUR-T algorithm picks tasks from the training set using Eq. (1) where the numerator is set to 1 and the denominator is set to ^{E}; (iv) the AGN algorithm picks tasks with a uniform distribution over the training set.

Learner model. We consider a neural CrossEnt-BC learner (see Section 4.2). The learner's scoring function H is parameterized by a 6-layer Convolutional Neural Network (CNN). The CNN takes as input a feature mapping of the agent's current position in a task, and outputs a score for each action. The learner minimizes the cross-entropy loss between its predictions and the demonstrations.

Results. Figs. 6b and 7b, show the reward convergence curves on the test set for the different curriculum algorithms averaged over 5 runs. The CUR algorithm leads to faster reward convergence compared to the AGN algorithm, which is the common approach for training a neural policy. CUR-CL is competitive with CUR in this setting which highlights the importance of the learner's difficulty.

7 Discussion and Conclusions

We presented a unified curriculum strategy, with theoretical guarantees, for the sequential MaxEnt-IRL and CrossEnt-BC learner models, based on the concept of difficulty scores. Our proposed strategy is independent of the learner's internal dynamics and is applicable in both teacher-centric and learner-centric settings. Experiments on a synthetic car driving environment and on navigation-based environments demonstrated the effectiveness of our curriculum strategy.

Our work provides theoretical underpinnings of curriculum design for teaching via demonstrations, which can be beneficial in educational applications such as tutoring systems and also for self-curriculum design for imitation learners. As such we do not see any negative societal impact of our work. Some of the interesting directions for future work include: obtaining convergence bounds for CrossEnt-BC and other learner models, designing curriculum algorithms for reinforcement learning agents based on the concept of difficulty scores, and designing approaches to efficiently approximate the learner's policy using less queries.

References

[1] S. Schaal. Learning from demonstration. In *NeurIPS*, pages 1040–1046, 1997.

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[4] Maya Cakmak and Andrea L Thomaz. Eliciting good teaching from humans for machine learners. *Artificial Intelligence*, 217:198–215, 2014.

[5] Daphna Buchsbaum, Alison Gopnik, Thomas L Griffiths, and Patrick Shafto. Children's imitation of causal action sequences is influenced by statistical and pedagogical evidence. *Cognition*, 120(3):331–340, 2011.

[6] Patrick Shafto, Noah D Goodman, and Thomas L Griffiths. A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive psychology*, 71:55–89, 2014.

[7] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[8] Stuart Russell. Learning agents for uncertain environments. In *COLT*, pages 101–103, 1998.

[9] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, page 1, 2004.

[10] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

[11] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *AISTATS*, pages 182–189, 2011.

[12] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[13] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, pages 49–58, 2016.

[14] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. *RSS*, 2017.

[15] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 2018.

[16] Thomas J Walsh and Sergiu Goschin. Dynamic teaching in sequential decision making environments. *UAI*, 2012.

[17] Maya Cakmak and Manuel Lopes. Algorithmic and human teaching of sequential decision tasks. In *AAAI*, volume 26, 2012.

[18] Daniel S Brown and Scott Niekum. Machine teaching for inverse reinforcement learning: Algorithms and applications. In *AAAI*, 2019.

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

[20] Kareem Amin, Nan Jiang, and Satinder Singh. Repeated inverse reinforcement learning. In *NeurIPS*, pages 1815–1824, 2017.

[21] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. Automatic curriculum graph generation for reinforcement learning agents. In *AAAI*, volume 31, 2017.

[22] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *CoRL*, pages 482–495, 2017.

[23] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, 2018.

[24] Wojciech Czarnecki, Siddhant Jayakumar, Max Jaderberg, Leonard Hasenclever, Yee Whye Teh, Nicolas Heess, Simon Osindero, and Razvan Pascanu. Mix & match agent curricula for reinforcement learning. In ICML, 2018.

[25] Sanmit Narvekar and Peter Stone. Learning curriculum policies for reinforcement learning. In AAMAS, pages 25–33, 2019.

[26] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Robert Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In ICLR, 2018.

[27] Martin A Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In ICML, 2018.

[28] Parameswaran Kamalaruban, Rati Devidze, Volkan Cevher, and Adish Singla. Interactive teaching algorithms for inverse reinforcement learning. In IJCAI, 2019.

[29] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. Cognition, 48(1):71–99, 1993.

[30] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In ICML, pages 41–48, 2009.

[31] Wojciech Zaremba and Ilya Sutskever. Learning to execute. arXiv preprint arXiv:1410.4615, 2014.

[32] Daphna Weinshall, Gad Cohen, and Dan Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. In ICML, 2018.

[33] Daphna Weinshall and Dan Amir. Theory of curriculum learning, with convex loss functions. arXiv preprint arXiv:1812.03472, 2018.

[34] Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. Machine learning, 23(2-3):279–303, 1996.

[35] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In ICLR, 2016.

[36] Michael Bain. A framework for behavioural cloning. In Machine Intelligence 15, pages 103–129, 1995.

[37] Weiyang Liu, Bo Dai, Ahmad Humayun, Charlene Tay, Chen Yu, Linda B Smith, James M Rehg, and Le Song. Iterative machine teaching. In ICML, 2017.

[38] Alex Graves, Marc G Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In ICML, pages 1311–1320, 2017.

[39] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In ICML, pages 2304–2313, 2018.

[40] Tianyi Zhou and Jeff Bilmes. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In ICLR, 2018.

[41] Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. Curriculum learning by optimizing learning dynamics. In AISTATS, pages 433–441, 2021.

[42] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. Journal of Machine Learning Research, 21:1–50, 2020.

[43] Mark K Ho, Michael Littman, James MacGlashan, Fiery Cushman, and Joseph L Austerweil. Showing versus doing: Teaching by demonstration. In NeurIPS, 2016.

[44] Mark K Ho, Michael Littman, and Joseph L Austerweil. Teaching by intervention: Working backwards, undoing mistakes, or correcting mistakes. In Conference of the Cognitive Science Society, 2017.

[45] Mark K Ho, Michael Littman, Fiery Cushman, and Joseph L Austerweil. Effectively learning from pedagogical demonstrations. In Conference of the Cognitive Science Society, 2018.

[46] Mark K Ho, Fiery Cushman, Michael Littman, and Joseph L Austerweil. People teach with rewards and punishments as communication not reinforcements. Journal of Experimental Psychology: General, page 520–549, 2019.

[47] Mark K Ho, Fiery Cushman, Michael Littman, and Joseph L Austerweil. Communication in action: Planning and interpreting communicative demonstrations. Journal of Experimental Psychology: General, 2021.

[48] Sally A Goldman and Michael J Kearns. On the complexity of teaching. Journal of Computer and System Sciences, 50(1):20–31, 1995.

[49] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N. Rafferty. An overview of machine teaching. CoRR, abs/1801.05927, 2018.

[50] Scott Cheng-Hsin Yang, Yue Yu, arash Givchi, Pei Wang, Wai Keen Vong, and Patrick Shafto. Optimal cooperative inference. In AISTATS, pages 376–385, 2018.

[51] Francisco S Melo, Carla Guerra, and Manuel Lopes. Interactive optimal teaching with unknown learners. In IJCAI, pages 2567–2573, 2018.

[52] Weiyang Liu, Bo Dai, Xingguo Li, Zhen Liu, James Rehg, and Le Song. Towards black-box iterative machine teaching. In ICML, 2018.

[53] Yuxin Chen, Adish Singla, Oisin Mac Aodha, Pietro Perona, and Yisong Yue. Understanding the role of adaptivity in machine teaching: The case of version space learners. In NeurIPS, 2018.

[54] Teresa Yeo, Parameswaran Kamalaruban, Adish Singla, Arpit Merchant, Thibault Asselborn, Louis Faucon, Pierre Dillenbourg, and Volkan Cevher. Iterative classroom teaching. In AAAI, 2019.

[55] Anette Hunziker, Yuxin Chen, Oisin Mac Aodha, Manuel Gomez Rodriguez, Andreas Krause, Pietro Perona, Yisong Yue, and Adish Singla. Teaching multiple concepts to a forgetful learner. In NeurIPS, 2019.

[56] Farnam Mansouri, Yuxin Chen, Ara Vartanian, Jerry Zhu, and Adish Singla. Preference-based batch and sequential teaching: Towards a unified view of models. In NeurIPS, 2019.

[57] Luis Haug, Sebastian Tschiatschek, and Adish Singla. Teaching inverse reinforcement learners via features and demonstrations. In NeurIPS, 2018.

[58] Sebastian Tschiatschek, Ahana Ghosh, Luis Haug, Rati Devidze, and Adish Singla. Learner-aware teaching: Inverse reinforcement learning with preferences and constraints. In NeurIPS, 2019.

[59] Brian D Ziebart. Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy. PhD thesis, University of Washington, 2010.

[60] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. The principle of maximum causal entropy for estimating interacting processes. IEEE Transactions on Information Theory, 59(4):1966–1980, 2013.

[61] Andrew Y Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In ICML, 2000.

[62] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Feature construction for inverse reinforcement learning. NeurIPS, 23:1342–1350, 2010.

[63] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, 2018.

[64] Alexis Jacq, Matthieu Geist, Ana Paiva, and Olivier Pietquin. Learning from a learner. In ICML, 2019.

[65] Daniel S Brown, Jordan Schneider, Anca Dragan, and Scott Niekum. Value alignment verification. In ICML, 2021.

[66] E. W. Dijkstra. A note on two problems in connexion with graphs. Numer. Math., 1:269–271, December 1959.

[67] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In ICLR, 2019.

[68] Chaitanya K. Joshi, Thomas Laurent, and Xavier Bresson. On learning paradigms for the travelling salesman problem. In NeurIPS graph representation learning workshop, 2019.

[69] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? In ICLR, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately re ect the paper's contributions and scope? [Yes] In Sections 3 and 4, we present our curriculum strategy and provide theoretical analysis as described in the introduction. The experimental results mentioned in the abstract and introduction are detailed in Sections 5 and 6.
   (b) Did you describe the limitations of your work? [Yes] We discuss the limitations of our work in Sections 5 and 7. We point out that convergence bounds for CrossEnt-BC learners have not been obtained yet and that it would be bene cial to explore approaches to estimate the learner's polic using fewer queries.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] As stated in Section 7, this work presents the theoretical underpinnings of curriculum design for teaching via demonstrations. This can be bene cial for education applications and for self-curriculum design for imitation learners. As such in the present form, we do not see any direct negative societal impacts of our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We con rm that our paper conforms with the ethics review guidelines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] In Section 4.1, we state our assumptions of a deterministic MDP and linear reward structure. Similarly, we state our assumption of a linear scoring function for the CrossEnt-BC learner in Section 4.2. The simpli cations considered are reiterated in the proofs.
   (b) Did you include complete proofs of all theoretical results? [Yes] Complete proofs of all theoretical results are included in the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code and instructions are included as a URL.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Training details are presented in the Appendix and are also present in the code.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Error bars are included in all the result graphs, as can be seen in gures.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The details are provided in the Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identi - able information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A List of Appendices

In this section, we provide a brief description of the content in the appendices of the paper.

Appendix B provides proofs for MaxEnt-IRL learner.
Appendix C provides proofs for CrossEnt-BC learner.
Appendix D provides a detailed description of the synthetic navigation-based experiments.

B Proofs for MaxEnt-IRL Learner

B.1 Auxiliary Lemma

Lemma 1. Consider the MaxEnt-IRL learner defined in Section 4.1. Then, at time step t, we have

$$\langle \theta_t, g_t \rangle = \log \frac{L_t(\theta_t)}{E(\theta_t)} + K_t,$$

where $K_t = \log \frac{Z(\theta)}{Z(\theta_t)} - \langle \theta_t, \theta_t \rangle$ is a constant independent of θ.

Proof. Consider the following:

$$\begin{aligned}
(\theta_t) &= \frac{1}{\prod_{a^t | s^t} \pi_\theta(a^t | s^t)} \\
&= \frac{P_0(s_0^t)}{P_0(s_0^t) \prod_{a^t | s^t} \pi_\theta(a^t | s^t)} \\
&\overset{(i)}{=} \frac{P_0(s_0^t)}{P_0(s_0^t) \prod_{a^t | s^t} \pi_\theta(a^t | s^t) \; T(s_{t+1}^t | s^t, a^t)} \\
&= \frac{P_0(s_0^t)}{P(\tau_t | \theta)} \\
&= \frac{P_0(s_0^t) \, Z(\theta)}{\exp(\langle \theta, \theta_t \rangle)}
\end{aligned}$$

where (i) is due to the deterministic transition dynamics of the MDP. Thus, we have:

$$\langle \theta, \theta_t \rangle = \log \frac{P_0(s_0^t) \, Z(\theta)}{(\theta_t)}.$$

Then, we get:

$$\begin{aligned}
\langle \theta_t, \theta_t - \theta_t \rangle &= \log \frac{P_0(s_0^t) \, Z(\theta)}{(\theta_t)} - \log \frac{P_0(s_0^t) \, Z(\theta_t)}{{}_t(\theta_t)} - \langle \theta_t, \theta_t \rangle \\
&= \log \frac{{}_t(\theta_t)}{(\theta_t)} \frac{Z(\theta)}{Z(\theta_t)} - \langle \theta_t, \theta_t \rangle \\
&= \log \frac{L_t(\theta_t)}{E(\theta_t)} + K_t,
\end{aligned}$$

where $K_t = \log \frac{Z(\theta)}{Z(\theta_t)} - \langle \theta_t, \theta_t \rangle$ is a constant independent of θ. □

B.2 Proof of Theorem 1

Technical conditions. Let $\Theta = \mathbb{R}^d$, and for each time step t, the learning rate η_t satisfies the following condition:

$$\eta_t^2 \|g_t\|^2 \leq 2\eta_t \langle \theta_t, g_t \rangle, \tag{3}$$

where g_t is given in Section 4.1. We decompose the gradient as $g_t = \alpha(\quad_t) + \quad$, where $\quad ? (\quad_t)$, and $\quad 2 \mathbb{R}$. Then, the above condition can be reduced to the following:

$$\quad_t^2 \; j \; j^2 k \quad\quad _t k^2 + k \; k^2 \quad\quad 2 \; _t j \; jk \quad\quad _t k^2$$

$$=) \quad\quad _t \quad\quad \frac{2j \; jk \quad\quad _t k^2}{j \; j^2 k \quad\quad _t k^2 + k \; k^2}:$$

When the gradient g_t primarily aligns with $\alpha(\quad_t)$, and has a small magnitude to control variance, the above condition further simplifies as follows:

$$_t \quad\quad \frac{2}{j \; j}:$$

Smaller values of $j \; j$ would impose less stringent condition on $_t$. From Lemma 1, one can easily observe that our curriculum strategy indeed aims to align the gradient g_t with $\alpha(\quad_t)$:

$$\arg\max \frac{L_t(\;)}{E(\;)} = \arg\max \log \frac{L_t(\;)}{E(\;)} = \arg\max f \; h \quad\quad _t; g_t(\;) i g :$$

We remark that these technical conditions are only required for our theoretical analysis, and not for our experiments.

Proof. Consider the following:

$$\begin{aligned}
_t \quad E_{;\; L} &= E_{\; _t j \; E;\; L} \left[k \quad\quad _t k^2 \quad k \quad\quad _{t+1} (\; _t) k^2 \right] \\
&= E_{\; _t j \; E;\; L} \left[k \quad\quad _t k^2 \quad k \quad\quad _t + \; _t g_t k^2 \right] \\
&= E_{\; _t j \; E;\; L} \left[\quad_t^2 k g_t k^2 \quad 2 \; _t \, h \quad\quad _t; g_t i \right] \\
&\overset{(i)}{=} 2 \; _t E_{\; _t j \; E;\; L} [\, h \quad\quad _t; g_t i \,] \\
&\overset{(ii)}{=} 2 \; _t E_{\; _t j \; E;\; L} \log \frac{L_t(\; _t)}{E(\; _t)} + K_t \\
&= 2 \; _t \log \frac{L}{E} + 2 \; _t K_t ; \qquad (4)
\end{aligned}$$

where the approximation (i) is due to Eq. (3), and (ii) is due to Lemma 1. Then, from (4), we have:

$$\frac{@ \; _t}{@ \; E} \quad\quad \frac{2 \; _t}{E} < 0; \text{ and}$$

$$\frac{@ \; _t}{@ \; L} \quad\quad \frac{2 \; _t}{L} > 0:$$

□

B.3 Proof of Theorem 2

Proof. From Lemma 1, we have that

$$\arg\max \quad\quad _t; \quad\quad = \arg\max \log \frac{L_t(\;)}{E(\;)} = \arg\max \frac{L_t(\;)}{E(\;)}:$$

Thus, our curriculum teaching algorithm picks the demonstration to provide by optimizing the following objective:

$$_t \quad\quad \arg\max \quad\quad _t; \quad :$$

For a bounded feature mapping we have that $\quad L, 8 \; .$ Any optimal solution $_t$ to the above problem satisfies: $^{\;t} = \frac{L}{k \quad _t k} (\quad\quad _t)$. Since in our setting the teacher's demonstrations are

restricted to trajectories obtained by executing policy π_E in the MDP M, we assume that within the set of available teacher's demonstrations, the optimal feature vector has the following form [28, 37]:

$$\mu_t = \mu_t(\mu_E) + \epsilon_t,$$

where $\mu_t \in \left[0, \frac{L}{\|\mu_t\|}\right]$ bounds the magnitude of the gradient in the desired direction $\nabla\ell(\mu_t)$, and ϵ_t represents the deviation from the desired direction, s.t. $\max_t \|\epsilon_t\| \le \epsilon$. We further define the following terms: $z_{max} = \max_t \|\mu_t - \mu_E\|$, $\epsilon_{max} = \max_t \epsilon_t$, and $\alpha = \min_t \alpha_t \mu_t$.

Consider the following:

$$\begin{aligned}
\|\mu_{t+1} - \mu_E\|^2 &= \left\|\mu_t + \alpha_t \epsilon_t - \alpha_t \frac{\nabla\ell(\mu_t)}{\|\nabla\ell(\mu_t)\|} - \mu_E\right\|^2 \\
&= \|\mu_t - \mu_E\|^2 + \alpha_t^2 \left\|\epsilon_t - \frac{\nabla\ell(\mu_t)}{\|\nabla\ell(\mu_t)\|}\right\|^2 - 2\alpha_t \left\langle \epsilon_t - \frac{\nabla\ell(\mu_t)}{\|\nabla\ell(\mu_t)\|}, \mu_t - \mu_E \right\rangle \\
&= \|\mu_t - \mu_E\|^2 + \alpha_t^2 \left\|\epsilon_t - \frac{\nabla\ell(\mu_t) - \nabla\ell(\mu_E)}{\|\nabla\ell(\mu_t)\|}\right\|^2 - 2\alpha_t \left\langle \epsilon_t - \frac{\nabla\ell(\mu_t) - \nabla\ell(\mu_E)}{\|\nabla\ell(\mu_t)\|}, \mu_t - \mu_E \right\rangle \\
&= \|\mu_t - \mu_E\|^2 + \alpha_t^2 \|\epsilon_t\|^2 + \alpha_t^2 \left\|\frac{\nabla\ell(\mu_t) - \nabla\ell(\mu_E)}{\|\nabla\ell(\mu_t)\|}\right\|^2 + 2\alpha_t^2 \left\langle \epsilon_t, \frac{\nabla\ell(\mu_t) - \nabla\ell(\mu_E)}{\|\nabla\ell(\mu_t)\|} \right\rangle - 2\alpha_t \left\langle \epsilon_t, \mu_t - \mu_E \right\rangle \\
&\quad - 2\alpha_t^2 \left\langle \frac{\nabla\ell(\mu_t) - \nabla\ell(\mu_E)}{\|\nabla\ell(\mu_t)\|}, \mu_t - \mu_E \right\rangle \\
&\overset{(i)}{\le} 1 + \alpha_t^2 \epsilon_t^2 - 2\alpha_t \mu \|\mu_t - \mu_E\|^2 + \alpha_t^2 \epsilon^2 + L^2 \\
&\quad + 2\alpha_t (1 - \alpha_t \mu) \|\mu_t - \mu_E\| + 2\alpha_t (1 - \alpha_t \mu) L \|\mu_t - \mu_E\| + 2\alpha_t^2 L \\
&\overset{(ii)}{\le} (1 - \alpha_t \mu)^2 \|\mu_t - \mu_E\|^2 + \alpha_t^2 (\epsilon + L)^2 + 2\alpha_t (1 - \alpha_t \mu)(\epsilon + L) z_{max} \\
&\overset{(iii)}{\le} (1 - \alpha\mu)^2 \|\mu_t - \mu_E\|^2 + \alpha_{max}^2 (\epsilon + L)^2 + 2\alpha_{max} (1 - \alpha\mu)(\epsilon + L) z_{max} \\
&\overset{(iv)}{\le} (1 - \alpha\mu)^2 \|\mu_t - \mu_E\|^2 + \alpha_{max} \{1 + 2(1 - \alpha\mu) z_{max}\} (\epsilon + L);
\end{aligned}$$

where (i) uses the inequalities $\epsilon_t \le L$, and $\|\mu_t\| \le$, along with the Cauchy-Schwarz inequality; (ii) utilizes the fact that $\|\mu_t\| \le z_{max}$; (iii) is obtained by substituting $\alpha = \min_t \alpha_t \mu_t$, and $\alpha_{max} = \max_t \alpha_t$; (iv) is obtained when $\alpha_{max}(\epsilon + L) \le 1$. Note that the inequality (i) is valid when $1 - \alpha_t \mu_t > 0, \forall t$.

With the inequality $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$ for positive a, b, and utilizing recurrence, we obtain:

$$\begin{aligned}
\|\mu_{t+1} - \mu_E\| &\le (1 - \alpha\mu) \|\mu_t - \mu_E\| + \sqrt{\alpha_{max} \{1 + 2(1 - \alpha\mu) z_{max}\} (\epsilon + L)} \\
&\le (1 - \alpha\mu)^t \|\mu_1 - \mu_E\| + \sqrt{\alpha_{max} \{1 + 2(1 - \alpha\mu) z_{max}\} (\epsilon + L)} \sum_{s=0}^{t} (1 - \alpha\mu)^s \\
&= (1 - \alpha\mu)^t \|\mu_1 - \mu_E\| + \sqrt{\alpha_{max} \{1 + 2(1 - \alpha\mu) z_{max}\} (\epsilon + L)} \frac{1}{\alpha\mu} \\
&\le \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon;
\end{aligned}$$

for $t = \left\lceil \log_{1-\alpha\mu} \frac{\alpha\mu}{2\|\mu_1 - \mu_E\|} \right\rceil = O\left(\log \frac{1}{\epsilon}\right)$, and $\alpha_{max}(\epsilon + L) \le \frac{\alpha^2 \mu^2 \epsilon^2}{4\{1 + 2(1 - \alpha\mu) z_{max}\}}$. □

C Proofs for CrossEnt-BC Learner

C.1 Auxiliary Lemma

Lemma 2. Consider the CrossEnt-BC learner defined in Section 4.2. Then, at time step t, we have

$$\langle \epsilon_t, g_t \rangle \le \log \frac{\ell_t(\theta_t)}{\ell_E(\theta_t)}.$$

Proof. Consider the following:

$$\begin{aligned}
\log \ (\ _t) &= \log \ \prod \ a^t j s^t \\
&= \sum \ \log \ a^t j s^t \\
&= \sum \ \log \ \sum_{a^0} \exp \ H \ s^t ; a^0 \ \sum \ H \ s^t ; a^t \\
&\overset{(i)}{=} \sum \ \log \ \sum_{a^0} \exp \ H \ _t \ s^t ; a^0 \ \sum \ \Big[\ _t ; \ E_{a^0 \ _t (j s^t)} \ (s^t ; a^0) \ \Big] \ ; \ (s^t ; a^t)
\end{aligned}$$

where (i) is due to the first-order Taylor approximation of $\log \sum_{a^0} \exp \ H \ s^t ; a^0$ around $_t$. Then, we have:

$$\begin{aligned}
\log \frac{ _t (\ _t)}{ (\ _t)} &= \log \ _t (\ _t) \ \log \ (\ _t) \\
&\quad + \Big[\ _t ; \ \sum \ (s^t ; a^t) \ \sum \ E_{a^0 \ _t (j s^t)} \ (s^t ; a^0) \Big] \\
&= h \ _t ; g_t i
\end{aligned}$$

□

C.2 Proof of Theorem 3

Technical conditions. Let $= R^d$, and for each time step, the learning rate $_t$ satisfies the following condition:

$$^2_t \, k g_t k^2 \ 2 \ _t \, j h \ _t ; g_t i j \ ; \tag{5}$$

where g_t is the gradient of the CrossEnt-BC learner as given in section 4.2. We can further simplify the above condition, similar to Section B.2.

Proof. Consider the following:

$$\begin{aligned}
_t \ ^{E} ; \ ^{L} &= E_{ \ _t j \ ^{E} ; \ ^{L}} \Big[k \ \ _t k^2 \ k \ \ _{t+1} \ (\ _t) k^2 \Big] \\
&= E_{ \ _t j \ ^{E} ; \ ^{L}} \Big[k \ \ _t k^2 \ k \ \ _t + \ _t g_t k^2 \Big] \\
&= E_{ \ _t j \ ^{E} ; \ ^{L}} \Big[\ ^2_t \, k g_t k^2 \ 2 \ _t \, h \ \ _t ; g_t i \Big] \\
&\overset{(i)}{=} 2 \ _t E_{ \ _t j \ ^{E} ; \ ^{L}} [\ h \ \ _t ; g_t i] \\
&\overset{(ii)}{=} 2 \ _t E_{ \ _t j \ ^{E} ; \ ^{L}} \ \log \frac{ ^{L}_t (\ _t)}{ ^{E} (\ _t)} \\
&= 2 \ _t \log \frac{ ^{L}}{ ^{E}} ;
\end{aligned} \tag{6}$$

where the approximation (i) is due to Eq. (5), and (ii) is due to Lemma 2. Then, from (6), we have:

$$\begin{aligned}
&\frac{@ \ _t}{@ \ ^{E}} \ \frac{2 \ _t}{ ^{E}} < 0; \text{ and} \\
&\frac{@ \ _t}{@ \ ^{L}} \ \frac{2 \ _t}{ ^{L}} > 0:
\end{aligned}$$

□

D Additional Details for Learner-Centric Experiments

In this appendix, we present additional experimental details for the synthetic navigation-based environments considered in Section 6.

D.1 Environment MDPs

We first formally define the environment MDPs for the shortest path and TSP inspired environments described in Section 6.

Shortest path environment. A task in the shortest path environment is represented by a grid-world containing the agent, goals, muds, and bombs. Each possible configuration of a grid-world, including the agent's location and direction, is associated with a state in the shortest path environment MDP, M_{path}. The size of the state space sees a combinatorial growth with the size of the grid, corresponding to different ways of placing bombs/muds/goals. Hence, the state-space is intractably large to enumerate. The agent's action space consists of 3 actions: $\mathcal{A} = \{$move; left; right$\}$. The actions left or right changes the agent's direction accordingly. The agent moves one step forward in its current direction with the action move. The environment reward function $R^{E_{path}}$ has a -1 reward value for each action performed by the agent. Reaching a goal cell has a $+10$ reward value. There is a reward value of -1 for encountering a cell with mud and a reward value of -5 for encountering a bomb. Reaching a goal or a bomb ends the agent's episode.

Each state s is characterized by a feature mapping $\phi_{E_{path}}(s)$ which encodes the agent's location and direction, as well as the position of bombs, muds, and goals in the grid-world. In our environment, we consider grid-worlds of size 6×6, and each cell in the grid has a binary feature vector of length 7 as shown in Table 1. The first 4 features are a one-hot encoding representation of the agent's location and direction in the grid-world. The last 3 binary features represent the presence or absence of either a mud, bomb, or goal respectively at a cell. Consequently, the feature mapping $\phi_{E_{path}}(s)$ is of dimension $6 \times 6 \times 7$.

TSP environment. A task in the TSP environment is represented by a grid-world containing the agent and goal cells. Each possible configuration of a grid-world is associated with a state in the TSP environment MDP, M_{tour}, similar to M_{path}. The agent's action space $\mathcal{A} = \{$move; left; right$\}$ is defined the same as for M_{path}. In this environment, the reward function $R^{E_{tour}}$ has a $+10$ reward value for completion of a successful tour, i.e., arriving back at the initial location after having visited all the goals in the grid-world. Similar to the shortest path environment, there is a reward value of -1 for each action performed by the agent. The agent's episode ends on the completion of a successful tour or after a certain time horizon.

Each state s in the TSP environment is characterized by a feature mapping $\phi_{E_{tour}}(s)$ similar to the shortest path environment. We again consider grid-worlds of size 6×6, and each cell in the grid has a binary feature vector of length 5 as shown in Table 2. The first 4 features are a one-hot encoding representation of the agent's location and direction in the grid-world. The last binary feature represents the presence or absence of a goal at a given cell. Hence, the feature mapping $\phi_{E_{tour}}(s)$ is of dimension $6 \times 6 \times 5$.

D.2 Dataset Generation

Here, we outline the dataset generation process. We create separate training, validation, and test sets for both of our navigation environments. Further, optimal paths were computed for all the tasks in the training set for both environments. These are provided as demonstrations to the learner during the training phase. In the case of multiple optimal paths for a task, each optimal path was included as a unique demonstration.

Agent facing North
Agent facing South
Agent facing West
Agent facing East
Mud
Bomb
Goal

Table 1: Shortest path task features

Agent facing North
Agent facing South
Agent facing West
Agent facing East
Goal

Table 2: TSP task features

Algorithm 2 Scheduling Mechanism

Initialization: parameters a, b and total training epochs N.
for Epoch $e = 1, 2, \ldots, N$ do
  Curriculum strategy computes a preference over all demonstrations.
  Scheduling size is computed as $X = \begin{cases} b\|\mathcal{D}\| + \frac{e}{aN}(1 - b)\|\mathcal{D}\| & \text{if } e < aN \\ \|\mathcal{D}\| & \text{otherwise} \end{cases}$
  The X most preferred demonstrations are provided to the learner in random batches.

Shortest path environment. For the shortest path navigation tasks, we sample grid-worlds containing several muds and bombs, both in the range $\{0, \ldots, 12\}$. The agent's initial position and location of goals, muds, and bombs are all sampled at random without overlap. The training, validation, and test sets contain 100, 10, 30 grid-worlds respectively for each combination of muds and bombs, leading to datasets of size 16900, 1690, and 5070 respectively. Additionally, each dataset contains an equal percentage of grid-worlds with a single goal cell and with two goal cells.

TSP environment. For the TSP navigation tasks, we sample grid-worlds containing goal cells in the range $\{2, \ldots, 4\}$. The agent's initial position and location of goals are sampled at random without overlap. The training, validation, and test sets contain 200, 100, 500 grid-worlds respectively for each unique number of goal cells in a task, leading to datasets of size 1500, 300, and 1500 respectively.

D.3 Teacher's Difficulty Score

As explained in Sections 3 and 6, for the learner-centric setting we define the teacher's difficulty $^{E}(\ )$ using a task-specific difficulty.

Shortest path environment. For the shortest path tasks we define the following difficulty score:

$$^{E}(\ ) = \frac{\#\text{goals} \quad \#\text{optimal\_paths}}{\text{optimal\_reward}}. \tag{7}$$

Intuitively, the difficulty score in Eq. (7) is proportional to the difficulty of a task as the greater the number of goals present and optimal paths, the more challenging the task is for the learner. Additionally, a higher optimal reward implies a shorter path to a goal that is less challenging for the learner.

TSP environment. For the TSP tasks we define the teacher's difficulty score as:

$$^{E}(\ ) = \frac{\#\text{goals}}{\text{optimal\_reward} \quad \text{greedy\_gap}}; \tag{8}$$

where *greedy gap* is defined as the difference in reward between the optimal tour and the greedy tour for the given task. In the greedy tour, the agent repeatedly navigates to the closest goal which has not been visited yet. Once all goals have been visited, the agent returns to its initial location. The greedy tour is not necessarily the optimal tour for a task.

Following a similar intuition as before, we see that the difficulty score of Eq. (8) is proportional to the difficulty of a task. The greater the number of goals and the lower the optimal reward, the greater the difficulty of the task for the learner. Further, tasks with a larger *greedy_gap* are more complex for the learner.

In both Eqs. (7) and (8) the denominator for the training tasks are linearly transformed to make all values ≤ 1.

D.4 Scheduling Mechanism

As commonly done in prior work [32, 69] when training neural networks using curriculum learning, we incorporate randomization in the training process for our CURROT algorithm and its variants using a scheduling mechanism. Demonstrations of higher preference are prioritized at the beginning of

Convolution	Input feature mapping $6 \times 6 \times d$
	Conv2D, kernel size = 3, padding = 1, $d \to 32$ ReLU
Residual Block 1	Conv2D, kernel size = 3, padding 1, $32 \to 32$ ReLU Conv2D, kernel size = 3, padding 1, $32 \to 32$ ReLU Conv2D, kernel size = 3, padding 1, $32 \to 32$ ReLU
Residual Block 2	Conv2D, kernel size = 3, padding 1, $32 \to 32$ ReLU Conv2D, kernel size = 3, padding 1, $32 \to 32$ ReLU Conv2D, kernel size = 3, padding 1, $32 \to 32$ ReLU
Fully Connected	Linear, $6 \times 6 \times 32 \to 512$
Fully Connected	Linear, $512 \to 256$
Fully Connected	Linear, $256 \to 3$

Table 3: Network architecture

training, while during later stages, all demonstrations are provided with uniform probability to the learner.

In our experiments, we use a linear scheduling mechanism [69], where the first training epoch includes a fraction b of the total demonstrations. The number of demonstrations included grows linearly every subsequent epoch such that by the time a fraction a of the total epochs are completed, all the demonstrations are included. Algorithm 2 details the scheduling mechanism. The demonstrations in an epoch are provided to the learner in randomly ordered batches. In our experiments we set $a = 0.8$ and $b = 0.5$.

D.5 Learner model

Training hyperparameters. For both navigation environments, the learners were trained for 40 epochs with an initial learning rate of 0.01 and a batch size of 32 demonstrations. The learning rate was decayed by a factor of 0.5 after every 500 batches of demonstrations. The learning rate decay rule ensures the learning rate is consistent across the different curriculum algorithms since CUR and its variants utilize a different number of training tasks in each epoch due to the scheduling mechanism. Our models were trained on Nvidia Tesla V100 GPUs.

Network Architecture The learner's neural network takes as input the feature mapping $\phi(s) \in \mathbb{R}^{6 \times 6 \times d}$ of a state s. The dimension of the feature mapping is given by $6 \times 6 \times d$, where $d = 7$ for states in the shortest path navigation environment and $d = 5$ for states in the TSP navigation environment. In turn, the learner's neural network outputs a vector of size 3, which provides a probability distribution over actions after the softmax function is applied. The architecture of the neural network is provided in Table 3.

D.6 Curriculum Visualization

In addition to the results presented in Section 6, we visualize the curriculum generated by our CUR algorithm for the shortest path and TSP environments in Figs. 8 and 9 respectively. In Figs. 8 and 9, the y-axis represents different features of the tasks provided to the learner, normalized in the range $[0, 1]$. The x-axis represents the number of demostrations provided to the learner.

Shortest path environment. Fig. 8 shows that at the beginning of training, the CUR algorithm picks tasks with a fewer goals and a higher number of muds/bombs. We hypothesize that this teaches the agent how to avoid muds and bombs while navigating to a goal. During later stages of training, CUR picks tasks with higher optimal rewards and a greater number of goals. Here we

Figure 8: Shortest path environment curriculum visualization.

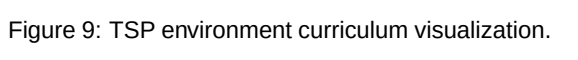

Figure 9: TSP environment curriculum visualization.

believe the agent is taught how to identify the path with maximum reward among all paths that lead to a goal. Essentially the learner is rst taught the general navigation task followed by the most dif cult concept of deciding the optimal path.

TSP environment. Fig. 9 illustrates that at the beginning of training μ_C selects tasks with a greater number of goals, but with a low greedy gap. This would teach the learner the general navigation problem of visiting all goals. As training progresses, μ_C picks tasks with a greater greedy gap. We hypothesize that these tasks teach the learner the most dif cult concept of planning the optimal tour.