On Complexity of Teaching a Family of Linear Behavior Cloning Learners

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study optimal teaching for a family of Behavior Cloning learners that learn using a linear hypothesis class. In this setup, a knowledgeable teacher can demonstrate a dataset of state and action tuples, and is required to teach an optimal policy to an entire family of BC learners using smallest possible dataset. We analyse the linear family and design a novel teaching algorithm called 'TIE' that achieves the instance optimal teaching complexity for the entire family. However, we show that this problem is NP-hard for action spaces with $|\mathcal{A}| > 2$ and provide an approximation algorithm with a $\log(|\mathcal{A}| - 1)$ guarantee on the optimal teaching size. We present empirical results to demonstrate the effectiveness of our TIE algorithm compared to various baselines in different real-world teaching environments.

1 Motivation

Behavior Cloning (BC) [7, 13, 33] is an important paradigm of learning in Reinforcement Learning(RL), that has been applied extensively to solve real-world problems like teaching machines to drive autonomous vehicles [26, 27], fly planes [30], perform robotic manipulations [20] etc. These real-world environments have large state space where the ability to generalize using linear or neural hypothesis class becomes essential for effective learning.

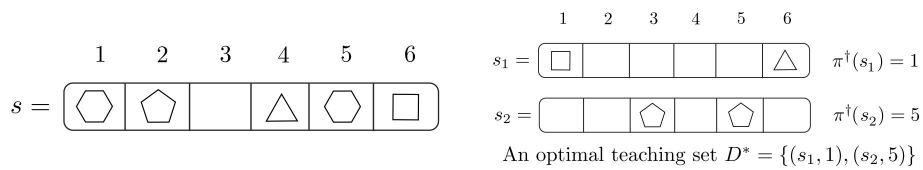
However, naively teaching an optimal policy to a BC learner using i.i.d. trajectories often demands a sample size that scales with the horizon length and the complexity of the learner's hypothesis class [4, 29]. In scenarios like teaching machines to drive cars, an expert teacher may possess the knowledge of a (near)-optimal policy and the hypothesis class employed by the learner and can leverage this knowledge to construct a small, non-i.i.d. dataset to teach the target policy to the BC learner far more efficiently. This problem is known as optimal Machine Teaching and the size of smallest teaching set so produced is called Teaching Dimension [16, 35].

Several existing works in machine teaching have studied teaching individual linear learners like linear support vector machines (SVM) [21] and linear perceptron [25] but mainly in classification settings. These surrogate learners exhibit optimization biases that arguably make them easier to teach. Furthermore, the optimal teaching set so produced is very learner specific and does not work for other linear learners. In contrast, in many scenarios like teaching a classroom of students [36], the teacher needs to provide a dataset that can teach an entire family of learners and it cannot base its teaching on the bias of individual surrogate learners. In this work, we focus on the task of optimally teaching a family of linear BC learners that satisfy the consistency property, i.e., they all produce a (subset of) hypothesis that is consistent with a dataset. We seek to answer the following question:

What is the smallest dataset required to teach a policy to a family of consistent linear BC learners?

To illustrate the challenge of optimally teaching the linear BC family using a minimal dataset, consider the following example below.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.



(a) (b)

Figure 1: a) A board in a "Pick the Right Diamond" game. In this example 1, the target policy says to pick the diamond with the highest edge breaking the tie in favor of the rightmost slot if any. There are a total of $5^n - 1$ candidate teaching state and action pairs. We ask what is the minimum set of demonstrations of such boards would allow the teacher to teach the target policy to consistent linear learners. b) Only two carefully chosen demonstrations are sufficient to teach.

Example 1 (Pick the Right Diamond)**.** *The game is shown in Figure 1a. There is a board with $n = 6$ slots where each slot can have one of 4 different types of diamonds or can be empty. The game rule says that one must pick the most expensive diamond i.e. one with the highest number of edges, first; and if there are ties one must pick the rightmost one. The game continues until the board is empty. The teacher wants to find a minimal demonstration set to convey this rule to the agent.*

There are $5^n - 1$ number of states with $\mathcal{A} = [n]$. Consider the family of consistent linear BC learners with a two-dimensional feature space denoting slot index and number of edges in the slot. A naive teacher would demonstrate target action in all $5^n - 1$ states which grows exponentially with n. However, a clever teacher succeeds by just demonstrating two states(refer to Section 4.1 for complete results), thereby significantly saving the teaching cost from $O(5^n)$ to 2.

Towards this end, we make the following contributions:

1. We formulate the problem of optimally teaching a family of linear BC learners and show that this problem reduces to optimally teaching the hardest (unbiased) member in the family, i.e., a version space BC learner (Lemma 1).
2. We characterize optimal teaching in terms of covering extreme rays of version space cone and design a novel algorithm called 'TIE' 1 to optimally teach the linear BC family (Theorem 4).
3. However, as shown in Theorem 7, solving this problem is NP-hard for instances with $|\mathcal{A}| > 2$. We propose an efficient, approximately optimal algorithm with an approximation ratio guarantee of $\log(|\mathcal{A}| - 1)$ on teaching dimension (Theorem 8).
4. Through a set of experiments on real-world environments, we demonstrate the effectiveness of our TIE algorithm compared to other baselines (Section 4).

2 Problem Formulation

Consider a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \gamma, \mu)$ where $\mathcal{S}$ is a state space, $\mathcal{A}$ a finite action space, R is reward function, P is transition distribution, γ is the discount factor and μ is the initial state distribution. For simplicity, we assume that $\mathcal{S}$ is finite, however, we also extend our algorithm to an infinite setting under reasonable assumption later on. Let $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ be a feature representation function. Given a fixed $w \in \mathbb{R}^d$, it induces a set of policies Π_w given as follows:

$$\forall s \in \mathcal{S},\, \Pi_w(s) = \Delta\left(\arg\max_{a \in \mathcal{A}} w^\top \phi(s, a)\right).$$

Let $\Pi = \cup_{w \in \mathbb{R}^d} \Pi_w$ and $\Pi_{\text{Det}} = \{w \in \Pi : \Pi_w \in \mathcal{A}^{\mathcal{S}}\}$ denote all stochastic and deterministic policy induced by $\mathcal{H} = \mathbb{R}^d$. The value of policy $\pi \in \Pi$ is denoted by $V_\mu^\pi = \mathbb{E}\left[\sum_{t=0}^\infty \gamma^t r(s_t, a_t)\right]$. An optimal policy π^* is one that achieves an optimal value $\pi^* = \arg\max_{\pi \in \Delta(\mathcal{A})^{\mathcal{S}}} V_\mu^\pi$.

2.1 The Learner Family

We consider a Behavior Cloning (BC) learner $\mathcal{L} : D \to 2^{\mathcal{H}}$ that learns using a linear hypothesis class $\mathcal{H} = \mathbb{R}^d$ and only has access to a teaching dataset $D \subseteq \mathcal{S} \times \mathcal{A}$. It reduces the problem of learning an optimal policy to a supervised learning problem [1, 29] by doing empirical risk minimization (ERM),

$$\mathcal{L}(D) \leftarrow \arg\min_{w\in\mathcal{H},\pi\in\Pi_w} \sum_{(s,a)\in D} \mathbb{E}_{a'\sim\pi(s)}[l_{\mathcal{L}}(a',a)].$$

Note that $\mathcal{L}(D)$ is the entire set of ERM hypothesis minimizer obtained by minimizing learner-specific loss function $l_{\mathcal{L}} : \mathcal{A} \times \mathcal{A} \to \mathbb{R}^+$. During deployment in MDP environment, the learner arbitrarily chooses a consistent $w \in \mathcal{L}(D)$ and a $\pi \in \Pi_w$, thereby suffering a worst-case value risk of $R_V(D;\mathcal{L}) = V_\mu^{\pi^*} - \min_{w\in\mathcal{L}(D),\pi\in\Pi_w} V_\mu^\pi$.

Consistent Linear BC Learner: In this work, we consider teaching a family of linear BC learners, denoted by $\mathfrak{C}$, that always outputs a (subset of) consistent hypotheses i.e. on input dataset D, the ERM hypothesis set agrees with D, i.e., $\forall w \in \mathcal{L}(D), \pi \in \Pi_w, \pi(s) = a,\ \forall(s,a) \in D$.

Note that $\mathcal{L}(D)$ is non-empty if D is generated by a deterministic policy $\pi \in \Pi_{\text{Det}}$, i.e., $\pi(s) = a, \forall(s,a) \in D$. In linear classification settings, learners like linear SVM, perceptron, and logistic regression are some popular examples of consistent learners. We note that not all consistent learners are created equal; many exhibit strong biases towards certain hypotheses, influenced by their surrogate loss functions [14] or iterative gradient-based learning procedures [32]. In contrast, a linear version space BC learner is one of the simplest consistent learners in $\mathfrak{C}$ as it maintains the entire version space of consistent hypotheses without any specific bias for one hypothesis over the other.

Linear Version Space (LVS) BC Learner: We consider a consistent linear BC learner that maintains the entire version space of hypothesis $\mathcal{V}(D)$ consistent with dataset D given as follows:

$$\mathcal{V}(D) = \{w \in \mathbb{R}^d : w^\top \psi_{sab} > 0,\ \forall(s,a) \in D, a \neq b\} \tag{1}$$

where $\psi_{sab} := \phi(s,a) - \phi(s,b)$ is the feature difference vector induced for strictly preferring action a over b in state s. Equivalently, a version space learner can be seen as an ERM learner minimizing zero-one loss function $\mathcal{L}_{0\text{-}1}$, i.e., $\mathcal{V}(D) = \mathcal{L}_{0\text{-}1}(D)$.

Remark 1. *We use the following notation:* $\Psi(D)$ *is the set of all feature difference vectors induced by* D *and is defined as* $\Psi(D) = \{\psi_{sab} : (s,a) \in D, b \in \mathcal{A}, b \neq a\}$. *We denote the primal cone of* $\Psi(D)$ *induced by dataset* D *as* $\text{cone}(\Psi(D)) := \{\sum_{\psi\in\Psi(D)} \lambda_\psi \psi : \lambda_\psi \geq 0, \lambda \neq 0\}$*, and its dual as* $\text{cone}^*(\Psi(D)) := \{w \in \mathbb{R}^d : \langle w, \psi\rangle > 0, \forall\psi \in \Psi(D)\}$. *Note that* $\mathcal{V}(D)$ *is the dual of* $\Psi(D)$. *Refer to Example 2 for a concrete example.*

2.2 The Teacher

In our setup, there exists a helpful teacher who controls the demonstration dataset $D \subseteq \mathcal{S} \times \mathcal{A}$ provided to the learner and has the following teaching objective:

It wants to unambiguously teach a target optimal policy π^* *to the entire family of consistent linear BC learners* $\mathfrak{C}$ *using as few demonstrations as possible.*

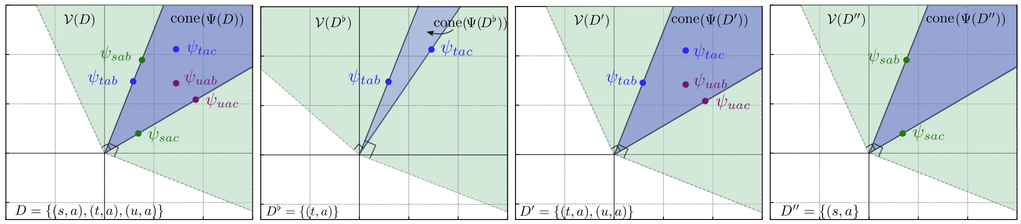
Formally, given a teaching instance $(\mathcal{M}, \phi, \pi^*)$, the optimal teaching problem of the teacher is defined by the following optimization problem :

$$\begin{aligned} (\textbf{Teach-}\mathfrak{C}) \qquad & D^* \leftarrow \min_{D\subseteq\mathcal{S}\times\mathcal{A}} && |D| \\ \text{s.t.} \qquad & \max_{\mathcal{L}\in\mathfrak{C}, w\in\mathcal{L}(D),\pi\in\Pi_w} && R_{\mathcal{L}}(\pi,\pi^*) = 0 \end{aligned} \tag{2}$$

where $R_{\mathcal{L}}(\pi,\pi') = \sum_{s\in\mathcal{S}} l_{\mathcal{L}}(\pi(s),\pi'(s))$ is a learner specific true risk. We note that the constraint requires the teacher to jointly teach all consistent learner to zero risk. The size of the optimal teaching set $TD(\mathfrak{C}) = |D^*|$ is called the teaching dimension of the class $\mathfrak{C}$.

Remark 2. *Teaching the entire family* $\mathfrak{C}$ *is not free, i.e., if the teacher knows the bias of individual learners, it can possibly teach them more efficiently. For example, an optimal SVM just requires two examples to teach in* $\mathbb{R}^d$*[21]. However, such teaching sets are not even a valid teaching set for other learners in the family* $\mathfrak{C}$ *like version space learners, see Figure 3a for an example.*

In finite state space, a naive teacher could succeed in teaching by demonstrating a full dataset $D_{\mathcal{S}} = \{(s,\pi^*(s)) : s \in \mathcal{S}\}$ defined on the entire state space, which can be practically infeasible in



(a) (b) (c) (d)

Figure 2: A simple illustration on importance of extreme rays. D, D', D'' succeed in teaching but $D^\flat$ fails depending on if they cover the extreme rays of $\mathrm{cone}(\Psi(D))$.

large state space environments. However, a clever teacher can utilize the linear feature representation structure of the learner to find a much smaller dataset to teach the target optimal policy.

We assume that π^* is realizable under a linear policy family which has been widely used in both learning theory [23, 31] and optimal teaching literature [17, 37].

Assumption 1 (Realizability)**.** *A policy $\pi : \mathcal{S} \times \mathcal{A}$ is realizable under a linear policy family if and only if $\exists w \in \mathbb{R}^d$ s.t. $\forall s \in \mathcal{S}, \{\pi(s)\} = \arg\max_{a \in \mathcal{A}} w^\top \phi(s,a)$.*

In general, our teacher can teach any realizable policy $\pi : \mathcal{S} \to \mathcal{A}$ to the learner, but for simplicity, we only focus on teaching an optimal π^*. We note that under realizability, $\psi_{s,\pi^*(s),a} \neq 0, \forall a \neq \pi^*(s)$ and $\mathcal{V}(D_\mathcal{S})$ is a non-empty cone in $\mathbb{R}^d$ as shown in figure 2a. The following lemma connects optimal teaching the entire family $\mathfrak{C}$ to optimally teaching a linear version space BC learner. The proof of this lemma can be found in the Appendix.

Lemma 1. *Optimally teaching the family of consistent linear BC learners is equivalent to optimally teaching linear version space BC learners.*

Hence, the teacher can achieve its objective of optimally teaching family $\mathfrak{C}$ by just focusing on optimally teaching linear version space (LVS) BC learner. From now on, we will focus on optimally teaching a target optimal policy π^* to a LVS learner given by the following optimization problem:

$$\begin{aligned} (\textbf{Teach-LVS}) \qquad & D^* \leftarrow \min_{D \subseteq \mathcal{S} \times \mathcal{A}} \quad |D| \\ \text{s.t.} \quad & \forall w \in \mathcal{V}(D), \pi \in \Pi_w, \forall s \in \mathcal{S}, \quad \pi(s) = \pi^*(s). \end{aligned} \tag{3}$$

This requires the teacher to find a minimal data D^* that induces π^* uniquely on the version space.

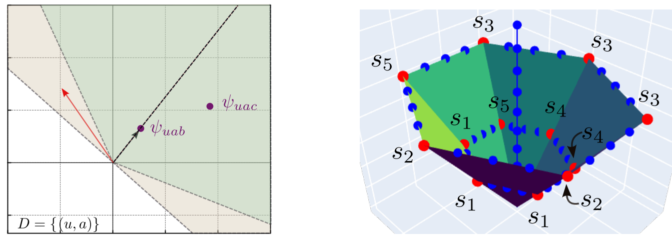
Previous works have studied the problem of optimal teaching of version space learners, but have mostly been limited to either a finite hypothesis setting [6, 16] or highly structured hypothesis classes like axis-aligned rectangles [12, 16] which is very different from our linear version space setting. Before delving into the algorithm, we present an illustrative example in $\mathbb{R}^2$.

Example 2 (An instance of teaching linear version space BC learner in $\mathbb{R}^2$)**.** *Let $\mathcal{S} = \{s, t, u\}$, $\mathcal{A} = \{a, b, c\}$, and $\pi^*(s) = a, \forall s \in \mathcal{S}$. Consider the full demonstration set $D = \{(s,a),(t,a),(u,a)\}$ that induce $\Psi(D) = \{\psi_{sab}, \psi_{sac}, \psi_{tab}, \psi_{tac}, \psi_{uab}, \psi_{uac}\}$ as indicated by dots in Figure 2a. The primal cone $\mathrm{cone}(\Psi(D))$ is shown in blue, and the version space $\mathcal{V}(D; \mathbb{R}^2)$ is in green. We note that the primal cone is supported by two extreme rays.*

The subset $D^\flat$ is not a valid/feasible teaching set as its version space $\mathcal{V}(D^\flat)$ (shown in green in Figure 2b) is wider than $\mathcal{V}(D)$ and contains some w's that do not induce π^* in all states, thus violating the feasibility condition in equation 3. On the other hand, both D' and D'' induce the correct version space $\mathcal{V}(D_\mathcal{S})$ (as shown in green in Figures 2c and 2d) on the learner and succeeds in teaching π^* to it. Furthermore, D'' is the smallest optimal set among them to do so. The problem become challenging as we move to higher dimensions where we can have large number of extreme rays as shown in Figure 3b.

3 Teaching Algorithm and Analysis

We first describe a naive teaching algorithm that frames optimal teaching as an infinite covering problem in the hypothesis/weight space and highlight the difficulty to solve it using the simple



(a) (b)

Figure 3: a.) Optimal teaching set D of (biased) consistent learner like SVM induce a larger space of weights w some of which(shown in light red color regions) are inconsistent wrt π^* and so they cannot succeed in teaching LVS learner. b.) Optimal teaching example in higher dimension $d \geq 3$ can have a large number of extreme rays to be covered using a subset of states making it a hard problem 7.

inconsistent hypothesis elimination method studied in prior works [16]. We then characterize the optimal teaching goal in terms of covering extreme rays of target version space and devise an optimal teaching algorithm called 'TIE' based on this insight.

3.1 Optimal Teaching as an Infinite Set Cover Problem in w Space

Demonstrating $(s, \pi^*(s))$ at a state s induces $A-1$ inequalities: $w^\top \phi(s, \pi^*(s)) > w^\top \phi(s, b), \forall b \neq \pi^*(s)$ on the learner. Each such inequality, rewritten as $w^\top \psi_{s\pi^*(s)b} > 0$, eliminates a halfspace $W_{sb} := \{w : w^\top \psi_{s\pi^*(s)b} \leq 0\}$. Therefore, the effect of demonstrating $(s, \pi^*(s))$ is to eliminate the set of weights $W_s := \cup_{b \neq \pi^*(s)} W_{sb}$. The full demonstration set $D_\mathcal{S} = \cup_{s \in \mathcal{S}} \{(s, \pi^*(s))\}$ over all states eliminates the union $\cup_{s \in \mathcal{S}} W_s$, such that only the version space $\mathcal{V}(D_\mathcal{S})$ survives. We are interested in finding the smallest demonstration set that also produces $\mathcal{V}(D_\mathcal{S})$ which is equivalent to covering the infinite collection inconsistent weight space $\mathcal{V}(D_\mathcal{S})^C$ by a smallest finite collection of infinite subsets W_s. This is a set cover problem:

$$\min_{T \subseteq \mathcal{S}} |T| \quad \text{s.t.} \quad \mathcal{V}(D_\mathcal{S})^C = \cup_{t \in T} W_t.$$

It is not immediately clear how to solve this infinite set cover problem. In the next section, we characterize the version space cone $\mathcal{V}(D_\mathcal{S})$ in terms of extreme rays and show that the problem is equivalent to covering the extreme rays of feature difference $\mathrm{cone}(\Psi(D_\mathcal{S}))$.

3.2 Teaching as a Finite Set Cover Problem in Extreme Rays of $\mathrm{cone}(\Psi(D_\mathcal{S}))$

In this section, we show that the infinite set cover problem for the teacher can be simplified to covering only the extreme rays of $\mathrm{cone}(\Psi(D_\mathcal{S}))$ using the feature difference vector induced by a subset of states $T \subseteq \mathcal{S}$. Before doing that, we introduce some definitions below.

Definition 1 (Extreme Ray and its Cover)**.** *An ray induced by a vector $v \in \mathbb{R}^d \backslash \{0\}$ is the set $\mathcal{R} = \{cv : c > 0\}$. A ray $\mathcal{R}$ is called an extreme ray of a cone $K \subseteq \mathbb{R}^d$ if for any $x, y \in K$, $x + y \in \mathcal{R} \implies x, y \in \mathcal{R}$. We say that a state $s \in \mathcal{S}$ covers a ray $\mathcal{R}$ if $\exists b \neq \pi^*(s) : \psi_{s\pi^*(s)b} \in \mathcal{R}$. Similarly, $T \subseteq \mathcal{S}$ is said to cover $\mathcal{R}$ if $\exists s \in T$ that covers $\mathcal{R}$.*

We first show that it is not necessary to teach with the full demonstration set $D_\mathcal{S}$ or, equivalently, induce the full set of induced difference vectors $\Psi(D_\mathcal{S})$. Instead, we just need to induce subset $U \subseteq \Psi(D_\mathcal{S})$ that covers the extreme rays of $\mathrm{cone}(\Psi(D_\mathcal{S}))$ as shown by Lemma 2.

Lemma 2 (Necessary and Sufficient Condition for Teaching)**.** *A subset of states $T \subseteq \mathcal{S}$ is a valid teaching set if and only if it can induce a set of all extreme rays of target $\mathrm{cone}(\Psi(D_\mathcal{S}))$.*

The proof can be found in the appendix. To construct an optimal teaching set, the teacher first needs to find all extreme rays and then find smallest subset of states $T \subseteq \mathcal{S}$ that cover all those extreme rays. This is our new set cover problem with the universe as a set of all extreme rays Ψ^* of $\mathrm{cone}(\Psi(D_\mathcal{S}))$ defined by $U = \Psi^*$. Note that each state s covers a subset of extreme rays $V_s \subseteq \Psi^*$ by its feature difference vectors and the teacher wants to choose the smallest subset of states to cover U. Unlike the infinite set cover problem (3.1), this is a standard finite set cover problem.

3.3 The Optimal Teaching Algorithm and Analysis

Algorithm 1 Teach using Iterative Elimination (TIE)

def MinimalExtreme($\mathcal{X}$)**:**

for each $x_j \in \mathcal{X}$ **do**
  Solve LP($x_j, \mathcal{X}/\{x\}$) defined by 3
  if $v_j > 0$ **then**
    $\mathcal{X} \leftarrow \mathcal{X} \backslash x_j$ ▷ eliminate x_j if not necessary
return $\mathcal{X}$ ▷ extreme vectors

def OptimalTeach($\mathcal{S}, \mathcal{A}, \pi^*, \phi$)**:**

let $\Psi(D_{\mathcal{S}}) = \{\psi_{s\pi^*(s)b} \in \mathbb{R}^d : s \in \mathcal{S}, b \in \mathcal{A}, b \neq \pi^*(s)\}$ ▷ compute feature differences
$\Psi^* \leftarrow$ **MinimalExtreme**($\Psi(D_{\mathcal{S}})$)
for $s \in \mathcal{S}$ **do**
  $V_s \leftarrow \left\{\psi \in \Psi^* : \exists \psi_{s\pi^*(s)b} \in \Psi(D_{\mathcal{S}}), \hat{\psi}_{s\pi^*(s)b} = \hat{\psi}\right\}$ ▷ extreme rays covered by s
$\{V_s : s \in T^* \subseteq \mathcal{S}\} \leftarrow$ SetCover($\Psi^*, \{V_s\}|_{s \in \mathcal{S}}$) ▷ T^* is smallest cover of all extreme rays
teach $D^* = \{(t, \pi^*(t)) : t \in T^*\}$ to the agent ▷ D^* is the minimum demonstration set

Lemma 3 shows how to compute extreme rays of $\text{cone}(\mathcal{X})$ formed by a finite set $\mathcal{X}$ which when applied to $\mathcal{X} = \Psi(D_{\mathcal{S}})$ allows the teacher to find Ψ^*. The proof can be found in the appendix.

Lemma 3 (Extreme Ray Test)**.** *Given a set of vectors $\mathcal{X} \in \mathbb{R}^d$ and $x \in \mathcal{X}$, the following linear program determines if x is the only vector in $\mathcal{X}$ that lies on an extreme ray of* $\text{cone}(\mathcal{X})$*. The objective value of LP$(x, \mathcal{X})$ is $-\infty$ if $x \notin \text{cone}(\mathcal{X} \backslash \{x\})$, and strictly positive otherwise.*

$$\textit{LP}(x, \mathcal{X}) : \qquad \min_w \quad \langle w, x \rangle \textit{ s.t. } \langle w, x' \rangle \geq 1 \quad \forall x' \in \mathcal{X} \backslash \{x\}.$$

Note that to finding all the extreme rays is equivalent to finding one representative vector for each extreme ray. If LP$(x, \mathcal{X}) = -\infty$, x is the only vector on one of the extreme ray of $\text{cone}(\mathcal{X})$ and we should keep x. If LP$(x, \mathcal{X}) > 0$, then x is not a unique representative of some extreme ray of $\text{cone}(\mathcal{X})$ in set $\mathcal{X}$, in which case, we can remove x. Employing this test iteratively to $\mathcal{X} = \Psi(D_{\mathcal{S}})$ provides unique representative for all the extreme rays of $\text{cone}(\Psi(D_{\mathcal{S}}))$. When this process terminates, the surviving set $\Psi^* \subseteq \Psi(D_{\mathcal{S}})$ contains exactly one vector on each extreme ray of $\text{cone}(\Psi(D_{\mathcal{S}}))$. Next, the teacher solves a finite set cover problem defined by instance $(U, \{V_s\}|_{s \in \mathcal{S}})$ to find minimum teaching set. We provide the complete teaching algorithm'TIE' in Algorithm 1. Our algorithm 'TIE' achieves the following guarantee on Teaching Dimension.

Theorem 4 (Optimal Teaching in Finite State Setting)**.** *Given an optimal teaching problem instance $(\mathcal{M}, \phi, \pi^*)$ 2, our teaching algorithm TIE 1 correctly finds the optimal teaching set D^* and achieves the teaching dimension $TD(\mathfrak{C})$.*

Our algorithm also works in an infinite state setting under mild assumption as stated by the next corollary. Proof for both Theorem 4 and Corollary 5 can be found in the appendix.

Corollary 5 (Optimal Teaching for Infinite State Setting)**.** *Assuming* $\text{cone}(\Psi(D_{\mathcal{S}}))$ *is a closed convex cone with finite extreme rays and the teacher knows the extreme rays to state mapping, our algorithm TIE 1 correctly finds the optimal teaching set D^*.*

Behavior Cloning learners suffer from the issue of distribution shift [29] which is undesirable. Next, we show that in the presence of our helpful teacher, this issue is not present. Furthermore, we could argue that since BC learners learn directly in policy space and do not require further planning like IRL learners[3, 8], they are the most canonical example of learners to teach using demonstration in MDP setting.

Corollary 6 (Optimal Value Guarantee)**.** *Under teaching by our algorithm TIE, the entire family of learners $\mathcal{L} \in \mathfrak{C}$ achieve zero value risk, i.e., $R_V(D^*; \mathcal{L}) = 0$.*

We note that for instances with $|\mathcal{A}| = 2$, the subsets V_s in the set cover problem contain only one element, and thus the corresponding set cover problem is computationally efficient to solve. However, solving the set cover problem for a general $\mathcal{A}$ is NP-hard problem as each state can cover as much as $|\mathcal{A}|-1$ extreme rays. We show that no teacher can avoid this hardness by giving a poly-time reduction from a finite set cover problem to optimal teaching problem; proof can be found in the appendix.

Theorem 7. *Finding an optimal teaching set for a linear version space BC learner is an NP-hard for instance with action space size $|\mathcal{A}| > 2$.*

Next, we instantiate the set cover problem by an efficient greedy procedure which leads to the following guarantee on optimal teaching by *'TIE'* with a computation complexity of $O((|\mathcal{S}||\mathcal{A}|)^3)$.

Corollary 8 (Approximately Optimal Teaching)**.** *Our algorithm TIE 1 can efficiently teach a family of consistent linear BC learners $\mathfrak{C}$ and it finds an approximate optimal teaching set $\tilde{D}$ such that $|\tilde{D}| \leq \log(|\mathcal{A}|-1)|D^*|$[34].*

4 Experiments

We evaluate our teaching algorithm *TIE* on three environments: 1) *Pick the Right Diamond*, 2) *Visual Programming in Maze with Repeat Loops* and 3) *Polygon Tower environment* (provided in the appendix). Through these experiments, we aim to demonstrate the following: a) Our algorithm *TIE* finds an optimal or a near-optimal teaching set in all these environments. b) The optimal teaching dataset so produced is competitive with a learner-specific optimal teaching set and can teach any consistent linear BC learners, and c) *TIE* performs significantly better than competitive baselines like Teach-Random and Teach-All that we define below.

Baselines: We consider two baselines. 1) Teach-All: This teacher simply teacher the target action in all states to the learner, 2) Teach-Random: This teacher draws states uniformly at random $s \sim U(\mathcal{S})$ and adds it to a collection until the collection becomes a valid teaching set, i.e., it induces the target cone $\mathcal{V}(D_{\mathcal{S}})$. We note that the teaching set produced by prior works [21, 25] are specialized to individual learners and do not yield a feasible set for teaching the entire family of consistent linear learners. Furthermore, their teacher directly constructs covariate vectors (features) in $\mathbb{R}^d$ and is not able to choose individual states, thus, not directly applicable to our setting.

4.1 Pick the Right Diamond

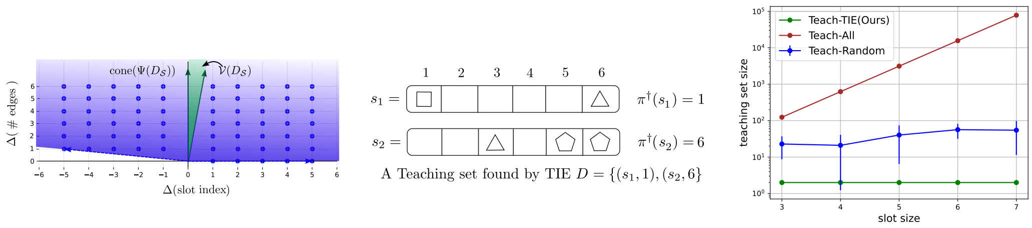
Recall the game from Example 1. A state in $\mathcal{S} = \{\hexagon, \pentagon, \square, \triangle, o\}^n/\{o\}^n$ consists of a n dimensional board with one of four types of diamond or be empty(o). Each action in action space $\mathcal{A} = [n]$ represents picking an object in one of the cells. The complete description of the MDP environment can be found in the appendix.

Feature representation & optimal policy: The learner uses a natural feature function in $\mathbb{R}^2$ given as follows, $\phi(s, a) = [a, \#\text{edges of diamond at } a]$, where $[\#\text{edges of diamond at } a]$ is 0 if the slot is empty. The optimal policy is to collect the diamonds in order of decreasing value i.e. from a large to a small number of edges. In case of ties, the learner should choose the rightmost diamond. This policy is feasible under the above featurization, for example, $w^* = [1, 10]$ uniquely induces π^*. For a board of size $n = 6$, there are a total of $5^6 - 1$ states and their feature difference vectors $\Psi(D_{\mathcal{S}})$ are shown as blue dots in Figure 4a. The primal cone $\text{cone}(\Psi(D_{\mathcal{S}}))$ is the blue-shaded area. It contains two extreme rays, both need to be covered for successful teaching. The version space is denoted in green.

Optimal teaching set: We note that any teaching set that covers the two extreme rays is a valid teaching set. When run on a board instance of size $n = 6$, our algorithm *TIE* produces a teaching set of size two as illustrated in Figure 4b. This is an instance optimal teaching set and shows a dramatic improvement over teaching all $5^6 - 1$ states. We perform experiments on boards of different sizes and found that *TIE* significantly outperforms other two baselines as shown in Figure 4c.

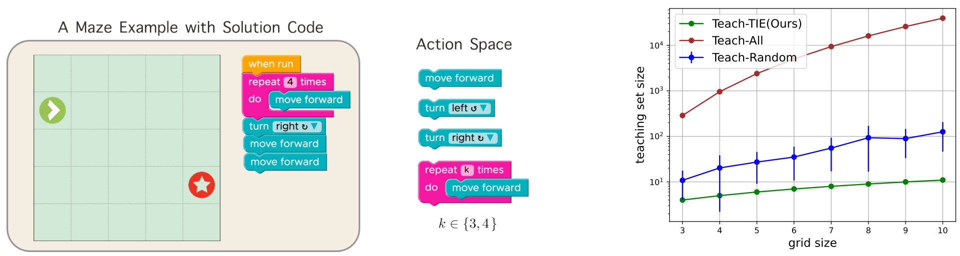
4.2 Visual Block Programming in Maze with Repeat Loop

We consider a real-world visual programming platform used for teaching kids/learners to write code to complete visual task in a maze environment [5, 9, 11, 15]. Further, we choose a domain that aims to teach learners to use repeat code blocks to write succint code to complete a navigation-based task



(a) (b) (c)

Figure 4: Optimal teaching in "Pick the right diamond" with $n = 6$ slots. a) Feature difference vectors $\Psi(D_{\mathcal{S}})$ induced by target policy is shown as blue dots, primal cone $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$ as blue area, and dual version space $\mathcal{V}(D_{\mathcal{S}})$ as green area. b) A teaching set produced by *TIE* on board of size 6. c) Comparison of our *TIE* algorithm with other baselines.



(a) (b)

Figure 5: a) An example of programming task in 5×5 with solution code. The maze contains a turtle facing one of four directions (shown by green arrow) and a goal cell (shown by red star). The optimal (smallest) solution code to lead the turtle to the goal is shown on the side. The action apace consisting of 5 basic code block is shown on the right. b) Performance of *TIE* compared to baselines on this domain with different maze sizes.

in maze environments of different sizes. The environment state consists of a $n \times n$ maze with a turtle (shown in green in Figure 5a) facing one of four directions, a goal cell (shown by a red star), and a (partial) piece of code that can be executed to move the turtle in the maze. The learners objective is to assemble multiple code blocks in sequence to write a piece of code that can solve a given maze task.

The action space $\mathcal{A}$ consists of n actions (each representing a basic code block) available to the learner to write code and is given as follows: *Turn-Left (TL)*: turns turtle to its left, *Turn-Right (TR)*: turns turtle to its right, *Move-Forward (MV)*: moves turtle forward by one cell, *Repeat-k-Times-Move (R_k-MV)* is a complex block with repeat loop that moves the turtle forward by k cells in a single command where $k \in \{3, \cdots, n-1\}$ The task is to teach the agent to write most succint piece of code that can be executed to make the turtle reach the goal cell. This is captured by a reward function that gives a reward of -1 to the first three code blocks (*TL/TR/MV*) and a reward of -2 to repeat blocks R_k*-MV*. [1] The complete description of the MDP defining this problem can be found in the appendix.

Feature representation & optimal policy: We consider an execution-guided feature representation[11] that takes an initial board with a partial piece of code and constructs a feature vector by first executing the partial code to get an intermediate state and extracting features from that state. We use a natural feature representation $\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^d$ that encodes the relative orientation and distance of the goal cell from the turtle cell; refer to the appendix for more details. The optimal policy is realizable by a linear policy under this representation. The teacher knows ϕ and can construct a dataset D of (state and optimal action) tuples and provide it to the learners. Its goal is to teach the target optimal policy of writing a succinct code to the entire family of learners $\mathfrak{C}$.

[1] We note that repeats are complex code blocks that have two components and should be used only when it provide an advantage, i.e., it can substitute more than two basic blocks.

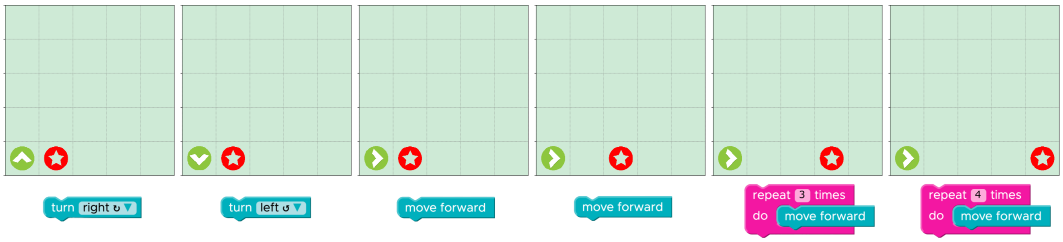


Figure 6: Optimal Teaching Set produced by *TIE* on a goal-reaching coding task with 5×5 maze. The demonstration consists of states with an initial board without any partial code. The optimal action demonstrated to the learner is shown below each state.

Optimal teaching set: We run our algorithm *TIE* on environments with different sizes of maze and observe that it is able to find an optimal teaching set for each of the environments; refer to Figure 6 for an example on 5×5 maze. This optimal teaching set demonstrates each action exactly once on a suitable maze state where that action is an optimal one. Our algorithm performs significantly better than the other two baselines: *Teach-Random* and *Teach-All* when run on maze of different sizes as shown by Figure 5b. We also trained other candidate consistent learners like linear SVM, linear perception, and linear logistic regression on teaching set obtained by *TIE* and verified that all of them achieve a risk of zero as claimed by our Theorem 4.

5 Related Work

Several prior works have studied optimal teaching of version space learners but mostly in finite or countable infinite version space settings[16, 18]. Some works like [36] has studied teaching mulitple learners simultaneously but in unsupervised learning setting of teaching mean. Instead, we study teaching a family of consistent BC learners in linear hypothesis space setting.

Comparatively, studies on optimal teaching of different linear learners are highly relevant to our work. For example, [21, 25] examined teaching linear learners like like SVM, perceptron and logistic regression which can be seen as individual instances of consistent linear BC learners. These works focus on teaching individual learners where teacher could exploit strong biases of these learners to arguably teach them relatively easily. On the other hand, we aim to teach entire family of consistent linear BC learners where the teacher cannot base its teaching on bias of individual learners. Additionally, [22] delved into the optimal teaching of iterative learners like gradient descent learners which have also been shown to be biased learners [32]. Further, [19, 28] explored the teaching dimension of kernel learners for teaching a non-linear boundary in $\mathbb{R}^{\mathcal{K}}$ space. Furthermore, these studies typically assume a more powerful teacher capable of constructing arbitrary covariate and label pairs, whereas we restrict the teacher to select states from a fixed state space and teach the learner to generalize to other states using feature covariates induced by the feature function.

Another significant line of research involves Teaching by Demonstration in a RL setting. Relevant studies by [8, 10] have focused on teaching linear IRL learners[2, 24] which are different kind of imitation learners that learn using a linear reward model and require planning access to environment to eventually learn an optimal policy. Unlike them, our linear BC learners learn directly in the policy space by only using teaching demonstration and are not limited by access to MDP environment.

6 Discussion & Conclusion

We studied the problem of optimal teaching of a family of linear learners in a behavior cloning setting. We provided an efficient algorithm that achieved an approximately optimal guarantee of $\log(|\mathcal{A}| - 1)$ on the teaching dimension. Our work focused mainly on teaching linear learners and we hope future works would extend this *teaching a family* setting to more complex non-linear learners like neural networks. Another interesting future direction would be to study optimal teaching for family of linear learners under budget constraints. It would be also interesting to apply our method to more complex real-world settings like teaching a class of kids to learn to code.

References

[1] *High Performance Outdoor Navigation from Overhead Data using Imitation Learning*, pages 262–269. 2009.

[2] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[3] Stephen Adams, Tyler Cody, and Peter A Beling. A survey of inverse reinforcement learning. *Artificial Intelligence Review*, 55(6):4307–4346, 2022.

[4] Alekh Agarwal, Nan Jiang, Sham M. Kakade, and Wen Sun. *Reinforcement Learning: Theory and Algorithms*. 2021. `https://rltheorybook.github.io/`.

[5] Umair Ahmed, Maria Christakis, Aleksandr Efremov, Nigel Fernandez, Ahana Ghosh, Abhik Roychoudhury, and Adish Singla. Synthesizing tasks for block-based programming. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22349–22360. Curran Associates, Inc., 2020.

[6] Martin Anthony, Graham Brightwell, and John Shawe-Taylor. On specifying boolean functions by labelled examples. *Discrete Applied Mathematics*, 61(1):1–25, 1995.

[7] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[8] Daniel S Brown and Scott Niekum. Machine teaching for inverse reinforcement learning: Algorithms and applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7749–7758, 2019.

[9] Rudy Bunel, Matthew J. Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. Leveraging grammar and reinforcement learning for neural program synthesis. *CoRR*, abs/1805.04276, 2018.

[10] Maya Cakmak and Manuel Lopes. Algorithmic and human teaching of sequential decision tasks. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[11] Xinyun Chen, Chang Liu, and Dawn Song. Execution-guided neural program synthesis. In *International Conference on Learning Representations*, 2019.

[12] 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. *Advances in Neural Information Processing Systems*, 31, 2018.

[13] Felipe Codevilla, Eder Santana, Antonio M. Lopez, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[15] Aleksandr Efremov, Ahana Ghosh, and Adish Kumar Singla. Zero-shot learning of hint policy via reinforcement learning and program synthesis. In *Educational Data Mining*, 2020.

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

[17] Sally A Goldman and H David Mathias. Teaching a smart learner. In *Proceedings of the sixth annual conference on computational learning theory*, pages 67–76, 1993.

[18] Hayato Kobayashi and Ayumi Shinohara. Complexity of teaching by a restricted number of examples. 01 2009.

[19] Akash Kumar, Hanqi Zhang, Adish Singla, and Yuxin Chen. The teaching dimension of kernel perceptron. In *International Conference on Artificial Intelligence and Statistics*, pages 2071–2079. PMLR, 2021.

[20] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10(6):799–822, 1994.

[21] Ji Liu, Xiaojin Zhu, and Hrag Ohannessian. The teaching dimension of linear learners. In *International Conference on Machine Learning*, pages 117–126. PMLR, 2016.

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

[23] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[24] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[25] Xuezhou Zhang Hrag Gorune Ohannessian, Ayon Sen, Scott Alfeld, and Xiaojin Zhu. Optimal teaching for online perceptrons. *University of Wisconsin-Madison*.

[26] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

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

[28] Hong Qian, Xu-Hui Liu, Chen-Xi Su, Aimin Zhou, and Yang Yu. The teaching dimension of regularized kernel learners. In *International Conference on Machine Learning*, pages 17984–18002. PMLR, 2022.

[29] Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[30] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. 02 1992.

[31] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014.

[32] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

[33] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *CoRR*, abs/1905.13566, 2019.

[34] Valika K. Wan and Khanh Do Ba. Approximating set cover, 2005. Accessed: 2023-10-15.

[35] Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[36] Xiaojin Zhu, Ji Liu, and Manuel Lopes. No learner left behind: On the complexity of teaching multiple learners simultaneously. pages 3588–3594, 08 2017.

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

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract clearly reflects the main contribution of studying optimal teaching problem for family of linear BC learners and providing an (approximately) optimal algorithm for teaching the family. Our experiments support highlight the effectiveness of our teaching algorithm over other baselines.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in last section of the paper.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide complete theorem with assumptions in main paper and proofs in appendix.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the experimental setup in the paper with additional details provided in appendix. We also provide code to replicate our results in supplementary material.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide code for constructing simulation environment and our algorithms in supplementray material.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the relevant details to reproduce the experimental results is provided in appendix and supplementary materials.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For experiments that involved randomness like our Teach-Random algorithm, we showed error bars in the plot.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We performed our experiments on a Apple Macbook M1 laptop with 16GB and our code can be run on any standard machine.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our algorithm is intended to help learners to speed up learning and we do not envision safety or privacy concerns with our work.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide details about our code and how to run it in the readme file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

7 Appendix

Given a finite set of vectors $\mathcal{X} = \{x_i \in \mathbb{R}^d : i \in [n]\}$, we define the primal cone generated by this set as

$$\text{cone}(\mathcal{X}) = \left\{ \sum_{i \in \mathcal{X}} \lambda_i x_i, \;\; \lambda_i \geq 0, \;\; \forall i \in \mathcal{X} \right\}. \tag{4}$$

Given any set U, we define the dual cone as

$$\text{cone}^*(U) := \{y \,|\, y^T x > 0, \; \forall x \in U, x \neq 0\}. \tag{5}$$

In particular, if the finite set $\mathcal{X}$ has $x_i \neq 0$ for all $i \in [n]$, we have

$$\text{cone}^*(\mathcal{X}) := \{y \,|\, y^T x_i > 0, \; i = 1, 2, \ldots, n\}. \tag{6}$$

We prove some basic properties about cones in $\mathbb{R}^d$.

Proposition 9. *For any finite sets U, V s.t. $U \subseteq V \subset \mathbb{R}^d$, we have that,*

1. $\text{cone}^*(U) = \text{cone}^*(\text{cone}(U))$.

2. $\text{cone}^*(U) \supseteq \text{cone}^*(V)$.

3. $\text{cone}(U) = \text{cone}(V) \implies \text{cone}^*(U) = \text{cone}^*(V)$.

Proof. 1 For any $w \in \text{cone}^*(U), \langle w, u_i \rangle > 0, \forall u_i \in U \implies \forall i, \lambda_i \geq 0, \sum_i \lambda_i u_i \neq 0, \langle w, \sum_i \lambda_i u_i \rangle > 0 \implies w \in \text{cone}^*(\text{cone}(U))$. For the opposite direction, let $\forall \lambda_i \geq 0, \sum_i \lambda_i u_i \neq 0, \langle w, \sum_i \lambda_i u_i \rangle > 0$. For a fixed i, choose $\lambda_i = 1$ and $\lambda_j = 0, \forall j \neq i$. Then, we have $\langle w, u_i \rangle > 0, \forall u_i \in U$, thus, $w \in \text{cone}^*(U)$.

2 Now, for second part of the proposition, let $x \in \text{cone}^*(V)$ i.e. $\langle x, v \rangle > 0, \forall v \in V$. Since, $U \subseteq V$, this implies $\langle w, u_i \rangle > 0, \forall u_i \in U$. Thus, $x \in \text{cone}^*(U)$, thus proving the statement.

3 Finally, for the third part, we have that $\text{cone}^*(U) = \text{cone}^*(\text{cone}(U)) = \text{cone}^*(\text{cone}(V)) = \text{cone}^*(V)$, where the first and third equality follows from part 1 of this proposition and second equality follows from the premise. $\square$

7.1 Finding extreme rays of primal cone

In the remainder, we assume that the finite set $\mathcal{X} = \{x_i \in \mathbb{R}^d : i \in [n]\}$ contains all nonzero vectors, and recall the definitions of cone (4) and dual cone (6). Our problem is to find a set $\mathcal{X}^* \subset \mathcal{X}$ of minimum cardinality such that $\text{cone}^*(\mathcal{X}^*) = \text{cone}^*(\mathcal{X})$.

Note that by realizability, we have that $\text{cone}^*(\mathcal{X})$ is nonempty. We can define $\text{cone}^*(\mathcal{X})$ alternatively as follows

$$\begin{aligned} \text{cone}^*(\mathcal{X}) &:= \{\alpha z \,|\, \alpha > 0, \; z \in P(\mathcal{X})\} \\ \text{where} \;\; P(\mathcal{X}) &:= \{z \,|\, z^T x_i \geq 1, \; i \in \mathcal{X}\}. \end{aligned} \tag{7}$$

Proof. Any z satisfying (7) clearly has $z^T x_i > 0$ for all $i \in \mathcal{X}$, so $z \in \text{cone}^*(\mathcal{X})$. Conversely, given any y with $y^T x_i > 0$ for all $i \in \mathcal{X}$, we set $\alpha = \min_{i \in \mathcal{X}} y^T x_i > 0$ and $z = y/\alpha$ to get α and z satisfying (7). $\square$

The key element of the algorithm is an LP of the following form, for some $x_j \in \mathcal{X}$:

$$\begin{aligned} \text{LP}(x_j, \mathcal{X}/\{x_j\}) : \qquad & \min_w \; w^T x_j \\ & \text{subject to } w^T x_i \geq 1 \; \forall i \in \mathcal{X}/\{x_j\}. \end{aligned} \tag{8}$$

Note that this problem can be written alternatively, using the notation of (7), as

$$\min_w \; w^T x_j \text{ subject to } w \in P(\mathcal{X}/\{x_j\}). \tag{9}$$

The dual of (8) will also be useful in motivating and understanding the approach:

$$\begin{aligned} &\text{LP-Dual}(x_j, \mathcal{X}/\{x_j\}) : \quad \max_{\lambda_i, i \in \mathcal{X}/\{x_j\}} \sum_{i \in \mathcal{X}/\{x_j\}} \lambda_i \\ &\text{s.t.} \sum_{i \in \mathcal{X}/\{x_j\}} \lambda_i x_i = x_j, \quad \lambda_i \geq 0 \text{ for all } i \in \mathcal{X}/\{x_j\}. \end{aligned} \tag{10}$$

We prove a lemma with several observations.

Lemma 10 (Proof of Lemma 3). *Suppose that $\mathcal{X}$ is not empty. Let $x_j \in \mathcal{X}$. We have the following.*

(i) When (8) is unbounded, (10) is infeasible, so $x_j \notin \text{cone}(\mathcal{X}/\{x_j\})$. Furthermore, $\exists w \in \mathbb{R}^d$ s.t. $w \in \text{cone}^*(\mathcal{X}/\{x_j\})$ but $w \notin \text{cone}^*(\mathcal{X})$.

(ii) if (8) has a solution, the optimal objective value must be positive.

(iii) When (8) has a solution with a positive optimal objective, then $x_j \in \text{cone}(\mathcal{X}/\{x_j\})$.

Proof. (i) From LP duality, when (8) is unbounded, then (10) is infeasible, giving the first part of the result. For the second part, we note by the feasibility condition of 8 that the optimal solution $w^* \in \text{cone}^*(\mathcal{X}/\{x_j\})$ but since ${w^*}^T x_j < 0$, that is, $w^* \notin \text{cone}^*(\mathcal{X})$, giving us the result.

(ii) If (8) were to have a solution with optimal objective 0, then by LP duality, the optimal objective of (10) would also be zero, so the only possible value for λ is $\lambda_i = 0$ for all $i \in \mathcal{X}/\{x_j\}$. The constraint of (10) then implies that $x_j = 0$, which cannot be the case, since we assume that all vectors in $\mathcal{X}$ are nonzero.

(8) cannot have a solution with *negative* optimal objective value, because by LP duality, (10) would also have a solution with negative objective value. However, the value of the objective for (10) is non-negative at all feasible points, so this cannot happen.

(iii) When (8) has a solution with positive optimal objective, then LP duality implies that (10) has a solution with the same objective. Thus, there are nonnegative λ_i, $i \in \mathcal{X}/\{x_j\}$, not all 0, such that the constraint in (10) is satisfied, giving the result.

□

Lemma 11. *Let U and V be finite sets with $U \subseteq V \subseteq \mathbb{R}^d$ and $\text{cone}^*(V)$ is non-empty. Then $\text{cone}(U) = \text{cone}(V)$ and $\text{cone}^*(U) = \text{cone}^*(V)$ if and only if U contains at least one vector on each of the extreme rays of $\text{cone}(V)$.*

Proof. For the sufficiency direction, we note that for a set $U \subseteq V$, if U contains at least one vector on each of the extreme rays of $\text{cone}(V)$ then $\text{cone}(U) = \text{cone}(V)$ (since all the vectors in a cone can be expressed as a conic combination of extreme vectors of the cone). Furthermore, by Proposition 3, we have $\text{cone}^*(U) = \text{cone}^*(V)$.

For necessity direction, suppose that U does not contain any vector on a certain extreme ray $\mathcal{R}$ of V. Then $U \subseteq V/\mathcal{R}$. Thus $\text{cone}(U) \subseteq \text{cone}(V/\mathcal{R}) \subsetneq \text{cone}(V)$ and thus $\text{cone}^*(U) \supseteq \text{cone}^*(V/\mathcal{R})$. Let $r \in \mathcal{R} \subset V$. Then LP-Dual$(r, U)$ will be infeasible, so LP(r, U) is either infeasible or unbounded. But LP(r, U) is feasible because pointedness of $\text{cone}(V)$ means that $\text{cone}^*(V)$ is nonempty, so $\text{cone}^*(U) \supset \text{cone}^*(V)$ is also nonempty and so the constraints of LP(r, U) are guaranteed to hold for some w. We conclude that LP(r, U) is unbounded, so there is a direction w such that $w^T x > 0$ for all $x \in U$ but $w^T r < 0$. This vector w belongs to $\text{cone}^*(U)$ but not to $\text{cone}^*(V)$, so $\text{cone}^*(U) \subsetneq \text{cone}^*(V)$, as required. □

Lemma 12 (Proof of Lemma 2). *For successful teaching using $T \subseteq \mathcal{S}$, the teacher needs to cover each extreme ray of $\text{cone}(\Psi(D_\mathcal{S}))$ using the feature difference vectors induced by teaching π^* on T.*

Proof Sketch. Teaching is successful if the learner can recover a non-empty subset of target consistent version space i.e. for a teaching subset of states $T \subseteq \mathcal{S}, \mathcal{V}(D_T; \mathcal{H}) = \mathcal{V}(D_\mathcal{S}; \mathcal{H})$. We also have that $\mathcal{V}(D_T; \mathcal{H}) \supseteq \mathcal{V}(D_\mathcal{S}; \mathcal{H})$ by Definition 1 and hence for successful teaching, the teacher has to induce complete $\mathcal{V}(D_{\mathcal{S}}; \mathcal{H})$ on the learner. From Lemma 11, teaching is successful i.e. the learner recovers the complete $\mathcal{V}(D_{\mathcal{S}}; \mathcal{H}) = \mathrm{cone}^*(\Psi(D_{\mathcal{S}}))$ if and only if the teacher is able to cover(induce at least one vector on) each extreme ray of $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$. $\square$

Theorem 13 (Proof of Theorem 4)**.** *Given an optimal teaching problem instance $(\mathcal{M}, \phi, \pi^*)$ 2, our teaching algorithm TIE 1 correctly finds the optimal teaching set D^* and achieves the teaching dimension $TD(\mathfrak{C})$.*

Proof. Lemma 12 tells us that for valid teaching, the teacher must induce at least one feature difference vector on each of the extreme rays of $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$. Since $\mathcal{S}, \mathcal{A}$ is finite, there are only a finite number of extreme rays possible. The iterative elimination procedure in **MinimalExtreme** in Algorithm 1 first finds unique representatives for each extreme ray of $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$. This follows from lemma 10. Let $\mathcal{X}$ be the surviving set of vectors at the start of an iteration where x_j is considered. We have that if $x_j \in \mathrm{cone}(\mathcal{X}/\{x\})$ it will get eliminated by the extreme ray test 3 and on the other hand if x_j is unique representative for an extreme ray in $\mathcal{X}$, we have $x_j \notin \mathrm{cone}(\mathcal{X}/\{x\})$ and thus x_j will not get eliminated. At every iteration, we either eliminate a vector in $\Psi(D_{\mathcal{S}})$ or that vector is a unique representative for an extreme ray of $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$ and cannot be eliminated. Thus, at the end of the iterative elimination procedure, we recover a set of unique representative vectors Ψ^* for each extreme ray of $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$.

The next step involves finding a smallest subset of states $T \subseteq \mathcal{S}$ that can cover all the extreme rays. This is done by a set cover problem defined on lines 4-7 of the **OptimalTeach** procedure in Algorithm 1. The set of unique representatives of extreme rays forms the universe to be covered and each state defines a subset of representatives for extreme rays that it can cover. The minimum number of subsets that can cover the entire universe is the minimum number of states that covers all the extreme rays giving us $T^* \subseteq \mathcal{S}$ as an optimal solution for teaching problem. For instance with $|\mathcal{A}| = 2$, every state can induce at most one extreme ray so picking one state for each extreme rays gives the optimal teaching set. $\square$

Theorem 14 (Proof of Corollary 5)**.** *Assuming $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$ is a closed convex cone with finite extreme rays and the teacher knows the feature representation mapping, our algorithm TIE 1 correctly finds the optimal teaching set D^*.*

Proof. Since the convex cone $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$ is closed, we know that extreme rays must be contained in it. Furthermore, an extreme ray must be induced by one of the states. The teacher knows the states that induce each extreme ray or a subset of it and can construct set cover problem to solve the optimal teaching problem. $\square$

Theorem 15 (Proof of NP-Hardness in Theorem 7)**.** *Finding an optimal teaching set for a linear version space BC learner is a NP-hard for instance with action space size $|\mathcal{A}| > 2$.*

Proof. We provide a poly-time reduction from the set cover problem to the optimally teaching version space BC learner problem 3. Since the set cover is an NP-hard problem, this implies that optimal teaching is NP-hard to solve as well. Let $P = (U, \{V_i\}_{i \in [n]})$ be an instance of set cover problem where U is the universe and $\{V_i\}_{i \in [n]}$ is a collection of subsets of U. We transform P into an instance of optimal teaching problem $Q = (\mathcal{M}, \phi, \pi^{)}$.

Construction: For each subset V_i of P, we create a state s_i of Q. For each element k in the universe U of P, we create an extreme ray vector ψ_k of feature difference vectors in Q. The complete construction is given as follows :

1. $\mathcal{S} = [n], \mathcal{A} = [A]$ where $A = \max_{i \in [n]} |V_i| + 1$.
2. The target policy is $\pi^*(s) = A$, $\forall s \in \mathcal{S}$.
3. $\Psi = \{\psi_k = (\cos(\frac{2\pi k}{n}), \sin(\frac{2\pi k}{n}), 10) : k \in [|U|]\}$.
4. for each $s \in \mathcal{S}$ we construct feature vectors $\{\phi(s, a) : a \in \mathcal{A}\}$ such that the feature differences map to extreme rays ψ's. Enumerating over element of $V_s := \{V_{s1}, \cdots, V_{s|V_s|}\}$,

we define the induced feature difference vectors as,

$$\psi_{sAb} = \psi_{V_{sb}}, \quad \forall b < |V_s| - 1 \tag{11}$$
$$\psi_{sAb} = \psi_{V_{s|V_s|}}, \forall |V_s| - 1 \leq b \leq A - 1 \tag{12}$$

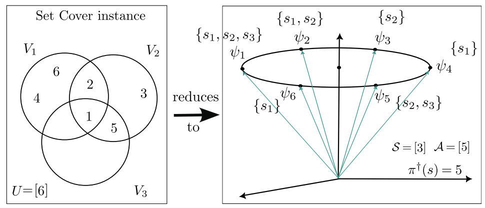


Figure 7: An reduction example from a set cover problem to optimal teaching LBC problem

Claim 16. *A solution of optimal teaching LBC instance $(\mathcal{S}, \mathcal{A}, \pi^*, \phi)$ gives a solution to the set cover problem $(U, \{V_i\}_{i\in[n]})$ and vice versa.*

Finding a collection of subsets $\{V_i\}_{i\in[n]}$ of smallest size that covers all elements in universe U is equivalent to selecting a subset of states $\mathcal{S}$ of smallest size that covers all the extreme rays defined by Ψ.

For a solution $\{V_j\}_{j\in T^*}$ s.t. $T^* \subseteq [n]$ to the set cover instance $(U, \{V_i\}_{i\in[n]})$, the set of states indexed by $T^* \subseteq \mathcal{S}$ is a solution to the optimal teaching instance $(\mathcal{S}, \mathcal{A}, \pi^*, \phi)$ and vice versa. The argument follows from a direct translation between two instances. See Figure 7. ☐

8 More Experimental Results

8.1 Polygon Tower

Let the state space be $\mathcal{S} = \{2, \ldots, n\}$, the action space be $\mathcal{A} = [n+1]$, the feature function be $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^3$ given by

$$\phi(s, a) = \begin{cases} [0, 0, s] & \text{if } a = n+1 \\ \left[-s \cdot \cos(\frac{2\pi a}{s}), -s \cdot \sin(\frac{2\pi a}{s}), 0\right] & \text{otherwise} \end{cases} \tag{13}$$

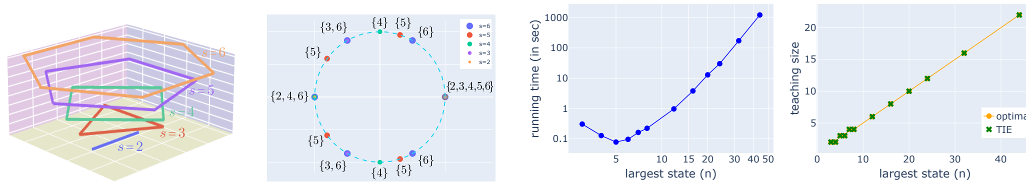
We note that for a fixed state s, the feature vectors for actions $1 \ldots n$ lie on a polygon of radius s centered around the origin on the xy plane.

Target Policy The teacher wants to teach the target policy $\pi^\dagger$ where $\forall s \in \mathcal{S}$, $\pi^\dagger(s) = n+1$. The policy is realizable: for example, $w = [0, 0, 1]$ induces this policy. The feature difference vectors induced by $\pi^\dagger$ on $\mathcal{S}$ is given as $\Psi(D_{\mathcal{S}}) = \{[s \cdot \cos(\frac{2\pi a}{s}), s \cdot \sin(\frac{2\pi a}{s}), s] : s \in \mathcal{S}, a \neq n+1\}$. These difference vectors lie on elevated polygons as shown in Figure 8(a). In particular, state s induces a s-gon of radius s centered at $(0, 0, s)$. Figure 8(b) shows the top view of the extreme rays of the primal cone $\mathrm{cone}(\Psi(D_{\mathcal{S}}))$. The extreme rays are shown as dots and the states that cover each extreme ray are labeled.

Optimal Teaching The polygon tower problem has an interesting structure that allows us to characterize the minimum demonstration set.

Proposition 17. *The optimal teaching set T^* of the polygon tower consists of all states in $\mathcal{S}$ that are not divisible by any other states in $\mathcal{S}$.*

Proof. For any pair of states s, s' such that $s' > s$ and $s' \bmod s = 0$, then s' fully covers the induced difference vectors of the characteristic of s so teaching state s is not required if s' is taught. For example, state 6 in Figure 8(b) covers all the extreme rays induced by states 2, and 3. Conversely, if a state s is not a factor of any other states in $\mathcal{S}$ then it must be taught because s induces the extreme ray $[s \cdot \cos(\frac{2\pi}{s}), s \cdot \sin(\frac{2\pi}{s}), s]$ that can only be covered by s. ☐



(a) (b) (c) (d)

Figure 8: Polygon Tower. a) All feature difference vectors for $n = 6$. b) Top-down view of the extreme vectors of the primal cone for $n = 6$. c) TIE running time on polygon tower with increasing n. d) The teaching dimension (optimal) vs. the demonstration set size found by TIE. They overlap. In fact, TIE finds the exact correct optimal teaching sets on polygon tower.

We run TIE with greedy set cover on a family of polygon towers with $n \in \{3, 4, 5, 6, 7, 8, 12, 16, 20, 24, 32, 44\}$. We verify TIE's solution to the ground truth minimum demonstration set established in Proposition 17.

We observe that TIE always recovers the correct minimum demonstration set. This can be observed from the overlap curve of the optimal size of the teaching set (shown in orange) and the size of the teaching set found by TIE (shown in green) in Figure 8(d).

We also observe that TIE runs quickly. We plot the running time of TIE over instance size n in a log-log plot in Figure 8(c). For each n, we average the running time over 3 independent trial runs. The straight line of this log-log plot shows that our algorithm indeed runs in polynomial time. The empirical estimate of the slope of the linear curve (after omitting the first three outlier points for small n) turns out to be 4.67 implying a running time of order $O(n^5)$ on this family of instances. Our algorithm has a worst-case running time of order $O((|\mathcal{S}||\mathcal{A}|)^3)$ and for $|\mathcal{S}| = |\mathcal{A}| = n$ as in this example, it is $O(n^6)$.

8.2 Pick the Right Diamond

The MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \gamma, \mu)$ that describes the Block Programming problem is defined as follows :

1. A state $s \in \mathcal{S}$ is specified by an n size board where the cells are indexed $\{1, \cdots, n\}$. Each cell contains one of the four diamonds or be empty leading to a total of $5^n - 1$ states of non-empty boards.
2. The action space is given as $\mathcal{A} = \{1, \cdots, n\}$ where each action a represents picking an object at location a and removing it from board.
3. The learner receives a reward of -1 for picking the rightmost diamond with the largest edge and -2 for all other actions on a non-empty board. Once the board is empty it receives a reward of 0. The disocunt factor γ is 0.9.
4. The environment transitions deterministically to update the board if agent picked the right object, i.e., the rightmost object with largest edge otherwise it remains the same. The initial state distribution μ is uniform on $\mathcal{S}$.

The optimal policy defined by the reward structure above is to pick the diamond in order of decreasing number of edges. In case of ties, the rightmost diamond should be picked.

8.3 Visual Block Programming in Maze with Repeat Loop

The MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \gamma, \mu)$ that describes the Block Programming problem is defined as follows :

1. A state $s \in \mathcal{S}$ is specified by an $n \times n$ board with a turtle cell and a goal cell $\in [n^2] \times [n^2]$ and a turtle orientation $\in \{L, R, U, D\}$ denoting whether the turtle is facing left, right, up and down, refer to figure 6 for an example state. There is also a partial code of upto a constant size c, giving us a total of $4c(n^4 - n^2)$ states.

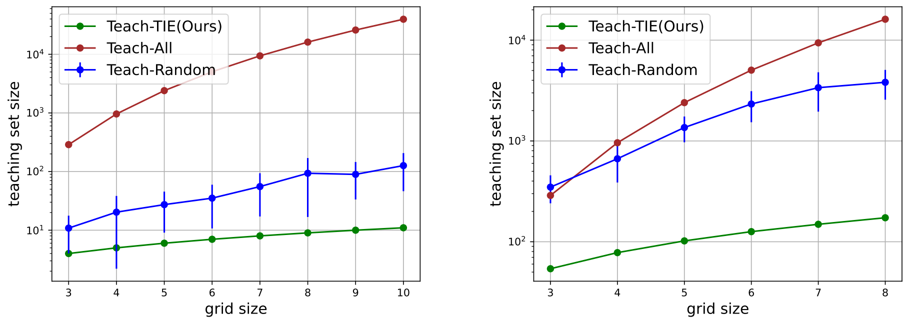


Figure 9: Performance of TIE compared to other baselines on visual programming task with local(on the left) and global features(on the right).

2. The abstract action space is given as $\mathcal{A} = \{TL, TR, MV\} \cup \{R_k\text{-}MV : k \in \{3, \cdots, n-1\}\}$ where *TL, TR, MV* represent simple code block that when executed allows the turtle to turn left, right and move forward actions respectively and R_k-MV represents a complex block of repeat loop that allows the turtle to move forward by k-step. In total, we have n actions where taking an action meaning adding the corresponding code block to the partial code in the state.
3. The learner receives a reward of -1 for using a simple code block action i.e. action $a \in \{TL, TR, MV\}$ and -2 for taking complex action R_k-*MV*. The disocunt factor γ is 0.9.
4. The environment transitions deterministically to update the orientation/position of the agent based on its chosen action. The initial state distribution μ is uniform on $\mathcal{S}$.

The goal of the teacher is to teach the optimal policy to write a succint piece of code which when executed helps to lead the learner to the goal cell. The teacher has to do this by showing smallest size of (state, action) demonstration dataset.

Local vs Global Features: We compared teaching multiple learner using different feature functions. In particular, we use two features representation : 1) a local feature representation, and 2) a global feature representation. We find that teaching a learner with local feature representation requires significantly less teaching dataset to teach the target policy. Furthermore, this dataset is also very human intuitive. In both scenarios, our teacher is significantly better when compared to the Teach-Random and Teach-All teaching baselines as shown in figure 9.

9 Feature Representation for Visual Programming

9.1 Local Feature Representation

This feature representation effectively captures the spatial relationship between the turtle and the goal, as well as the impact of different actions.

9.1.1 State and Action description

- `board`: A 2D array representing the game board with cells indicating the agent's orientation (U for up, D for down, L for left, R for right).
- `agent_pos`: A tuple (x, y) representing the agent's current position on the board.
- `goal_pos`: A tuple (x, y) representing the goal's position on the board.
- `action`: A string representing the specific action taken by the agent (e.g., *'TL', 'TR', 'MV', 'R_k-MV'* etc.).

9.1.2 Feature Vector Construction

1. **Relative Quadrant of Goal**: Compute the relative quadrant of goal from agent's orientation.

2. **Forward Distance**: Compute the distance from the agent to the goal in the direction the agent is facing.

We refer the interested readers to the supplementary material for the code.

9.2 Global Feature Representation

This feature representation captures the essential spatial and action-related aspects of the maze from global perspective.

9.2.1 State and Action description

- `board`: A 2D array representing the game board, where each cell indicates the agent's orientation (U for up, D for down, L for left, R for right).
- `agent_pos`: A tuple (x, y) representing the agent's current position on the board.
- `goal_pos`: A tuple (x, y) representing the goal's position on the board.
- `action`: A string representing the specific action taken by the agent (e.g., *'TL', 'TR', 'MV', 'R_k-MV'* etc.).

9.2.2 Feature Vector Construction

1. **Position Comparison**:
    - x_comp: Comparison of the agent's x-coordinate with the goal's x-coordinate.
    - y_comp: Comparison of the agent's y-coordinate with the goal's y-coordinate.
2. **Orientation Index**:
    - Mapping of agent's orientation to an index (0 for U, 1 for D, 2 for L, 3 for R).
3. **Forward Distance**:
    - The distance from the agent to the goal in the direction the agent is facing.

The size of the feature vector is $3 \times 3 \times 4 \times (n-1) \times k,$ where n is the size of the maze, and k is the number of possible actions.

Refer to supplementary materials for the code.

10 Compute Resources

We ran all our experiments on an Apple Macbook M1 laptop with 16GB ram.