Reinforcement Teaching

by

Calarina Muslimani

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

© Calarina Muslimani, 2022

Abstract

While traditional machine learning algorithms learn to solve a task directly, meta-learning aims to learn about and improve another learning algorithm's performance. However, existing meta-learning methods either only work with differentiable algorithms or are handcrafted to improve a specific component of an algorithm. Therefore, we develop a unifying meta-learning framework called *reinforcement teaching* to improve the learning process of any algorithm. Within the reinforcement teaching framework, a teaching policy is learned through reinforcement to improve a student's learning. To effectively learn such a teaching policy, we develop a reward function based on *learning progress*, allowing the teacher's policy to maximize the student's performance more quickly. Further, we introduce a *parametric-behavior embedder* that learns a representation of the student's learnable parameters from its input-output behavior. Finally, to demonstrate the effectiveness of reinforcement teaching, we perform a case study applying reinforcement teaching to the automatic curriculum learning domain. In this setting, a curriculum policy is learned that selects sub-tasks for a reinforcement learning student, outperforming handcrafted heuristics and previously proposed reward functions. To that end, reinforcement teaching is a framework capable of unifying different meta-learning approaches while effectively leveraging existing tools from reinforcement learning research.

Preface

A part of this thesis is in joint collaboration with Alex Lewandowski, Dale Schuurmans, Matthew E. Taylor, and Jun Luo. No part of this thesis has been previously published.

"It is not the critic who counts; not the woman who points out how the strong woman stumbles, or where the doer of deeds could have done them better. The credit belongs to the woman who is actually in the arena, whose face is marred by dust and sweat and blood; who strives valiantly; who errs, who comes short again and again, because there is no effort without error and shortcoming; but who does actually strive to do the deeds; who knows great enthusiasms, the great devotions; who spends herself in a worthy cause; who at the best knows in the end the triumph of high achievement, and who at the worst, if she fails, at least fails while daring greatly, so that her place shall never be with those cold and timid souls who neither know victory nor defeat."

- Theodore Roosevelt

Acknowledgements

First and foremost, I would like to thank my advisor, Matthew E. Taylor, for his constant support throughout this thesis. I am truly grateful to have an advisor that acts as my champion. Secondly, I would like to acknowledge the consistent efforts of my collaborator, Alex Lewandowski. I would also like to thank my writing coach, Dr. Antonie Bodley, for all of her feedback on this thesis.

Most importantly, I would like to thank my family for their ongoing love and support throughout my graduate program. Finally, I am incredibly grateful for my partner, Kerrick. On my worst days, you always find a way to show me the light.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

As machine learning becomes ubiquitous, there is a growing need for algorithms that generalize better, learn more quickly, and require less data. One way to improve machine learning methods without having to hand-engineer the underlying algorithm is through a process called meta-learning. Meta-learning can be thought of as "learning to learn," in which the goal is to learn about and improve another machine learning algorithm [1].

Various sub-domains have emerged that design handcrafted solutions for learning about and improving a specific component of a machine learning process. The work in these sub-domains has focused on solving one specific problem, whether that be finding the best way to augment data [2], sample minibatches [3], adapt objectives [4], or poison rewards [5]. Consequently, the meta-learning methods used in these domains have been over-fit to solve the specific problem and, therefore, may not be easily applied to solve new problems in a different domain. Furthermore, current literature fails to recognize that a more general framework can be used to address multiple problems across these varied sub-domains simultaneously.

Therefore, this thesis takes an important step toward answering the following question:

Can we develop a unifying framework for improving machine learning algorithms that can be applied across sub-domains and other learning problems?

As a critical step toward this unifying framework, we introduce *reinforcement teaching*, a novel solution concept that frames meta-learning in terms of a Markov decision process (MDP), in which a teaching policy interacts with a student's learning process to achieve a goal. In reinforcement teaching, a teacher learns a policy through reinforcement learning to improve the student's learning process. The teacher observes a representation of the student's behavior and then takes actions that adjust components of the student's learning process. The teacher changes aspects of the student's learning process that the student is unable to change on its own, such as the objective, optimizer, data, or environment. The teacher's reward is then based on the student's relative improvement.

Notably, the choice of action space for the teacher results in different meta-learning problem instances (See Figure 1.1). By having a flexible action space, our single teaching architecture can learn a variety of policies, such as a curriculum policy to sequence tasks for an RL student or a step-size adaptation policy for a supervised learning student.

Our reinforcement teaching framework has several advantages to both gradient-based meta-learning and other RL teaching methods:

1. Our MDP formalism is domain agnostic and thus does not rely on problem-specific heuristics [2, 3, 6–11] or access to the optimal student model, as required for most machine teaching approaches [5, 12].

2. Unlike gradient descent meta-learning methods [13–15], our framework learns a teaching policy; therefore, the teacher has the ability to adapt to the student's needs at each step in the student's learning process.

3. Our MDP approach does not assume all student learning components are fully-differentiable, an assumption typically necessary for gradient descent meta-learning methods.

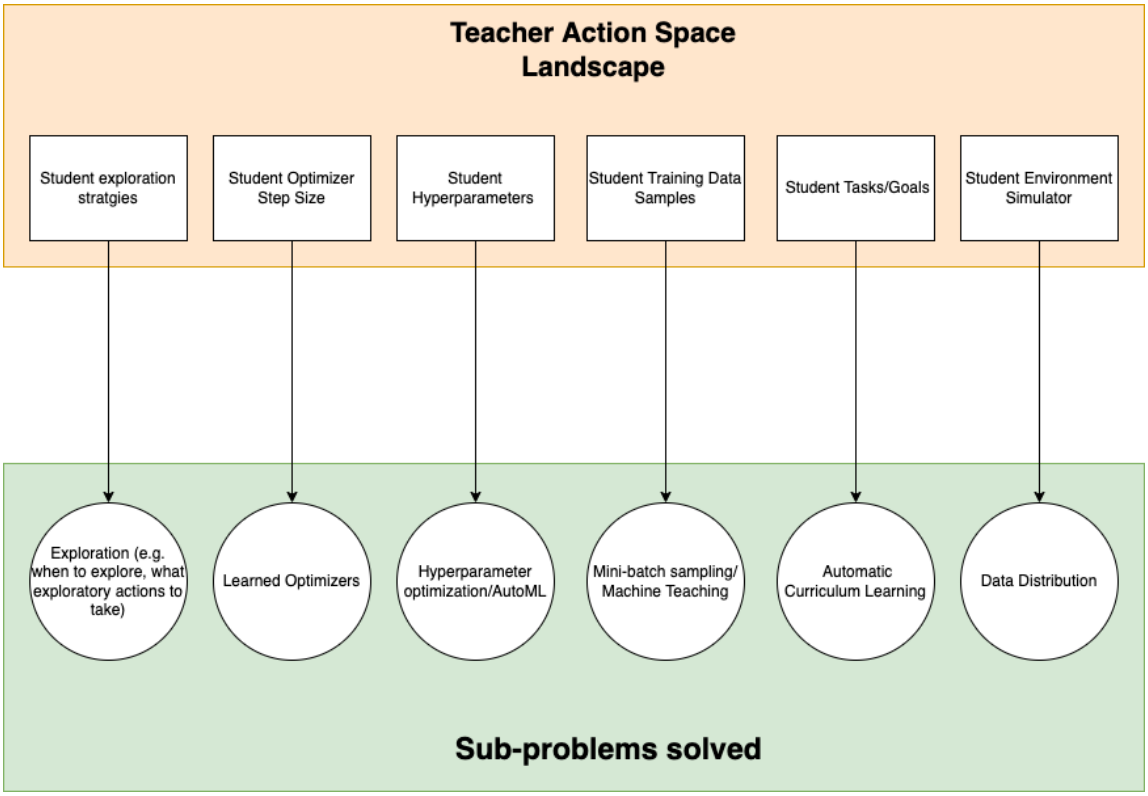


Figure 1.1: This diagram demonstrates how the choice of teacher action space leads to the teacher learning a policy to solve various sub-problems.

To showcase the potential of reinforcement teaching, we perform a case study applying reinforcement teaching to the automatic curriculum learning domain. In both discrete and continuous control settings, we show that reinforcement teaching can be used to learn a policy that selects sub-tasks for a reinforcement learning student. This curriculum policy guides the student's experience and improves its learning on a target task.

1.1 Thesis Contributions

This thesis makes the following contributions:

1. The reinforcement teaching framework is formalized as an MDP in which the teacher learns a policy that helps a student to reach a goal quickly.

2. Rather than having the teacher learn directly from the student's parameters, a

parametric-behavior embedder learns a state representation from the student's behavior (i.e., inputs and outputs). This provides a domain-agnostic state representation that improves the teacher's learning.

3. A *learning progress* reward function is defined that further accelerates learning by improving the teacher's credit assignment.

4. We demonstrate the effectiveness of reinforcement teaching in the curriculum learning domain in which the teacher learns a policy that selects sub-tasks for an RL student and improves its learning over several RL teaching baselines.

Contributions (1-3) allow our framework to be leveraged by different kinds of students in different problem settings. This allows existing approaches to be unified under our reinforcement teaching framework.

1.2 Thesis Outline

We begin with Chapter Two by discussing background information on the domains relevant to the reinforcement teaching framework. In Chapter Three, we focus on the proposed framework and the methodologies used. This is followed by Chapter Four, where we delve into an application of reinforcement teaching, curriculum learning, and the associated experimental results. Finally, we end the thesis in Chapter Five by discussing limitations and future work.

Chapter 2

Background

In this chapter, we provide background information on reinforcement learning as well as the domains pertinent to the reinforcement teaching framework. We then discuss the gap in research that this thesis fills.

2.1 Reinforcement Learning

In the reinforcement learning (RL) paradigm, agents interact with an environment with the goal of maximizing a scalar reward signal. Through trial and error, an RL agent learns which actions yield a higher reward. RL problems are commonly modeled as a Markov decision process (MDP). An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \mu, \gamma \rangle$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. The state must include all relevant past agent-environment interactions necessary to make future decisions. If this property holds, a state s is considered Markov. More formally, a state s is Markov if and only if:

$$P(S_{t+1}|S_t, A_t) = P(S_{t+1}|S_1, A_1, S_2, A_2..., S_t, A_t)$$

At every time-step t, the agent finds itself in a state $s \in \mathcal{S}$ and must choose an action a from the set $\mathcal{A}$. Further, the transition rule, T of the environment (stochastic or deterministic), determines the $p(s', r|s, a)$, the probability of transitioning to state s' and receiving reward r, given the agent was in state s and executed action a. Next, the reward function $R : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ maps a state and an action to a scalar reward,

and μ is the initial state distribution. Lastly, $\gamma \in [0, 1]$ is the discount factor, which determines the present value of future rewards. The agent's goal is to maximize the expected sum of discounted rewards, G_t at any time-step, t:

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

An agent interacts with its given environment by taking actions according to its policy π. A policy is a mapping from states to a probability distribution over all possible actions, $\pi : \mathcal{S} \rightarrow \mathcal{A}$. An agent aims to find the optimal policy, π_*, that maximizes the sum of discounted rewards by following policy π_* until termination T, with $T = \infty$ in the continuing setting.

Further, the state-action value, $q_\pi(s, a)$, defines the value of taking action a in state s under policy π. More formally, the state-action value function $q_\pi(s, a)$ is the expected sum of discounted rewards given that an agent is in state s, takes action a, and thereafter follows policy π.

$$q_\pi(s, a) \doteq E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right]$$

The state-action value function can be estimated from the agent's experience with the environment. In an iterative process, $q_\pi(s, a)$ can be learned and can converge to the optimal state-action value function $q_*(s, a)$ under certain conditions [16, 17]:

$$q_*(s, a) \doteq max_\pi q_\pi(s, a) \ \forall s, a$$

Furthermore, given $q_*(s, a)$, an agent can obtain the optimal policy π_* by acting greedily with respect to $q_*(s, a)$:

$$\pi_*(s, a) = argmax_{a \in \mathcal{A}} q_*(s, a)$$

2.2 Meta-Learning

Meta-learning is often described as "learning to learn," in which the goal is for a meta-learner to learn about and improve upon a base learning algorithm. Inherent

to meta-learning are two objectives: (1) the base learning objective is to learn a task (e.g., image classification), and (2) the meta-learning objective is to improve the base learner's ability. These objectives are hierarchical in nature; therefore, meta-learning can be cast as a bi-level optimization process. In this setup, while a base learner is solving a task, a meta-learner updates the base learning algorithm such that the baser learner improves the meta-objective [1]. This two objective process is a defining feature across various meta-learning methods.

However, meta-learning approaches differ in what component of the base learner's algorithm the meta-learner learns about and improves on. For example, in MAML, the meta-learner aims to learn the best base learner initialization [18]; whereas, in the learned optimizer literature, the meta-learner can learn how to adjust the step-size of the base-learning algorithm. Other examples include the related sub-areas of machine-teaching and curriculum learning. In machine teaching, the goal of the meta-learner is to learn the best sequence of training datasets for the base-learner [12], whereas, in curriculum learning, the meta-learner's goal is to learn the best sequence of source tasks for the base-learner [19, 20].

2.3 Machine Teaching

Machine teaching is a general teacher-student paradigm in which a teacher is used to guide a student. A widely studied application of machine teaching is the supervised learning setting in which a teacher is tasked with choosing the best training set such that a machine learning student can learn a target model [3, 12]. Recent work has applied machine teaching to RL students. In the RL student setting, machine teaching has been used to study a wide range of problems, from finding the best set of demonstrations to finding the best reward shaping strategy [5, 21].

The primary issue with traditional machine teaching approaches is the strict assumptions they make on behalf of the teacher. In traditional approaches, the teacher must have full knowledge of the student and the optimal target model [12]. Unfor-

tunately, these assumptions are unrealistic in practice. If we assume the teacher has access to the optimal target model, then we already know how to solve the task at hand. Therefore, it is not necessary to train another student agent/model to solve the task. To relax these assumptions, recent work treats the teacher's task of finding the optimal training set for the student as an RL problem [3, 22, 23].

2.4 Automatic Curriculum Learning

Learning tasks of increasing difficulty is fundamental to human learning. In machine learning, this "easy to hard" approach is called *curriculum learning*. In traditional curriculum learning approaches, a designer must (1) be able to order all sub-tasks by difficulty and (2) design a curriculum of sub-tasks that results in improved student learning on the target task. To develop such a curriculum, these methods are often time-consuming and rely on expert domain knowledge [24, 25]. In contrast, automatic curriculum learning aims to learn the optimal curriculum over tasks such that an agent's competence on a target task or over a set of tasks is maximized [19].

In the past literature, curricula have been learned through a variety of mechanisms. One subset of automatic curriculum learning focuses on learning the best sequence of initial states for an RL agent to learn from. For a typical RL environment, an agent is placed in a start state, and its job is to learn how to reach a goal state. In environments with rich state spaces, this task is made quite challenging by the large amount of exploration required for the agent to reach the goal state and obtain a learning signal. One approach to alleviate this issue is to allow the agent to start learning from states closer to the goal state and, over time, increase the distance between the agent's start state and the goal state. Therefore, at later stages in the curriculum, the agent is placed progressively closer to the target start state. This approach has seen success in several continuous control robotic domains [26, 27].

Another body of work surrounding automatic curriculum learning aims to learn the best sequence of goals for an RL agent. Racaniere et al. (2019) and Florensa

et al. (2017) apply Generative Adversarial Networks [28] in this setting. By using dual agents, one agent is used to generate goals and another is used to evaluate the proposed goal's quality. Further, Zhang et al. (2020) developed a curriculum of goals of increasing complexity by using a measure of epistemic uncertainty based on an ensemble of value functions. This method enables the proposed goals to be neither "too easy" nor "too difficult"; however, this method maintains multiple networks for each value function and thus can be computationally expensive.

Another promising line of research is student-teacher curriculum learning. In this setting, a teacher agent learns, via RL, to provide appropriate tasks to a student agent based on the student's current skill set [29]. Narvekar *et al.* (2017) proposed a two-level MDP, in which a teacher agent operates in a curriculum MDP that chooses tasks for the student, and the student agent operates in an MDP associated with the given task [6, 7]. Campero et al. (2020) extended the teacher-student curriculum formulation in the Deep RL context by taking an adversarial training approach. The student maximizes its reward by reaching goals quickly, therefore, would rather receive "easy" tasks from the teacher. However, the teacher in this case maximizes its reward by only proposing goals to the student that are of appropriate difficulty. This yields an adversarial relationship between the teacher and student. As noted, this method makes progress towards applying teacher-student curriculum learning in the Deep RL setting. However, by the construction of the teacher's reward function, it is likely more suited for navigation-related tasks and may not easily port over to the continuous control setting.

These approaches can be contrasted with other formulations of the student-teacher setting that treats the teacher's problem of selecting a task as a multi-armed bandit [30, 31]. In this setting, the teacher agent relies on learning progress [32] to detect how much improvement a student is making on a given task. Using a multi-armed bandit algorithm, the teacher's goal is to find a policy over tasks that continually maximizes the student's overall learning progress. The use of bandits simplifies the

problem because we no longer need to represent the state of the student's learning process. However, it sacrifices the teacher's ability to differentiate between different students.

2.5 Learning Progress

Connected to the idea of teaching, is a rich body of literature on learning progress. Learning progress prescribes that a learning agent should focus on tasks for which it can improve on. This mechanism drives the agent to learn easier tasks before incrementally learning tasks of increasing complexity [32]. Learning progress has been characterized in several ways, such as the change in model loss, model complexity, and prediction accuracy. In addition, learning progress has been successfully applied in a variety of contexts, including curriculum learning [29–32], developmental robotics [32–34], and intelligent tutoring systems [35].

2.6 Gaps in Literature

Across the various domains we have previously discussed, researchers have sought to learn a teaching policy via RL to control a particular aspect of a student ML process [3, 4, 6–11, 36, 37]. In these works, the overarching goal is for the teacher to learn how to adjust the student's learning algorithm in order to improve the student's task ability. However, the specific component of the student's learning algorithm the teacher learns about is dependent on the sub-domain. For example, in research focused on adaptive loss functions, an RL teacher learns a teaching policy to adapt the parameters of the student's loss function [8]; while in the data simulator literature, an RL teacher learns a policy that adjusts the student's data distribution [11]. Notably, these works are narrowly focused on solving one specific problem. Consequently, the solution methods are typically handcrafted to fit the specific problem. Therefore, the solution method to solve one problem may not easily generalize to solve a new

problem. This thesis, in contrast, focuses on crafting a general-purpose solution method that can be used to solve various meta-learning sub-problems. Therefore, we propose the reinforcement teaching framework as a *problem-agnostic* approach that enables tractable meta-learning across diverse problems/domains.

Chapter 3

Reinforcement Teaching

In this chapter, we describe the reinforcement teaching framework. We start by describing the framework's goal, followed by the abstraction of the student learning process that forms the "teaching environment." We then describe the Teaching MDP — the backbone of the reinforcement teaching framework.

3.1 Reinforcement Teaching

In reinforcement teaching, *student* refers to any learning agent or machine learning model, and *teacher* refers to an RL agent whose role is to adapt to and improve the student's learning process. We formulate the teacher's responsibility of improving the student's learning process as an MDP, in which a teaching policy interacts with a student's learning process to achieve some goal.

By using the MDP formalism, reinforcement teaching is *agnostic* to the meta-learning problem of interest. Therefore, the teacher is not limited in how to improve the student's learning process. As an analogy, consider a personal trainer in a gym — the goal of a personal trainer is to help an athlete more quickly reach their fitness and health goals. The trainer has several means to accomplish this; they can customize the athlete's workouts *and* their meal plans. In other words, the trainer is not confined to just one form of help. This is directly analogous to the teacher's role in the reinforcement teaching framework. The teacher can improve the student's learning

Full Form	**Notation**
Student	f_θ
Student learnable parameters	θ
Student learning domain	$\mathcal{D}$
Student learning algorithm	$\mathcal{A}lg$
Student objective function	$J(f_\theta, \mathcal{D})$
Student performance measure	$m(f_\theta, \mathcal{D})$
Student performance threshold	m^*
Teaching environment	$\mathcal{E}(\Theta)$
Student input	x_i
Student output	$f_\theta(x_i)$
Mini-state	$\{x_i, f_\theta(x_i)\}_{i=1}^M$
Parametric-behavior embedder	PE
Learning progress reward function	LP

Table 3.1: This table shows the primary notation associated with the reinforcement teaching framework. The details of each term will be described in Section 3.1.

process by using various methods. For example, the teacher can learn a curriculum policy [6, 7] or a policy to sample appropriate training data [3] with the goal of improving the student's learning.

3.2 Teaching Environment

To start, we define the student learning process and its components. Consider a student f_θ, with learnable parameters $\theta \in \Theta$. The student receives experience from a learning domain $\mathcal{D}$, which can be labelled data (supervised learning), unlabelled data (unsupervised learning), or an MDP (reinforcement learning). How the student interacts with its domain and how it learns, given that interaction is specified by the student's learning algorithm $\mathcal{A}lg$ that optimizes the student's objective function $J(f_\theta, \mathcal{D})$. Over time, the student's learning algorithm updates the student's parameters through interaction with the learning domain and the objective function, $\theta_{t+1} \sim \mathcal{A}lg(f_{\theta_t}, \mathcal{D}, J)$. The goal is to maximize a performance measure, $m(f_\theta, \mathcal{D})$, that evaluates the student's current ability.

One natural choice for m is the objective function directly optimized by $\mathcal{A}lg$, but m can also be a non-differentiable surrogate objective such as accuracy in classification or the Monte-Carlo return in RL. Note that the learning domain, $\mathcal{D}$, can contain many tasks. This would occur, for example, in the domain of curriculum learning, in which there are several sub-tasks that an agent needs to learn to solve a more difficult target task. In this setting, we can simply index $\mathcal{D}$ by the sub-task index i, $\mathcal{D}_i$.

The combination of the student, learning domain, learning algorithm, and performance measure is hereafter referred to as the *student learning process*: $\mathcal{E}(\Theta) = \{f_\theta, \mathcal{D}, \mathcal{A}lg, m\}$. Specifically, the student learning process summarizes the components contributing to the student's parameters as it learns the optimal parameters that maximize its performance measure $\theta^* = \arg\max_\theta m(f_\theta, \mathcal{D})$.

The teacher's goal is to then learn about and improve the student's learning process via reinforcement learning; therefore, $\mathcal{E}(\Theta)$ will act as the environment the teacher

interacts in, namely the *teaching environment*.

3.3 Teaching MDP

We now describe the state representation, reward function, and action space used to construct the Teaching MDP.

3.3.1 States of the Teaching MDP

We define the state of the teaching environment to be the current learnable parameters of the student, $s_t = \theta_t$. The state space is then the set of possible parameters, $\mathcal{S} = \Theta$, as similarly suggested in Narvekar et al. (2017) and Zhang et al. (2020). The initial state distribution, μ, is determined by the initialization method of the parameters, such as Glorot initialization for neural networks [38]. Furthermore, the transition function, T, is then defined through the learning algorithm, $\theta_{t+1} \sim \mathcal{A}lg(f_{\theta_t}, \mathcal{D}, J)$. Although the learning algorithm is known, the explicit transition dynamics (i.e., probability of transitioning from θ_t to θ_{t+1}) is typically unknown. This can be true in several cases. To start, there can be randomness due to the student's optimizer J. For example, if the student's optimizer used stochastic or batch gradient descent. Moreover, randomness can also arise due to the student's learning domain, D. For example, if the student was an RL agent, then the student's environment dynamics are typically unknown. Therefore, due to the unknown transition dynamics, at every time-step, it is only possible to sample $\mathcal{E}(\Theta)$ for the next state θ_{t+1} and reward r_{t+1}.

The sequence of learnable parameters, $\{\theta_t\}_{t\geq 0}$, form a Markov chain, as long as f_θ, $\mathcal{D}$ and $\mathcal{A}lg$ do not maintain a state that depends on the parameter history. This occurs when the learning algorithm $\mathcal{A}lg$ is Stochastic Gradient Descent [39, 40]. While adaptive optimizers, like Adam [41], violate the Markov property of $\mathcal{A}lg$, we show empirically in Section 4.4 that this does not hinder the teacher's ability to learn an effective curriculum policy.

Although Θ is a Markov state representation, it is not ideal for learning a teaching

policy. First, the parameter space is large and mostly unstructured. Ideally, the teacher's state representation should be much smaller than the concatenated set of parameters θ. In addition, using the parameter state representation would hinder the teacher's ability to generalize to new student models with different architectures or activations.

Parametric-behavior Embedder To avoid learning from the parameters directly, we propose the *parametric-behavior embedder* (PE), a novel method that learns a representation of the student's parameters from the student's behavior. To capture the student's behavior, we use the inputs and outputs of f_θ. For example, if we consider an RL student, the inputs of f_θ would be the environment states the student encounters, and the outputs can be the corresponding $Q(s, a)$ values or the corresponding actions with the highest value, $\arg\max_a Q(s, a)$.

To learn the PE state representation, we first assume that we have a dataset or replay buffer to obtain the student inputs, x_i. Then we can randomly sample a minibatch of M inputs, $\{x_i\}_{i=1}^M$ and retrieve the student's corresponding outputs, $f_\theta(x_i)$. The set of inputs and outputs $\hat{s} = \{x_i, f_\theta(x_i)\}_{i=1}^M$, or mini-state, provides local information about the true underlying state $s = \theta$.

To learn a representation from the mini-state, we recognize that $\hat{s}$ is a set and use a permutation invariant function h to provide the PE state representation $h(\hat{s})$ [42]. The input-output pair is jointly encoded before pooling, $h(\hat{s}) = h_{pool}\left(\{h_{joint}(x_i, f_\theta(x_i))\}_{i=1}^M\right)$, where h_{pool} is a pooling operation over the minibatch dimension (See Figure 3.1). We note that our parametric-behavior embedder does assume that the student has no memory. One example of the student having memory is if the student uses a recurrent neural network. If the student did have a memory, the student's input-output behavior would not fully capture the state of the student's learning process. However, we leave this consideration for future work.

We argue that the local information provided by the student's behavior, for a

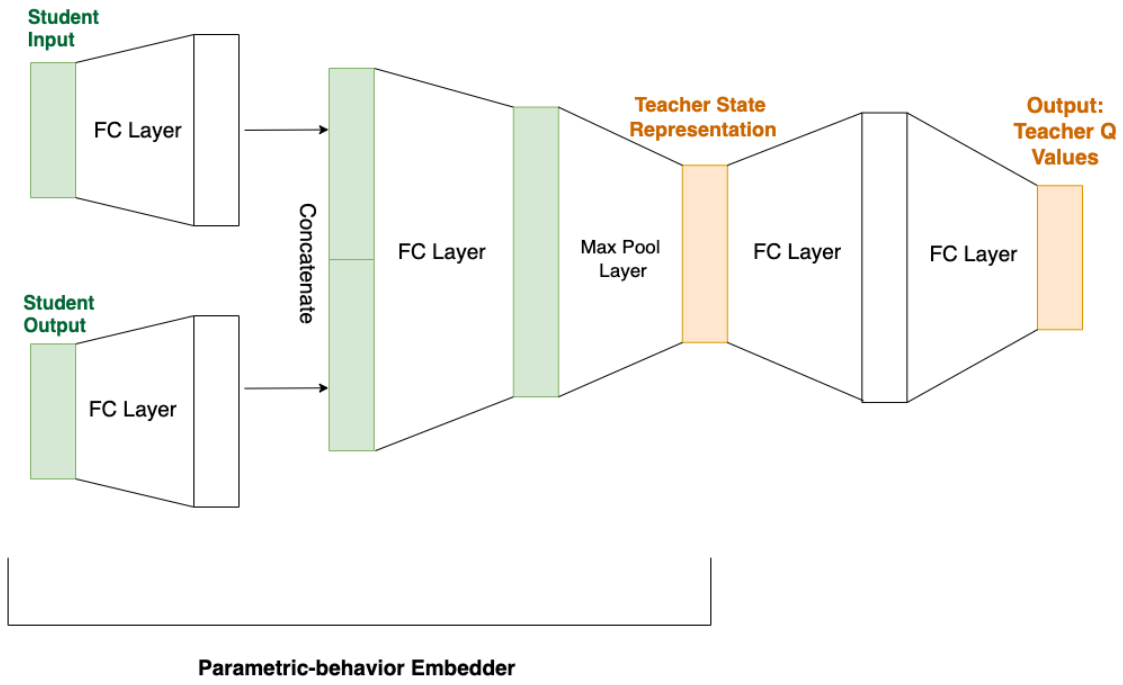


Figure 3.1: This is the basic architecture used for the teacher. Green indicates learning the parametric-behavior embedder state representation. Orange indicates learning the corresponding action-value functions.

large enough minibatch of inputs and outputs, is enough to summarize pertinent information about θ while still maintaining the Markov property. Moreover, methods that attempt to learn directly from the parameters must learn to ignore aspects of the parameters that have no bearing on the student's progress. This is inefficient for even modest neural networks. We demonstrate in Section 4.4 that learning from the PE state representation allows the teacher to learn an effective curriculum policy over several teaching baselines.

3.3.2 Rewards of the Teaching MDP

To specify the reward function for the teaching environment, we first identify common criteria for training and measuring a learner's performance. For ease of reference, let the student performance measure $m(f_{\theta_t}, \mathcal{D}) = m(\theta_t)$ at time-step t.

In the episodic learning setting, one naive approach, adapted from Narvekar et al. (2017), is the time-to-threshold reward function. With this reward function, the

teacher is trained until the student reaches a pre-defined performance condition, such as a sufficiently high performance measure (i.e., $m(\theta_t) \geq m^*$ for some threshold m^*). In this case, the teacher's reward function is constant $R(\theta) = -\mathbb{I}\left(m(\theta_t) < m^*\right)$ until the student's performance condition, $m(\theta_t) \geq m^*$, is reached, which then terminates the teacher's episode. Therefore, the teacher is rewarded for taking actions such that the student reaches a performance threshold m^* as quickly as possible.

Another common training procedure in the undiscounted, episodic case is to train a learner for T time-steps and record the final performance. In this case, the teacher's reward function is zero everywhere except that $R(\theta_T) = m(\theta_T)$; thus, the teacher is rewarded for taking actions that improve the student's final performance. Furthermore, similar to the discussion in Section 3.3.1, both sparse reward functions described are Markov, as long as the performance measure m is Markov.

Learning Progress Reward At a high level, the time-to-threshold and final performance reward functions serve to inform the teacher on which actions increase the student's performance; however, we demonstrate in Section 4.4 that these reward functions are insufficient to enable the teacher to learn an adequate teaching policy.

We argue that these sparse reward formulations lack integral information about the student's learning process. As an alternative, we propose to shape the time-to-threshold reward function using the student's *learning progress*. The learning progress signal provides feedback about the student's capacity for improvement and can better inform the teacher about how its policy influences the student.

We define Learning Progress (LP) as the change in the student's performance measure given the student learning domain $\mathcal{D}$:

$$LP(\theta', \theta) = m(\theta') - m(\theta) \tag{3.1}$$

at subsequent states θ and θ' of the student's learning process. To shape the time-to-threshold reward, we can simply add the learning progress term, $LP(\theta', \theta)$, to

the existing reward $R(\theta')$ previously described. Therefore, our resulting LP shaped reward function is:

$$R(\theta', \theta) = -\mathbb{I}\left(m(\theta') < m^*\right) + LP(\theta', \theta) \tag{3.2}$$

until $m(\theta') \geq m^*$, terminating the episode. It follows that learning progress is a potential-based reward shaping, given by $R' = R + \Phi(\theta') - \Phi(\theta)$, where the potential is the performance measure $\Phi(\theta) = m(\theta)$.

This means that combining learning progress with the time-to-threshold reward does not change the optimal policy [43]. With the inclusion of learning progress, the teacher can now identify actions that improve the student's performance, even without having the student reach the performance threshold. The learning progress term provides critical information on how the teacher's action affected the student's performance in its learning domain $\mathcal{D}$. The learning progress term indicates the extent to which the teacher's adjustment (i.e., action) improved or worsened the student's performance. For example, consider the curriculum learning setting in which the teacher's actions correspond to changing the student's sub-tasks. The teacher would propose a sub-task for the student to learn in. If this sub-task was too difficult, then the student's learning progress would be 0 or negative. The learning progress term then indicates to the teacher that given the student's current skill level (as described by the student's parameters), this sub-task was too difficult and did not improve the student's performance. Therefore, the teacher will be deterred from selecting this sub-task. We show empirically in Section 4.4 that the LP reward is a richer reward signal that can enable the teacher to learn a more effective teaching policy.

Moreover, it is important to note that given the reward function, the teacher's goal is to maximize its average discounted (i.e., γ is non-zero) sum of rewards. Issues may arise if the teacher were to simply act greedily (i.e., γ is zero) and only maximize its immediate reward. We hypothesize that by only maximizing immediate reward, the teacher may be incentivized to propose any action that results in higher learning

progress irrespective of its relevance to the student's target task. This can result in the teacher proposing actions that are not necessary for the student. We leave it to future work to empirically investigate the effects of using an RL teacher agent versus a bandit teacher agent.

3.3.3 Actions of the Teaching MDP

Thus far, we have described the state representation and reward function of the Teaching MDP. However, in order for the teacher to improve the student's learning process, it must be able to intervene. The teacher's responsibility is to oversee the student's learning process and take actions that adjust this process. The action set, $\mathcal{A}$, enables the teacher to control any component of the student's learning process. An action can change internal components of the student, such as its hyperparameters, learning algorithm, or objective function, as well as external components such as the student's learning domain (e.g., the task or data sample). Notably, the choice of action space leads to different meta-learning problem instances, thereby allowing our reinforcement teaching framework to be used across a wide range of problems. This can include curriculum learning (learning a policy for sequencing tasks of $\mathcal{D}$), learning to sample (learning a policy for sampling mini-batches from $\mathcal{D}$), and adaptive optimization (learning to adapt the student's step-size).

Furthermore, the frequency of the teacher's intervention is also dependent on the instantiation of the meta-learning problem. More specifically, the action set determines the time-step of the teaching MDP. The base time-step of the teaching environment is each student parameter update. The teacher operates at this frequency in settings where it controls an aspect of the learning algorithm, such as the step-size. However, in other meta-learning problem instances, such as curriculum learning, the teacher acts at a slower rate, thereby inducing a semi-MDP [44]. In the curriculum learning setting, the teacher's action changes the student's task. The student typically has a prolonged interaction in the proposed environment and makes several parameter up-

dates before the teacher takes another action. Therefore, depending on the problem instance, the teacher and the student may operate at different time scales.

With the states, transitions, rewards, and actions now fully defined, we have constructed the Teaching MDP, $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, \mu, \gamma \rangle$, as summarized in Figure 3.2. We are now able to learn a teaching policy for the Teaching MDP. When discussing specific instantiations of the Teaching MDP, we will specify what the teacher controls.

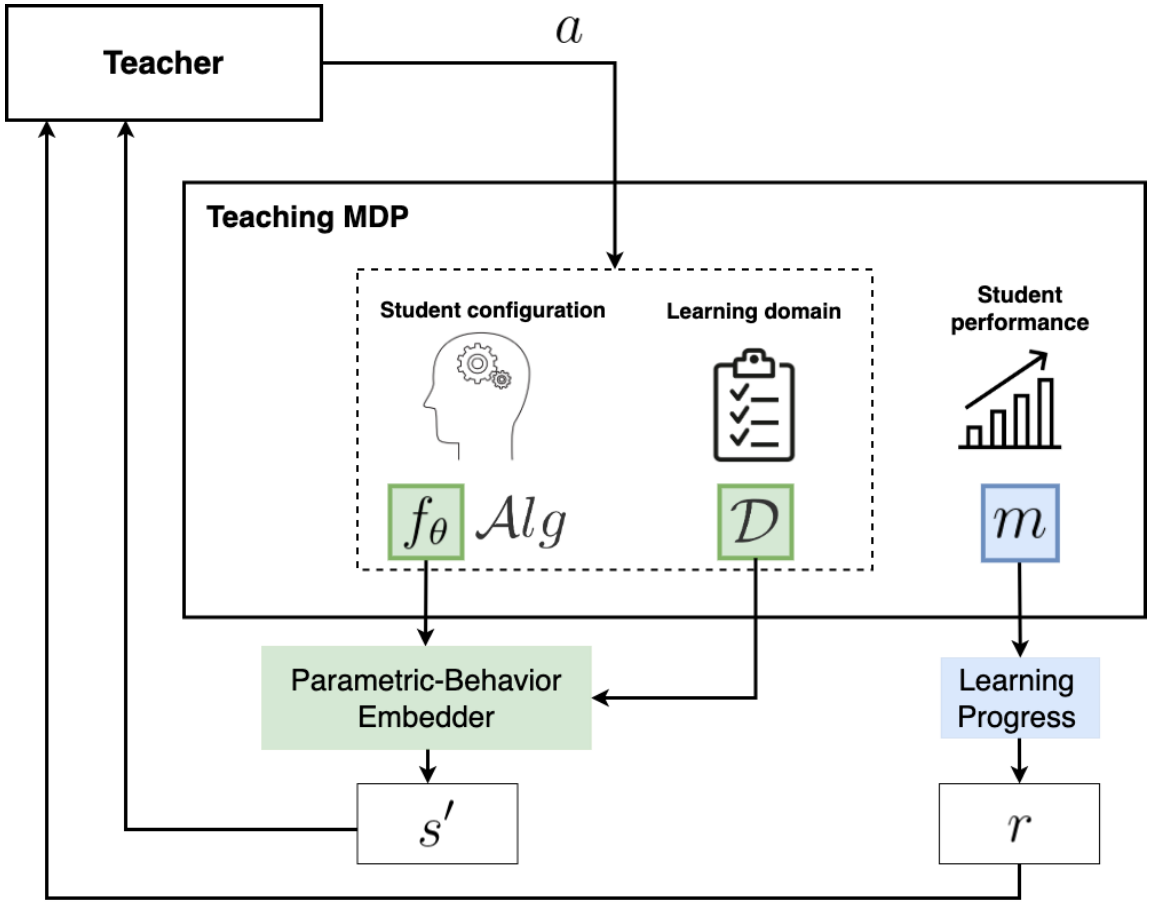


Figure 3.2: Teaching MDP: The teacher takes actions $a \in \mathcal{A}$, influencing an aspect of the student, $f_\theta, \mathcal{A}lg$ or its learning domain $\mathcal{D}$. The student will then update its model under the new configuration. The student's learning process will then output r, s'.

3.4 Teacher-Student Interaction Protocol

We summarize the basic interaction between the teacher and student when the teacher uses the parameter state representation and LP reward function:

1. The teacher RL agent selects an action $A \in \mathcal{A}$. This action will change a component of the teaching environment $\mathcal{E}(\Theta)$.

2. The student will then train its model under the new configuration.

3. We retrieve the updated student parameters $\theta^{s'}$ and set the next teacher state, S' to be $\theta^{s'}$.

4. We calculate the performance measure $m(\theta^{s'})$ on data/task $\mathcal{D}$.

5. We calculate the LP reward based on Eqn. 3.2 to get the teacher reward R.

6. We store the teacher's transition (S, A, R, S') in a replay buffer and update the teacher's model parameters according to the teacher's RL algorithm ψ^T.

We refer to the RL algorithm used to train the teacher agent as ψ^T, and the student's ML algorithm as ψ^s. The inputs to the framework are the teacher and student learning algorithms, the teacher's action set $\mathcal{A}$, and the student performance threshold m^*. The teacher's action set is problem-dependent and therefore defines the frequency of the teacher's action. For example, if the teacher's goal is to learn an adaptive step-size for the student's optimizer, then the teacher will propose an action (e.g., step-size) at every student time-step. However, if the teacher's goal is to learn a curriculum policy, then the teacher will propose an action (e.g., sub-task) at every student episode after several student parameter updates. Therefore, the time-scale and frequency of the teacher-student interaction can vary. Lastly, the output of this teacher-student process will be a teaching policy that the teacher can now use to improve the learning process of new students. The reinforcement teaching framework is further detailed in Algorithm 1.

Algorithm 1 Reinforcement Teaching Framework

Input: teacher RL algorithm ψ^T, student ML algorithm ψ^s, teacher action set $\mathcal{A}$, initial teacher parameters θ_T, learning domain $\mathcal{D}$, and student performance threshold $m^* \in [0, 1]$

Loop for each teacher episode:

Initialize student parameters θ_s and $m(\theta_s) = 0$

Set initial teacher state $S_0 = \theta_s$

While $m(\theta^s) < m^*$ **do**:

Choose teacher action $A \in \mathcal{A}$ and update $\mathcal{E}(\Theta)$

Train student via ψ^s and observe θ'_s

Evaluate student on task $\mathcal{D}$ to obtain $m(\theta'_s)$

Calculate R' based on Eqn. 3.2.

$S' = \theta'_s$

Store transition (S, A, R', S') in replay buffer

Update θ^T according to ψ^T

Chapter 4

Experiments

In this chapter, we discuss the experiments and the results of our proposed reinforcement teaching framework in the curriculum learning domain.

4.1 Environments

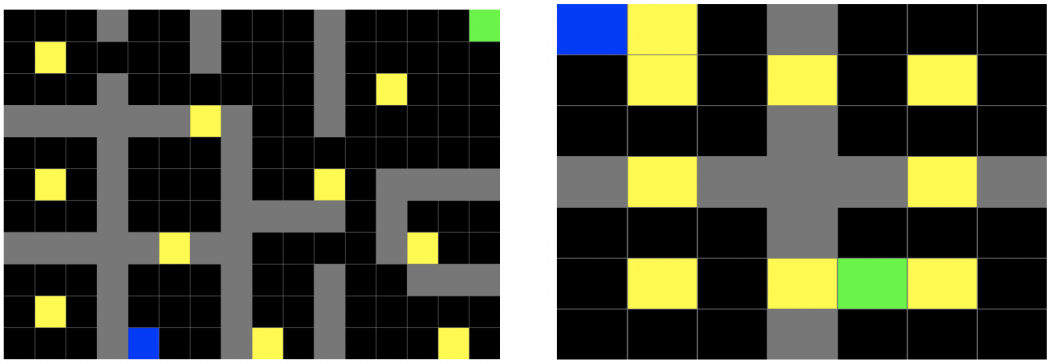
In our experiments, three environments were used to highlight the effectiveness of the reinforcement teaching framework in both discrete and continuous control environments. We describe the environments in order of increasing complexity. See Table 4.1 for a summary of the environment characteristics.

	Maze	**Four Rooms**	**Fetch Reach**
Env action type	Discrete	Discrete	Continuous
Number of env actions	4	3	NA
Env state space type	Discrete	Continuous	Continuous
Dimension of env state	1	243	10
Max number of time-steps	40	40	50
Env reward	$R(t) = 0$ except $R(T) = (.99)^T$	$R(t) = 0$ except $R(T) = 1 - 0.9 * \frac{T}{maxsteps}$	$R(t) = -1$ except $R(T) = 0$
Teacher action	Start state	Start state	Goal distribution
Number of teacher actions	11	10	9

Table 4.1: Environment characteristics. T denotes the time-step at termination.

Maze The Maze environment is an 11×16 discrete grid with several blocked states (see Figure 4.1a). An agent can take four deterministic actions: up, down, left, or right. If an agent's action takes the agent off the grid or into a blocked state, the agent will remain in its original location. The environmental reward is 0 until the agent

reaches the goal state. Once the agent reaches the goal state, the reward is $(.99)^t$, where t is the number of time steps it took the agent to reach the goal. To make this environment more difficult, we limited the max number of time-steps per episode to only 40. Therefore, the agent cannot simply randomly explore until it reaches the goal. Furthermore, in this environment, the teacher's action will change the student's start state. The teacher can start the student at 11 possible locations, including the start state of the target task. The teacher's action set contains both impossible tasks (e.g., start states that are completely blocked off) and irrelevant tasks (e.g., start states that are not necessary to learn for the target task). This environment is useful to study for several reasons. First, the reduced maximum time-step makes exploration difficult thus curriculum learning becomes a necessity. Secondly, the set of impossible and irrelevant sub-tasks in the teacher's action set ensure that the teacher is able to learn to avoid these actions and only suggest actions that enable the student to learn the target task efficiently (i.e., navigating from the blue to green state, see Figure 4.1a).



(a) Maze (b) Four Rooms

Figure 4.1: The green square represents the goal state, and the blue square represents the start state of the target task. Yellow squares indicate the teacher's possible actions — possible starting states for the student.

Four Rooms The Four Rooms environment is adapted from the MiniGrid suite [45]. It is a discrete state and action grid-world. Although the state space is discrete,

it is very large. The state encodes each grid tile with a 3 element tuple. The tuple contains information on the color and object type in the tile. Due to the large state space, this environment requires a neural network function approximator on behalf of the RL student agent. The large state space makes Four Rooms much more difficult than the tabular Maze environment. Similar to the Maze domain, Four Rooms has a fixed start and goal state, as shown in see Figure 4.1b. In addition, the objective is for an agent to navigate from the start state to the goal state as quickly as possible. In our implementation, we used the compact state representation and reward function provided by the developers. The state representation is fully observable and encodes the color and objects of each tile in the grid. The reward function is $1 - 0.9 * \frac{time-step}{maxsteps}$ for successfully reaching the goal and 0 otherwise. To make the environment more challenging, we reduced the maximum number of time-steps to 40. Moreover, there were three actions: turn left, turn right, and go forward. If an agent's action takes the agent off the grid or into a blocked state, the action is undone. Moreover, the teacher's action set is similar to the maze environment in that the action changes the student's starting state to one of 10 possible states (as shown by the yellow states in 4.1b).

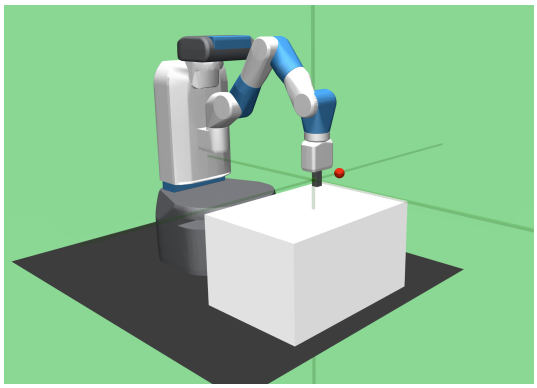


Figure 4.2: Image of Fetch Reach environment

Fetch Reach Fetch Reach is a continuous state and action simulated robotic environment [46]. It is based on a 7-DoF Fetch robotics arm, which has a two-fingered

parallel end-effector (see Figure 4.2). In Fetch Reach, the end-effector starts at a fixed initial position, and the objective is to move the end-effector to a specific goal position. The goal position is 3-dimensional and is randomly selected for every episode. Therefore, an agent has to learn how to move the end-effector to random locations in 3D space. Furthermore, the observations in this environment are 10-dimensional and include the Cartesian position and linear velocity of the end-effector. The actions are 3-dimensional and specify the desired end-effector movement in Cartesian coordinates. In addition, each action is applied for 20 simulator steps. As for the environmental reward, the reward function is sparse and binary. The agent receives a reward of 0 if the end-effector is at the goal position (within a tolerance of 5 cm) and -1 otherwise. As for the teacher's action set in Fetch Reach, the teacher controls the goal distribution. The goal distribution determines the location the goal is randomly sampled from. There are 9 actions in total, each action gradually increasing the maximum distance between the goal distribution and the starting configuration of the end-effector. Therefore, "easier" tasks are ones in which the set of goals are very close to the starting configuration. Conversely, "harder" tasks are ones in which the set of goals are far from the starting configuration of the end-effector. It is important to note, however, that the goal distribution of each action subsumes the goal distribution of the previous action. For example, if action 1 allows the goal to be sampled within the interval [0, .1], then action 2 allows the goal to be sampled within the interval [0, .2]. This allows for learning on "easy" tasks to be useful for learning on "harder" tasks.

4.2 Experimental Details

To formalize curriculum learning through reinforcement teaching, we establish the teaching MDP. To begin, the teacher's actions will control an aspect of the student's environment discussed in Section 4.1. For the student's learning algorithm, we used Q-learning [47], PPO [48], and DDPG [49], for the Maze, Four Rooms, and Fetch

Reach environments, respectively. This highlights that reinforcement teaching can be useful for a variety of students. For the PPO and DDPG students, we used existing open-source implementations [50, 51]. For a complete list of the hyperparameters used across these algorithms, see Table 4.2.

For the teacher's state representation, we consider two variants of the parametric-behavior embedder that use different student outputs f_θ. In both cases, the inputs are the states that the student encounters during its learning process. For PE-QValues, the embedded student outputs are the state-action values, whereas for PE-Action, the embedded student outputs are one-hot encodings of the student's greedy policy. In addition, for all reward functions, the performance measure is the student's return on the target task. Table 4.3 provides the performance threshold, m^*, for each environment.

Next, to train the teacher, we use the vanilla DQN [52] with a decaying epsilon policy. We follow the teacher-student interaction protocol as described in Algorithm 1. See Table 4.3 for a complete list of the training hyperparameters and Appendix A for the teacher hyperparameters and the associated grid-searches. Moreover, to access the teacher's curriculum policy, we evaluate the trained teacher's policy on a

	Maze	**Four Rooms**	**Fetch Reach**
Student Agent Type	Tabular Q Learning	PPO	DDPG
Optimizer	NA	ADAM	ADAM
Batch size	NA	256	256
Learning rate	.5	.001	.001
Gamma	.99	.99	NA
Entropy coefficient/Epsilon	.01	.01	NA
Adam epsilon	NA	10^{-8}	10^{-3}
Clipping epsilon	NA	.2	NA
Maximum gradient norm	NA	.5	NA
GAE	NA	.95	NA
Value loss coefficient	NA	.5	NA
Polyak-averaging coefficient	NA	NA	.95
Action L2 norm coefficient	NA	NA	1
Scale of additive Gaussian noise	NA	NA	.2
Probability of HER experience replay	NA	NA	NA
Actor Network	NA	3 layers with 64 units each, Tanh activation	3 layers with 256 units each, ReLU activation
Critic Network	NA	3 layers with 64 units each, Tanh activation	3 layers with 256 units each, ReLU activation

Table 4.2: Student hyperparameters.

	Maze	Four Rooms	Fetch Reach
Student training iterations	100	50	50
# episodes/epochs per student training iteration	10	25	1
Cycles per epoch	NA	NA	6
Batches per cycle	NA	NA	5
Evaluation episodes/rollouts	30	40	80
Max # of environment steps	40	40	50
Performance Threshold	.77 (discounted return)	.6 (discounted return)	.9 (success rate)
# of teacher episodes	300	90	50

Table 4.3: Hyperparameters used in the teacher-student training procedure.

newly initialized student. Therefore, we can determine the impact of the teacher's learned curriculum on the student's learning efficiency and final performance on the target task

4.3 Baselines

To analyze the effectiveness of our reinforcement teaching framework, we compare it against the following baselines: L2T [3], Narvekar et al. (2017) [6], a random teacher policy, and a student learning the target task from scratch (no teacher). Narvekar et al. (2017) and L2T [3] are both RL teaching methods in which a teacher is trained, via RL, to adjust a component of the student learning process. In Narvekar et al. (2017), the RL teacher is used to learn an optimal curriculum policy for an RL student. Further, in the L2T framework, the teacher is tasked with learning how to sample minibatches of data for a supervised learning student.

Narvekar et al. (2017) are representative of approaches that use the parameter state representation and the time-to-threshold reward function, while the L2T framework is representative of domain-specific approaches that rely on handcrafted heuristics. In the L2T framework, the teacher's state representation includes several heuristics about the data and student model, and is heavily designed for the task of minibatch sampling. Therefore, to apply this method to the curriculum learning setting, we used an approximation of their state representation. This approximated state repre-

sentation includes a one hot vector of the teacher's action, the student's target task score, the student's source task score, and the current student episode number. Furthermore, for the reward function, the L2T framework considers a sparse variant of the time-to-threshold reward (see Table 4.4). Next, to highlight the importance of our learning progress reward function, we ablate over three additional reward functions used in the RL teaching literature. This includes Matiisen et al. (2017) [29], in which the teacher's reward is strictly the learning progress term (see Definition 3.1), and Ruiz et al. (2019) [11], in which the teacher's reward is based on the student's performance. See Table 4.4 for full details of the reward baselines. In these ablations, we fixed the teacher's state representation to be the PE-Action, and just changed the reward. This ensures any difference in the teacher's policy results from the reward function alone.

Baseline	**Reward Function**	**State Representation**
Ours	R $= -\mathbb{I}\left(m(\theta) < m^*\right) + LP(\theta', \theta)$	Behavior Embebber
Narvekar et al. (2017)	R $= -\mathbb{I}\left(m(\theta) < m^*\right)$	Student Parameters
Fan et al. (2018)	R $= 0$ except $R(T) = -\log(\frac{T}{max-steps})$	Heuristic-based
Matiisen et al. (2017)	R $= LP(\theta', \theta)$	NA
Ruiz et al. (2019)	R $= m(\theta')$	NA
Sparse Ruiz et al. (2019)	R $= 0$ except $R(T) = m(\theta')$	NA

Table 4.4: This table highlights the reward and state representation used across various baselines. For Fan et al. (2018) baseline, T denotes the last teacher-time-step in the teacher's episode, and max-steps is the maximum number of teacher time-steps.

4.4 Results

In this section, we first compare our reinforcement teaching method with the existing RL teaching baselines. Then, to emphasize the significance of our learning progress reward function, we show results ablating over several rewards functions. All results are averaged over 30 seeds with shaded regions indicating 95% confidence intervals (CI). To test differences between methods, we also report the mean area under the

student's learning curve (AUC) when trained using each of the teacher's learned curricula. We use a one-tailed independent-samples Welch t-test (i.e., equal variances are not assumed) to determine if there is a difference in the average AUC between methods with $\alpha = .05$. The Welch t-test was found to be more robust to violations of their assumptions compared to other parametric and non-parametric tests (e.g., t-test, ranked t-test, and Mann-Whitney non-parametric test) [53]. In certain results we found the normality assumption to be violated, therefore the Welch t-test a better choice than others.

4.4.1 Comparison of Existing Baselines

Across all environments, we found that by using the PE state representation and the LP reward signal together, the teacher is able to learn a superior curriculum policy compared to the baselines. These teacher policies generated a curriculum of start/goal states for the student that improved the student's learning efficiency and/or final performance, as shown in Figures 4.3 and 4.4.

More specifically, we found that in the Maze domain, the PE-QValues + LP teacher policy initially selected starting states close to the target goal state. However, as the student's skill set improved over time, the teacher adapted its policy and selected starting states farther away from the goal state (see Figure 4.5). This curriculum policy enabled the student to solve the Maze task (reach the performance threshold) in approximately 400 student training episodes. Comparatively, the L2T, random teacher, and learning from scratch baselines were unable to make any progress on the student's learning even after 10,000 student training episodes (see Figure 4.3). Our PE-Action + LP method and the Narvekar et al. (2017) baseline resulted in a similar speedup on behalf of the student's learning, however the student's average performance did not surpass the performance threshold. In addition, with our method (PE-QValues + LP) the teacher only selected "irrelevant" and "impossible" tasks 2.6% of the time. The Maze domain was designed to contain "irrelevant" and

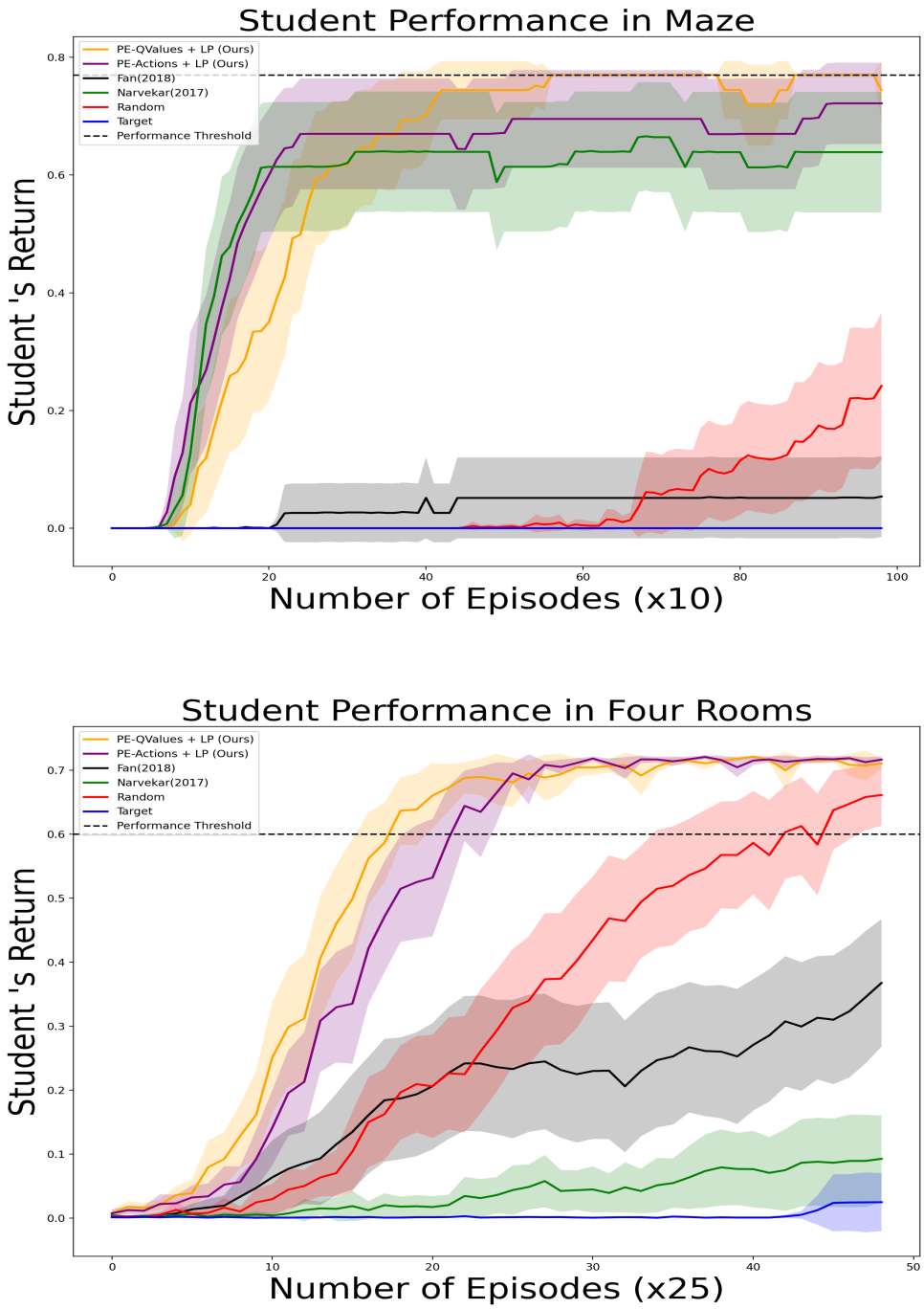


Figure 4.3: Top: Maze, Bottom: Four Rooms. This figure shows the student's learning curves on the target tasks with the assistance of the respective trained teacher policies. Purple/orange curves indicate our methods. Our method outputs a more effective teaching policy resulting in improved student final performance and/or learning efficiency across both environments. Dotted line indicates performance threshold m^*.

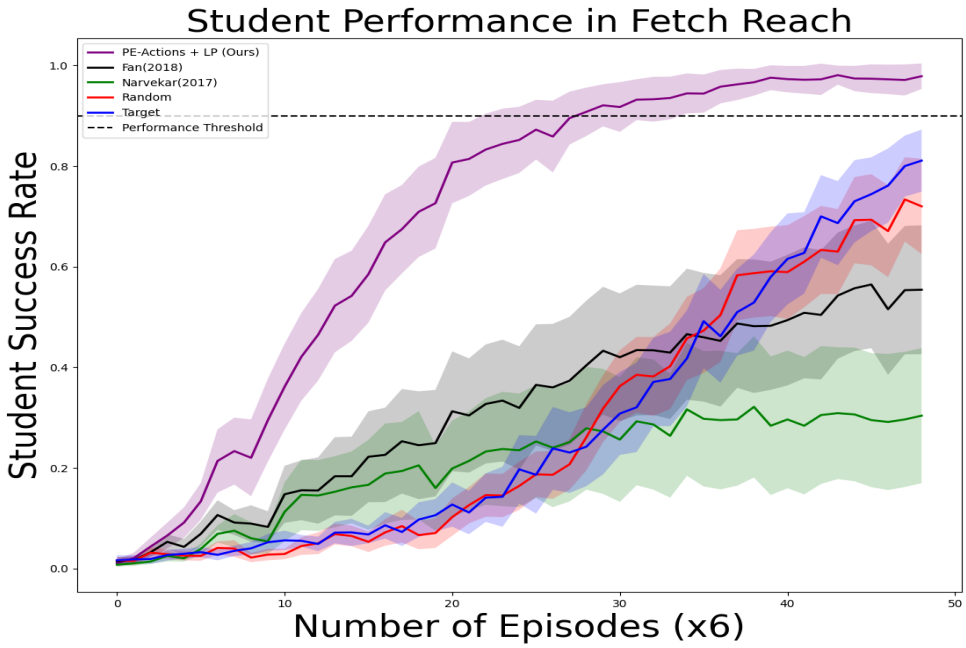


Figure 4.4: This figure shows the student's learning curve in Fetch Reach with the assistance of the respective trained teacher policies. Our method (purple curve) achieves a superior teaching policy resulting in improved student final performance compared to the baselines. Dotted line indicates performance threshold m^*.

	Maze	Four Rooms	Fetch Reach
PE-Actions + LP (Ours)	58.05 ± 3.61	23.95 ± 0.50	33.82 ± 1.06
PE-QValues + LP (Ours)	58.55 ± 1.48	25.60 ± 0.51	NA
L2T	3.44** ± 2.38	9.02** ± 1.79	15.49** ± 2.22
Narvekar et al. (2017)	54.3 ± 4.15	1.88**± 0.64	5.87** ± 1.52
Random curriculum	4.17** ± 1.36	14.86** ± 1.14	13.41** ± 1.06
Learning from stratch	0** ± 0	0.16**± 0.11	13.76** ± 1.13

Table 4.5: This table shows the student's average area under the curve when using the trained teachers' curriculum. ** Indicates a significant difference (p<.05) between baseline and both of our methods (PE QValues/Actions + LP reward).

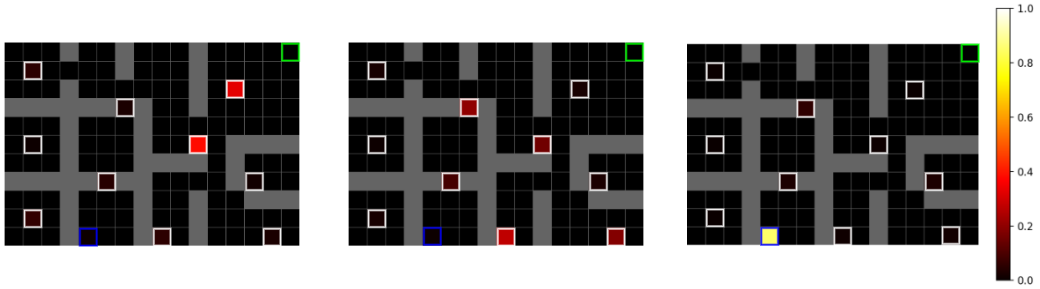


Figure 4.5: The beginning (left), middle (center), and ending (right) stages of the curriculum generated by the PE-QValues + LP method for the Maze environment. States outlined in white indicate possible teacher actions. States outlined in blue/-green indicate the target start and goal state respectively. Brighter colors (more yellow/white) indicates the start state was chosen more frequently by the teacher. Darker red/black indicates the start state was chosen less frequently by the teacher.

"impossible" tasks in order to test whether RL teachers can learn to avoid them. However, it is likely that in other environments the set of "irrelevant" or "impossible" tasks is unknown. Therefore, it is important that with our method, the RL teacher is able to learn to avoid these tasks in order to improve the student's learning ability.

Moreover, the Narvekar et al. (2017) curriculum policy only improved the student's performance in the Maze domain. This is not surprising because (1) as noted in Section 3.3.1, the student parameters are Markov and (2) the parameters in this domain are simply the student's tabular state-action value table. This tabular representation is small and does not come with the same issues as the parameters of a function approximator as described in Section 3.3.1.

Next, in the Four Rooms domain, by using the PE state representation and LP reward, the teacher's learned curriculum enables the student to surpass the performance threshold after only training for 500 episodes. The random teacher was the second best baseline in this domain. However, the random curriculum over doubled the amount of student training time, with the student finally reaching the performance threshold after approximately 1,125 student training episodes (see Figure 4.3).

As for the Narvekar et al. (2017) baseline, issues with the parameter state representations quickly arose in the Four Rooms domain. Although the parameter state representation is a Markov state, generalization in parameter-space is difficult even for this student's relatively small neural network (40k parameters). This resulted in the student's average return plateauing around .1 (see Figure 4.3).

Furthermore, while the L2T baseline uses a heuristic state representation that is much smaller than the parameter representation, it is only able to improve over the Narvekar et al. (2017) and target baselines in the Four Rooms domain. We hypothesize that the sparse reward function used in this method is a major contributing factor to its poor performance. In this method the teacher's reward is 0 except for a positive reward once the student's performance on the target task surpasses the performance threshold. With this reward formulation, the teacher is not able to easily recognize the impact of each of its actions on the student's performance. This can make it difficult for the teacher to learn an adequate policy.

Moreover, in both the Four Rooms and Maze environments, when learning from scratch (without the assistance of RL teachers) the student is unable to make any progress on the target task. This highlights the importance of our reinforcement teaching framework in the curriculum learning setting. There are environments, especially hard-exploration environments, in which learning from scratch is infeasible. Therefore, learning curriculum policies becomes essential to be able to solve this class of environments.

Lastly, in the Fetch Reach environment, all methods yielded similar student performance gains as in Four Rooms. Only our teaching method, using PE-Actions + LP, was able to output an effective curriculum that drastically improved the student's learning efficiency and final performance compared to all other baselines (see Figure 4.4). This highlights that our method is not limited to grid-worlds and can be helpful in more complex robotic environments.

To that end, by using the LP reward, the teacher can quickly identify actions

that improve the student's performance, even without having the student reach the performance threshold. Moreover, by using the PE state representation, the teacher is still able to learn about Θ via the student input-output behavior, without the issues of directly learning from Θ. For these reasons, with our method the teacher is able to learn a superior curriculum policy that enables the student to quickly solve various discrete and continuous environments.

4.4.2 Teacher Learning Efficiency

We have demonstrated that our reinforcement teaching approach can successfully train an RL teacher to learn effective curriculum policies. These curriculum policies can improve both the learning efficiency and final performance of RL students in discrete and continuous environments. However, a typical consequence of using RL to train a teacher is the additional training computation. In our method, there is both an inner RL training loop to train the student, and an outer RL training loop to train the teacher. Although this is true, we show that our method can greatly improve the teacher's learning efficiency and therefore reduce the overall amount of computation. In the Maze environment, by using either PE state representations with the LP reward, the teacher is able to plateau to its final average return in approximately 150 teacher episodes. More impressively, in Four Rooms and Fetch Reach, the teacher is able to plateau in approximately 40 and 25 teacher episodes respectively (see Figure 4.6).

However, one can still argue that the extra compute necessary to train the teacher can be used to train a student on the target task from scratch. It is critical to recall that there are environments, particularly hard exploration environments, where learning a task from scratch is not feasible without assistance. To further emphasize this point, in the Maze environment, we trained an RL student from scratch with the same amount of compute required to train the RL teacher with our approach. We defined compute as the total number of student episodes involved to train the teacher

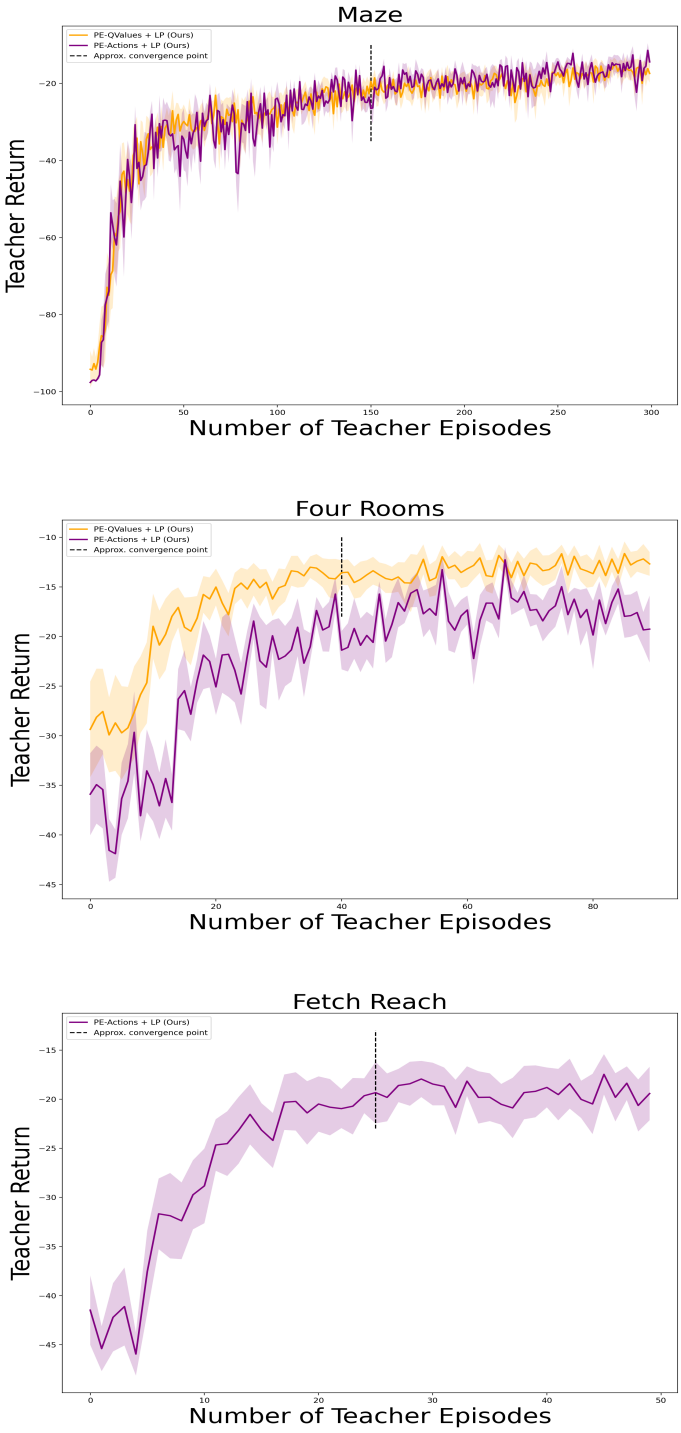


Figure 4.6: Top: Maze, Middle: Four Rooms, Bottom: Fetch Reach. This figure shows the teacher's training curves when using our method (PE state + LP reward) in each environment. We observe that with our method, the teacher can quickly plateau to its maximum cumulative return.

to plateau. More specifically, it took approximately 150 teacher episodes until the teacher reached its best cumulative return. For each teacher episode, the student had a maximum of 1,000 training episodes. Therefore, we trained the student from scratch on the target task for $1,000$ X 150 episodes. We found that even with the same amount of compute, the student is unable to learn the Maze task from scratch.

Moreover, with our framework, the RL agent is able to learn curriculum policies that can be useful for new student agents with differing learning rate, learning algorithm or neural network architecture (See Figure 4.7). In the transfer experiments, we used the same trained teacher as in 4.3, except we changed either the student's learning rate, learning algorithm, or neural network architecture. Therefore, the teacher is "seeing" a different student during its evaluation than it did during its training. In the Maze experiments, the teacher was originally trained with Q-Learning students that used a learning rate of .5. During the Maze transfer experiment, we evaluated the teacher's learned curriculum policy with either (1) SARSA students or (2) Q-Learning students with a learning rate of .0001 or .25. Moreover, in the Four Rooms transfer experiment, we evaluated the teacher's learned curriculum policy on PPO students that had a deeper neural network architecture (i.e., more neural network layers) than the PPO students encountered during the teacher's training.

In most cases, we found that the trained teacher is able to generalize to students it had not seen before and still output effective curriculum that resulted in improved student performance in terms of both final performance and learning efficiency. The teacher's generalization success is likely due to our parametric-behavior embedder. The parametric-behavior embedder learns a representation from the student's behavior, therefore as long as new students (i.e., students with different hyperparameters than the ones the teacher trained with) still behave similarly during the task, the teacher should be able to effectively generalize. Interestingly, in the Maze transfer experiments, in which the student's learning rate changed to 0.0001, we found that the teacher could not generalize when using the PE-QValues state representation.

This resulted in extremely poor student performance. This is not surprising because the PE-QValues state representation encodes the student's Q-Values. Stark changes to the learning rate can drastically change the resulting Q-Values, which can make it difficult for the teacher to generalize. Therefore, between the two PE variants, the PE-Actions state representation may be more robust to changing students. To that end, this highlights the importance of using our reinforcement teaching method to learn effective and robust curriculum policies.

4.4.3 Ablation of Reward Functions

To highlight the importance of our reward function on the teacher's learned policy, we ablate over various reward functions used in the literature. In this setting, we fix the state representation to be our PE representation. All results are averaged over 10 seeds with shaded regions indicating 95% confidence intervals (CI). See Table 4.3 for a complete list of the training hyperparameters and Appendix A for the teacher hyperparameters and the associated grid-searches.

To test differences between methods, we also report the mean area under the student's learning curve (AUC) when trained using each of the teacher's learned curricula. We use a one-tailed independent-samples Welch t-test (ie., equal variances are not assumed) to determine if there is a difference in the average AUC between methods with $\alpha = .05$. Across all environments, we found that the student achieves a comparable or higher average AUC value when trained with a teacher utilizing our LP reward (See Table 4.6).

More specifically, in both the Maze and Four Rooms environment, we found that using our LP reward function resulted in a significantly higher student AUC value compared to the Matiisen et al. (2017) baseline. Matiisen et al. (2017) uses a reward formulation based solely on the learning progress signal. This is unlike our LP reward function, which uses the learning progress signal as a shaping function. A limitation of the Matiisen et al. (2017) reward design is that it can easily be affected by the

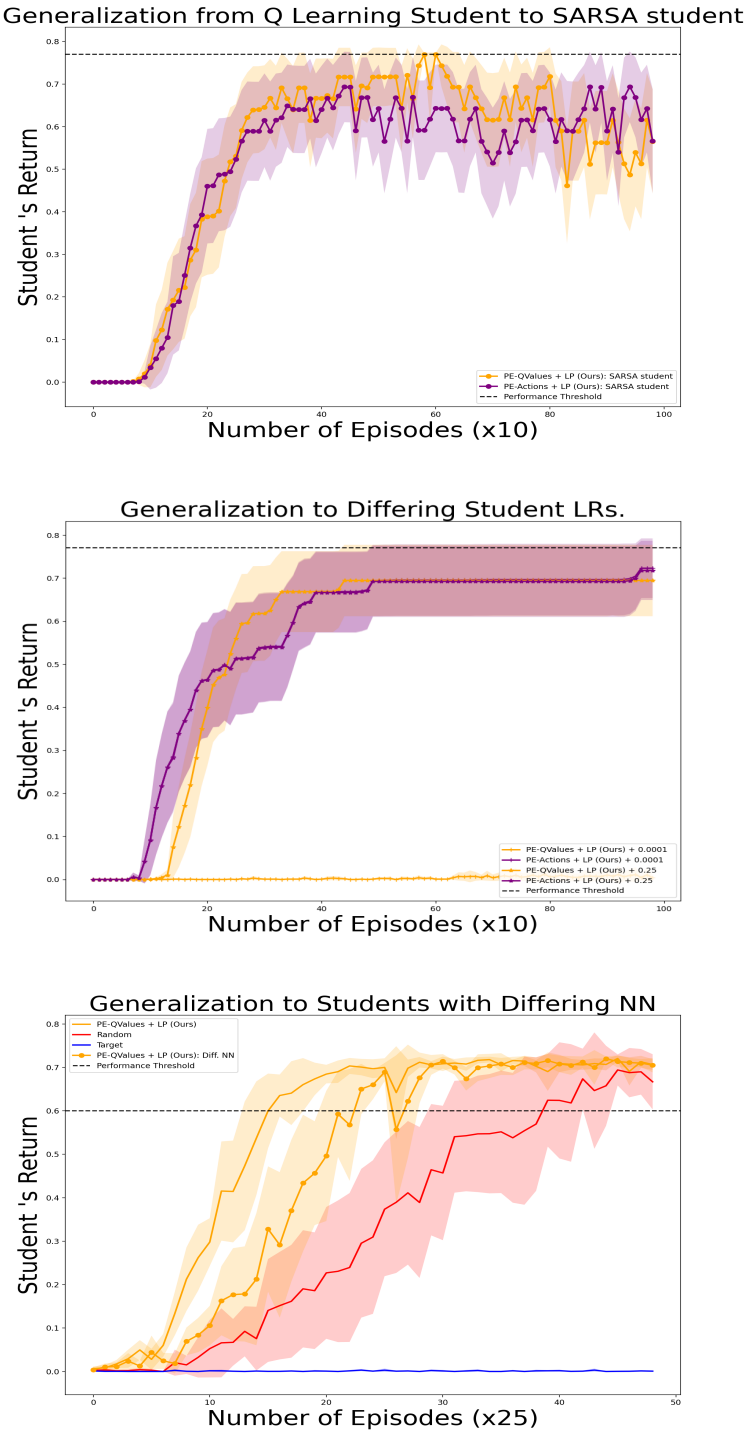


Figure 4.7: Top: Maxe environment, teacher is evaluated on students with different learning rates, Middle: Maze environment, teacher is evaluated on students with a different learning algorithm, Bottom: Four Rooms environment, teacher is evaluated on students with a different neural network architecture. All curves are averaged over 10 runs with shaded regions indicating 95% confidence intervals.

Reward Ablation			
	Maze	Four Rooms	Fetch Reach
LP (Ours)	58.55 ± 1.48	26.54 ± 0.46	34.55 ± 1.61
Matiisen et al. (2017) reward	10.79* ± 1.33	21.28* ± 1.36	34.39 ± 1.51
Time-to-threshold	45.11* ± 4.46	20.25* ± 2.65	16.52* ± 4.63
Fan et al. (2018) reward	9.89* ± 3.89	23.60 ± 2.43	14.18* +/- 3.20
Ruiz et al. (2019) reward	62.26 ± 1.25	27.15 ± 0.77	19.60* ± 3.66
Sparse variant of Ruiz et al. (2019)	61.16 ± 1.27	24.90 ± 1.26	22.63* ± 4.12

Table 4.6: Ablation of teacher reward functions with fixed PE state. Reporting mean area under the student's learning curve over 10 runs. * Indicates a significant difference ($p<.05$) between baseline and our method (PE state + LP reward).

size of the teacher's action space. For example, if the teacher's action set consists of several sub-tasks that may be "easy" but not necessary for the student to learn in order to improve learning on the target task. By using the learning progress signal alone, the teacher is always incentivized for proposing easier tasks first (i.e., tasks for which the student can make progress). Therefore, this can delay the teacher from selecting a more challenging but relevant task which is required for the student to learn the target task.

Moreover, in both the Maze and Four Rooms environment, there were several teacher actions that fell under the category of "easy" but not necessary." Therefore, it is not surprising that in these domains the teacher was unable to learn an adequate policy under the Matiisen et al. (2017) reward function. The Fetch Reach domain did not include unnecessary actions that can hinder the teacher's curriculum policy, therefore we found that the Matiisen et al. (2017) achieves a comparable policy to our LP reward.

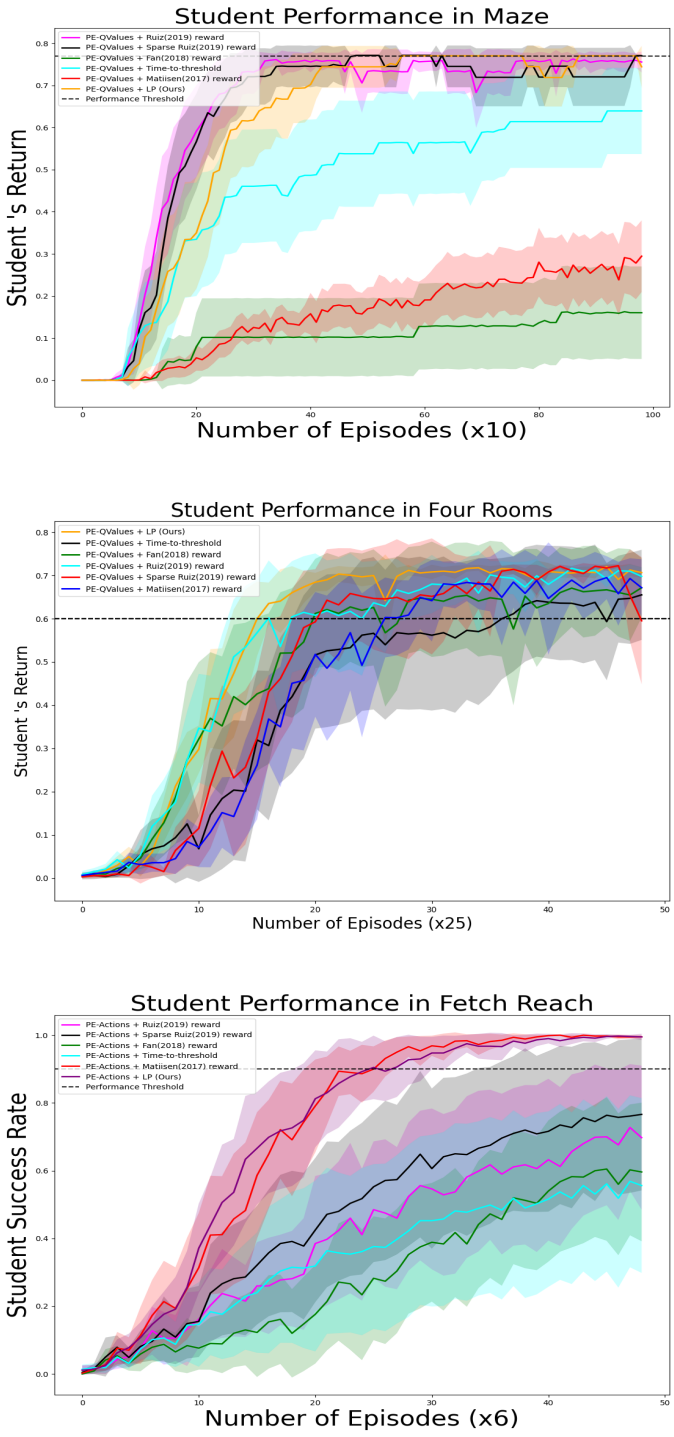


Figure 4.8: Top: Maze, Middle: Four Rooms, Bottom: Fetch Reach. Student learning curves on the target tasks with the assistance of the respective trained teacher policies. Purple/orange curves indicate our method using the LP reward function. By using the LP reward function, the teacher is able to learn a comparable or better teaching policy resulting in improved student final performance and/or learning efficiency across all environments. Dotted line indicates performance threshold m^*.

Furthermore, we found that using our LP reward function resulted in a significantly higher student AUC value compared to the Narvekar et al. (2017) and Fan et al. (2018) baselines in two out of three of the environments. In both of these reward formulations the teacher is rewarded for taking actions such that the student reaches a performance threshold as quickly as possible. Although simple approaches, the teacher is not able to quickly differentiate between the impact of different actions on the student's performance. In the Narvekar et al. (2017) formulation, each action has the same cost of -1, therefore it takes several teacher training episodes before the teacher can update its action-value network and learn which actions are better or worse for the student. A similar issue arises in the Fan et al. (2018) formulation.

Our LP reward function maintains the primary benefits from both the Matiisen et al. (2017) and Narvekar et al. (2017)/Fan et al. (2018) reward formulations. Similar to Matiisen et al. (2017), our LP reward function benefits from the learning progress term but does not fall into the trap of selecting irrelevant actions. Instead by using the learning progress term to shape the time-to-threshold reward, the teacher is encouraged to select actions that allow the student to solve the target task as quickly as possible.

Lastly, when comparing our LP reward function to the Ruiz et al. (2019) baselines, we found that our LP reward function resulted in a significantly higher student AUC value in the Fetch Reach environment. The reward function used in the Ruiz et al. (2019) baseline is based on the performance measure of the target task. Therefore, the teacher is rewarded for actions that result in the student's target task performance being high. We hypothesize that the main limitation of this reward function is that it only considers the student's performance measure on the target task. It does not take into account how well the student is performing on the intermediate sub-tasks. During the early stages of the student's learning process, the student may not be making progress on the target task. Assuming the target task is difficult, it can take several iterations learning on the "easier" sub-tasks before the student can make any

progress on the target task. This can result in a sparse reward formulation early on which can hinder the teacher's ability to learn which actions are promising for the student.

This confirms that the LP reward is critical for our reinforcement teaching method. To that end, we have successfully demonstrated that reinforcement teaching can be used to learn effective curricula that improve student learning.

Chapter 5

Conclusions and Future Work

We conclude this thesis by discussing limitations and future work.

5.1 Limitations and Future Work

In this thesis, we presented reinforcement teaching: a general formulation for meta-learning using reinforcement learning. We focused on one domain application of reinforcement teaching: curriculum learning. Under this framework, an RL teacher learns a curriculum policy that updates the tasks for an RL student. We have demonstrated that using the learning progress shaped reward function and the parametric-behavior embedder state representation enables the teacher to learn a superior policy most efficiently compared to several baselines. Although this thesis focused on curriculum learning, several other meta-learning problems can be formulated using Reinforcement Teaching (see Figure 1.1). For example, reinforcement teaching can be used to learn a step-size adaption policy that continually changes the step-size of the student [54]. It can also be used to learn a teaching policy to sample mini-batches of student training data.

Furthermore, some of the main limitations of reinforcement teaching are the limitations of current reinforcement learning algorithms. For example, we chose to use an episodic formulation in designing the reward function because RL algorithms currently struggle in the continuing setting with average reward [55]. In addition, RL

algorithms can have difficulty learning if the number of discrete actions is too large; therefore, we limited the size of the action set.

Moreover, there are several interesting extensions for the teacher's state representation. First, our parametric-behavior embedder learns a representation of the student's input-output behavior which in turn becomes the teacher's state representation. The use of the student's input-output behavior was a design choice for which we describe its benefits in Section 3.3.1. However, another possibility is to learn a representation directly from the raw student parameters, and use this learned parameter representation as the teacher's state. Although this may present challenges in terms of generalization and scalability, future work can empirically investigate the impact of learning the teacher's state representation in an end-to-end manner.

In addition, it is important to note that our parametric-behavior embedder does assume that the student has no memory. One example of the student having memory is if the student uses a recurrent neural network. If the student did have a memory, the student's input-output behavior would not fully capture the state of the student's learning process. Future work can consider ways to extend the parametric-behavior embedder to account for student memory.

Another limitation of the reinforcement teaching methodology is that it is focused on learning student-specific teaching policies. The RL teacher only learns from one type of student. An interesting extension of reinforcement teaching could consider the setting in which the teacher learns from multiple, diverse student agents. For example, consider Cliff World, a tabular Maze environment in which an RL agent must learn how to travel from a start state to a goal state without falling off the cliff. In such environment, the SARSA and Q-Learning algorithms learn two very different policies [47]. Therefore, it would be interesting to discover whether our RL teacher can learn to customize its teaching policies depending on the policy of the student agent.

Furthermore, reinforcement teaching has only considered the teacher-student rela-

tionship in which the teacher is an RL agent, and the student is a model/agent. An important next step could be to consider the setting in which the teacher is used to teach a human student, as is the case for cognitive tutors [22, 23].

5.2 Conclusion

In this thesis, we presented reinforcement teaching, a general formulation that allows a teacher to learn a policy to adjust components of the student's learning process. To facilitate the teacher's learning, we introduced a learning progress based reward function to allow the teacher to recognize promising actions more quickly. In addition, we developed the parametric-behavior embedder that learns a representation of the student's behavior. Our reward and state design does not rely on problem-specific heuristics and can therefore more easily generalize to different problem settings.

While reinforcement learning as a method for meta-learning has certain limitations, reinforcement teaching provides a unifying framework for the meta-learning problem formulation. As reinforcement learning algorithms improve, the set of meta-learning problems solvable by reinforcement teaching will continue to increase.

Bibliography

[1] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis Machine Intelligence*, no. 01, pp. 1–1, 5555, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3079209.

[2] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 113–123, 2019.

[3] Y. Fan, F. Tian, T. Qin, X.-Y. Li, and T.-Y. Liu, "Learning to teach," *International Conference on Learning Representations*, 2018.

[4] L. Wu *et al.*, "Learning to teach with dynamic loss functions," *Advances in Neural Information Processing Systems*, vol. 32, 2018.

[5] X. Zhang, Y. Ma, A. K. Singla, and X. Zhu, "Adaptive reward-poisoning attacks against reinforcement learning," in *ICML*, 2020.

[6] S. Narvekar, J. Sinapov, and P. Stone, "Autonomous task sequencing for customized curriculum design in reinforcement learning," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2536–2542. DOI: 10.24963/ijcai.2017/353.

[7] S. Narvekar and P. Stone, "Learning curriculum policies for reinforcement learning," *Proceedings of the 2019 International Conference on Autonomous Agents & Multiagent Systems*, 2019.

[8] C. Huang *et al.*, "Addressing the loss-metric mismatch with adaptive loss alignment," in *ICML*, 2019.

[9] D. Almeida, C. Winter, J. Tang, and W. Zaremba, "A generalizable approach to learning optimizers," *ArXiv*, vol. abs/2106.00958, 2021.

[10] F. M. Garcia and P. S. Thomas, "A meta-mdp approach to exploration for lifelong reinforcement learning," in *Advances in Neural Information Processing Systems*, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/c1b70d965ca504aa751ddb62ad69c63f-Paper.pdf.

[11] N. Ruiz, S. Schulter, and M. Chandraker, "Learning to simulate," in *International Conference on Learning Representations*, 2019.

[12] X. Zhu, A. K. Singla, S. Zilles, and A. N. Rafferty, "An overview of machine teaching," *ArXiv*, vol. abs/1801.05927, 2018.

[13] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, PMLR, 2017, pp. 1126–1135.

[14] Z. Xu, H. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 2402–2413.

[15] K. Javed and M. White, "Meta-learning representations for continual learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 1818–1828.

[16] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Mach. Learn.*, vol. 16, no. 3, 185–202, 1994, ISSN: 0885-6125. DOI: 10.1023/A:1022689125041. [Online]. Available: https://doi.org/10.1023/A:1022689125041.

[17] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvari, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine Learning*, vol. 38, pp. 287–308, 2004.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 1126–1135.

[19] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer, "Automatic curriculum learning for deep rl: A short survey," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, ser. IJCAI'20, Yokohama, Yokohama, Japan, 2021, ISBN: 9780999241165.

[20] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *J. Mach. Learn. Res.*, vol. 21, 181:1–181:50, 2020.

[21] D. S. Brown and S. Niekum, "Machine teaching for inverse reinforcement learning: Algorithms and applications," in *AAAI*, 2019.

[22] K. Georgila, M. G. Core, B. D. Nye, S. Karumbaiah, D. Auerbach, and M. Ram, "Using reinforcement learning to optimize the policies of an intelligent tutoring system for interpersonal skills training," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19, Montreal QC, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2019, 737–745, ISBN: 9781450363099.

[23] J. Bassen *et al.*, "Reinforcement learning for the adaptive scheduling of educational activities," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2020, 1–12, ISBN: 9781450367080. [Online]. Available: https://doi.org/10.1145/3313831.3376518.

[24] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, 1633–1685, 2009, ISSN: 1532-4435.

[25] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *J. Mach. Learn. Res.*, vol. 21, 181:1–181:50, 2020.

[26] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," *Conference on Robot Learning*, 2017.

[27] B. Ivanovic, J. Harrison, A. Sharma, M. Chen, and M. Pavone, "Barc: Backward reachability curriculum for robotic reinforcement learning," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 15–21, 2019.

[28] I. J. Goodfellow *et al.*, "Generative adversarial networks," *Advances in neural information processing systems*, 2014.

[29] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, "Teacher-student curriculum learning," *arXiv:1707.00183*, 2017. arXiv: 1707.00183 [cs.LG]. [Online]. Available: http://arxiv.org/abs/1707.00183v2.

[30] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer, "Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments," in *CoRL*, 2019.

[31] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[32] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, 2007. DOI: 10.1109/TEVC.2006.890271.

[33] D. Blank, D. Kumar, L. Meeden, and J. Marshall, "Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture," *Cybernetics & Systems*, Dec. 2003. DOI: 10.1080/01969720590897107.

[34] O. P.-Y. Moulin-Frier Clément Nguyen Sao Mai, "Self-organization of early vocal development in infants and machines: The role of intrinsic motivation," *Frontiers in Psychology*, 2014. DOI: 10.3389/fpsyg.2013.01006.

[35] B. Clement, D. Roy, P.-Y. Oudeyer, and M. Lopes, "Multi-armed bandits for intelligent tutoring systems," *Journal of Educational Data Mining*, vol. 7, no. 2, 20–48, 2015. DOI: 10.5281/zenodo.3554667.

[36] M. Dennis *et al.*, "Emergent complexity and zero-shot transfer via unsupervised environment design," *Advances in neural information processing systems*, vol. 34, 2020.

[37] A. Campero, R. Raileanu, H. Küttler, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette, "Learning with amigo: Adversarially motivated intrinsic goals," *International Conference on Learning Representations*, 2020.

[38] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9, PMLR, 2010, pp. 249–256.

[39] S. Mandt, M. D. Hoffman, and D. M. Blei, "Stochastic gradient descent as approximate bayesian inference," *J. Mach. Learn. Res.*, vol. 18, no. 1, 4873–4907, 2017, ISSN: 1532-4435.

[40] A. Dieuleveut, A. Durmus, and F. Bach, "Bridging the gap between constant step size stochastic gradient descent and markov chains," *The Annals of Statistics*, vol. 48, no. 3, pp. 1348–1382, 2020.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[42] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[43] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, 1999.

[44] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[45] M. Chevalier-Boisvert, L. Willems, and S. Pal, *Minimalistic gridworld environment for openai gym*, https://github.com/maximecb/gym-minigrid, 2018.

[46] M. Plappert *et al.*, *Multi-goal reinforcement learning: Challenging robotics environments and request for research*, 2018. DOI: 10.48550/ARXIV.1802.09464.

[47] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*, ser. Adaptive computation and machine learning. MIT Press, 2018, ISBN: 0262193981. [Online]. Available: http://www.worldcat.org/oclc/37293240.

[48] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017. arXiv: 1707.06347 [cs.LG]. [Online]. Available: http://arxiv.org/abs/1707.06347v2.

[49] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[50] L. Willems and K. Karra, *Pytorch actor-critic deep reinforcement learning algorithms: A2c and ppo*, 2020.

[51] P. Dhariwal *et al.*, *Openai baselines*, https://github.com/openai/baselines, 2017.

[52] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[53] C. Colas, O. Sigaud, and P.-Y. Oudeyer, *A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms*, 2019.

[54] A. Lewandowski, C. Muslimani, D. Schuurmans, M. E. Taylor, and J. Luo, *Reinforcement teaching*, 2022. DOI: 10.48550/ARXIV.2204.11897.

[55] Y. Wan, A. Naik, and R. S. Sutton, "Learning and planning in average-reward markov decision processes," in *International Conference on Machine Learning*, PMLR, 2021, pp. 10 653–10 662.

Appendix A: Hyperparameters

A.1 Teacher Hyperparameters

In the Maze experiments, for the DQN teacher, we performed a grid search over batch size $\in \{64, 128, 256\}$, learning rate $\in \{.001, .005\}$, and minibatch $\in \{75, 100\}$. Next, in the Four Rooms experiments, for the DQN teacher, we performed a grid search over batch size $\in \{128, 256\}$, and minibatch $\in \{75, 100\}$. We use a constant learning rate of .001. Lastly, in the Fetch Reach experiments, for the DQN teacher, we performed a grid search over batch size $\in \{128, 256\}$. We use a constant learning rate of .001 and mini-batch size of 200. The best hyperparameters for each of the baselines are reported in the tables.

	Hyperparameters used across all envs
Teacher agent type	DQN
Optimizer	ADAM
Gamma	.99
Tau	10^{-3}
Target network update frequency	1
Starting epsilon	.5
Epsilon decay rate	.99
Value network	2 layers with 128 units each, Relu activation

Table A.1: Fixed teacher hyperparameters used across all methods.

	Maze	
	PE-QValues x LP	PE-Actions x LP
Batch size	256	256
Learning rate	.005	.001
Minibatch size	100	100
	Four Rooms	
	PE-QValues x LP	PE-Actions x LP
Batch size	128	128
Learning rate	.001	.001
Minibatch size	100	100
	Fetch Reach	
	PE-QValues x LP	PE-Actions x LP
Batch size	NA	256
Learning rate	NA	.001
Minibatch size	NA	200

Table A.2: Teacher hyperparameters with our reinforcement teaching method (Parametric-behavior state and LP reward function).

Maze			
Baseline	**Batch size**	**Learning rate**	**Mini-batch size**
Narvekar et al. (2017)	128	.001	NA
L2T (Fan et al., 2018)	128	.005	NA
PE state x Time-to-threshold	256	.001	75
PE state x Fan(2018) reward	256	.001	100
PE state x Ruiz(2019) reward	128	.001	100
PE state x Sparse Ruiz(2019)	128	.001	75
PE state x Matiisen(2017)	256	.001	100
Four Rooms			
Baseline	**Batch size**	**Learning rate**	**Mini-batch size**
Narvekar et al. (2017)	128	.001	NA
L2T (Fan et al., 2018)	256	.001	NA
PE state x Time-to-threshold	256	.001	100
PE state x Fan(2018) reward	256	.001	100
PE state x Ruiz(2019) reward	128	.001	100
PE state x Sparse Ruiz(2019)	128	.001	75
PE state x Matiisen(2017)	256	.001	75
Fetch Reach			
Baseline	**Batch size**	**Learning rate**	**Mini-batch size**
Narvekar et al. (2017)	128	.001	NA
L2T (Fan et al., 2018)	256	.001	NA
PE state x Time-to-threshold	256	.001	200
PE state x Fan(2018) reward	256	.001	200
PE state x Ruiz(2019) reward	128	.001	200
PE state x Sparse Ruiz(2019)	128	.001	200
PE state x Matiisen(2017)	128	.001	200

Table A.3: Teacher agent hyperparameters for the baselines.