KNOW THY STUDENT: INTERACTIVE LEARNING WITH GAUSSIAN PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning often involves interaction between multiple agents. Human teacher-student settings best illustrate how interactions result in efficient knowledge passing where the teacher constructs a curriculum based on their students' abilities. Prior work in machine teaching studies how the teacher should construct optimal teaching datasets assuming the teacher knows everything about the student. However, in the real world, the teacher doesn't have complete information and must probe before teaching. Our work proposes a simple probing algorithm which uses Gaussian processes for inferring student-related information, before constructing a teaching dataset. We study this in two knowledge settings where the student is a tabula rasa or has partial knowledge of the domain. We study this in the ridge regression, support vector machines and offline reinforcement learning domains. Our experiments highlight the importance of probing before teaching, demonstrate how students can learn much more efficiently with the help of an interactive teacher, and outline where probing combined with machine teaching would be more desirable than passive learning.

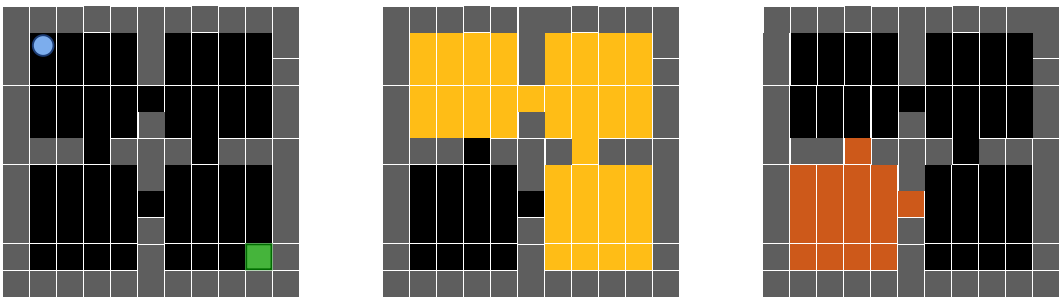
1 INTRODUCTION

In natural systems, learning often involves interaction between multiple agents. Teacher-student interactions best illustrate how interactions result in efficient knowledge passing. One way teachers interact with students is through concept tests—short, targeted *tests* administered at the start of a class to help teachers gauge which concepts students have understood (Caceffo et al., 2016; Treagust, 1988; Adams & Wieman, 2011). By first understanding what their students struggle on, teachers can adapt the curriculum to best address the needs of their class and avoid repeating topics their students have already mastered. In other words, teachers construct "test sets" to inform them how they should later construct "training sets" for their students.

We take inspiration from this form of teacher-student interactions in reframing how we'd like to efficiently train and test machine learning models. As ML practitioners and researchers, we implicitly interact with our model learners in this way to diagnose the learner. For example, we evaluate our models by analyzing how well it performs on a test set. Receiving feedback on the test set from the model informs us about where the model fails and the model's training dataset (eg. properties like class imbalance (Horn et al., 2018; Rahman & Davis, 2013)). This feedback can then suggest remedies for how to retrain our model (eg. adding more examples of minority classes to balance the dataset (Hernandez et al., 2013)). The collective learning that takes place between a teacher and a student inspires our work to design principled methods for constructing training and test datasets based on feedback from the student.

We focus on the problem setup where the teacher must first probe the student, then construct a teaching dataset. The performance of the teacher is based on how well the student performs on a held-out test dataset. We build on prior work in machine teaching (Zhu et al., 2018; Liu et al., 2016; Cakmak & Lopes, 2012) for constructing optimal teaching datasets. We use Gaussian Processes (Williams & Rasmussen, 2006) as a tractable means for inferring student parameters when probing. We combine machine teaching with probing, and apply this framework to linear models for ridge regression, support vector machines (SVMs) and offline reinforcement learning settings.

We study two student knowledge settings. One is the *tabula rasa setting* where the student has no knowledge of the domain. We study this setting in the ridge regression and SVM domain where



(a) (b) (c)

Figure 1: Teacher needs to figure out what the student knows vs. doesn't know. (a) depicts an example of a navigation environment where the student (blue, upper left) has to navigate to a goal location (green, lower right). (b) illustrates that in some cases, the student has partial knowledge of the environment; from any of the yellow-highlighted cells, the student knows how to navigate to the green cell. (c) highlights the cells from which the student does not know how to navigate to the goal. The ideal teacher is one that can identify these states and construct examples from these states.

the teacher probes the student to determine how a student learns before constructing a dataset. The second setting is where the student has *partial knowledge* of the domain, eg. it has already explored and mastered parts of the environment. We study this in the offline reinforcement learning (RL) domain where the teacher probes to determine what the student already knows.

In summary, our work's contributions are the following:

- We show that the teacher must probe the student, otherwise the student can have arbitrarily bad performance.
- We show that if the teacher does probe, the learner can learn from much fewer training examples than the teacher. In the RL setting, this can lead up to $2 - 16\text{k}\times$ reduction in training samples used.
- We show that in certain cases (such as when sampling from your training distribution is expensive), machine teaching is a way to recover the optimal model parameters which passive learning cannot do.

2 Related works

Machine teaching The machine teaching problem is to determine an optimal training set, given knowledge about the student algorithm (eg. ridge regression) and the optimal model parameters θ^* (Zhu et al., 2018; Zhu, 2015; Simard et al., 2017). An *optimal* dataset is defined by the cardinality of the dataset, also known as the teaching dimension (Goldman et al.). Machine teaching has been broadly applied to settings like inverse reinforcement learning (Cakmak & Lopes, 2012; Brown & Niekum, 2019; Hadfield-Menell et al., 2016) where the goal is to construct a small demonstration dataset for the learner to most confidently learn the underlying reward function. Our work builds on results from Liu et al. (2016) which derive the teaching dimension of linear learners in settings including ridge regression and SVMs.

Imitation learning The imitation learning problem looks at how a student should learn a policy from an expert demonstration dataset, either by inferring the underlying reward that produced the behavior (inverse reinforcement learning, IRL) (Abbeel & Ng, 2004) or directly learning the policy (behavioral cloning, BC) (Torabi et al., 2018). Our work is close to the interactive imitation learning setting Ross et al. (2011) where we rely on a teacher to provide action labels to states. The key difference is that we rely on a teacher to first probe and determine which states it should provide labels to; these might be states the student has never visited before.

Algorithm 1 Diagnosing student model

1: Initialize priors $p(z|H = \varnothing)$.
2: **for** $t = 1...T$ probing iterations **do**
3: Sample a student latent $z_t \sim p(z|H_t)$
4: *Optional*: Construct a train dataset $D_{\text{train}} = g(z_t)$ and pass to learner, $\theta = A(D_{\text{train}})$.
5: Construct a test probe x_{test} and get feedback from learner, $f(\theta, x_{\text{test}})$.
6: Use feedback to update beliefs $p(z|H)$
7: **Output** $p(z|H)$

Active learning The active learning problem looks at how a student should query a teacher for labels on unlabelled examples (Settles, 2009). Active learning is different from machine teaching in that the learners choose the datapoints they want to be trained on, rather than being guided by a teacher. However, the active learning framework might not be the right framework in settings where the learner doesn't have access to examples its never seen before. For example, DAgger (Ross et al., 2011) limits interaction between the learner and the teacher to the states the learner visits. This doesn't fully leverage what the learner *could* learn from the teacher, such as the teacher's privileged access to states that are important for a task of interest.

3 Interactive learning with Gaussian Processes

We study the problem of the teacher needing to infer what it doesn't know about the student before constructing a training dataset for the student. We use a Gaussian Process with a radial-basis function kernel for inference; alternative inference methods can be used as well. In our settings, the teacher must either infer something about the student's learning algorithm (eg. a hyperparameter) or the student's knowledge base (eg. its original training dataset).

Notation We denote $\mathcal{X}$ as the input space and $\mathcal{Y}$ as the output space. $f_\theta : \mathcal{X} \to \mathcal{Y}$ is the student model parameterized by model weights $\theta \in \Theta$. We refer to D as a dataset, and $\mathcal{D}$ as the space of datasets. A student runs an algorithm $A : \mathcal{D} \to \Theta$.

Our method for inference is simple:

1. **Sample hypothesis**: We sample a hypothesis z from our posterior distribution $p(z|H)$ where H is the history set of inputs and values given to the process.
2. **Train the learner**: We construct a train dataset based on the sampled hypothesis, $D_{\text{train}}(z) = \{(x_{\text{train}}, y_{\text{train}})\}$. We pass the dataset to the learner for it to run its learning algorithm A and learn initial model parameters, $\theta = A(D_{\text{train}})$. *This is only applicable in the tabula rasa setting where the learner hasn't been trained before.*
3. **Probe for feedback**: We construct a test probe $g(z) = x_{\text{test}}$ and send it to the learner. We receive feedback from the learner on that probe, $\texttt{feedback}(f_\theta(x_{\text{test}}))$.
4. **Update beliefs**: We use the feedback to update our beliefs, $p(z|H)$.

This is summarized in Algorithm 1. After a few iterations, the teacher uses $p(z|H)$ to then construct a training dataset for the student. We apply this framework to the ridge regression, SVMs, and offline RL domains. What the teacher needs to infer in each domain varies, however the underlying mechanism is the same as outlined above.

After probing the student, the teacher constructs the optimal teaching dataset based on the resulting $p(z|H)$. The optimal teaching dataset and teaching dimension lower bound have been studied in Liu et al. (2016) for ridge regression and SVM with linear models. This means that if the teacher *did* have complete information about its student (ie. knowing the student's regularization parameter λ), the teacher can construct the optimal teaching dataset D^*_{teach}. In later sections, we discuss how to exactly construct the optimal dataset in both settings. These domains illustrate how teaching and probing co-occur in a realistic setting where agents only have partial information about each other. Below we discuss each of the domains in more detail and what the teacher needs to infer in each domain.

3.1 RIDGE REGRESSION

A linear model student running ridge regression (without a bias) minimizes the following objective:

$$l(\mathbf{w}) = \sum_{i=1}^{N} (y_i - \mathbf{x}_i \mathbf{w})^2 + \lambda \frac{\|\mathbf{w}\|^2}{2}, \tag{1}$$

where $\mathbf{w}$ are the linear model weights, $\mathbf{x}$ is the input, y is the target, and λ is the regularizer. We denote the optimal model weights to be $\mathbf{w}^*$. From Liu et al. (2016), the optimal teaching dataset consists of a single datapoint and is a function of the learner's regularizer λ,

$$D(\lambda) = \{(x_{\text{teach}}, y_{\text{teach}})\} = \{(a\mathbf{w}^*, \frac{\lambda + \|\mathbf{x}_{\text{t}}\|^2}{a})\}, \tag{2}$$

where a is any nonzero real number. The inhomogeneous ridge regression case (with a bias) is also discussed in Liu et al. (2016). We omit that setting as we use ridge regression as an illustrative example for optimal teaching with probing. However, our analysis is easily extendable to that setting.

To simulate a teacher not having complete information about the student, we assume that the teacher does not know the learner's regularization parameter $\hat{\lambda}^*$. This simulates a teacher not having complete information about *how* a student learns from data.

Concretely, the inference steps outlined in Section 3 are:

1. We sample a hypothesis for $\hat{\lambda} \sim p(\lambda|H)$.
2. We construct a training dataset $D(\hat{\lambda}) = (x_{\text{train}}, y_{\text{train}})$ based on the optimal teaching dataset outlined in Equation 2.
3. We construct a test probe $g(\hat{\lambda}) = x_{\text{test}}$ for the learner and the learner sends back its prediction y. [1] We simulate what a learner with $\lambda = \hat{\lambda}$ would have responded and calculate the differences in their responses, ie. $(f_{A(D(\hat{\lambda}))}(x_{\text{test}}) - f_{A(D(\lambda^*))}(x_{\text{test}}))^2$.
4. We use the difference as the target function for the Gaussian Process. It is the feedback used to update our beliefs over λ.

3.2 SUPPORT VECTOR MACHINES (SVM)

A linear SVM student minimizes the following objective:

$$l(\mathbf{w}) = \sum_{i=1}^{N} \max(1 - y_i \mathbf{x}_i \mathbf{w}, 0) + \lambda \frac{\|\mathbf{w}\|^2}{2} \tag{3}$$

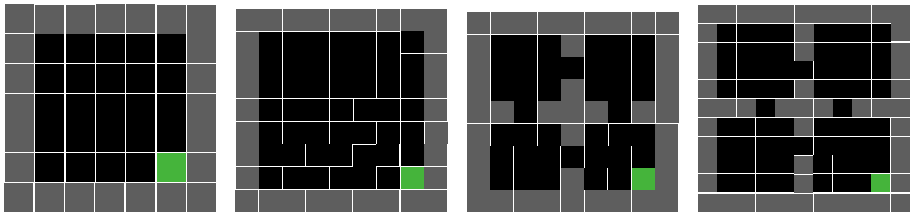
From Liu et al. (2016), the optimal teaching dataset consists of N identical training items where $N = \lceil \lambda \|\mathbf{w}^*\|^2 \rceil$. The teaching dataset is

$$D(\lambda) = \{(x_i, y_i)\}_{i=1}^{N} = \{(\frac{\lambda \mathbf{w}^*}{\lceil \lambda \|\mathbf{w}^*\|^2 \rceil}, 1)\}_{i=1}^{N} \tag{4}$$

Similar to ridge regression, the teacher does not know the learner's regularization parameter $\hat{\lambda}^*$ and must infer this through interaction. Concretely, the inference steps are:

1. We sample a hypothesis for $\hat{\lambda} \sim p(\lambda|H)$.
2. We construct a training dataset $D(\lambda) = (x_{\text{train}}, y_{\text{train}})$ based on the optimal teaching dataset outlined in Equation 4 for SVMs.

[1] Empirically, we found that both randomly sampled probes or a training point from Step 2 allow for fast inference.



(a) EmptyRoom 7x7 (b) EmptyRoom 9x9 (c) 4 Rooms 9x9 (d) 4 Rooms 11x11

Figure 2: Environments used for the RL setting. The learner can be initialized anywhere in the environment. Its goal is to navigate to the green grid cell.

3. We construct a random test probe and pass it to the learner. The SVM learner sends back the distance of that point from its separating hyperplane. [2] We simulate what a learner with $\lambda = \hat{\lambda}$ would have responded and calculate the differences in their responses, ie. $(f_{A(D(\hat{\lambda}))}(x_{\text{test}}) - f_{A(D(\lambda^*))}(x_{\text{test}}))^2$.
4. We use the difference as feedback to update our beliefs over λ.

3.3 Offline reinforcement learning

A offline RL student updates its Q-values given a dataset of trajectories $D = \{\tau_i\}_{i=1}^N$ where $\tau = [s_0, a_0, r_0, ..., s_{T-1}, a_{T-1}, r_{T-1}, s_T]$ is a trajectory of states and actions: $Q(s_t, a_t) = r_t + \gamma * V(s_{t+1}) \forall t = 0...T-1$ [3]

In this domain, we assume the student can be tabula rasa or partially trained. An example is given in Figure 1. To the best of our knowledge, there isn't prior work in studying the teaching dimension problem in the offline RL setting. Nonetheless, intuitively we would like the teacher to propose trajectories that maximize coverage over the environment and minimizes overlaps with states the learner has already seen. The teacher is tasked to infer which states the learner has seen in order to provide demonstrations that are novel to the learner.

The inference steps follow as,

1. We sample a state $s \sim p(s|H)$.
2. We do *not* construct a training dataset in the BC setting.
3. We pass the sampled state s to the learner and the learner sends back the action they would take, $a_{\text{learner}} = \arg\max_a Q_{\text{learner}}(s, a)$. The feedback is a binary function of whether the learner takes the same action as the teacher would, $\mathbb{1}[a_{\text{learner}} = a_{\text{teacher}}]$.
4. We use the difference as feedback to update our beliefs over $\mathcal{S}$.

An open question that we hope to later pursue is the choice of target function (Step 3).

4 Experiments

We now evaluate the utility of understanding the learner before teaching and compare different teaching methods against passive learning (learning without a teacher and just from the original training dataset). In particular, we try to answer the following research questions (**RQ**s):

RQ1: How important is it to understand the learner when teaching?

RQ2: When would you prefer teaching with probing over passive learning?

[2] We also tried receiving class predictions from the learner but found this to be an extremely sparse feedback signal. There might be smart ways to go about this, such as considering approaching for optimal probing. Related works in this direction include optimal experiment design (Pronzato, 2008; Foster et al., 2019).

[3] If function approximators are used, then this turns into fitting a parametric model π_θ where we represent the policy π in terms of a soft Q function Q_θ similar to Reddy et al. (2019).

(a) MSE (1D) (b) MSE (2D) (c) MSE (3D)

Figure 3: Mean squared error on held out data. Proper diagnosis of the student matters as the hypothesis space over possible student models grow and when the number of feature dimensions grows.

We examine this in three settings: Ridge regression, classification with support vector machines and offline RL in sequential decision making settings. The ridge regression and classification settings are meant to be simple, illustrative examples of how probing would be combined with optimal teaching, when the optimal teaching dataset can be determined with full knowledge of the student Liu et al. (2016). The regression and two-way separable classification data are generated from Pedregosa et al. (2011) without added noise. We use the MiniGrid environments (Chevalier-Boisvert et al., 2018) for the RL setting; examples of the environments are shown in Figure 2.

4.1 Ridge Regression

We compare the accuracy and sample efficiency (number of training examples) across three methods.

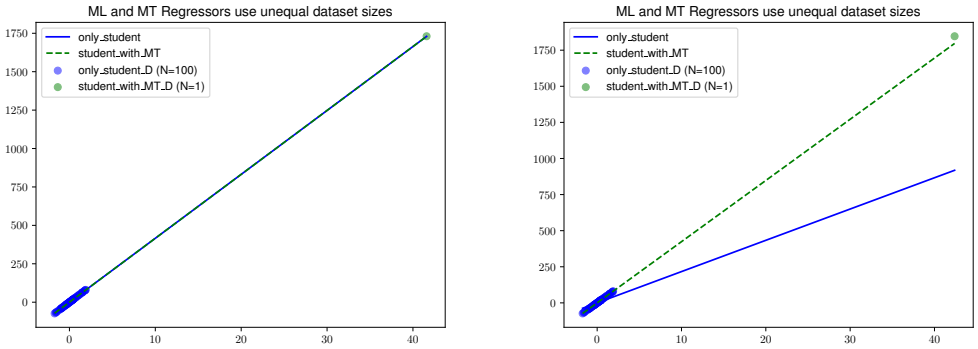
1. Passive learning (`PL`): Train a ridge regression learner on a fixed training dataset of size N. Here, the learner doesn't rely on a teacher.
2. Random teaching (`RT`): The teacher randomly samples a λ and constructs a training dataset $D(\lambda)$ based on Equation 2.
3. Interactive teaching (`IT`): The teacher runs Algorithm 1and takes the mean of $p(\lambda|H)$ as $\hat{\lambda}$. In our experiments, we allow the teacher to probe the student 5 times, ie. $T = 5$ in Algorithm 1. It constructs the training dataset as $D(\hat{\lambda})$.

Figure 3 compares `RT` and `IT` to answer **RQ1**: How important is it for the teacher to learn about the student before teaching? We compare a teacher that randomly samples across an increasing hypothesis space for λ compared to a teacher that probes the student. We see that randomly guessing information about the student can lead to arbitrarily bad performance for the learner; these observations hold also across increasing feature dimension. With only 5 probes, `IT` allows the student to predict with minimal error. Thus, to answer **RQ1**: It's important for the teacher to learn about the student, otherwise the student can perform poorly under larger belief spaces and feature spaces.

4.2 Support Vector Machine

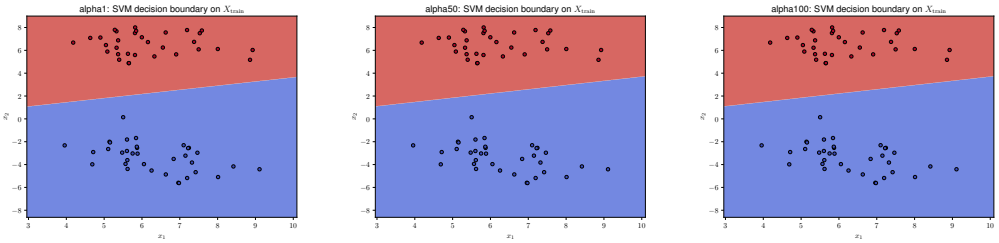
Figure 4 compares `PL` and `IT` to answer **RQ2**: When would you prefer teaching with probing over passive learning? With a fixed training dataset size $N = 100$, we vary the learner's regularization parameter from $\lambda = 1.0$ to $\lambda = 50.0$. We see that with particular learners (ie. low regularization), both methods do comparably in Figure 5a. However, with other learners (ie. high regularization), `PL` is not able to recover the best model. An interactive teacher, on the other hand, does enable the student to recover the best model. To answer **RQ2**: These results suggest teaching is preferable in low-data settings where sampling from the training distribution is expensive; the setting doesn't easily allow for more data to remedy the effects of high regularization or peculiarities in the student's algorithm.

We conducted the same experiments as in ridge regression for the two-way separable classification with SVMs. However, we found separable classification settings with SVMs unable to yield dif-



(a) $\lambda = 1.0$ (b) $\lambda = 50.0$

Figure 4: The learner trained on the teaching dataset (green) always recovers θ^* where as this is not always the case for the passive learner (blue).



(a) $\lambda = 1.0$ (b) $\lambda = 50.0$ (c) $\lambda = 100.0$

Figure 5: Inferring correct parameter in SVM is not as important to do well.

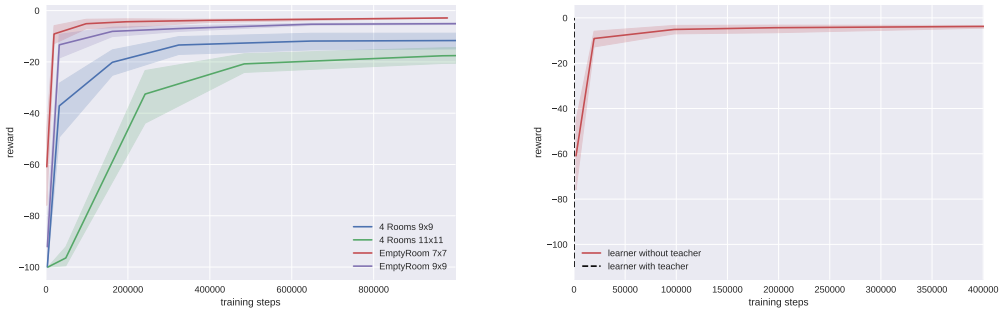
ferences among the three methods: all three methods learn to separate the data within a reasonable range over λ. Figure 5 provides intuition into why the three methods perform similarly on the classification domain. The separating hyperplane is not sensitive to increasing λ values. Therefore, when the classes are easy to separate, learners from each method will classify the points the same way.

4.3 Offline Reinforcement Learning

There are four environments we use for the RL setting; examples of the environments are shown in Figure 2. The learner can be initialized anywhere in the environment. The learner's goal is to navigate to the green cell. The reward for this setting is sparse: The learner receives -1 with every step except for when it reaches the goal where it receives 0 as its reward. The learner is evaluated based on how well it's able to navigate to the goal from any initial state. The teacher is first trained by running ϵ-greedy Q-learning with $\epsilon = 0.3$. for 5000 episodes, each episode with 100 steps.

We assume different levels of student knowledge: Either the student has no experience (it has never interacted with the environment), it knows one optimal path, it knows a few optimal paths, or it knows all optimal paths to the goal. An example is illustrated in Table 1 on the left-hand column. The goal of the teacher is to then infer which states the student can perform well in, and ideally construct a teaching dataset of trajectories where the student would perform poorly.

We compare four different methods. One is Passive learning (`PL`) where the passive learner trains without a teacher. Two is Random teaching (`RT`) where the teacher randomly samples a state from the environment and executes its policy until it reaches the goal; this is the trajectory that is sent over per sample. Three is Random probing (`RP`) where the teacher randomly samples a state for probing when approximating the target function for the Gaussian process. It then picks the states with the



(a) (b)

Figure 6: Comparing the number of training samples needed for a learner with vs. without a teacher. (a) shows the number of samples needed a learner without a teacher across environments. (b) compares the number of samples needed by a learner with (red curve) and without a teacher (black horizontal line $x \approx 50$) on the EmptyRoom 7x7 environment. With the help of a teacher sending demonstrations, the learner can learn much more quickly.

highest expected score as the initial states for running its policy and collecting trajectories. Four is Interactive teaching (`IT`) where the teacher samples a state with the highest expected information gain for probing and approximating the target function. Similarly to `RP`, it picks the states with the highest expected score as the initial states for collecting trajectories.

To answer **RQ1**, we compare against `RT` in Table 1 for EmptyRoom 7x7; results for the other environments can be found in the appendix. The first takeaway is that probing methods like `RP` and `IT` both teach better than `RT` (middle column). Additionally, teachers that probe students avoid sending repeated states which the learner has already seen before (right-hand column). Interestingly, `RP` tends to perform better or comparably to `IT`. This suggests that that the target function we use (action matching) might not be ideal for determining states to probe a student with.

To answer **RQ2**, we compare `PL` with `IT` in Figure 6. Noticeably, the student can benefit a lot from having a teacher provide demonstrations as it's able to learn from much fewer samples and still achieve high performance. Thus, a benefit to having a teacher is avoiding inefficient exploration.

5 Conclusion and Future Work

Our work showcases three simple settings which highlight the utility of having a teacher first learn about its student before teaching. We believe this form of collective learning—algorithmically constructing test sets for the learner in order to construct model-tailored training—can pave principled ways for diagnosing models and proposing remedies.

There are several interesting questions which we hope to later explore. These include,

- What is an optimal target function (form of feedback) to use for the Gaussian Process? The offline RL experiments suggest that we should revisit this choice.
- Under what conditions would you prefer machine teaching over passive learning? We'd like to formalize this as a function of the training set and problem difficulty. Under which domains (or types of domains) would machine teaching be useful for?
- How should we sample from a desirable data distribution given what we know about the student?

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Twenty-first international conference on Machine learning - ICML '04*, pp. 1, Banff, Alberta,

Canada, 2004. ACM Press. doi: 10.1145/1015330.1015430. URL http://portal.acm.org/citation.cfm?doid=1015330.1015430.

Wendy K. Adams and Carl E. Wieman. Development and validation of instruments to measure learning of expert-like thinking. *International Journal of Science Education*, 33:1289 – 1312, 2011.

Daniel S. Brown and Scott Niekum. Machine Teaching for Inverse Reinforcement Learning: Algorithms and Applications. *arXiv:1805.07687 [cs, stat]*, August 2019. URL http://arxiv.org/abs/1805.07687. arXiv: 1805.07687.

Ricardo Caceffo, Steve Wolfman, Kellogg S. Booth, and Rodolfo Azevedo. Developing a computer science concept inventory for introductory programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pp. 364–369, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450336857. doi: 10.1145/2839509.2844559. URL https://doi.org/10.1145/2839509.2844559.

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

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Adam Foster, Martin Jankowiak, Eli Bingham, Paul Horsfall, Yee Whye Teh, Tom Rainforth, and Noah Goodman. Variational bayesian optimal experimental design. *arXiv preprint arXiv:1903.05480*, 2019.

Sally A Goldman, Michael J Kearns, T Bell Laboratories, and Murray Hill. On the Complexity of Teaching. pp. 29.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. Cooperative Inverse Reinforcement Learning. *arXiv:1606.03137 [cs]*, November 2016. URL http://arxiv.org/abs/1606.03137. arXiv: 1606.03137.

Julio Hernandez, Jesús Ariel Carrasco-Ochoa, and José Francisco Martínez-Trinidad. An empirical study of oversampling and undersampling for instance selection methods on imbalance datasets. In *Iberoamerican Congress on Pattern Recognition*, pp. 262–269. Springer, 2013.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset, 2018.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Luc Pronzato. Optimal experimental design and some related control problems. *Automatica*, 44(2): 303–325, 2008.

M Mostafizur Rahman and Darryl N Davis. Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, 3(2):224, 2013.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations*, 2019.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *arXiv:1011.0686 [cs, stat]*, March 2011. URL http://arxiv.org/abs/1011.0686. arXiv: 1011.0686.

Burr Settles. Active learning literature survey. 2009.

Patrice Y. Simard, Saleema Amershi, David M. Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Christopher Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, and John Wernsing. Machine Teaching: A New Paradigm for Building Machine Learning Systems. *arXiv:1707.06742 [cs, stat]*, August 2017. URL `http://arxiv.org/abs/1707.06742`. arXiv: 1707.06742.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

David F. Treagust. Development and use of diagnostic tests to evaluate students' misconceptions in science. *International Journal of Science Education*, 10:159–169, 1988.

Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

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

Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018.

A APPENDIX

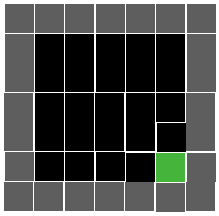
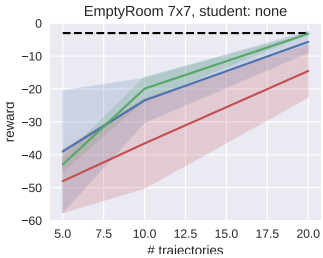
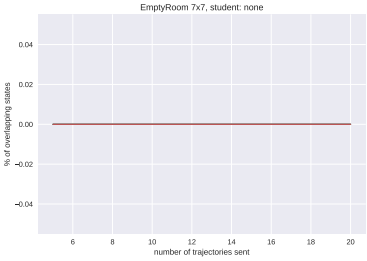
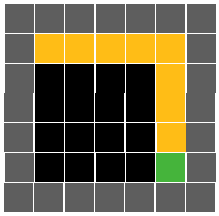
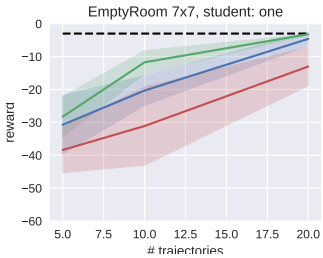
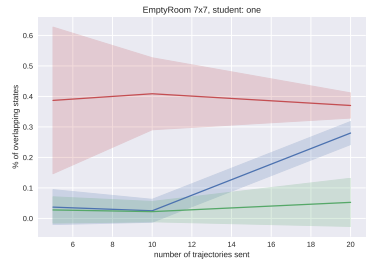
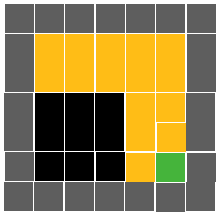
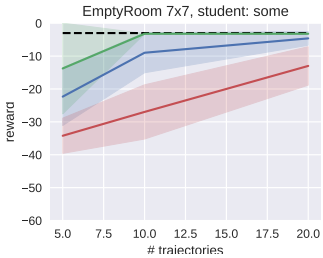
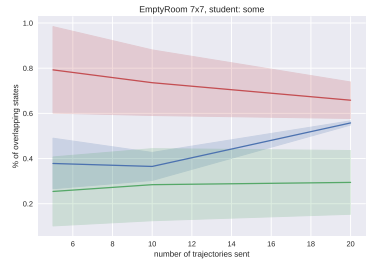
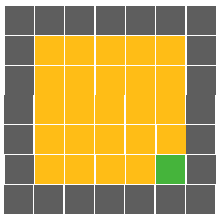
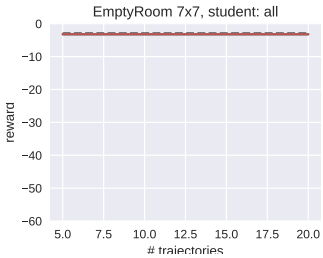
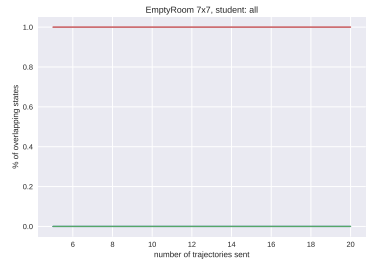
Student experience	Performance	% overlap
teacher: gp teacher: random_probe teacher: random_sample 		
		
		
		

Table 1: Student experience with how different teachers teach the student, EmptyRoom 7x7.

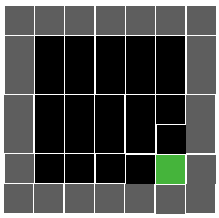
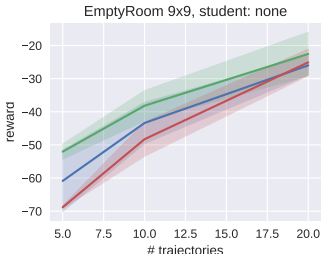
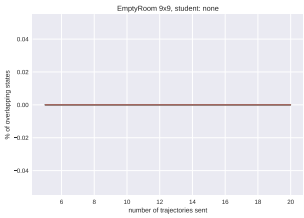
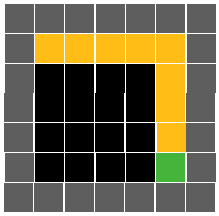
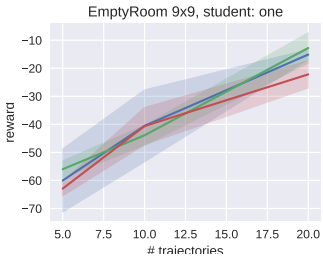
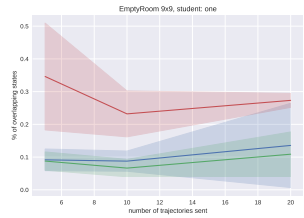
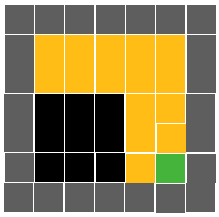
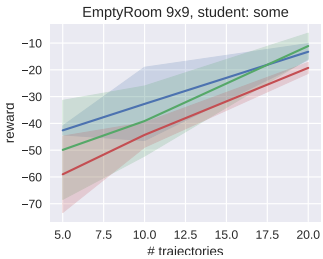
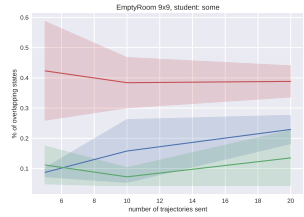
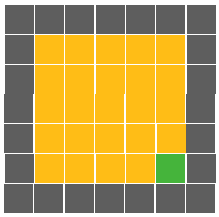
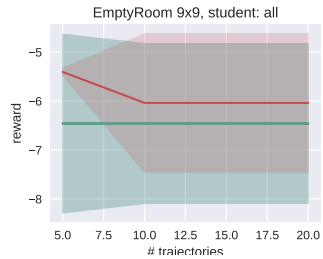
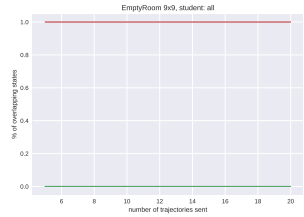
Student experience	Performance	% overlap
teacher: gp teacher: random_probe teacher: random_sample 		
		
		
		

Table 2: Student experience with how different teachers teach the studen, EmptyRoom 9x9.

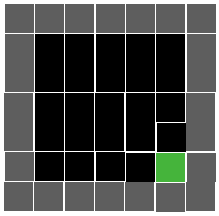
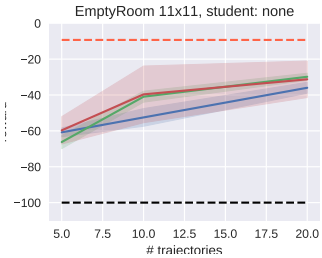
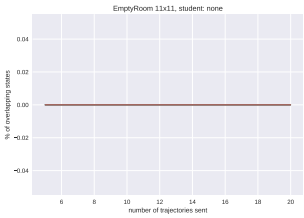
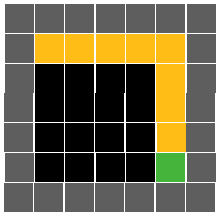
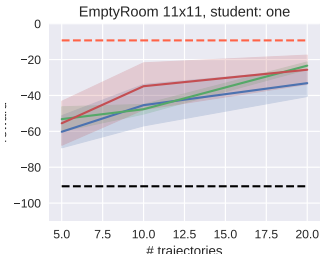
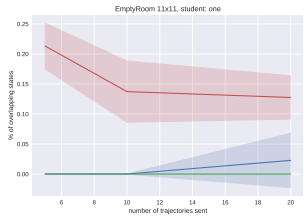
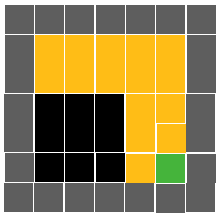
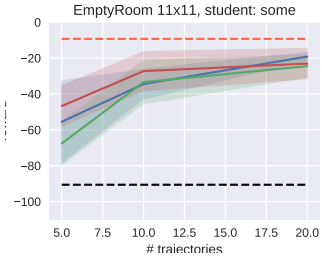
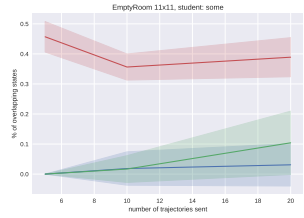
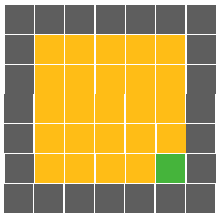
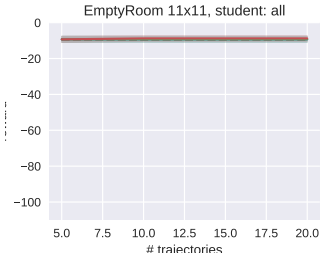
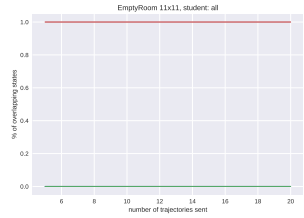
Student experience	Performance	% overlap
teacher: gp teacher: random_probe teacher: random_sample 		
		
		
		

Table 3: Student experience with how different teachers teach the studen, EmptyRoom 11x11.

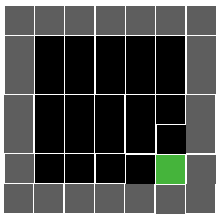
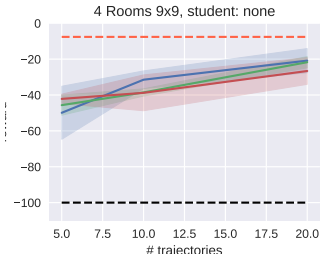
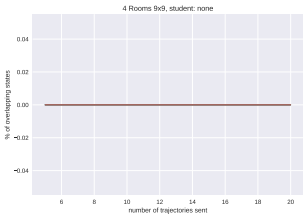
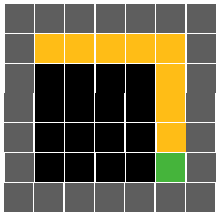
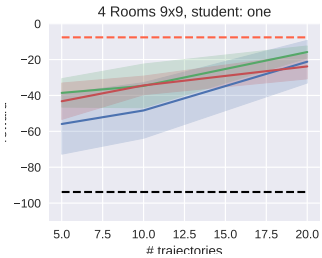
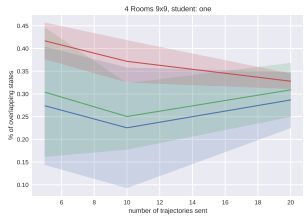
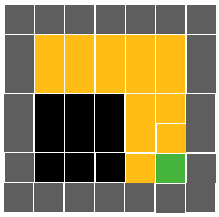
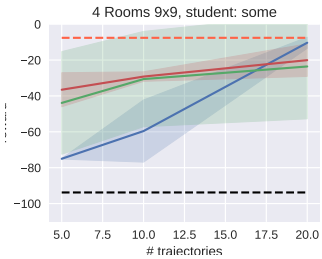
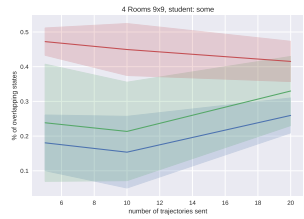
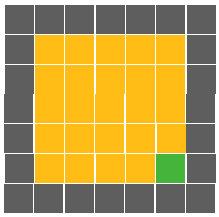
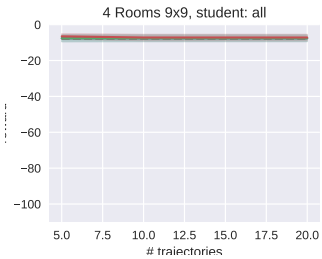
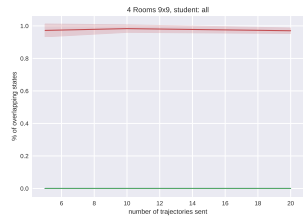
Student experience	Performance	% overlap
teacher: gp teacher: random_probe teacher: random_sample 		
		
		
		

Table 4: Student experience with how different teachers teach the student, 4 rooms 9.

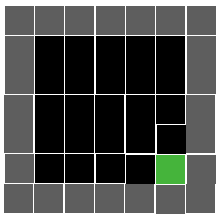
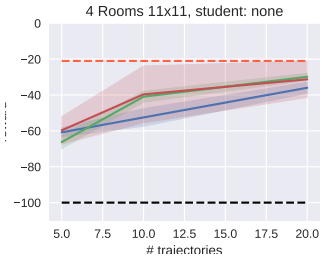
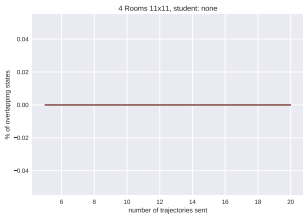
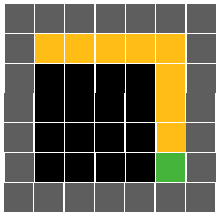
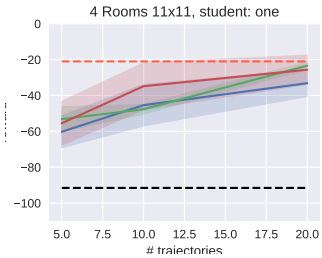
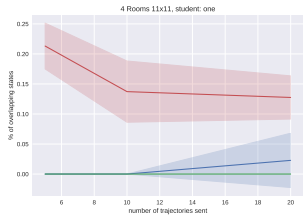
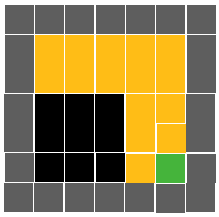
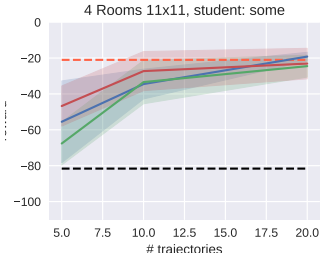
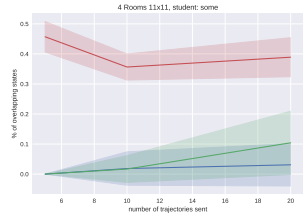
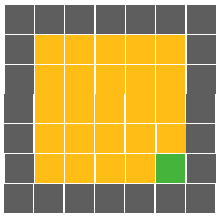
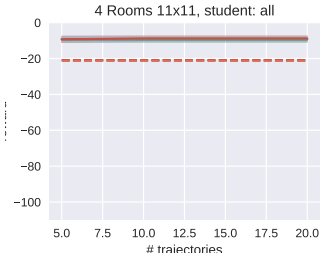
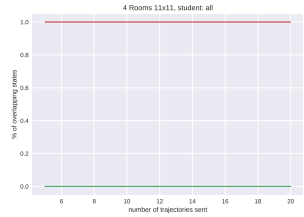
Student experience	Performance	% overlap
teacher: gp teacher: random_probe teacher: random_sample 		
		
		
		

Table 5: Student experience with how different teachers teach the student, 4 rooms 11.