Help an Agent Out: Student/Teacher Learning in Sequential Decision Tasks

Lisa Torrey
St. Lawrence University
ltorrey@stlawu.edu

Matthew E. Taylor
Lafayette College
taylorm@lafayette.edu

ABSTRACT

Research on agents has led to the development of algorithms for learning from experience, accepting guidance from humans, and imitating experts. This paper explores a new direction for agents: the ability to teach other agents. In particular, we focus on situations where the teacher has limited expertise and instructs the student through action advice. The paper proposes and evaluates several teaching algorithms based on providing advice at a gradually decreasing rate. A crucial component of these algorithms is the ability of an agent to estimate its confidence in a state. We also contribute a student/teacher framework for implementing teaching strategies, which we hope will spur additional development in this relatively unexplored area.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Miscellaneous

General Terms

Algorithms, Performance

Keywords

Reinforcement Learning, Inter-agent teaching, Transfer Learning

1. INTRODUCTION

Agents are becoming increasingly common in industry, education, and domestic environments. Significant advances have been made in autonomous learning and learning with human guidance. However, less attention has been paid to the question of how agents could best teach each other. For instance, an existing robot in a factory should be able to instruct a newly arriving robot, even if it is from a different manufacturer, has a different knowledge representation, or is not optimal itself.

This paper investigates a variety of teaching methods in sequential decision tasks. In particular, we consider a reinforcement learning student that must learn from autonomous exploration of the environment under the guidance of another agent. In order to minimize inter-operability requirements, the teacher and student are presumed not to know each others' internal workings; teachers can only help students by suggesting actions. Furthermore, the teacher may have limited expertise in the student's task, so it should be careful not to over-advise the student. The primary question we address is: how should the teacher decide when to give advice?

The teaching context is related to the more well-studied problem of *transfer learning* [16, 17, 18], in which an agent uses knowledge from a source task to aid its learning in a target task. However, transfer algorithms often assume that agents can directly access the internal knowledge representation from the source task, which is too strong an assumption to make in teaching.

Another related area is *learning from experts* [4, 12], where agents may imitate experts or ask for their advice. This setting puts the student in charge of the process; our work gives more control to the teacher, because the teacher is the agent with more initial knowledge. Our setting also focuses on non-expert teachers whose knowledge may not be complete.

We introduce a family of teaching methods and conduct a set of experiments to evaluate and compare them. Experiments take place in both discrete and continuous tasks, and with varying levels of teacher expertise.

Empirical results from this study highlight a number of insights into inter-agent teaching. First, teachers can make better decisions about when to give advice if they are able to judge their own confidence in their knowledge. Second, teachers can also make better decisions if they are able to ask about the student's confidence and compare it to their own. Third, there are multiple factors that affect the relative performance of different teaching algorithms: the domain, the teacher's level of expertise, and even the student's exploration strategy.

The primary contributions of this paper are to suggest effective algorithms for teaching, to highlight interesting empirical observations, and to provide an open-source framework for evaluating teaching methods. Our hope is that this paper enables and inspires the agents community to develop further methods by which agents can teach other agents.

2. BACKGROUND AND RELATED WORK

This section provides a summary of important background information, and a survey of existing work in relevant areas.

2.1 Reinforcement Learning

This paper focuses on *reinforcement learning* (RL), a popular formulation for sequential decision tasks [7, 14]. RL tasks are typically framed as Markov decision processes (MDPs) and defined by the 4-tuple of state set, action set, transition function, and reward function: $\{S, A, T, R\}$. A learner chooses which action to take in a state via a policy, $\pi : S \mapsto A$, which is modified over time. A better policy gives the learner better performance, which is defined as the expected (discounted) total reward. To learn an optimal policy, agents need to balance exploration and exploitation.

Many RL algorithms are based on building an action-value function, $Q : S \times A \mapsto \mathbb{R}$, which maps state-action pairs to their expected return. In large or continuous state spaces, agents typically factor the state into features: $s = \langle x_1, x_2, \ldots, x_n \rangle$. In such cases, RL methods typically use function approximation to represent the Q-function, which produces state space abstraction.

2.2 Humans Teaching Agents

Some methods for humans teaching agents are relevant to the current work. For instance, *Learning from Demonstration* (LfD) [2] includes a broad category of work that focuses on agents learning to mimic a human demonstrator. Much of the existing LfD research focuses on compensating for variance/errors in actions demonstrated by humans and determining where the estimate of the human action is accurate. LfD methods are naturally centered on the student, whereas the current work centers on a (non-human) teacher.

Inverse reinforcement learning [1] is another increasingly popular paradigm. In this setting, an agent must act in an MDP without a reward signal. The agent observes a human, tries to infer the human's reward function, and then maximizes this function. In contrast, in this work we assume the MDP does have a reward signal, and we do not assume that the student will do best by imitating the teacher.

There is a wide range of other work on providing human help to an agent, such as giving high-level programmatic advice [10]. These methods may inform future work with agent teachers, but they would require teachers and students to be able to communicate more than just immediate action advice.

2.3 Agents Teaching Agents

In *transfer learning* (TL), an agent uses knowledge from a source task to aid its learning in a target task [16, 17, 18]. Often, the agent is allowed to copy source-task knowledge directly, and then continues to learn from that starting point. In contrast, this work assumes that a student agent cannot directly transfer all knowledge completely and immediately from its teacher, because their internal knowledge representations are not known *a priori*, or are even incompatible.

Probabilistic policy reuse (PPR) is a TL technique in which the agent uses a transferred policy with probability ψ, explores with probability ϵ, and exploits the current policy with probability $1 - \psi - \epsilon$ [5, 6]. By decaying ψ over time, the agent can initially leverage transferred knowledge, and then learn to improve upon it. The PPR framework will be used later in this paper for balancing the need to exploit the teacher's knowledge with the need for the student to learn autonomously.

Ask For Help is a method for agents to learn from other expert agents [3, 4]. In this system, an agent asks for advice when its confidence in a state is low. Our work builds upon the idea of making decisions based upon confidence in a state, but we consider the teacher's confidence as well as the student's. Another difference is that we have the teacher make the advice decisions, since it is more knowledgeable than the student.

Experience replay [9] has been successfully used to share recorded experiences between agents in Q-Learning and in batch reinforcement settings [8], but it requires the teacher to store experience samples, and the student must have an identical state representation. Tan [15] extends this idea, allowing agents to share episodes and entire policies, but is also restricted to Q-learning agents with identical representations.

Learning by watching [19] is another example of experience sharing, improved upon by *imitation learning*, which allows agents to have different action sets but still requires them to have the same state representation [12].

Others, including Nunes and Oliveira [11], have considered groups of agents simultaneously learning in a single environment, where agents share experience among themselves. In contrast, this work focuses on teaching rather than cooperative learning.

3. METHODS

Our goal in this paper is to explore ways that agent teachers can help agent students to learn sequential decision tasks. A core assumption is that agents cannot necessarily understand each others' internal workings and thus are limited to teaching via communication, rather than other forms of knowledge transfer (e.g., directly copying a Q-function). This, of course, is the case when humans teach each other as well.

Humans often teach skills by guiding students' actions completely at first, then reducing guidance gradually as students become more capable. We propose a similar approach for agent teachers, and we begin by applying Probabilistic Policy Reuse [5, 6].

PPR allows an agent to learn a task faster by taking advantage of an existing policy. The PPR method, described in Algorithm 1, changes only the action selection step of the learning process. With probability ψ, the agent exploits an old policy; the rest of the time, it uses normal ϵ-greedy action selection. The value of ψ decays over time according to a decay rate v so that the agent makes less use of old policies as it improves its own.

Method 1 Probabilistic Policy Reuse: learning a new policy with assistance from an old policy

```
Load old policy π_old
Initialize new policy π_new
Set ψ = 1
for each training episode do
  for each state s do
    if random(0,1) < ψ then
      Take action π_old(s)
    else if random(0,1) < ε then
      Take a random action
    else
      Take action π_new(s)
    end if
    Update policy
  end for
  Decay ψ = ψv
end for
```

PPR has some good characteristics for teaching. It allows a teacher to give advice frequently at first and less frequently over time. However, recall another core assumption we make: teachers do not always have complete expertise. They may have more knowledge in some states than others and could even have some incorrect knowledge. This can also be true in human teaching as well, though we may be less likely to admit it.

For a teacher with limited expertise, PPR may not be the optimal teaching algorithm. A PPR teacher provides action advice with a global probability ψ that is uniform across all states. If the teacher is more confident in some states than others, it makes more sense for advice probabilities to be higher in some states than others. We therefore propose several new algorithms that use *teacher confidence* to make advice probabilities state-specific.

We will consider the student's confidence eventually as well. However, we focus first on the teacher's confidence, because of the issue of limited expertise.

3.1 Measuring Confidence

To support these new algorithms, we need a way to estimate an agent's confidence in a state. A common approach to this problem is *Q-value interval estimation*, where confidence is measured by the difference between the highest and lowest Q-values in a

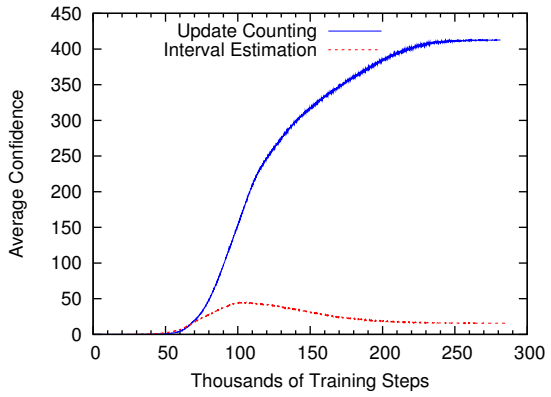

Figure 1: This graph shows how an agent's average confidence per state changes while learning to navigate a maze, using two different confidence measures.

state [4]. However, the interval-estimation measure produces counterintuitive results in some domains.

For example, consider an agent learning to navigate a maze. The agent gains confidence as it first discovers paths to the goal state, but after a while it begins to *lose* confidence, as shown in Figure 1. This is because maze states are highly connected — the true Q-values of actions in these states are not very different. Interval estimation is therefore not a stable confidence measure for maze-like domains. Since such domains are used in our experiments, we do not use interval estimation.

Instead, we introduce *visit counting*, an approach that measures confidence in terms of the number of times an agent has visited a state. It is easily implemented in discrete settings by keeping a table of state visit-counts. When an agent visits a state s, it increments the visit-count $v(s)$, and its confidence in that state is $v(s)$. However, the visit-counting approach is also adaptable to continuous settings. For instance, when using tile-coding, we can use *tile* visit-counts, rather than state visit-counts. When an agent visits a state with active tiles $\langle t_1, t_2, \ldots, t_n \rangle$, it increments the tile visit-counts $v(t_1), v(t_2), ..., v(t_n)$, and its confidence in that state is $\sum_i v(t_i)/n$.

However, we note that visiting a state does not always give an agent more knowledge about it. It seems clear that knowledge is gained only if the agent makes a non-zero Q-value update. It may be possible to be even more restrictive than this, depending on the domain and the learning algorithm. If small negative rewards are received at each step by default, true knowledge gains may be represented only by *positive* Q-value updates. Or, if an agent uses optimistic Q-value initialization, knowledge gains may be represented only by *negative* Q-value updates.

We therefore suggest a metric that we label *update counting*, in which confidence is measured by the number of times an agent has made a non-trivial Q-value update in a state. The definition of "non-trivial" is necessarily domain-dependent and agent-dependent. Update counts extend to continuous settings with tile coding the same way that visit-counts do. This measure has more intuitive behavior than interval estimation, as Figure 1 shows.

3.2 State-Specific Probabilities

We now describe several new teaching algorithms that extend PPR for use with non-expert teachers. These algorithms all make use of a global probability ψ, but they also use confidence measures to compute state-specific advice probabilities.

A state-specific advice probability should have several general properties. First, it should decay over time. Second, it should be higher in states where the teacher is more confident. Third, teachers with lower confidence levels should give less advice overall than teachers with higher confidence levels. Fourth, for high-performing teachers, the state-specific probabilities should all start to converge to ψ.

Note that we calculate an advice *probability* in each state, not just a binary decision on whether or not to give advice. This allows the teacher to smoothly decrease its guidance over time, and cleanly integrates with the PPR framework.

We propose three algorithms that meet these specifications in different ways. In a state s, a teacher must compute a probability of giving advice $p(s)$. Let $c_t(s)$ represent the teacher's confidence in that state.

Our first algorithm computes:

$$p(s) = \begin{cases} 0 & \text{if } c_t(s) < 1 \\ \psi & \text{if } c_t(s) \geq 1 \end{cases}$$

We label this algorithm *conditional-PPR*, since its advice is conditional upon having at least some confidence. We use the threshold $c_t(s) = 1$ as the confidence cutoff in order to draw the line clearly between *no knowledge* and *some knowledge*. This approach provides a simple but logical alternative to PPR that allows a teacher to avoid giving advice in unfamiliar states. For a teacher who is confident in all states, this algorithm becomes equivalent to regular PPR.

Our second algorithm computes:

$$p(s) = \begin{cases} 0 & \text{if } c_t(s) < 1 \\ \psi \frac{c_t(s)+f}{\max(c_t)+f} & \text{if } c_t(s) \geq 1 \end{cases}$$

Here $\max(c_t)$ represents the maximum confidence level the teacher has experienced in any state during its training.

We label this algorithm *proportional-PPR*, since its advice probability is proportional to confidence. In states with higher confidence, it gives advice with a higher probability, up to a maximum of ψ. This approach allows the teacher to scale its level of guidance more precisely in different states.

The floor parameter $f \geq 0$ can be used to shrink the range of probabilities this function produces. As f increases, the minimum advice probability for states with non-zero confidence rises. In the limit, this algorithm becomes equivalent to conditional-PPR.

Our third algorithm computes:

$$p(s) = \begin{cases} 0 & \text{if } c_t(s) < 1 \\ \min\left(1 - \frac{c_s(s)}{c_t(s)+d}, \psi\right) & \text{if } c_t(s) \geq 1 \end{cases}$$

Here $c_s(s)$ represents the *student's* confidence in state s.

We label this algorithm *relative-PPR*, since its advice probability depends on the relationship between the student's confidence and the teacher's confidence. In states where the teacher has much higher confidence than the student, it gives advice with a higher probability, up to a maximum of ψ. As the student's confidence in a state grows, the advice probability decreases, and eventually it stops altogether.

The delay parameter $d \geq 0$ can be used to slow down this process. As d increases, teachers require students to reach higher levels of confidence before they stop giving advice. In the limit, this algorithm also becomes equivalent to conditional-PPR.

This approach combines the probability concepts of PPR, the confidence concepts of Ask-For-Help, and the additional concept of comparing student and teacher confidence. Of our three algorithms, it determines advice probabilities in the most sophisticated way.

```
XXXXXXXXXXXXXXXXXXXX
X  XXX  X       X  X
XX X  X X X X X  X
X            XXX   X
X   XXX    X   X XXX
X X              X X
X     X       X XX  X
X  XX         X X   X
X    XX  X  X   X  X
XX    X    X    X  X
X   X  XXXX       XX
XX   X         X X XX
X     X   X    XXXXX
XX  X     XX XX X XX
XX       XXXX       X
X  X XX  XX  XX    X
X  XXX         XXXX X
X X      X          X
XXX    X    XX    @X
XXXXXXXXXXXXXXXXXXXX
```

```
XXXXXXXXXXXXXXXXXXXX
X  XXX  X       X  X
XX X  X X X X X  X
X            XXX   X
X   XXX    X   X XXX
X X              X X
X     X       X XX  X
X  XX         X X   X
X    XX  X  X   X  X
XX    X    X    X  X
X   X  XXXX       XX
XX   X         X X XX
X     X   X    XXXXX
XX  X     XX XX X XX
XX       XXXX       X
X  X XX  XX  XX    X
X  XXX         XXXX X
X X      X          X
XXX    X    XXOOO @X
XXXXXXXXXXXXXXXXXXXX
```

Figure 2: This figure shows the maze world (left) and the cliff world (right). Wall cells are labeled X, cliff cells are labeled O, and the @ is the goal cell.

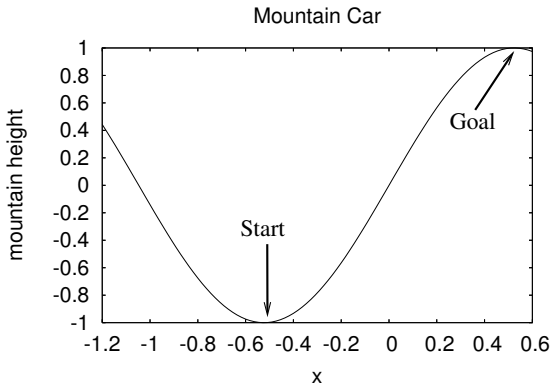

Figure 3: This figure shows the traditional mountain car environment, where x is the agent's location.

4. EVALUATION

To evaluate and compare these teaching algorithms, we perform teaching experiments in three domains: maze world, cliff world, and mountain car. The goal in all of these domains is to complete episodes using a minimum number of steps.

Our maze world comes from the Ask-For-Help work [4]. It is a 20x20 grid in which each open cell is a different state, and the agent starts in a random state. Actions corresponding to the four cardinal directions allow the agent to move between cells. The reward function is 0 for every transition except one in which the agent enters the goal cell. Reaching the goal ends the episode, and there is no limit to the number of steps an agent can take.

Our cliff world is identical to the maze world, except that it contains cells that represent pits, arranged in a row like a cliff. If an agent enters a cliff cell, it gets reset back to its starting position. Figure 2 shows an illustration of the maze and cliff environments.

Mountain Car is a benchmark RL task, commonly used to test algorithms in a simple continuous state task [13]. The car starts near the bottom of the hill with zero velocity, and must drive to the top of the hill. However, its motor is underpowered, so the car must build up enough energy to reach the goal by moving back and forth. Figure 3 shows an illustration of this environment.

States in this domain are described by two continuous features: the car's position and velocity. The agent has three actions: accelerate in the +x direction, accelerate in the -x direction, or do not accelerate at all. The transition function is a simple physics simulation involving position, velocity, acceleration, and gravity. The car is given a reward of -1 at every step until it reaches the goal location. Reaching the goal ends the episode, but there is also a limit of 1000 steps per episode.

We train agents in these worlds with learning algorithms that are appropriate to their setting. In the maze world, we use simple tabular Q-learning. In the cliff world, we use tabular Sarsa, since the Sarsa variant of Q-learning is better for handling state spaces where exploration can be extremly costly [14]. In mountain car, we use Sarsa(λ) with 16 tile codings to handle the continuous features.

In each learning algorithm, we use default parameters for these domains that produce reasonable learning curves for independent agents. For the maze and cliff, these are $\alpha = 0.15$, $\epsilon = 0.05$, and $\gamma = 0.99$. For mountain car, they are $\alpha = 0.25$, $\epsilon = 0.05$, $\gamma = 1.0$, and $\lambda = 0.25$.

To produce teachers with limited expertise, we do not allow them to train until their policies converge. Instead, we train teachers for only 10, 20, or 30 episodes. In all the domains, we allow teacher episodes to be as long as necessary for the teacher to reach a goal state. Otherwise, two teachers who have trained for 10 episodes could have very different confidence levels, which would be a large source of unnecessary variability in the experiments.

Each teacher then gives advice to students using the four teaching algorithms we have discussed: regular PPR, conditional-PPR, proportional-PPR, and relative-PPR. We use the update-count metric for estimating confidence. For the maze and cliff worlds, we use state visit-counts that are incremented when a visit to a state produces a non-zero Q-value update. For mountain car, we use tile visit-counts that are incremented when a visit to a state produces a positive Q-value update.

Each teaching algorithm also has parameters that must be set. They all share the decay-rate parameter v; proportional-PPR also has the floor parameter f and relative-PPR has the delay parameter d. Since appropriate values for these parameters are domain-dependent, we determine a set of 3 reasonable choices for each parameter in each domain and select the best of those settings.

To produce smooth learning curves for the students, we average the performance of 100 students for each experiment. To lend perspective to the student learning curves, we also show curves for *independent* agents and *direct-transfer* agents. Independent agents learn without a teacher, and should therefore learn more slowly than any student. Direct-transfer agents copy the teacher's entire Q-function, and should therefore learn more quickly than all other students. In fact, because they have direct access to their teacher's brains, direct-transfer agents should represent a bound for student performance.

To compare the algorithms quantitatively as well as visually, we compute areas under learning curves and perform paired t-tests to check for significant differences between algorithms.

4.1 Maze World Results

Figures 4, 5, and 6 show how students learn from teachers with varying levels of expertise in the maze world. As expected, teachers with more training are more helpful to their students. However, the PPR algorithm provides some benefits to students at all levels of training, and our state-specific versions produce similar results.

At the lowest training level, regular PPR is more beneficial than the rest, and this difference is statistically significant. This result is surprising, since it means that giving advice in unfamiliar maze

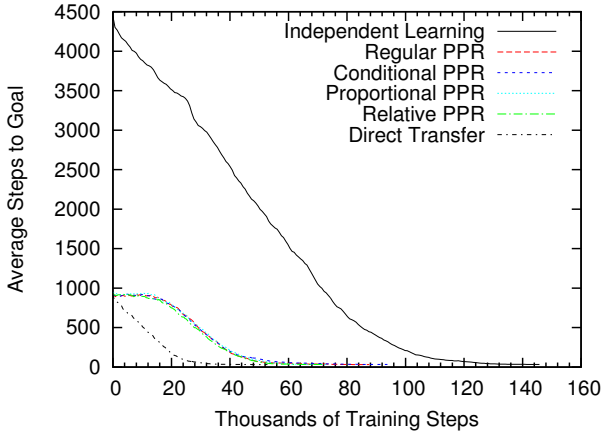

Figure 4: Maze world performance of students whose teachers had 30 episodes of training

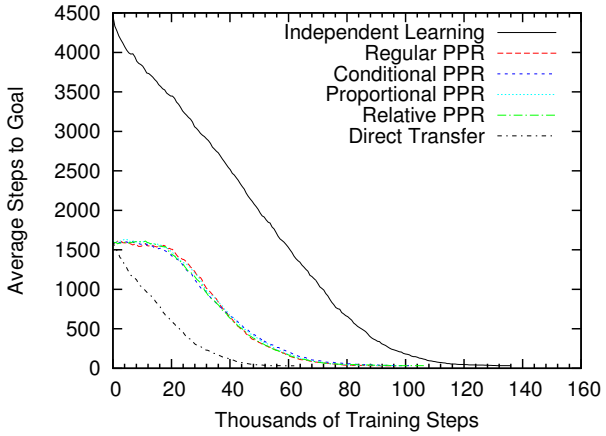

Figure 5: Maze world performance of students whose teachers had 20 episodes of training

states is somehow helpful. Further inspection reveals that advice in these states is random, and thus equivalent to exploration. This changes the student's exploration rate from constant ϵ to something correlated with decaying ψ. If we let all the students use $\epsilon = \psi$, the difference between regular PPR and the other approaches disappears. This indicates that the student's exploration rate can have a significant impact on the effectiveness of a teaching algorithm.

We find that reasonable parameter settings in the maze world are $v \in \{0.9, 0.99, 0.999\}$, $f \in \{10, 100, 1000\}$, and $d \in \{0, 10, 100\}$. Table 1 shows the best settings for each algorithm. Note that the floor parameters for proportional-PPR tend to be the higher ones, which makes that approach comparable to conditional-PPR. This means a teacher in the maze world does best using even small amounts of knowledge to their fullest degree. The delay rate d for relative-PPR is not particularly important; all the choices produce similar results. This means relative-PPR is not very sensitive to parameter selection in this domain.

One other important difference between the teaching algorithms in the maze world is the amount of advice they provide to students. Table 2 shows the average number of times each algorithm gives advice while training the students. The state-specific approaches give drastically less advice than regular PPR. They may therefore

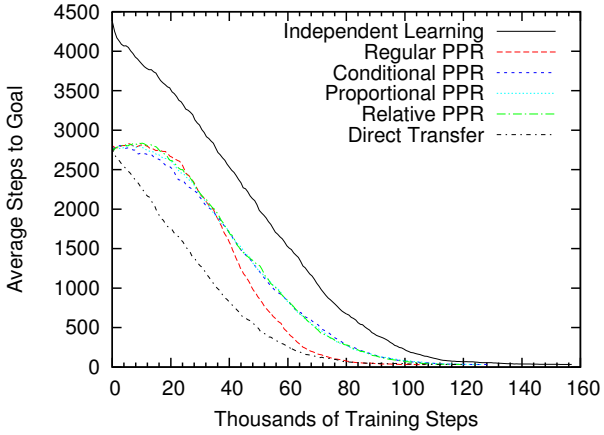

Figure 6: Maze world performance of students whose teachers had 10 episodes of training

	Training = 30	Training = 20	Training = 10
PPR	$v = 0.99$	$v = 0.99$	$v = 0.99$
c-PPR	$v = 0.99$	$v = 0.999$	$v = 0.99$
p-PPR	$v, f = 0.999, 100$	$v, f = 0.99, 1000$	$v, f = 0.99, 1000$
r-PPR	$v, d = 0.999, 0$	$v, d = 0.99, 0$	$v, d = 0.9, 100$

Table 1: Best parameter settings in the maze world

be preferable in this domain simply for their advice efficiency.

The maze world contains no critical decision points; there is no state in which a particular action is crucial to take or avoid. Teachers can therefore give some non-optimal advice in this domain without causing harm. This is the reason that regular PPR performs comparably to state-specific versions in the maze world. Our next experiments in the cliff world will tell a different story.

4.2 Cliff World Results

Figures 7, 8, and 9 show how students learn from teachers with varying levels of expertise in the cliff world. In these experiments, all of the state-specific algorithms are more beneficial than regular PPR, and these differences are statistically significant.

The reason for this change is that cliffs make it dangerous to give advice in unfamiliar states, as is done by regular PPR. States near cliffs are critical decision points. Some teachers become familiar with the states near the cliff during their limited training, but others do not. Our algorithms allow only the confident teachers to give advice in those states. The more critical decision points are in a domain, the more superior state-specific approaches should be for teachers with limited expertise.

Reasonable parameter settings in the cliff world are the same as in the maze world. Table 3 shows the best settings for each algorithm. The amounts of advice given by teachers in the cliff world are shown in Table 4. As in the maze world, the state-specific approaches give drastically less advice than regular PPR.

	Training = 30	Training = 20	Training = 10
PPR	55,728	112,505	209,512
conditional-PPR	1,275	2,025	617
proportional-PPR	2,496	1,121	623
relative-PPR	592	353	225

Table 2: Amounts of advice given by teachers while training the students in the maze world

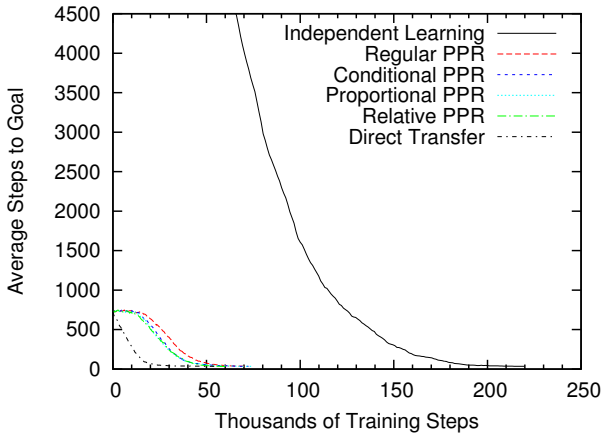

Figure 7: Cliff world performance of students whose teachers had 30 episodes of training

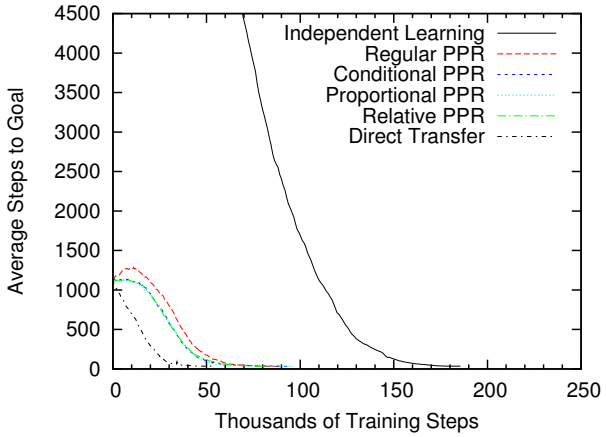

Figure 8: Cliff world performance of students whose teachers had 20 episodes of training

4.3 Mountain Car Results

Figures 10, 11, and 12 show how students learn from teachers with varying levels of expertise in mountain car. In these experiments, conditional-PPR and proportional-PPR produce similar results to regular PPR. However, relative-PPR is more beneficial, and these differences are statistically significant.

These results can be best understood by looking at the confidence mechanics in mountain car. Since it is a tile-coding domain, update counts are assigned to tiles rather than states, and an agent's confidence in a state is the average update-count of the state's active tiles. One effect of spreading confidence across tiles is that confidence tends to grow quickly throughout the state space. Unseen states can still receive confidence if their component tiles have been seen (due to visiting nearby states). It does not take long for an agent to have at least some confidence in nearly all states.

In this situation, conditional-PPR and proportional-PPR should approach equivalency with regular PPR, and their performance reflects this. Relative-PPR is better equipped to handle these confidence mechanics, because it takes both teacher and student confidence into account. It backs off quickly in states where the student gains confidence quickly, but keeps giving advice in the less common states.

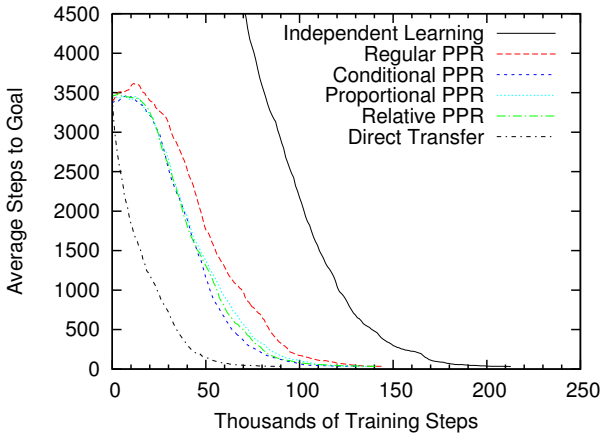

Figure 9: Cliff world performance of students whose teachers had 10 episodes of training

	Training = 30	Training = 20	Training = 10
PPR	$v = 0.99$	$v = 0.9$	$v = 0.9$
c-PPR	$v = 0.99$	$v = 0.999$	$v = 0.99$
p-PPR	$v, f = 0.999, 1000$	$v, f = 0.999, 1000$	$v, f = 0.999, 1000$
r-PPR	$v, d = 0.999, 0$	$v, d = 0.999, 100$	$v, d = 0.99, 10$

Table 3: Best parameter settings in the cliff world

We find that reasonable parameter settings in mountain car are $v \in \{0.99, 0.97, 0.95\}$, $f \in \{100, 1000, 10000\}$, and finally $d \in \{10, 100, 1000\}$. Table 5 shows the best settings for each algorithm. Note that the parameters for relative-PPR are more important in this domain; the performance of the approach varies more than it does in the maze and cliff worlds.

The amounts of advice given by teachers in mountain car are shown in Table 6. The state-specific approaches use only slightly less advice than regular PPR. Since teachers tend to have at least some confidence in nearly all states, they give substantially more advice.

5. FUTURE WORK AND CONCLUSIONS

The literature on learning agents naturally focuses on algorithms that agents can use to learn. This paper contributes an initial study of algorithms that agents can use to *teach*. It focuses on agents teaching other agents in sequential decision tasks. We assume a broadly applicable setting, in which teachers and students interact through action advice and in which teachers have limited expertise.

This paper contributes a family of teaching methods for advising students. The algorithms are based on Probabilistic Policy Reuse, but they use the concept of agent confidence to make advice probabilities state-specific. Empirical results show that state-specific methods, particularly one that takes both teacher and student confidence levels into account, are effective in the teaching setting.

	Training = 30	Training = 20	Training = 10
PPR	53,509	30,489	114,102
conditional-PPR	1,562	2,547	962
proportional-PPR	3,060	2,527	1,664
relative-PPR	728	1,255	416

Table 4: Amounts of advice given by teachers while training the students in the cliff world

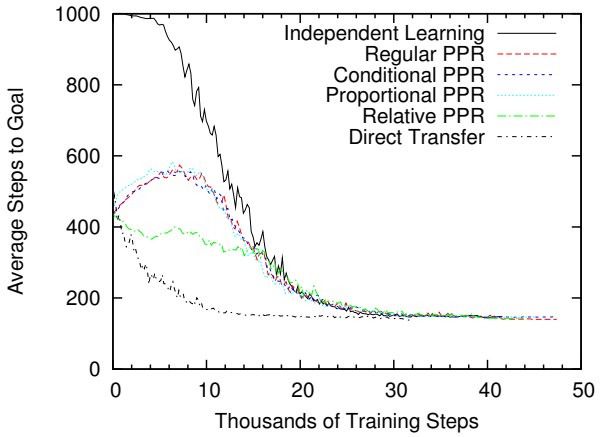

Figure 10: Mountain car performance of students whose teachers had 30 episodes of training

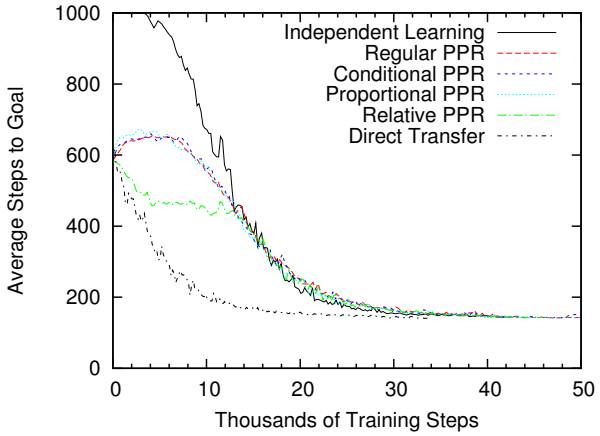

Figure 11: Mountain car performance of students whose teachers had 20 episodes of training

There are many potential directions for future work in this area. Teachers could explicitly reason about the expense of communication versus the expected gain, which would be appropriate in domains where communication has a non-zero cost. Teachers could use student experience to adjust their confidence levels. There could also be multiple teachers, with different areas of expertise, that must coordinate with each other.

Students could also be given more active roles than they currently have in this work. For instance, a student could estimate its teacher's performance and decide whether or not to use the advice it receives in a certain area of the state space, or select which advice to follow when multiple teachers are present.

Finally, algorithms that allow agents to teach each other may also inform strategies for agents to teach humans. Agents could make particularly patient teachers, and using some of the ideas in this paper, they could also be responsive to student learning. For instance, an agent teacher could attempt to estimate a human's confidence through visit counts, reaction times, and other non-verbal cues, and then use this estimate to decide whether to provide advice.

We hope that this paper encourages others to continue studying inter-agent teaching, as well as providing a set of algorithms and results to serve as benchmarks.

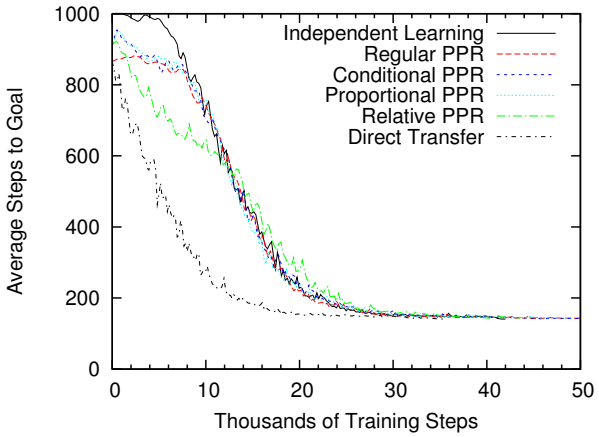

Figure 12: Mountain car performance of students whose teachers had 10 episodes of training

	Training = 30	Training = 20	Training = 10
PPR	$v = 0.97$	$v = 0.97$	$v = 0.95$
c-PPR	$v = 0.97$	$v = 0.97$	$v = 0.95$
p-PPR	$v, f = 0.97, 1000$	$v, f = 0.97, 1000$	$v, f = 0.95, 1000$
r-PPR	$v, d = 0.99, 100$	$v, d = 0.99, 100$	$v, d = 0.99, 100$

Table 5: Best parameter settings in mountain car

6. REFERENCES

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

[2] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009.

[3] J. A. Clouse. An introspection approach to querying a trainer. Technical Report 96-13, University of Massachusetts, Amherst, MA, USA, 1996.

[4] J. A. Clouse. *On integrating apprentice learning and reinforcement learning*. PhD thesis, University of Massachusetts, 1996.

[5] F. Fernández, J. García, and M. Veloso. Probabilistic policy reuse for inter-task transfer learning. *Robotics and Autonomous Systems*, 58(7):866–871, 2010. Advances in Autonomous Robots for Service and Entertainment.

[6] F. Fernandez and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, 2006.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.

[8] S. Kalyanakrishnan and P. Stone. Batch reinforcement

	Training = 30	Training = 20	Training = 10
PPR	21,911	25,426	31,769
conditional-PPR	21,840	23,983	21,951
proportional-PPR	21,745	23,782	21,857
relative-PPR	18,859	19,806	20,687

Table 6: Amounts of advice given by teachers while training the students in mountain car

learning in a complex domain. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 650–657, May 2007.

[9] L. J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

[10] R. Maclin and J. W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22(1-3):251–281, 1996.

[11] L. Nunes and E. Oliveira. Learning from multiple sources. In *In Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1106–1113, 2004.

[12] B. Price and C. Boutilier. Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research*, 19:569–629, 2003.

[13] S. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158, 1996.

[14] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[15] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.

[16] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

[17] M. E. Taylor, P. Stone, and Y. Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

[18] L. Torrey, T. Walker, J. W. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 412–424, 2005.

[19] S. D. Whitehead. A complexity analisys of cooperative mechanisms in reinforcement learning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 607–613, 1991.