Generalizing Apprenticeship Learning across Hypothesis Classes

Thomas J. Walsh THOMASWA@CS.RUTGERS.EDU
Kaushik Subramanian KAUSUBBU@EDEN.RUTGERS.EDU
Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Rutgers University, Piscataway, NJ 08854 USA

Carlos Diuk CDIUK@PRINCETON.EDU
Princeton University, Princeton, NJ 08544 USA

Abstract

This paper develops a generalized apprenticeship learning protocol for reinforcement-learning agents with access to a teacher who provides policy traces (transition and reward observations). We characterize sufficient conditions of the underlying models for efficient apprenticeship learning and link this criteria to two established learnability classes (KWIK and Mistake Bound). We then construct efficient apprenticeship-learning algorithms in a number of domains, including two types of relational MDPs. We instantiate our approach in a software agent and a robot agent that learn effectively from a human teacher.

1. Introduction

Teachers unquestionably increase the speed and efficacy of learning in humans. Yet in the field of reinforcement learning (RL), almost all learning agents gain experience solely by interaction with their environment—teachers are not in the loop. This work addresses this disconnect by proposing a generalized protocol for *apprenticeship learning* within the reinforcement-learning paradigm, characterizing sufficient conditions for efficient apprenticeship learning that cover a wide swath of important AI domains, and showing there is a potentially exponential improvement in sample complexity when an agent can interact with a teacher instead of learning on its own.

The idea of integrating a teacher into the learning process has been proposed in several different forms. For instance, in the early computational learning theory literature, *equivalence queries* (Angluin, 1988) played the role of teacher and were shown to increase the class of learnable concepts in the supervised learning setting. In this work, we expand the apprenticeship protocol of Abbeel & Ng (2005) to cover a wider array of model classes. We consider a teacher with a policy π_T that can deliver a *trace*, a sequence of states, actions, and rewards obtained by executing π_T from a start state, to the learning agent after seeing it behaving suboptimally. We note that this scenario is different from *inverse reinforcement learning* (Abbeel & Ng, 2004), where the reward function is inferred from sequences of states and actions. Instead, our agents see the actual rewards and transitions induced by the teacher's policy and act to try to maximize this observable reward function.

We characterize a class of reinforcement-learning environments for which an agent can guarantee that only a polynomial number of example traces are needed to act near-optimally. Specifically, this class includes all KWIK-learnable domains from the autonomous case and all deterministic domains from the Mistake Bound (MB) learning class, a set that contains many models that thwart autonomous agents. These results generalize earlier theoretical results in a handful of RL representations, including flat MDPs, linear MDPs, Stochastic STRIPS, and deterministic OOMDPs.

2. Terminology

In this section, we propose a protocol for apprenticeship learning in RL domains and a supervised learning framework that will allow us to study the efficiency of such learners.

2.1. Reinforcement Learning

A reinforcement-learning (Sutton & Barto, 1998) agent interacts with an environment described by

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

Algorithm 1 The Apprenticeship-Learning Protocol

The agent starts with S, A and γ, a time-cap H and has access to episodic environment E
The teacher has policy π_T.
for each new start state s_0 from E **do**
  $t = 0$
  while The episode has not ended and $t < H$ **do**
    The agent chooses a_t.
    $\langle s_{t+1}, r_t \rangle = E.progress(s_t, a_t)$
    $t = t + 1$
  if the teacher believes it has a better policy for that episode **then**
    The teacher provides a trace τ starting from s_0.

a Markov Decision Process (MDP), which is a 5-tuple $\langle S, A, \mathcal{T}, R, \gamma \rangle$ with states S, actions A, transition functions $\mathcal{T} : S \times A \mapsto \Pr[S]$, reward function $R : S \times A \mapsto [R_{min}, R_{max}]$, and discount factor γ. An agent's deterministic[1] policy $\pi : S \mapsto A$ induces a value function over the state space, defined by the Bellman Equations: $Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s') Q^\pi(s', \pi(s'))$ and $V^\pi(s) = Q^\pi(s, \pi(s))$. An optimal policy π^* for an MDP is a policy such that $\forall s, V^*(s) = V^{\pi *}(s) \geq V^{\pi'}(s), \forall \pi'$. The goal of a standard (autonomous) reinforcement-learning agent is to achieve behavior close to this optimal value from its own experience sampling $\mathcal{T}$ and R on each step.

2.2. Generalized Apprenticeship Learning

This paper considers a paradigm where the agent's experience is augmented with experience produced by a teacher and the criterion is to find a policy whose value function is nearly as good as, or better than, the value function induced by the teacher's policy. Formally, we define the *Apprenticeship Learning Protocol* for episodic domains where each episode has a length of at most $H = Poly(|M|, |A|, R_{\max}, \frac{1}{1-\gamma})$ in Algorithm 1. Here, $|M|$ is a measure of environment complexity, described later.

Intuitively, the agent is allowed to interact with the environment, but, unlike standard RL, at the end of an episode, the teacher can provide the agent with a trace of its own behavior starting from the original start state. The criteria the teacher uses to decide when to send a trace is left general here, but one specific test of value, which we use in our experiments, is for the teacher to provide a trace if at any time t in the episode, $Q^{\pi_T}(s_t, a_t) < Q^{\pi_T}(s_t, \pi_T(s_t)) - \epsilon$. That is, the agent chooses an action that appears worse than the teacher's choice in some state. Traces are of the form: $\tau = (s_0, a_0, r_0), ...(s_t, a_t, r_t), ...(s_g, r_g)$ where s_0 is the initial state, and s_g is a terminal (goal) state or some other state if the H cutoff is reached. Notice that the trajectory begins before (or at) the point where the agent first acted sub-optimally, and may not even contain the state in which the agent made its mistake.

Since the teacher's policy may not be optimal, this trace could potentially prescribe behavior worse than the agent's policy. We distinguish between these traces and their more helpful brethren with the following definition.

Definition 1. *A **valid trace** (with accuracy ϵ) is a trace supplied by a teacher executing policy π_T delivered to an agent who just executed policy π_A starting from state s_0 such that $V^{\pi_T}(s_0) - \epsilon > V^{\pi_A}(s_0)$.*

This allows agents to outperform their teacher without being punished for it. With deterministic policies, this definition means that at no time in a valid trace does the teacher prescribe an action that is much worse than any of the actions the agent used in that state. Note that when the teacher enacts optimal behavior ($\pi_T = \pi^*$), only valid traces will be provided.

We now introduce a teacher into the learning loop in a way that allows us to characterize the *efficiency* of learning analogous to the way the PAC-MDP framework (Strehl et al., 2009) has been used to characterize efficient behavior in the autonomous setting. We define PAC-MDP-Trace learning as follows:

Definition 2. *An RL agent is said to be* PAC-MDP-Trace *if, given accuracy parameters ϵ and δ, and following the protocol outlined in Algorithm 1, the number of valid traces (with accuracy ϵ) received by the agent over its lifetime is bounded by $Poly(|M|, |A|, R_{\max}, \frac{1}{1-\gamma})$ with probability $1 - \delta$, where $|M|$ measures the complexity of the MDP's representation, specifically the description of $\mathcal{T}$ and R.*

2.3. Frameworks for Learning Models

In this section, we introduce a class of dynamics where PAC-MDP-Trace behavior can be induced. In autonomous reinforcement learning, the recent development of the KWIK (Li et al., 2008) or "Knows What It Knows" framework has unified the analysis of models that can be efficiently learned. The KWIK-learning protocol consists of an agent seeing an infinite stream of inputs $x_t \in X$. For every input, the agent has the choice of predicting a label ($\hat{y}_t \in Y$) or admitting $\perp$ ("I don't know"). If the agent predicts $\perp$, it sees a noisy observation z_t of the true label. A hypothesis h^* in class $\mathcal{H}$ is said to be KWIK learnable if, with prob-

[1] The results of this paper can also be extended to the case where the teacher and learner use stochastic policies.

ability $1-\delta$, two conditions are met. (1) Every time the agent predicts $\hat{y}_t \neq \perp$, $||\hat{y}_t - h^*(x_t)|| \leq \epsilon$. (2) The number of times $\hat{y}_t = \perp$ is bounded by a polynomial function of the problem description.

In autonomous reinforcement-learning, if $\mathcal{T}$ and R are efficiently KWIK learnable, efficient behavior can be achieved by optimistically filling in the $\perp$ predictions. These results cover a large number of models common to autonomous reinforcement learning, including flat MDPs and DBNs (Li et al., 2008). However, several relevant and intuitively appealing classes are not KWIK learnable. For instance, conjunctions of n terms are not KWIK learnable (see Section 4.2). An environment with a "combination lock" of n tumblers that need to be set to the correct digits for a high reward action ("unlock") to be effective, can require an exponential number of suboptimal steps. But, in the trace setting, learning to open such a lock is simple: the agent only needs the teacher to supply a single trace to learn the combination! Thus, there are clearly models that are learnable in the apprenticeship setting that are not autonomously learnable.

Prior work (Angluin, 1988) has established a link between teachable hypothesis classes and the mistake bound (MB) framework (Littlestone, 1987). However, both of these considered only the prediction setting (not sequential decision making). The MB learning protocol for model learning is essentially the same as the KWIK protocol except for three changes. (1) In MB, there is no $\perp$ prediction. The agent must always predict a $\hat{y}_t \in \{0, 1\}$ and receives a true label when it is wrong. (2) MB is only defined for deterministic hypothesis classes, so instead of z_t, the agent will actually see the true label. (3) Efficiency is characterized by a polynomial bound on the number of mistakes made. It follows (Li et al., 2008) that any efficient KWIK learning algorithm for a deterministic hypothesis class can become an efficient algorithm for MB by simply replacing all $\perp$ labels with an arbitrary element from Y. We now introduce the following related criteria, called a mistake-bounded predictor (MBP) and will later show that if $\mathcal{T}$ and R for an MDP are learnable in this framework, it is PAC-MDP-Trace learnable.

Definition 3. *A* **mistake-bounded predictor** *(MBP) is an online learner with accuracy parameters ϵ and δ that takes a stream of inputs from set X and maps them to outputs from a set Y. After predicting any $\hat{y}_t$, the learner receives a (perhaps noisy) label z_t produced by an unknown function from a known hypothesis class. An MBP must make no more than a polynomial (in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and some measure of the complexity of the hypothesis class) number of mistakes with probability $1-\delta$. Here, a mistake occurs if, for input x_t, the learner produces $\hat{y}_t$ and $||h^*(x_t) - \hat{y}_t|| > \epsilon$, where h^* is the unknown function to be learned.*

Like KWIK, MBP observations can be noisy, and like MB, the learner is allowed to make a certain number of mistaken predictions. In fact, we can formalize the relationships with the following propositions.

Proposition 1. *Any hypothesis class that is KWIK or MB learnable is MBP learnable.*

Proof. A KWIK learner can be used in its standard form, except that whenever it predicts $\perp$, the MBP agent should pick a $\hat{y}_t \in Y$ arbitrarily. Also, the underlying KWIK learner should not be shown new labels when it does not predict $\perp$, though the "outer" MBP agent does receive them.

MB learners are defined for deterministic hypothesis classes, so we can assume no noisy observations exist. In this setting, the MB and MBP protocols line up. □

We can combine MBP learners in several ways and still preserve the MBP properties. For instance, consider the following *MB-partitioned* class.

Proposition 2. *Consider two "low-level" MBP-learnable classes C_0 and C_1 with the input space X and disjoint output sets Y_0 and Y_1, respectively. Consider a "high-level" MB-learnable class C mapping X to $\{0, 1\}$. The composition of these classes where the output of the class C learner is used to select which low-level MBP class to use (if the output of the high-level learner is i, use class C_i) is MBP-learnable.*

Proof. On input x, get a prediction i from the C learner. Then, query the C_i learner and report its response as the solution. Observe y. Define i such that $y \in Y_i$, then train C with (x, i) and C_i with (x, y). By construction, all learners get the appropriate training data and will individually make a small number of mistakes. When they make accurate predictions, the overall learner is accurate. □

As an example, consider a factored MDP with a reward function defined as follows. If a predetermined conjunction c_R over all n factors is false, then the agent receives reward $R_{\min} < 0$ and otherwise it gets a reward drawn from a distribution over $[0, R_{\max}]$. Given that information (but not c_R or the distribution), the reward function can be learned using an MB conjunction learner and a KWIK learner for the distribution when the conjunction is true, because the cases are always discernible. In contrast, a class where the distribution when the conjunction is true was over $[R_{min}, R_{max}]$ is *not* covered under this case because the outputs sets of

the learning problems overlap (making it unclear how to solve the top-level learner).

Several simpler combinations of MBP learners also preserve the MBP properties. These include *input-partition*, where low level learners are chosen based on some known function of X (a degenerate case of *MB-partition* without the "high-level" MB learning); *union*, where the outputs of several MBP learners with the same input sets give predictions (one can simply use the "low-level" learner who has made the fewest mistakes and give samples to all the learners); and *cross-product*, where learners with disjoint inputs and outputs have their predictions combined. These forms of combination have proven useful in building KWIK learners for autonomous RL (Li et al., 2008) and we use *MB-partition* in Section 4.3.

3. Efficiency Results

In this section, we link the class of MBP learnable functions to efficient PAC-MDP-Trace learning agents.

3.1. MBP-Agent and Efficient Apprenticeship Learning

We introduce a model-based RL algorithm (MBP-Agent, Algorithm 2) for the apprenticeship setting that uses an MBP learner as a module for learning the dynamics of the environment. Notice that because MBP learners never acknowledge uncertainty, our algorithm for the apprenticeship setting believes whatever its model tells it (which could be mistaken). While autonomous learners run the risk of failing to explore under such conditions, the MBP-Agent can instead rely on its teacher to provide experience in more "helpful" parts of the state space, since its goal is simply to do at least as well as the teacher. Thus, even model learners that default to pessimistic predictions when little data is available (as we see in later sections and as are used in our experiments), can be used successfully in the MBP-Agent algorithm. Algorithm 2 has the following property.

Theorem 1. *An MBP-learner is PAC-MDP-Trace for any domain where the transitions and rewards are efficiently MBP learnable.*

The heart of the argument is an extension of the standard Explore-Exploit lemma, we call the *Explore-Exploit-Explain Lemma*.

Lemma 1. *On each trial, we can define a set of known state,action ($\langle s, a\rangle$) pairs as the ones where the MBP currently predicts transitions accurately. One of these outcomes occurs: (1) The agent will encounter an unknown $\langle s, a\rangle$ (explore) with high probability. (2) The agent will execute a policy π_t whose value is better or not much worse than the teacher's policy π_T (exploit). (3) The teacher's trace will encounter an unknown $\langle s, a\rangle$ (explain) with high probability.*

Algorithm 2 MBP-Agent

The agent knows ϵ, δ, and A and has access to the environment E, teacher T, and a planner P.
Initialize MBP learners $L_T(\epsilon_T, \delta)$ and $L_R(\epsilon_R, \delta)$
for each episode **do**
  $s_0 = E.startState$
  while episode not finished **do**
    $a_t = P.getPlan(s_t, L_T, L_R)$.
    $\langle r_t, s_{t+1}\rangle = E.executeAct(a_t)$
    $L_T.Update(s_t, a_t, s_{t+1}); L_R.Update(s_t, a_t, r_t)$
  if T provides trace τ starting from s_0 **then**
    $\forall\langle s, a, r, s'\rangle$ Update $L_T(s, a, s')$ and $L_R(s, a, r)$

Lemma 1 proves Theorem 1 because MBP can only make a polynomial number of mistakes, meaning cases (1) and (3) can only happen a polynomial number of times. Below is a sketch of the lemma's proof.

Proof. The quantity $V^{\pi_T}(s_0)$ is the value, in the real environment, of the teacher's policy and $V^{\pi_A}(s_0)$ is the value, in the real environment, of the agent's current policy. Analogously, we can define $U^{\pi_T}(s_0)$ as the value, in the agent's learned MDP, of the teacher's policy and $U^{\pi_A}(s_0)$ is the value, in the agent's learned MDP, of the agent's policy.

First, note that with proper settings of ϵ_T and ϵ_R we can guarantee that $||V^{\pi} - U^{\pi}|| \leq \frac{\epsilon}{2}$. Then, by any of several simulation lemmata, such as Lemma 12 from Strehl et al. (2009), if $|U^{\pi_A}(s_0) - V^{\pi_A}(s_0)| > \frac{\epsilon}{2}$, then, with high probability, case (1), explore, will happen. That is because executing π_A in the real environment will produce a sample of $V^{\pi_A}(s_0)$ and the only way it can be different from the agent's conception of $(U^{\pi_A}(s_0))$, is if an unknown $\langle s, a\rangle$ is reached with sufficiently high probability.

Next, we consider the case where $U^{\pi_A}(s_0)$ and $V^{\pi_A}(s_0)$ are within $\frac{\epsilon}{2}$ of one another. If $V^{\pi_A}(s_0) \geq V^{\pi_T}(s_0) - \epsilon$, that means π_A is nearly optimal relative to π_T, and case (2), exploit, happens.

Finally, we consider the case where $U^{\pi_A}(s_0)$ and $V^{\pi_A}(s_0)$ are within $\frac{\epsilon}{2}$ of one another *and* $V^{\pi_A}(s_0) < V^{\pi_T}(s_0) - \epsilon$. Note that $U^{\pi_A}(s_0) \geq U^{\pi_T}(s_0)$ (because π_A was chosen as optimal). Chaining inequalities, we have $U^{\pi_T}(s_0) \leq U^{\pi_A}(s_0) \leq V^{\pi_A}(s_0) + \frac{\epsilon}{2} < V^{\pi_T}(s_0) - \frac{\epsilon}{2}$. We're now in a position to use a simulation lemma again: since $|U^{\pi_T}(s_0) - V^{\pi_T}(s_0)| > \frac{\epsilon}{2}$,

then, with high probability, case (3), explain, will very likely happen when the teacher generates a trace. □

In summary, KWIK-learnable models can be efficiently learned in the autonomous RL case, but MB learning is insufficient for exploration. MBP covers both of these classes, and is sufficient for efficient apprenticeship learning, so models that were autonomously learnable as well as many models that were formerly intractable (the MB class), are all efficiently learnable in the apprenticeship setting. As an example, the combination lock described earlier could require an exponential number of tries using a KWIK learner in the autonomous case, and MB is insufficient in the autonomous case because it does not keep track of what combinations have been tried. But in the apprenticeship setting, the MBP-Agent can get the positive examples it needs (see Section 4.2) and will succeed with at most n (one for each irrelevant tumbler) valid traces.

4. Example Domain Classes

We now present upper bounds and constructive algorithms for efficient apprenticeship learning in several widely used RL representations. We note that while some of these classes are efficiently learnable in the autonomous case, others are provably intractable.

4.1. Flat and Linear MDPs

In the autonomous setting, flat MDPs (where the states are simply propositional members of a set S) can be learned with a KWIK bound of $\tilde{O}(\frac{S^2A}{\epsilon^2})$ (Li et al., 2008). Following Theorem 1, this gives us a polynomial PAC-MDP-Trace bound, a result that is directly comparable to the apprenticeship-leaning result under the earlier protocol described by Abbeel & Ng (2005). One difference between the two protocols is that theirs requires all traces to be given before learning starts and our learners are oblivious to when they have met their goal of matching or exceeding the teacher.

The same work considered apprenticeship learning of linear dynamics. We note that these domains are also covered by Theorem 1 as recent results on KWIK Linear Regression (Walsh et al., 2009) have shown that such n-dimensional MDPs are again KWIK learnable with a bound of $\tilde{O}(\frac{n^3}{\epsilon^4})$.

4.2. Classical MB Results

While the results above are interesting, flat and linear MDPs are known to be efficiently learnable in the autonomous case. We now describe two MBP-learnable classes that we will use as the backbone of RL algorithms for relational MDPs, some of which are intractable in the autonomous case.

As mentioned earlier, KWIK and MB are separable when learning monotone conjunctions[2] over n literals when the number of literals relevant to the conjunction (n_R) can be as many as n. In KWIK, conjunctions of size $k = O(1)$ are efficiently learnable: The system simply enumerates all n^k conjunctions of this size, predicts $\perp$ unless all the hypotheses agree on a prediction, and eliminates wrong hypotheses. However, when the conjunction is of size $O(n)$, the $O(2^n)$ hypothesis space can result in an exponential number of $\perp$ predictions. This situation arises because negative examples are highly uninformative. In the combination lock, for example, the agent has no idea which of the 2^n settings will allow the lock to be unlocked, so it must predict $\perp$ at every new combination. Note though that if it does see this one positive example it will have learned the correct combination.

In contrast, learners in the MB setting can efficiently learn conjunctions by exploiting this asymmetry. Specifically, an MB agent for conjunction learning (detailed in (Kearns & Vazirani, 1994)) can maintain a set of literals $l_j \in L_H$ where $l_j = 1$ for every positive example it has seen before. If $\forall(l_j \in L_H), l_j = 1$ in x_t, the agent correctly predicts *true*, otherwise it defaults to *false*. By using such defaults, which KWIK agents cannot, and by only counting the highly informative positive samples, each of which subtracts at least one literal from L_H, polynomial sample efficiency is achieved.

Another class that is MB learnable is the class of k-term-DNF (disjunctive normal form of $k = O(1)$ terms). k-term-DNF are of the form $(l_i \wedge l_j \wedge ...)_1 \vee ... \vee (... \wedge ...)_k$, that is, a disjunction of k terms, each of at most size n. This class of functions is known to be MB learnable (Kearns & Vazirani, 1994) by creating a conjunction of new literals, each representing a disjunction of k original literals (for $k = 3$ we would have $l_{ijm} = l_i \vee l_j \vee l_m$), and then using the conjunction learning algorithm described above.

4.3. Learning Stochastic STRIPS Operators

We now describe a relational MDP class that, when given some background information about the environment's possible dynamics, can be MBP learned using *MB-partition* with an MB conjunction learner, and a KWIK learner. The class is Stochastic STRIPS with rewards (Walsh et al., 2009), where states are com-

[2]The results extend to the non-monotone setting using the standard method of including all negated literals.

Table 1. A Stochastic STRIPS rule in Blocks World.

PutDown(B, To): *Reward* = −1
PRE: *Holding*(B) ∧ *Clear*(To) ∧ *Block*(To)
$\omega_1(p_1 = 0.8)$:
ADD: *On*(B, To) , *EmptyHand*()
DEL: *Holding*(B) *Clear*(To)
$\omega_2(p_2 = 0.2)$:
ADD: ∅ **DEL:** ∅

prised of objects O, and predicates P (e.g. On(b, c)). Actions $a \in A$ (see Table 1) are parameterized and their dynamics are described by two parts. First, a *pre-condition* c^a, which is a conjunction (over P and the action's parameters) that determines whether the action will actually execute in the current state or return a unique "failure" signal. The second part of an action's description is a set of possible effects Ω^a, where each ω_i^a is a pair of "Add" and "Delete" lists describing the changes to the current state and has an associated probability $p_i^a \in \Pi^a$. Previous work has established that while the probabilities of these outcomes cannot be efficiently learned by traditional "counting" methods, they can be KWIK learned using a linear regression technique (Walsh et al., 2009). This approach is necessary because sometimes the difference between states s_t and s_{t+1} are explainable by several of the effects. Given this result and the conjunction learning algorithm above, we have the following result:

Proposition 3. *Stochastic STRIPS operators are PAC-MDP-Trace learnable if the agent is given the set of possible effects (Ω^a, but not Π^a) beforehand by using Algorithm 2 and MB-partitioning between an MB conjunction learner (for the preconditions) and a KWIK-LR learner (for the effect probabilities).*

Proof. Each transition sample is either a "failure" (c^a is $false$) or a transition that returns a next state s' (without a failure signal), so the output spaces are disjoint as required by *MB-partition*. We use a conjunction learner for each action (*MB-CON*^{a}) to predict whether the preconditions of a grounding of that action hold. As in earlier work on deterministic STRIPS (Walsh & Littman, 2008), these learners produce a pessimistic version of the preconditions, the most specific hypothesis possible on the conjunction. Thus, unless a series of actions exists that, with some probability, lead to the goal without failure of these pessimistic preconditions, the agent will request a trace by acting randomly for H steps (or ending the episode if possible). Each τ received because of such a request or some other suboptimal policy (with respect to the teacher) will provide positive examples of the preconditions, updating each *MB-CON*^{a} so no more than $|A|n$ traces will be needed where n is the number of literals within the action's scope.

The other part of the partition is learning each p_i^a, which is done separately from the conjunction learning with a mixture of real experience and trace tuples and has a known KWIK bound of $\tilde{O}(\frac{|A||\Omega|^3}{\epsilon^4})$. Thus, given Ω^a, the dynamics are MBP learnable, and thus the domain can be PAC-MDP-Trace learned. □

Table 2. An OOMDP operator for walking right with a limit of $x = 5$.

MoveRight(Obj, Loc): *Reward* = −1
c_1: *ClearToRight*(Loc) ∧ *GoodFooting*(Obj, Loc)
ω_1 : Obj1.x = min(2 + Obj1.x, 5)
c_2: *ClearToRight*(Loc) ∧ *WetFloor*(Obj, Loc) ∧ *Freezing*(Loc)
ω_2 : Obj1.x = min(1 + Obj1.x, 5)
c_3: *WallToRight*(Loc)
ω_3 : Obj1.x = Obj1.x

We note that these results generalize the findings of Walsh et al. (2009) (which did autonomous learning of only the probabilities) and Walsh & Littman (2008), which used an MB-like algorithm to prove the efficiency of "Trace-Learning" *deterministic* STRIPS operators. The use of a conjunction learner in this case relies on a unique failure signal when the preconditions of an action fail. We now investigate a different type of relational MDP, with conditional outcomes that do not provide this signal.

4.4. Deterministic OOMDP Operators

Object-oriented MDPs (Diuk et al., 2008) or OOMDPs are made up of objects with attributes (e.g. *agent6.location* = 15) and predicates that must be defined in terms of these attributes (e.g. *On*(A,B): $A.y = B.y + 1$). Actions (as in Table 2) are described by condition-effect pairs $\langle c_i^a, \omega_i^a \rangle$ such that in state s_t, the condition (a conjunction over the predicates) that holds (conditions may not overlap) governs which effect occurs. The effects themselves describe changes to the objects' attributes.

We now consider the problem of learning each c_i^a given the $k = O(1)$ possible effects $(\omega_1^a ... \omega_k^a)$ for each action. Notice that this case is different than the precondition learning done in the Stochastic STRIPS case because there is no longer a single failure signal for a c_i^a not matching s_t. Moreover, state transitions might not always allow the learner to unambiguously determine which effect occurred. For instance, invoking the

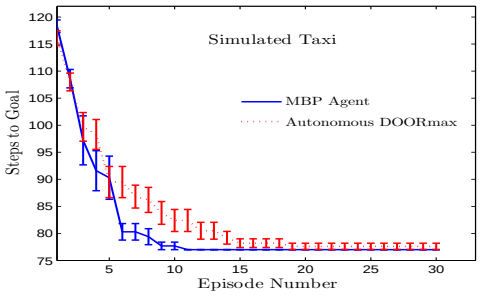

Figure 1. A KWIK learner (autonomous) and an MBP learner (apprenticeship) in the Taxi Domain

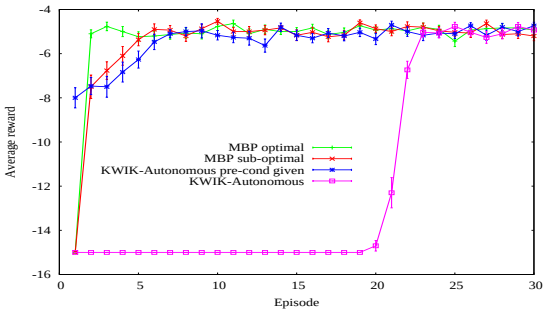

Figure 2. Autonomous learners and apprenticeship learners in a noisy 3-blocks world. Traces were given after each episode and the results were averaged over 30 trials.

MoveRight action (Table 2) when $o1.x = 4$, and then observing $o1.x = 5$, does not tell us which of (ω_1, ω_2) actually occurred, so it is not immediately clear which c_i^a, should be updated. Here we give a solution in the apprenticeship setting using an MB k-term-DNF learner.

Proposition 4. *Deterministic OOMDP conditions are PAC-MDP-Trace learnable if the agent is given Ω^a for each a beforehand by using Algorithm 2 and an MB k-term-DNF learner.*

Proof. The "trick" here is instead of representing the condition that causes an effect to occur, we learn the conditions that do not cause effect ω_i, which is $c_1 ... \vee c_{i-1} \vee c_{i+1} \vee ..c_k$. Since each c_j is an arbitrarily sized condition, we are learning a k-term-DNF for each condition *not* occurring. While extra steps are needed to to negate experience tuples and interpret the predictions, this insight gives us the desired result. □

This result extends the previous (KWIK) sample complexity results for autonomous OOMDPs learning, which limited the size of each c_i^a to $O(1)$. It is an open question as to whether this particular approach can be extended to the stochastic setting.

5. Experiments

Our first experiment is in a simulated deterministic "Taxi" domain (from Diuk et al. (2008)): a 5×5 grid world with walls, a taxi, a "passenger" and 4 possible destination cells. The agent must learn about navigation and the conditions for picking up and dropping off a passenger. We use the deterministic OOMDP representation from above, but to accommodate the autonomous agent both learners used conjunction learning routines (*KWIK-Enumeration* and *MB-Con*) and effects were constructed to be unambiguous. Figure 1 shows MBP-Agent (using *MB-Con*) with apprenticeship learning reaching optimal behavior ahead of the current state-of-the-art autonomous approach, DOORMAX (Diuk et al., 2008), a KWIK-based algorithm. Here, and in the later robot version, traces were provided at the end of each episode if the agent's policy was worse than the teacher's. Each trial in Figure 1 used 4 to 7 such traces.

Next, we consider a Stochastic STRIPS noisy-blocks world from Table 1. There are also two "extra" pickup/putdown actions that do nothing with probability 0.8. Figure 2 shows 4 agents in a 3-blocks version of this domain. The MBP-Agent with an optimal teacher learns the preconditions and to avoid the extra actions from a single trace. The MBP-Agent with a suboptimal teacher (who uses the extra actions 50% of the time) eventually learns the probabilities and performs optimally in spite of its teacher. Both MBP agents efficiently learn the preconditions. In contrast, a KWIK learner *given* the preconditions mirrors the suboptimal-trace learner, and a KWIK learner for both the preconditions and probabilities requires an inordinate amount of exploration to execute the actions correctly. We recorded similar results with 4-blocks although 2 optimal traces are needed.

Our last experiment was a version of the taxi environment above, but now on a physical Lego Mindstorms^{TM} robot. A human demonstrated traces by controlling the robot directly. Unlike the simulated version, the real-world transitions are stochastic (due to noise). So in this experiment, the robot was given all the conditions it might encounter and had to learn the effects and probabilities of the actions. We provided traces after each episode until it completed the task twice on its own from random start states. We ran this experiment in two different settings: one where the dynamics of the environment's borders (physical walls) were given, and one where the robot had to learn about behavior near walls. The first setting contained 6 condition settings and required 5 traces and a total

of 159 learning steps. In the harder setting, there were 60 conditions to learn and the agent required 10 traces and 516 learning steps. This experiment demonstrates the realizability and robustness of our apprenticeship-learning protocol in real world environments with sensor imprecision and stochasticity.

6. Related Work and Conclusions

A number of different protocols and complexity measures have been described for apprenticeship learning in RL. The one used in this paper is an extension of the interaction described by Abbeel & Ng (2005) with the change that teachers no longer have to give all of their traces up front. This feature, along with the description of a large class of transition functions learnable under the apprenticeship paradigm, separates our work from previous efforts to train robots starting from traces (Smart & Kaelbling, 2002) and other systems for learning from demonstration (Chernova & Veloso, 2009), though the use of learned models in the latter is similar to ours. The field of Inverse Reinforcement Learning (Abbeel & Ng, 2004), also sometimes called apprenticeship learning, attempts to learn an agent's reward function by observing a sequence of actions (not rewards) taken by a teacher. In contrast to this interaction, our learners actually see samples of the transitions *and* rewards collected by the teacher and use this "experience" in a traditional model-based RL fashion. Recent work in Imitation Learning (Ratliff et al., 2006) took MDP instances and trajectories and tried to generalize these behaviors based on assumptions about the linearity of costs with respect to the feature space. Work on policy cloning (Khardon, 1999) used a protocol close to ours for observing traces, though the goal (learning a particular policy) was different.

In this work, we have extended the previous protocol for apprenticeship learning, defined a measure of sample complexity for this interaction (PAC-MDP-Trace), and shown that two large and widely studied classes of models (KWIK and MB and a combination of the two) are sufficient for PAC-MDP-Trace learning. We have shown how to use these findings constructively to learn and act in RL environments that were otherwise intractable, including two forms of relational MDPs. We have also provided empirical evidence in two simulated domains and a demonstration of efficient apprenticeship learning on a real robot.

Acknowledgments

We thank Roni Khardon for helpful comments and DARPA IPTO FA8650-06-C-7606 for funding.

References

Abbeel, Pieter and Ng, Andrew Y. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

Abbeel, Pieter and Ng, Andrew Y. Exploration and apprenticeship learning in reinforcement learning. In *ICML*, 2005.

Angluin, Dana. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1988.

Chernova, Sonia and Veloso, Manuela. Interactive policy learning through confidence-based autonomy. *JAIR*, 34(1):1–25, 2009.

Diuk, Carlos, Cohen, Andre, and Littman, Michael L. An object-oriented representation for efficient reinforcement learning. In *ICML*, 2008.

Kearns, Michael J. and Vazirani, Umesh V. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994.

Khardon, Roni. Learning to take actions. *Machine Learning*, 35(1):57–90, 1999.

Li, Lihong, Littman, Michael L., and Walsh, Thomas J. Knows what it knows: A framework for self-aware learning. In *ICML*, 2008.

Littlestone, Nick. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1987.

Ratliff, Nathan D., Bradley, David M., Bagnell, J. Andrew, and Chestnutt, Joel E. Boosting structured prediction for imitation learning. In *NIPS*, 2006.

Smart, William D. and Kaelbling, Leslie Pack. Effective reinforcement learning for mobile robots. In *ICRA*, 2002.

Strehl, Alexander L., Li, Lihong, and Littman, Michael L. Reinforcement learning in finite MDPs: PAC analysis. *JMLR*, 10(2):413–444, 2009.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, March 1998.

Walsh, Thomas J. and Littman, Michael L. Efficient learning of action schemas and web-service descriptions. In *AAAI*, 2008.

Walsh, Thomas J., Szita, István, Diuk, Carlos, and Littman, Michael L. Exploring compact reinforcement-learning representations with linear regression. In *UAI*, 2009.