Active Imitation Learning via State Queries

Kshitij Judah
Alan Fern
Thomas Dietterich
School of EECS, Oregon State University

Abstract
We consider the problem of active imitation learning. In passive imitation learning, the goal is to learn a target policy by observing full trajectories of it. Unfortunately, generating such trajectories requires substantial effort and can be impractical in some cases. Active imitation learning reduces this effort by querying the teacher about individual states. Given such a query, the teacher may either suggest an action for the state or declare that the state is “bad” in the sense that the teacher’s policy would never go there. Standard active-learning techniques, do not account for the state-visitation likelihood of the target policy, and hence can perform poorly by asking many “bad” queries. We describe a new approach to this problem that is inspired by viewing query selection in the framework of Bayesian active learning, resulting in a version of Query-by-Committee for active imitation learning. Our experiments in two test domains show promise for our approach compared to a number of alternatives.

1. Introduction
Traditional imitation learning involves learning a policy that performs nearly as well as a target policy from a set of trajectories of the target. Generating a set of such trajectories can often be tedious or even impractical (e.g. real-time low-level control of multiple game agents). We consider active imitation learning where full trajectories are not required, but rather the learner queries the teacher about specific states. The teacher can either suggest an action in a query state or if the teacher would never expect to encounter such a state they may declare the state as “bad”. The goal is to learn a policy as good as the teachers with as few queries as possible.

There is much work on i.i.d. supervised active learning (Settles, 2009). Active learning has somewhat also been studied in the reinforcement learning (RL) setting (Clouse, 1996; Mihalkova & Mooney, 2006; Gil et al., 2009; Doshi et al., 2008), where learning is based on both autonomous exploration and queries to a teacher. However, we do not assume rewards and learn only from teacher queries. More closely related is recent work on confidence based autonomy (Chernova & Veloso, 2009) (and dogged learning (Grollman & Jenkins, 2007)), where a policy is learned as it is executed. When the learner is uncertain about what to do at a particular state the policy is paused and the teacher is queries about what action to take, which results in an updated policy. One difficulty in applying this approach is setting the uncertainty threshold for querying the teacher. Prior work (Chernova & Veloso, 2009) suggests an automated threshold selection approach, however, our experiments show that it is not always effective. Other prior work (Shon et al., 2007) studies active imitation learning in a multiagent setting where the teacher is itself a reward seeking agent and hence not necessarily helpful. We focus on the case of an always helpful teacher.

We propose a new approach to active imitation learning inspired by the framework of Bayesian active learning and leads to a new variant of the query-by-committee approach for imitation learning.

2. Problem Formulation
We consider active imitation learning in the framework of Markov decision processes (MDPs). An MDP is a tuple \((S, A, T, R, s_0)\), with state set \(S\), action set \(A\), transition function \(T(s, a, s')\) denoting the probability of transitioning to state \(s'\) after taking action \(a\) in state \(s\).
where \( R(s) \) is the reward function giving the reward in state \( s \), and \( s_0 \in S \) is an initial state. A deterministic MDP is one where each state and action lead to exactly one other state. A stationary policy is a mapping from states to actions \( \pi : S \mapsto A \), such that \( \pi(s) \) indicates the action to take in state \( s \) when following \( \pi \). The \( H \)-horizon value of a policy is the expected total reward of trajectories that start in \( s_0 \) and then follow the actions of \( \pi \) for \( H \) steps. For MDPs with large state spaces, we will assume a parameterized space of policies, where \( \pi_\theta \) denotes the policy with parameters \( \theta \).

In imitation learning, a teacher has an unknown target policy \( \pi_T \), and the goal is to learn a policy \( \pi_\theta \), whose \( H \)-horizon value is not much worse than that of \( \pi_T \). In the passive setting, the learner is provided with a set of full trajectories of \( \pi_T \) starting in \( s_0 \). To help avoid the cost of generating full trajectories, active imitation learning allows the learner to pose state queries. In a state query, the state \( s \) is presented to the teacher, and the teacher can respond by returning either (a) the action of the target policy \( \pi_T(s) \), or (b) a bad-state response \( \bot \), which indicates that the target policy \( \pi_T \) does not visit state \( s \).

In addition to having access to the teacher for answering queries, we assume that the learner has access to a simulator of the MDP. The input to the simulator is a policy \( \pi \), a horizon \( H \), and possibly a random seed. The simulator output is a state trajectory that results from executing \( \pi \) for \( H \) steps starting in \( s_0 \). The learner interacts with this simulator as part of its query selection process. The simulator never provides a reward signal, which means that the learner cannot find \( \pi \) by pure reinforcement learning.

**Bad State Responses.** The distinguishing aspect of this active learning model is the possibility of bad-state responses. In practice, allowing for such responses is important, due to the fact that there are no constraints on which states the learner is allowed to present to the teacher. In particular, certain learners might ask queries about states that the teacher would never encounter. In such cases, the teacher is likely to be uncertain or agnostic about the action, since the choice does not arise for the teacher or is unimportant. For example, if the goal is to learn to fly a helicopter, queries in states where the helicopter is in an unrecoverable fall would likely be considered to be bad states by a teacher. Here it is preferable to allow the teacher to convey such states as bad to the learner rather than force them to provide an arbitrary action label.

Our algorithms do not assume that the teacher will always provide bad state responses in such situations; they simply allow them as an option. However, there are a number of reasons why it is desirable to attempt to minimize the number of queries asked in such states. First, a teacher is likely to become frustrated when a large percentage of queries are in bad states. Second, if actions are suggested in such states, the advice may not be as reliable. Third, if actions are provided in bad states, learning a policy that covers both good and bad states may require a more complex policy representation. Fourth, it is difficult to incorporate the information contained in such bad state responses into standard classification learning algorithms. Thus, one of the novelties of our proposed algorithm is to attempt to minimize the number of such queries.

**Relation to Passive Imitation Learning.** In our formulation, it is easy to use the simulator and state queries to generate a data set suitable for a passive imitation learner. In particular, trajectories of \( \pi_T \) can be generated by passing the simulator a policy that always queries the teacher about the input state and returns the resulting action. Thus, a baseline approach to active imitation learning is to simply generate a set of such trajectories and then apply any passive imitation learning algorithm. One positive aspect of this approach is that it completely avoids bad-state responses. However, this comes at a very high interaction cost for the teacher, who is queried at each step of a trajectory. Intuitively, an active learner can potentially reduce the number of queries by not asking about states where confidence is high given previous queries.

**Relation to I.I.D. Active Learning.** In active learning for i.i.d. classification, the learner is typically given access to a target distribution over inputs (in our case states). The active learner must then decide which of those points to query. In the case of active imitation learning, the target state distribution corresponds to the distribution induced by the unknown target policy and is not directly available. This complicates the direct application of i.i.d. active learners in our setting. In concept, one could assume a uniform distribution over the state space and apply an i.i.d. active learning algorithm. However, as our experiments indicate, this approach will often produce poor results.

**3. Imitation Query-by-Committee**

We first describe the problem of noiseless Bayesian active learning, then we describe our approach first for deterministic MDPs and then for the general case.

**Noiseless Bayesian Active Learning.** Bayesian active learning assumes a finite set of hypotheses \( \mathcal{H} = \{ h_1, \ldots, h_n \} \), each of which is a mapping from a finite set of tests \( \mathcal{T} = \{ x_1, \ldots, x_N \} \) to a finite set of
values $\mathcal{X}$. In the noiseless case, these mappings are deterministic, and each pair of hypotheses is assumed to differ on at least one test. Thus, hypotheses can be uniquely determined by observing the values of all tests. The Bayesian active learning problem specifies a prior distribution $P(h)$ over $\mathcal{H}$ from which a hypothesis is drawn. The goal is to compute a strategy for selecting a sequence of tests to query in order to quickly identify the unknown hypothesis. Specifically, we would like a strategy that minimizes the expected number of queries needed to identify the hypothesis. Computing an optimal strategy is NP-hard in general. However, a number of effective heuristics with approximation guarantees are available. We employ the generalized binary search heuristic as described later.

**Deterministic MDPs.** In a deterministic MDP, each policy corresponds to a unique state trajectory of length $H$ (the horizon) starting at $s_0$. Hence, here we will use the terms policy and trajectory interchangeably. Note that for parametric policy representations, there will in general be a many-to-one mapping from policy parameters to paths. We can pose active imitation learning as Bayesian active learning as follows.

The hypothesis set is the set of all paths $\mathcal{H} = \{p_1, \ldots, p_{|s|^{H-1}}\}$ of length $H$. Our goal is to determine the path $p^T$ corresponding to the teacher policy $\pi^T$ by performing a series of tests (queries to the teacher) that can have multiple outcomes (teacher responses). The set of possible tests is simply the set of MDP states, i.e., $T = S$, and the possible values for a test are either an action or the bad-state response $\perp$. The value assigned to a test $s$ by a hypothesis path corresponding to policy $\pi$ is either the action $\pi(s)$, if $\pi$ goes through $s$, and otherwise $\perp$ if $\pi$ does not go through $s$. Finally, the prior over hypothesis paths could be taken to be uniform or, for parametric policies, induced by a prior over policy parameters.

Given this formulation, one could directly apply any heuristic for Bayesian active learning in order to select queries for an active imitation learning problem. It is illustrative to consider how this formulation is naturally drawn to ask queries that are likely to be on the path of the target policy and hence avoid bad-state responses. In order to compute a query, typical heuristic policies first compute the posterior hypothesis distribution given the results of all prior queries. A heuristic is then applied to select a query that is maximally informative about discriminating the true hypothesis. For active imitation learning, if a potential query state is unlikely to be encountered by a policy according to the posterior, then that state will not be very discriminating, since most policies will agree that the test outcome should be $\perp$. Hence such state are not likely to be queried, which agrees with intuition.

**Query-by-Committee for Large MDPs.** Unfortunately, the above approach is only be practical for small MDPs and horizons, since the number of tests is equal to the number of states, and the hypothesis space size is exponential in the horizon. To address this, we introduce an approximation called imitation query-by-committee (IQBC). First, instead of dealing with all possible hypothesis paths, we sample a manageable subset, or committee, of $K$ paths (details below) that is treated as an exemplar-based representation of the current posterior over hypotheses. Second, instead of considering the entire state-space as the possible tests, we only consider tests corresponding to states that lie on at least one of the $K$ paths. Given this representative set of hypotheses and tests, we then employ the generalized binary search (GBS) heuristic to select a query state. GBS selects a state where the hypotheses maximally disagree on the possible outcomes. We employ the entropy of the vote distribution as a measure of disagreement (Dagan & Engelson, 1995): $VE(s) = -\sum_{x \in \mathcal{X}} \frac{V(s,x)}{K} \log \frac{V(s,x)}{K}$ where $V(s,x)$ is the number of committee members that vote for outcome $x$ on $s$.

In our implementation, we generate the set of representative paths via a two step process. First, let $\mathcal{D}$ be the data set consisting of the outcomes of all previous queries. We generate a committee of parametric policies based on $\mathcal{D}$ using a standard bagging approach (Breiman, 1996). Multiple bootstrap samples of $\mathcal{D}$ are created, and from each policy is learned via supervised learning. This produces a set of policies with parameters $\{\theta_1, \ldots, \theta_K\}$, one for each bootstrap sample. Each of the $K$ committee policies is then executed on the simulator to generate a total of $K$ paths. This approach assumes a procedure for learning a policy from a set of query-response pairs. In our implementation we apply standard classification learners to learn a mapping from states to actions. Such learners, however, do not naturally handle queries labeled by the $\perp$ response. Those data points are ignored by our learner and simply memorized in order to avoid repeated queries. Incorporating $\perp$ responses is important future work.

**Stochastic MDPs.** A naive way of applying our approach for deterministic MDPs to the stochastic case is to run each policy in the committee once to generate a trajectory and use the resulting set of paths as the hypothesis set. However, this approach is very unreliable, especially in a highly stochastic MDP where the state visitation of a policy is highly influenced by the ran-
is not possible to immediately exit. Rather in a trap adjacent to a trap. However, once the agent enters a trap, it is impossible to pass through walls. There is a solid wall between each adjacent pair of non-trap rooms.

Grid World with Traps. This domain is a 30x30 deterministic grid world divided into 9 10x10 rooms, where each room is either of type trap or non-trap. In each location, the agent can move left, right, up, or down and is not allowed to pass through walls. There is a solid wall between each adjacent pair of non-trap rooms with a two-way door in the middle of the wall. There is no wall between a non-trap room and an adjacent trap. However, once the agent enters a trap, it is not possible to immediately exit. Rather in a trap room, the agent must travel to the lower-left corner of the room from where it is teleported back to the start state of the grid. The teacher’s target policy specifies a shortest path, avoiding all trap rooms, from a start location in the upper-left non-trap room to the lower-right non-trap room. Episodes terminate when the goal is reached or at a horizon of 60 steps. We use a non-stationary reward function, which for state \( s \) and time \( t \) assigns a small negative reward proportional to the distance between \( s \) and the target trajectory at time \( t \). However, if \( s \) is in a room that is not visited by the target trajectory (such as a trap room), there it receives a large negative reward.

The policy is represented as a linear logistic regression classifier over features of state-action pairs. The feature vector for a state-action pair is defined as follows: There are four features for each room, but all features are zero except those corresponding to the current room. The value for the \( i \)'th component of the current room is \(+1\) (\(-1\)) if taking action \( a \) in state \( s \) increases (respectively, decreases) the distance to the center of the \( i \)'th side of the room. A door is located in the center of each wall. Using this feature representation, it is easy to learn a policy for moving toward each of the doors when in a particular room.

We run experiments on 16 problem variants that differ in terms of the start location in the initial room. Each experiment provides each learner with an initial training set consisting of the first 5 state-action pairs on the target path (equivalent to 5 steps of passive imitation learning). Each learner is then allowed to pose 55 additional queries. After each query, the current policy is evaluated (for purposes of the experiment) by running it in the simulator and reporting the total reward. This provides a learning curve showing total reward versus number of queries. We report the average learning curve across the 16 initial starting states.

We run experiments with three types of teachers that differ in their bad-state responses. Given a query \( s \) all of the teachers give a bad-state response \( \bot \) if \( s \) is more than \( \tau \) cells away from the target path. Otherwise, the teacher responds by giving the target action in \( s \). Our first teacher uses \( \tau = \infty \), meaning that the teacher always responds with an action rather than with a bad-state response. In states off of the target path, the action suggested is on the shortest path to the goal. The two other teachers use \( \tau = 1 \) and \( \tau = 2 \), which means that they are unwilling to provide guidance outside of a small region near the target path.

Figures 1a-c show the learning curves for the grid world when responses are provided by teachers using \( \tau = \infty \), \( \tau = 2 \), and \( \tau = 1 \) respectively. We see that...
when the teacher always suggests an action ($\tau = \infty$), IQBC starts to learn immediately and improves with each subsequent query. Passive, on the other hand, has plateaus in its learning curve and hence learns more slowly overall. The reason for the plateaus is that passive asks queries throughout its trajectory in a room and does not learn about the next room until actually arriving their, meaning that the overall policy performance will be poor. IQBC on the other hand learns how to act in a room after a small number of queries in the room and then immediately begins asking queries in the next room on the target path, rather than continuing to ask queries in the “already learned” room. This illustrates the ability of IQBC to better leverage generalization along a target path compared to passive. When such generalization is not possible, then passive is likely to be difficult to improve upon.

Random and SQBC are comparably much worse than IQBC and Passive in this domain. This is because these methods ask many queries that are not relevant to the target path, in particular, many queries in trap rooms. Those queries are useless for learning the target policy. In addition, in this domain it is not possible for the policy representation to capture the shortest path policy in trap rooms (go to the lower-left corner), which has the effect of making states in those room look quite uncertain. This in turn leads SQBC to prefer querying states in such rooms.

Finally, the performance of CBA was quite poor in this domain. In particular, the automatic threshold mechanisms of CBA turns out to not work robustly in this domain. At some point in the learning curve, the threshold is set so that CBA decides to not ask any further queries. In each such case the policy is highly sub-optimal. This illustrates the sensitivity of the CBA approach to the threshold parameter. While it might have been possible to design a new threshold strategy for this domain, the choices appears to be highly domain specific.

Now consider the results for $\tau = 2, 1$, where the teacher provides bad-state responses. The performance of IQBC decreases to some degree and becomes more comparable to Passive. The decreased performance for IQBC is due to the fact that it is not able to avoid all bad-state responses and for such queries does not obtain useful training data. As we move from a less strict ($\tau = 2$) to a more strict ($\tau = 1$) teacher we see that performance of IQBC degrades more, noting that $\tau = 1$ is an extremely strict teacher that will only answer queries directly on the target path. The performance of the other methods are equally poor for these stricter teachers as for $\tau = \infty$.

**CartPole.** Our second domain, known as cart pole (or inverted pendulum), is a common toy reinforcement learning benchmark. In this domain, there is a cart on which rests a vertical pole. The objective is to keep the pole balanced by applying left or right force to the cart. An episode ends when either the pole falls or the cart goes out of bounds. A state is represented by four variables $(x, \dot{x}, \theta, \dot{\theta})$ describing the position and velocity of the cart and the angle and angular velocity of the pole. There are only two actions left and right. For our experiments, we made slight modifications to the usual setting where we allow the pole to fall down and become horizontal and the cart to go out of bounds (we used default $[-2.4, 2.4]$ range as the bound for the cart). We let each episode run for a fixed length of 5000 time steps. This opens up the possibility of spending significant time in bad states when the pole has fallen or gone out of bounds. Again we represent the policy via a linear logistic regression classifier with features corresponding to state variables. The target teacher policy corresponded to a hand-coded policy that can balance the pole indefinitely.

We ran experiments form 30 random initial states close to the equilibrium start state $(0.0, 0.0, 0.0, 0.0)$. For each start state a policy is learned and a learning curve is generated measuring performance as function of number of queries posed to the teacher. In order to measure performance, we give +1 reward for each time step where the pole is kept balanced and the cart is within bounds and -1 reward for each time step where
pole has fallen or the cart is out of bounds. The final learning curve is the average of the individual curves. We ran experiments for two types of teachers. The “always” teacher always returns an action for any query even in “bad states”. The “never” teacher, never provides actions in bad states, but instead returns ⊥. We defined a bad state as one for which a state variable falls outside of the range of that state variable visited by the teacher policy.

Figures 2a-b shows the learning curves for the cart pole for the “always” and “never” teacher respectively. For the “always” teacher, IQBC performs best among all the algorithms. Passive, SQBC and Random approach optimal performance as well but at a slower rate. CBA on the other hand tends to settle on a suboptimal performance because it often gets confident prematurely and stops asking further queries. A close look at the states that were queried by IQBC reveals that IQBC tends to focus on states very close to the states visited by the teacher’s policy. In fact, IQBC never queries at any state that is outside the bounds of the state variables of states that the teacher policy visits. Hence IQBC stays close to the part of the state space visited by the teacher. Furthermore, IQBC tends to query states close to the decision boundary where there is maximum uncertainty about the correct action. This allows IQBC to gather labels on these uncertain states and to converge more quickly to the true decision boundary compared to passive.

For the “never” teacher, we see that the performance of IQBC remains largely unaffected. The other active learners experience a delay in learning mainly due to bad state responses from the teacher. Both Random and SQBC seem to settle at a suboptimal performance of 3500 at the end of the learning curve suggesting that more time is needed to learn an optimal policy with a “never” teacher. Finally CBA performs poorly with the “never” teacher indicating that most of the queries were far from the teacher’s trajectory, again highlighting a limitation of CBA that it can confidently execute actions that can take it far away from the states of interest before querying the teacher.

5. Future Work

One area of future work is to develop policy optimization algorithms that take ⊥ responses and other forms of teacher input into account. We are also interested in posing queries that corresponding to short action sequences for the teacher to critique, rather than querying single states. We also plan to evaluate our approach on more realistic problems including control problems from real-time strategy games and learning policies for complex visual tracking problems. Finally, we plan to conduct experiments with human teachers, which is a critical aspect of this research agenda.

References


