time \( t = 1 \) and then execute \( \pi \) for \( T \) steps. We use \( d^*_t \) to denote the state distribution induced at time step \( t \) by starting in \( s_1 \sim I \) and then executing \( \pi \). Note that \( d^*_1 = I \) for all policies. We use \( d_a = \frac{1}{T} \sum_{t=1}^{T} d^*_t \) to denote the state distribution induced by policy \( \pi \) over \( T \) time steps. To sample an \((s,a)\) pair from \( d^*_t \), we start in \( s_1 \sim I \), execute \( \pi \) to generate a trajectory \( T = (s_1, a_1, \ldots, s_T, a_T, s_{T+1}) \) and set \((s,a) = (s_t,a_t)\).

Similarly, to sample from \( d_a \), we first sample a random time step \( t \in \{1, \ldots, T\} \), and then sample an \((s,a)\) pair from \( d^*_t \). Note that in order to sample from \( d^*_t \) (or \( d^* \)), we need to execute \( \pi^* \). Throughout the paper, we assume that the only way \( \pi^* \) can be executed is by querying the expert for an action in the current state and executing the given action, which puts significant burden on the expert.

The regret of a policy \( \pi \) with respect to an expert policy \( \pi^* \) is equal to \( V(\pi^*) - V(\pi) \). In imitation learning, the goal is to learn a policy \( \pi \) from a hypothesis class \( H \) (e.g. linear action classifiers), that has a small regret. In the passive setting of imitation learning, the learner is provided with a training set of full execution trajectories of \( \pi^* \) and the state-action pairs (or a sample of them) are passed to an i.i.d. supervised learning algorithm. To help avoid the cost of generating full trajectories, active imitation learning allows the learner to pose action queries. In an action query, a state \( s \) is presented to the expert and the expert returns the desired action \( \pi^*(s) \).

In addition to having access to the expert for answering queries, we assume that the learner has access to a simulator of the MDP. The input to the simulator is a policy \( \pi \) and a horizon \( T \). The simulator output is a state trajectory that results from executing \( \pi \) for \( T \) steps starting in the initial state. The learner is allowed to interact with this simulator as part of its query selection process. The simulator is not assumed to provide a reward signal, which means that the learner cannot find \( \pi \) by pure reinforcement learning.

Since our analysis in the next two sections is based on reducing to i.i.d. active learning and comparing to i.i.d. passive learning, we briefly review the Probably Approximately Correct (PAC) (Valiant, 1984) learning formulation for the i.i.d. setting. Here we consider the realizable PAC setting, which will be the focus of our initial analysis. Section 4.3, extends to the non-realizable, or agnostic setting. In passive i.i.d. supervised learning, \( N \) i.i.d. data samples are drawn from an unknown distribution \( D_X \) over an input space \( \mathcal{X} \) and are labeled according to an unknown target classifier \( f: \mathcal{X} \rightarrow \mathcal{Y} \), where \( \mathcal{Y} \) denotes the label space. In the realizable PAC setting it is assumed that \( f \) is an element of a known class of classifiers \( H \) and given a set of \( N \) examples a learner then outputs a hypothesis \( h \in H \). Let \( \epsilon_f(h, D_X) = \mathbb{E}_{x \sim D_X}[h(x) \neq f(x)] \) denote the generalization error of the returned classifier \( h \). Standard PAC learning theory provides a bound on the number of labeled examples that are sufficient to guarantee that for any distribution \( D_X \), with probability at least \( 1 - \delta \), the returned classifier \( h \) will satisfy \( \epsilon_f(h, D_X) \leq \epsilon \). We will denote this bound by \( N_a(\epsilon, \delta) \), which corresponds to the label/query complexity of i.i.d. passive supervised learning for a class \( H \). We will also denote a passive learner that achieves this label complexity as \( L_p(\epsilon, \delta) \).

In i.i.d. active learning, the learner is given access to two resources rather than just a set of training data: 1) A “cheap” resource (Sample) that can draw an unlabeled sample from \( D_X \) and provide it to the learner when requested, 2) An “expensive” resource (Label) that can label a given unlabeled sample according to target concept \( f \) when requested. Given access to these two resources, an active learning algorithm is required to learn a hypothesis \( h \in H \) while posing as few queries to Label as possible. It can, however, pose a much larger number of queries to Sample (though still polynomial) as it is cheap. We use \( N_a(\epsilon, \delta) \) to denote the label complexity (i.e. number of calls to Label) that is sufficient for an active learner to return an \( h \) that for any \( D_X \) with probability at least \( 1 - \delta \) satisfies \( \epsilon_f(h, D_X) \leq \epsilon \). Similarly, we will denote an active learner that achieves this label complexity as \( L_a(\epsilon, \delta, D) \), where the final argument \( D \) indicates that the Sample function used by \( L_a \) samples from distribution \( D \). It has been shown that often \( N_a \) can be exponentially smaller than \( N_p \) for hypothesis classes with bounded complexity (e.g. VC-dimension). In particular, in the realizable case, ignoring \( \delta \), \( N_p = O(\frac{1}{\epsilon}) \) whereas \( N_a = O(\log(\frac{1}{\epsilon})) \) giving exponential improvement in label complexity over passive learning.

4 Reductions for Active Imitation Learning

We now consider reductions from active imitation learning to active i.i.d. learning for the cases of deterministic non-stationary and stationary policies.

4.1 Non-Stationary Policies

Syed and Schapire (2010) analyze the traditional reduction from passive imitation learning to passive i.i.d. learning for non-stationary policies. The algorithm uses queries to sample \( N \) expert trajectories, noting that the state-action pairs at time \( t \) can be viewed as i.i.d. draws from distribution \( d^*_t \). The algorithm, then returns the non-stationary policy \( \hat{\pi} = (\hat{\pi}_1, \ldots, \hat{\pi}_T) \), where \( \hat{\pi}_t \) is the policy returned by running the learner \( L_p \) on examples from time \( t \). Let \( \epsilon_t = e_{\hat{\pi}^*_t}(\hat{\pi}_t, d^*_t) \) be the generalization error at time