No-Regret Reductions for Imitation Learning and Structured Prediction

Stéphane Ross
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
stephaneross@cmu.edu

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ggordon@cs.cmu.edu

J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dbagnell@ri.cmu.edu

Abstract

Sequential prediction problems such as imitation learning, where future observations depend on previous predictions (actions), violate the common i.i.d. assumptions made in statistical learning. This leads to poor performance in both theory and often in practice. Some recent approaches (Daume 2009; Ross 2010a) provide stronger performance guarantees in this setting, but remain somewhat unsatisfactory as they train either non-stationary or a stochastic policies and require a large number of iterations. In this paper, we propose a new iterative algorithm, which trains a stationary deterministic policy, that can be seen as a no regret algorithm in an online learning setting. We show that any such no regret algorithm, combined with additional reduction assumptions, must find a policy with good performance under the distribution of observations it induces in such sequential settings. We additionally show that this new approach outperforms previous approaches on two challenging imitation learning problem and a benchmark sequence labeling problem.

1 INTRODUCTION

Sequence Prediction problems arize commonly in practice. For instance, most robotic systems must be able to predict/make a sequence of actions given a sequence of observations releaved to them over time. In complex robotic systems where standard control methods fail, we must often resort to learning a controller that can make such predictions. Imitation learning techniques, where expert demonstrations of good behavior are used to learn a controller, have proven very useful in practice and have led to state-of-the art performance in a variety of applications (Schaal 1999, Abbeel 2004, Ratliff 2006, Silver 2008, Argall 2009 Chernova 2009, Ross 2010a). A typical approach to imitation learning is to train a classifier or regressor to predict an expert's behavior given training data of the encountered observations (input) and actions (output) performed by the expert. However since the learner's prediction affects future input observations/states during execution of the learned policy, this violate the crucial i.i.d. assumption made by all statistical learning approaches.

Ignoring this issue leads to poor performance both in theory and practice (Ross 2010a). In particular, a classifier that makes a mistake with probability ϵ under the distribution of states/observations encountered by the expert can make as many as $T^2\epsilon$ mistakes in expectation over T-steps under the distribution of states the classifier itself induces (Ross 2010a). Intuitively this is because as soon as the learner makes a mistake, it may encounter completely different observations than those under expert demonstration leading to a compounding or errors.

Recent approaches (Ross 2010a) demonstrate that it is possible to achieve better performance guarantees with an expected number of mistakes linear (or nearly so) in the task horizon T and classification error ϵ if training occurs over several iterations and the learner is allowed to influence the input states where expert demonstration is provided by executing its own controls in the system. One approach (Ross 2010a) involves learning a non-stationary policy by training a different policy for each time step in sequence starting from the first step, conditioned on the previously learned policies. Unfortunately this is impractical when the task horizon T is large or ill-defined. Another approach called SMILe (Ross 2010a), similar to SEARN (Daume 2009) and CPI (Kakade 2002), involves training a stationary stochastic policy (a distribution over a finite set of policies) over several iterations of training by adding a new learned policy to the mixture at each iteration of training. Learning a stochastic policy may be unsatisfactory for practical applications as some policies in the mixture are worse than others, the learned controller may be unstable, and further these methods require iterating that grows cubic

in T.

We propose a new meta-algorithm for imitation learning which learns a stationary deterministic policy guaranteed to perform well under its induced distribution of states (number of mistakes/costs that grows linearly in T and classification cost ϵ). We take a reduction-based approach (Beygelzimer 2005) that enables reusing existing supervised learning algorithms. Our approach is simple to implement, has no free parameters except the supervised learning algorithm sub-routine, and requires a number of iterations that scales nearly linearly with the effective horizon of the problem. It naturally handles continuous as well as discrete predictions. Our approach is closely related to no regret online learning algorithms (Cesa-Bianchi 2004, Hazan 2006, Kakade 2008) (in particular *Follow-The-Leader*) but better leverages the expert in our setting. Additionally, we show that any no-regret learner can be used in a particular fashion to learn a policy that achieves similar guarantees.

We begin by establishing our notation and setting, discuss related work, and then present the DAGGER (Dataset Aggregation) method. We analyze this approach using a no-regret and a reduction approach (Beygelzimer 2005). Beyond the reduction analysis, we consider the sample complexity of our approach using online-to-batch (Cesa-Bianchi 2004) techniques. We demonstrate DAGGER is scalable and outperforms previous approaches in practice on two challenging imitation learning problems: 1) learning to steer a car in a 3D racing game (*Super Tux Kart*) and 2) and learning to play *Super Mario Bros.*, given input image features and corresponding actions by a human expert and near-optimal planner respectively. Following (Daume 2009) in treating structured prediction as a degenerate imitation learning problem, we apply DAGGER to the OCR (Taskar 2003) benchmark prediction problem achieving results competitive with Maximum Margin Markov Networks (Taskar 2003, Ratliff 2007) and SEARN (Daume 2009) using only single-pass, greedy prediction.

2 PRELIMINARIES

We begin by introducing notation relevant to our imitation learning setting. We denote by Π the class of policies the learner is considering and T the task horizon. For any policy π, we let d_π^t denote the distribution of states at time t if the learner executed policy π from time step 1 to $t-1$. Furthermore, we denote $d_\pi = \frac{1}{T}\sum_{t=1}^{T} d_\pi^t$ the average distribution of states if we follow policy π for T steps. Given a state s, we denote $C(s,a)$ the expected immediate cost of performing action a in state s for the task we are considering and denote $C_\pi(s) = \mathbb{E}_{a\sim\pi(s)}[C(s,a)]$ the expected immediate cost of π in s. We assume C is bounded in $[0,1]$. The total cost of executing policy π for T-steps (*i.e.*, the cost-to-go) is denoted $J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{s\sim d_\pi^t}[C_\pi(s)] = T\mathbb{E}_{s\sim d_\pi}[C_\pi(s)]$.

In imitation learning, we may not necessarily know or observe true costs $C(s,a)$ for the particular task. Instead, we observe expert demonstration and seek to bound $J(\pi)$ for any cost function C based on how well π mimics the expert's policy π^*. Denote ℓ the observed surrogate loss function we minimize instead of C. For instance $\ell(s,\pi)$ may be the expected 0-1 loss of π with respect to π^* in state s, or a squared/hinge loss of π with respect to π^* in s. Importantly, in many instances, C and ℓ may be the same function– for instance, if we are interested in optimizing the learner's ability to predict the actions chosen by an expert.

Our goal is to find a policy $\hat{\pi}$ which minimizes the observed surrogate loss under its induced distribution of states, i.e.:

$$\hat{\pi} = \arg\min_{\pi\in\Pi} \mathbb{E}_{s\sim d_\pi}[\ell(s,\pi)] \quad (1)$$

As system dynamics are assumed both unknown and complex, we can not compute d_π and can only sample it by executing π in the system. Hence this is a non-i.i.d. supervised learning problem due to the dependence of the input distribution on the policy π itself. The interaction between policy and the resulting distribution make optimization difficult as it results in a non-convex objective even if the loss $\ell(s,\cdot)$ is convex in π for all states s.

We briefly review previous approaches and their performance guarantees.

2.1 Supervised Approach to Imitation

The traditional approach to imitation learning ignores the change in distribution and simply trains a policy π that performs well under the distribution of states encountered by the expert d_{π^*}. This can be achieved using any standard supervised learning algorithm. It finds the policy $\hat{\pi}_{sup}$:

$$\hat{\pi}_{sup} = \arg\min_{\pi\in\Pi} \mathbb{E}_{s\sim d_{\pi^*}}[\ell(s,\pi)] \quad (2)$$

Assuming $\ell(s,\pi)$ is the 0-1 loss (or upper bound on the 0-1 loss) implies the following performance guarantee with respect to any task cost function C:

Theorem 2.1. *(Ross 2010a) Let $\mathbb{E}_{s\sim d_{\pi^*}}[\ell(s,\pi)] = \epsilon$, then $J(\pi) \leq J(\pi^*) + T^2\epsilon$.*

Proof. Follows immediately from the result in Ross (2010a) from the fact ϵ is an upper bound on the expected 0-1 loss of π on d_{π^*}. □

Note that this bound is tight, i.e. there exist problems such that a policy π with ϵ 0-1 loss on d_{π^*} can incur extra cost that grows quadratically in T. Kaariainen (2006) demonstrated this in a sequence prediction setting[1] and

[1] In their example, training to predict the next output in the sequence with the previous correct output as input and an error rate of $\epsilon > 0$ can lead to an expected number of mistakes of

Ross (2010a) provided another example where this occurs in an imitation learning setting where $J(\hat{\pi}_{sup}) = (1-\epsilon T)J(\pi^*) + T^2\epsilon$. Hence the traditional supervised learning approach has poor performance guarantees due to the quadratic growth in T. Instead we would prefer approaches that can guarantee growth linear or near-linear in T and ϵ. The following two approaches from (Ross 2010a) achieve this on some classes of imitation learning problem, including all those where surrogate loss ℓ bounds C.

2.2 Forward Training

The forward training algorithm introduced in (Ross 2010a) trains a non-stationary policy (one policy π_t for each time step t) iteratively over T iterations, where at iteration t, π_t is train to mimic π^* on the distribution of states at time t induced by the previously trained policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$. The key idea is that by doing so, π_t is trained on the actual distribution of states it will encounter during the execution of the learned policy. Hence the forward algorithm guarantees that the expected surrogate loss under the distribution of states induced by the learned policy matches the average surrogate loss during training, and hence improved performance.

We here provide a theorem slightly more general than the one provided in (Ross 2010a) that applies to any policy π that can guarantee ϵ surrogate loss under its own distribution of states. This will be useful to bound the performance of our new approach presented in Section 3.

Let $Q_t^{\pi'}(s,\pi)$ denote the t-step cost of executing π in initial state s and then following policy π' and assume $\ell(s,\pi)$ is the 0-1 loss (or an upper bound on the 0-1 loss), then we have the following performance guarantee with respect to any task cost function C:

Theorem 2.2. *Let π be such that $\mathbb{E}_{s\sim d_\pi}[\ell(s,\pi)] = \epsilon$, and $Q^{\pi^*}_{T-t+1}(s,\pi) - Q^{\pi^*}_{T-t+1}(s,\pi^*) \leq u$ for all $t \in \{1,2,\ldots,T\}$, $d^t_\pi(s) > 0$, then $J(\pi) \leq J(\pi^*) + uT\epsilon$.*

Proof. We here follow a similar proof to Ross (2010a). Given our policy π, consider the policy $\pi_{1:t}$, which executes π in the first t-steps and then execute the expert π^*. Then

$$
\begin{aligned}
&J(\pi)\\
&= J(\pi^*) + \textstyle\sum_{t=0}^{T-1}[J(\pi_{1:T-t}) - J(\pi_{1:T-t-1})]\\
&= J(\pi^*) + \textstyle\sum_{t=1}^{T}\mathbb{E}_{s\sim d^t_\pi}[Q^{\pi^*}_{T-t+1}(s,\pi) - Q^{\pi^*}_{T-t+1}(s,\pi^*)]\\
&\leq J(\pi^*) + u\textstyle\sum_{t=1}^{T}\mathbb{E}_{s\sim d^t_\pi}[\ell(s,\pi)]\\
&= J(\pi^*) + uT\epsilon
\end{aligned}
$$

The inequality follows from the fact that since $\ell(s,\pi)$ is an upper bound on the 0-1 loss of π in s, then with probability less than $\ell(s,\pi)$, π is different than π^* in s, that could lead to a maximum increase in cost-to-go of u. When π and π^* are the same in s then $Q^{\pi^*}_{T-t+1}(s,\pi) - Q^{\pi^*}_{T-t+1}(s,\pi^*) = 0$. □

In the worst case, u could be $O(T)$ and the forward algorithm wouldn't provide any improvement over the traditional supervised learning approach. However, in many cases u is $O(1)$ or sub-linear in T and the forward algorithm leads to improved performance. For instance if C is the 0-1 loss with respect to the expert, then $u \leq 1$. Additionally if π^* is able to recover from mistakes made by π, in the sense that within a few steps, π^* is back in a distribution of states that is close to what π^* would be in if π^* had been executed initially instead of π, then u will be $O(1)$. [2]

A drawback of the forward algorithm is that it is impractical when T is large (or undefined) as we must train T different policies sequentially and can't stop the algorithm before we complete all T iterations. Hence it can't be applied to most real-world applications as T is usually very large or not well defined.

2.3 Stochastic Mixing Iterative Learning

SMILe, proposed by Ross (2010a), alleviates this problem and can be applied in practice when T is large or undefined by adopting an approach similar to SEARN (Daume 2009) where a stochastic stationary policy is trained over several iterations. Initially SMILe starts with a policy π_0 which always queries and executes the expert's action choice. At iteration n, a policy $\hat{\pi}_n$ is trained to mimic the expert under the distribution of trajectories π_{n-1} induces and then updates $\pi_n = \pi_{n-1} + \alpha(1-\alpha)^{n-1}(\hat{\pi}_n - \pi_0)$. This update is interpreted as adding probability $\alpha(1-\alpha)^{n-1}$ to executing policy $\hat{\pi}_n$ at any step and removing probability $\alpha(1-\alpha)^{n-1}$ of executing the queried expert's action. At iteration n, π_n is a mixture of n policies and the probability of using the queried expert's action is $(1-\alpha)^n$. We can stop the algorithm at any iteration N by returning the re-normalized policy $\tilde{\pi}_N = \frac{\pi_N - (1-\alpha)^N\pi_0}{1-(1-\alpha)^N}$ which doesn't query the expert anymore. Ross (2010a) showed that with α to be $O(\frac{1}{T^2})$ and N as $O(T^2\log T)$ can insure near-linear regret in T and ϵ for some class of problems. [3]

$\frac{T}{2} - \frac{1-(1-2\epsilon)^{T+1}}{4\epsilon} + \frac{1}{2}$ over sequences of length T at test time. We can see this is bounded by $T^2\epsilon$ and behaves as $\Theta(T^2\epsilon)$ for small ϵ.

[2] This is the case for instance in Markov Desision Processes (MDPs) when the markov chain defined by the system dynamics and policy π^* is rapidly mixing. In particular, if it is α-mixing with exponential decay rate δ then u is $O(\frac{1}{1-\exp(-\delta)})$.

[3] The guarantees are less strong than the forward algorithm as it requires a stronger requirement that each of the policies π_n can recover in a number of steps that does not grow faster than $O(\frac{1}{(1-\alpha)^n})$ as n increases. As π_n still executes the expert with probability $(1-\alpha)^n$ this can hold in many cases without making any assumptions about the learned policies (see (Ross 2010a)).

3 DATASET AGGREGATION

We now present DAGGER (Dataset Aggregation), an iterative algorithm that trains a deterministic policy that achieve good performance guarantees under its induced distribution of states.

In its simplest form, the algorithm proceeds as follows. At the first iteration, it uses the expert's policy to gather a dataset of trajectories $\mathcal{D}$ and train a policy $\hat{\pi}_2$ that best mimics the expert on those trajectories. Then at iteration n, it uses $\hat{\pi}_n$ to collect more trajectories and adds those trajectories to the dataset $\mathcal{D}$. The next policy $\hat{\pi}_{n+1}$ is the policy that best mimics the expert on the whole dataset $\mathcal{D}$. In other words, DAGGER proceeds by collecting a dataset at each iteration under the current policy and trains the next policy under the aggregate of all collected datasets. The intuition behind this algorithm is that over the iterations, we are building up the set of inputs that the learned policy is likely to encounter during its executing based on previous experience (training iterations). This algorithm can be interpreted as a *Follow-The-Leader* algorithm in that at iteration n picks the best policy $\hat{\pi}_{n+1}$ under all trajectories seen so far over the iterations.

To better leverage the presence of the expert in our imitation learning setting, we optionally also allow the algorithm to use a modified policy $\pi_i = \beta_i \pi^* + (1-\beta_i)\hat{\pi}_i$ at iteration i that queries the expert to choose controls a fraction of the time while collecting the next dataset. This is often desirable in practice as the first few policies, with relatively few data-points, may make many more mistakes and visit states irrelevant as the policy improves.

We will typically use $\beta_1 = 1$ so that we do not have to specify an initial policy $\hat{\pi}_1$ before getting data from the expert's behavior. Then we could choose $\beta_i = p^{i-1}$ to have a probability of using the expert that decays exponentially as in SMILe and SEARN. We show below the only requirement is that $\{\beta_i\}$ be a sequence such that $\overline{\beta}_N = \frac{1}{N}\sum_{i=1}^{N}\beta_i \to 0$ as $N \to \infty$. The simple case is handled with $\beta_i = I(i=1)$ for I the indicator function. The general DAGGER algorithm is detailed in Algorithm 3.1. The main result of our analysis in the next section is the following guarantee for DAGGER . Let $\pi_{1:N}$ denote the sequence of policies $\pi_1, \pi_2, \ldots, \pi_N$. Assume ℓ is strongly convex and bounded over Π. Suppose $\beta_i \leq (1-\alpha)^{i-1}$ for all i for some constant α independent of T. Let $\epsilon_N = \min_{\pi\in\Pi}\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{s\sim d_{\pi_i}}[\ell(s,\pi)]$ be the true loss of the best policy in hindsight. Then the following holds in the infinite sample case (infinite number of sample trajectories at each iteration):

Theorem 3.1. *For* DAGGER *, if N is $\tilde{O}(T)$ there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $\mathbb{E}_{s\sim d_{\hat{\pi}}}[\ell(s,\hat{\pi})] \leq \epsilon_N + O(1/T)$*

In particular, this holds for the policy $\hat{\pi} = \arg\min_{\pi\in\hat{\pi}_{1:N}} \mathbb{E}_{s\sim d_\pi}[\ell(s,\pi)]$. [4] If the task cost function C corresponds to the surrogate loss ℓ then this bound tells us directly that $J(\hat{\pi}) \leq T\epsilon_N + O(1)$. For arbitrary task cost function C, then if ℓ is an upper bound on the 0-1 loss with respect to π^*, combining this result with Theorem 2.2 yields that:

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1-\beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

Theorem 3.2. *For* DAGGER *, if N is $\tilde{O}(uT)$ there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $J(\hat{\pi}) \leq J(\pi^*) + uT\epsilon_N + O(1)$.*

Finite Sample Results In the finite sample case, suppose we sample m trajectories with π_i at each iteration i, and denote this dataset D_i. Let $\hat{\epsilon}_N = \min_{\pi\in\Pi}\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{s\sim D_i}[\ell(s,\pi)]$ be the training loss of the best policy on the sampled trajectories, then using the Hoeffding bound leads to the following guarantee:

Theorem 3.3. *For* DAGGER *, if N is $O(T^2\log(1/\delta))$ and m is $O(1)$ then with probability at least $1-\delta$ there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $\mathbb{E}_{s\sim d_{\hat{\pi}}}[\ell(s,\hat{\pi})] \leq \hat{\epsilon}_N + O(1/T)$*

A more refined analysis taking advantage of the strong convexity of the loss function (Kakade 2009) may lead to tighter generalization bounds that require N only of order $\tilde{O}(T\log(1/\delta))$.

Theorem 3.4. *For* DAGGER *, if N is $O(u^2T^2\log(1/\delta))$ and m is $O(1)$ then with probability at least $1-\delta$ there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $J(\hat{\pi}) \leq J(\pi^*) + uT\epsilon_N + O(1)$.*

4 THEORETICAL ANALYSIS

The theoretical analysis of DAGGER only relies on the no-regret property of the underlying *Follow-The-Leader* algorithm on strongly convex losses (Kakade 2009) which picks the sequence of policies $\hat{\pi}_{1:N}$. Hence the presented results also hold for *any* other no regret online learning algorithm we would apply to our imitation learning setting.

[4] Note, however, it is not necessary to find the best policy in the sequence that minimizes the loss under its distribution. For instance, the same guarantee holds for the policy which would uniformly randomly pick one policy in the sequence $\hat{\pi}_{1:N}$ and execute that policy for T steps.

In particular, we can consider the results here a reduction of imitation learning to no-regret learning where we treat mini-batches of trajectories under a single policy as a single online-learning example. We first briefly review concepts of online learning and no regret that will be used for this analysis.

4.1 Online Learning

In online learning, an algorithm must provide a policy π_n at iteration n which incurs a loss $\ell_n(\pi_n)$. After observing this loss, the algorithm can provide a different policy π_{n+1} for the next iteration which will incur loss $\ell_{n+1}(\pi_{n+1})$. The loss functions ℓ_{n+1} may vary in an unknown or even adversarial fashion over time. A no-regret algorithm is an algorithm that produces a sequence of policies $\pi_1, \pi_2, \ldots, \pi_N$ such that the average regret with respect to the best policy in hindsight goes to 0 as N goes to ∞:

$$\frac{1}{N}\sum_{i=1}^{N}\ell_i(\pi_i) - \min_{\pi\in\Pi}\frac{1}{N}\sum_{i=1}^{N}\ell_i(\pi) \leq \gamma_N \qquad (3)$$

for $\lim_{N\to\infty}\gamma_N = 0$. Many no-regret algorithms guarantee that γ_N is $\tilde{O}(\frac{1}{N})$ (e.g. when ℓ is strongly convex) (Hazan 2006, Kakade 2008, Kakade 2009).

4.2 No Regret Algorithms Guarantees

Now we show that no-regret algorithms can be used to find a policy which has good performance guarantees under its own distribution of states in our imitation learning setting. To do so, we choose the loss functions to be the loss under the distribution of states of the current policy chosen by the online algorithm: $\ell_i(\pi) = \mathbb{E}_{s\sim d_{\pi_i}}[\ell(s,\pi)]$.

Let $\epsilon_N = \min_{\pi\in\Pi}\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{s\sim d_{\pi_i}}[\ell(s,\pi)]$ denote the loss of the best policy in hindsight after N iterations and let $\ell_{\max}$ be an upper bound on the one step loss, i.e. $\ell_i(s,\hat{\pi}_i) \leq \ell_{\max}$ for all policies $\hat{\pi}_i$, and state s such that $d_{\hat{\pi}_i}(s) > 0$.

For our analysis of DAGGER , we need to bound the total variation distance between the distribution of states encountered by π_i and $\hat{\pi}_i$, which continues to call the expert. The following lemma is useful:

Lemma 4.1. $||d_{\pi_i} - d_{\hat{\pi}_i}||_1 \leq 2T\beta_i$.

Proof. Let d be the distribution of states over T steps conditioned on the fact that π_i executed π^* at least once over T steps. Since π_i always executes $\hat{\pi}_i$ over T steps with probability $(1-\beta_i)^T$ we have that $d_{\pi_i} = (1-\beta_i)^T d_{\hat{\pi}_i} + (1-(1-\beta_i)^T)d$. Thus

$$\begin{aligned}
&||d_{\pi_i} - d_{\hat{\pi}_i}||_1 \\
&= (1-(1-\beta_i)^T)||d - d_{\hat{\pi}_i}||_1 \\
&\leq 2(1-(1-\beta_i)^T) \\
&\leq 2T\beta_i
\end{aligned}$$

The last inequality follows from the fact that $(1-\beta)^T \geq 1-\beta T$ for any $\beta \in [0,1]$. □

Note that this bound is only better than the trivial bound $||d_{\pi_i} - d_{\hat{\pi}_i}||_1 \leq 2$ for $\beta_i \leq \frac{1}{T}$. Assume β_i is non-increasing and define n_β the largest $n \leq N$ such that $\beta_n > \frac{1}{T}$. Then we have the following guarantee:

Theorem 4.1. *For* DAGGER *, there exists a policy* $\hat{\pi} \in \hat{\pi}_{1:N}$ *s.t.* $\mathbb{E}_{s\sim d_{\hat{\pi}}}[\ell(s,\hat{\pi})] \leq \epsilon_N + \gamma_N + \frac{2\ell_{\max}}{N}[n_\beta + T\sum_{i=n_\beta+1}^{N}\beta_i]$*, for* γ_N *the average regret of* $\hat{\pi}_{1:N}$*.*

Proof. Using the last lemma we have that $\mathbb{E}_{s\sim d_{\hat{\pi}_i}}(\ell_i(s,\hat{\pi}_i)) \leq \mathbb{E}_{s\sim d_{\pi_i}}(\ell_i(s,\hat{\pi}_i)) + 2\ell_{\max}\min(1,T\beta_i)$. Then:

$$\begin{aligned}
&\min_{\hat{\pi}\in\hat{\pi}_{1:N}}\mathbb{E}_{s\sim d_{\hat{\pi}}}[\ell(s,\hat{\pi})] \\
&\leq \tfrac{1}{N}\textstyle\sum_{i=1}^{N}\mathbb{E}_{s\sim d_{\hat{\pi}_i}}(\ell(s,\hat{\pi}_i)) \\
&\leq \tfrac{1}{N}\textstyle\sum_{i=1}^{N}[\mathbb{E}_{s\sim d_{\pi_i}}(\ell(s,\hat{\pi}_i)) + 2\ell_{\max}\min(1,T\beta_i)] \\
&\leq \gamma_N + \tfrac{2\ell_{\max}}{N}[n_\beta + T\textstyle\sum_{i=n_\beta+1}^{N}\beta_i] + \min_{\pi\in\Pi}\textstyle\sum_{i=1}^{N}\ell_i(\pi) \\
&= \gamma_N + \epsilon_N + \tfrac{2\ell_{\max}}{N}[n_\beta + T\textstyle\sum_{i=n_\beta+1}^{N}\beta_i]
\end{aligned}$$

□

Under an error reduction assumption that for any input distribution, there is some policy $\pi \in \Pi$ that achieves surrogate loss of ϵ, this implies we are guaranteed to find a policy $\hat{\pi}$ which achieves ϵ surrogate loss under its own state distribution in the limit, provided $\overline{\beta}_N \to 0$. For instance, if we choose β_i to be of the form $(1-\alpha)^{i-1}$, then $\frac{1}{N}[n_\beta + T\sum_{i=n_\beta+1}^{N}\beta_i] \leq \frac{1}{N\alpha}[\log T + 1]$ and this extra penalty becomes negligible for N as $\tilde{O}(T)$. As we need at least $\tilde{O}(T)$ iterations to make γ_N negligible, the number of iterations required by DAGGER is similar to that required by any no-regret algorithm. Note that this is not as strong as the general error or regret reductions consider in (Beygelzimer 2007, Ross 2010a, Daume 2009) which require only classification: here we require a no-regret method or strongly convex surrogate loss function, a stronger (albeit common) assumption.

Finite Sample Case: The previous results hold if the online learning algorithm observes the infinite sample loss, i.e. the loss on the true distribution of trajectories induced by the current policy π_i. In practice however the online learning would only observe its loss on a small sample of trajectories at each iteration. We wish to bound the true loss under its own distribution of the best policy in the sequence as a function of the regret on the finite sample of training trajectories.

At each iteration i, we assume the algorithm proceeds by sampling m trajectories using π_i and then observes the loss $\ell_i(\pi) = \mathbb{E}_{s\sim D_i}(\ell(s,\pi))$, where D_i is the dataset of those m trajectories. In this case, the online learning algorithm guarantees $\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{s\sim D_i}(\ell(s,\pi_i)) -$

$\min_{\pi\in\Pi} \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{s\sim D_i}(\ell(s,\pi)) \leq \gamma_N$. Let $\hat{\epsilon}_N = \min_{\pi\in\Pi} \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{s\sim D_i}[\ell(s,\pi)]$ the training loss of the best policy in hindsight, then we have the following:

Theorem 4.2. *For* DAGGER *, with probability at least* $1-\delta$*, there exists a policy* $\hat{\pi} \in \hat{\pi}_{1:N}$ *s.t.* $\mathbb{E}_{s\sim d_{\hat{\pi}}}[\ell(s,\hat{\pi})] \leq \hat{\epsilon}_N + \gamma_N + \frac{2\ell_{\max}}{N}[n_\beta + T\sum_{i=n_\beta+1}^{N}\beta_i] + \ell_{\max}\sqrt{\frac{2\log(1/\delta)}{mN}}$*, for* γ_N *the average regret of* $\hat{\pi}_{1:N}$*.*

Proof. Let Y_{ij} denote the difference between the expected per step loss of $\hat{\pi}_i$ under the distribution of states d_{π_i} and the average per step loss of $\hat{\pi}_i$ under the j^{th} sample trajectory with π_i at iteration i. The random variables Y_{ij} over all $i \in \{1,2,\ldots,N\}$ and $j \in \{1,2,\ldots,m\}$ are independent, zero mean and bounded in $[-\ell_{\max}, \ell_{\max}]$. By Hoeffding's inequality $\frac{1}{mN}\sum_{i=1}^{N}\sum_{j=1}^{m} Y_{ij} \leq \ell_{\max}\sqrt{\frac{2\log(1/\delta)}{mN}}$ with probability at least $1-\delta$. By following a similar proof to the previous theorem, we obtain that with probability at least $1-\delta$:

$$
\begin{aligned}
&\min_{\hat{\pi}\in\hat{\pi}_{1:N}} \mathbb{E}_{s\sim d_{\hat{\pi}}}[\ell(s,\hat{\pi})] \\
&\leq \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \mathbb{E}_{s\sim d_{\hat{\pi}_i}}[\ell(s,\hat{\pi}_i)] \\
&\leq \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \mathbb{E}_{s\sim d_{\pi_i}}[\ell(s,\hat{\pi}_i)] + \tfrac{2\ell_{\max}}{N}[n_\beta + T\textstyle\sum_{i=n_\beta+1}^{N}\beta_i] \\
&= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \mathbb{E}_{s\sim D_i}[\ell(s,\hat{\pi}_i)] + \tfrac{1}{mN}\textstyle\sum_{i=1}^{N}\sum_{j=1}^{m} Y_{ij} \\
&\quad + \tfrac{2\ell_{\max}}{N}[n_\beta + T\textstyle\sum_{i=n_\beta+1}^{N}\beta_i] \\
&\leq \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \mathbb{E}_{s\sim D_i}[\ell(s,\hat{\pi}_i)] + \ell_{\max}\sqrt{\tfrac{2\log(1/\delta)}{mN}} \\
&\quad + \tfrac{2\ell_{\max}}{N}[n_\beta + T\textstyle\sum_{i=n_\beta+1}^{N}\beta_i] \\
&\leq \hat{\epsilon}_N + \gamma_N + \ell_{\max}\sqrt{\tfrac{2\log(1/\delta)}{mN}} + \tfrac{2\ell_{\max}}{N}[n_\beta + T\textstyle\sum_{i=n_\beta+1}^{N}\beta_i]
\end{aligned}
$$

□

Due to the use of the Hoeffding bound, this bound suggests we need a total number of trajectories Nm to be $O(T^2\log(1/\delta))$ to make the generalization error term be $O(1/T)$ and negligible over T-step trajectories. More refined tools (Sridharan 2008, Kakade 2009) that leverage the strong convexity of loss ℓ lead to a faster convergence rate and require only $O(T\log T\log(1/\delta))$ trajectories.

5 EXPERIMENTS

To demonstrate the efficacy and scalability of DAGGER , we apply it to two challenging imitation learning problems and a sequence labeling task (handwriting recognition).

5.1 Super Tux Kart

Super Tux Kart is an open source 3D racing game similar to the popular Mario Kart. Our goal is to train the computer to steer the cart moving at fixed speed on a particular race track, based on the current game image features as input (see Figure 1). A human expert is used to provide demonstrations of the correct steering (analog joystick value in [-1,1]) for each of the observed game images. For all methods, we use a linear controller as the base learner which updates the steering at 5Hz; i.e. given the vector of image features[5] x, the output steering $\hat{y} = w^T x + b$. We optimize for w and b by minimizing the ridge regression objective: $L(w,b) = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i + b - y_i)^2 + \frac{\lambda}{2}w^T w$, for regularization parameter $\lambda = 10^{-3}$.

Figure 1: Image from Super Tux Kart's Star Track.

We compare performance on a race track called Star Track. As this track floats in space, the cart can fall off the track at any point (the cart is repositioned at the center of the track when this occurs). We measure performance in terms of the average number of falls per lap. For SMILe and DAGGER , we used 1 lap of training per iteration (1000 data points) and run both methods for 20 iterations. For SMILe we choose parameter $\alpha = 0.1$ as in Ross (2010a), and for DAGGER the parameter $\beta_i = I(i=1)$ for I the indicator function. Figure 2 shows 95% confidence intervals on the average falls per lap of each method after 1, 5, 10, 15 and 20 iterations as a function of the total number of training data collected. We first observe that with the baseline supervised approach where training always occurs under the expert's trajectories that performance does not improve as more data is collected. This is because most of the training laps are all very similar and do not help the learner to learn how to recover from mistakes it makes. With SMILe we obtain some improvements but the policy after 20 iterations still falls off the track about twice per lap on average. This is in part due to the stochasticity of the policy which sometimes make bad choices of actions. For DAGGER , we were able to obtain a policy that never falls off the track after 15 iterations of training. Though even after 5 iterations, the policy we obtain almost never falls off the track and is significantly outperforming both SMILe and the baseline supervised approach. Furthermore, the policy obtain by DAGGER is smoother and looks qualitatively better than the policy obtained with SMILe. A video available on YouTube (Ross 2010b) shows a qualitative comparison of the behavior obtained with each method.

[5]The image features are simply the LAB color values of each pixel in a 25x19 resized image of the 800x600 original image, yielding a total of 1425 features.

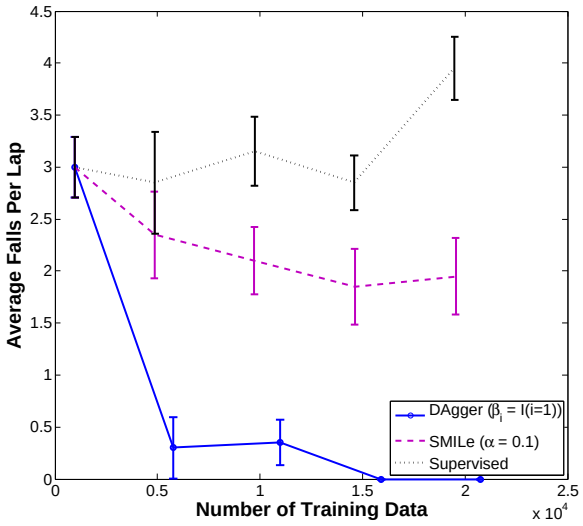

Figure 2: Average falls/lap as a function of training data.

5.2 Super Mario Bros.

Super Mario Bros. is a popular platform video game where the character, Mario, must move across each stage by avoiding being hit by enemies and falling into gaps, and before running out of time. We used the open source simulator from a recent Mario Bros. AI competition (Togelius 2009) which can randomly generate stages of varying difficulty (more difficult gaps and types of enemies). Our goal is to train the computer to play this game based on the current game image features as input (see Figure 3). Our expert in this scenario is a near-optimal planning algorithm that has full access to the game's internal state and can simulate exactly the consequence of future actions. An action consists of 4 binary variables indicating which subset of buttons we should press in {left,right,jump,speed}. For all methods, we use 4 independent linear SVM as the

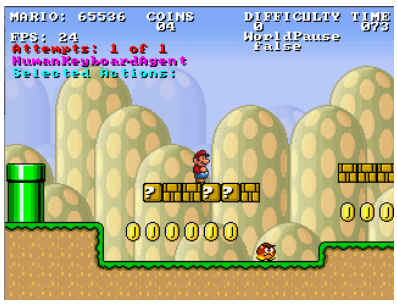

Figure 3: Captured image from Mario Bros.

base learner which update the 4 binary actions at 5Hz; i.e. given the vector of image features[6] x, the k^{th} output binary variable $\hat{y}_k = I(w_k^T x + b_k > 0)$. We optimize for w_k, b_k by minimizing the SVM objective using stochastic gradient descent (Ratliff 2007).

We compare performance in terms of the average distance travelled by Mario per stage before dieing, running out of time or completing the stage, on randomly generated stages of difficulty 1 with a time limit of 60 seconds to complete the stage. The total distance of each stage varies but is around 4200-4300 on average, so performance can vary roughly in [0,4300]. Stages of difficulty 1 are fairly easy for an average human player but contains most types of enemies and gaps, except with fewer enemies and gaps than stages of harder difficulties. For SMILe and DAGGER , we collected 5000 data points per iteration (each stage is about 150 data point if run to completion) and run both methods for 20 iterations. For SMILe we choose parameter $\alpha = 0.1$ as in Ross (2010a), and for DAGGER we obtained results with 3 different choice of the parameter β_i: 1) $\beta_i = I(i = 1)$ for I the indicator function; 2) $\beta_i = 0.8^{i-1}$ and 3) $\beta_i = 0.9^{i-1}$. Figure 4 shows 95% confidence intervals on the average distance travelled per stage at each iteration as a function of the total number of training data collected. Again here we observe that with

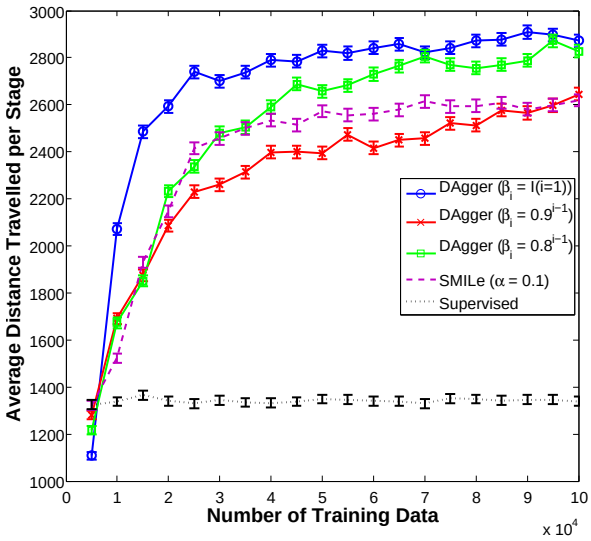

Figure 4: Average distance travelled per stage as a function of training data.

the supervised approach, performance stagnates as we collect more data from the expert demonstrations, as this does not help the particular errors the learned controller makes. In particular, a reason the supervised approach gets such a low score is that under the learned controller, Mario is

[6]For the input features, each image is discretized in a 22x22 cell grid centered around Mario. 14 binary features are used to describe the objects present in each cell (types of ground, enemies, blocks and other special items). We use a history of those features over the last 4 images as input, in addition to other features describing the history of the last 6 actions and the state of Mario (small,big,fire,touches ground). This gave us a total of 27152 binary features as input which are usually very sparse (around 300 features are 1 on average).

often stuck at some location against an obstacle instead of jumping over it. Since the expert always jumps over obstacles at a significant distance away , the controller didn't learn how to get unstuck in situations where its right next to an obstacle. On the other hand, all the other iterative methods perform much better as they eventually learn to get unstuck in those situations by encountering them at the later iterations. Again in this experiment, DAGGER outperforms SMILe. When using β_i that is non-zero for $i > 1$, we observe that convergence is slightly slower, and both results with 0.8^{i-1} and 0.9^{i-1} could have benefited from more iterations as performance was still improving at the end of the 20 iterations. A video available on YouTube (Ross 2010c) also shows a qualitative comparison of the behavior obtained with each method.

5.3 Handwriting Recognition

Finally, we demonstrate the efficacy of our approach on a structured prediction problem involving recognizing handwritten words given the sequence of images of each character in the word. We follow (Daume 2009) in adopting a view of structured prediction as a degenerate form of imitation learning where the system dynamics are deterministic and trivial in simply passing on earlier predictions made as inputs for future predictions. We use the data-set of Taskar (2003) which has been used extensively in the literature to compare several structured prediction approaches. This data-set contains roughly 6600 words (for a total of over 52000 characters) partitioned in 10 folds. The image of each characters is 8x16 binary pixels (128 input features). We consider the large data-set experiment which consists in training on 9 folds and testing on 1 fold and repeating this over all folds. Performance is measured in terms of the character accuracy on the test folds.

We consider predicting the word by predicting each character in sequence in a left to right order, using the previously predicted character to help predict the next, following the greedy SEARN approach in (Daume 2009). Hence in addition to the 128 pixel features, we also use 26 indicator binary features to encode the previously predicted letter in the word. We consider training an SVM as the base classifier, using the all-pairs reduction to binary classification (Beygelzimer 2005).

Here we compare our method to SMILe as well as SEARN, using the same approximations used in (Daume 2009). We also compare these approaches to 2 baseline, a non-structured approach which simply predicts each character independently and the supervised training approach where training is conducted with the previous character always correctly labeled. We try 2 different parameter α for SEARN, $\alpha = 0.1$ and $\alpha = 1$ (pure policy iteration), and run all approaches for 20 iterations. Figure 5 shows the performance of each approach on the test folds after each iteration as a function of training data. The baseline result

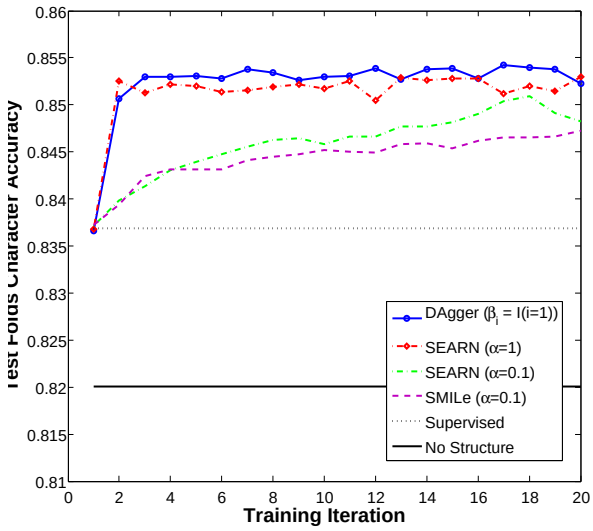

Figure 5: Character accuracy on test folds as a function of training iteration.

without structure achieves 82% character accuracy by just using an SVM that predicts each character independently. When adding the previous character feature, but training with always the previous character correctly labeled (supervised approach), performance increases up to 83.6%. Using DAgger increases further performance to 85.5%. Surprisingly, we observe SEARN with $\alpha = 1$, which is a pure policy iteration approach performs very well on this experiment and better than using a small $\alpha = 0.1$. We believe this is due to the fact that only a small part of the input is influenced by the current policy (the previous predicted character feature) so this makes this approach no as unstable as in general reinforcement/imitation learning problems. SEARN and SMILe with small $\alpha = 0.1$ performs similarly but significantly worse than DAgger. Note that we chose the simplest (greedy, one-pass) decoding to illustrate the benefits of the DAGGER approach with respect to existing reductions. Similar techniques can be applied to multi-pass or beam-search decoding leading to results that are competitive with the state-of-the-art.

6 FUTURE WORK

We show that by batching over iterations of interaction with a system, no-regret methods, including the presented DAGGER approach can provide a learning reduction with strong performance guarantees in both imitation learning and structured prediction. In future work, we will consider more sophisticated strategies then simple greedy forward decoding for structured prediction, as well as using base classifiers that rely on Inverse Optimal Control (Abbeel 2004, Ratliff 2006) techniques to learn a cost function for

a planner to aid prediction in imitation learning. Further we believe techniques similar to those presented, by leveraging a cost-to-go estimate, may provide an understanding of the success of online methods for reinforcement learning and suggest a similar data-aggregation method that can guarantee performance in such settings.

References

P. Abbeel and A. Y. Ng (2004). Apprenticeship learning via inverse reinforcement learning. In *ICML*.

B. D. Argall, S. Chernova, M. Veloso and B. Browning (2009). A Survey of Robot Learning from Demonstration. In *Robotics and Autonomous Systems*.

A. Beygelzimer, V. Dani, T. Hayes, J. Langford, and B. Zadrozny (2005). Reductions Between Classification Tasks. In *International Conference on Machine Learning*.

L. Bottou (2009). sgd code at `http://www.leon.bottou.org/projects/sgd`.

N. Cesa-Bianchi, A. Conconi and C. Gentile (2004). On the generalization ability of on-line learning algorithms. In *IEEE Transactions on Information Theory*.

S. Chernova and M. Veloso (2009). Interactive Policy Learning through Confidence-Based Autonomy. In *JAIR*.

H. Daume, J. Langford and D. Marcu (2009). Search-based structured prediction. In *Machine Learning Journal*.

E. Hazan, A. Kalai, S. Kale and A. Agarwal (2006). Logarithmic regret algorithms for online convex optimization. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*.

M. Kääriäinen (2006). Lower bounds for reductions. *Atomic Learning workshop*.

S. Kakade and J. Langford (2002). Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*.

S. Kakade and S. Shalev-Shwartz (2008). Mind the duality gap: Logarithmic regret algorithms for online optimization. In *Advances in Neural Information Processing Systems*.

S. Kakade and A. Tewari (2009). On the generalization ability of online strongly convex programming algorithms. In *Advances in Neural Information Processing Systems*.

S. Ross and J. A. Bagnell (2010). Efficient reductions for imitation learning. In *AISTATS*.

S. Ross, G. Gordon and J. A. Bagnell (2010). Comparison of Imitation Learning Approaches on Super Tux Kart. `http://www.youtube.com/watch?v=V00npNnWzSU`.

S. Ross, G. Gordon and J. A. Bagnell (2010). Comparison of Imitation Learning Approaches on Mario Bros. `http://www.youtube.com/watch?v=anOI0xZ3kGM`.

N. Ratliff, D. Bradley, J. A. Bagnell and J. Chestnutt (2006). Boosting structured prediction for imitation learning. In *NIPS*.

N. Ratliff, J. A. Bagnell, M. Zinkevich. (2007) (Online) Subgradient Methods for Structured Prediction, Artificial Intelligence and Statistics.

D. Roth, K. Small and I. Titov (2009). Sequential Learning of Classifiers for Structured Prediction Problems. Proc. of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS).

S. Schaal (1999). Is imitation learning the route to humanoid robots? In *Trends in Cognitive Sciences*.

D. Silver, J. A. Bagnell and A. Stentz (2008). High Performance Outdoor Navigation from Overhead Data using Imitation Learning. In *Proceedings of Robotics Science and Systems* (RSS).

K. Sridharan, N. Srebo and S. Shalev-Shwartz (2008). Fast rates for regularized objectives. In *Advances in Neural Information Processing Systems*.

B. Taskar, C. Guestrin, D. Koller (2003). Max-margin markov networks. In *Advances in Neural Information Processing Systems*.

J. Togelius and S. Karakovskiy (2009). Mario AI Competition. `http://julian.togelius.com/mariocompetition2009`.