A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

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 theory and often in practice. Some recent approaches (Daumé III et al., 2009; Ross and Bagnell, 2010) provide stronger guarantees in this setting, but remain somewhat unsatisfactory as they train either non-stationary or 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 demonstrate that this new approach outperforms previous approaches on two challenging imitation learning problems and a benchmark sequence labeling problem.

1 INTRODUCTION

Sequence Prediction problems arise commonly in practice. For instance, most robotic systems must be able to predict/make a sequence of actions given a sequence of observations revealed 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 and Ng, 2004; Ratliff et al., 2006; Silver et al., 2008; Argall et al., 2009; Chernova and Veloso, 2009; Ross and Bagnell, 2010). 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 most statistical learning approaches.

Ignoring this issue leads to poor performance both in theory and practice (Ross and Bagnell, 2010). 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 and Bagnell, 2010). 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 of errors.

Recent approaches (Ross and Bagnell, 2010) can guarantee an expected number of mistakes linear (or nearly so) in the task horizon T and error ϵ by training over several iterations and allowing the learner to influence the input states where expert demonstration is provided (through execution of its own controls in the system). One approach (Ross and Bagnell, 2010) learns a non-stationary policy by training a different policy for each time step in sequence, starting from the first step. Unfortunately this is impractical when T is large or ill-defined. Another approach called SMILe (Ross and Bagnell, 2010), similar to SEARN (Daumé III et al., 2009) and CPI (Kakade and Langford, 2002), trains a stationary stochastic policy (a finite mixture of policies) by adding a new policy to the mixture at each iteration of training. However this may be unsatisfactory for practical applications as some policies in the mixture are worse than

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15. Copyright 2011 by the authors.

others and the learned controller may be unstable.

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 et al., 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 et al., 2004; Hazan et al., 2006; Kakade and Shalev-Shwartz, 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 et al., 2005). Beyond the reduction analysis, we consider the sample complexity of our approach using online-to-batch (Cesa-Bianchi et al., 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 Daumé III et al. (2009) in treating structured prediction as a degenerate imitation learning problem, we apply DAGGER to the OCR (Taskar et al., 2003) benchmark prediction problem achieving results competitive with the state-of-the-art (Taskar et al., 2003; Ratliff et al., 2007; Daumé III et al., 2009) using only single-pass, greedy prediction.

2 PRELIMINARIES

We begin by introducing notation relevant to our 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 demonstrations 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 cannot 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 makes 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 now briefly review previous approaches and their 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 bounded in $[0,1]$:

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

Proof. Follows from result in Ross and Bagnell (2010) since ϵ is an upper bound on the 0-1 loss of π in 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. Kääriäinen (2006) demonstrated this in a sequence prediction setting[1] and

[1] In their example, an error rate of $\epsilon > 0$ when trained to predict the next output in sequence with the previous correct output as input can lead to an expected number of mistakes of $\frac{T}{2} - \frac{1-(1-2\epsilon)^{T+1}}{4\epsilon} + \frac{1}{2}$ over sequences of length T at test time. This is bounded by $T^2\epsilon$ and behaves as $\Theta(T^2\epsilon)$ for small ϵ.

Ross and Bagnell (2010) provided an imitation learning example 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 and Bagnell (2010) achieve this on some classes of imitation learning problems, including all those where surrogate loss ℓ upper bounds C.

2.2 Forward Training

The forward training algorithm introduced by Ross and Bagnell (2010) 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 trained to mimic π^* on the distribution of states at time t induced by the previously trained policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$. By doing so, π_t is trained on the actual distribution of states it will encounter during execution of the learned policy. Hence the forward algorithm guarantees that the expected loss under the distribution of states induced by the learned policy matches the average loss during training, and hence improves performance.

We here provide a theorem slightly more general than the one provided by Ross and Bagnell (2010) 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 bounded in $[0, 1]$:

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

Proof. We here follow a similar proof to Ross and Bagnell (2010). 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

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

The inequality follows from the fact that $\ell(s, \pi)$ upper bounds the 0-1 loss, and hence the probability π and π^* pick different actions in s; when they pick different actions, the increase in cost-to-go $\leq u$. □

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 cannot stop the algorithm before we complete all T iterations. Hence it can not be applied to most real-world applications.

2.3 Stochastic Mixing Iterative Learning

SMILe, proposed by Ross and Bagnell (2010), alleviates this problem and can be applied in practice when T is large or undefined by adopting an approach similar to SEARN (Daumé III et al., 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 and Bagnell (2010) showed that choosing α in $O(\frac{1}{T^2})$ and N in $O(T^2 \log T)$ guarantees near-linear regret in T and ϵ for some class of problems.

3 DATASET AGGREGATION

We now present DAGGER (Dataset Aggregation), an iterative algorithm that trains a deterministic policy that achieves 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 $\mathcal{D}$.

[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 $O(\frac{1}{1-\exp(-\delta)})$.

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.

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 execution based on previous experience (training iterations). This algorithm can be interpreted as a *Follow-The-Leader* algorithm in that at iteration n we pick the best policy $\hat{\pi}_{n+1}$ in hindsight, i.e. under all trajectories seen so far over the iterations.

To better leverage the presence of the expert in our imitation learning setting, we optionally 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 datapoints, may make many more mistakes and visit states that are 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, parameter-free version of the algorithm described above is the special case $\beta_i = I(i = 1)$ for I the indicator function, which often performs best in practice (see Section 5). 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 $\hat{\pi} = \arg\min_{\pi \in \hat{\pi}_{1:N}} \mathbb{E}_{s \sim d_\pi}[\ell(s, \pi)]$. [3] If the task cost function C corresponds to (or is upper bounded by) 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:

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 Azuma-Hoeffding's inequality 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 and Tewari, 2009) may lead to tighter generalization bounds that require N only of order $\tilde{O}(T \log(1/\delta))$. Similarly:

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. $J(\hat{\pi}) \leq J(\pi^*) + uT\hat{\epsilon}_N + O(1)$.*

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 and Tewari, 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. In particular, we can consider the results here a reduction of imitation learning to no-regret online 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

[3] It is not necessary to find the best policy in the sequence that minimizes the loss under its distribution; the same guarantee holds for the policy which uniformly randomly picks one policy in the sequence $\hat{\pi}_{1:N}$ and executes that policy for T steps.

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 ∞:

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

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 et al., 2006; Kakade and Shalev-Shwartz, 2008; Kakade and Tewari, 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 must choose the loss functions to be the loss under the distribution of states of the current policy chosen by the online algorithm: $\ell_i(\pi) = \mathbb{E}_{s\sim d_{\pi_i}}[\ell(s,\pi)]$.

For our analysis of DAGGER, we need to bound the total variation distance between the distribution of states encountered by $\hat{\pi}_i$ and π_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 the distribution of states over T steps conditioned on π_i picking 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 $d_{\pi_i} = (1-\beta_i)^T d_{\hat{\pi}_i} + (1-(1-\beta_i)^T)d$. 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 $\beta \in [0,1]$. □

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

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

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

□

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 considered in (Beygelzimer et al., 2005; Ross and Bagnell, 2010; Daumé III et al., 2009) which require only classification: 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 algorithm 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 trajectories.

At each iteration i, we assume the algorithm samples m trajectories using π_i and then observes the loss $\ell_i(\pi) = \mathbb{E}_{s\sim D_i}(\ell(s,\pi))$, for D_i the dataset of those m trajectories. The online learner 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. Following a similar analysis to Cesa-Bianchi et al. (2004), we obtain:

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} be the difference between the expected per step loss of $\hat{\pi}_i$ under state distribution 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 all zero mean, bounded in $[-\ell_{\max}, \ell_{\max}]$ and form a martingale (considering the order $Y_{11}, Y_{12}, \ldots, Y_{1m}, Y_{21}, \ldots, Y_{Nm}$). By Azuma-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 $1-\delta$. Hence, we obtain that with probability at least $1-\delta$:

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

□

The use of Azuma-Hoeffding's inequality suggests we need Nm in $O(T^2\log(1/\delta))$ for the generalization error to be $O(1/T)$ and negligible over T steps. Leveraging the strong convexity of ℓ as in (Kakade and Tewari, 2009) may lead to a tighter bound requiring only $O(T\log(T/\delta))$ 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 a 3D racing game similar to the popular Mario Kart. Our goal is to train the computer to steer the kart 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

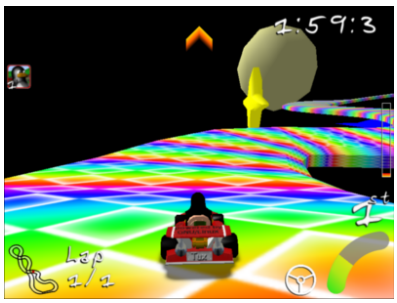

Figure 1: Image from Super Tux Kart's Star Track.

controller as the base learner which updates the steering at 5Hz based on the vector of image features[4].

[4]Features x: LAB color values of each pixel in a 25x19 resized image of the 800x600 image; output steering: $\hat{y} = w^T x + b$ where w, b minimizes 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 regularizer $\lambda = 10^{-3}$.

We compare performance on a race track called Star Track. As this track floats in space, the kart can fall off the track at any point (the kart 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 ($\sim$1000 data points) and run both methods for 20 iterations. For SMILe we choose parameter $\alpha = 0.1$ as in Ross and Bagnell (2010), 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

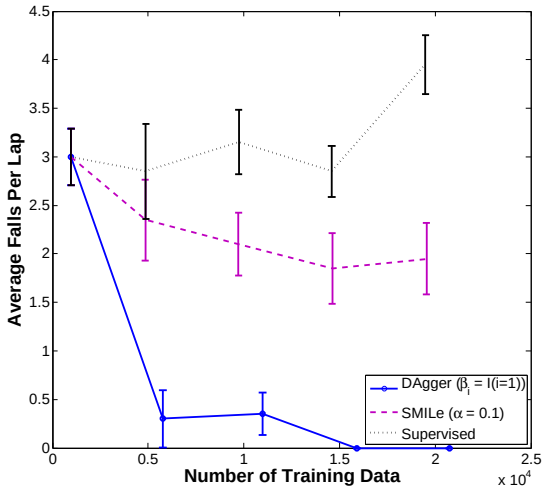

Figure 2: Average falls/lap as a function of training data.

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 makes 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 obtained by DAGGER is smoother and looks qualitatively better than the policy obtained with SMILe. A video available on YouTube (Ross, 2010a) shows a qualitative comparison of the behavior obtained with each method.

5.2 Super Mario Bros.

Super Mario Bros. is a platform video game where the character, Mario, must move across each stage by avoid-

ing being hit by enemies and falling into gaps, and before running out of time. We used the simulator from a recent Mario Bros. AI competition (Togelius and Karakovskiy, 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

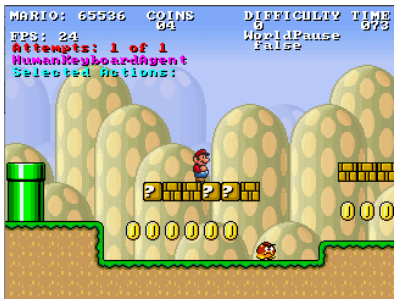

Figure 3: Captured image from Super Mario Bros.

all methods, we use 4 independent linear SVM as the base learner which update the 4 binary actions at 5Hz based on the vector of image features[5].

We compare performance in terms of the average distance travelled by Mario per stage before dying, 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 contain most types of enemies and gaps, except with fewer enemies and gaps than stages of harder difficulties. We compare performance of DAgger, SMILe and SEARN[6] to the supervised approach (Sup). With each approach we collect 5000 data points per iteration (each stage is about 150 data points if run to completion) and run the methods for 20 iterations. For SMILe we choose parameter $\alpha = 0.1$ (Sm0.1) as in Ross and Bagnell (2010). For DAGGER we obtain results with different choice of the parameter β_i: 1) $\beta_i = I(i = 1)$ for I the indicator function (D0); 2) $\beta_i = p^{i-1}$ for all values of $p \in \{0.1, 0.2, \ldots, 0.9\}$. We report the best results obtained with $p = 0.5$ (D0.5). We also report the results with $p = 0.9$ (D0.9) which shows the slower convergence of using the expert more frequently at later iterations. Similarly for SEARN, we obtain results with all choice of α in $\{0.1, 0.2, \ldots, 1\}$. We report the best results obtained with $\alpha = 0.4$ (Se0.4). We also report results with $\alpha = 1.0$ (Se1), which shows the unstability of such a pure policy iteration approach. 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 the supervised

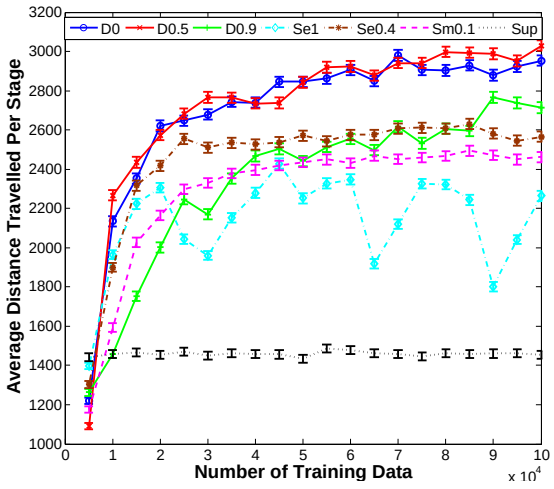

Figure 4: Average distance/stage as a function of data.

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 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 did not learn how to get unstuck in situations where it is 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, and also outperforms SEARN for all choice of α we considered. When using $\beta_i = 0.9^{i-1}$, convergence is significantly slower could have benefited from more iterations as performance was still improving at the end of the 20 iterations. Choosing 0.5^{i-1} yields slightly better performance (3030) then with the indicator function (2980). This is potentially due to the large number of data generated where mario is stuck at the same location in the early iterations when using the indicator; whereas using the ex-

[5]For the input features x: each image is discretized in a grid of 22x22 cells centered around Mario; 14 binary features describe each cell (types of ground, enemies, blocks and other special items); a history of those features over the last 4 images is used, in addition to other features describing the last 6 actions and the state of Mario (small,big,fire,touches ground), for a total of 27152 binary features (very sparse). The k^{th} output binary variable $\hat{y}_k = I(w_k^T x + b_k > 0)$, where w_k, b_k optimizes the SVM objective with regularizer $\lambda = 10^{-4}$ using stochastic gradient descent (Ratliff et al., 2007; Bottou, 2009).

[6]We use the same cost-to-go approximation in Daumé III et al. (2009); in this case SMILe and SEARN differs only in how the weights in the mixture are updated at each iteration.

pert a small fraction of the time still allows to observe those locations but also unstucks mario and makes it collect a wider variety of useful data. A video available on YouTube (Ross, 2010b) 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 Daumé III et al. (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 dataset of Taskar et al. (2003) which has been used extensively in the literature to compare several structured prediction approaches. This dataset contains roughly 6600 words (for a total of over 52000 characters) partitioned in 10 folds. We consider the large dataset experiment which consists of 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 and a linear SVM[7], following the greedy SEARN approach in Daumé III et al. (2009). Here we compare our method to SMILe, as well as SEARN (using the same approximations used in Daumé III et al. (2009)). We also compare these approaches to two 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. Again we try all choice of $\alpha \in \{0.1, 0.2, \ldots, 1\}$ for SEARN, and report results for $\alpha = 0.1$, $\alpha = 1$ (pure policy iteration) and the best $\alpha = 0.8$, 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 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 performance further to 85.5%. Surprisingly, we observe SEARN with $\alpha = 1$, which is a pure policy iteration approach performs very well on this experiment, similarly to the best $\alpha = 0.8$ and DAgger. Because there is only a small part of the input that is influenced by the current policy (the previous

[7]Each character is 8x16 binary pixels (128 input features); 26 binary features are used to encode the previously predicted letter in the word. We train the multiclass SVM using the all-pairs reduction to binary classification (Beygelzimer et al., 2005).

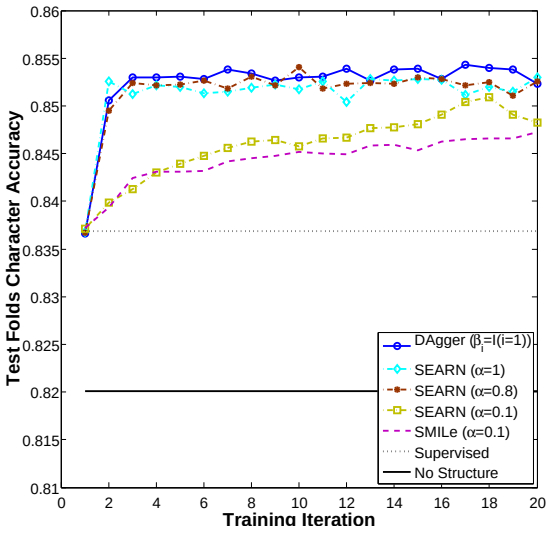

Figure 5: Character accuracy as a function of iteration.

predicted character feature) this makes this approach not as unstable as in general reinforcement/imitation learning problems (as we saw in the previous experiment). 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 than simple greedy forward decoding for structured prediction, as well as using base classifiers that rely on Inverse Optimal Control (Abbeel and Ng, 2004; Ratliff et al., 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.

Acknowledgements

This work is supported by the ONR MURI grant N00014-09-1-1052, Reasoning in Reduced Information Spaces, and by the National Sciences and Engineering Research Council of Canada (NSERC).

References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 2009.

A. Beygelzimer, V. Dani, T. Hayes, J. Langford, and B. Zadrozny. Error limiting reductions between classification tasks. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.

L. Bottou. sgd code, 2009. URL `http://www.leon.bottou.org/projects/sgd`.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. 2004.

S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. 2009.

H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning*, 2009.

E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the 19th annual conference on Computational Learning Theory (COLT)*, 2006.

M. Kääriäinen. Lower bounds for reductions, 2006. Atomic Learning workshop.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, 2002.

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

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

N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

N. Ratliff, J. A. Bagnell, and M. Zinkevich. (Online) subgradient methods for structured prediction. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

S. Ross. Comparison of imitation learning approaches on Super Tux Kart, 2010a. URL `http://www.youtube.com/watch?v=V00npNnWzSU`.

S. Ross. Comparison of imitation learning approaches on Super Mario Bros, 2010b. URL `http://www.youtube.com/watch?v=anOI0xZ3kGM`.

S. Ross and J. A. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

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

D. Silver, J. A. Bagnell, and A. Stentz. High performance outdoor navigation from overhead data using imitation learning. In *Proceedings of Robotics Science and Systems (RSS)*, 2008.

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

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