Imitation Learning

Saurabh Gupta
Convert into a Supervised Training Problem

Most General Case

$$a_t \sim \pi(o_t, T, R)$$
Behavior Cloning

**Train Time**

Assume an expert $e$ can solve this MDP.

1. Ask the expert $e$ to solve this MDP.
2. Collect labeled dataset $D$ from expert.
3. Train a function $\pi(o_t)$ that mimics $\pi_e(o_t)$ on $D$.

**Test Time**

Train with back-propagation
Supervision from Human Expert

**Train Time**

1. Ask the expert to solve this MDP.
2. Collect labeled dataset $D$ from expert.
3. Train a function $\pi(o_t)$ that mimics $\pi_e(o_t)$ on $D$.

**Test Time**
Supervision from Human Algorithmic Expert

**Train Time**

1. Instrument the environment such that it becomes a known MDP.

   - Fully Observed System
   - Known or Learned Transition Function
   - Known Reward Function

2. Train a function $\hat{\pi}(o_t)$ that mimics $\pi(o_t, T, R)$

**Test Time**

- Fully Observed System
- Known or Learned Transition Function
- Known Reward Function

---

Train Time:

- $o_t, r_t$:
  - Agent: $a_t \sim \pi(o_t, T, R)$
  - World

Test Time:

- $o_t = s_t$
- $s_{t+1} \sim \hat{T}(s_t, a_t)$
- $R(s_{t+1}, s_t, a_t)$
- $o_t \equiv s_t$
Supervision from Algorithmic Expert

Deep Sensorimotor Learning

rll.berkeley.edu/deeplearningrobotics

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Supervision from Human Algorithmic Expert

Train with back-propagation

Behavior Cloning

Does it always work?

No, data mis-match problem
Fix Data Mis-Match Problem

**DAgger: Dataset Aggregation**

Collect labels on states visited by $\pi(o_t)$ instead of $\pi_e(o_t)$.

1. train $\pi_\theta(a_t|o_t)$ from human data $\mathcal{D} = \{o_1, a_1, \ldots, o_N, a_N\}$
2. run $\pi_\theta(a_t|o_t)$ to get dataset $\mathcal{D}_\pi = \{o_1, \ldots, o_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $a_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

---

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 \( \varepsilon \) under the distribution of states/observations encountered by the expert can make as many as \( \frac{T^2}{2\varepsilon} \) 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 \( \varepsilon \) 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...
Forward Training and SMILe Algorithm

Initialize \( \pi_1^0, \ldots, \pi_T^0 \) to query and execute \( \pi^* \).

\[ \text{for } i = 1 \text{ to } T \text{ do} \]

Sample \( T \)-step trajectories by following \( \pi^{i-1} \).

Get dataset \( D = \{(s_i, \pi^*(s_i))\} \) of states, actions taken by expert at step \( i \).

Train classifier \( \pi_i^i = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim D}(e_\pi(s)) \).

\( \pi_j^i = \pi_j^{i-1} \) for all \( j \neq i \)

\[ \text{end for} \]

Return \( \pi_1^T, \ldots, \pi_T^T \)

Algorithm 3.1: Forward Training Algorithm.

Initialize \( \pi^0 \leftarrow \pi^* \) to query and execute expert.

\[ \text{for } i = 1 \text{ to } N \text{ do} \]

Execute \( \pi^{i-1} \) to get \( D = \{(s, \pi^*(s))\} \).

Train classifier \( \hat{\pi}^{*i} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim D}(e_\pi(s)) \).

\( \pi^i = (1 - \alpha)^i \pi^* + \alpha \sum_{j=1}^{i} (1 - \alpha)^{j-1} \hat{\pi}^{*j} \).

\[ \text{end for} \]

Remove expert queries: \( \hat{\pi}^N = \frac{\pi^N - (1 - \alpha)^N \pi^*}{1 - (1 - \alpha)^N} \)

Return \( \hat{\pi}^N \)

Algorithm 4.1: The SMILe Algorithm.
DAgger

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in $\Pi$.

for $i = 1$ to $N$ do

Let $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$.
Sample $T$-step trajectories using $\pi_i$.
Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by $\pi_i$ and actions given by expert.
Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$.
Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.

end for

Return best $\hat{\pi}_i$ on validation.

**Algorithm 3.1: DAGGER Algorithm.**
Super Tux Kart

Figure 2: Average falls/lap as a function of training data.
weights in the mixture are updated at each iteration.

...ent descent (Ratliff et al., 2007; Bottou, 2009).

SVM objective with regularizer 

...tal of 27152 binary features (very sparse). The 

...cial items); a history of those features over the last 4 images is 

...scribe each cell (types of ground, enemies, blocks and other spe-

...we choose parameter 

...pletion) and run the methods for 20 iterations. For SMILe 

...iteration (each stage is about 150 data points if run to com-

...sions as performance was still improving at the end of the 

...considered. When using 

...SMILe, and also outperforms SEARN for all choice of 

...in those situations by encountering them at the later iter-

...perform much better as they eventually learn to get unstuck 

...stacle. On the other hand, all the other iterative methods 

...it. Since the expert always jumps over obstacles at a sig-

...is that under the learned controller, Mario is often stuck at 

...lar, a reason the supervised approach gets such a low score 

...particular errors the learned controller makes. In particu-

...from the expert demonstrations, as this does not help the 

...approach, performance stagnates as we collect more data 

...collected. Again here we observe that with the supervised 

...vals on the average distance travelled per stage at each it-

...uation as a function of the total number of training data 

...Figure 4 shows 95% confidence inter-

...uation approach. Figure 4 shows 95% confidence inter-

...Average Distance Travelled Per Stage

<table>
<thead>
<tr>
<th>Number of Training Data</th>
<th>Average Distance Travelled Per Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>D0.5</td>
</tr>
<tr>
<td>2</td>
<td>D0.9</td>
</tr>
<tr>
<td>3</td>
<td>Se1</td>
</tr>
<tr>
<td>4</td>
<td>Se0.4</td>
</tr>
<tr>
<td>5</td>
<td>Sm0.1</td>
</tr>
<tr>
<td>6</td>
<td>Sup</td>
</tr>
</tbody>
</table>

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

Figure 5: Character accuracy as a function of iteration.
Thank you