Reduction of Imitation Learning to No-Regret Online Learning

Stephane Ross

Joint work with Drew Bagnell & Geoff Gordon
Imitation Learning

- Expert Demonstrations
- Machine Learning Algorithm
- Policy
Imitation Learning

• Many successes:
  – Legged locomotion [Ratliff 06]
  – Outdoor navigation [Silver 08]
  – Helicopter flight [Abbeel 07]
  – Car driving [Pomerleau 89]
  – etc...
Example Scenario
Learning to drive from demonstrations

Input:
Camera Image

Output:
Steering in [-1,1]
- Hard left turn
- Hard right turn
Supervised Training Procedure

Expert Trajectories

Learned Policy: 

\[
\hat{\pi}_{sup} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim D(\pi^*)}[\ell(\pi, s, \pi^*(s))]
\]
Poor Performance in Practice
# Mistakes Grows Quadratically in T!

\[ J(\hat{\pi}_{\text{sup}}) \leq T^2 \varepsilon \]

Exp. # of mistakes over T steps

Avg. loss on \(D(\pi^*)\)

# time steps

Reason: Doesn’t learn how to recover from errors!

[Ross 2010]
Reduction-Based Approach & Analysis

Hard Learning Problem  

Easier Related Problem(s)

Performance: $f(\epsilon)$  

Performance: $\epsilon$

Example: Cost-sensitive Multiclass classification to Binary classification [Beygelzimer 2005]
Previous Work: Forward Training

• Sequentially learn one policy/step
• # mistakes grows linearly:
  \[ J(\pi_{1:T}) \leq T\varepsilon \]
• Impractical if \( T \) large

[Ross 2010]
Previous Work: SMILe

[Ross 2010]

• Learn stochastic policy, changing policy slowly
  – \( \pi_n = \pi_{n-1} + \alpha_n(\pi'_n - \pi^*) \)
  – \( \pi'_n \) trained to mimic \( \pi^* \) under \( D(\pi_{n-1}) \)
  – Similar to SEARN [Daume 2009]

• Near-linear bound:
  – \( J(\pi) \leq O(T\log(T)\epsilon + 1) \)

• Stochasticity undesirable

Steering from expert
DAgger: Dataset Aggregation

- Collect trajectories with expert $\pi^*$
DAgger: Dataset Aggregation

- Collect trajectories with expert $\pi^*$
- Dataset $D_0 = \{(s, \pi^*(s))\}$
DAgger: Dataset Aggregation

• Collect trajectories with expert $\pi^*$

• Dataset $D_0 = \{(s, \pi^*(s))\}$

• Train $\pi_1$ on $D_0$
DAgger: Dataset Aggregation

- Collect new trajectories with $\pi_1$
DAgger: Dataset Aggregation

• Collect new trajectories with $\pi_1$

• New Dataset $D_1' = \{(s, \pi^*(s))\}$

Steering from expert
DAgger: Dataset Aggregation

- Collect new trajectories with $\pi_1$
- New Dataset $D_1' = \{(s, \pi^*(s))\}$
- Aggregate Datasets:
  $$D_1 = D_0 \cup D_1'$$

Steering from expert
DAgger: Dataset Aggregation

• Collect new trajectories with $\pi_1$

• New Dataset $D_1' = \{(s, \pi^*(s))\}$

• Aggregate Datasets:
  $\quad D_1 = D_0 \cup D_1'$

• Train $\pi_2$ on $D_1$

Steering from expert
DAgger: Dataset Aggregation

- Collect new trajectories with $\pi_2$
- New Dataset $D_2' = \{(s, \pi^*(s))\}$
- Aggregate Datasets: $D_2 = D_1 \cup D_2'$
- Train $\pi_3$ on $D_2$

Steering from expert
DAgger: Dataset Aggregation

• Collect new trajectories with $\pi_n$

• New Dataset $D_n' = \{(s, \pi^*(s))\}$

• Aggregate Datasets:
  $D_n = D_{n-1} \cup D_n'$

• Train $\pi_{n+1}$ on $D_n$
Online Learning

Learner

Adversary
Online Learning
Online Learning

Learner

\[ \text{Current Hypothesis } h_n \]

Adversary

\[ \text{Pick Loss } L_n \]
Online Learning

Learner

Adversary

Current Hypothesis $h_n$

Pick Loss $L_n$

Next Hypothesis $h_{n+1}$
Online Learning

Learner

Adversary

Current Hypothesis $h_n$

Pick Loss $L_n$

Next Hypothesis $h_{n+1}$

Pick Loss $L_{n+1}$
Online Learning

Learner

Adversary

Current Hypothesis $h_n$

Pick Loss $L_n$

Next Hypothesis $h_{n+1}$

Pick Loss $L_{n+1}$

Avg. Regret: $\gamma_n = \frac{1}{n} \left[ \sum_{i=1}^{n} L_i(h_i) - \min_{h \in H} \sum_{i=1}^{n} L_i(h) \right]$
DAgger as Online Learning

Learner

Current Policy $\pi_n$

Pick Loss $L_n$

Next Policy $\pi_{n+1}$

Adversary

Pick Loss $L_{n+1}$

$$L_n(\pi) = \mathbb{E}_{s \sim D(\pi_n)}[\ell(\pi, s, \pi^*(s))]$$
DAgger as Online Learning

\[ \pi_{n+1} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{n} L_i(\pi) \]

\[ L_n(\pi) = \mathbb{E}_{s \sim D(\pi_n)} \left[ \ell(\pi, s, \pi^*(s)) \right] \]
DAgger as Online Learning

\[ \pi_{n+1} = \arg \min_{\pi \in \Pi} \sum_{i=1}^{n} L_i(\pi) \]

Follow-The-Leader (FTL)

Adversary

Current Policy \( \pi_n \)

Next Policy \( \pi_{n+1} \)

Pick Loss \( L_n \)

Pick Loss \( L_{n+1} \)

\[ L_n(\pi) = \mathbb{E}_{s \sim D(\pi_n)} [\ell(\pi, s, \pi^*(s))] \]
Theoretical Guarantees of DAgger

- Best policy $\pi$ in sequence $\pi_{1:N}$ guarantees:

$$J(\pi) \leq T(\varepsilon_N + \gamma_N) + O(T/N)$$

**Avg. Loss on Aggregate Dataset**

**Avg. Regret of $\pi_{1:N}$**

**Iterations of DAgger**
Theoretical Guarantees of DAgger

• Best policy $\pi$ in sequence $\pi_{1:N}$ guarantees:
  \[
  J(\pi) \leq T(\varepsilon_N + \gamma_N) + O(T/N)
  \]
  Avg. Loss on Aggregate Dataset
  Avg. Regret of $\pi_{1:N}$
  Iterations of DAgger

• For strongly convex loss, $N = O(T \log T)$ iterations:
  \[
  J(\pi) \leq T\varepsilon_N + O(1)
  \]
Theoretical Guarantees of DAgger

• Best policy $\pi$ in sequence $\pi_{1:N}$ guarantees:

$$J(\pi) \leq T(\epsilon_N + \gamma_N) + O(T/N)$$

- Avg. Loss on Aggregate Dataset
- Avg. Regret of $\pi_{1:N}$
- Iterations of DAgger

• For strongly convex loss, $N = O(T\log T)$ iterations:

$$J(\pi) \leq T\epsilon_N + O(1)$$

• Any No-Regret algorithm has same guarantees
Theoretical Guarantees of DAgger

- If sample \( m \) trajectories at each iteration, w.p. \( 1 - \delta \):

\[
J(\pi) \leq T(\hat{\varepsilon}_N + \gamma_N) + O(T\sqrt{\log(1/\delta) / \sqrt{Nm}})
\]

Empirical Avg. Loss on Aggregate Dataset  
Avg. Regret of \( \pi_{1:N} \)
Theoretical Guarantees of DAgger

- If sample \( m \) trajectories at each iteration, w.p. 1-\( \delta \):
  \[
  J(\pi) \leq T(\hat{\epsilon}_N + \gamma_N) + O\left(T \sqrt{\log(1/\delta) / \sqrt{Nm}}\right)
  \]

  Empirical Avg. Loss on Aggregate Dataset

  Avg. Regret of \( \pi_{1:N} \)

- For strongly convex loss, \( N = O(T^2 \log(1/\delta)) \), \( m=1 \), w.p. 1-\( \delta \):
  \[
  J(\pi) \leq T\hat{\epsilon}_N + O(1)
  \]
Experiments: 3D Racing Game

Input: Resized to 25x19 pixels (1425 features)

Output: Steering in [-1,1]
DAgger Test-Time Execution
Average Falls/Lap

![Graph showing the average falls per lap versus the number of training data samples. The graph includes different lines for DAgger, SMILe (0.1), and Supervised learning methods.]
Experiments: Super Mario Bros
From Mario AI competition 2009

Input:

Output:

Jump in \{0,1\}
Right in \{0,1\}
Left in \{0,1\}
Speed in \{0,1\}

Extracted 27K+ binary features from last 4 observations
(14 binary features for every cell)
Test-Time Execution
Average Distance/Stage
Conclusion

• Take-Home Message
  – Simple iterative procedures can yield much better performance.

• Can also be applied for **Structured Prediction**:
  – NLP (e.g. Handwriting Recognition)
  – Computer Vision [Ross & al., CVPR 2011]

• Future Work:
  – Combining with other Imitation Learning techniques [Ratliff 06]
  – Potential extensions to Reinforcement Learning?
Questions
Structured Prediction

• Example: Scene Labeling

Image

Graph Structure over Labels
Structured Prediction

• Sequentially label each node using neighboring predictions
  – e.g. In Breath-First-Search Order (Forward & Backward passes)
Structured Prediction

• Input to Classifier:
  – Local image features in neighborhood of pixel
  – Current neighboring pixels’ labels

• Neighboring labels depend on classifier itself

• DAgger finds a classifier that does well at predicting pixel labels given the neighbors’ labels it itself generates during the labeling process.
Experiments: Handwriting Recognition

Input:

Image current letter:

Previous predicted letter:

Output:

Current letter in \{a,b,...,z\}

[Taskar 2003]
Test Folds Character Accuracy

Better