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

Stéphane Ross, Geoffrey J. Gordon, J. Andrew Bagnell

Presenter: Amit Joshi

21st October 2021

Main Problem

- Sequential prediction problems where future observations depend on previous predictions/actions violate the independently and identically distributed (i.i.d.) assumptions, leading to poor performance
- Due to compounding error, if a wrong action is taken, then the observations in far-future time steps will be vastly different
  - Ex: a robot who turned right instead of left will have very different observations next timestep
- This paper aims to learn a STATIONARY policy that achieves a linear bound in the number of errors
  - Stationary - same policy for each timestep

Main Problem: Compounding Error

- A typical stationary policy with a task horizon T and mistake probability ϵ is expected to make $T^2\epsilon$ mistakes over the task horizon (check proof in paper by Ross and Bagnell)
- Intuitively, making a mistake at timestep 1 will greatly alter the observations seen

Motivation

- Sequence Prediction problems are common in robotics
- Most systems must make a sequence of actions that affect their observations
- Imitation Learning (learning to match an expert's actions) has achieved SOTA performance in a variety of tasks
  - Typically trains a classifier or regressor to produce an expert's actions given observations
- Aim is to come up with a no-regret online (on-the-fly) policy using imitation learning
- Nonstationary policies have been developed but this is infeasible if T is large or unknown

Problem Setting

- π: the class of policies
- $d_\pi = \frac{1}{T}\sum_{t=1}^{T} d_\pi^t$: represents the average distribution of states after following π policy for T timesteps
- $C_\pi(s) = \mathbb{E}_{a\sim\pi(s)}[C(s,a)]$ cost of acting according to policy π for one timestep
- $J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{s\sim d_\pi^t}[C_\pi(s)] = T\mathbb{E}_{s\sim d_\pi}[C_\pi(s)]$: Total expected cost

Problem Setting: Surrogate Loss Function

- May not know exact cost C(s,a)
- Can upper bound $J(\pi)$ by using a surrogate loss function, which is greater than or equal to actual cost.
- In Imitation Learning, can be any of the following:
  - 0-1 Loss (whether the action matches with expert or not)
  - Squared Loss
  - Actual cost, C itself
- Want to find the policy which minimizes this surrogate loss:
  - $\hat{\pi} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim d_\pi}[\ell(s, \pi)]$
- Note the dependence of the distribution on the policy itself, not i.i.d.

Related Work: Forward Training

- Trains a nonstationary policy for T iterations, where π_t is trained to mimic the expert on the distribution of states resulting from acting from policies $\pi_1, \pi_2, \ldots \pi_{t-1}$
- Num mistakes grows linearly in T, (proof in paper)
- Need to run T iterations

Initialize $\pi_1^0, \ldots, \pi_T^0$ to query and execute π^*.
for $i = 1$ **to** T **do**
  Sample T-step trajectories by following π^{i-1}.
  Get dataset $\mathcal{D} = \{(s_i, \pi^*(s_i))\}$ of states, actions taken by expert at step i.
  Train classifier $\pi_i^i = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim \mathcal{D}}(e_\pi(s))$.
  $\pi_j^i = \pi_j^{i-1}$ for all $j \neq i$
end for
Return $\pi_1^T, \ldots, \pi_T^T$

Algorithm 3.1: Forward Training Algorithm.

Related Work: Stochastic Mixing Iterative Learning (SMILe)

- Stationary
- Starts with π_0 which simply executes expert's choice
- Trains $\hat{\pi}_n$ to mimic expert on current distribution of trajectories
- Updates $\pi_n = \pi_{n-1} + \alpha(1-\alpha)^{n-1}(\hat{\pi}_n - \pi_0)$
- Guarantees near linear regret in T and ϵ

Related Work: Stochastic Mixing Iterative Learning (SMILe)

Initialize $\pi^0 \leftarrow \pi^*$ to query and execute expert.
for $i = 1$ **to** N **do**
  Execute π^{i-1} to get $\mathcal{D} = \{(s, \pi^*(s))\}$.
  Train classifier $\hat{\pi}^{*i} = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim \mathcal{D}}(e_\pi(s))$.
  $\pi^i = (1-\alpha)^i \pi^* + \alpha \sum_{j=1}^{i} (1-\alpha)^{j-1} \hat{\pi}^{*j}$.
end for
Remove expert queries: $\tilde{\pi}^N = \frac{\pi^N - (1-\alpha)^N \pi^*}{1-(1-\alpha)^N}$
Return $\tilde{\pi}^N$

Algorithm 4.1: The SMILe Algorithm.

Proposed method: DAGGER (dataset aggregation) algorithm

- Uses expert's policy to generate dataset of trajectories (state, expert action)
- Trains $\hat{\pi}_2$ on dataset to mimic expert, create even more trajectories with $\hat{\pi}_2$
- Repeat

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.

Proposed method: DAGGER expert-use optimization

- When populating the dataset of new trajectories, may want to use expert

$$\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$$

- Especially helpful in first few iterations, when policy is randomly initialized
  - Exponentially decay β_i over iterations
- Note: π^* is expert, $\hat{\pi}_i$ is our policy trained on $\mathcal{D}$, π_i adds (aggregates) to dataset

Proposed method: DAGGER significance

- Can guarantee bounds on total cost that are near-linear if infinite samples are taken per iteration (see paper for proofs)
  - Assuming ℓ (surrogate loss) convex and bounded over all policies
  - Assuming $\beta_i < (1-\alpha)^{i-1}$ for some constant α
  - $\epsilon_N = \min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_{\pi_i}} [\ell(s, \pi)]$: true cost of best policy in hindsight
- **Theorem 3.1.** *For* DAGGER*, if* N *is* $\tilde{O}(T)$ *there exists a policy* $\hat{\pi} \in \hat{\pi}_{1:N}$ *s.t.* $\mathbb{E}_{s \sim d_{\hat{\pi}}}[\ell(s, \hat{\pi})] \leq \epsilon_N + O(1/T)$

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

Proposed method: DAGGER significance, continued

- What about the finite sample case, when only a finite number of samples are sampled per iteration?
- $\hat{\epsilon}_n = \min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim D_i}[\ell(s, \pi)]$
- **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.* $\mathbb{E}_{s \sim d_{\hat{\pi}}}[\ell(s, \hat{\pi})] \leq \hat{\epsilon}_N + O(1/T)$
- **Theorem 3.4.** *For* DAGGER, *if* N *is* $O(u^2 T^2 \log(1/\delta))$ *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.* $J(\hat{\pi}) \leq J(\pi^*) + uT\hat{\epsilon}_N + O(1)$.

Theory: Online Learning and No-Regret Algorithms

- An online learning algo applies π_i and incurs loss $\ell_i(\pi_i)$ for each iteration
- No-regret algo produces sequences of policies $\pi_1, \pi_2, \ldots \pi_N$ average regret compared to the overall best policy in hindsight goes to zero as N goes to infinity. Description below:

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

Theory: Using DAGGER to give a No-Regret algo

- Using the expert-use optimization, we can bound the difference between states seen by policies that samples trajectories (π_i), and policy that we are training ($\hat{\pi}_i$)

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 $\beta \in [0,1]$. □

Theory: Using No-Regret Algorithms to show that DAGGER gives good performance

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* $\hat{\pi}_{1:N}$.

- n_β is the largest constant $n \leq N$ such that $\beta_n > \frac{1}{T}$
- General Idea for proof: last lemma bounded distribution of states between sampled trajectories policy (which uses expert with some probability) and policy we are learning.
- So, we can bound the policy we are learning with the best policy's loss in hindsight.

Theory: Using No-Regret Algorithms to show that DAGGER gives good performance (finite sample case where m samples are picked per iteration

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* $\hat{\pi}_{1:N}$.

- The last term comes from the fact that

  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 $1-\delta$.
- Y_{ij} is difference between expected per step loss at iteration i and the average per step loss of sample j at iteration i
- See paper for full proof details

Experimental Setup: Super Tux Kart

- Expert: Human expert that controls the joystick (analog value in range [-1,1]), y_i
- Features: LAB color values of resized image, x
- Output: Steering value $\hat{y} = w^\top x + b$ that minimizes ridge regression objective
- Objective: $L(w, b) = \frac{1}{n} \sum_{i=1}^{n} (w^\top x + b - y_i) + \lambda w^\top w$
- regularizer $\lambda = 10^{-3}$
- Baseline: SMILe, Supervised
- Metrics: falls per lap (of star track)
- 1 lap of training per iteration, 20 iterations
- Agent moves at frequency 5 Hz

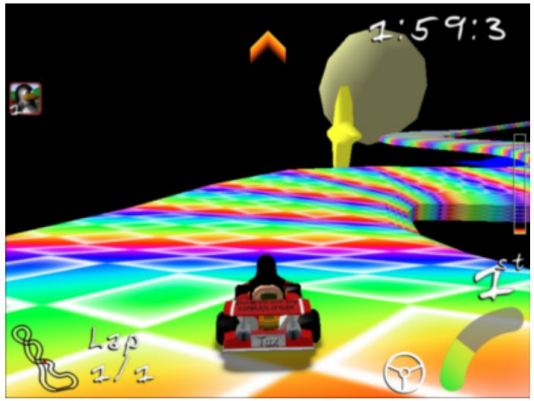

Figure 1: Image from Super Tux Kart's Star Track.

Experimental Results: Super Tux Kart

- DAGGER never falls off after 15 iterations
- significantly outperforms baseline
- Supervised falls go up after too many iterations
- SMILe did not improve after 15 iterations

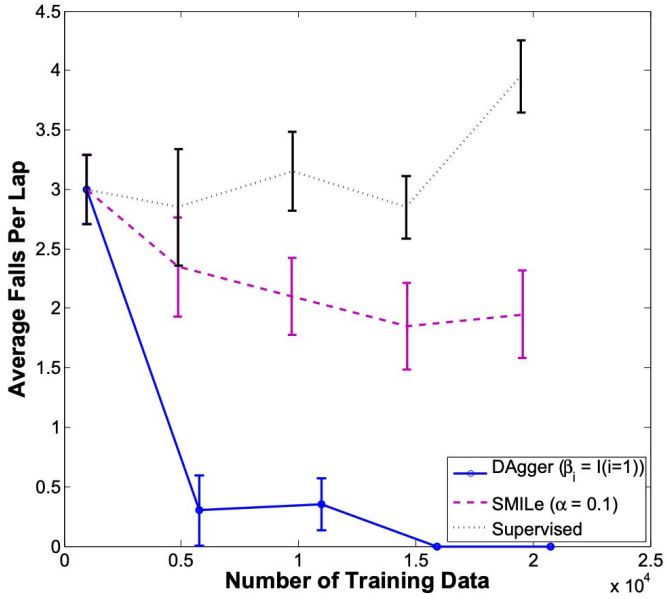

Figure 2: Average falls/lap as a function of training data.

Experimental Setup: Super Mario Bros

- Expert: Near-optimal planning algorithm will full access to game state, can simulate consequences of actions exactly
- Stages are generated from simulation to diversify enemies and difficulty
- Action: subset of {left, right, up, speed}. Chosen by 4 linearly independent SVMs
- Image broken down into 22x22 grid, containing info about enemies, special items, etc as binary features. Plus same info from few timesteps ago
- $\hat{y}_k = I(w_k^T x + b_k > 0)$, kth binary feature, with weight/bias trained to optimize SVM objective
- 20 iterations, 5000 data points per iteration (each stage is ~150 data points)

Experimental Setup: Super Mario Bros

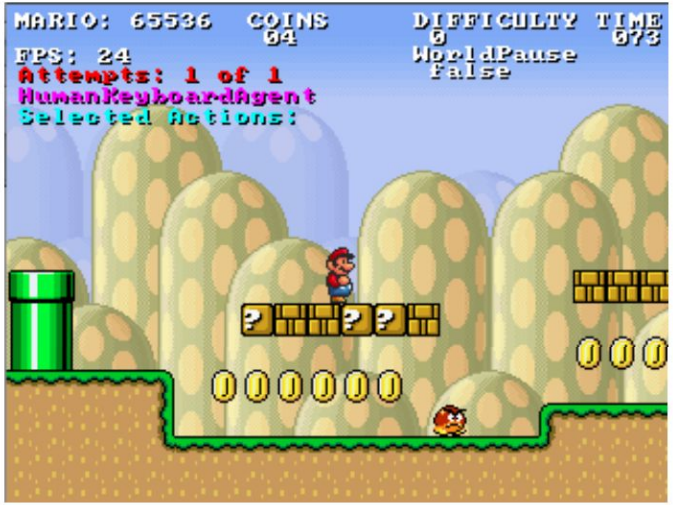

Figure 3: Captured image from Super Mario Bros.

Experimental Results: Super Mario Bros

- Tested different beta values for dagger expert use optimization
- 0.5 worked the best
- 0.9 caused slow convergence expert shouldn't generate trajectories frequently as iterations goes on
- Supervised performs poorly

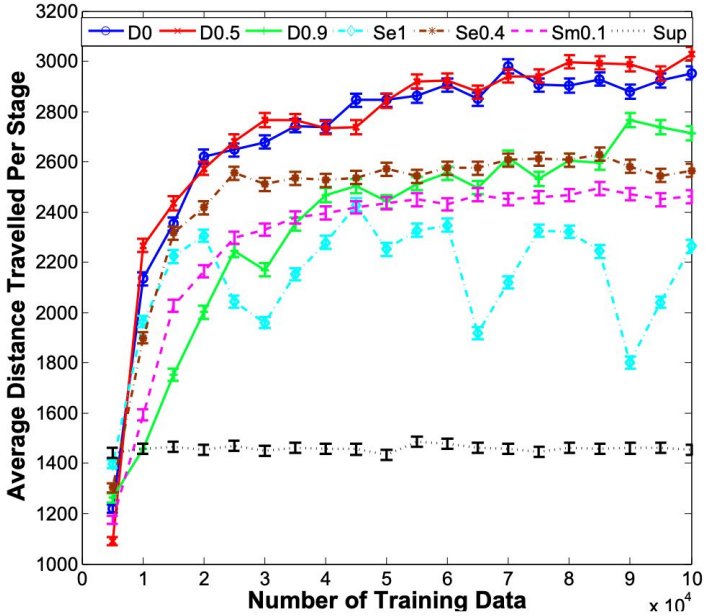

Figure 4: Average distance/stage as a function of data.

Experimental Setup: Handwriting Recognition

- ~6600 word dataset (~52000) characters
- Uses SVM to predict word in left to right character order, uses previously predicted character (tries to predict pairs)
- Each character has 128 features (pixels), plus 26 binary features for previous character
- Multiclass SVM was modeled as reduction to all pairs binary classification
- Baselines: SMILe, SEARN (similar to policy iteration)
- More baselines:
  - Two non-structured methods that don't consider previously predicted character
  - Supervised approach where training is conducted with previous character correctly labelled

Experimental Results: Handwriting Recognition

- Supervised performed better than no structure, showing that structure works
- DAGGER outperformed supervised (83.6% vs 85.5%)
- Pure Policy iteration (SEARN $\alpha = 1$) performed well, since policy doesn't influence current state much (only affects previously predicted character)

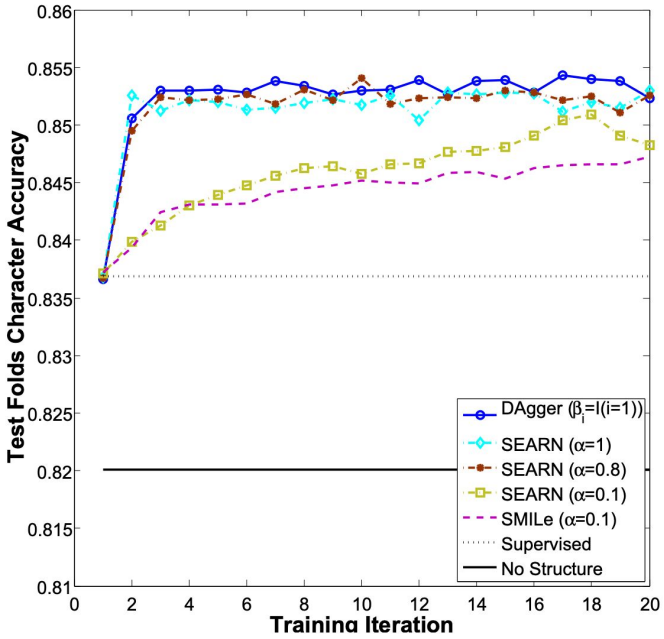

Figure 5: Character accuracy as a function of iteration.

Discussion of Results

- The Handwriting Experiment showed that policy iteration works when policy doesn't influence state much. The task horizon was only 2 for any given prediction, so the error didn't compound much.
- The Super Tux Kart experiment showed how the stochasticity of SMILe is a disadvantage, occasionally led to bad actions. This is visually apparent; the SMILe agent makes less smooth movements compared to DAGGER agent.
- The Super Mario Bros experiment showed that the expert use optimization is important as a fine balance must be found between generating expert "optimal" trajectories and our nonexpert policy's trajectories
    - Must observe nonexpert states like being stuck at an obstacle, but also collect a wider variety of data by using expert

Critique/Limitations/Open Issues

- Their linear error proof was based on choosing expert action at least once, what if expert action was never chosen?
- Could using expert policy more often to generate trajectories be a good idea when policy learned appears to be close to expert policy?
- When would using policy iteration work just as well, as was the case with Handwriting Recognition?
- Practical challenge: how would you use learned policy to generate many trajectories quickly in real-world, (as is necessary in DAGGER)?

Future Work for Paper / Reading

- Will consider more sophisticated strategies for decoding than simple greedy forward
  - perhaps try beam search
- I would like to see how their handwriting recognition compares to a Connectionist Temporal Classification (CTC) baseline
  - Uses a reduction to Hidden Markov Models and uses the HMM Forward Algo to give the most likely word
- Using base classifiers that rely on Inverse Optimal Control techniques to learn a cost function for a planner to aid prediction in imitation learning
- Keep using similar techniques as DAGGER (cost-to-go-method)

Extended Readings

- Talks about Forward Training and SMILe in greater detail: 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.
- More experiments: S. Ross. Comparison of imitation learning approaches on Super Mario Bros, 2010b. URL http://www. youtube.com/watch?v=anOI0xZ3kGM.
- Inverse optimal control: P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the 21st International Conference on Machine Learning (ICML), 2004.
- Inverse optimal control: 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.

Summary

- Aims to fix the compounding error problem by using policy learned to generate trajectories instead of expert (as in supervised imitation learning)
- Previous approaches have nonstationary policies (different for each timestep) or stochastic attempts where mixing policies can lead to poor performance
- Aggregating dataset from learned policy performs better
  - Expert policy should be used to help sample, especially in the beginning when learned policy is poor
- Proven that in finite sample case, the number of mistakes DAGGER makes grows approximately linearly in number of iterations
- Proven (using no-regret guarantees) that with high probability, DAGGER gives good performance guarantees.

Thank you!! Here is the full paper I based my presentation off of

Citation: Ross, S.; Gordon, G. J.; and Bagnell, J. A. 2011. No-regret reductions for imitation learning and structured prediction. Aistats 15: 627–635. URL http://proceedings.mlr.press/v15/ ross11a/ross11a.pdf.

Link: https://arxiv.org/pdf/1011.0686.pdf