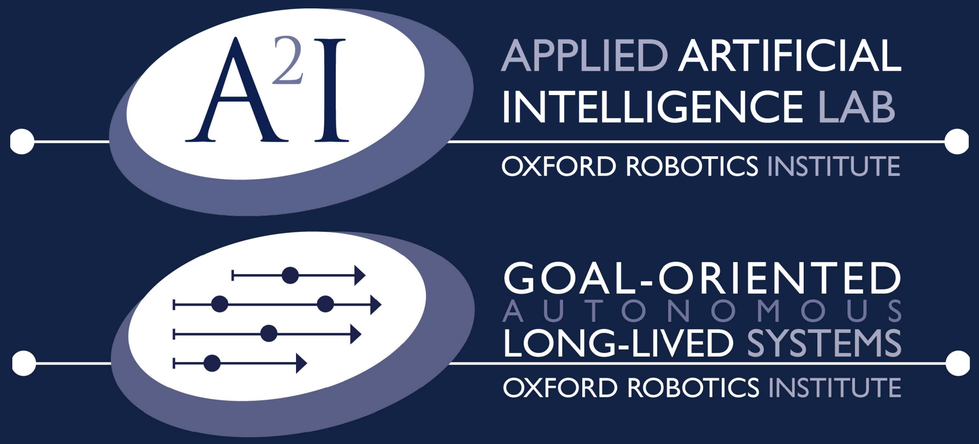
Imitation Learning Review

Branton DeMoss

March 2022



Policy Learning

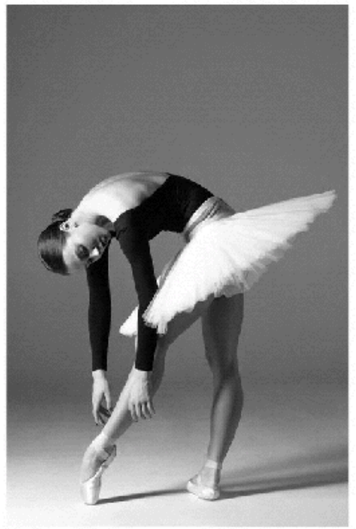
One approach to learning good robotic control policies is Reinforcement Learning (RL), which learns a policy that maximizes future expected rewards. Limitations

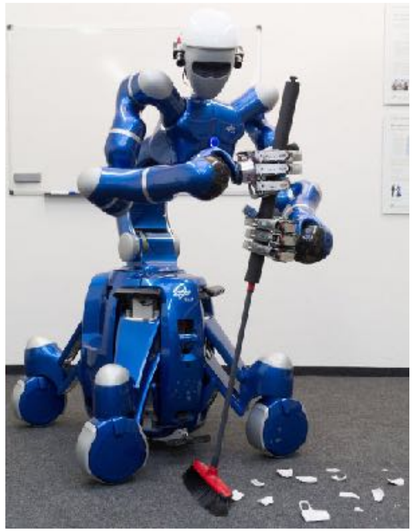
1. Rewards must be specifiable. Consider tasks like dancing, driving, or dextrous manipulation.

2. Not sample efficient, requiring many thousands or millions of trials to learn good policies.

3. Because of 2, it is difficult to get working in the real world, and has found most of its success in simulable environments.

    - Training in simulation introduces the so-called sim2real problem.





Policy Gradients

Policy gradients let us distill accumulated rewards into policies. *Even if those rewards are non-differentiable:*

- Return function: $J(\theta) = \sum_{s \in S} d^{\pi}(s) V^{\pi}(s)$
- How to maximize this? Policy gradient theorem:

$$\begin{aligned} \nabla_{\theta} J(\theta) &\propto \mathbb{E}_{\pi} \left[Q^{\pi}(s,a) \nabla_{\theta} \log \pi_{\theta}(a \,|\, s) \right] \\ &= \mathbb{E}_{(s,a,R) \sim \tau} \left[R \nabla_{\theta} \log \pi_{\theta}(a \,|\, s) \right] \end{aligned}$$

- Where τ are real sampled trajectories. These gradients have *high variance* and require many steps to converge a policy.
- Policy gradients are an example of an *on-policy* RL method.

Imitation Learning

Imitation Learning (IL) is a set of learning techniques which use demonstration datasets to directly learn policies. Three main branches of Imitation Learning include:

1. Behaviour Cloning (BC)

$$\max_{\theta} \mathbb{E}_{s\sim d_{\pi^*}, a\sim\pi^*(s)} \left[\log \pi_\theta(a \mid s)\right]$$

2. Inverse Reinforcement Learning (IRL)

$$\arg\min_{\psi} \left| \mathbb{E}_{\pi^{r_\psi}} \left[\mathbf{f}(\mathbf{s}, \mathbf{a})\right] - \mathbb{E}_{\pi^*} \left[\mathbf{f}(\mathbf{s}, \mathbf{a})\right] \right|$$

3. Adversarial Imitation Learning ($\subseteq$ IRL)

$$\psi = \arg\max_{\psi} \frac{1}{N} \sum_{x\sim p^*} \log D_\psi(\mathrm{x}) + \frac{1}{M} \sum_{x\sim p_\theta} \log\left(1 - D_\psi(\mathrm{x})\right)$$

Imitation Learning

Why run Imitation Learning[1]? It addresses almost all of the listed problems with RL:

- Sample Efficiency
  - Highly supervised problem!
- Safety/Exploration
  - Directly copy nearly optimal expert policy - don't explore!
- Unknown rewards
  - If we know expert is optimal for task we care about, don't care about rewards.

So what *is* a problem?

[1] *Should I Run Offline Reinforcement Learning or Behavioral Cloning? ICLR 2022 Blind Submission.*

Covariate Shift, informally

The problem of distributional mismatch in Imitation Learning has long been understood:

"[...] *the network must not solely be shown examples of accurate driving, but also how to recover once a mistake has been made.*" - Pomerleau (1989)

- The fundamental problem is that the training and test distributions do not match:

$$d^{\pi}(s) \neq d^{*}(s)$$

- Intimately related to the Offline RL Q estimation error problem. Intuitively, agent has not seen *its own* induced state distribution, and fails to recover at test time.

- "*In supervised learning, the model is a function of the data. In RL, the data is a function of the model.*"

ALVINN

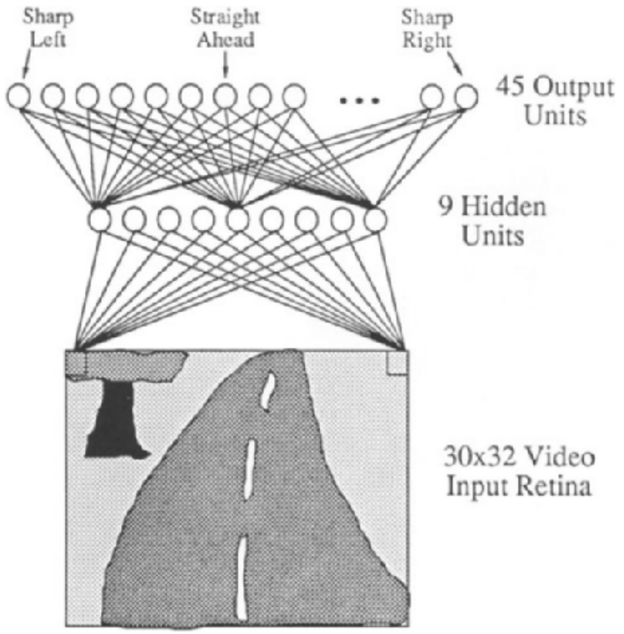


Figure 3: NAVLAB, the CMU autonomous navigation test vehicle.

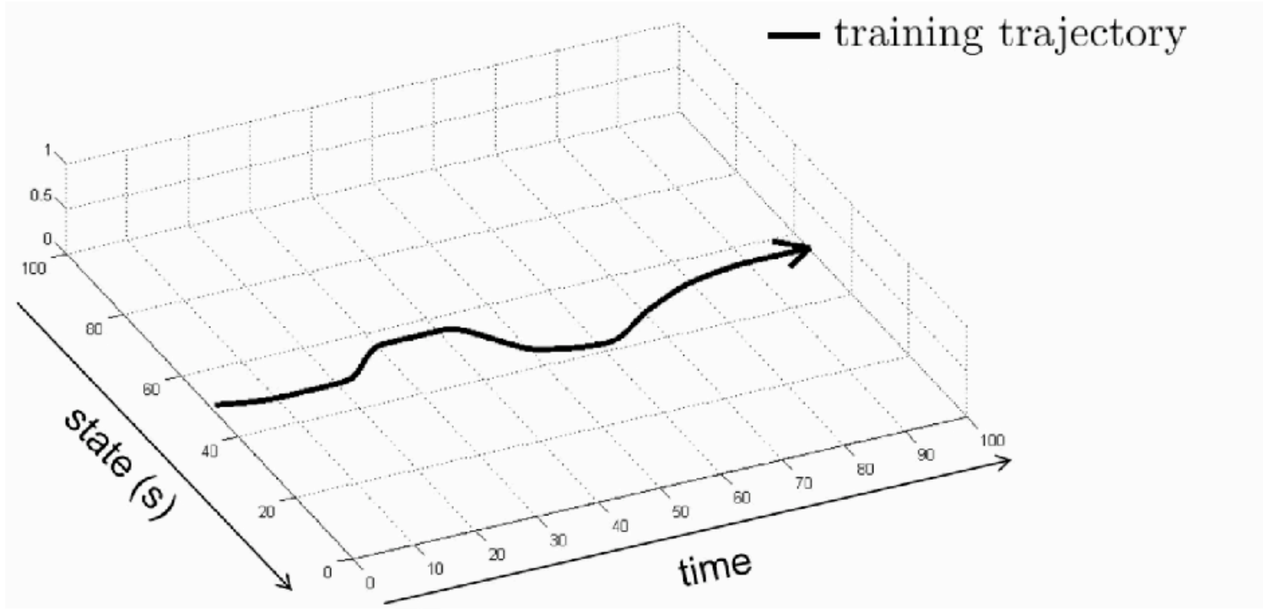
ALVINN: AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK

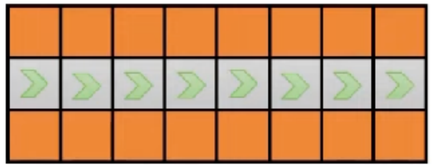
If you've just invented behavioral cloning, why not deploy it on "a modified Chevy van equipped with 3 Sun computers, a Warp, a video camera, and a laser range finder"?

Covariate Shift, more formally

Vanilla BC incurs quadratic regret:

1. Assume ϵ error in policy cloning
2. Decision process has length T
3. Cost function $c(s,a) = \begin{cases} 0 & \text{if } a = \pi^*(s) \\ 1 & \text{otherwise} \end{cases}$
4. $$\mathbb{E}\left[\sum_t c(s_t, a_t)\right] \leq \epsilon T + (1-\epsilon)(\epsilon(T-1) + (1-\epsilon)(\ldots))$$
$$\approx O(\epsilon T^2)$$
5. See *Efficient Reductions for Imitation Learning* (2010) for the general proof.





Correcting Behavior Cloning with Online Supervision

Correct compounding error with a simple procedure:

1. Do supervised behavior cloning
2. Act with the learned policy in the environment
3. Occasionally* query expert for corrective actions
4. Behavior clone the results - now we get optimal actions in our own distribution

This is DAgger. It provably reduces regret to $O(\epsilon T)$! We just need an oracle and optimal corrections.

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.

*Whole thread of literature on optimizing this

Assumptions

Three key assumptions underlie successful Imitation Learning in the DAgger framework:

1. Optimality

$$\pi_\theta \leftarrow^{BC} \pi_{obs} \approx \pi^*$$

   - Observed expert demonstrations are optimal

2. Online setting

$$d^{\pi_\theta}(s) = ?$$

   - What is our induced distribution? Without an estimate, face **hard** bootstrapping errors.

3. Oracle access

$$\min_{s \sim d^{\pi_\theta}} D_{KL}\left[\pi_\theta(\cdot \mid s), \pi^*(\cdot \mid s)\right]$$

   Need oracle in our *own* distribution, not the optimal one!

   - Supervision on action to return to d^{π^*}

Disagreement Regularized Imitation Learning

How can we remove the need for an oracle in the DAgger framework?

1. Train an ensemble of policies $\Pi_E = \{\pi_1, \ldots, \pi_E\}$ with BC.

2. Define an ensemble uncertainty cost:

$$C_U(s,a) = \mathrm{Var}_{\pi \sim \Pi_E}(\pi(a \mid s)) = \frac{1}{E}\sum_{i=1}^{E}\left(\pi_i(a \mid s) - \frac{1}{E}\sum_{j=1}^{E}\pi_j(a \mid s)\right)^2$$

3. Optimize this in the online setting with Policy Gradients.

Will this always lead us back to the support?
What are the failure modes?

Algorithm 1 Disagreement-Regularized Imitation Learning (DRIL)

Input: Expert demonstration data $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$
Initialize policy π and policy ensemble $\Pi_{\mathrm{E}} = \{\pi_1, ..., \pi_E\}$
for $e = 1, E$ **do**
  Sample $\mathcal{D}_e \sim \mathcal{D}$ with replacement, with $|\mathcal{D}_e| = |\mathcal{D}|$.
  Train π_c to minimize $J_{\mathrm{BC}}(\pi_c)$ on $\mathcal{D}_c$ to convergence.
end for
for $i = 1, ...$ **do**
  Perform one gradient update to minimize $J_{\mathrm{BC}}(\pi)$ using a minibatch from $\mathcal{D}$.
  Perform one step of policy gradient to minimize $\mathbb{E}_{s \sim d_\pi, a \sim \pi(\cdot|s)}[C_{\mathrm{U}}^{\mathrm{clip}}(s,a)]$.
end for

DRIL Results

Regret is reduced to $O(\epsilon\kappa T)$, almost achieves DAgger regret bound.

1. Regret of a policy $\pi \in \Pi$ is: $R_\Pi(\pi) = J(\pi) - \min_{\pi' \in \Pi} J(\pi')$

2. Problem specific quantity κ which measures the tradeoff between the concentration of the demonstration data and the uncertainty of the ensemble outside of support.

$$\alpha(U) = \max_{\pi \in \Pi} \sup_{s \in U} \frac{d^\pi(s)}{d^*(s)}$$

$$\beta(U) = \min_{s \notin U, a \in A} \mathrm{Var}_{\pi \in \Pi_E} \left[\pi(a \mid s) \right]$$

$$\kappa = \frac{\alpha(U)}{\beta(U)}$$

In general, κ is problem specific and cannot be computed.

DRIL Results

They demonstrate superiority in a simple finite MDP.

Feels a little contrived, but demonstrates correctness in simple, understandable setting.

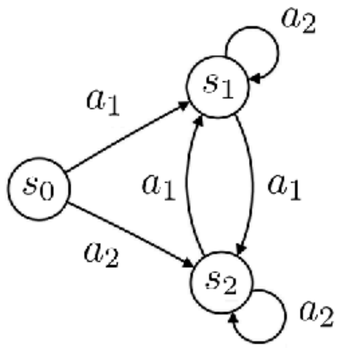


Figure 1: Example of a problem where behavioral cloning incurs quadratic regret.

Example 1. *Consider the tabular MDP given in (Ross & Bagnell, 2010) as an example of a problem where behavioral cloning incurs quadratic regret, shown in Figure 1. There are 3 states* $\mathcal{S} = (s_0, s_1, s_2)$ *and two actions* (a_1, a_2)*. Each policy* π *can be represented as a set of probabilities* $\pi(a_1|s)$ *for each state* $s \in \mathcal{S}$ [2]*. Assume the models in our ensemble are drawn from a posterior* $p(\pi(a_1|s)|\mathcal{D})$ *given by a Beta distribution with parameters* $Beta(n_1 + 1, n_2 + 1)$ *where* n_1, n_2 *are the number of times the pairs* (s, a_1) *and* (s, a_2) *occur, respectively, in the demonstration data* $\mathcal{D}$*. The agent always starts in* s_0 *and the expert's policy is given by* $\pi^\star(a_1|s_0) = 1, \pi^\star(a_1|s_1) = 0, \pi^\star(a_1|s_2) = 1$*. For any* (s, a) *pair, the task cost is* $C(s, a) = 0$ *if* $a = \pi^\star(s)$ *and 1 otherwise. Here* $d^\star_\pi = (\frac{1}{T}, \frac{T-1}{T}, 0)$*. For any* π*,* $d_\pi(s_0) = \frac{1}{T}$ *and* $d_\pi(s_1) \leq \frac{T-1}{T}$ *due to the dynamics of the MDP, so* $\frac{d_\pi(s)}{d^\star_\pi(s)} \leq 1$ *for* $s \in \{s_0, s_1\}$*. Writing out* $\alpha(\{s_0, s_1\})$*, we get:* $\alpha(\{s_0, s_1\}) = \max_{\pi \in \Pi} \sup_{s \in \{s_0, s_1\}} \frac{d_\pi(s)}{d^\star_\pi(s)} \leq 1.$

Furthermore, since s_2 *is never visited in the demonstration data, for each policy* π_i *in the ensemble we have* $\pi_i(a_1|s_2), \pi_i(a_2|s_2) \sim Beta(1, 1) = Uniform(0, 1)$*. It follows that* $\mathrm{Var}_{\pi \sim \Pi_E}(\pi(a|s_2))$ *is approximately equal* [3] *to the variance of a uniform distribution over* $[0, 1]$*, i.e.* $\frac{1}{12}$*. Therefore:*

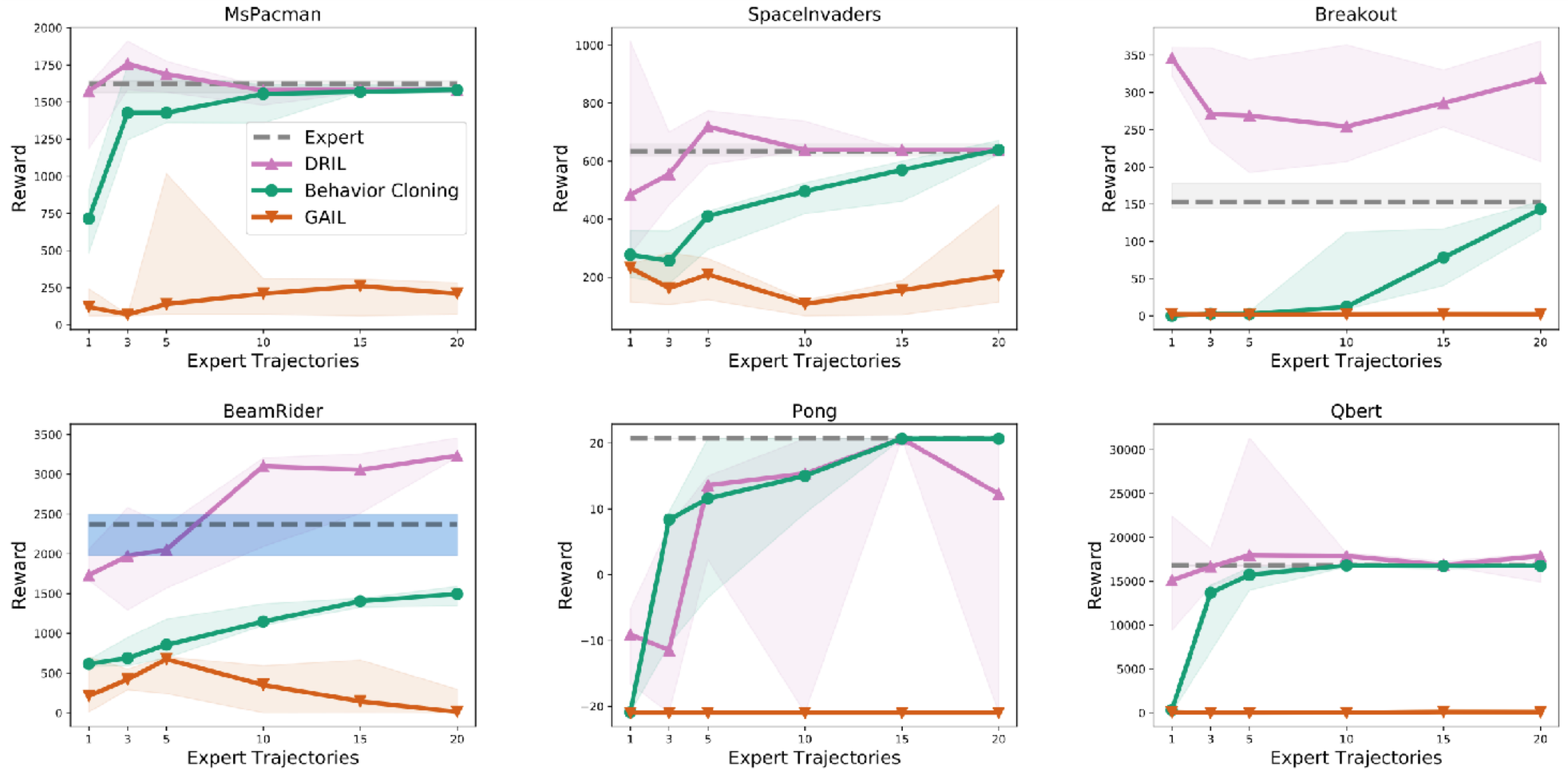
$$\kappa = \min_{\mathcal{U} \subseteq \mathcal{S}} \frac{\alpha(\mathcal{U})}{\beta(\mathcal{U})} \leq \frac{\alpha(\{s_0, s_1\})}{\beta(\{s_0, s_1\})} \lesssim \frac{1}{\frac{1}{12}} = 12$$

Applying our result from Theorem 3, we see that our algorithm obtains an $\mathcal{O}(\epsilon T)$ *regret bound on this problem, in contrast to the* $\mathcal{O}(\epsilon T^2)$ *regret of behavioral cloning* [4]*.*

Meta comment: People seem to really like the paper structure of: Background, Idea, Proof in simplified setting, Empirical superiority in more difficult settings.

DRIL Results

Nice results on Atari and DM Control!



Criticism: Only compares with BC and GAIL, ignores many more recent, competitive methods.

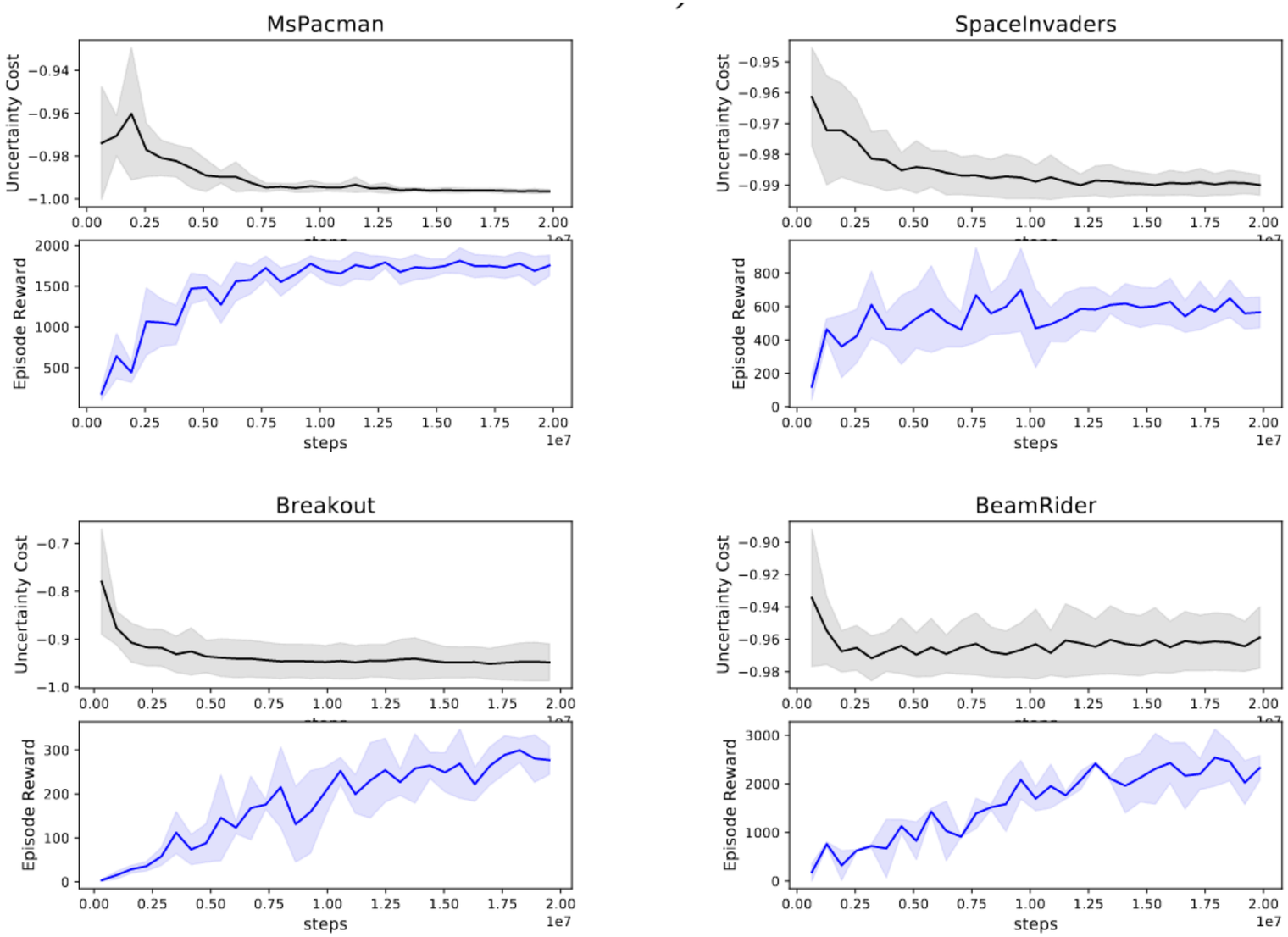
DRIL Results

Uncertainty decreases as training progresses, as we expect.

Does uncertainty matter? Could we use any function which distinguishes in- and out- of distribution samples?

~~Oracle~~

What about optimality?



Behavioral Cloning From Noisy Demonstrations

How can we address the optimality assumption of BC?

- Prior approaches to this issue involve expensive relabeling procedures, screening non-optimal demonstrations* or annotating the degree of non-optimality**.
- Idea: As long as sub-optimality is bounded, the *mode behavior* is optimal.
- Problem: How do we learn the mode? Only have a finite demonstration dataset.
- Solution: Estimate modal policy using an ensemble. Distill ensemble mode into student policy with REINFORCE gradients.

*Obtaining good performance from a bad teacher. In Programming by Demonstration vs. Learning from Examples Workshop at ML, volume 95, 1995.
**Wu et al. Imitation Learning from Imperfect Demonstration. PMLR 2019.

Behavioral Cloning From Noisy Demonstrations

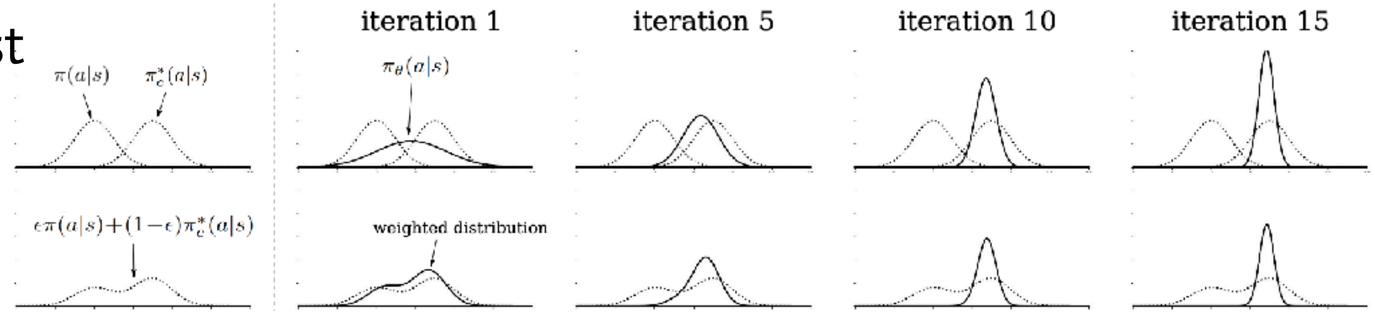
1. Initialize a set of policies $\Pi_K = \{\pi_{\theta^1}, \ldots, \pi_{\theta^K}\}$ and the ensemble $\pi_\theta(a \mid s) = \frac{1}{K} \sum_k \pi_{\theta^k}(a \mid s)$

2. Train each π_{θ^k} on disjoint subsets of the demonstration data, with the modified objective: $\mathbb{E}_{s \sim d^e, a \sim \pi^e(\cdot \mid s)} \left[\pi_\theta(a \mid s) \cdot \hat{R}(s, a) \right]$

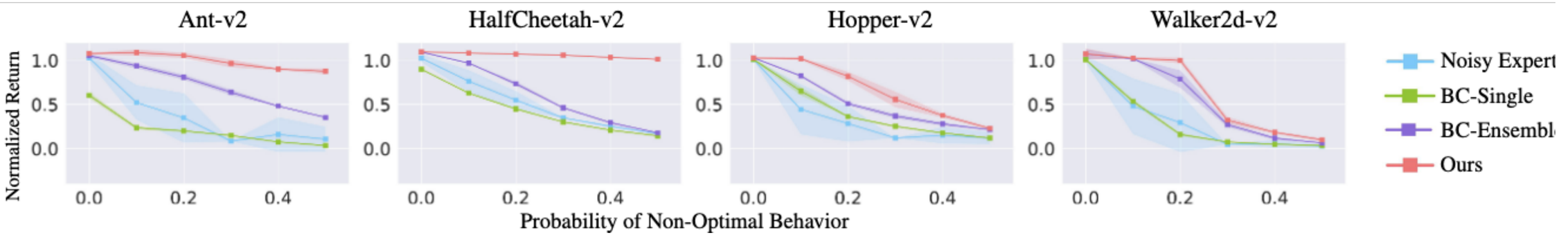
3. Update $\hat{R}$ each iteration to the latest ensemble policy action probs.

Algorithm 1 Behavioral Cloning from Noisy Demonstrations

Given the expert demonstrations $\mathcal{D}$.
Set $\hat{R}(s, a) = 1$ for $\forall (s, a) \in \mathcal{D}$.
Split $\mathcal{D}$ into K disjoint sets $\{\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^K\}$.
for iteration = $1, M$ **do**
  for $k = 1, K$ **do**
    Initialize parameters θ^k.
    for $l = 1, L$ **do**
      Sample a random minibatch of N state-action pairs (s_n, a_n) from $\mathcal{D}^k$.
      Calculate a sampled gradient $\frac{1}{N} \sum_{n=1}^{N} \nabla_{\theta^k} \log \pi_{\theta^k}(s_n, a_n) \cdot \hat{R}(s_n, a_n)$.
      Update θ^k by gradient ascent using the sampled gradient.
    end for
  end for
  Copy $\pi_{\theta_{old}} \leftarrow \pi_\theta$.
  Set $\hat{R}(s, a) = \pi_{\theta_{old}}(a|s)$ for $\forall (s, a) \in \mathcal{D}$.
end for
return π_θ.



BC From Noisy Demonstrations Results



Above: Method scales with ϵ much more favorably than alt approaches. Wish they showed full range of ϵ

Below: Vanilla BC indeed saturates, cannot scale with suboptimal demonstrations.

Table 1: The experimental results against IL methods that require the environment interactions.

	IC-GAIL	2IWIL	T-REX	GAIL	DRIL	Ours
Ant-v2	0.631 ± 0.162	1.042 ± 0.021	0.586 ± 0.124	0.003 ± 0.004	1.071 ± 0.023	1.055 ± 0.053
HalfCheetah-v2	0.941 ± 0.103	1.024 ± 0.059	0.001 ± 0.113	0.106 ± 0.003	0.065 ± 0.006	1.093 ± 0.092
Hopper-v2	1.233 ± 0.152	1.223 ± 0.135	0.441 ± 0.219	0.000 ± 0.001	0.910 ± 0.099	1.003 ± 0.045

Summary

The DAgger paradigm has been highly successful for studying IL. We've chiseled away at the limitations over the last decade:

1. Optimality assumption: Learn the mode.

2. Oracle supervision: Measure and minimize uncertainty to stay away from OOD.

DAgger has given IL firm theoretical grounding, and given us sharp problems to solve.

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.

We are algorithm- rather than data-limited


Thank you!