Algorithmic Human-Robot Interaction

Inverse Reinforcement Learning

CSCI 7000

Prof. Brad Hayes

Computer Science Department

University of Colorado Boulder

Reinforcement Learning

Function: $O \rightarrow A$

$\mathbf{o}$

$\pi_\theta(\mathbf{a}|\mathbf{o})$

$\mathbf{a}$

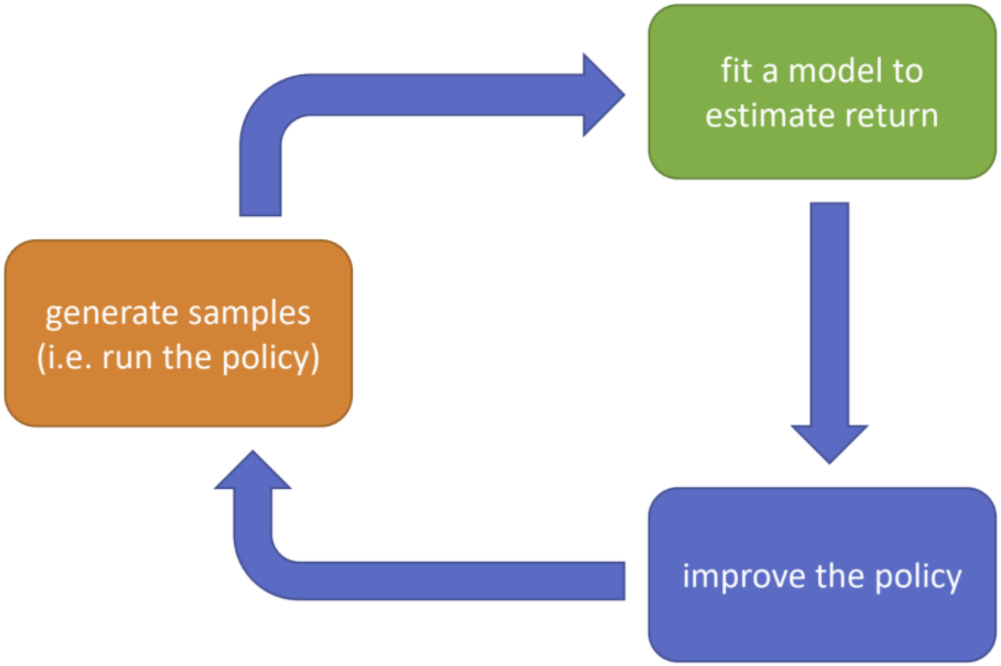
$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy

$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)

$\mathbf{o}_t$ – observation

$\mathbf{s}_t$ – state



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)

fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)

estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)

$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)

optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

Anatomy of an RL Algorithm

Trajectory: τ
Trajectory length: T
Policy: π
Reward: r
Parameters: θ
Gradient: ∇
J: Expected reward

Policy Differentiation

$$\theta^\star = \underbrace{\text{Best } \theta \text{ based on expected reward over the } T\text{-length trajectory}}_{J(\theta)}$$

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = \text{Probability of given trajectory using policy } \pi_\theta$$

$J(\theta) =$ Expected reward value given trajectory (τ) sampled from π_θ

Gradient of policy w.r.t. θ is equal to policy * gradient of log policy

$\nabla_\theta J(\theta) =$ Take gradient of $J(\theta)$ $=$ Use convenient identity to get rid of non-log policy $=$ Sample from policy instead of enumerating all possible trajectories

Expand policy(trajectory) and take log gradient

Rewrite substituting in worked-out gradient

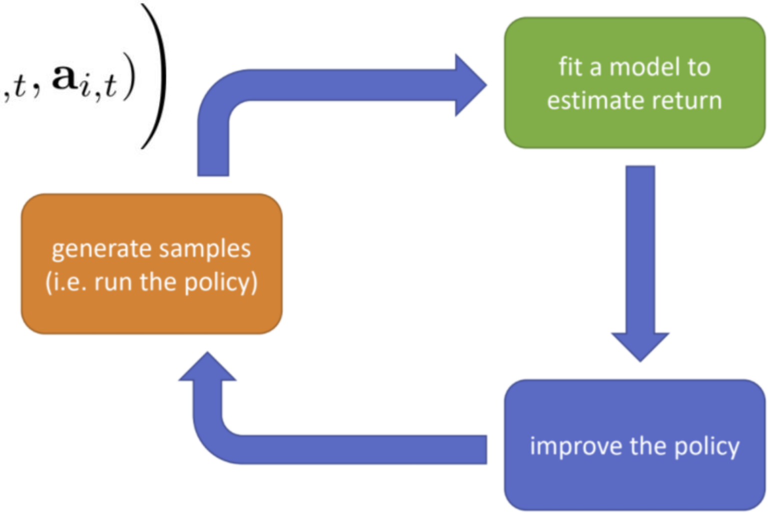
Using the Policy Gradient

$$\nabla_\theta J(\theta) = \boxed{E_{\tau \sim \pi_\theta(\tau)}}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\boxed{\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right)}\right]$$

$$\boxed{\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)}$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})\right)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau) r(\tau)$$



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Using the Policy Gradient

- **Recall:**
  - We defined a policy π as a function that says which action to do from a given world state
- **Policies are usually not discrete:**
  - A policy will typically output a likelihood for each action
  - In CartPole, $\pi(s) = [0.88, 0.12]$ would represent 88% chance to move left, and 12% chance to move right
- **How to turn an arbitrary function into probabilities?**
  - Softmax is one good option:
  - $\sigma(z, j) = \frac{e^{z_j}}{\sum_{k=0}^{|Z|-1} e^{z_k}}$

Using the Policy Gradient

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Q-Function replacement of reward sum:

$$\sum_t r(s_t^i, a_t^i) \rightarrow \hat{Q}_{i,t} = \sum_{t'=t}^{T} r(s_{t'}, a_{t'})$$

- **Where do we get $\nabla_\theta \log \pi_\theta$?**
  - Can derive it analytically if you have an expression for π_θ and do the calculus
  - Can derive it automatically if you're using a computation graph to express π_θ:

```
predictions = policy.predict(states)
nll = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=predictions)
weighted_nll = tf.multiply(nll, q_values)
loss = tf.reduce_mean(weighted_nll)
gradients = loss.gradients(loss, theta)
```

Prob. of "correct" action

Cross-Entropy Loss:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Inverse Optimal Control / Inverse RL

Given:

- State and action space
- Roll-outs from π^*
- Dynamics Model (sometimes)

Goal:

- Recover reward function
- Use reward function to derive $\pi \approx \pi^*$

Challenges:

- Problem is underspecified
- Hard to evaluate your learned reward
- Demonstrations might be good, but not optimal

Maximum Entropy IRL
(Ziebart et al. 2008)

Notation:

$$\tau = \{s_1, a_1, ..., s_t, a_t, ..., s_T\}$$

trajectory

$$R_\psi(\tau) = \sum_t r_\psi(s_t, a_t)$$

learned reward

$$\mathcal{D} : \{\tau_i\} \sim \pi^*$$

expert demonstrations

$$p(\tau) = \frac{1}{\boxed{Z}} \exp(R_\psi(\tau))$$

Probability is exponential in the reward (good trajectories are more likely)

$$\max_\psi \sum_{\tau \in D} \log p_{r_\psi}(\tau)$$

Maximize likelihood of a given reward function parameterization

$$\boxed{Z = \int \exp\left(R_\psi(\tau)\right) d\tau}$$

Integral over all possible trajectories ☹

MaxEnt IRL

1. Initialize ψ (reward function params), gather demonstrations D
2. Solve for optimal policy $\pi(a|s)$ with reward r_ψ
3. Solve for state visitation frequencies $p(s|\psi)$
4. Compute gradient: $\nabla_\psi L = -\frac{1}{|D|}\sum_{\tau_d \in D} \frac{dr_\psi}{d\psi}(\tau_d) - \sum_s p(s|\psi)\frac{dr_\psi}{d\psi}(s)$
5. Update ψ with one gradient step using $\nabla_\psi L$
6. GOTO 2

Must solve the whole MDP in the inner loop of finding the reward function!

Making IRL work for complex problems

Must handle:

(1) Unknown dynamics
(2) Solving MDP in an inner loop

$$p(\tau) = \frac{1}{Z} \exp(R_\psi(\tau))$$

$$\max_\psi \sum_{\tau \in D} \log p_{r_\psi}(\tau)$$

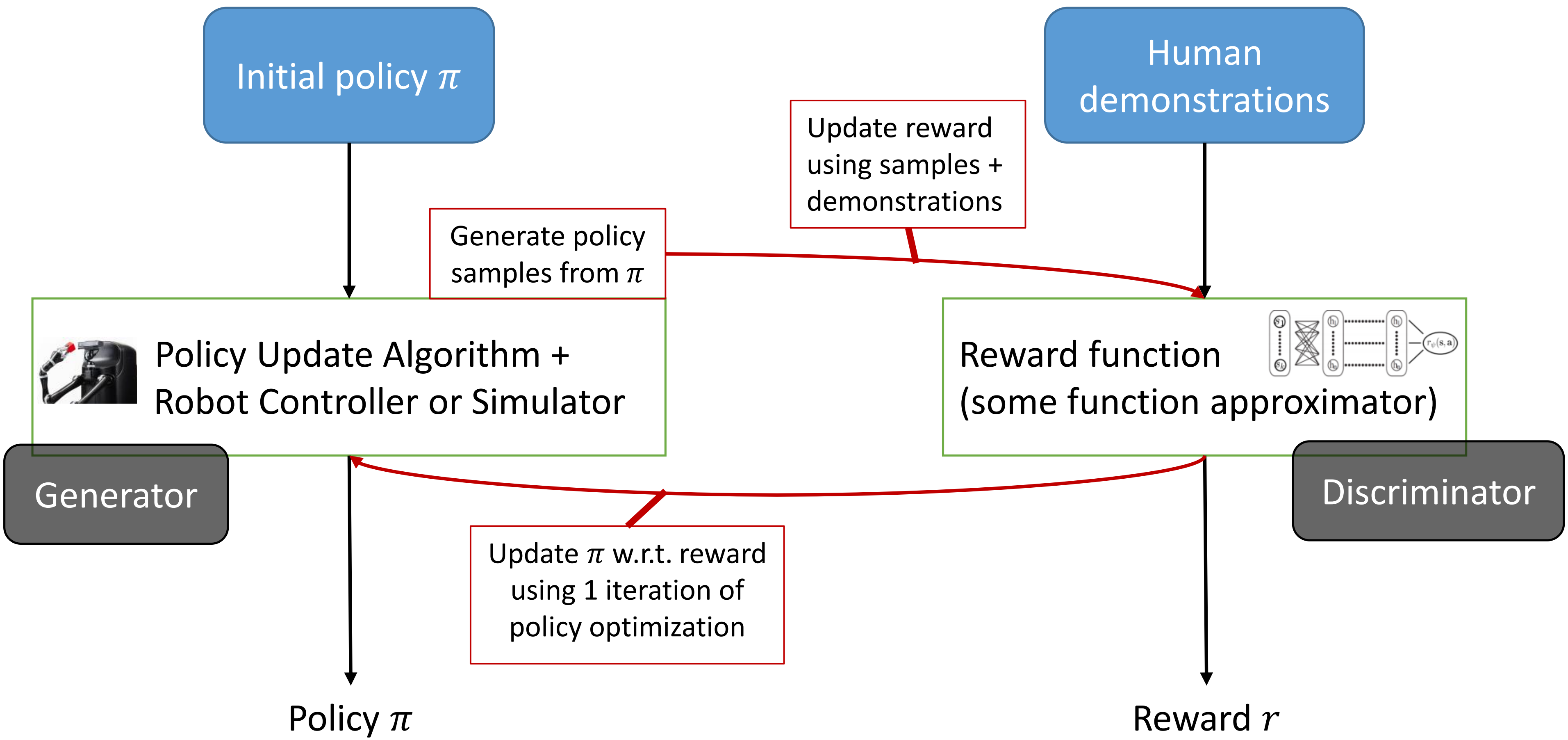
$$\boxed{Z = \int \exp\left(R_\psi(\tau)\right) d\tau}$$

Adaptively sample increasingly close to ψ by constructing a policy to do it for us

Sample from what?
Can't sample from policies near ψ because we're solving for ψ!

Can sample to approximate Z!

Guided Cost Learning (Generative Adversarial Imitation)



Guided Cost Learning: Results

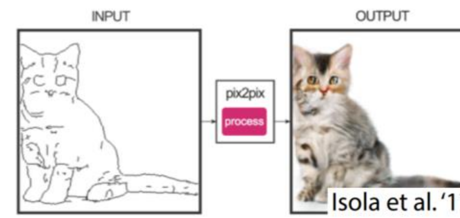
Brief Aside: Generative Adversarial Networks

Inverse RL <=> GANs

Similar to inverse RL, **GANs** learn an objective for generative modeling.

Zhu et al.'17

Arjovsky et al.'17



Isola et al.'17

	trajectory τ	$\longleftrightarrow$	**sample x**	
Inverse RL	**policy** $\pi \sim q(\tau)$	$\longleftrightarrow$	**generator** G	GANs
	reward r	$\longrightarrow$	**discriminator** D	

IRL Recap

Goal: Infer reward function underlying expert demonstrations

Evaluating the partition function (Z):

- Initial approaches solve the MDP in the inner loop of IRL (or assume known dynamics).
- Can estimate Z using sampling!

Connection to Generative Adversarial Networks:

- Sampling-based MaxEnt IRL is a GAN with a special form of discriminator, using RL to optimize the generator.

Further Reading

<u>**Classic Papers**</u>

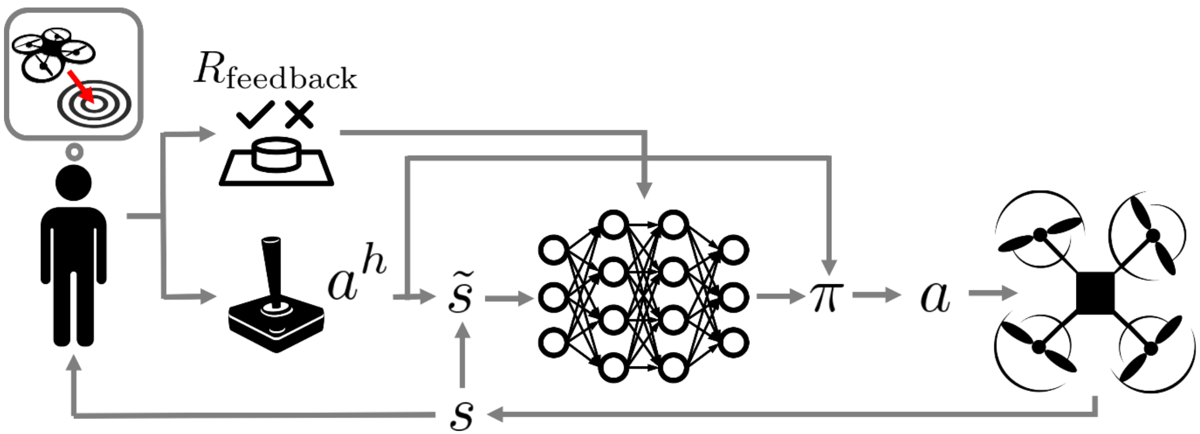
- **Abbeel & Ng ICML '04.** *Apprenticeship Learning via Inverse Reinforcement Learning.* Good introduction to inverse reinforcement learning
- **Ziebart et al. AAAI '08**. *Maximum Entropy Inverse Reinforcement Learning.* Introduction of probabilistic method for inverse reinforcement learning

<u>**Modern Papers**</u>

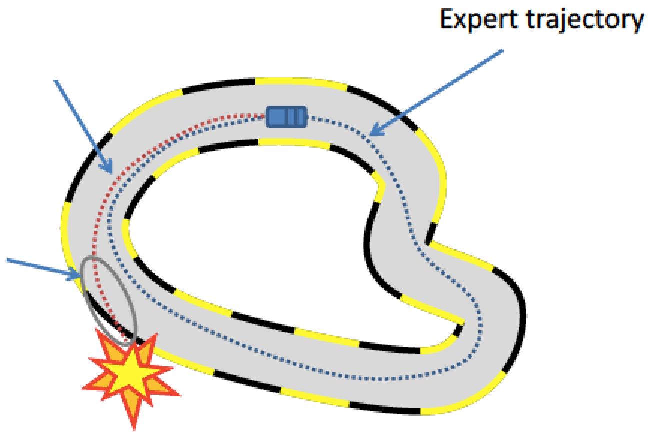
- **Wulfmeier et al. arXiv '16**. *Deep Maximum Entropy Inverse Reinforcement Learning.* MaxEnt IRL using deep reward functions
- **Finn et al. ICML '16**. *Guided Cost Learning*. Sampling-based method for MaxEnt IRL that handles unknown dynamics and deep reward functions
- **Ho & Ermon NIPS '16**. *Generative Adversarial Imitation Learning*. IRL method building on Abbeel & Ng '04 using generative adversarial networks

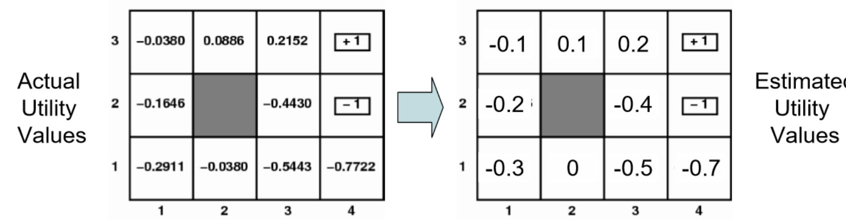
Papers for Today:

Shared Autonomy via Deep Reinforcement Learning
Pro: Chandan Naik
Con: Ian Loefgren



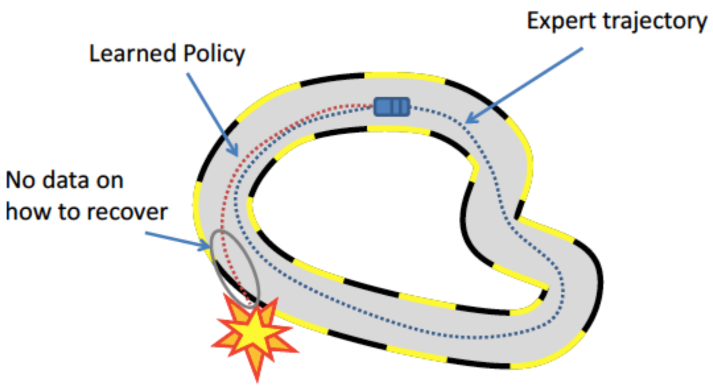
A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning
Pro: Nishank Sharma
Con: Ashwin Vasan





But how can we use this to play Mario Kart?

DAgger (Dataset Aggregation)



- Iterative algorithm
- Trains a stationary deterministic policy
- No regret algorithm in an online learning setting

[under reasonable assumptions, it] “must find a policy with good performance under the distribution of observations it induces in such sequential settings”

“A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”. Ross, Gordon, & Bagnell, 2011

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.

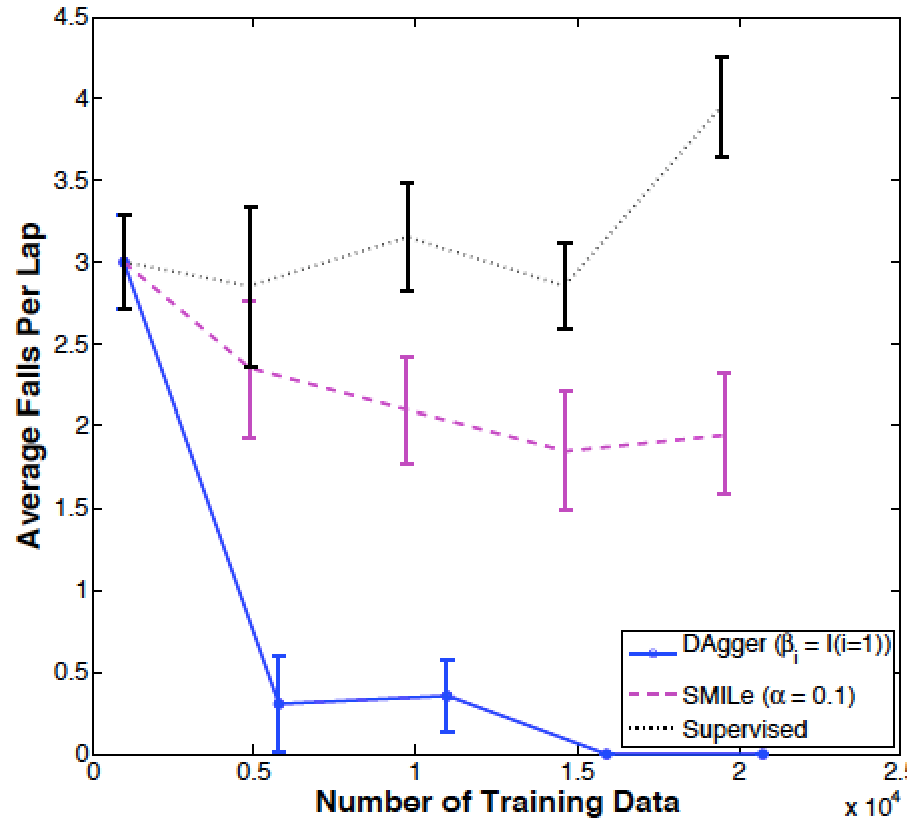
Algorithm 3.1: DAGGER Algorithm.

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 $\mathcal{D}$.

Insight:

1) Combine learned policy with novel human demonstrations
2) Train over all of human demos
3) Learn about areas of the state space not initially reached

Super Tux Kart



Designing Your Evaluation

Evaluation Design:

What are your hypotheses about your system?

How will you test them?

What are you trying to prove with this work?

Experiment Design:

Do you need human subjects?

Are your conditions likely to test your hypotheses?

Within-subjects or between-subjects?

Protocol design:

Someone not on your project should be able to run your experiment with this script!