Imitation Learning from Observations

Artem Tsypin

Samsung AI Center
Moscow

May 14, 2021

Contents

Section

MDP

The RL setting assumes an underlying MDP which is defined as a 5-tuple:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle .$$

Where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $P = p(s' \mid s, a)$ is the transition function of the environment.
The RL task is to maximize discounted cumulative reward:

$$\mathbb{E}_{\tau \sim \pi} \sum_{t \geq 0} \gamma^t r_t \to \max_{\pi}$$

Reward in RL tasks

In some environments such as video games reward is very natural.

Reward in RL tasks

But there is a huge number of environments where coming up with a meaningful reward is a tedious task. For example, reward in humanoid is heavily engineered to encourage desired behavior.

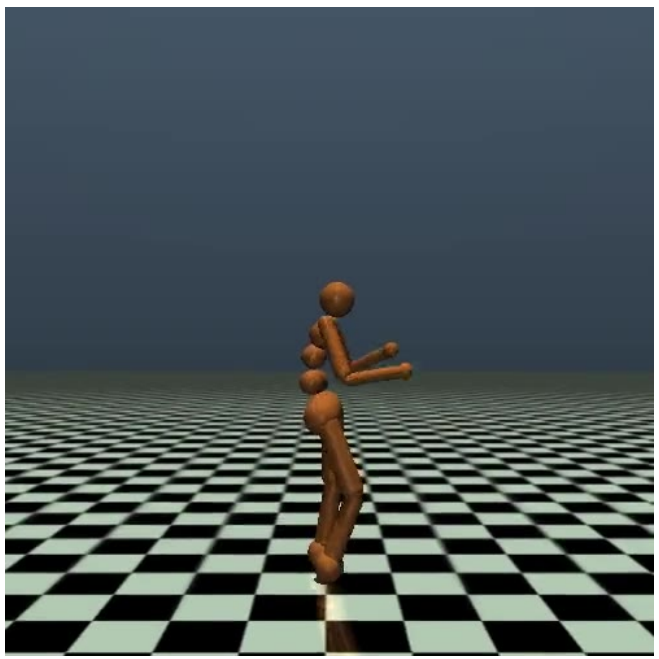


Figure: OpenAI Gym Humanoid-v3

$$r(s,a) = w_0 v(s) - w_1 ||a||_2^2 + w_2[|\theta_{body}| < \epsilon] + w_3[h_{body} > h].$$

Imitation Learning

Reinforcement Learning:

- **Given** Environment and reward function.
- **Find** Policy which maximizes cumulative reward.

Imitation Learning:

- **Given** Environment and examples of expert trajectories.
- **Find** Policy which imitates the expert.

Expert Trajectories

In IL it is assumed that a dataset of expert trajectories is given.

$$D = \{T_1, \ldots, T_n\}.$$

Imitation Learning. $T_i = (s_0, a_0, \ldots, a_{N_i-1}, s_{N_i})$. Requires expert's actions to be recorded.

Imitation Learning from Observations. $T_i = (s_0, \ldots, s_{N_i})$. A harder task but lots of data is available (Youtube videos, human demonstrations, etc.)

Section

Imitation Learning from Observations

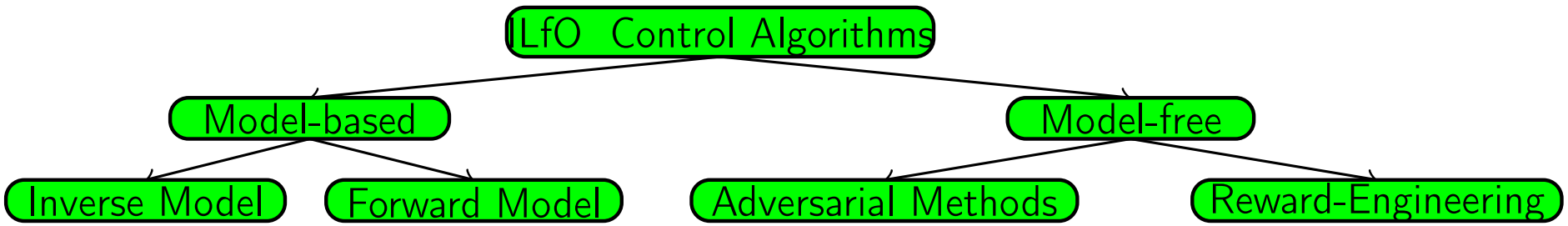


Figure: Classification of Imitation Learning from Observation algorithms.

Paper [1].

[1] Recent Advances in Imitation Learning from Observation https://arxiv.org/pdf/1905.13566.pdf.

Imitation Learning from Observations

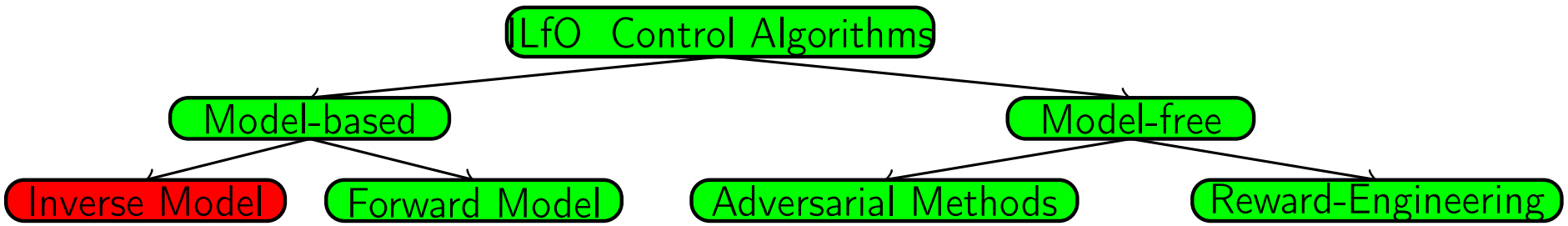


Figure: Behavior Cloning from observations with Inverse Dynamics Model.

Paper [2].

[2] Behavioral Cloning from Observation https://arxiv.org/pdf/1805.01954.pdf.

Behavior Cloning

$T_i = (s_0, a_0, \ldots, a_{N_i-1}, s_{N_i})$. Then the policy is trained in a supervised fashion by maximizing likelihood:

$$\sum_{\tau \in D} \sum_{(s,a) \in \tau} \log \pi(a \mid s) \to \max_{\pi}$$

Pros: Easy to train, requires no interaction with the environment.

Cons: Suffers from covariate shift.

Behavior Cloning from Observations

$T_i = (s_0, \ldots, s_{N_i})$. How to infer actions?

Train an Inverse Dynamics Model $p_\theta(a_i \mid s_i, s_{i+1})$.

Inverse dynamics model is parametrized by θ and trained to maximize

$$\sum_{\tau \in D} \sum_{(s_i, a_i, s_{i+1}) \in \tau} \log p_\theta(a_i \mid s_i, s_{i+1}) \to \max_\theta$$

Note that Inverse dynamics model can be trained iteratively.

Behavior Cloning from Observations

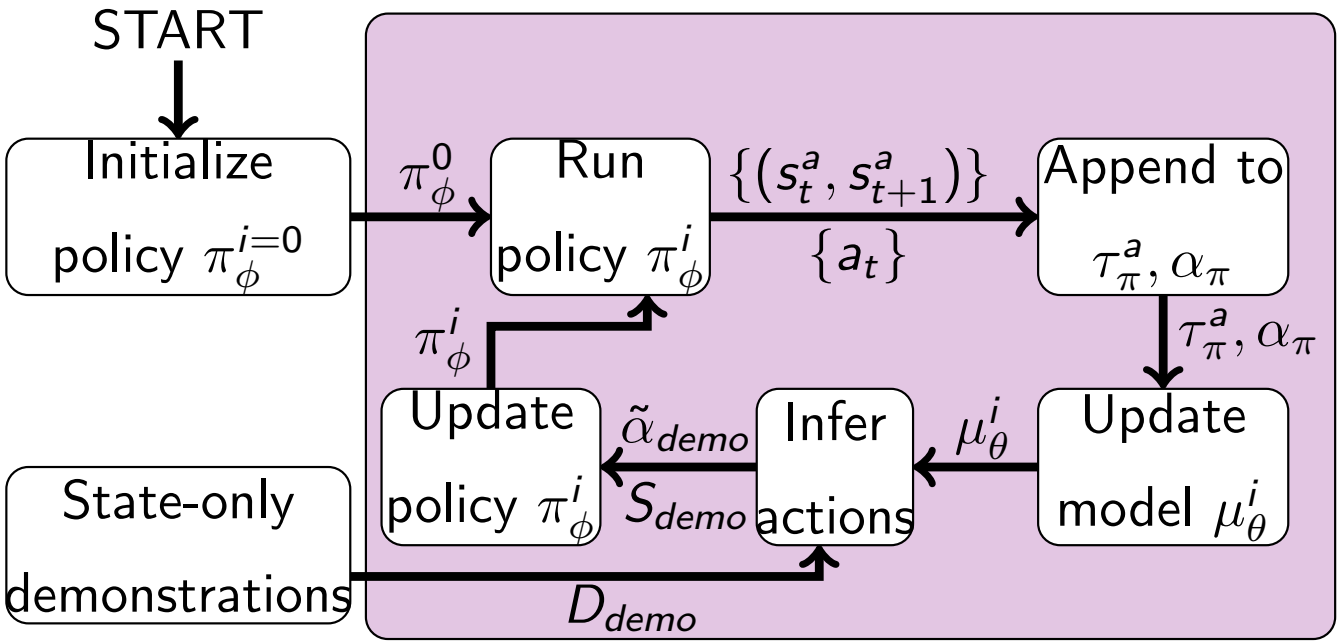
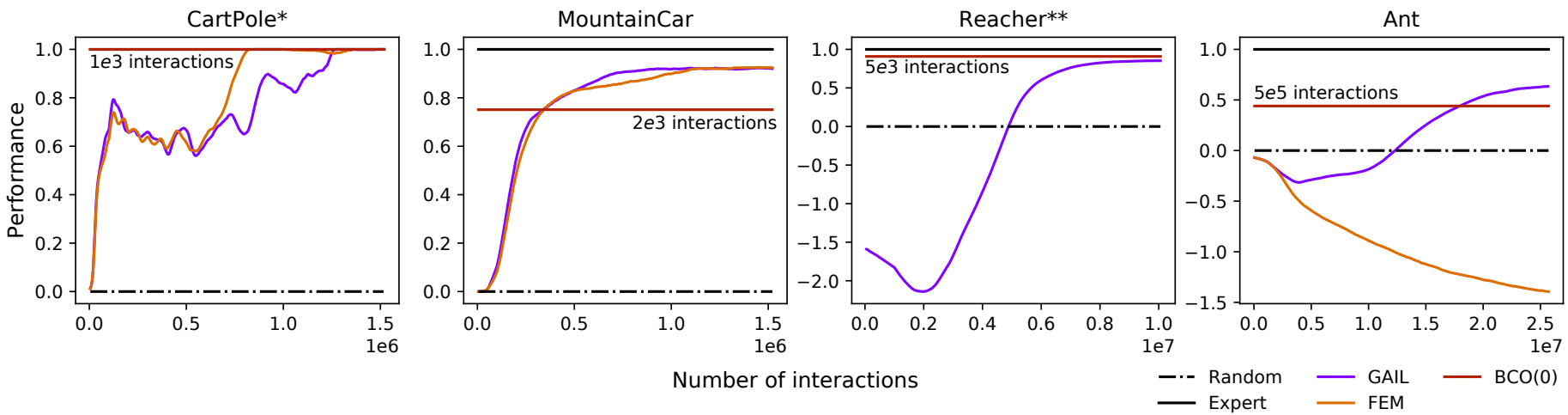


Figure: Inverse dynamics model is trained on expert dataset. Then it is used to infer actions that were taken by expert. Then normal behavior cloning is used to train policy.

Behavior Cloning from Observations

The results are reported for version of the method with no iterative updates. The main advantage of the method is sample efficiency in terms of interactions with the environment.



Imitation Learning from Observations

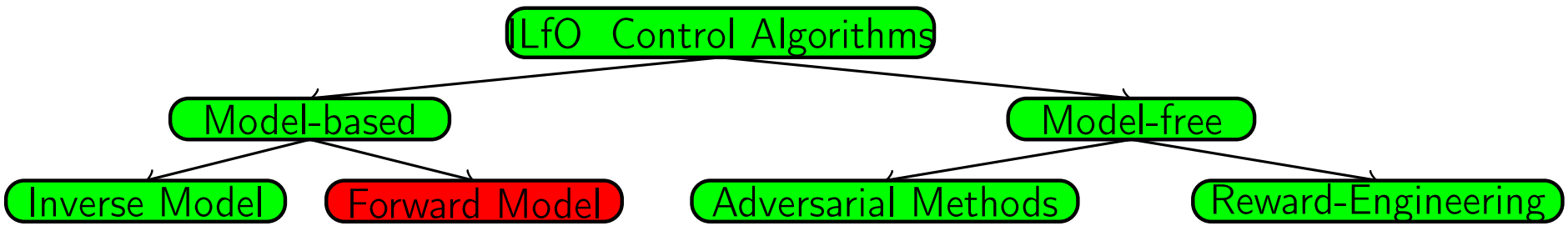


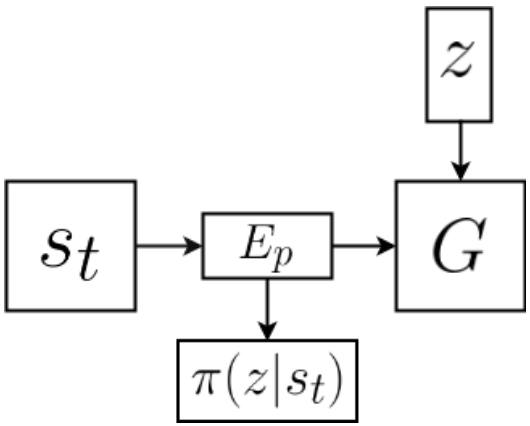
Figure: Imitation learning with forward dynamics model.

Paper [3].

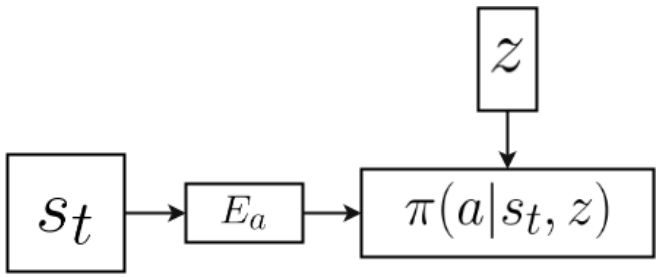
[3] Imitating Latent Policies from Observation
https://arxiv.org/pdf/1805.07914.pdf.

ILPO

The idea is to infer latent actions using forward dynamics model $p_\theta(s_{t+1} \mid s_t, z_t)$ and then remap them to real actions by interacting with the environment.



(a) Latent Policy Network



(b) Action Remapping Network

Figure: The latent policy network learns a latent policy, $\pi(z|s)$, and a forward dynamics model, G. The action remapping network learns $\pi(a|s_t, z)$ to align the latent actions z with ground-truth actions a. We train embeddings, E_a and E_p, concurrently with each network.

ILPO

function ILPO($s_0^*, s_1^*, \ldots, s_N^*$)
    Step 1: Learning latent policies
    for $k \leftarrow 0 \ldots \#Epochs$ **do**
        for $i \leftarrow 0 \ldots N-1$ **do** ▷ *(Omitting batching for clarity)*
            Train latent dynamics parameters
$\theta \leftarrow \theta - \nabla_\theta \min_z \| G_\theta(E_{p\theta}(s_i^*), z) - s_{i+1}^* \|_2^2$
            Train latent policy parameters
$\omega \leftarrow \omega - \nabla_\omega \| \sum_z \pi_\omega(z|s_i^*) G_\theta(E_{p\theta}(s_i^*), z) - s_{i+1}^*) \|_2^2$
        end for
    end for
end function

ILPO

function ILPO($s_0^*, s_1^*, \ldots, s_N^*$)

Step 2: Action remapping. Observe state s_0

for $t \leftarrow 0 \ldots \#Interactions$ **do**

Choose latent action

$z_t \leftarrow \arg\max_z \pi_\omega(z|E_{a\xi}(s_t))$

Take ϵ-greedy action

$a_t \leftarrow \arg\max_a \pi_\xi(a|z_t, E_{a\xi}(s_t))$

Observe state s_{t+1}

Infer closest latent action

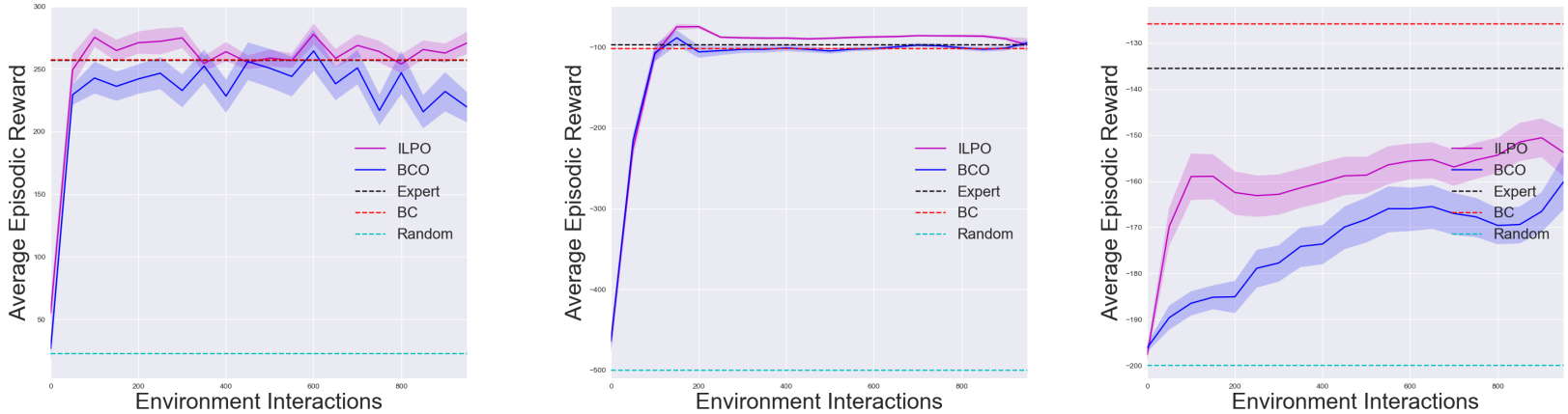
$z_t = \arg\min_z \|E_{p\theta}(s_{t+1}) - E_{p\theta}(G_\theta(E_{p\theta}(s_t), z))\|_2$

Train action remapping parameters

$\xi \leftarrow \xi + \nabla_\xi \log \frac{\pi_\xi(a_t|z_t, E_{a\xi}(s_t))}{\sum_a \pi_\xi(a|z_t, E_{a\xi}(s_t))}$

end for

ILPO experiments



(a) Cartpole (b) Acrobot (c) Mountain car

Figure: Classic control imitation learning results.

Pros: Sample Efficient.

Cons: Requires a generative model of the environment. Fishy expert data collection. Only dicrete actions.

Imitation Learning from Observations

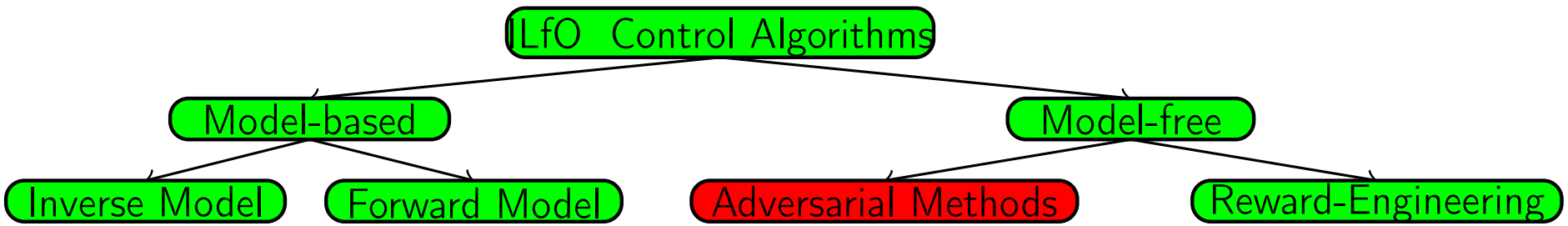


Figure: Imitation Learning through adversarial learning.

Paper [4].

[4] Generative Adversarial Imitation from Observation https://arxiv.org/pdf/1807.06158.pdf.

GANs in IL

It's a well-known fact that GAN models match data distribution p_{data} to generator distribution p_{gen} by minimizing a divergence. Can we use it for Imitation Learning?
In case of IL the policy is generator which outputs state-action pairs (s, a) and the discriminator tries to distinguish pairs from expert trajectories $(s, a) \sim \rho_{\pi_E}(s, a)$ from pairs generated by the policy $(s, a) \sim \rho_{\pi}(s, a)$.

GAIL

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0

2: **for** $i = 0, 1, 2, \ldots$ **do**

3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$

4: Update the discriminator with the loss

$$-\left(\hat{\mathbb{E}}_{\tau_i}[\log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\log(1 - D_w(s,a))]\right)$$

5: Take a policy step using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$.

$$\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] + \lambda \nabla_\theta H(\pi_\theta),$$

$$Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[-\log(D_{w_{i+1}}(s,a)) \,|\, s_0 = \bar{s}, a_0 = \bar{a}]$$

6: **end for**

Inverse RL

One way to formalize Imitation Learning is through Inverse RL. Cost function instead of reward function is considered:

$$c(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}.$$

First a cost function is learned in a way such it is minimal for the expert trajectories and maximal for non-expert trajectories. Such cost function could be obtained by solving Maximum Entropy Inverse RL:

$$\begin{aligned} IRL_\psi(\pi_E) = &\arg\max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} -\psi(c) + \left(\min_{\pi \in \prod} -\lambda_H H(\pi) + \right. \\ &\left. \mathbb{E}_\pi[c(s, a)]\right) - \mathbb{E}_{\pi_E}[c(s, a)] \, . \end{aligned} \tag{1}$$

Inverse RL

The second step is to plug learned cost function into any RL algorithm to obtain the policy:

$$RL(c) = \underset{\pi \in \prod}{\arg\min} - \lambda_H H(\pi) + \mathbb{E}_\pi[c(s,a)] \ . \qquad (2)$$

GAIL

GAIL solves the Inverse RL task (1), (2) by matching occupancy measure of the agent to that of the expert.

Definition 1

The occupancy measure $\rho_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as

$$\rho_\pi(s, a) = \pi(a \mid s) \sum_{t=0}^{\infty} \gamma^t P(s = s_t \mid \pi)$$

.

Theorem 2

$$RL \circ IRL_\psi(\pi_E) = \arg\min_{\pi \in \Pi} -H(\pi) + \psi^*(\rho_\pi - \rho_{\pi_E}).$$

GAIL

We can now use following convex regularizer:

$$\psi_{GA}(c) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[g(c(s,a))] & \text{if } c < 0 \\ +\infty & \text{otherwise, } \textit{where} \end{cases} \tag{3}$$

$$g(x) = \begin{cases} -x - \log(1 - e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases} \tag{4}$$

GAIL

Such regularizer has a convex conjugate:

$$\psi^*_{GA}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S}\times\mathcal{A}}} \sum_{s,a} \rho_\pi(s,a)\log(D(s,a)) + \rho_{\pi_E}(s,a)\log(1 - D(s,a)) \ , \tag{5}$$

Then according to theorem 2:

$$RL \circ IRL_\psi(\pi_E) = \min_{\pi \in \prod} \max_{D \in (0,1)^{\mathcal{S}\times\mathcal{A}}} - \lambda_H H(\pi) + \\ + \mathbb{E}_\pi[\log(D(s,a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] \ . \tag{6}$$

GAIfO

GAIL naturally generalizes to the case of observation-only trajectories.
In this case generator is a policy which generates state-action pairs (s, s') and the discriminator tries to distinguish pairs from expert trajectories $(s, s) \sim \rho^s_{\pi_E}(s, s')$ from pairs generated by the policy $(s, s') \sim \rho^s_{\pi}(s, s')$.

GAIfO

Analogous reasoning could be used to prove that:

$$RLfO \circ IRLfO_{\psi}(\pi_E) = \min_{\pi \in \Pi} \psi^*_{GA}(\rho^s_{\pi} - \rho^s_{\pi_E}) = \tag{7}$$

$$\min_{\pi \in \Pi} \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{S}}} \mathbb{E}_{\pi}[\log(D(s, s'))] + \mathbb{E}_{\pi_E}[\log(1 - D(s, s'))]. \tag{8}$$

Note that the entropy term is ommited.

GAIfO

1: **Input:** state-only expert trajectories $\tau_E = \{(s, s')\}$, initial policy and discriminator parameters θ_0, w_0
2: **while** Policy Improves **do**
3: Execute π_ϕ and store the resulting state transitions $\tau = \{(s, s')\}$
4: Update D_θ using loss

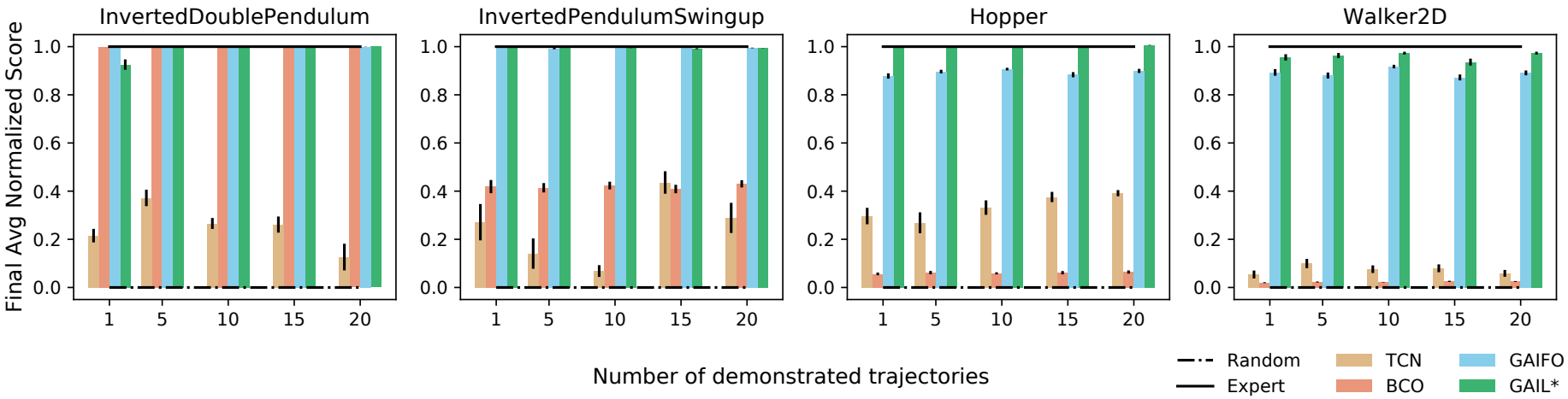
$$-\Big(\mathbb{E}_\tau[\log(D_\theta(s, s'))] + \mathbb{E}_{\tau_E}[\log(1 - D_\theta(s, s'))]\Big)$$

5: Update π_ϕ by performing *TRPO* updates with reward function

$$-\Big(\mathbb{E}_\tau[\log(D_\theta(s, s'))]\Big)$$

6: **end while**

GAIfO Experiments



Pros: is almost as good as GAIL.

Cons: *very* sample inefficient. Is an on-policy algorithm.

Imitation Learning from Observations

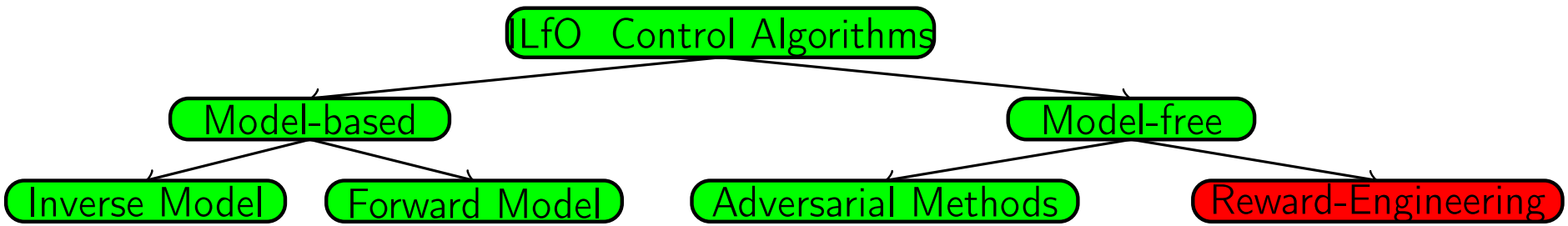


Figure: Imitation learning with reward engineering.

Paper [5].

[5] Internal Model from Observations for Reward Shaping https://arxiv.org/pdf/1806.01267.pdf.

Trajectory Likelihood

Let's take a look at the total reward for the episode: $R = \sum_{t=0}^{T} r_t$. If $r_t = \log(p_E(s_{t+1} \mid s_t))$ then R can be viewed as likelihood of the trajectory:

$$R = \sum_{t=0}^{T} \log(p_E(s_{t+1} \mid s_t)) = \log p_E(\tau).$$

What are the ways to train $p_E(s_{t+1} \mid s_t)$? A straightforward approach is to assume Gaussian distribution on states and train a generative model.

Internal Model for Reward Learning

procedure Training Demonstrations
  Given *trajectories* τ from expert agent
  for $\boldsymbol{s^i_t}, \boldsymbol{s^i_{t+1}} \in \tau$ **do**
    $\boldsymbol{\theta}^* \leftarrow \arg\min_{\boldsymbol{\theta}} \left[-\sum_{i,t} \log p(\boldsymbol{s^i_{t+1}} | \boldsymbol{s^i_t}; \boldsymbol{\theta}) \right]$
  end for
end procedure
procedure Reinforcement Learning
  for $t = 1, 2, 3, \ldots$ **do**
    Observe *state* $\boldsymbol{s_t}$
    Execute *action* $\boldsymbol{a_t}$, and observe *state* $\boldsymbol{s_{t+1}}$
    $r_t \leftarrow -\psi\Big(\|\boldsymbol{s_{t+1}} - \boldsymbol{\theta}(\boldsymbol{s_t})\| \Big)$
    Update network using $(\boldsymbol{s_t}, \boldsymbol{a_t}, r_t, \boldsymbol{s_{t+1}})$

IMRL Experiments.

Bizarre experiments with a bunch of hand-crafted reward functions and no comparison to other methods in terms of sample efficiency or performance.

$$r_t = -\|\boldsymbol{p_{ee}} - \boldsymbol{p_{tgt}}\|_2 + r_t^{env} \tag{9}$$

$$r_t = -100 \tanh(\|\boldsymbol{p_{ee}} - \boldsymbol{p_{tgt}}\|_2) + r_t^{env} \tag{10}$$

$$r_t = -10 \tanh(\|\boldsymbol{s_{t+1}} - \boldsymbol{\theta_{+a}}(\boldsymbol{s_t}, \boldsymbol{a_t})\|_2) + r_t^{env} \tag{11}$$

$$r_t = -\tanh(\|\boldsymbol{s_{t+1}} - \boldsymbol{\theta_g}(\boldsymbol{s_{t+1}})\|_2) + r_t^{env} \tag{12}$$

$$r_t = -10 \tanh(\|\boldsymbol{s_{t+1}} - \boldsymbol{\theta}(\boldsymbol{s_t})\|_2) + r_t^{env} \tag{13}$$

Pros: After initial training such reward could be paired with any RL algorithm. **Cons**: Requires learning generative model and handcrafted functions.

Section

PMI Imitation Learning

Can we use the reward engineering approach but avoid training a generative model and instead train a discriminative model $p(s_{t+1} \mid s_t)$ which can later be used to obtain reward for the agent?

If so then the trained reward would be compatible with any RL algorithm and also could be used in offline RL.

InfoNCE Loss

Intuitively such reward can tell us how likely is state s_{t+1} given state s_t from expert's point of view.
Such a discriminative model $f(s_{t+1}, s_t)$ can be trained with InfoNCE loss:

$$L = -\mathbb{E}_X \log \frac{f(s_{t+1}, s_t)}{\sum_{s_j \in X} f(s_j, s_t)} \to \min_f, \tag{14}$$

where $X = (s_0, \ldots, s_M)$ consists of a positive example from $p(s_{t+1} \mid s_t)$ and $M - 1$ negative from $p(s_{t+1})$.

Optimal solution

The optimal f: $f^*(s, s_{t+1}) \propto \frac{p(s_{t+1}|s_t)}{p(s_{t+1})}$.

Note that the pointwise mutual information (pmi) is defined as $\log \frac{p(s_{t+1}|s_t)}{p(s_{t+1})}$

Reward function

RLfO (2) can be reformulated in terms of reward function:

$$RLfO(r) = \arg\max_{\pi \in \prod} \lambda_H H(\pi) + \mathbb{E}_\pi[r(s, s')] \ . \tag{15}$$

The following reward functions are considered :

$$r(s_t, s_{t+1}) = \log f_\theta^*(s_{t+1}, s_t) = \log \frac{p(s_{t+1} \mid s_t)}{p(s_{t+1})} + c \tag{16}$$

$$\tilde{r}(s_t, s_{t+1}) = -\log \frac{f_\theta^*(s_{t+1}, s_t)}{\sum_{s_j \in X^{IL}} f_\theta^*(s_j, s_t)} \tag{17}$$

Intuitively $\tilde{r}$ says how likely is the expert to transition to state s_{t+1} compared to states $s_j \in X^{IL}$.

SoftMax reward

In out experiments $f_\theta = \exp(\phi_1(s_{t+1})^T \phi_2(s_t))$. Then $\tilde{r}$ becomes a softmax. Temperature parameter τ is added to control the resulting distribution:

$$\tilde{r}_\tau(s_t, s_{t+1}) = -\log \frac{\exp(\frac{\phi_1(s_{t+1})^T \phi_2(s_t)}{\tau})}{\sum_{s_j \in X^{IL}} \exp(\frac{\phi_1(s_j)^T \phi_2(s_t)}{\tau})}. \quad (18)$$

Consequently r becomes a cosine:

$$r(s_t, s_{t+1}) = \log f_\theta^*(s_{t+1}, s_t) = \phi_1(s_{t+1})^T \phi_2(s_t) \quad (19)$$

DSSM training

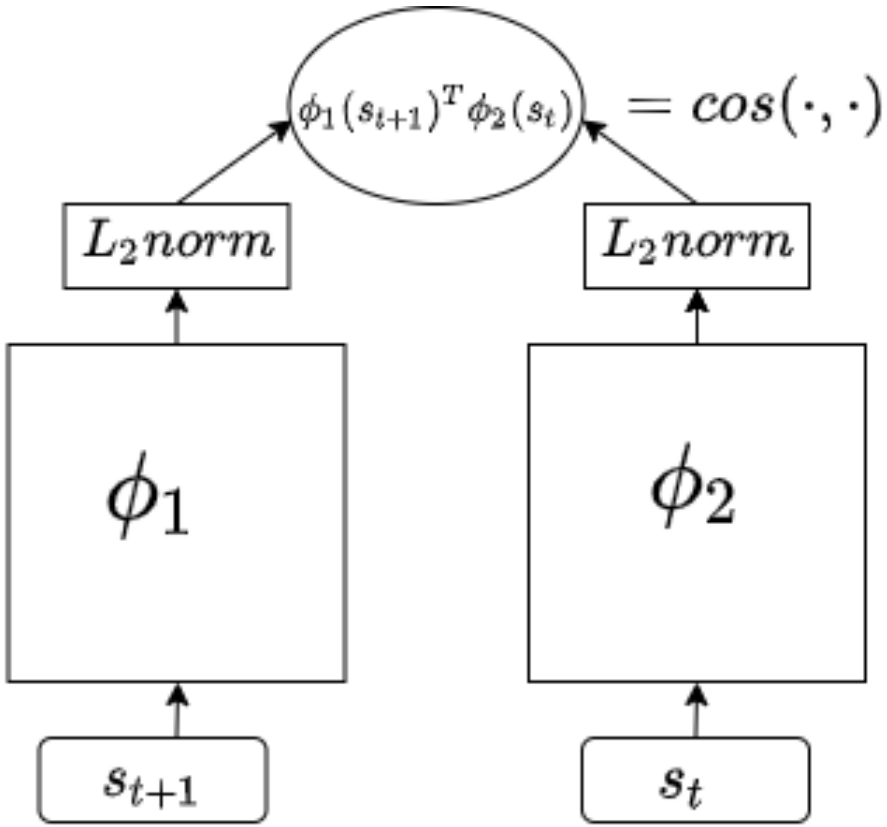


Figure: DSSM architecture.

DSSM training

f_θ is trained to distinguish true next state from $p_E(s_{t+1} \mid s_t)$ from possible next states from $p_E(s_{t+1})$. But the expert trajectories consist only from positive examples. How do we evaluate the performance of f_θ?

$$metrics = \frac{1}{n}\sum_{k=1}^{n}\frac{1}{T_k}\sum_{t=1}^{T_k}[\arg\max_{s \in S_t^k} f_\theta(s_t, s) = s_{t+1}], \quad (20)$$

where n is the number of expert trajectories, T_k is the length of k-th trajectory, S_t^k is the set of possible states to where agent can transition from state s_t of trajectory k.

DSSM training

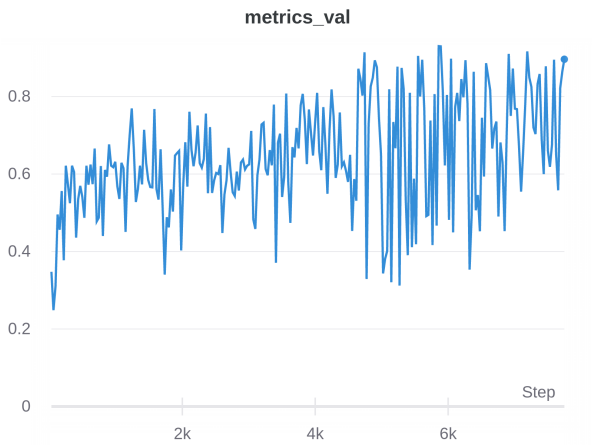


Figure: *Metrics* on val

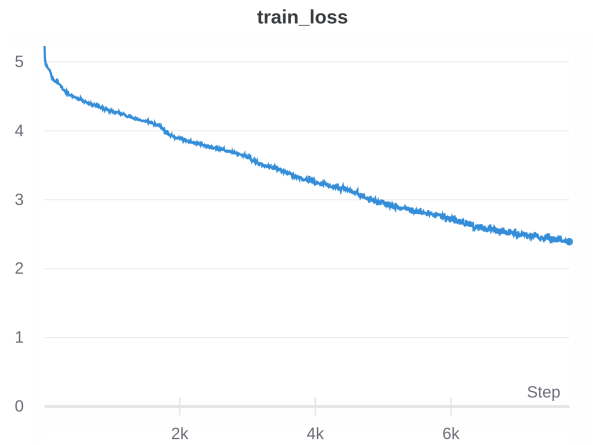


Figure: Train loss

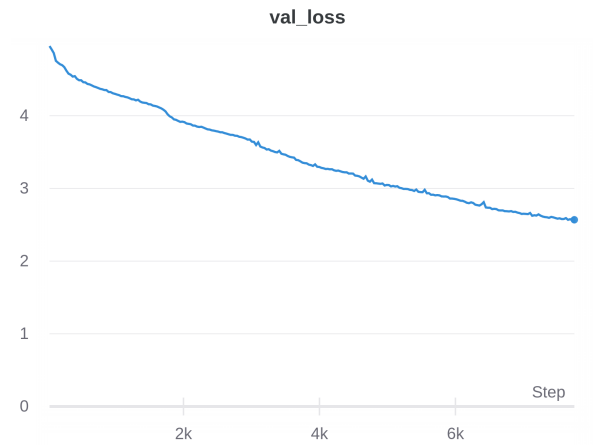


Figure: Validation loss

Figure: Training $f_\theta(s_t, s_{t+1})$ on expert trajectories from MountainCar

Expert training

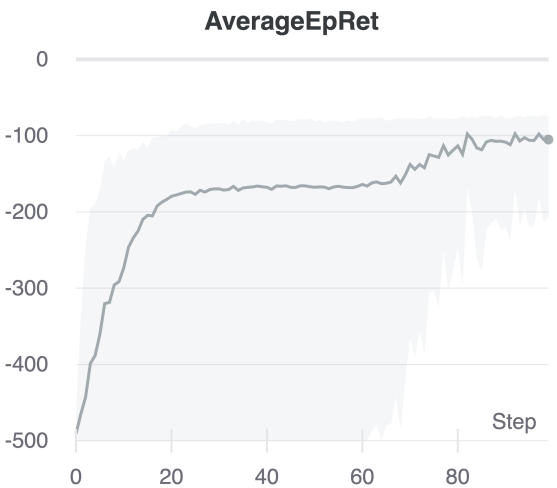


Figure: Acrobot

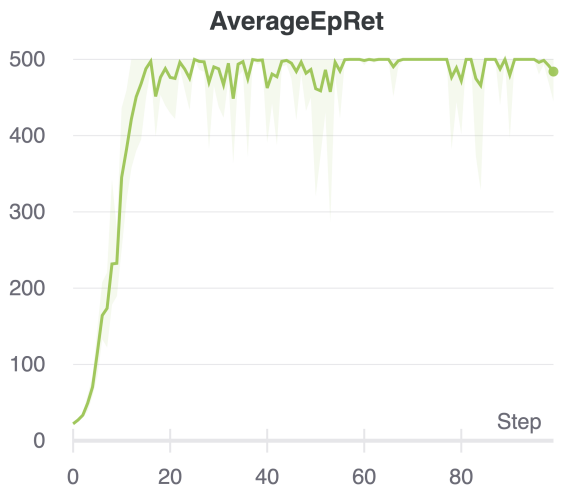


Figure: CartPole

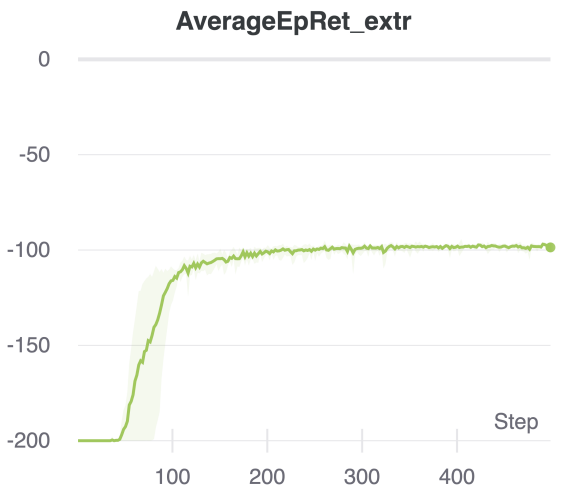


Figure: MountainCar

Figure: Environment reward for experts in different environments.

Comparison of rewards

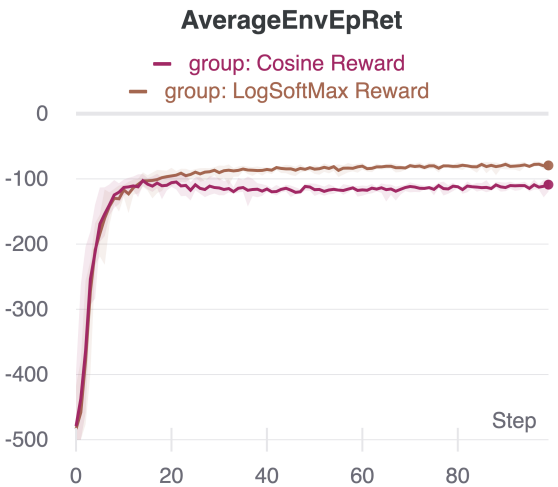


Figure: Acrobot

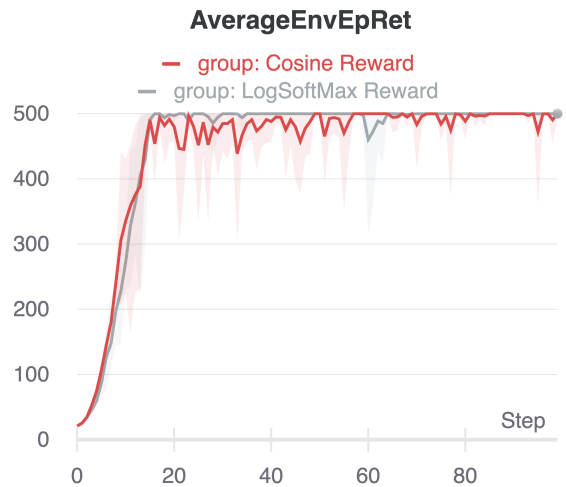


Figure: CartPole

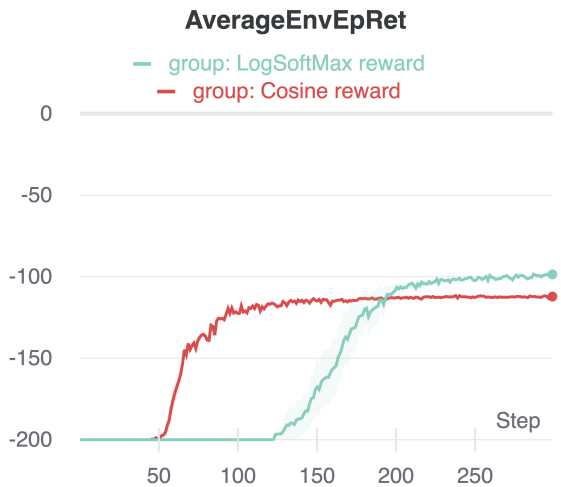


Figure: MountainCar

Figure: Environment reward for agents with different reward functions.

SoftMax reward recap

$$\tilde{r}_\tau(s_t, s_{t+1}) = -\log \frac{\exp\left(\frac{\phi_1(s_{t+1})^T \phi_2(s_t)}{\tau}\right)}{\sum_{s_j \in X^{IL}} \exp\left(\frac{\phi_1(s_j)^T \phi_2(s_t)}{\tau}\right)}. \quad (21)$$

Where should the negatives X^{IL} come from?

How does the temperature parameter influence training?

Negatives

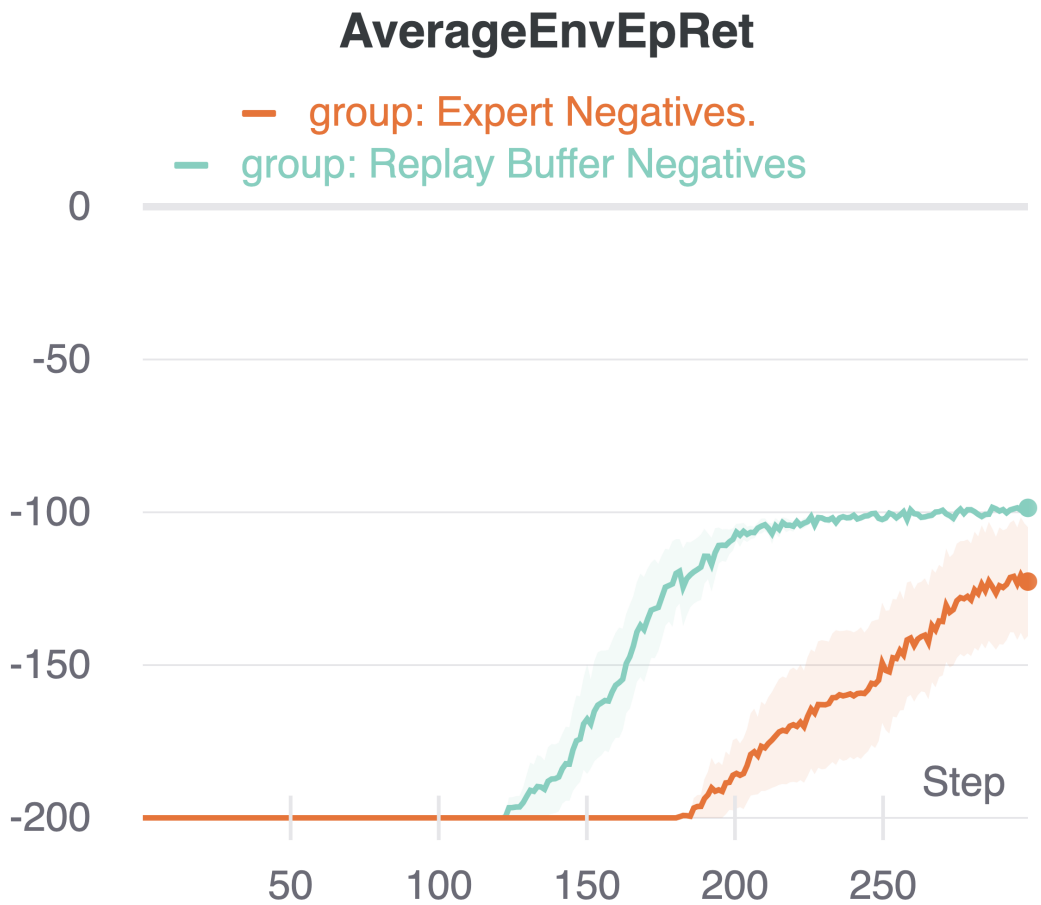


Figure: Environment reward for agents with different negatives X^{IL}.

Temperature

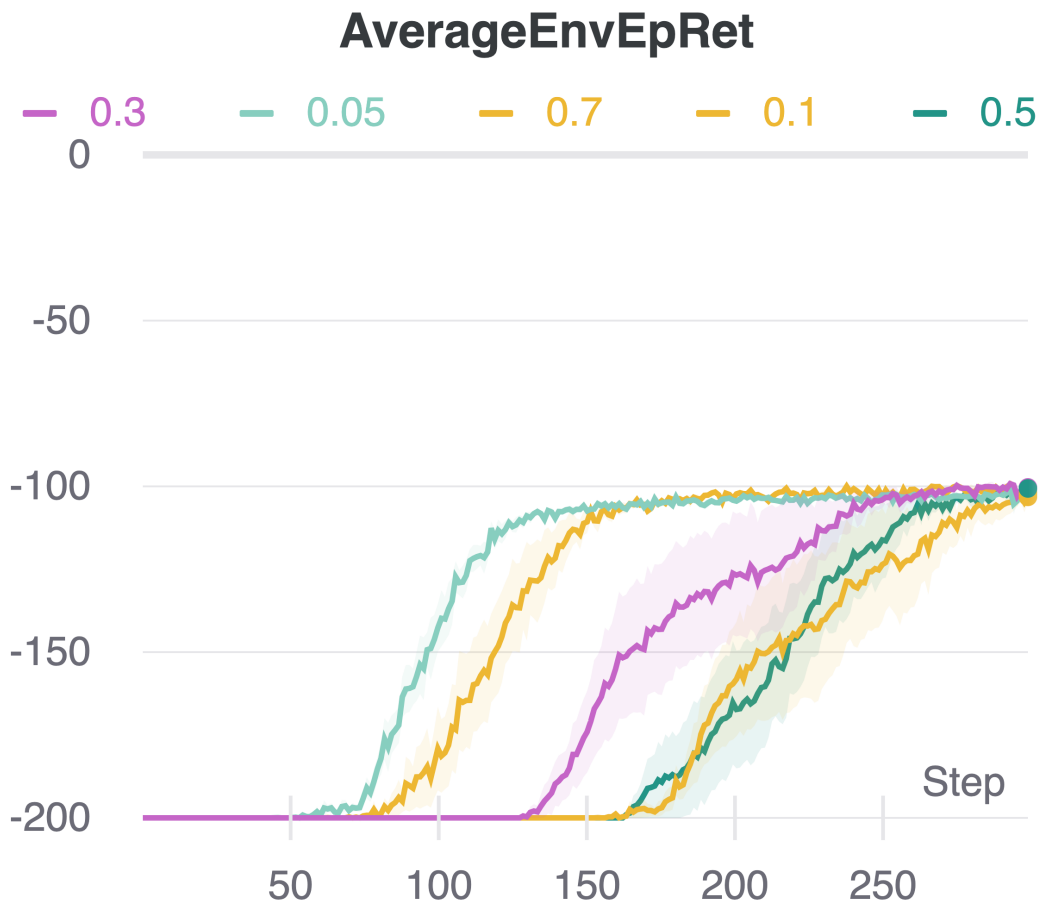


Figure: Environment reward for agents with different temperatures τ.

Entropy Coefficient

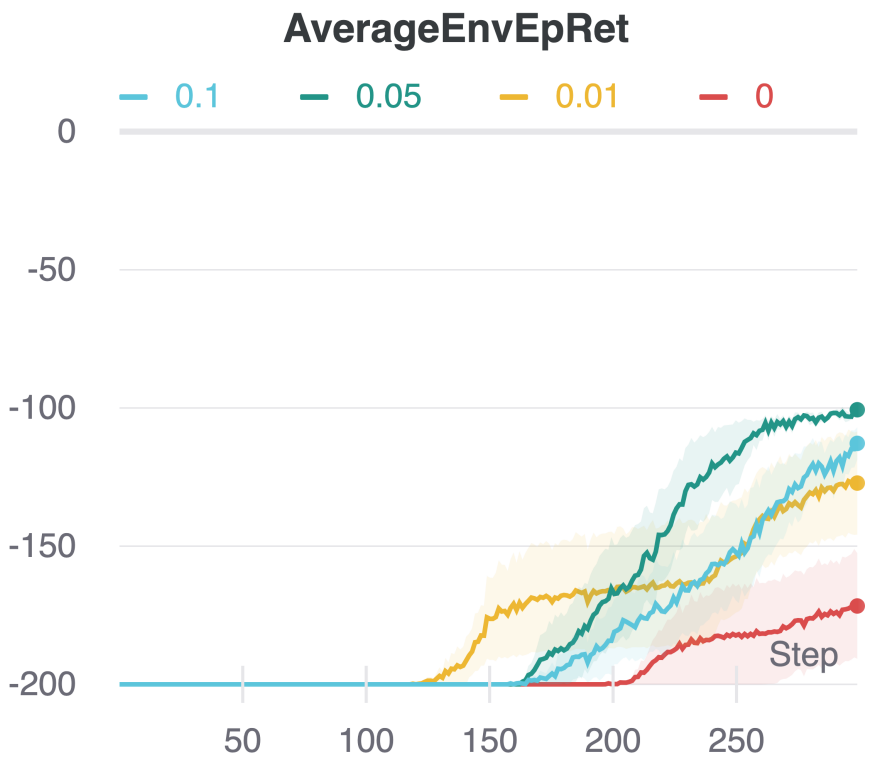


Figure: Environment reward for agents with different entropy coefficients λ_H.

$$RLfO(r) = \arg\max_{\pi \in \prod} \lambda_H H(\pi) + \mathbb{E}_\pi[r(s, s')] \ .$$

Results

Expert Performance		
Acrobot	CartPole	MountainCar
-77.42	500	-98.29

Table: Expert performance in different environments

IL agents performance			
Environment	BCO	GAIL	**PMI IL**
CartPole	471.5	**500.0**	**500.0**
Acrobot	-95.1	**-78.9**	-79.3
MountainCar	-123.1	-99.9	**-98.6**

Table: Comparison with other methods

Questions

Questions?