Robot Learning
Weekly Exercise 7

Marc Toussaint & Wolfgang Hönig
Learning & Intelligent Systems Lab, Intelligent Multi-Robot Coordination Lab, TU Berlin
Marchstr. 23, 10587 Berlin, Germany

Summer 2024

1 Literature: Adversarial Inverse Reinforcement Learning

Here is an advanced paper on inverse RL applied to robotics problems:

J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning, 2018-08-13. URL: http://arxiv.org/abs/1710.11248, arXiv:1710.11248[cs]

The paper was a big step forward in enabling Deep Learning methods for Inverse RL, namely by formulating a loss function similar to Generative Adversarial Networks (GANs) – actually following the original idea formulating InvRL as a discriminative (max margin) problem [3]. A followup paper [4] provides a nicer summary of the history of InvRL ideas and proposes improvement on Adversarial InvRL, but without robotics applications.

The paper webpage https://sites.google.com/view/adversarial-irl provides some videos. Here the questions:

a) Let's start with the experiments in Section 7.2: The setting of the evaluation is *transfer learning*. Be able to explain Table 1: What are the two domains and what kind of transfer is tested? What does "TRPO, ground truth" mean (TRPO is a standard RL method)?

"point mass" is not properly defined. Presumably it is $q, \dot{q} \in \mathbb{R}^2$, and $u \in \mathbb{R}^2$ are direct accelerations; the dynamics a Euler integration.

The quadruped ant is a standard gym environment https://gymnasium.farama.org/main/environments/mujoco/ant/, but the details of the mutilation are not given. Shrink is clear, but what does "disable" mean? Is the dimensionality of the controls reduced? (That would be possible when only transferring the learned reward function.)

Ground Truth: TRPO is trained using the true reward function. The others (GAIL, AIRL) are trained using the InvRL-learned reward function but then evaluated with the true reward function (which is also not clearly stated). TRPO was also used to generate the expert data for inverse RL, which is why it is generally best.

b) In Section 7.3, the setting of evaluation is *imitation learning*. How is that different to the setting of 7.2? How does AIRL compare with GAIL (a pure imitation learning method) and the TRPO expert?

c) The last equation in Sec. 4 (page 4) defines the discriminator $D_\theta(s, a)$. In GANs, a discriminator outputs the probability of whether the input data point is from the "original source" instead of from the learned generative model. What exactly is the meaning of the output of the $D_\theta(s, a)$ defined here?

In logistic regression $p_\theta(y = 1|x) = \frac{\exp\{f_\theta(x)\}}{\exp\{f_\theta(x)\}+\exp\{-f_\theta(x)\}}$ is class $y = 1$ probability when $f_\theta(x)$ the energy of class $y = 1$ and $-f_\theta(x)$ the energy of class $y = 0$. f_θ is then trained to minimize the classification loss under this probabilitic model.

In our case, $f_\theta(s, a)$ is the energy of class "expert-took-this-action", while $\log \pi(a|s)$ is the energy of the class "adversarial-took-this-action". So this is like logistic regression, but the energy of the "other" class is not assumed $-f_\theta$ but $\log \pi(a|s)$, which makes sense as $\pi(a|s)$ is known and $\log \pi(a|s)$ truely the energy. (As is clear from multi-class classification, it is perfectly fine to have separate energy models for each class. The special assumption made in binary classification, that the two energies are f and $-f$ is not mandatory.)

[Note that, as in GANs, Alg. 1 describes an algorithm that also improves the "generative model" (here the learned policy π) whenever the discriminative model was improved.]

d) At first it might be unclear why learning $D_\theta(s,a)$ is related to extracting an underlying reward function. The last equation in Sec 6 (page 6) is quite crucial to understand this – explain roughly why the two neural nets $g_\theta(s)$ and $h_\phi(s)$ in Eq.(4) end up estimating reward and value functions.

First, suddenly the discriminator is extended to be a function of (s,a,s') instead of just (s,a). That's of course "legal" to do, but not well motivated if one understands that discriminator to be a classifier of expert-vs-adversarial-took-this-action, which should only depend on a.

But it is not wrong to make it a function of (s,a,s'); and the special form (4) then explains this: Note that γ plays a crucial role in (4). This particular form of the energy function forces h_ϕ to learn a very particular state-only-dependent term that relates to discounting and recursiveness of values; while the g_θ term may depend on action. The last equation in Sec. 6 illustrates how this forces the training to converge to a Q and V function.

2 Inverse RL on a Toy Control Problem

Consider a trivial control domain, with state $x \in \mathbb{R}$, controls $u \in [-1,1]$, and deterministic state transitions $x_{t+1} = x_t + u_t$.
The expert policy π^* is deterministic and chooses $\pi(x) = \text{clip}(-x)$, where $\text{clip}(x) = \max\{-1, \min\{+1, x\}\}$ (a typical notation for clipping you should get used to).

a) What is a reward function $R(x)$ (depending on state only), such that the expert policy π^* is optimal? Derive the Q-function $Q^{\pi^*}(x,u)$ for your reward function $R(x)$ and prove that π^* is optimal. Assume a given discounting $\gamma \in [0,1)$. Is π^* the only optimal policy for your $R(x)$, or do equally optimal policies exist?

Obvious reward function $R(x=0) = 1$ and zero otherwise.

The value function in state $x=0$ is $M = \frac{1}{1-\gamma}$, the value in other states is $V^*(x) = \gamma^{\lceil |x| \rceil} M$.

The Q-function equals the value function for optimal controls; and γV for non-optimal controls (moving away). Assuming $x > 0$, whenever $u \leq \text{fmod}(x)$ we are still optimal.

b) For a given γ, there exist many reward functions $R(x)$ such that π^* is optimal. (Rescaling R is trivial – neglect this.) Describe a space of alternative reward functions such that π^* is still optimal; e.g., find some (non-trivial) $F(x)$ such that for $R(x) \leftarrow R(x) + F(x)$, π^* is still optimal.

Given any monotone decreasing function $g(x)$, $R(x) = g(|x|)$ should lead to the same optimal behavior.

Or adding a constant to any such R should also lead to the same behavior.

[Note, this sounds like a question about reward shaping (=changing R while guaranteeing invariance of the optimal policy) [2]. However, this question is slightly different, as we have a concrete deterministic dynamics and do not require invariance w.r.t. all possible world dynamics.]

c) Now, conversely, find a (minimal) variation $F(x)$ such that for $R(x) \leftarrow R(x) + F(x)$, π^* is not optimal anymore.

[This illustrates how a choice of reward function can discriminate between policies; as is implicit in adversarial InvRL.]

3 Practical Exercise: Exploration in RL

In this exercise, we will revisit the Continuous Mountain Car problem from `gym`. Previously, running SAC with default parameters from StableBaselines3 did not perform well. This week, we will explore whether exploration can make things work better.
One way to explore in RL is by adding noise to the actions taken. The paper *Pink Noise Is All You Need: Colored Noise Exploration In Deep Reinforcement Learning* (https://openreview.net/pdf?id=hQ9V5QN27eS) compares three types of noise:

- Gaussian (white) noise
- Ornstein-Uhlenbeck (OU) noise
- Pink noise

Our goal is to compare the effects of these noises on agent actions during training.

a) Review the `ActionNoise` wrapper from StableBaselines3 (https://stable-baselines3.readthedocs.io/en/master/_modules/stable_baselines3/common/noise.html#ActionNoise), and the Pink Noise paper. Implement a child class `MyPinkNoise(ActionNoise)` that returns pink noise when called. Skeleton code is provided; you need to implement the call and reset methods.

b) StableBaselines3 includes implementations of Gaussian and OU noise (https://stable-baselines3.readthedocs.io/en/master/common/noise.html). Using your pink noise implementation, plot the different noise traces by plotting the 1D action on the y-axis and the time step on the x-axis with `scale=0.3` for all noises.

What do you observe?

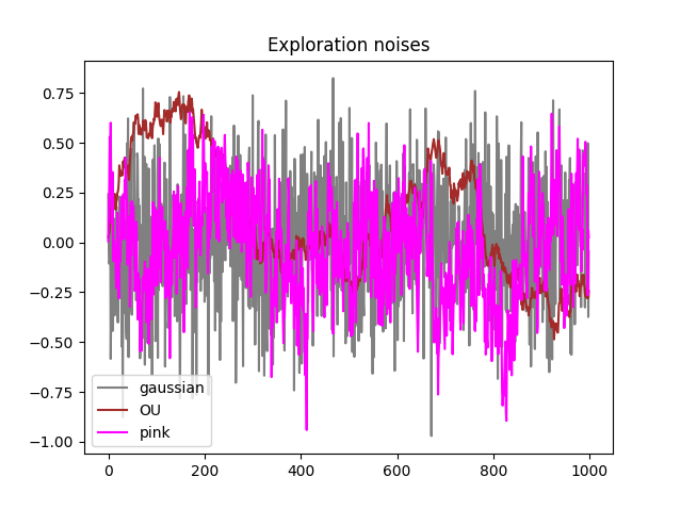
c) Use all three noise types to train SAC on MountainCarContinuous with default parameters. Using `scale=0.3`, train for `total_timesteps=2e4`.

What do you observe? Plot the learning curves of all training runs.

HINT: It is not expected that all noises will lead to successful training. You do not need to adjust any SAC parameters.

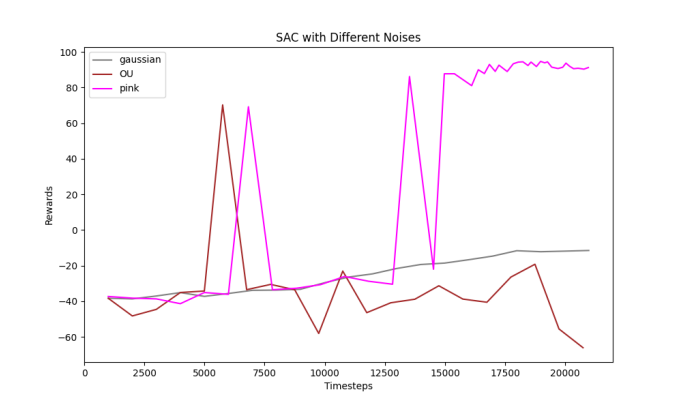
a) Look at the solution code in the supplementary materials for an implementation of the `MyPinkNoise` class.

b) Samples of the different noises look as follows:



We can see that OU noise is highly correlated over time. White noise (Gaussian) is completely uncorreltaed. Pink noise meets the middle ground: it is correlated, but less than OU. This guarantees exploration, while still staying closer to the actual policy than when OU noise is being used.

c) The learning curves look as follows:



We can see that only the agent that uses pink noise for explration learns to solve the task.

This gives us the following insights: Pink noise exploration can help to solve problems such as MountainCar. Using the noise can prevent us from needing meticulous hyperparameter tuning. Secondly, we see that pink noise meets the sweet-spot of noise being just enough correlated over an episode. Without any correlation, the agent learns only slowly and converges to a local optimum (return 0). With OU noise, the exploration is so stong that the executed actions are too far of the current policy for learning to succeed.

NOTE: If you would like to use pink noise in your own future projects, there exists an official implementatino of the paper, which includes a wrapper for StableBaselines: https://github.com/martius-lab/pink-noise-rl.

References

[1] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning, 2018-08-13. URL: http://arxiv.org/abs/1710.11248, arXiv:1710.11248[cs].

[2] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999. URL: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/NgHaradaRussell-shaping-ICML1999.pdf.

[3] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000. URL: http://www.datascienceassn.org/sites/default/files/Algorithms%20for%20Inverse%20Reinforcement%20Learning.pdf.

[4] A. Tucker, A. Gleave, and S. Russell. Inverse reinforcement learning for video games, 2018-10-24. URL: http://arxiv.org/abs/1810.10593, arXiv:1810.10593[cs,stat].