Non-Adversarial Imitation Learning

This is a short note on non-adversarial imitation learning (NAIL) proposed by Arenz and Neumann (2020), with the goal of mitigating the instability, if not lack of convergence, of adversarial imitation learning (AIL). The idea is to look at AIL as a form of divergence minimization (this perspective has been adopted before; Ghasemipour et al., 2020) and derive an upper of the divergence measure (Kullback-Leibler divergence specifically) which has good algorithmic properties.

1 Adversarial Imitation Learning

Let us denote a dataset of N state-action pairs sampled by some policy with $\mathcal{D} = \{(s_{i,t}, a_{i,t})\}_{i=1}^{N}$. The goal of the imitation learning algorithm is to find a parameterized policy π which minimizes the *reverse* KL divergence between the marginal state-action distribution generated by π, which we denote with $Q_\pi(s,a)$, and the marginal state-action distribution in the dataset $P(s,a)$. The objective can be written as:

$$\begin{aligned} &\min_\pi \mathbb{E}_{Q_\pi(s,a)} \left[\log \frac{Q_\pi(s,a)}{P(s,a)} \right] \\ &= \mathbb{E}_{Q_\pi(s,a)} \left[\log r(s,a) \right] \end{aligned} \tag{1}$$

where $r(s,a)$ is called the density ratio.

However, we do not directly know the data distribution $P(s,a)$ as we only have samples from it. One way to make (1) still optimizable is to estimate the density ratio and optimize $Q_\pi(s,a)$ with respect to the estimated density ratio. It is possible to estimate density ratio using a binary classification approach, where we design a classifier to input a state-action pair and output the probability of whether the state-action pair is from the dataset (labeled as real) or generated by the policy (labeled as fake).

Let us denote the class labels with c. If we show the classifier equal number of state-action pairs from the real and fake distributions, the classifier should output the probability of a state-action pair being fake as:

$$\begin{aligned} P(c = \textit{fake}) &= \frac{Q_\pi(s,a)}{Q_\pi(s,a) + P(s,a)} = \frac{Q_\pi(s,a)/P(s,a)}{Q_\pi(s,a)/P(s,a) + 1} \\ &= \frac{r(s,a)}{r(s,a)+1} = \frac{\exp\left(\log r(s,a)\right)}{\exp\left(\log r(s,a)\right) + 1} \end{aligned} \tag{2}$$

Thus, if we parameterize a model outputting such a probability using a sigmoid transformation, we can interpret the sigmoid logits as the log density ratio of the two distributions.

Once the log density ratio is estimated, the policy can be trained to minimize the cumulative expected log density ratio using reinforcement learning:

$$\min_\pi \mathbb{E} \left[\sum_t \log r(s_t, a_t) \right] \tag{3}$$

where the expectation is taken with respect to a simulator.

Thus, AIL alternates between two steps:

1. Estimate the density ratio between the current policy's marginal state-action distribution and the data distribution using binary classification
2. Train the policy to minimize the cumulative expected log density ratio

The down side of this method is that when the marginal distribution generated by the policy gets close to the data distribution, the density ratio tends towards 1 and the log density ratio tends towards zero for all state-action pairs. This causes the policy to be trained in a way that favors all actions equally, which is a bad policy. The result of this is a cyclic pattern where the policy performance would increase, for example as measured by the discriminator binary cross entropy loss. However, when it gets close to the maximum value of discriminator binary cross entropy loss (around 0.67), the it would start dropping as the log density ratio reward reaches zero. When it drops to a level that is too low, it would start increasing again. There exists methods such as using discriminator gradient penalty to stabilize training, however they do not ultimately solve the problem.

2 Non-Adversarial Imitation Learning

NAIL transforms the reverse KL minimization in (1) in to one of a latent variable problem and derives a lower bound of the marginal reverse KL divergence. Specifically, the authors view the state-action pairs as observations, denoted with $o = (s, a)$, from a latent trajectory distribution $P(\tau)$ with observation probabilities $P(o|\tau)$. Below is my interpretation based on the original paper (Arenz and Neumann, 2020) and a related paper (Becker et al., 2020).

Under this view, the reverse KL divergence is transformed as follow:

$$\int_o Q_\pi(o) \log \frac{Q_\pi(o)}{P(o)} = \int_\tau \int_o P(o|\tau) Q_\pi(\tau) \left[\log \frac{P(o|\tau) Q_\pi(\tau)}{P(o)} - \log Q_\pi(\tau|o) \right] \tag{4}$$

We assume the observation probability $P(o|\tau)$ is fixed and not part of the model parameterization as it only depends on the simulator.

We also define a new notion of log density ratio:

$$\log \tilde{r}_t(o) = \log \frac{Q_t(o)}{P(o)} \tag{5}$$

which is defined with respect to the marginal observation distribution generated by the policy at iteration t.

We will swap the new log density ratio into (4) by multiplying both the numerator and the denominator of the first fraction with $Q_t(o)$:

$$\begin{aligned}
&\int_\tau \int_o Q_\pi(o, \tau) \log \frac{P(o|\tau) Q_\pi(\tau)}{P(o)} \left[\frac{Q_t(o)}{Q_t(o)} - \log Q_\pi(\tau|o) \right] \\
&= \int_\tau \int_o Q_\pi(o, \tau) \left[\log \frac{Q_t(o)}{P(o)} + \log \frac{P(o|\tau) Q_\pi(\tau)}{Q_t(o)} - \log Q_\pi(\tau|o) \right] \\
&= \int_\tau \int_o Q_\pi(o, \tau) \left[\log \frac{Q_t(o)}{P(o)} + \log \frac{P(o|\tau) Q_\pi(\tau)}{P(o|\tau) Q_t(\tau)} \frac{Q_t(\tau|o)}{1} - \log Q_\pi(\tau|o) \right] \\
&= \int_\tau \int_o Q_\pi(o, \tau) \left[\log \frac{Q_t(o)}{P(o)} + \log \frac{P(o|\tau) Q_\pi(\tau)}{P(o|\tau) Q_t(\tau)} \frac{Q_t(\tau|o)}{1} - \log Q_\pi(\tau|o) \right] \\
&= \int_\tau \int_o Q_\pi(o, \tau) \left[\log \frac{Q_t(o)}{P(o)} + \log \frac{P(o|\tau) Q_\pi(\tau)}{P(o|\tau) Q_t(\tau)} - \log \frac{Q_\pi(\tau|o)}{Q_t(\tau|o)} \right] \\
&= OLEB - \mathbb{E}_{Q_\pi(o)} \left[D_{KL}[Q_\pi(\tau|o) || Q_t(\tau|o)] \right]
\end{aligned} \tag{6}$$

The name $OLEB$ was borrowed from a blog[1], which as we see here can be seen as the reverse KL divergence analog of the evidence lower bound.

[1] http://boris-belousov.net/2019/07/24/reverse-elbo/

Plugging in the log density ratio, $OLEB$ can be written as:

$$OLEB = \int_{\tau} \int_{o} P(o|\tau) Q_{\pi}(\tau) \log \tilde{r}_t(o) + D_{KL} \left[Q_{\pi}(\tau) || Q_t(\tau) \right] \tag{7}$$

The objective function in (7) suggest the following objective for the reinforcement learning problem:

$$\min_{\pi} \mathbb{E} \left[\sum_{t} \log \tilde{r}(s_t, a_t) - \log \tilde{\pi}(a_t|s_t) \right] \tag{8}$$

where $\tilde{\pi}(a_t|s_t)$ is the policy from the previous iteration. Thus, when the log density ratio reaches zero, the optimizing policy will simply ignores the density ratio and instead tries to match itself.

3 Related Ideas

The NAIL policy objective in (8) can be interpreted as regularizing the new policy's KL divergence from the previous policy. This echos with the original Generative Adversarial Imitation Learning's using of trust region policy optimization, which also performs KL regularized policy updates (Ho and Ermon, 2016). Adding the log generator probability to the density ratio objective also appeared in a recent recommendation for training generative models (Zhang et al., 2022).

4 Implementation

We set up our training loop following the discriminator actor-critic framework (Kostrikov et al., 2018) and train the reinforcement learning policy using the soft actor-critic algorithm (Haarnoja et al., 2018). The only difference is that we use the NAIL custom reward defined in (8).

To put NAIL and AIL on equal footing, we maintain a reference policy $\tilde{\pi}$ for both algorithms and update it to the current policy at the end of each training epoch, where a training epoch is measured by a certain number of environment steps. During each training epoch, we use the reference policy to sample transitions. This is absolutely crucial as updating the policy in the middle of an epoch will lead to highly unstable behavior.

It is also crucial to handle environment terminal states as described in Kostrikov et al. (2018). In the mountain car environment, not handling the terminal state correctly will cause the algorithms to not converge.

Lastly, when calculating the policy likelihood bonus for the NAIL algorithm, I used a higher softmax temperature parameter than what's used in rollouts and RL policy training ($\beta = 1$ vs $\beta = 0.1$). This prevents the policy likelihood from becoming too large prematurely, which will lead to difficulty of improving performance.

References

Arenz, O., & Neumann, G. (2020). Non-adversarial imitation learning and its connections to adversarial methods. *arXiv preprint arXiv:2008.03525*.

Becker, P., Arenz, O., & Neumann, G. (2020). Expected information maximization: Using the i-projection for mixture density estimation. *arXiv preprint arXiv:2001.08682*.

Ghasemipour, S. K. S., Zemel, R., & Gu, S. (2020). A divergence minimization perspective on imitation learning methods. *Conference on Robot Learning*, 1259–1277.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International conference on machine learning*, 1861–1870.

Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. *Advances in neural information processing systems*, *29*.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., & Tompson, J. (2018). Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*.

Zhang, D., Chen, R. T., Malkin, N., & Bengio, Y. (2022). Unifying generative models with gflownets. *arXiv preprint arXiv:2209.02606*.