Discussion of GOLD

Abstract

GOLD does not maximize the expected reward. It maximizes the expected reward of training examples only. *Acknowledgement: This piece originates entirely from discussion with Kyunghyun Cho. Written in June 2022.*

1 Introduction

Pang & He (2021) propose GOLD (generation by off-policy and offline learning from demonstrations), which improves the performance of a few supervised conditional text generation tasks including question answering, extractive and abstractive summarization, and machine translation.

A few recent exciting applications of GOLD include DeepMind's AlphaCode (Li et al., 2022) where GOLD serves as a crucial algorithm to fine-tune the code generator on the supervised CodeContests dataset, as well as Dou et al. (2022) where the vision-language pretraining approach achieves state-of-the-art image captioning results with help from GOLD.

In on-policy policy gradient algorithms in text generation, during training, a sequence is sampled from the current policy (i.e., text generator) and the reward is computed on this sampled sequence. GOLD, on the contrary, is an off-policy and offline policy gradient algorithm, and we do not sample from the current policy during training. Instead, we sample from another policy: the behavioral policy in the case of GOLD – the behavioral policy corresponds to the policy induced by the dataset.

Sampling from the behavioral policy requires us to incorporate the importance weighting term $\pi_\theta(\tau)/\pi_b(\tau)$ into the policy gradient, where τ is a trajectory of states and actions. Without loss of generality, assume that the trajectory only has one step. Then, the importance weight becomes $\pi_\theta(a \mid s)/\pi_b(a \mid s)$ for some action a and state s. However, we do not know what $\pi_b(a \mid s)$ is. A naive workaround is to use the empirical distribution as an approximation to the true behavioral policy.

To summarize, here is the original story: to optimize $J(\theta) = \mathbb{E}_{s,a}[r(s,a)]$, we arrive at this off-policy policy gradient in which we need to approximate π_b using an empirical distribution. If we ponder on this approximation, we would ask: Is this approximation too liberal? Does it end up changing the original objective J?

It turns out the answer is yes – it changes the original objective a bit. So, instead of calling GOLD's gradient an approximation to the gradient of the original objective J, I think it would be more prudent to revise the original objective J. Rest assured that the policy-gradient-based algorithm and implementation in the GOLD paper stay exactly the same.

For the details of the algorithm, I refer readers to the original paper Pang & He (2021) or an article published in the ICLR 2022 blog post track, Kantharaju & Sankar (2022). Below contains an explanation of what our updated objective is, while the gradient and the implementation remain unchanged.

Quick background. Given a context $\boldsymbol{x}$, we want to generate a sequence of tokens $\boldsymbol{y} = (y_0, \ldots, y_T)$. The generation process can be viewed as a sequential decision-making process. At time-step t, the policy π_θ takes an action $a_t \in \mathcal{V}$, transits to the next state $s_{t+1} = (\boldsymbol{y}_{0:t}, \boldsymbol{x})$, and receives a reward r_t. The policy corresponds to the generation model: $\pi_\theta(a_t \mid s_t) = p_\theta(a_t \mid \boldsymbol{y}_{0:t-1}, \boldsymbol{x})$. We can thus

represent a sequence as a trajectory $\tau = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$. The set of trajectories derived from the training data is called demonstrations which show the desired behavior of a policy.

2 From Objective to Gradient

Without loss of generality, we assume $\gamma = 1$ and the trajectory contains only one step. Here is our *original* objective function.

$$J(\theta) = \mathbb{E}_{s\sim p_{s_0}, a\sim\pi_\theta}[r(s,a)]. \tag{1}$$

Now we compute the gradient of the objective.

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla \mathbb{E}_{s\sim p_{s_0}, a\sim\pi_\theta}[r(s,a)] \\
&= \nabla \sum_s p_{s_0}(s) \sum_a \pi_\theta(a \mid s) r(s,a) \\
&= \sum_s p_{s_0}(s) \sum_a r(s,a) \nabla \pi_\theta(a \mid s) \\
&= \mathbb{E}_s \left[\sum_a r(s,a) \nabla \pi_\theta(a \mid s) \right] \\
&= \mathbb{E}_s \left[\sum_a \frac{\pi_b(a \mid s)}{\pi_b(a \mid s)} r(s,a) \nabla \pi_\theta(a \mid s) \right] \\
&= \mathbb{E}_s \left[\mathbb{E}_{a\sim\pi_b(\cdot|s)} \left[r(s,a) \frac{\nabla \pi_\theta(a \mid s)}{\pi_b(a \mid s)} \right] \right].
\end{aligned}$$

Using the policy gradient "log-derivative trick," we have the following derivation.

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_s \left[\mathbb{E}_{a\sim\pi_b(\cdot|s)} \left[r(s,a) \frac{\pi_\theta(a \mid s)}{\pi_b(a \mid s)} \frac{\nabla \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \right] \right] \\
&= \mathbb{E}_s \left[\mathbb{E}_{a\sim\pi_b(\cdot|s)} \left[r(s,a) \frac{\pi_\theta(a \mid s)}{\pi_b(a \mid s)} \nabla \log \pi_\theta(a \mid s) \right] \right].
\end{aligned} \tag{2}$$

The GOLD algorithm uses the empirical distribution as the behavioral policy, which appears in the denominator of the importance weight.

$$\tilde{\pi}_b(a \mid s) = \begin{cases} \frac{1}{|D(s)|} \sum_{(s',a')\in D(s)} \mathbb{1}(a = a'), & \text{if } D(s) \neq \varnothing \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $D(s) = \{(s', a') \in D \mid s' = s\}$ and D is the set of demonstrations.

Given Pang & He (2021)'s assumption that $\pi_b(a \mid s) \propto C$ if $D(s) \neq \varnothing$ for some $C \in \mathbb{R}$, with the above approximation, we arrive at the following gradient based on Eq. (2):

$$\nabla_\theta H(\theta) = \mathbb{E}_s \left[\frac{1}{M} \sum_{m=1}^{M} r\left(s, a^{(m)}\right) \frac{\pi_\theta\left(a^{(m)} \middle| s\right)}{C} \nabla \log \pi_\theta\left(a^{(m)} \middle| s\right) \right]. \tag{4}$$

Given that $C \in \mathbb{R}$ is a constant, it can be ignored from the gradient in implementation.

Does the approximation change the original objective J, though? We now investigate. H is an objective function – we do not yet know what it looks like but we will derive now. It turns out H and J are slightly different.

3 From Approximated Gradient to Objective

Ignoring the constant C, we have the following:

$$\begin{aligned}\nabla H(\theta) &= \mathbb{E}_s\left[\frac{1}{M}\sum_{m=1}^{M}\left[r\left(s,a^{(m)}\right)\nabla\pi_\theta\left(a^{(m)}\middle|s\right)\right]\right] \\ &= \mathbb{E}_s\left[\mathbb{E}_{a\sim\tilde{\pi}_b(\cdot|s)}\left[r(s,a)\nabla\pi_\theta(a\mid s)\right]\right] \\ &= \mathbb{E}_s\left[\nabla\mathbb{E}_{a\sim\tilde{\pi}_b(\cdot|s)}\left[r(s,a)\pi_\theta(a\mid s)\right]\right] \\ &= \mathbb{E}_s\left[\nabla Z(s)\mathbb{E}_{a\sim\pi'(\cdot|s)}[r(s,a)]\right], \end{aligned} \tag{5}$$

where $\pi'(a|s) = \frac{1}{Z(s)}\tilde{\pi}_b(a\mid s)\pi_\theta(a\mid s)$ and $Z(s) = \sum_a \tilde{\pi}_b(a\mid s)\pi_\theta(a\mid s)$. We call $Z(s)$ the partition function. Next,

$$\nabla H(\theta) = \mathbb{E}_s\left[\nabla\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[r(s,a)\tilde{\pi}_b(a\mid s)\right]\right],$$

where π_b is defined in Eq. (3).

Finally, we have

$$H(\theta) = \mathbb{E}_{s\sim p_{s_0},a\sim\pi_\theta}\left[r(s,a)\tilde{\pi}_b(a\mid s)\right]. \tag{6}$$

In other words, we would maximize the expected reward only if the trajectory appears in the training set.

4 What's Going On?

Here is the original story. Eq. (1) is our original objective. After approximating the behavioral policy, we arrive at the gradient defined in Eq. (4).

The approximation, in fact, slightly changes the original objective. We end up optimizing an updated objective: Eq. (6).

The lesson is that perhaps it would be more prudent to start from Eq. (6), and then achieve the gradient Eq. (4), which stays unchanged.

Interpretation using coaching. Observe the gradient in Eq. (5), π' is similar to the coaching oracle (He et al., 2012) where $\pi^*_{\text{coach}}(a\mid s) \propto \pi^*_{\text{uniform}}(a\mid s)\pi(a\mid s)$. The uniform oracle π^*_{uniform} assigns uniform probabilities only on valid actions that achieve the final goal; in our case, the valid actions correspond to actions in the training corpus. The coaching oracle makes sure that invalid actions (in our case, any actions not appearing in the training corpus) are not assigned any probability; at the same time, the coaching oracle prefers actions that are also preferred by the current policy π. Therefore, optimizing $H(\theta)$ can be seen as maximizing the expected reward under the product policy π', which is the product of our current policy and the behavioral policy.

References

Zi-Yi Dou, Aishwarya Kamath, Zhe Gan, Pengchuan Zhang, Jianfeng Wang, Linjie Li, Zicheng Liu, Ce Liu, Yann LeCun, Nanyun Peng, Jianfeng Gao, and Lijuan Wang. Coarse-to-fine vision-language pre-training with fusion in the backbone. *arXiv preprint arXiv:2206.07643*, 2022.

He He, Jason Eisner, and Hal Daume. Imitation learning by coaching. *Advances in Neural Information Processing Systems*, 25, 2012. URL `https://proceedings.neurips.cc/paper/4545-imitation-learning-by-coaching.pdf`.

Pavan Kantharaju and Aiswarya Sankar. An understanding of learning from demonstrations for neural text generation. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/text-gen-via-lfd/`. https://iclr-blog-track.github.io/2022/03/25/text-gen-via-lfd/.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*, 2022. URL https://storage.googleapis.com/deepmind-media/AlphaCode/competition_level_code_generation_with_alphacode.pdf.

Richard Yuanzhe Pang and He He. Text generation by learning from demonstrations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=RovX-uQ1Hua.