RL Framework for incorporating natural language feedback alongside numerical reward.

Tanay Gahlot
tgahlot@student.ethz.ch

Abstract

Adaptation of large language models to downstream task has shown promising results but it remains costly and resource intensive for majority of production applications. In this work, we focus on improving the convergence rate to reduce the cost of RL based fine-tuning method. Currently, majority of RL based fine-tuning techniques focus on numerical rewards and ignore the natural language feedback. We propose two techniques to combine natural language feedback with numerical feedback in existing policy gradient setup. Firstly, we combine the objective of policy gradient with imitation learning objective to get a speedup of 1.5x. Secondly, we introduce expert trajectories into the episode to get 2x improvement in convergence rate.

1 Introduction

Fine tuned language models show promising results in a zero/few shot setting [12] but often this is not enough for a task with unknown environment and non-stationary data [14]. Furthermore, the data collection for continuous fine-tuning is resource intensive and can prove prohibitive for low resource tasks [5]. For such tasks, learning by interacting with the end-user provides an alternative to resource intensive fine-tuning. We explore the techniques to incorporate the following types of user feedback in our RL framework to speedup training:

1. **Numerical feedback** on whether the response was considered satisfactory or not.
2. Free-form **Textual feedback** on what was wrong in the case of an unsatisfactory response.
3. **Expected response** in the case of an unsatisfactory response.

For instance, in the conversation below, the user is looking for food rich in fiber, and expects the results to be supported by evidence:

User: "Could you please help me figure out what the best food sources are for fiber in an American diet?"
Bot: "Raspberries, pears, apples, green peas, and broccoli are known for having high fiber."
User: "That's so helpful to know. Do you have a source you can share with me for those facts? A website link?"
Bot: "I don't have a website link, but I can tell you that a lot of fruits and vegetables are high in fiber."

The feedback will be:

1. Numerical feedback = 0 i.e unsatisfied.
2. Textual feedback = "You can tell me the source so I feel comfortable with the information, like Mayo Clinic or USDA Nutrient Database."

Foundations of Reinforcement Learning, FS2024. Department of Computer Science, ETH Zürich.

3. Expected response = "As per the USDA Nutrient Database, Fruits(Raspberries, pears and apples) and vegetables(green peas, and broccoli) are high in fiber."

2 Related Work

While many techniques have been proposed to incorporate Numerical feedback in RL, natural language feedback like Textual feedback and Expected response remains largely unexplored[2]. Therefore, we will focus on the challenges in incorporating textual feedback as a means to improving convergence.

2.1 Textual Feedback

Textual feedback is currently incorporated by using the following approaches:

1. By using supervised learning with an objective to predict the textual feedback given the input and the output. In this way model becomes aware of the potential pitfalls in generating output y to the input x.[14]. In this case, the feedback is only applicable to the input x and output y.
2. By using textual feedback to generate a small synthetic preference dataset specifying how the feedback should (and should not) be applied. It then fine-tunes the language model in accordance with the synthetic preference data while minimizing the divergence from the original model for prompts where the feedback does not apply[8]. While this approach overcomes the limitation of the first approach, the synthetic preference dataset may not be a true

2.2 Expected Response

Incorporating expected response to fine-tune language model is similar to learning from demonstration in Imitation learning. It can be performed in offline and online setup. In offline setting, the demonstration data is sampled from the expert policy:

$$D = \{(s_i, a_i)\}_{i=1}^{i=n} \text{ where } a_i \sim \pi_E \tag{1}$$

The offline dataset D is used to find a policy using supervised learning. Most prominent objective used in practice is Maximum likelihood estimation(MLE).

$$\max_{\pi \in \Pi} \sum_{(s_i, a_i) \in D} log(\pi(a_i|s_i)) \tag{2}$$

which is equivalent to minimizing KL divergence between expert policy π_E and π_θ. While offline demonstrations are easier to acquire, it has lower convergence rate in comparison to on-policy learning[3]. Furthermore, there is no long term planning which can result in cascading errors.

Online imitation learning avoids the long term planning and cascading error pitfall by sampling the dataset from the agent policy instead of expert policy:

$$D = \{(s_i, a_i)\}_{i=1}^{i=n} \text{ where } a_i \sim \pi_\theta \tag{3}$$

In online setting, three approaches are most prominent in practice:

1. Data Aggregation[7]: Iteratively build up a policy via supervised learning on aggregated data from the expert.
2. Policy Aggregation[6]: Iteratively build up a policy by mixing newly trained policies.
3. Value Aggregation[9]: Iteratively build up a policy by minimizing experts cost to go $A^{\pi_E}(s, a)$.

3 Algorithms

We being by formulating the language generation problem as a Markov decision process(MDP) where humans can provide feedback at various stages. Then, we discuss the feedback in formal terms which enables us to reason about them. We start by discussing how numerical rewards are incorporated into MDP using policy gradient method. At last, we discuss how natural language feedback can be incorporated into existing policy gradient methods.

3.1 Language Generation as a Markov decision process

Consider a language model $M : \mathcal{X} \to \mathcal{Y}$ which, given an input of some type $x \in \mathcal{X}$, outputs text $\hat{y} \in \mathcal{Y}$. Importantly, while x can be of any format, we restrict ourselves to cases where y is in the space of *natural language* (*i.e.*, $\mathcal{Y} \subseteq \Sigma^\star$ for some alphabet Σ). These models are generally realized as a parameterized, conditional probability distribution $\pi_\theta(y|x)$, where θ are the model parameters. This distribution is often estimated autoregressively: the probability of a sentence y given an input x is decomposed into the product of the probabilities of each token in the sentence, conditioned on the previous tokens.

Consider a finite, discrete-time MDP [10], which is a tuple $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $p : \mathcal{S} \times \mathcal{A} \to Dist(\mathcal{S})$ is the environment transition probability function, and $\gamma \in [0, 1)$ is the discount factor.

The process of decoding a sentence y given an input x can be modeled as a finite, discrete-time MDP where

1. $\mathcal{S}$ is all possible strings $\Sigma^\star$
2. $\mathcal{A}$ is all possible tokens Σ
3. $a_t =$ argmax $\pi_\theta(a|s_t)$ where π_θ is the language model defined above.
4. Next state is given by $s_{t+1} = Concat(s_t, a_t)$ and the transition probability is $p(s_{t+1}|s_t, a_t) = 1$ where Concat function concatenates the previously generated string and input s_t with the latest token a_t
5. $r(s_t, a_t)$ is the discounted numerical feedback which might be received at the current step or in the future.
6. $\mu_0 = x$ where x is the input string.
7. The MDP terminates when $a_t = EOS$ where EOS is the end of sequence token.
8. $y = Concat(a_0, a_1, a_2, ..., EOS)$

3.2 Formalizing human Feedback

Human feedback can be formalized as:

1. Numerical feedback N which takes an input and output and returns a single score ($\mathcal{X} \times \mathcal{Y} \to \mathcal{N} \subseteq \mathbb{R}$).
2. Textual feedback $\mathcal{F} \subseteq \Sigma^\star$.
3. Expected response $\mathcal{E} \subseteq \Sigma^\star$.

3.3 Policy gradient method and numerical reward

The value function $V^\pi : \mathcal{S} \to \mathbb{R}$ of a policy is defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r_t | s_0 = s]$, where $\mathbb{E}_\pi$ denotes the expectation of following π in the MDP and T is a random variable denoting when a terminal state is reached. Similarly, the state-action value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r_t | s_0 = s, a_0 = a]$. The advantage A^π is then given by $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. We denote ρ_π as the distribution over trajectories $\tau = (s_0, a_0, r_0, \ldots, s_T)$ sampled by π when interacting with the MDP.

The policy gradient theorem states that, in order to optimize the RL objective $\mathbb{E}_{s_0 \sim \rho_0}[V^\pi(s_0)]$, a parameterized policy should be updated with respect to the loss function,

$$J_{\text{PG}}(\pi_\theta) = \mathbb{E}_{\tau \sim \rho_{\pi_\theta}} \left[\sum_{t=0}^{T-1} \gamma^t \cdot \hat{Q}_t \log \pi_\theta(a_t|s_t) \right], \tag{4}$$

where $\hat{Q}_t$ is an unbiased estimate of $Q^\pi(s_t, a_t)$. In our case, $\hat{Q}_t$ is the empirically observed future(i.e numerical feedback N) discounted return following s_t, a_t.

3.4 Incorporating expected response

Expected response $\mathcal{E}$ can be thought of as demonstration which are sampled from the expert policy π_E in an online setting.

$$\mathcal{E} = \{(s_i, a_i)\}_{i=1}^{i=n} \tag{5}$$

where $a_i \sim \pi_E$. For each trajectory τ sampled from the agents policy π, we have a corresponding expert trajectory defined as:

$$\tau = (s_0, a_0, r_0, \ldots, s_T) \tag{6}$$

where $\tau \sim \rho_{\pi_E}$. The expert trajectory starts from the same initial state s_0 as agent's trajectory. In subsequent section, we will discuss approaches to incorporate expert demonstrations to policy gradient methods.

3.4.1 Joint loss

The objective of policy gradient methods can be modified to accommodate expert demonstrations by adding behaviour cloning objective. We chose KL divergence between the expert's action distribution and that of the imitating policy:

$$J_{\text{IL}}(\pi_\theta) = E_{s\sim d^{\pi_\theta}, a\sim\pi_E(.|s)}[\log(\frac{\pi_E(a|s)}{\pi_\theta(a|s)})] \tag{7}$$

The joint objective is:

$$J(\pi_\theta) = \alpha J_{\text{PG}}(\pi_\theta) + (1-\alpha) J_{\text{IL}}(\pi_\theta) \tag{8}$$

The gradient $\nabla_\theta J(\pi_\theta)$ can be estimated by estimating the gradient of its components:

$$\nabla J(\pi_\theta) = \alpha \nabla J_{\text{PG}}(\pi_\theta) + (1-\alpha)\nabla J_{\text{IL}}(\pi_\theta) \tag{9}$$

∇J_{PG} can be estimated using policy gradient theorem[11]. Since minimizing KL divergence is equivalent to maximizing the likelihood objective[3], ∇J_{IL} can be computed similar to gradient of maximum likelihood objective.

3.4.2 Injected into trajectory

Expert trajectory $\tau \sim \rho_{\pi_E}$ can be artificially introduced in addition to agent trajectories in the training episode of policy gradient with a high numerical reward $R(\tau)$. It can be done by using two environment, one for expert and the other for agent. At the start of the episode, both expert and agent take actions using agent's policy as shown in the 1. After n steps, expert stops following the agents policy and starts acting based on its own policy. n is sampled from a normal distribution with the forking probability p. Agent continues to act on the basis of its own policy.

At the end of episode, we will have two trajectory, one corresponding to the expert τ_E and the other corresponding to the agent τ_θ. The policy gradient objective will be:

$$J_{\text{PG}}(\pi_\theta) = \mathbb{E}_{\tau_\theta\sim\rho_{\pi_\theta}, \tau_E\sim\rho_{\pi_E}}\left[\sum_{t=0}^{T-1} R_t(\tau)\log\pi_\theta(a_t|s_t)\right], \tag{10}$$

where $R_t(\tau)$ is the reward to go at time t for the trajectory τ. Note that the log probs are corresponding to the agent's policy, even for expert trajectory i.e $(a_t, s_t) \in \tau_E$.

4 Evaluation

4.1 Setup

The proposed approaches are compared against REINFORCE[13] in Mujocu InvertedPendulum[4] environment. The metric of comparison is rewards over episode since its a good proxy to convergence rate. Furthermore, to weed out the brittleness[1], we compute this metric over multiple seeds. It also helps in quantifying the variance of approaches. The candidate for comparison are:

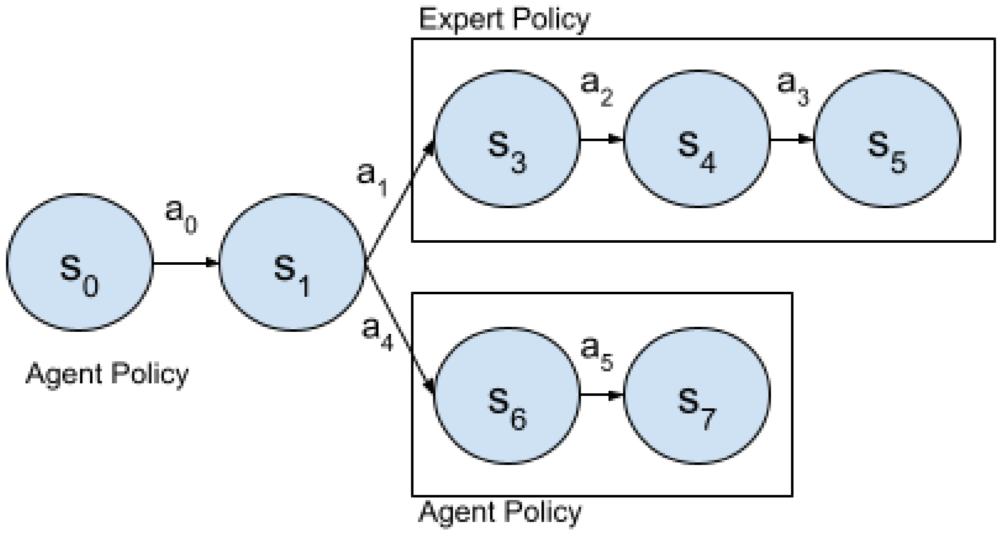

Figure 1: Explanation of how the expert trajectory are introduced in the set of agent trajectories

1. **Base**: Perform policy gradient with numerical reward.
2. **Exp**: Combine policy gradient(with numerical reward) and expert demonstration using
   (a) By introducing expert trajectory.
   (b) By introducing joint objective.

In the setup above, the expert demonstration comes from an Agent trained using REINFORCE. It can achieve the maximum reward for the env i.e 1000(before the env truncates).

To generate the expert trajectory, two env were used, one for expert and the other for the agent. The expert env took action using agent policy until the forking point. Post that, it employed expert policy. The forking point is decided based on the forking probability which is set to .4. It implies that expert env will follow agent policy for first 4 steps on average and expert policy thereafter.

For the joint loss approach, we used alpha=.5. It implies that we give equal importance to policy gradient objective and imitation learning objective.

4.2 Result

Joint loss provides better convergence rate in comparison to REINFORCE as shown in 2. At 2000 episode, it provides the same reward as REINFORCE at 3000 episode. This implies 1.5x speedup in convergence. However, joint loss seems to have higher variance in comparison to REINFORCE.

Trajectory injection performs better than REINFORCE as shown in 3. The difference in convergence rate is apparent in the first 600 episodes. Trajectory injection achieves reward=20 in half the steps required for REINFORCE. This implies a 2x speedup in convergence. However similar to joint loss, trajectory injection has higher variance in comparison to the REINFORCE.

5 Conclusion

Joint loss and Trajectory injection shows promising results of combining learning from numerical reward(reinforcement learning) and learning from expert demonstration(imitation learning) in improving convergence rates. While the techniques show positive results in the inverted-pendulum environment, they are yet to be tested in a language modelling setup. Furthermore, higher variance in the proposed techniques point to headroom in improving the convergence rate even further and stabilizing the fine-tuning overall.

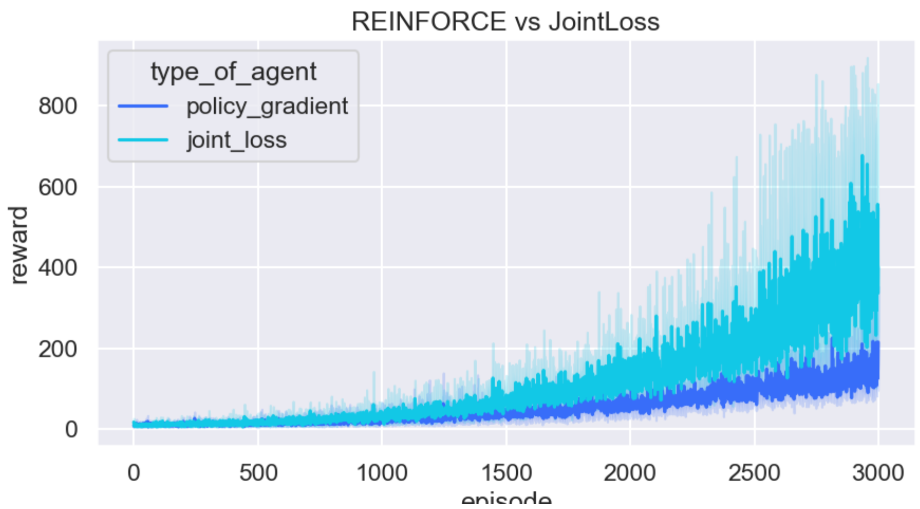

Figure 2: Comparison of Policy gradient(REINFORCE) and Joint loss in terms of rewards over episodes.

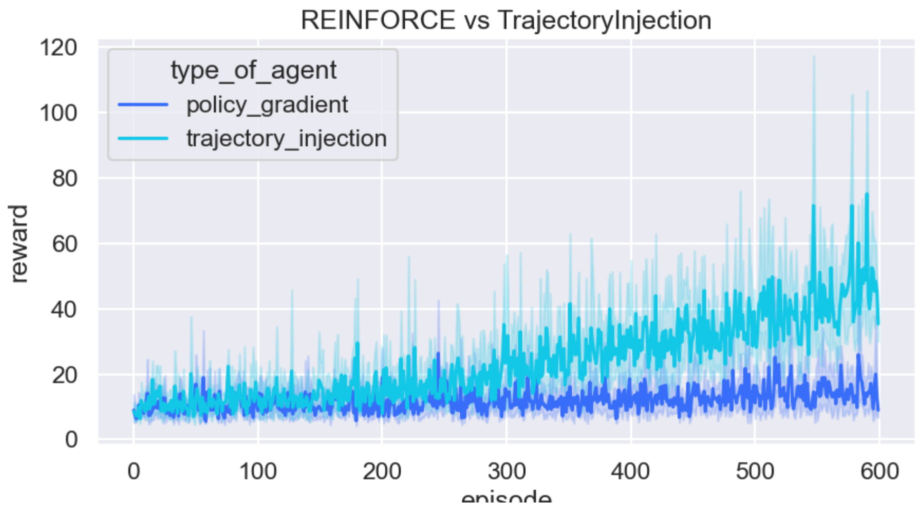

Figure 3: Comparison of Policy gradient(REINFORCE) and Trajectory injection in terms of rewards over episodes.

References

[1] Joshua Achiam. Spinning up as a deep rl researcher. *OpenAI Docs*, 2018.

[2] Patrick Fernandes, Aman Madaan, Emmy Liu, António Farinhas, Pedro Henrique Martins, Amanda Bertsch, José G. C. de Souza, Shuyan Zhou, Tongshuang Wu, Graham Neubig, and André F. T. Martins. Bridging the Gap: A Survey on Integrating (Human) Feedback for Natural Language Generation. *Transactions of the Association for Computational Linguistics*, 11: 1643–1668, 12 2023. ISSN 2307-387X. doi: 10.1162/tacl_a_00626. URL `https://doi.org/10.1162/tacl_a_00626`.

[3] Niao He. Week 07: Imitation learning, 2024. URL `https://moodle-app2.let.ethz.ch/mod/resource/view.php?id=1048809`.

[4] Jehandad Khan, Khalid Munawar, Raja Amer Azeem, and Muhammad Salman. Inverted pendulum with moving reference for benchmarking control systems performance. In *2009 American Control Conference*, pages 3764–3768, 2009. doi: 10.1109/ACC.2009.5160094.

[5] Dataset over algorithm. Dataset over algorithm. *Space machine*, 2016.

[6] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[7] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[8] Moritz Stephan, Alexander Khazatsky, Eric Mitchell, Annie S Chen, Sheryl Hsu, Archit Sharma, and Chelsea Finn. Rlvf: Learning from verbal feedback without overgeneralization, 2024.

[9] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International conference on machine learning*, pages 3309–3318. PMLR, 2017.

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[11] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[12] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022.

[13] Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach Learn 8, 229–256 (1992)*, 1992. URL `https://doi.org/10.1007/BF00992696`.

[14] Jing Xu, Megan Ung, Mojtaba Komeili, Kushal Arora, Y-Lan Boureau, and Jason Weston. Learning new skills after deployment: Improving open-domain internet-driven dialogue with human feedback, 2022.