Direct Preference Optimization vs. Reward Modeling for RLHF in the MiniGrid Environment

Juan David Guerra
School of Computer Science
McGill University

Shahrad Mohammadzadeh
School of Computer Science
McGill University

Taz Scott-Talib
School of Computer Science
McGill University

Abstract

Our study compares Direct Preference Optimization (DPO) [1] with a reward-model-based Proximal Policy Optimization (PPO) in the MiniGrid environment from the Gym suite. Using an RLHF-supporting class, we integrated preference data directly into our models for learning. We trained each model, evaluating its performance against various environment benchmarks.

1 Introduction

The PPO approach relies on traditional machine learning techniques to produce a reward model, resulting in potential generalization downfalls from suboptimal hyperparameter values and model biases. Conversely, DPO provides a closed-form solution to the policy optimization problem, producing only classification loss. Our goal with this project was to measure the inaccuracy caused by the reward model in the PPO approach, which comparing the two algorithms' performance (in terms of mean cumulative and instantaneous reward) would allow us to do. In addition, we aimed to draw conclusions on how well predefined human preferences translate to solving traditional RL challenges like a grid world. Finally, we wanted to compare the efficiency of either technique and analyze the results in relation to the quantity of data gathered.

2 Background & Related Work

In recent times, Large Language Models have had the spotlight from the general public and scientific community. Due to this, lots of research has gone into creating optimal algorithms and architectures for these applications. One of these algorithms is the Direct Preference Optimization framework, which is commonly used as a way to align LLMs with human feedback. DPO strays from the typical RLHF methodology, wherein it is a closed-form algorithm mapping human preferences to a policy instead of classical techniques that try to model a reward function. This framework has seen great success in the LLM community, however, it has not yet been expanded into other areas of reinforcement learning. We wish to adapt this technique to evaluate its efficacy in other areas of AI to utilize the algorithm to its maximum potential.

The tutorial by Eric Yang Yu [3] guides through implementing Proximal Policy Optimization (PPO) algorithm from scratch using PyTorch, covering policy network setup, advantage calculation, and gradient calculation for policy update. It emphasizes a step-by-step approach, explaining key concepts and providing code snippets to build a functional PPO implementation for RL tasks.

3 Methodology

3.1 Adapting RLHF to Gym Environments

This work presents a methodology for adapting Reinforcement Learning from Human Feedback (RLHF) to Gym environments. We achieve this by integrating synthetic feedback generation and

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

reward model instantiation techniques. We create a custom Gym minigrid environment of size 30x30, where the goal is navigation to a designated state. A reward function is defined based on the Manhattan distance between the current state s_t and the target state s_{goal}. This design incentivizes actions that move the agent closer to the goal (lower distance translates to higher reward). To incorporate human bias, we prioritize the up direction ((0, 1) in grid coordinates) during reward calculation. This is achieved by modifying the reward function as follows:

$$\texttt{reward}(s_t, a, s_{t'}) = -\left(\|s_{t'} - s_{\text{goal}}\|_1 - \alpha \cdot I(a = \text{up})\right) \tag{1}$$

where α is a hyperparameter controlling the bias strength towards the up direction. $I()$ is the indicator function, returning 1 if the condition is true (a == up) and 0 otherwise. Furthermore, to simulate human decision-making imperfections, we introduce an epsilon (ϵ) error in reward assignment. During synthetic data generation, with probability ϵ, a random action is chosen instead of the optimal one based on the biased reward function. This injects stochasticity, mimicking human tendencies to deviate from optimal choices due to uncertainty.

3.2 Reward Model Training with PyTorch

We train a two-layer Multi-Layer Perceptron (MLP) using PyTorch to learn the reward function from the synthetic dataset. The input to the MLP is the state-action pair (s_t, a), and the output is the predicted reward. The Bradley-Terry loss function is employed for optimization:

$$\texttt{loss}(\texttt{pred\_reward}, \texttt{actual\_reward}) = -\ln\left(\sigma(\texttt{pred\_reward}_i - \texttt{pred\_reward}_j)\right) \tag{2}$$

where $\sigma()$ is the sigmoid function, `pred_reward_i` and `pred_reward_j` are the model's predicted rewards for two different data points in a mini-batch. This loss function minimizes the difference between the predicted and actual rewards, enabling the model to learn the underlying relationship between state-action pairs and their associated rewards.

3.3 Motivation for PPO and DPO Comparison

Synthetic feedback and a trained reward model constitute the foundational components for evaluating the performance of PPO (Proximal Policy Optimization) and DPO (Direct Policy Optimization) agents. By integrating the learned reward function into both RL algorithms, we meticulously assess their efficacy in navigating a complex minigrid environment. The comparative analysis of PPO and DPO agents yields invaluable insights into their respective strengths and weaknesses in attaining human-aligned objectives within Gym environments. Such systematic examination contributes significantly to the advancement of RL methodologies and their practical applicability.

3.4 Direct Preference Optimization

For consistency, identical training data was used for both the reward model and DPO model, with a 95% training split. No validation data was used to ensure a fair comparison of performance. The policy model π_* is a feedforward neural network with hidden layers of 32 and 16 nodes, followed by a softmax layer. It takes only state coordinates as inputs and outputs a probability vector for the action space. The DPO training process requires a reference model π_{ref}. The original paper has a supervised fine-tuning model π_{SFT} (a language model pre-trained on data relevant to the desired task) [1]. Since we did not have access to this, the authors suggest initializing π_{ref} using preferred completions in the reference data [1], which we did, training a linear model of identical architecture to π_*. π_{ref} was also used as a benchmark for performance (see 5.2). The main policy model π_* was trained for 100 epochs using a loss function similar to the original DPO implementation [1]. This function combined "losses", a metric aiming to increase the preferences reflected in the labeled pairs of the synthetic dataset, and "rewards", a metric that punishes probabilities that stray too far from those of the reference model.

3.5 Proximal Policy Optimization

This study conducts an assessment of the efficacy of DPO through a comparative analysis with a Proximal Policy Optimization (PPO) agent. The PPO model is trained with Reinforcement Learning with Human Feedback (RLHF) to ensure equitable treatment. Conforming to established PPO methodologies outlined in the seminal work by Schulman et al. [2], the training process incorporates

standard techniques such as gradient clipping, advantage estimation, and actor rollouts. Both the actor and critic models are structured with neural networks of identical dimensions ([2, 32, 16, 4] for the actor and [2, 32, 16, 1] for the critic). Optimal hyperparameters for PPO, including actor and critic learning rates, gamma, and gradient clip epsilon, are determined empirically, considering computational limitations. Training spans 20 epochs, with 25 actor policy rollouts per epoch, each comprising a maximum of 400 steps. Gradient clipping is applied to update the actor model, while the critic model is updated using Mean Squared Error (MSE) loss with additional clipping. The assessment framework adheres to guidance provided in a referenced tutorial [3].

4 Experiments

4.1 Hyperparameters

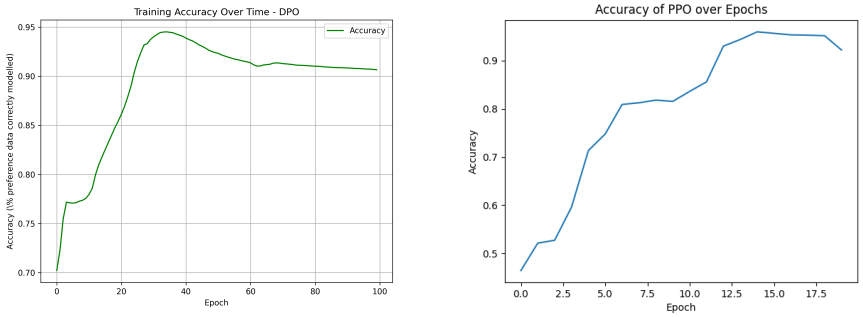
We attempted to match the original implementation's hyperparameters in π_* as closely as possible, and ended up using a learning rate $\alpha = 10^{-6}$, weighting coefficient $\beta = 0.9$, the `RMSProp` optimizer and a batch size of 64 after experimenting with several value combinations. For the PPO model, we used actor learning rate $\alpha_a = 0.005$, critic learning rate $\alpha_c = 0.3$, $\alpha_c = 0.9$, ADAM optimizer and gradient clip $\epsilon = 0.001$.

4.2 Benchmarks

Comparing our results was made easier when we knew the scale of possible performance from different methods. We developed two separate benchmarks, and averaged their rewards over 500 trajectory simulations. The first, a simple algorithm reaching the goal as quickly as possible, achieved a mean return of 172, an upper bound for performance. The second, the basic reference model from the DPO method, produced an average reward of 114.

4.3 Results

The following figures demonstrate the accuracy, loss and cumulative return achieved by both RLHF methods over the course of training. Rewards were computed by setting the current model state to evaluation at each epoch, and running 50 trajectory simulations on the test environment.



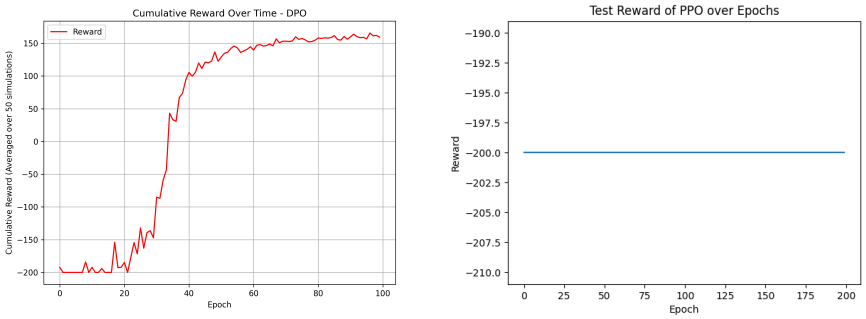
(a) DPO (b) PPO

Figure 1: Accuracy of DPO and PPO-Based Reward Modelling

The investigation aimed to assess the DPO model's performance across various metrics in the adapted grid world environment. Initially, we compared the training dynamics of DPO and PPO models. Accuracy was measured based on their ability to select the superior action from a set of hand-crafted preferences used to train the human feedback reward model. While PPO demonstrated faster initial increases in training accuracy, DPO reached a higher and more stable plateau. Specifically, around epoch 20, PPO exhibited a sharp decrease to 90% accuracy, indicating potential overfitting to biases present in the synthetic dataset. In contrast, DPO showed a smoother decline, potentially beneficial for testing scores. This behavior suggests that DPO may unlearn biases better than PPO. Moreover, DPO displayed significantly less variance in accuracy scores throughout training compared to PPO, further highlighting its reliability. Thus, despite peaking later than PPO, DPO exhibited superior training accuracy due to its steadier performance and higher plateau.

When comparing the loss functions of both models (see Appendix), we start to see the first instance of DPO truly outperforming PPO in both scores and variance. Although we cannot directly compare

numerical values for the two loss functions as they are calculated, we can compare them through interpretation. As can be seen in Figure 3, the loss function for the DPO model proceeds in a very smooth manner achieving near 0 loss at its lowest and afterwards starts to increase again. Comparing this to the sporadic loss function of the PPO model as well as a significantly higher lowest point, we can say for certain that the training dynamics for DPO run much smoother and give a more confident result in the performance/application of DPO to the environment than PPO. This means that although the PPO model is able to achieve high results, it is not confident in doing so and still requires learning to solidify its outputs. On the other hand, the DPO algorithm is much more confident in its predictions and doesn't require more tuning/training to maintain a consistent path.



(a) DPO (b) PPO

Figure 2: Cumulative Reward of DPO and PPO-Based Reward Modelling

When comparing both benchmarks, we observe that DPO converges to a near-optimal return of over 160, trailing the peak reward trajectory by only 10 (the equivalent of five suboptimal moves). It shows a clear improvement over π_{ref}'s straightforward preferred action selection (return of 114), indicating that the incorporation of preferences as well as divergence in π_* resulted in a learning process more suited to the actual test environment.

The PPO method, conversely, is not able to reach the goal state, getting stuck in certain states due to heavy preferences for suboptimal actions. Though its accuracy is high when measured on the synthetic dataset, this performance does not translate to the rewards of the actual test environment, demonstrating an adoption of the faults in the initial labeling.

5 Conclusion

5.1 Discussion

DPO appears capable of deducing the latent reward structure of the environment indirectly from preference data, even without direct environmental interaction during training. While noise and bias from labeled pairs affect both the reference model and PPO, DPO may mitigate these effects by emphasizing more reliable or consistent preferences. This phenomenon manifests in trade-offs between accuracy, loss, and enhanced reward performance during testing. Across various metrics, DPO consistently demonstrates less variance in loss and accuracy, indicative of its robustness. Moreover, its performance efficiency and ease of optimization are notable advantages, attributed to its reduced number of tunable hyperparameters.

5.2 Future Work

To enhance future analysis, it's crucial to determine if the PPO model proposed here was truly optimal. This necessitates utilizing greater computational resources, as the models were trained on standard laptops ill-suited for complex calculations. Increased computational power would facilitate exploring the optimal hyperparameter space for PPO, potentially yielding improved results. Greater confidence in the PPO model's optimality would enable a more robust comparison with DPO, affirming whether DPO genuinely outperforms PPO. Furthermore, conducting experiments with model-based human feedback methods in more intricate RL environments could offer conclusive evidence to support this project's findings. Exploring diverse problems beyond LLMs could shed light on how DPO implicitly learns an environment and its reward structure.

6 References

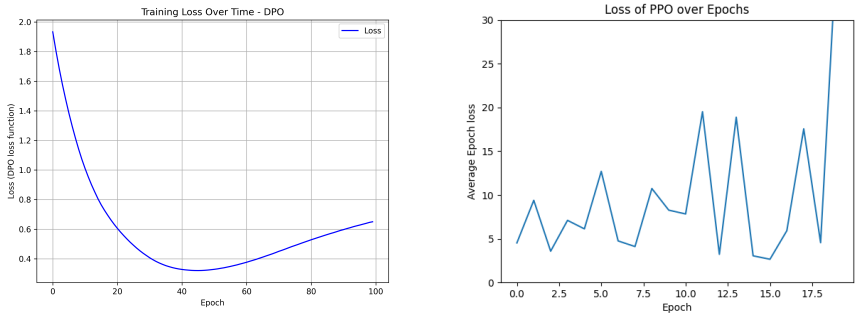
References

[1] Rafael Rafailov, Stella Biderman, Leo Hernandez-Garcia, Joel Leike, Ryan Lowe, Maximilian Schulte, and Ori Zoref. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[3] Eric Yang Yu. Ppo for beginners. https://github.com/ericyangyu/PPO-for-Beginners, 2020.

7 Appendix



(a) DPO (b) PPO

Figure 3: Loss of DPO and PPO-Based Reward Modelling