Assignment 3: LM Alignment

Instructor: Yejin Choi *CSE 517/447 Win 24*

Due at 11:59pm PT, Feb 26, 2024
100 pt for both 447 and 517, 14% towards the final grade

This assignment is designed to help you experience working with steering language models with basic RL algorithms, including REINFORCE and DPO (Direct Preference Optimization). You'll also experience forecasting capabilities of GPT-4 and write a short reflection on your findings.

You will submit both your **code** and **write-up** (as PDF) via Gradescope. Remember to **specify your collaborators** (including AI tools like ChatGPT) and **how they contribute** to the completion of your assignment at the beginning of your writeup. If you work on the assignment independently, please specify so, too. **NOT properly acknowledging your collaborators will result in -2 % of your overall score on this assignment.** Please adhere to the assignment collaboration policy specified on the course website.

Required Deliverables

- **Code Notebook**: Section §1-4 share the same Python notebook: A3.ipynb. Please download it as a Python file (`A3.py`) and submit via Gradescope upon completing the assignment.
- **Write-up**:
  - For written answers and open-ended reports, produce a single PDF for §1-5 and submit it in Gradescope. We recommend using Overleaf to typeset your answers in LaTeX, but other legible typed formats are acceptable. We do not accept hand-written solutions because grading hand-written reports is incredibly challenging.
  - The suggested page limit is to make sure the reports do not get too long. We would not penalize shorter reports as long as they contain all necessary grading components. Longer reports do not directly result in higher scores. Alternatively, concise and on-point reports will be more favorable.

Datasets to Download:

- **For §1-3**: IMDB dataset from HuggingFace.
- **For §4**: Parallel IMDB dataset from HuggingFace.

Potentially helpful resources for this assignment:

- Sentiment classifier (reward model) documentation.
- REINFORCE algorithm blog post (not LM specific).
- KL-Divergence Wikipedia page.
- The DPO paper. The DPO GitHub repo. A helpful blog from HuggingFace.

Acknowledgement

This assignment is designed by Taylor Sorensen (the original REINFORCE code & questions, forecasting GPT-4 capability questions), Skyler Hallinan (the DPO code & walk-through), Melanie Sclar (scaffolding the REINFORCE code), and Alisa Liu (generating the pairwise dataset of DPO), and Liwei Jiang (finalizing all assignment instructions & starter code & DPO walk-through), with invaluable feedback from Yejin Choi.

1 Baseline Evaluation (15%)

In this assignment, you will use a GPT-based model (DistilGPT2) to steer movie reviews to have positive sentiment. IMDB is a fairly classic toy NLP dataset, generally used for sentiment classification. You will instead treat it as an environment—given the first 10 tokens of a movie review from the IMDB training set, you want to steer the model to generate 20 tokens which complete the review to be as positive as possible, as measured by a sentiment classifier.

As an introduction, you will set up the environment and make an *evaluation* function.

Python Packages: The following set of imports should be sufficient for you to complete the coding exercise for §1-4:

```
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM,
    AutoModelForSequenceClassification
from transformers import AdamW
from datasets import load_dataset
from tqdm import tqdm
import torch.nn.functional as F
from torch.nn.functional import softmax, log_softmax, kl_div
from matplotlib import pyplot as plt
from typing import Dict, Union, List, Tuple
```

Deliverables:

1. **Code:** You will complete code blocks denoted by `TODO:` for §1 in the Python notebook.

   Load your initial LM/tokenizer (the DistilGPT2 model), the reward model/tokenizer, and the IMDB dataset train and test splits into memory. Write an "evaluate" function to goes through the entire test set and does the following for each review:

   (a) Tokenize the review.

   (b) Truncate the review to the first 10 tokens ("seed review").

   (c) Given the truncated seed review, randomly generate 20 more tokens (in HuggingFace, do sample=True) with the LM so that the final review has a length of 30 tokens.

   (d) Get the probability of the generated review being positive from the classification model. Let this be the reward.

2. **Write-up:** Your report for §1 should be **no more than one page**.

   Q1.1: Run the pretrained model on the evaluation function over the entire test set. What is the average reward?

   Hint: On a free Google Colab T4 instance, with a generation batch size of 32, this should take about 5 minutes to run.

   Q1.2: Please provide 5 example generations from the model. How do you feel about the quality of the generations? What sentiment do they express? Feel free to say so in your report if the sentences don't express clear sentiment to you.

   Q1.3: What does the `loop` variable do in the `evaluate()` function?

2 REINFORCE Implementation (20%)

In this exercise, you will implement a basic REINFORCE algorithm for steering the sentiment of language model-generated movie reviews. Specifically, you will set up the environment and make a *training* function to train a model with REINFORCE. You will use the following hyperparameters: `epochs = 1, batch size = 32, learning rate = 1e-4, optimizer = AdamW`.

Deliverables:

1. **Code:** You will complete code blocks denoted by `TODO:` for §2 in the Python notebook.

   Implement the REINFORCE algorithm. Create a *training* function. You should be able to adapt your evaluation code from §1. As a reminder, REINFORCE uses the following loss function,

   $$L(x) = -r(x) * \log p_\theta(x)$$

   where x is a sampled text from the model.

2. **Write-up:** Your report for §2 should be **no more than two pages**.

Q2.1: Run your training function over the entire training set for 1 epoch. An efficient implementation with the recommended hyperparameters above should train the model in under 10 minutes. **Plot the reward over time and report the average reward on the test set of the final trained model.** A valid solution should achieve a final average reward of at least 0.75.

Q2.2: Provide 5 example generations from the model. Are they more positive than the original generations? What else do you notice about them?

Q2.3: What are the limitations of REINFORCE based on your observations of the generated reviews? Explain what might have caused such limitations.

Q2.4: What does `reset_model_optimizer()` do?

Q2.5: What's the shape of `log_probs` in `compute_reinforce_loss()`, and what does each dimension correspond to?

3 Regularization (15%)

To maintain the fluency and other desirable properties of the pre-steering pre-trained model (e.g., instruction-following, reasoning), it's a common practice to add a KL-divergence penalty between the original model and the modified model to regularize training.

As a reminder, KL-divergence is defined as such:

$$D_{\mathrm{KL}}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Deliverables:

1. **Code:** You will modify your REINFORCE training code in the Python notebook.

   Modify your training function to add KL-regularization. Specifically, add a KL-penalty to your loss function, with P being the distribution from the modified model and Q being the distribution from the original model:

   $$L' = L + \alpha D_{\mathrm{KL}}(P||Q)$$

2. **Write-up:** Your report for §3 should be **no more than three pages**.

Q3.1: First, run your REINFORCE training function with $\alpha = 0$ for 1 epoch. This is equivalent to training without the KL-penalty. Plot the KL-divergence between the original model and the new model over time. What trend do you see from the plot? Why could such a trend be undesirable in practice?

Q3.2: Try different α to experiment with the power of KL-regularization. For each setup, report the α values, the KL-divergence plots, reward plots, 5 sample generations, and comments on the samples' quality.

- A high α that prevents the model from being able to get a reward of 0.65 or more.
- A low α that the outputs are poor quality.
- An α that gets a good reward ($\geq$0.8) and output mostly natural text.

Q3.3: Based on your experience with REINFORCE with KL-regularization, identify one of its main limitations and propose a potential solution. There's no single right answer here; any justifiable answer will result in full credit.

Q3.4: Over which tokens do we compute the REINFORCE loss, and why?

4 DPO Implementation (35%)

Recall that DPO bypasses the reward modeling step of traditional RL for NLP (e.g., PPO) and instead optimizes the policy language model directly by using pairwise preference data. As such, for this problem you will use a new dataset—a synthetically generated set of parallel IMDB reviews—each row has a prompt, and one positive and one negative sentiment completion. Together, the prompt and the completion is a complete movie review. Our goal in this problem is to again create a model that can steer reviews towards positive sentiment; we use the `positive` completions from the dataset as the preferred response, and the `negative` completion as the rejected response.

4.1 DPO Background Walk-through

DPO is a simple RL-free algorithm for training language models from paired preference data.

General DPO Pipeline: In the standard RLHF (Reinforcement Learning From Human Feedback) setup, the general pipeline of DPO comprises two steps:

1. Sample model generations $y_1, y_2 \sim \pi_{\text{ref}}(\cdot \mid x)$ for prompts x, then obtain a human *preference* over the outputs denoted $y_w \succ y_l \mid x$ where y_w and y_l are the preferred and dispreferred completion amongst (y_1, y_2), respectively. The output is an offline preference dataset $\mathcal{D} = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}$, where y_w represent the preferred model completions, and y_l represent the dispreferred model completions.
2. Optimize the language model policy π_θ by minimizing $\mathcal{L}_{\text{DPO}}$ for a given π_{ref} (base reference model/policy), $\mathcal{D}$ (preference dataset), and β (a hyperparameter controlling the deviation from π_{ref}).

Note that for this exercise, we have already created the offline preference dataset for you, and you will use the provided dataset to implement the DPO training in Step 2.

Overall DPO Loss: Recall the DPO loss,

$$\mathcal{L}_{\text{DPO}}\left(\pi_\theta; \pi_{\text{ref}}\right) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta\left(y_w \mid x\right)}{\pi_{\text{ref}}\left(y_w \mid x\right)} - \beta \log \frac{\pi_\theta\left(y_l \mid x\right)}{\pi_{\text{ref}}\left(y_l \mid x\right)}\right)\right]$$

where σ is the logistic function, $\mathbb{E}$ represents the expected value, $\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}$ represent the log of the probability ratio of the preferred completion y_w from the parameterized policy vs. from the reference policy, and $\log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}$ represents the same log ratio but for the dispreferred compeltion y_l. Intuitively, we want to increase the likelihood of the preferred completions y_w and decrease the likelihood of the dispreferred completions y_l.

Note that in the starter code that we provide, we call y_w and y_l "chosen" and "rejected" completions, respectively.

Masking the Prompts for Computing: As shown by the DPO loss, the paradigm of DPO and other RLHF methods is to optimize the policy towards a preferred *completion* given a prompt, i.e., optimize for $\pi_\theta(y_w|x)$, rather than optimize for the entire sequence, i.e., $\pi_\theta(x + y_w)$. As such, when computing the log probabilities of the preferred/dispreferred completions from the policy and reference policy model for your DPO loss, make sure to calculate the log probabilities **only of the completions**, not the entire sequence. In your implementation, you can do this by masking out tokens that belong to the prompts (using the information of the prompt length) to sub-select the log probabilities of only the tokens that belong to the completion part.

4.2 Your DPO Implementation

You recommend the following hyperparameters for your experiments: `epochs = 1`, `batch size = 16`, `learning rate = 5e-7`, `optimizer = AdamW`, `max seq length = 128`.

Deliverables:

1. **Code:** You will complete code blocks denoted by `TODO:` for §4 in the Python notebook.
2. **Write-up:** Your report for §4 should be **no more than four pages**.

Q4.1: Run your training function for 2 epochs using the batch size 16. Plot the training losses over time and report the model's average loss on the test set for the final model. Note that an efficient implementation should run in about 10 minutes.

Note that due to the small amount of data and sensitivity to hyperparameters (like learning rate and β), your loss may be noisy).

Q4.2: Run the evaluation function using the same data and code from Section 1 using the DPO-trained model. How does the reward compare to the REINFORCE model? Why do you think the results look like this?

Q4.3: Provide 5 example generations from the DPO-trained model. Are they more positive than the original generations? What else do you notice about them?

Q4.4: Try 3 different values for β (note it's typically set to be in the range of 0.1 to 0.5). Plot the training loss plots and report the final test losses for each of these setups. What's the best setup that you found? How do different values of β impact the training differently?

Q4.5: What's the purpose of using σ, the logistic function, in $\mathcal{L}_{\text{DPO}}$?

Q4.6: What are the main differences and advantages of DPO compared to other online RL methods like PPO?

5 GPT-4 Capability Forecast (15%)

Large language models are surprisingly good at some things, and surprisingly bad at others. Test and build your intuition for GPT-4's capabilities in this online quiz.

Deliverables:

1. **Writeup:** After taking the quiz, you'll answer the following questions. Your report for §5 should be **no more than two pages**.

Q5.1: Report your accuracy and log loss.

Q5.2: Write a few paragraphs about your thoughts that answer the following questions:

- Was there anything you expected GPT-4 to be able to do that it couldn't do? Or vice versa?
- What patterns do you notice in the failure/success modes? Can you guess the reasons behind these failure/success modes?
- Do you trust GPT-4 to help you with a task more or less after taking the quiz?
- Using your wildest imagination, what capabilities do you hope the future GPT-x will have? Why are they important to have? What are some prerequisites if these capabilities were to be achieved in the future GPT-x? Be creative, and there's no single correct answer!