CS6103 Human-Centered AI Jan 17, 2025

Lecture 2: Alignment Techniques: RLHF, DPO

Lecturer: Arpit Agarwal *Scribe: Muskaan Jain, Anirudh Garg*

Contents

1 Mathematical Fundamentals

Pre-requisite mathematical fundamentals used in this lecture.

1.1 Maximum Likelihood Estimation (MLE)

Given some data D that is drawn from a statistical model M, and parameterized by θ. The likelihood function is defined as:

$$\mathcal{L}(\theta) = P(D|\theta)$$

where $P(D|\theta)$ is the probability of observing the data under M_θ. This function measures how well the statistical model with parameters θ explains observed data by calculating the probability of seeing that data under different parameter values of the model.

The goal of maximum likelihood estimation is to find the values of the model parameters that maximize the likelihood function over the parameter space:

$$\text{MLE: } \hat{\theta} = \arg\max_\theta \mathcal{L}(\theta)$$

If the data $D = \{x_i\}_{i=1}^n$ has been drawn i.i.d., we can define

$$l(\theta) = \log \mathcal{L}(\theta) = \log P(D|\theta) = \log \pi_i P(x_i|\theta) = \sum_i \log P(x_i|\theta)$$

This form is easier to work with while taking derivatives as we have a summation of logs instead of a product. It is also easy to see that

$$\hat{\theta} = \arg\max_{\theta} l(\theta) = \arg\max_{\theta} \mathcal{L}(\theta)$$

because the log function is monotonically increasing.

MLE is used to estimate parameters of probabilistic models, employing techniques like gradient descent for optimization.

1.2 KL Divergence

The Kullback–Leibler (KL) divergence (also called relative entropy) is a type of statistical distance: a measure of how much a model probability distribution Q is different from a true probability distribution P. Formally, for a distribution with finite support $\mathcal{Y}$:

$$KL(P||Q) = \sum_{y \in \mathcal{Y}} P(y) \log \frac{P(y)}{Q(y)}$$

Note:

- $KL(P||Q) > 0$
- $KL(P||Q) = 0 \iff P = Q$

KL Divergence is NOT symmetric in P and Q.

Example:

If $P = \text{Bernoulli}(p), Q = \text{Bernoulli}(q)$:

$$KL(P||Q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$$

2 Learning from Pairwise Comparisons

Before discussing the general problem of LLM alignment using pairwise comparisons, we will discuss a more classical problem of learning from pairwise comparisons. Here, there are n possible alternatives $\{y_1, y_2, ..., y_n\}$ and we have results of pairwise comparisons between these alternatives. In the i^{th} pair, the winner is y_w^i, and the loser is y_l^i.

Notation: $y_w \succ y_l$ means that alternative y_w is preferred to y_l.

Let us denote our data as:

$$D = \{(y_w^i, y_l^i)\}_{i=1}^m$$

In order to model the comparison data, we will use the probabilistic Bradley-Terry-Luce (BTL) model. Under the BTL model, each alternative y is associated with a reward $r_y^* \in \mathbb{R}$. Let us denote

$$r^* = (r_1^*, r_2^*, ..., r_n^*) \in \mathbb{R}^n$$

We assume that the data is drawn from the BTL model with rewards r^*. More precisely, the probability of observing $y_w \succ y_l$ is given by

$$P(y_w \succ y_l | r^*) = \frac{e^{r_{y_w}^*}}{e^{r_{y_w}^*} + e^{r_{y_l}^*}}$$

Given the data D our goal is to estimate r^*. An obvious approach for this is to perform maximum likelihood estimation (MLE):

$$\hat{r} = \arg\max_r \sum_{(y_w, y_l) \in D} \log P(y_w \succ y_l | r) \qquad \text{s.t.} \sum_{i=1}^{n} r_i = 0$$

$$= \arg\max_r \sum_{(y_w, y_l) \in D} \log \frac{e^{r_{y_w}}}{e^{r_{y_w}} + e^{r_{y_l}}} \qquad \text{s.t.} \sum_{i=1}^{n} r_i = 0$$

$$= \arg\max_r \sum_{(y_w, y_l) \in D} \log \frac{1}{1 + e^{r_{y_l} - r_{y_w}}} \qquad \text{s.t.} \sum_{i=1}^{n} r_i = 0$$

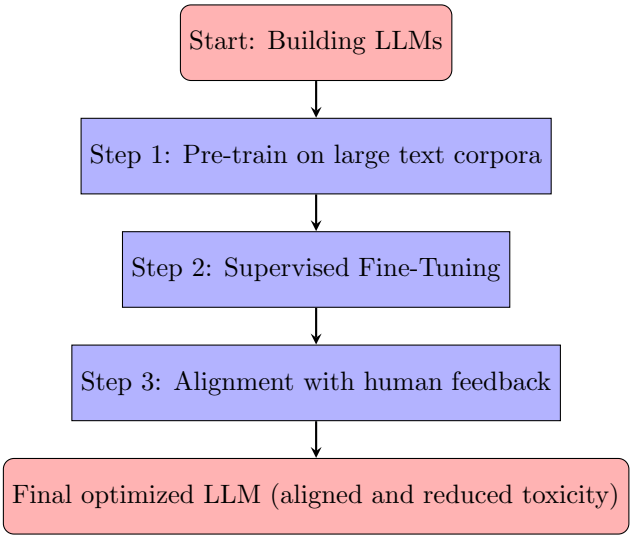
Note that the condition $\sum_{i=1}^{n} r_i = 0$ is imposed to ensure uniqueness as the model is invariant to shifts in rewards. This expression has no closed-form maximization, so we perform gradient ascent to find $\hat{r}$.

Note that this is possible as $\log \frac{1}{1+e^{r_{y_l} - r_{y_w}}}$, better known as the sigmoid function is convex in r.

Beyond MLE, spectral methods have also been proposed for learning the parameters of the BTL model [Negahban et al., 2017]. One can also provide theoretical bounds on the size of the data needed to achieve a certain accuracy for the BTL parameter estimation (beyond the scope of this course).

3 Primer on LLM Training

The following figure describes the different steps of the LLM training process. It highlights the importance of the alignment step which is required to ensure that the model is not toxic, abusive etc., and is aligned with human values.



Pre-training:
Large-scale unsupervised learning
on large text corpus
for next-word prediction
Objective: Predict next words.
Supervised Fine-tuning (SFT):
Supervised learning using gold-standard
annotated prompts and responses for
domain or task-specific optimization.
Still has toxic and harmful data.
Alignment:
Optimize using pairwise crowdsourced
or human-annotated data to align
with human values and reduce biases.
Techniques: RLHF, DPO

4 Collecting Pairwise Feedback for LLM Alignment

As we discussed in the last lecture, a common way to utilize human experts is to ask for pairwise comparisons between different alternatives. In the context of LLMs, this means asking for pairwise

comparisons between different responses give a prompt. Given the dataset of such pairwise comparisons over responses, the goal is to align the model so that it is more "agreeable" with human preferences.

In the following we will outline the process of collecting the pairwise preference dataset and then performing alignment using the dataset. We first introduce an abstract definition of an LLM policy that outputs responses given a prompt.

4.1 Defining the Policy π

The policy π is defined as a function that maps a given prompt x ($x \in \mathcal{X}$) to a probability distribution over all possible responses in the response space $\mathcal{Y}$. Formally,

$$\pi : \mathcal{X} \to \Delta(\mathcal{Y}),$$

where:

- All $y^i \in \mathcal{Y}$ belong to the response space $\mathcal{Y}$,
- All $x^i \in \mathcal{X}$ belong to the prompt space $\mathcal{X}$.

The policy π essentially provides a mechanism to generate responses for a given prompt x. Specifically, for each prompt $x \in \mathcal{X}$, the policy assigns a probability distribution $\pi(\cdot \mid x) \in \Delta(\mathcal{Y})$ over all possible responses in $\mathcal{Y}$.
Note: The symbol $\cdot$ in the notation $\pi(\cdot \mid x)$ acts as a placeholder for any element in the response space $\mathcal{Y}$. It signifies that the policy assigns probabilities to all potential responses, conditioned on the given prompt x.
Note: It is generally easy to sample from the probability distribution $\pi(\cdot \mid x)$ given the prompt x, but it is not very easy to calculate the entire probability distribution.

4.2 Sampling Responses and Human Feedback

After completing **Step 2** in the LLM training process, we acquire a policy denoted as π^{SFT}, representing the fine-tuned model.

We are give a set of m prompts $\{x^1, \cdots x^m\}$ which could be either generated by another LLM or extracted/hand-picked from past human interactions. For all prompts $x^i \in \mathcal{X}$, the responses y_w^i (winning response) and y_l^i (losing response) are sampled from the probability distribution $\pi^{\text{SFT}}(\cdot \mid x)$ as follows:

$$y_w^i, y_l^i \sim \pi^{\text{SFT}}(\cdot \mid x).$$

Once two responses have been generated using $\pi^{\text{SFT}}(\cdot \mid x)$, humans are tasked with labeling the better response in the pair. The final winner/loser can be decided by an aggregation of the votes of multiple humans or a single expert depending on the design of the crowdsourcing platform. This feedback is then used for further alignment and improvement of the policy.

4.3 Final Dataset

After the feedback collection step, we have a dataset D of the following form:

$$D = \{(x^i, y_w^i, y_l^i) \mid i = 1, \ldots, m\},$$

where:

- x^i denotes the prompt,
- y_w^i is the winning response of the pairwise comparison, and
- y_l^i is the losing response of the same comparison.

5 LLM Alignment Methods

Note: In this section we will denote the old policy as π^{ref} instead of π^{SFT} to allow for a more general exposition.

Given the dataset D and the reference policy π^{ref} our goal is to arrive at a policy π that improves upon π^{ref} by incorporating human preferences.

$$\pi^{\text{ref}} \to \pi$$

This process involves refining the policy so that it aligns better with human values and preferences. We will study two techniques for achieving this alignment, RLHF and DPO.

5.1 Assumptions

The alignment methods that we will discuss rely on two key assumptions about how the pairwise preference data is generated. Recent work in this area has focused on how to remove these assumptions but this is outside the scope of this lecture.

5.1.1 Existence of a Reward Function r^*

There exists a function $r^*(x, y)$ that assigns a real-valued reward to each response y given input x:

$$r^* : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}.$$

This reward function helps in ranking responses.

5.1.2 Bradley-Terry-Luce (BTL) Model

The probability of preferring y_1 over y_2, given x, follows:

$$P(y_1 \succ y_2 \mid x; r^*) = \frac{e^{r^*(x,y_1)}}{e^{r^*(x,y_1)} + e^{r^*(x,y_2)}}.$$

This is a logistic model for ranking responses. The reward function r^* is usually represented using a neural network whose parameters are learned.

5.2 Reinforcement Learning from Human Feedback (RLHF)

RLHF involves two steps:

- **Step 1: Maximum Likelihood Estimation (MLE) to get the reward function r^*:** We first use MLE to estimate the reward function r^* that measures the desirability of the response y given a prompt x.
- **Step 2: Use Reinforcement Learning to obtain the policy π:** We use RL to align the policy π^{ref} with human preferences.

5.2.1 Maximum Likelihood Estimation (MLE)

We use Maximum Likelihood Estimation (MLE), similar to what we did in Section 2. Under the BTL model with reward function r, the probability of the winning response y_w being greater than the losing response y_l given the prompt x as:

$$P(y_w \succ y_l \mid x; r) = \frac{e^{r(x,y_w)}}{e^{r(x,y_w)} + e^{r(x,y_l)}}\,.$$

We then estimate the reward function r^* by maximizing the likelihood of the product of these probabilities over all data points:

$$\hat{r} = \arg\max_r \sum_{i=1}^{m} \log P(y_w^i \succ y_l^i \mid x^i; r)\,.$$

In practice, the reward model is parameterized by the weights θ of the neural network that is used to represent the reward model, and these parameters are learned using the MLE.

5.2.2 RL: Updating π^{ref} using Policy Gradient Methods

Given a reward model r that is learned in the previous step, we define our objective as maximizing the following:

$$\mathcal{J}(\pi_\phi) = \sum_{i=1}^{m} \left(\sum_{y \in \mathcal{Y}} \pi_\phi(y \mid x^i) \cdot r(x^i, y) \right) - \beta\, \mathrm{KL}(\pi_\phi(\cdot \mid x) \,||\, \pi^{\text{ref}}(\cdot \mid x)),$$

$$\pi = \arg\max_\phi \mathcal{J}(\pi_\phi)$$

where:

- The first term represents the expected reward under the policy π_ϕ, where the policy assigns a probability $\pi_\phi(y \mid x^i)$ to response y and the reward $r^*(x^i, y)$ is given by the reward function.
- The second term is the Kullback-Leibler (KL) divergence between the new policy π_ϕ and the reference policy π^{ref}. This term serves two important purposes:
  - **Reason 1: Preventing the policy from overfitting to the preferred response.** The first term of the objective function is linear, which means that to maximize the objective, the new policy π_ϕ would likely assign a very high probability (close to 1) to the preferred response and ignore all other responses. To prevent this, we introduce the KL divergence as a regularizer. The KL term penalizes large deviations from π^{ref}, ensuring that the new policy does not overfit the preferred responses and keeps a more balanced distribution.
  - **Reason 2: Preserving the learned reward model.** The reward model r^* was learned based on the domain of π^{ref}, so we do not want the new policy π_ϕ to deviate too much from the domain of π^{ref}, where the reward function is well-defined. If π_ϕ moves too far from π^{ref}, the reward model might become arbitrary, and the learned rewards may no longer align with the intended preferences. Thus, the KL divergence helps preserve the knowledge embedded in π^{ref} and ensures that the policy stays within the domain where the reward model was valid.
- β is a hyperparameter that controls the trade-off between maximizing the expected reward (first term) and minimizing the KL divergence (second term). This hyperparameter must be tuned based on the dataset to find the optimal balance.

To optimize the objective function $\mathcal{J}(\pi_\phi)$, we perform gradient ascent. However, due to the complex nature of the response space $\mathcal{Y}$,[1] we cannot simply apply standard gradient ascent techniques. This necessitates the use of specific reinforcement learning (RL) techniques, such as policy gradient methods (PPO), which allow us to efficiently compute gradients and update the policy despite the complex structure of the response space.

5.3 Direct Preference Optimization (DPO)

Very recently, Rafailov et al. [2023] introduced DPO which aims to simplify the two-step procedure of reward learning followed by policy update. DPO observes that the reward model has one-to-one correspondence with the optimal policy, and it is possible to combine the two-steps of RLHF into a single step.[2]

We now describe the DPO procedure. First, converting the objective to a minimization problem,

$$\arg\max_{\pi} J(\pi) = \arg\min_{\pi} \sum_{i=1}^{m} \left(-\sum_{y \in \mathcal{Y}} \pi(y|x^i) \cdot r(x^i, y) + \beta\, KL\big(\pi(\cdot|x^i) \parallel \pi_{\text{ref}}(\cdot|x^i)\big) \right)$$

Summation over x^i can be replaced by an expectation value over x over the observed dataset $\mathcal{D}$, and the constant factor N can be dropped. Similarly, for y, we can replace the summation over $y \in \mathcal{Y}$ by the corresponding expectation value.

$$= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}} \mathbb{E}_{y\sim\pi(\cdot|x)} \left[-\frac{r(x,y)}{\beta} + \log \frac{\pi(y|x)}{\pi^{\text{ref}}(y|x)} \right]$$

Writing $-\frac{r(x,y)}{\beta}$ as $\log e^{-\frac{r(x,y)}{\beta}}$,

$$= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}} \mathbb{E}_{y\sim\pi(\cdot|x)} \left[\log e^{-\frac{r(x,y)}{\beta}} + \log \frac{\pi(y|x)}{\pi^{\text{ref}}(y|x)} \right]$$

And combining the terms into a single logarithm,

$$= \min_{\pi_\phi} \mathbb{E}_{x\sim\mathcal{D}} \mathbb{E}_{y\sim\pi_\phi(\cdot|x)} \left[\log \frac{\pi_\phi(y|x)}{exp(\frac{r(x,y)}{\beta}) \cdot \pi^{\text{ref}}(y|x)} \right]$$

We can write

$$\pi^*(y|x) = \frac{1}{Z(x)} \exp\left(\frac{r(x,y)}{\beta}\right) \cdot \pi^{\text{ref}}(y|x)$$

Where $Z(x)$ is the normalizing factor,

$$Z(x) = \sum_{y} \exp\left(\frac{r(x,y)}{\beta}\right) \cdot \pi^{\text{ref}}(y|x)$$

This gives us,

$$\max_{\pi_\phi} J(\pi) = \min_{\pi_\phi} \mathbb{E}_{x\sim\mathcal{D}} \mathbb{E}_{y\sim\pi_\phi(\cdot|x)} \left[\log \frac{\pi_\phi(y|x)}{\pi^*(y|x)} - \log Z(x) \right]$$

$\log Z(x)$ is constant with respect to π_Φ, so it can be dropped, and replacing the logarithm with the KL term, we get

$$= \min_{\pi_\phi} \mathbb{E}_{x\sim\mathcal{D}} KL\big(\pi_\phi(\cdot|x) \parallel \pi^*(\cdot|x)\big)$$

Therefore,

$$\pi^* = \arg\min J(\pi_\phi)$$

[1] Recall that LLMs are trained using next-token prediction, and the entire conditional $\pi(\cdot \mid x)$ is hard to calculate explicitly.

[2] Note that DPO is theoretically equivalent to RLHF under the BTL model, but the implementation details differ.

References

Sahand Negahban, Sewoong Oh, and Devavrat Shah. Rank centrality: Ranking from pairwise comparisons. *Oper. Res.*, 65(1):266–287, 2017. doi: 10.1287/OPRE.2016.1534. URL https://doi.org/10.1287/opre.2016.1534.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.