Preference Learning (Or, How to Build AGI)

Jeremy Goldwasser

March 18, 2025

In essence, training an LLM like ChatGPT boils down to 3 phases: Pre-training, supervised fine-tuning, and preference learning. This blog post provides a concise, well-written overview. In this document, we'll overview these three phases, going into depth on the two major frameworks for preference learning: RLHF and DPO.

1 Pre-Training

The first phase of training an LLM is to pre-train it on a ton of data. As you may be familiar, language models train only to predict the probability of the next token. A token can be thought of as the essential subwords that constitute a word. For example, the word "jumping" may be represented with the tokens "jump" and "-ing."

Why next-token prediction? Doing so provides a neat formulation to learn a probabilistic model for language. Say a document contains N tokens $x_1, \dots, x_N$. By the definition of conditional probability, the joint probability of the document may be factored as such:

$$\begin{aligned}\mathbb{P}(x_1, \dots, x_N) &= \textstyle\prod_{t=1}^{N} \mathbb{P}(x_t | x_{s<t}) \\ &= \mathbb{P}(x_1) \times \mathbb{P}(x_2|x_1) \times \mathbb{P}(x_3|x_2, x_1) \times \dots .\end{aligned} \tag{1}$$

Conceived this way, we see the LLM is actually a density model. Through autoregressive decoding, it can compute the probability of any string of tokens.

How much pre-training data is used? A staggering amount. Perusing through Wikipedia's list of LLMs, the largest public models like LLaMa 3 are trained on around 15 trillion tokens. This size has ballooned in recent years. GPT-3, arguably the first LLM, was trained on 300 billion tokens, 10 times bigger than its predecessor. The LLMs that captured the public's attention in 2022 used around 1-2 trillion.

For context, the entirety of English Wikipedia — 7 million articles — contains 5 billion words. LLMs are trained on multiple langauges at once, so the first LLaMa used about 25 billion tokens from Wikipedia. If it sounds like we're running out of data, it's because we are. Massive public datasets like CommonCrawl (1.2 trillion tokens) exist, but there aren't many.

As a result, companies that train LLMs have turned to proprietary data. This may entail licensing datasets from other companies. Reddit and Stack-Overflow, for example, have changed their privacy policies to prohibit companies from freely scraping their text to train LLMs. And, of course, a lot of data is obtained through sketchier means. A host of lawsuits from writers and news organizations argue their proprietary work has been used to train LLMs without their knowledge, consent, or remuneration.

The models themselves are enormous. GPT 3 and 4o are around 200 billion parameters; GPT-4 is rumored to be an order of magnitude larger. Even DeepSeek used over 600 billion parameters for their V3 model, famous for its frugal training cost of *only* \$6 million dollars. The staggering cost to train LLMs bodes poorly for academic LLM research.

2 Supervised Fine-Tuning

Once an LLM has been pre-trained, the next step is to fine-tune it on supervised datasets. These take the form (x, y), where x is a prompt and y is a response. InstructGPT and later LLMs train a single model on a diverse range of tasks. To name a few: Question answering, document summarization, machine translation, open-ended generation, rewriting, and so on (Table 1).

Use-case	**Prompt**
Brainstorming	List five ideas for how to regain enthusiasm for my career.
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.

Rewrite	This is the summary of a Broadway play:
	`"""`
	`{summary}`
	`"""`
	This is the outline of the commercial for that play:
	`"""`

Table 1: Prompts for various use-cases.

In the pre-LLM era, models would be trained directly on these datasets. However, pre-training helps greatly, since it instills a deep sense of linguistic fluency, background knowledge, and perhaps some degree of reasoning ability. While large in objective terms, the datasets used for fine-tuning pale in comparison. Pretraining data accounted for 98% of the text used to train InstructGPT.

Rather than retraining all of the weights of the network, it is common practice to only perform low-rank updates. Let W be a matrix of weights and biases at some layer of the network. After fine-tuning, the learned weights will be some $W' = W + \Delta$. The intuition of fine-tuning is that the same general capabilities should remain, but be adapted for some specific task. Therefore constrain Δ as a low-rank matrix product AB, fix W, and learn A and B. This technique is called Low-Rank Approximation, or LoRA.

In the language of RL – which we're about to enter – supervised fine-tuning is referred to as *behavioral cloning*, wherein a model trains to mimic the behavior of expert labels. But this is just supervised learning. Indeed, the connection of RLHF to RL is tenuous at best.

3 Preference Learning

So far, everything we've seen is fairly standard. Fine-tuning a model that has been pre-trained on vast unsupervised data is a well-known machine learning approach. In contrast, incorporating preference data is the major breakthrough for LLMs. This initially took the form of Reinforcement Learning from Human Feedback, or RLHF (Ouyang et al., 2022). More recently, Direct Preference Optimization (DPO) has emerged as a promising alternative.

In the following subsections, we first introduce preference learning from a statistical perspective. Then, we demonstrate how RLHF explicitly models preferences and uses them to optimize the LLM. Finally, we show how DPO circumvents the need for a reward model, and thus for RL.

3.1 Bradley-Terry Model

Modeling preferences has a genuine history in the statistics literature. The Bradley-Terry model models pairwise comparisons between items (Bradley and Terry, 1952). Pairwise data shows up in a variety of contexts. For example, there are 30 NBA teams and 82 games in a season. The Bradley-Terry model provides a inferential framework to evaluate the relative strength of each pair of teams, based on the pairwise outcomes of their games.

In this model, the event that item i is preferred over j is expressed as a Bernoulli random variable with success probability $\mathbb{P}(i \succ j)$. To obtain this probability, the model associates a positive real-valued score s_i with each item, where

$$\mathbb{P}(i \succ j) = \frac{s_i}{s_i + s_j}.$$

Bradley and Terry proposed using exponential scores. That is, we define $s_i = e^{\beta_i}$, and learn all β_i. Framed this way,

$$\mathbb{P}(i \succ j) = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j}}.$$

Taking a logit transform reveals the interpretation as $\beta_i - \beta_j$ being the log odds for this Bernoulli random variable:

$$\begin{aligned}\text{logit}\left(\mathbb{P}(i \succ j)\right) = \sigma^{-1}(\mathbb{P}(i \succ j)) &= \log\left(\frac{\mathbb{P}(i \succ j)}{\mathbb{P}(j \succ i)}\right) \\ &= \log\left(\frac{e^{\beta_i}}{e^{\beta_j}}\right) = \beta_i - \beta_j.\end{aligned}$$

Equivalently, $\mathbb{P}(i \succ j) = \sigma(\beta_i - \beta_j)$, for the logistic function $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$. (This is reminiscent of logistic regression, in which a binary output Y is modeled as $\mathbb{P}(Y = 1) = \sigma(\theta^\top x)$.)

The scores s_i and s_j may be learned via maximum likelihood estimation. For example, suppose preference rankings are observed within a set of n items. Using exponential scores and letting n_{ij} denote the number of times i is preferred over j, the MLE is

$$\begin{aligned}\hat{p}^{MLE} &= \underset{p_1,\dots p_n \geq 0}{\text{argmin}} \ -\mathcal{L}(p) \\ &= \underset{p \in \mathbb{R}^n_+}{\text{argmin}} - \log\left[\Pi_{i \neq j} \mathbb{P}(i \succ j)^{n_{ij}}\right] \\ &= \underset{p \in \mathbb{R}^n_+}{\text{argmin}} \sum_{i=1} \sum_{j=1} - \log\left[\sigma(\beta_i - \beta_j)^{n_{ij}}\right] \\ &= \underset{p \in \mathbb{R}^n_+}{\text{argmin}} \sum_{i=1} \sum_{j=1} n_{ij} \log\left[1 + e^{-(\beta_i - \beta_j)}\right] \\ &= \underset{p \in \mathbb{R}^n_+}{\text{argmin}} \sum_{i=1} \sum_{j=1} n_{ij} \left(\beta_i - \log\left[e^{\beta_i} + e^{\beta_j}\right]\right)\end{aligned}$$

3.2 RLHF

For brief "historical" context, RLHF itself was first proposed in Christiano et al. (2017), when OpenAI was primarily focused on robotics, not LLMs. (Imagine that.) Subsequent works extended it to the language context, albeit on specific tasks like summarization (Ziegler et al., 2020; Stiennon et al., 2020). InstructGPT was the first work that used preference learning for general-purpose LLMs (Ouyang et al., 2022). ChatGPT was released later that year, using the approach it publicized.

3.2.1 Reward Model

In the context of LLMs, preference datasets rank different responses to a prompt. Formally, they contain tuples of the form (x, y_w, y_ℓ). Here, x is a natural language prompt. y_w and y_ℓ are two responses, whose subscripts indicates the annotator's winning (w) and losing (ℓ) preferences. The first Claude model was trained on roughly a million such comparisons.

The essential idea of RLHF is use this data to learn a *reward model* $r_\phi(x, y)$ that scores a response y to a prompt x. Having trained this model, an RL algorithm then modifies the LLM to produce responses that maximize the reward.

How is the reward model trained? Via the maximum likelihood of a Bradley-Terry model. In scoring both responses to a prompt, it aims to assign high probability that the winning response was preferred. Recall that the Bradley-Terry model defines the probability that item i is preferred as $\sigma(\beta_i - \beta_j)$. In this context, the likelihood of the observed data is

$$\mathbb{P}(\text{Response } w \text{ preferred to } \ell|x) = \sigma(r_\phi(x, y_w) - r_\phi(x, y_\ell)).$$

The reward model, then, is trained to minimize the negative log-likelihood of the whole preference dataset $\mathcal{D} = \{(x^i, y_w^i, y_\ell^i)\}$:

$$\mathcal{L}_r(\theta|\mathcal{D}) = -\mathbb{E}_{(x,y_w,y_\ell)\sim\mathcal{D}} \log\left[\sigma(r_\phi(x, y_w) - r_\phi(x, y_\ell))\right]. \qquad (2)$$

3.2.2 RL

Once the reward model has been trained on human preferences, the next step is to update the fine-tuned LLM to generate responses aligned to it. This is where reinforcement learning allegedly occurs. RL in the broadest sense is choosing an action from some state based on a reward signal. (Ben Recht refers to this broad categorization as “RL Maximalism.") In RLHF, the state is the prompt; the policy is the LLM; the action is its generated response; and the reward is the plug-in score $r(x, y)$ learned from the preference data. As before, the updates to the LLM may occur via LoRA, not retraining all parameters.

Is this really RL? Debatable. Normally, RL is associated with sequential decision-making – robots playing ping-pong and the like. While we might be tempted to think that iteratively decoding tokens renders this sequential, there actually is just one step: The entire response generated. There is no notion of dynamics for an environment. Some people call one-step RL algorithms *contextual bandit*s.

The reward model isn't the only thing that gets optimized. Because the preference dataset is small in relative terms, optimizing only to the reward model runs the risk of overfitting. To mitigate this, a regularization term is added, which ensures the LLM doesn't stray too far away.

Formally, let π_θ be the LLM we optimize, and π_{ref} a reference LLM - namely, the pre-trained LLM before fine-tuning. Each LLM may compute the probability of decoding a sequence of tokens y in response to prompt x: By Eq. (1), $\pi(y|x)$ is the product of all next-token probabilities in y. To avoid π_θ prioritizing the reward samples at the expense of its prior linguistic capabilities, the KL divergence between $\pi_\theta(y|x)$ and $\pi_{\text{ref}}(y|x)$ is subtracted from the reward.

$$R(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}[r_\phi(x, y)] - \beta\mathbb{D}_{KL}(\pi_\theta(y|x)\|\pi_{\text{ref}}(y|x)). \qquad (3)$$

where $\mathbb{D}_{KL}(P\|Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right)$.

Typically, the objective (3) is optimized by Proximal Policy Optimization (Schulman et al., 2017). PPO was developed by OpenAI for general-purpose policy learning, not necessarily for RLHF. Nevertheless, it worked very well, and since 2017 has been their RL method of choice.

I won't go too heavy into the details of PPO. Essentially, it improves upon a prior policy learning algorithm, TRPO. That method addressed the unstable training and high sample inefficiency that plagues RL. It ensured smooth updates by constraining the KL Divergence on the Hessian matrix of the policy networks' parameters.

Computing the Hessian is computationally expensive, and in general TRPO is complex to implement. PPO circumvents these challenges by using a different, Hessian-free objective function. This clips the ratio between each action's probability ratios between the new and updated policy. As a result, the next policy has to be "proximal" to the previous one: It won't be encouraged to make a big step in certain directions.

3.3 DPO

Direct Preference Optimization (DPO) bolsters the claim that RLHF isn't really RL. It obviates the need for a reward model or reinforcement learning at all. Rather, it performs supervised learning, drawing a straightforward connection to RLHF.

Core to DPO is the following lemma.

Lemma 1. *Given a reward function $r(\cdot, \cdot)$, the RLHF objective in equation* (3) *is optimized by*

$$\pi_r(x, y) = \frac{1}{Z(x)} \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right),$$

with partition function $Z(x) = \sum_y \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$.

Proof. We apply the definition of KL divergence, turn the problem into a minimization, and apply the definition of $Z(x)$.

$$\begin{aligned}
&\max_\pi \mathbb{E}_{x\sim D,\, y\sim\pi}\Big[r(x, y)\Big] - \beta\, \mathbb{D}_{\mathrm{KL}}\Big(\pi(y|x) \,\|\, \pi_{\mathrm{ref}}(y|x)\Big) \\
&\quad = \max_\pi\ \mathbb{E}_{x\sim D}\, \mathbb{E}_{y\sim\pi(\cdot|x)} \left[r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right] \\
&\quad = \min_\pi\ \mathbb{E}_{x\sim D}\, \mathbb{E}_{y\sim\pi(\cdot|x)} \left[\log \frac{\pi(y|x)}{\pi_{\mathrm{ref}}(y|x)} - \frac{1}{\beta} r(x, y)\right] \\
&\quad = \min_\pi\ \mathbb{E}_{x\sim D}\, \mathbb{E}_{y\sim\pi(\cdot|x)} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\mathrm{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)} - \log Z(x)\right].
\end{aligned} \tag{4}$$

Define the denominator in the first term of (4) as $\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)$. Note this is a proper probability density, as it is non-negative and sums to 1. Putting it in, we re-organize the objective as

$$\begin{aligned}
&\min_{\pi}\ \mathbb{E}_{x\sim D}\left[\mathbb{E}_{y\sim\pi(y|x)}\log\left(\frac{\pi(y|x)}{\pi^*(y|x)}\right) - \log Z(x)\right] \\
=&\min_{\pi}\ \mathbb{E}_{x\sim D}\left[\mathbb{D}_{\text{KL}}\Big(\pi(y|x)\parallel\pi^*(y|x)\Big) - \log Z(x)\right] \qquad (13)
\end{aligned}$$

Gibbs inequality tells us the KL divergence is 0 if and only if the two probability distributions are identical. $Z(x)$ does not depend on π, so the objective is optimized at $\pi = \pi^*$. □

Having established Lemma 1, we can rearrange it to express the reward model in terms of the optimal policy.

Corollary 1. *Taking logs and rearranging Lemma 1 yields*

$$r(x,y) = \beta\log\frac{\pi_r(y|x)}{\pi_{ref}(y|x)} + \beta\log Z(x).$$

Lemma 1 and Corolary 1 define the optimal policy for any reward function r. This includes the ground-truth reward function r^* and its corresponding policy π^*.

The partition function $Z(x)$ is computationally intractable, as it would require computing the probability of all generated responses. Fortunately, we will not need to worry about it in the DPO objective.

We trained the reward model to minimize the equation 2. This maximizes the likelihood of the observed preferences $\{(x^i, y_w^i, y_\ell^i)\}$. For reward model $r(\cdot,\cdot)$, each preference probability under the Bradley-Terry model takes the form

$$\begin{aligned}
\mathbb{P}^r(y_w \succ y_\ell|x) &= \sigma(r(x,y_w) - r(x,y_\ell)) \\
&= \frac{1}{1+e^{-(r(x,y_w)-r(x,y_\ell))}} \\
&= \frac{1}{1+e^{r(x,y_\ell)-r(x,y_w)}}.
\end{aligned}$$

Using the ground-truth reward r^*, we apply corollary (1) expressing the reward in terms of its optimal policy. The $Z(x)$ terms cancel out.

$$\begin{aligned}
\mathbb{P}^*(y_w \succ y_\ell \mid x) &= \frac{1}{1+\exp\left(\beta\log\frac{\pi^*(y_\ell|x)}{\pi_{\text{ref}}(y_\ell|x)} - \beta\log\frac{\pi^*(y_w|x)}{\pi_{\text{ref}}(y_w|x)}\right)} \\
&= \sigma\left(\beta\log\frac{\pi^*(y_w\mid x)}{\pi_{\text{ref}}(y_w\mid x)} - \beta\log\frac{\pi^*(y_\ell\mid x)}{\pi_{\text{ref}}(y_\ell\mid x)}\right). \qquad (5)
\end{aligned}$$

This final equation (5) is the likelihood in the DPO objective. DPO aims to learn the optimal policy π^* that underlies this probabilistic model. It minimizes the negative log-likelihood that the winning responses are preferred.

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]. \tag{6}$$

Note the reward model is implicit. Having approximated π^*, the rewards $r^*(x, y)$ in Corollary 1 cannot be obtained due to their reliance on $Z(x)$; however, differences in rewards can be calculated, since the intractable terms cancel out. Indeed, the subtitle of the DPO paper is, *Your Language Model Is Secretly a Reward Model.*

3.4 RLHF vs DPO

Clearly, DPO is a simpler alternative to RLHF. There is no need to train a separate reward model, which may be cumbersome. More importantly, it optimizes the LLM based on supervised learning rather than RL. While PPO is a strong algorithm, reinforcement learning is notoriously challenging to implement effectively.

For this reason, DPO has swiftly emerged as a leading way to align LLMs with human preferences. Where efficiency is concerned, it has overtaken RLHF as the go-to way to train LLMs. Even Meta's LLaMa 3 was trained with DPO. The DPO paper's experiments are all competitive with RLHF.

Does this spell the end of RL? Not necessarily. Major players like OpenAI and DeepSeek have not switched to DPO, as there are a number of advantages to having a distinct reward model. One such advantage is flexibility. RLHF's modular system naturally facilitates expressing different kinds of preferences. One could have different sets of preferences for helpfulness, harmlessness, and factuality. DeepSeek also used rule-based rewards based on form and logical reasoning. Then, the final reward would take an average of these specific rewards, using some set of weights. Furthermore, these reward weights can be easily adjusted to changing priorities, enabling greater flexibility over model tuning.

References

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345. JSTOR 2334029.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. (2020). Learning to summarize with human feedback. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. (2020). Fine-tuning language models from human preferences.