Stat 9911
Principles of AI: LLMs
Training LLMs

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

February 12, 2025

Plan

- We plan to discuss training LLMs: pre- and post-training, supervised fine-tuning, etc.

Loss Minimization

- Overarching principle: Loss minimization.
- Given a class $f_w, w \in \mathcal{W}$ of models, run algorithm aiming to solve:

$$\min_{w \in \mathcal{W}} L(f_w),$$

where L is a loss function that depends on data.

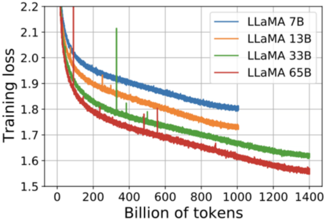


Figure 1: **Training loss over train tokens for the 7B, 13B, 33B, and 65 models.** LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

Figure: Llama 1 learning curves (Touvron et al., 2023)

Loss Minimization: Methods

- Stochastic first-order optimization methods (SGD, Adam, AdamW) used.
- Enabled by automatic differentiation (autodiff), which computes gradients automatically via back-propagation (backprop) given a computational graph.

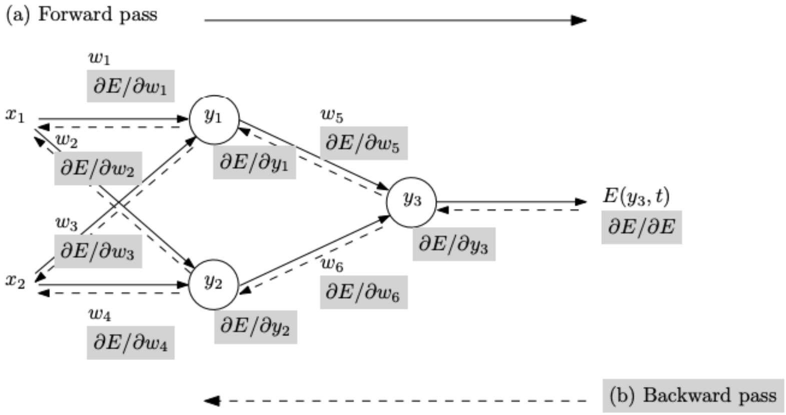


Figure: Source

Phases of Training

- Phases of training:
  - Pre-training
  - Post-training (fine-tuning, alignment).

Table of Contents

Pre-training

Post-training
- SFT
- Alignment
  - Preference Modeling
  - Using the Learned Reward
  - Direct Preference Optimization
  - GRPO

Pre-training

- Train LLMs on enormous amounts of data to maximize probability of observed text.
- MLE objective: $\min_w \sum_{x \in D} - \log p_w(x)$.
- Empirical maximizer over all distributions: empirical distribution.
- Data includes: Wikipedia, books, arXiv, Reddit, Stack Overflow, newspapers, etc.
- Next-word prediction forces the NN to learn a lot about the world.

Specific Datasets

- Common Crawl: A massive web crawl dataset. 400TB (Nov '24). "Crude and not ideal for direct use for LLM training due to artifacts arising from the conversion of HTML to plain text, sources of generally low quality". C4/RefinedWeb are cleaned versions.
- RedPajama: Open datasets with quality annotations. V1: 1 trillion tokens (CC/C4, ArXiv, GitHub, Wikipedia, Stack Exchange). V2 (Weber et al., 2024): more CC.
- TxT360, 15 T Tokens/10TB: CC (90%! of all data), papers (ArXiv, PubMed, Semantic Scholar), Wikipedia, Stack Exchange, FreeLaw, DeepMind Math, US Patent Office, Project Gutenberg-19, EuroParl
- Code: The Stack v2. 32TB code in 600+ programming languages.
- How to set and anneal data mix?

Data Cleaning

Data must be cleaned and filtered during pre- and post-training.

Text extraction and cleaning. We process the raw HTML content for non-truncated web documents to extract high-quality diverse text. To do so, we build a custom parser that extracts the HTML content and optimizes for precision in boilerplate removal and content recall. We evaluate our parser's quality in human evaluations, comparing it with popular third-party HTML parsers that optimize for article-like content, and found it to perform favorably. We carefully process HTML pages with mathematics and code content to preserve the structure of that content. We maintain the image `alt` attribute text since mathematical content is often represented as pre-rendered images where the math is also provided in the `alt` attribute. We experimentally evaluate different cleaning configurations. We find markdown is harmful to the performance of a model that is primarily trained on web data compared to plain text, so we remove all markdown markers.

De-duplication. We apply several rounds of de-duplication at the URL, document, and line level:

- **URL-level de-duplication.** We perform URL-level de-duplication across the entire dataset. We keep the most recent version for pages corresponding to each URL.
- **Document-level de-duplication.** We perform global MinHash (Broder, 1997) de-duplication across the entire dataset to remove near duplicate documents.
- **Line-level de-duplication.** We perform aggressive line-level de-duplication similar to `ccNet` (Wenzek et al., 2019). We remove lines that appeared more than 6 times in each bucket of 30M documents. Although our manual qualitative analysis showed that the line-level de-duplication removes not only leftover boilerplate from various websites such as navigation menus, cookie warnings, but also frequent high-quality text, our empirical evaluations showed strong improvements.

Figure: Llama 3 (Dubey et al., 2024)

Data Cleaning

- Remove undesirable sources to avoid learning about their topics.
- Quality filtering (e.g., excessive use of emojis, low complexity responses, etc.).

Heuristic filtering. We develop heuristics to remove additional low-quality documents, outliers, and documents with excessive repetitions. Some examples of heuristics include:

- We use duplicated n-gram coverage ratio (Rae et al., 2021) to remove lines that consist of repeated content such as logging or error messages. Those lines could be very long and unique, hence cannot be filtered by line-dedup.
- We use "dirty word" counting (Raffel et al., 2020) to filter out adult websites that are not covered by domain block lists.
- We use a token-distribution Kullback-Leibler divergence to filter out documents containing excessive numbers of outlier tokens compared to the training corpus distribution.

Model-based quality filtering. Further, we experiment with applying various model-based quality classifiers to sub-select high-quality tokens. These include using fast classifiers such as `fasttext` (Joulin et al., 2017) trained to recognize if a given text would be referenced by Wikipedia (Touvron et al., 2023a), as well as more compute-intensive Roberta-based classifiers (Liu et al., 2019a) trained on Llama 2 predictions. To train a quality classifier based on Llama 2, we create a training set of cleaned web documents, describe the quality requirements, and instruct Llama 2's chat model to determine if the documents meets these requirements. We use DistilRoberta (Sanh et al., 2019) to generate quality scores for each document for efficiency reasons. We experimentally evaluate the efficacy of various quality filtering configurations.

Figure: Llama 3 (Dubey et al., 2024)

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on

Sutton's Bitter Lesson

> In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess.
>
> A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years.
>
> Also important was the use of learning by self play to learn a value function
>
> Search and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research.

Bitter lesson is a bit too reductionist. We need many non-trivial ideas (e.g., transformers, position encoding). Just searching for techniques that can benefit from "moar compute" is not enough. Better to think of a bittersweet lesson (Felix Hill).

Sutton's Bitter Lesson for LLMs

- Current SOTA LLMs are trained with huge amounts of compute. Millions of GPU-hours
- Example 1:
  - Llama 3 405B: 16K H100 GPUs, at least 54 days.
  - If rented at 2.4 USD/hour (Nov 2024 price), costs around $50M.
- Example 2:
  - DeepSeek V3 671B: 2K H800 GPUs, 56 days.
  - Costs around $5.58M.

Infra for Training LLMs

- LLM training requires massive engineering/infra/algo effort

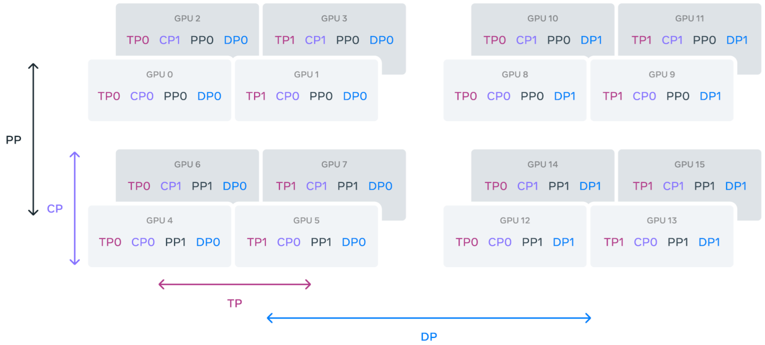


Figure 5 Illustration of 4D parallelism. GPUs are divided into parallelism groups in the order of [TP, CP, PP, DP], where DP stands for FSDP. In this example, 16 GPUs are configured with a group size of |TP|=2, |CP|=2, |PP|=2, and |DP|=2. A GPU's position in 4D parallelism is represented as a vector, $[D_1, D_2, D_3, D_4]$, where D_i is the index on the i-th parallelism dimension. In this example, GPU0[TP0, CP0, PP0, DP0] and GPU1[TP1, CP0, PP0, DP0] are in the same TP group, GPU0 and GPU2 are in the same CP group, GPU0 and GPU4 are in the same PP group, and GPU0 and GPU8 are in the same DP group.

Figure: Llama 3 parallelism

Infra for Training LLMs

Figure 5 | Example DualPipe scheduling for 8 PP ranks and 20 micro-batches in two directions. The micro-batches in the reverse direction are symmetric to those in the forward direction, so we omit their batch ID for illustration simplicity. Two cells enclosed by a shared black border have mutually overlapped computation and communication.

Figure: DeepSeek V3 scheduling

Table of Contents

Pre-training

Post-training
- SFT
- Alignment
  - Preference Modeling
  - Using the Learned Reward
  - Direct Preference Optimization
  - GRPO

Post-training

- Pre-training uses “generic" data. For any specific application, may use some additional data to fine-tune the model (Dai and Le, 2015). This can
  1. improve performance
  2. align LLMs to specific “values"/behaviors.

Components of Post-training

- InstructGPT, underlying ChatGPT (Ouyang et al., 2022): “language modeling—predicting the next token on a webpage—is different from following the user’s instructions helpfully and safely"
- Components, following Ziegler et al. (2019); Ouyang et al. (2022):
  1. Supervised fine-tuning/instruction finetuning (Dai and Le, 2015)
  2. Alignment via RLHF [special note: Ziegler et al. (2019) already has the entire “modern" RLHF pipeline fully developed]
     - 2.1 reward modeling/learning [broad area here is inverse reinforcement learning (Ng et al., 2000)]
     - 2.2 policy (=LM) optimization/RL (Christiano et al., 2017)
  3. these steps can be repeated; e.g., Llama 3 iterates six times (Dubey et al., 2024)
- See also Bai et al. (2022) and Nathan Lambert’s 2025 tutorial.

Post-training

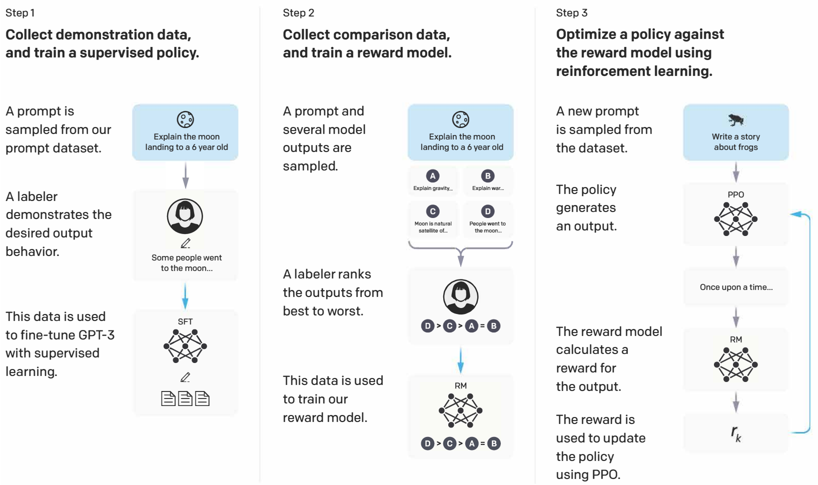
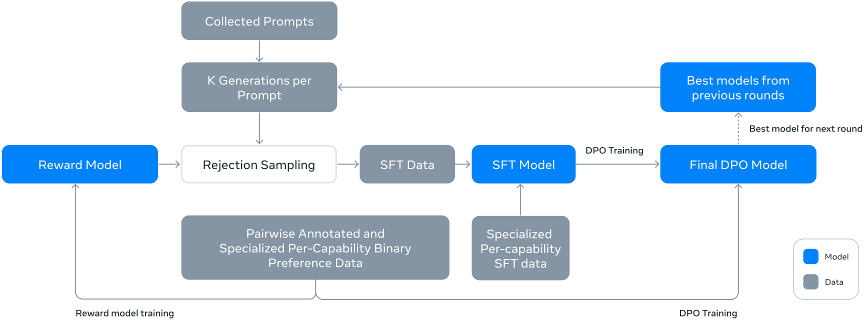


Figure: Post-training. Ouyang et al. (2022)

Llama 3 Post-training



DeepSeek-R1 Post-training

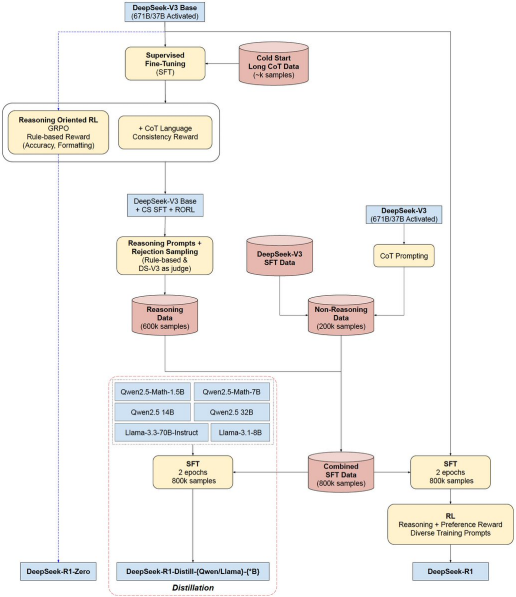


Figure: Source

Table of Contents

Pre-training

Post-training
- SFT
- Alignment
  - Preference Modeling
  - Using the Learned Reward
  - Direct Preference Optimization
  - GRPO

Supervised Fine-Tuning

- After pre-training, the model approximates the probability distribution of the text on the internet.
- For typical use-cases, we want the LLM to behave more like an assistant.

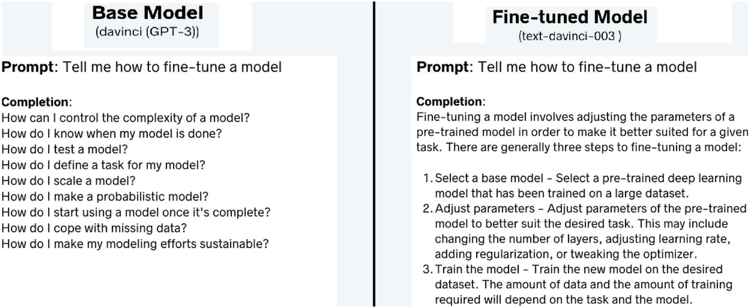


Figure: Source

- Example:
  - Given a question, answer it politely.
  - Given a follow-up, clarify and iterate.
- This kind of text is rare, especially for multi-turn conversations.
  - Stack Overflow is usually one turn.
  - Reddit contains limited multi-turn interactions.

Fine-Tuning

- Solution: Enrich the LLM with curated data.
- Ask humans to write Q&A format text. Companies may use their own user-submitted data.
- Train the model with the same loss function, starting from the pre-trained model, on this data. Given $q_1, a_1, q_2, a_2, \ldots, q_k$, predict a_k.
- More generally, fine-tuning amounts to targeted training of an already pre-trained model on small but task-specific data.

Small-Scale Fine-Tuning Can Work

- Post-training/alignment can be achieved with a relatively small number of carefully selected examples.
- Superficial Alignment Hypothesis (Zhou et al., 2023): Most capabilities are learned during pretraining. Alignment: which formatting style to use.
  - Select 1000 high-quality text pieces from web (e.g., highly rated answer on Stack Exchange; removing low-qual answers, e.g., “as mentioned"), diversity (a few from each SO community))

Source	#Examples	Avg Input Len.	Avg Output Len.
Training			
Stack Exchange (STEM)	200	117	523
Stack Exchange (Other)	200	119	530
wikiHow	200	12	1,811
Pushshift r/WritingPrompts	150	34	274
Natural Instructions	50	236	92
Paper Authors (Group A)	200	40	334
Dev			
Paper Authors (Group A)	50	36	N/A
Test			
Pushshift r/AskReddit	70	30	N/A
Paper Authors (Group B)	230	31	N/A

Table 1: Sources of training prompts (inputs) and responses (outputs), and test prompts. The total amount of training data is roughly 750,000 tokens, split over exactly 1,000 sequences.

Small-Scale Fine-Tuning Can Work

- Zhou et al. (2023) fine-tune Llama 2 70B for a small # of epochs.
- Get perf comparable to much larger models.

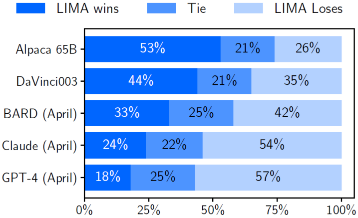


Figure 1: Human preference evaluation, comparing LIMA to 5 different baselines across 300 test prompts.

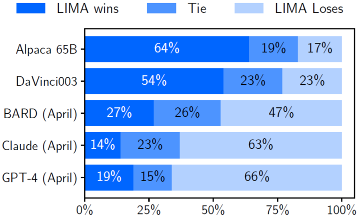


Figure 2: Preference evaluation using GPT-4 as the annotator, given the same instructions provided to humans.

Costs of Fine-Tuning

- How much does fine-tuning cost?
- Exact numbers hard to find, but (roughly) thousands of GPU-hours

Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	\$5.328M	\$0.238M	\$0.01M	\$5.576M

Table 1 | Training costs of DeepSeek-V3, assuming the rental price of H800 is \$2 per GPU hour.

- May still cost millions of USD, due to salary, data gathering, iteration, external red-teaming/safety, ...
- Note: DeepSeek V3's reported costs are much lower, but they lack: tool use, safety training, ... See more here

Effects of Fine-Tuning

- Improves performance on target tasks.
- May distort pretrained features, leading to worse out-of-distribution performance (Kumar et al., 2022).
- May degrade calibration (OpenAI, 2023).

Instruction Tuning

- Extends fine-tuning by incorporating multiple task variations, and instruction-following data (Wei et al., 2022; Sanh et al., 2022).
  - “For each dataset, we manually compose ten unique templates that use natural language instructions to describe the task for that dataset."
  - “For each dataset we include up to three templates that “turned the task around,” (e.g., for sentiment classification we include templates asking to generate a movie review)." [principle: transform data to have coverage in target distribution]
- Can automate it by letting LLM rewrite simple instructions into more complex ones; as in Evol-instruct (Xu et al., 2024a). [principle: find the task that the LLM can reasonably do instead of the human]

Table of Contents

Pre-training

Post-training
- SFT
- Alignment
  - Preference Modeling
  - Using the Learned Reward
  - Direct Preference Optimization
  - GRPO

Towards Alignment

- Suppose we want the model to be polite. No clear definition is available.
- First thought: collect a dataset of prompt-response pairs (x, y), where y is specifically chosen to be polite. For example, the humans writing the answers are asked to write politely.
- Inefficient: humans need to write a lot of responses.

Towards Alignment: Learning a Reward Model

- Instead, only ask humans about their preferences/rating. The most direct approach is as follows:
  - Human prompt x
  - LLM answer y
  - Human evaluation/reward $\tilde{r} := \tilde{r}(x, y)$ (high if good)
- Then use this data to learn a reward model $r : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ by regressing $\tilde{r}(x, y)$ onto (x, y), using supervised learning.
  - Use the same transformer architecture, just change the last layer to a scalar readout.

Towards Alignment: Using a Reward Model

- Simplest approach: Rejection sampling+SFT
  - Sample a number of responses
  - Keep the few best according to the reward
  - Use for SFT
- E.g., Llama-3 (Dubey et al., 2024) samples between 10-30 responses
- Will discuss further approaches

Eliciting a Reward

- Crucial challenge: eliciting a reward from humans can be hard. For instance:
  - On a scale of 1 to 10, how polite are the following answers?
    1. What time is it?
    2. Could you please let me know the time?
    3. Would you mind telling me the time?
    4. Can you tell me what the time is?

Learning a Reward by Modeling Preferences

- Collect preference data:
  - Human prompt x
  - LLM answers y_1, y_2
  - Human preference $I(y_1 \succ y_2)$, i.e., is y_1 preferred over y_2?
- To learn the reward, consider a probabilistic ranking model, where for all x, y_1, y_2, the probability that a human labeler prefers y_1 over y_2 given x is

$$P(y_1 \succ y_2 | x) = F(r(x, y_1), r(x, y_2))$$

  for some known function F.
- Fit this to the data to learn the reward r. [r is called a preference model in Bai et al. (2022)]

BTM Preference Model

- Most commonly used preference model: Bradley-Terry model (BTM) (Bradley and Terry, 1952), according to which for all x, y_1, y_2:

$$P(y_1 \succ y_2 | x) = \sigma\left(r(x, y_1) - r(x, y_2)\right),$$

  where σ is the sigmoid function, $\sigma : x \mapsto 1/(1 + e^{-x})$.
- Intuition: given two players with latent skills w_1, w_2, BTM models the probability that the first one wins a game against the second one by $\exp(w_1)/[\exp(w_1) + \exp(w_2)]$.
- The reward is only identified up to an x-dependent function, i.e., two rewards r, r' are equivalent if $r(x, y) = r'(x, y) + g(x)$ for all x, y.
  - Up to this, the model is identifiable: recover the reward from the probabilities via $r(x, y) = \mathrm{logit} P(y \succ y_0 | x) + r(x, y_0)$, for an any y_0.

BTM Likelihood

- Suppose that we have triples (x, y^+, y^-) of contexts x, preferred answers y^+, and less preferred answers y^-.
  - The contexts are sampled from some dataset D, which in some hypothetical limit follows a distribution ρ.
  - For the answers, assume for simplicity that they are sampled independently from some policy/LM $\mu(\cdot|x)$, i.e., $Y_1, Y_2|x \sim \mu(\cdot|x)$.
  - Then, given y_1 and y_2, they are re-labelled as y^+ and y^-, such that $P(Y^+ = y_1|x, y_1, y_2) = P(y_1 \succ y_2|x)$.
- Therefore, the joint pmf of (X, Y^+, Y^-) takes values

  $$P((X, Y^+, Y^-) = (x, y^+, y^-)) = 2\rho(x)\mu(y^+|x)\mu(y^-|x)P(y^+ \succ y^-|x).$$

  Only $P(y^+ \succ y^-|x)$ depends on the reward r.
- Hence, the log-likelihood is, up to terms that do not depend on r,

  $$\log P(y^+ \succ y^-|x) = \log \sigma\left(r(x, y^+) - r(x, y^-)\right).$$

Learning the Reward Function

- Given preference data D, maximize:

$$\max_{\theta \in \Theta} \sum_{(x, y^+, y^-) \in D} \log \sigma \left(r_\theta(x, y^+) - r_\theta(x, y^-) \right) .$$

- Population limit:

$$\mathbb{E}_{X \sim \rho} \mathbb{E}_{Y_1, Y_2 \sim \mu(\cdot|X)} \log \sigma \left(r_\theta(X, Y^+) - r_\theta(X, Y^-) \right)$$

  - The distribution of X, Y does not affect the resulting optimal r: For each X, Y for which the pmf is positive, we recover r (up to an x-shift).
  - However, the efficiency may be affected by the distribution. Liu et al. (2024) argue that the distribution of Y should ideally follow that of the desired target LLM for which we wish to use the reward. (circular!)
- Overfitting in sample objective? See Zhu et al. (2024)

Using the Learned Reward

- Saw rejection sampling+SFT.
- Reinforcement learning from human feedback (RLHF) based on the learned reward:
  - Fine-tune the model by policy maximization (LM is policy):

    $$\max_{w} \mathbb{E}_{X \sim D} \mathbb{E}_{Y \sim p_w(\cdot|X)}\, r(X, Y).$$

    This is a reinforcement learning problem.
  - To ensure that the model does not move too far from the pre-trained model p_{ref}, and that it does not overfit a potentially small dataset D, we may use KL regularization, as in proximal policy optimization (PPO) (Schulman et al., 2017):

    $$\max_{w} \mathbb{E}_{X \sim D} [\mathbb{E}_{Y \sim p_w(\cdot|X)}\, r(X, Y) - \beta\, \mathrm{KL}\,(p_w(\cdot|X) \,\|\, p_{\mathrm{ref}}(\cdot|X))]$$

    [referred to as the RLHF objective]

RLHF

- Two components:
  1. Learning a reward.
  2. Policy/LLM optimization.
- Jointly maximize over w, θ the BTM-MLE and PPO objectives:

$$\mathbb{E}_{X \sim D} \mathbb{E}_{Y_1, Y_2 \sim p_w(\cdot|X)} \log \sigma \left(r_\theta(X, Y^+) - r_\theta(X, Y^-) \right)$$
$$\mathbb{E}_{X \sim D} [\mathbb{E}_{Y \sim p_w(\cdot|X)} r_\theta(X, Y) - \beta \, \mathrm{KL} \left(p_w(\cdot|X) \, \| \, p_{\mathrm{ref}}(\cdot|X) \right)].$$

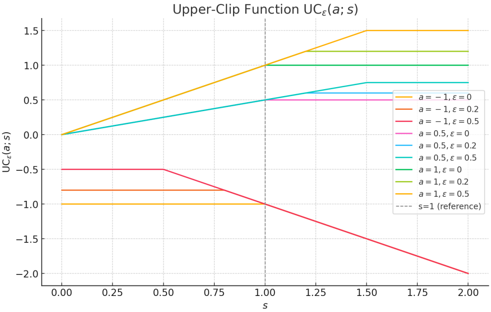
- Typically this is done sequentially:
  - first maximizing the first objective with respect to w (for a given p_w, e.g. the pre-trained model),
  - and then maximizing the second objective with respect to w for the given w.
- However one could imagine simultaneous optimization.

PPO

- Define the "upper-clip" function UC by the piecewise expression, for $s \geq 0$, $a \in \mathbb{R}$, and a clipping hyperparameter $\varepsilon \geq 0$,

$$\mathrm{UC}_\varepsilon(a; s) \;=\; \begin{cases} a \;\mathrm{min}\big(s,\; 1+\varepsilon\big), & a \geq 0, \\ a \;\mathrm{max}\big(s,\; 1-\varepsilon\big), & a < 0. \end{cases}$$

This clamps a "above" (when $a \geq 0$, it becomes $\leq a(1+\varepsilon)$; when $a \leq 0$, it becomes $\leq a$, but could be arbitrarily large and negative).



PPO

- Solve: $\max_w \mathbb{E}_{X\sim D}[\mathbb{E}_{Y\sim p_w(\cdot|X)}\, r(X,Y) - \beta\, \mathsf{KL}\left(p_w(\cdot|X) \,\|\, p_{\mathrm{ref}}(\cdot|X)\right)]$
- PPO objective (Schulman et al., 2017):

$$\mathbb{E}_{X\sim D}\mathbb{E}_{Y\sim p_{w_{\mathrm{old}}}(\cdot|X)}\left[\mathrm{UC}_\varepsilon\left(\widehat{A}; \frac{p_w(Y\mid X)}{p_{w_{\mathrm{old}}}(Y\mid X)}\right)\right].$$

where:

1. p_w: The current policy model that we optimize (take gradients),
2. $p_{w_{\mathrm{old}}}$: The previous policy model used for sampling Y,
3. $\widehat{A}$: The relative advantage:

$$\widehat{A} = \tilde{r}(X,Y) - \widehat{V}_\phi(X)$$

Here:

3.1 $\tilde{r}$ is augmented reward:

$$\tilde{r}(X,Y) = r(X,Y) - \beta\, \mathsf{KL}\left(p_w(\cdot|X) \,\|\, p_{\mathrm{ref}}(\cdot|X)\right)$$

3.2 $\widehat{V}_\phi$: value function estimate $\widehat{V}_\phi(X) \approx \mathbb{E}_{Y\sim p_w(\cdot|X)}\, \tilde{r}(X,Y)$

3.3 Contributes obj term $\phi \mapsto -c\mathbb{E}(\widehat{V}_\phi(X) - \tilde{r}(X,Y))^2$, $c>0$ is weight

Aside: RL and Advantage

- Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where:
  - $\mathcal{S}$ is the state space.
  - $\mathcal{A}$ is the action space.
  - $P(s' \mid s, a)$ is the transition probability function.
  - $r(s, a)$ is the reward function.
  - $\gamma \in [0, 1)$ is the discount factor.
- A policy π is a distribution over actions given states: $\pi(a \mid s) = P(a_t = a \mid s_t = s)$, defining the agent's decision-making.
- State-Action Trajectory: Given an initial state s_0, the trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ is generated as follows:
  - Sample $a_t \sim \pi(\cdot \mid s_t)$.
  - Sample $s_{t+1} \sim P(\cdot \mid s_t, a_t)$.
  - Observe reward $r_t = r(s_t, a_t)$.
- Value Function: Expected return when following policy π from state s_t: $V_\pi(s_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t\right]$.
- Action-Value Function: Expected return when taking action a_t in state s_t and following π after: $Q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t, a_t\right]$.
- Advantage Function: Measures how much better action a_t is compared to the policy's average: $A_t = Q_\pi(s_t, a_t) - V_\pi(s_t)$.

PPO

- KL-regularized version (Schulman et al., 2017):

$$\mathbb{E}_{X\sim D}\mathbb{E}_{Y\sim p_{w_{\text{old}}}(\cdot|X)}\left[\widehat{A}\frac{p_w(Y \mid X)}{p_{w_{\text{old}}}(Y \mid X)} - \beta\,\mathsf{KL}\left(p_{w_{\text{old}}}(\cdot|X)\,\|\,p_w(\cdot|X)\right)\right].$$

  Note reversion in order in the KL. With adaptive β.
- Per-token version:

$$\mathbb{E}_{X\sim D, Y\sim p_{w_{\text{old}}}(\cdot|X)}\left[\frac{1}{|Y|}\sum_{t=1}^{|Y|}\mathrm{UC}_{\varepsilon}\left(\widehat{A}_t; \frac{p_w(Y_t \mid X, Y_{<t})}{p_{w_{\text{old}}}(Y_t \mid X, Y_{<t})}\right)\right].$$

- Implementation details can be tricky

Introduction to DPO

- Can we simplify reward learning and fine-tuning into one step?
- Direct Preference Optimization (DPO) (Rafailov et al., 2024) achieves this.
- Based on a reverse KL representation of PPO obj:

$$\mathbb{E}_{X\sim D}[\mathbb{E}_{Y\sim p_w(\cdot|X)}\, r(X,Y) - \beta\,\mathrm{KL}\left(p_w(\cdot|X)\,\|\,p_{\mathrm{ref}}(\cdot|X)\right)]$$

  - Can write this as the reverse KL objective

$$-\beta\,\mathrm{KL}\left(p_w(\cdot|X)\,\|\,p^*(\cdot|X)\right) + C,$$

  where $p^*(\cdot|x) = p_{\mathrm{ref}}(\cdot|x)\exp(r(x,y)/\beta)/Z(x)$, and Z is a normalization factor, and C does not depend on p_w see e.g., Peters and Schaal (2007).

Reverse KL Representation

We want to prove that:

$$\mathbb{E}_{Y\sim p_w}[r(Y)] - \beta\,\mathsf{KL}(p_w \| p_{\mathrm{ref}}) = -\beta\,\mathsf{KL}(p_w \| p^*) + C,$$

where $p^*(y) = \frac{p_{\mathrm{ref}}(y)\exp(r(y)/\beta)}{Z}$ with normalization factor Z, and C does not depend on p_w. Expanding the KL divergence, $\mathsf{KL}(p_w \| p^*) = \sum_y p_w(y) \log \frac{p_w(y)}{p^*(y)}$. Now, $\log p^*(y) = \log p_{\mathrm{ref}}(y) + \frac{r(y)}{\beta} - \log Z$.Thus,

$$\mathsf{KL}(p_w \| p^*) = \sum_y p_w(y) \left(\log p_w(y) - \log p_{\mathrm{ref}}(y) - \frac{r(y)}{\beta} + \log Z \right).$$

Rewriting,

$$\mathsf{KL}(p_w \| p^*) = \mathsf{KL}(p_w \| p_{\mathrm{ref}}) - \frac{1}{\beta}\mathbb{E}_{Y\sim p_w}[r(Y)] + \log Z,$$

$$\mathbb{E}_{Y\sim p_w}[r(Y)] - \beta\mathsf{KL}(p_w \| p_{\mathrm{ref}}) = -\beta\mathsf{KL}(p_w \| p^*) + \beta \log Z.$$

Observe that Z does not depend on p_w

Reverse KL Optimization

- From the reverse KL objective representation, maximizing the PPO objective over all policies, given the pre-trained model p_{ref}, has solution, for all x with positive probability under D:

$$p(y|x) = p_{\mathrm{ref}}(y|x)\frac{\exp(r(x,y)/\beta)}{Z(x)}, \tag{1}$$

  where $Z(x)$ is a normalization factor.
- The values of x outside of the fine-tuning dataset do not affect the objective. Therefore, could use this parametrization for all x.
- Sampling from $p \propto p_{\mathrm{ref}} \exp(r/\beta)$ in autoregressive manner is non-trivial.
  - Reward may not be well-defined for partial sequences.
  - Rarely used directly.
- Gap: Fine-tuning typically uses a restricted parameterization (Wipf, 2024).

BTM Objective

- Substitute p into the BTM objective to eliminate the reward.
  - PPO optimality is equivalent to

$$r(x, y)/\beta = \log\left(\frac{p(y|x)}{p_{\mathrm{ref}}(y|x)}\right) - Z(x).$$

  - Hence, the BTM objective becomes

$$\sigma\left(\beta\left[\log\left(\frac{p(y_1|x)}{p_{\mathrm{ref}}(y_1|x)}\right) - Z(x)\right] - \beta\left[\log\left(\frac{p(y_2|x)}{p_{\mathrm{ref}}(y_2|x)}\right) - Z(x)\right]\right)$$
$$= \sigma\left(\beta\log\left(\frac{p(y_1|x)}{p_{\mathrm{ref}}(y_1|x)} \Big/ \frac{p(y_2|x)}{p_{\mathrm{ref}}(y_2|x)}\right)\right).$$

  - Since $\sigma(\log(a)) = a/(a+1)$, this can be simplified to

$$\frac{\frac{p^{\beta}(y_1|x)}{p^{\beta}_{\mathrm{ref}}(y_1|x)}}{\frac{p^{\beta}(y_1|x)}{p^{\beta}_{\mathrm{ref}}(y_1|x)} + \frac{p^{\beta}(y_2|x)}{p^{\beta}_{\mathrm{ref}}(y_2|x)}}.$$

Towards DPO

- Recall the RLHF objectives. The above procedure
  - first optimizes over all policies p_w in the second, PPO obj;
  - then substitutes the resulting relation between r_θ and the optimal policy in the first—BTM-MLE—objective
  - Implicitly requires that the space over which we optimize r_θ is large enough to include this solution
- Let

$$P_w(y_1 \succ y_2|x) := \frac{\frac{p_w^\beta(y_1|x)}{p_{\mathrm{ref}}^\beta(y_1|x)}}{\frac{p_w^\beta(y_1|x)}{p_{\mathrm{ref}}^\beta(y_1|x)} + \frac{p_w^\beta(y_2|x)}{p_{\mathrm{ref}}^\beta(y_2|x)}}.$$

- Then, computing the value of the BTM-MLE objective for the PPO-optimal policy we obtain the DPO objective:

$$\max_w \mathbb{E}_{X\sim D}\mathbb{E}_{Y^+,Y^-\sim p_w(\cdot|X)} \log P_w(Y^+ \succ Y^-|X).$$

- Use this objective to fine-tune an LLM?
  - Hard to obtain preference data for the optimization policy p_w (need preference labels for each policy iterate?)
  - Instead, use any pref data, or preference data from the base policy p_{ref}, or rejection sampled data (Liu et al., 2024), which can be closer to the optimal policy.

Final DPO Objective

- The final and "standard" DPO objective is often presented as

$$\max_w \mathbb{E}_{X,Y^+,Y^-\sim D} \log \sigma \left(\beta \log \left(\frac{p_w(Y^+|X)}{p_{\mathrm{ref}}(Y^+|X)} \right) - \beta \log \left(\frac{p_w(Y^-|X)}{p_{\mathrm{ref}}(Y^-|X)} \right) \right).$$

- Empirical results can sometimes be mixed; but used e.g., in Llama 3 (Dubey et al., 2024).

DPO Considerations

- Azar et al. (2024) argue that PPO/DPO can behave problematically when $P(y_1 \succ y_2|x) = 1$ for some x, y_1, y_2.
  1. For PPO $P(y_1 \succ y_2|x) = \sigma(r(x, y_1) - r(x, y_2))$: we would need $r(x, y_1) - r(x, y_2) = \infty$, so that for the optimum $p \propto p_{\mathrm{ref}} \exp(r/\beta)$, we have $p(y_1|x) = 1$ and $p(y_2|x) = 0$, regardless of the value of β.
  2. For DPO: objective is monotonic in $\frac{p_w(y_1|x)}{p_{\mathrm{ref}}(y_1|x)} / \frac{p_w(y_2|x)}{p_{\mathrm{ref}}(y_2|x)}$, maximized for $p_w(y_1|x) = 1$ and $p_w(y_2|x) = 0$, regardless of the value of β.
- Optimal policy does not take regularization into account.
- As a remedy, they propose *total preference optimization*:

$$\mathbb{E}_{X \sim D}[\mathbb{E}_{Y \sim p_w(\cdot|X), Y' \sim \mu(\cdot|X)} P(Y \succ Y'|X) - \beta \, \mathrm{KL} \left(p_w(\cdot|X) \, \| \, p_{\mathrm{ref}}(\cdot|X) \right)]$$

  where the reward $P(Y \succ Y'|X)$ is the win rate of p_w against a base policy μ when $Y \sim p_w(\cdot|X)$, $Y' \sim \mu(\cdot|X)$

Generalized Preference Optimization

- More generally, Azar et al. (2024) introduce Ψ-preference optimization, where $P(Y \succ Y'|X)$ is replaced by $\Psi(P(Y \succ Y'|X))$, for a general non-decreasing function Ψ.
- Taking $\Psi(q) = \log(q/(1-q))$, we have that, under the BTM with r,

$$\begin{aligned}\mathbb{E}_{Y'\sim\mu}\,\Psi(P(Y \succ Y'|X)) &= \mathbb{E}_{Y'\sim\mu}\,\log(\exp[r(X,Y)]/\exp[r(X,Y')]) \\ &= r(X,Y) - \mathbb{E}_{Y'\sim\mu}\,r(X,Y').\end{aligned}$$

  Thus ΨPO with this non-linearity recovers PPO.

More on PPO vs DPO

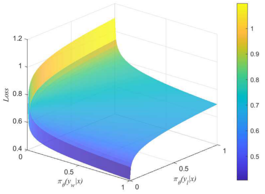
- Xu et al. (2024b) show that there can be policies that maximize the DPO objective, but not the PPO objective.
- Example: no x, three actions. Data D: one datapoint $\{(y^+ = y_1, y^- = y_2)\}$. Rewards $r_1 > r_2$.

Action	y_1	y_2	y_3
p_{ref}	0.5	0.5	0
p_{DPO}	0.1	0.0	0.9
p_{PPO}	1	0	0

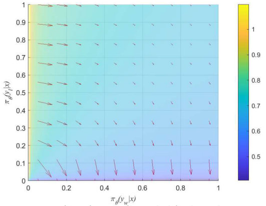
- PPO: $r_1 p_1 + r_2 p_2 + r_3 p_3 - \beta \mathrm{KL}(p \| p_{\mathrm{ref}})$. The KL divergence to p_{ref} enforces the probability of outputting y_3 to be zero.
- DPO: choose uniform p_{ref}. Obj is $(p_1, p_2) \mapsto \log(p_1^\beta / (p_1^\beta + p_2^\beta))$
- Xu et al. (2024b): "DPO might discover solutions exploiting out-of-distribution data, posing a risk of deviating excessively from the reference policy"

Additional issues with DPO

- There are a number of additional issues with DPO, see e.g., here
- Gradient dynamics can be slow to move desired probabilities (Feng et al., 2024)
  - Simple setting: No x, data $y_1 \succ y_2$. DPO obj: $\log\left(p_1^\beta/(p_1^\beta + p_2^\beta)\right)$
  - Find
$$\frac{\partial L}{\partial p_1} = \beta \frac{p_2^\beta}{p_1(p_1^\beta + p_2^\beta)}, \; \frac{\partial L}{\partial p_2} = -\beta \frac{p_2^{\beta-1}}{p_1^\beta + p_2^\beta},$$
    hence their ratio has absolute value p_2/p_1.
  - $\frac{\partial L}{\partial p_1} \geq 0$: increase the likelihood of chosen resp.; $\frac{\partial L}{\partial p_2} \leq 0$: decrease the likelihood of non-pref. resp.
  - Want $p_1 \geq p_2$. However, at some point $p_1 \approx p_2$, then it takes a long time to increase p_1



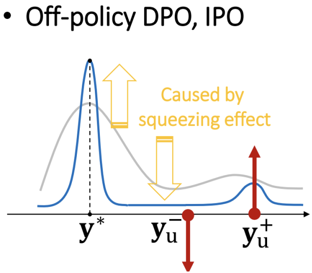
(a) The optimization plane (loss landscape) of DPO



(b) The gradient field of DPO

Additional issues with DPO

- "Gradient squeezing" (Ren and Sutherland, 2024): p_* of confident answers increases, whereas p_1, p_2 both decrease, esp. for off-policy data generated by a policy different from the current one.



- Confident answers are more likely to be correct than not, so DPO "accidentally" improves perf; related to the fact that best-of-N works well (Huang et al., 2024)

Empirical Usage

- Used in Llama 3 (Dubey et al., 2024).
- Tricks:
  - Mask out formatting tokens (the same tokens in both positive and negative answers may cause model instability due to the contrastive loss);
  - Add scaled NLL loss for chosen response

GRPO

- Can we find a version of PPO that does not require training an advantage estimator?
- Group Relative Preference Optimization, proposed in DeepSeekMath (Shao et al., 2024) aims to do this. Used in DeepSeek-R1 (Guo et al., 2025)
- Consider the KL-PPO objective, with the KL in the order from RLHF (Schulman et al., 2017), averaged over a group of G datapoints $X, Y^{(i)}$:

$$\mathbb{E}_{X\sim D, Y^{(i)}\sim p_{w_{\text{old}}}(\cdot|X), i\in[G]}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|Y^{(i)}|}\sum_{t=1}^{|Y^{(i)}|}\left\{\mathrm{UC}_{\varepsilon}\left(\widehat{A}_t^{(i)}; \frac{p_w(Y_t^{(i)} \mid X, Y_{<t}^{(i)})}{p_{w_{\text{old}}}(Y_t^{(i)} \mid X, Y_{<t}^{(i)})}\right) - \beta \mathrm{KL}(p_w(\cdot \mid X, Y_{<t}^{(i)}) \| p_{\text{ref}}(\cdot \mid X, Y_{<t}^{(i)}))\right\}\right],$$

where:

1. p_w: The current LM,
2. $p_{w_{\text{old}}}$: The previous LM, used for sampling,
3. p_{ref}: A reference model, typically the initial fine-tuned model, [as in RLHF, not PPO]
4. $\widehat{A}_t^{(i)}$: The estimated relative advantage

GRPO

- The difference between GRPO and PPO is that the advantage is estimated purely based on the rewards in the group/batch of datapoints.
- **Outcome Supervision**: rewards r_i are assigned at the end of each sampled output $X, Y^{(i)}$. For all tokens:

$$\widehat{A}_{i,t} = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})},$$

  [Intuition: instead of constructing a (parametric) value function estimator $\hat{V}_\phi$, estimate value by (non-parametric) sampling]
- **Process Supervision**: rewards are assigned at each reasoning step within the sampled outputs. For token t in $X, Y^{(i)}$:

$$\widehat{A}_{i,t} = \sum_{\text{index}(j) \geq t} \frac{r^i_{\text{index}(j)} - \text{mean}(R)}{\text{std}(R)},$$

  where:
  1. $\text{index}(j)$: The ending token index of the j-th reasoning step.
  2. $r^i_{\text{index}(j)}$: The reward assigned to the j-th step of the i-th output,
  3. $R = \{r^i_{\text{index}(j)} \mid i = 1, \ldots, G, j = 1, \ldots, K_i\}$: The set of all rewards across the group, where K_i is # of reasoning steps

DPO vs GRPO

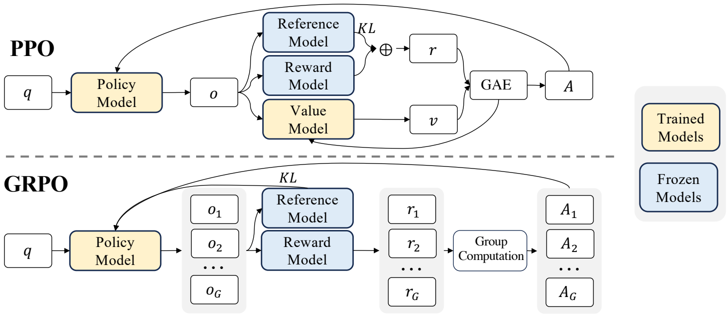


Figure: (Shao et al., 2024). Notation: $q \mapsto x$, $o \mapsto y$

Rewards in GRPO

- DeepSeek-R1 uses reward shaping:

$$r = \sigma(\alpha \cdot r_m) + (r_v - 1)$$

  where:
  - r_m: Reward output from the reward model.
  - r_v: Reward from the rule-based verifier ($r_v \in \{0, 1\}$).
  - σ: Sigmoid function.
  - α: Scaling coefficient for the reward model output (e.g., $\alpha = 0.5$).

Estimating KL Divergence

1. How to estimate the Kullback–Leibler (KL) divergence $\mathrm{KL}[q\|p] = \mathbb{E}_{X\sim q}\left[\log \frac{q(X)}{p(X)}\right]$, where we assume we can compute probabilities $p(x)$ and $q(x)$, but cannot sum over x?
2. An unbiased estimator using $X \sim q$ is

$$\log \frac{q(X)}{p(X)} \; = \; -\log r,$$

   where $r = \frac{p(X)}{q(X)}$.
3. A general way to reduce variance without introducing bias is to add a control variate with zero mean. One choice leads to $\lambda(r-1) \; - \; \log r$, which remains nonnegative for $\lambda = 1$.
   - We expect $r-1$ and $\log r$ to be positively correlated (increasing functions of r).

Estimating KL Divergence in GRPO

- In Dai et al. (2024), the KL divergence term

$$\mathbb{E}_{Y_t \sim p_{w_{\text{old}}}(\cdot|X)} \mathrm{KL}(p_w(\cdot \mid X, Y_{<t}) \| p_{\text{ref}}(\cdot \mid X, Y_{<t})),$$

  is estimated by

$$\frac{p_{\text{ref}}(Y_t \mid X, Y_{<t})}{p_w(Y_t \mid X, Y_{<t})} - \log\left(\frac{p_{\text{ref}}(Y_t \mid X, Y_{<t})}{p_w(Y_t \mid X, Y_{<t})}\right) - 1,$$

  [so $p = p_{\text{ref}}$, $q = p_w$]
- How is Y sampled?
  - Dai et al. (2024) say it is sampled with respect to $p_{w_{\text{old}}}$. But then the estimate is not unbiased. This means the objective is simply the expectation of the above under $p_{w_{\text{old}}}$, rather than the KL divergence itself.
  - Would need it to be sampled with respect to the current model p_w.

GRPO Summary

Initialize $p_w \leftarrow p_{\text{ref}}$
for $k = 1$ **to** K **do** ▷ Outer RL iterations
  Set $p_{\text{old}} \leftarrow p_w$
  for $m = 1$ **to** M **do** ▷ Data collection/update steps
    Sample a mini-batch of questions $\{x\} \sim \mathcal{D}$
    for each question x in the mini-batch **do**
      (a) Sample a group of G outputs: $\{y_1, y_2, \ldots, y_G\} \sim p_w(\cdot \mid x)$
      (b) For each output y_i, compute its scalar reward: $r_i \leftarrow r_\phi(x, y_i)$
      (c) Compute $\bar{r} = \frac{1}{G}\sum_{i=1}^{G} r_i$, and $\sigma = \mathrm{SD}(r_i, i \in [G])$
      (d) Set the group relative advantage: $\hat{A}_{i,t} = (r_i - \bar{r})/\sigma$
    for $j = 1$ **to** μ **do** ▷ Inner GRPO updates
      Perform a gradient ascent step on the GRPO objective
return p_w

GRPO Perf Dynamics in R1

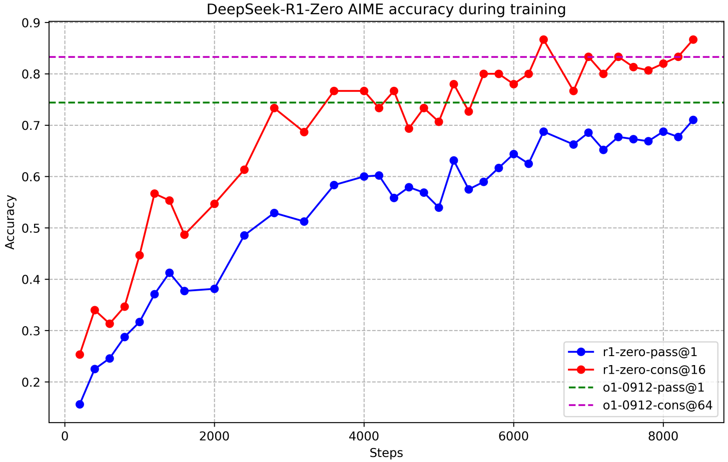


Figure: Guo et al. (2025)

GRPO Demo

- Colab demo, for Qwen 0.5B. Takes $\approx$ 2 hours on an A100. Based on Will Brown's demo, using trl (Transformer Reinforcement Learning) lib. Key file is `grpo_trainer.py` (Feb 2025)
  - Group Reward/Advantage
  - Per-token KL and loss
- See also Open-R1

References

M. G. Azar, Z. D. Guo, B. Piot, R. Munos, M. Rowland, M. Valko, and D. Calandriello. A general theoretical paradigm to understand learning from human preferences. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 4447–4455, 2024.

Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

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

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

References

D. Feng, B. Qin, C. Huang, Z. Zhang, and W. Lei. Towards analyzing and understanding the limitations of dpo: A theoretical perspective. *arXiv preprint arXiv:2404.04626*, 2024.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

A. Huang, A. Block, D. J. Foster, D. Rohatgi, C. Zhang, M. Simchowitz, J. T. Ash, and A. Krishnamurthy. Self-improvement in language models: The sharpening mechanism. *arXiv preprint arXiv:2412.01951*, 2024.

A. Kumar, A. Raghunathan, R. M. Jones, T. Ma, and P. Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=UYneFzXSJWh.

T. Liu, Y. Zhao, R. Joshi, M. Khalman, M. Saleh, P. J. Liu, and J. Liu. Statistical rejection sampling improves preference optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xbjSwwrQOe.

A. Y. Ng, S. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, 2000.

OpenAI. Gpt-4 technical report, 2023.

References

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022.

J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 36, 2024.

Y. Ren and D. J. Sutherland. Learning dynamics of llm finetuning. *arXiv preprint arXiv:2407.10490*, 2024.

V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

References

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.

M. Weber, D. Y. Fu, Q. G. Anthony, Y. Oren, S. Adams, A. Alexandrov, X. Lyu, H. Nguyen, X. Yao, V. Adams, B. Athiwaratkun, R. Chalamala, K. Chen, M. Ryabinin, T. Dao, P. Liang, C. Re, I. Rish, and C. Zhang. Redpajama: an open dataset for training large language models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=lnuXaRpwvw.

J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

D. Wipf. Unavoidable learning constraints alter the foundations of direct preference optimization. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024. URL https://openreview.net/forum?id=Kofno31ckd.

References

C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, Q. Lin, and D. Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=CfXh93NDgH.

S. Xu, W. Fu, J. Gao, W. Ye, W. Liu, Z. Mei, G. Wang, C. Yu, and Y. Wu. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*, 2024b.

C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. YU, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, and O. Levy. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=KBMOKmX2he.

B. Zhu, M. I. Jordan, and J. Jiao. Iterative data smoothing: Mitigating reward overfitting and overoptimization in rlhf. *arXiv preprint arXiv:2401.16335*, 2024.

D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.