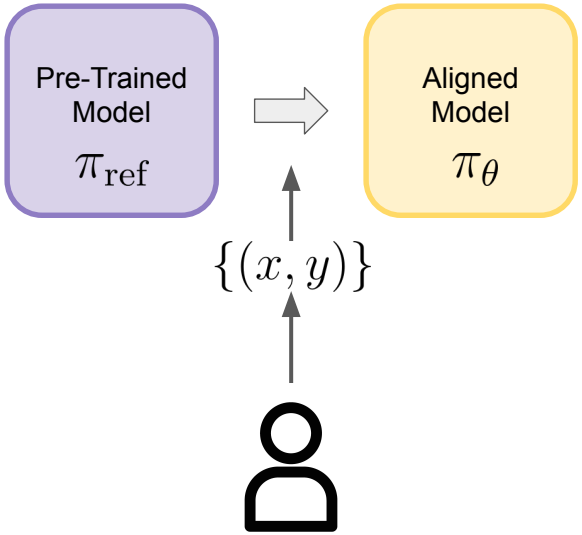
KTO: Model Alignment as Prospect Theoretic Optimization

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, Douwe Kiela

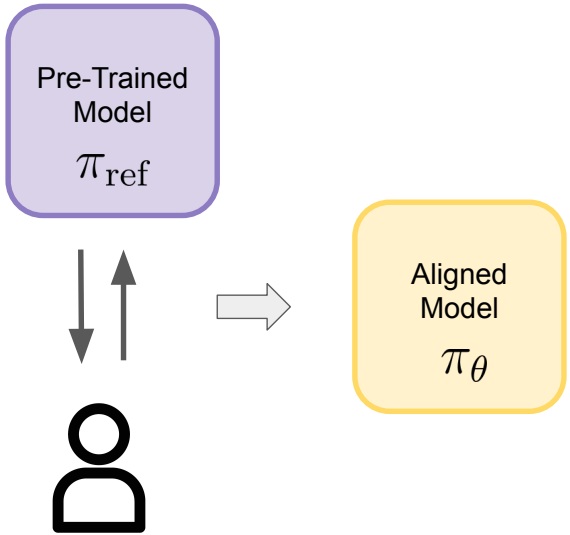
Presented by Kevin Liu & Nazar Misyats

LLM Alignment

- Goal: make model's outputs aligned with human preference
- Feedback-based methods complement fine-tuned approaches



Supervised Fine-Tuning (SFT)



Feedback-based

Background on feedback-based alignment

RLHF, PPO & DPO

Reinforcement Learning with Human Feedback (RLHF)

(Christiano et al., 2017)

- Human preference can be modeled via a reward

$$D = \{(x, \underbrace{y_w, y_l}_{y_w \succ y_l})\} \Longleftrightarrow r^*(x, y)$$

- The true reward is intractable, so we learn a reward model:

$$r_\phi(x, y)$$

- Alignment is obtained by maximizing the reward:

$$\max_\theta \mathbb{E}_{x \in D, y \in \pi_\theta}[r_\phi(x, y)] - \beta \mathrm{KL}(\pi_\theta(y|x) || \pi_{\mathrm{ref}}(y|x))$$

- Dual objective:
  1. Maximize reward
  2. Stay close to reference model
- In practice: RL with Proximal Policy Optimization (PPO)

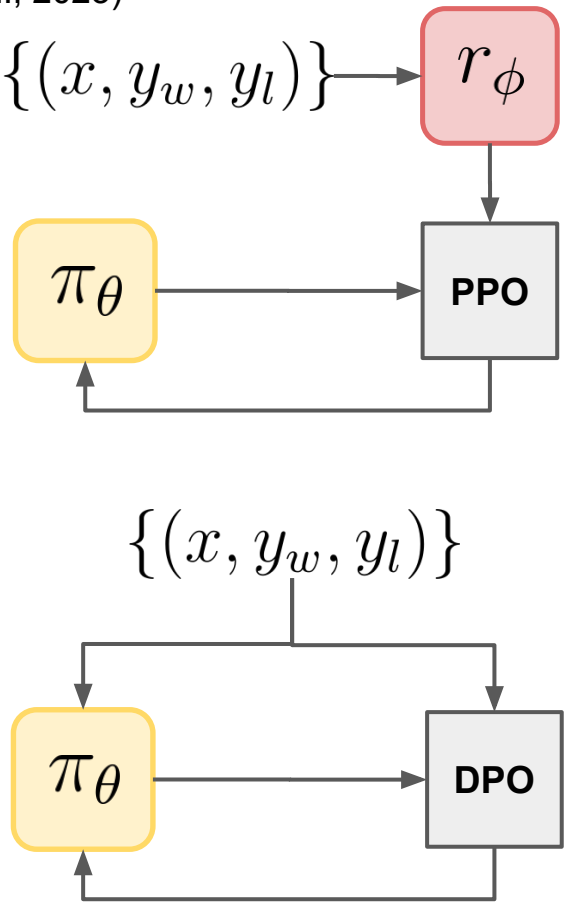
Direct Policy Optimization (DPO) (Rafailov et al., 2023)

- PPO requires training a reward model and then run RL with preference pairs
- **DPO directly optimizes the fine-tuned model from preference pairs**

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(r_\theta(x, y_w) - r_\theta(x, y_l) \right)$$

$$r_\theta(x, y) = \beta \log \frac{\pi_\theta(x, y)}{\pi_{\text{ref}}(x, y)}$$

Implied reward



Why does it work?

- Preference pairs are scarce and hard to obtain
- Human feedback takes many forms (e.g. approval/disapproval)
- Yes RLHF and DPO outperform naive SFT

- **Why RLHF & DPO work so well?**
- **Is there an underlying formalism?**

A prospect theoretic view of alignment

From theory to KTO

Prospect Theory (Tversky & Kahneman, 1992)

- Human perception of random variables is biased in a well-defined manner
- Losses weigh more than gains: higher gains are required to compensate
- Example:
  - 50% chance winning $2000, 50% chance losing $1000
  - Average is +$500, but subjectively the risk feels larger
- Humans judge outcomes relative to a *reference point*

- **Kahneman & Tversky *value function*:**

$$v(z;\lambda,\alpha,z_0) = \begin{cases} (z-z_0)^{\alpha} & \text{if } z \geq z_0 \\ -\lambda(z_0-z)^{\alpha} & \text{if } z < z_0 \end{cases}$$

- z_0 reference point
- α risk sensitivity
- λ loss aversion

- Later we write: $v(z;\lambda,\alpha,z_0) = v(z-z_0)$

Human Aware Losses (HALOs)

- Define an implied reward: measure of *suprisal* from π_{ref} to π_θ, normalized by *l(y)*

$$r_\theta(x, y) = l(y) \log \frac{\pi_\theta(x, y)}{\pi_{\text{ref}}(x, y)}$$

- Define a *human value* relative to a reference distribution *Q*(*Y'* | *x*)

$$v(r_\theta(x, y) - \mathbb{E}_Q[r_\theta(x, y')])$$

- A function *f* is a *human aware loss* (**HALO**) if:

$$\exists a_{x,y} \in \{\pm 1\} \text{ s.t. } f(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y\sim\mathcal{D}}[a_{x,y} v(r_\theta(x, y) - \mathbb{E}_Q[r_\theta(x, y')])] + C_\mathcal{D}$$

Constant

Feedback data

Does HALO matter?

- **DPO is HALO**, where Q places all masses on (x, y_l), and:

$$r_\theta(x,y) = \underbrace{\beta}_{l(y)} \log \frac{\pi_\theta(x,y)}{\pi_{\text{ref}}(x,y)} \qquad \mathcal{L}_{\text{DPO}}(\pi_\theta, \pi_{\text{ref}}) = \underbrace{\mathbb{E}_{x,y_l,y_w}}_{D}[\underbrace{-\log \sigma}_{a_{x,y}}(\underbrace{r_\theta(x,y_w) - r_\theta(x,y_l)}_{v})]$$

- RLHF with PPO(-Clip) **is HALO**
- Conditional SFT (Korbak et al., 2023) and SLiC (Zhao et al., 2023) are **not HALO**

- **HALO losses outperform non-HALO**

Motivating KTO

- HALO framework allows us to measure alignment from binary labels: desired/undersired output
- **Kahneman-Tversky Optimization (KTO)**: direct optimization on binary label
- Simpler and scalable data to collect (n to 2n samples, single output)

KTO Loss: DPO vs. KTO

DPO

- Input: a **preference pair** (x, y_w, y_l)
- Core idea: compare two outputs directly
- Objective: maximize **preference likelihood**

$$\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}$$

- Preferred output should score higher than dispreferred output

KTO

- Input: a **binary label** for one output y
- Core idea: score one output relative to a **reference point**
- Objective: maximize **utility-shaped feedback**

$$r_\theta(x, y) - z_0$$

- Desirable outputs should move above the baseline
- Undesirable outputs should move below it

KTO Loss: Reward, Reference Point, and Value

Reward

$$r_\theta(x, y) = \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

- reward is measurement of surprise

Reference point

$$z_0 = \mathrm{KL}(\pi_\theta(y'|x) \,\|\, \pi_{\text{ref}}(y'|x))$$

- Deviation from the reference model

Value function

$$v(x, y) = \begin{cases} \lambda_D \, \sigma(\beta(r_\theta - z_0)), \\ \lambda_U \, \sigma(\beta(z_0 - r_\theta)) \end{cases}$$

- β: saturation / risk sensitivity
- λ_D, λ_U: asymmetry in humand judgement

Default KTO loss

$$L_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y\sim D}\big[\lambda_y - v(x, y)\big]$$

KTO: Practical Implementation

Ideal reference point

$$z_0 = \mathrm{KL}(\pi_\theta(y'|x) \,\|\, \pi_{\mathrm{ref}}(y'|x))$$

- Depends on the full policy distribution
- Expensive to estimate exactly during training

Batched estimate

$$\hat{z}_0 = \max\left(0,\ \frac{1}{m}\sum_{i=1}^{m} \log \frac{\pi_\theta(y_j|x_i)}{\pi_{\mathrm{ref}}(y_j|x_i)}\right)$$

- Low variance but positive bias because of clamping

KTO Experiment

Hyperparameters, empirical comparisons, no-SFT behavior, and ablation study

KTO Experiment: What the Hyperparameters Reveal

Empirical findings

- For imbalanced data, reweight desirable and undesirable examples:

$$\frac{\lambda_D n_D}{\lambda_U n_U} \in [1, 1.5].$$

- Example: if there is 1 desirable and 10 undesirable examples, we can set $\lambda_U = 1$ and $\lambda_D \in [10, 15]$.
- **Gain sensitivity works better than loss sensitivity**: Rewarding good outputs is more important than avoiding bad ones.

Practical consequences

- Hyperparameters depend on the task.
- For safety-critical situations, stronger sensitivity to undesirable outputs can work better.
- **Dynamic hyperparameter selection** is a possible future direction.

KTO $\geq$ DPO Across Model sizes

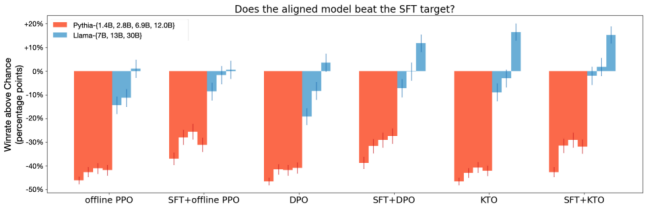


Figure 3. KTO is as good or better than DPO at all scales, as measured by the `GPT-4-0613`-judged winrate of the aligned model's generations against the outputs that would have been used for SFT. In fact, for the Llama models, KTO alone matches the performance of SFT+DPO and is significantly better than DPO alone. Error bars denote a 90% binomial confidence interval.

Experimental observations

- Across models from 1B to 30B, **SFT+KTO performs comparably to SFT+DPO**.
- For Llama models, **KTO alone can match SFT+DPO**: strong alignment ability.
- The advantage becomes significant for larger models (7B and 30B).

KTO Can Work Without SFT at Large sizes

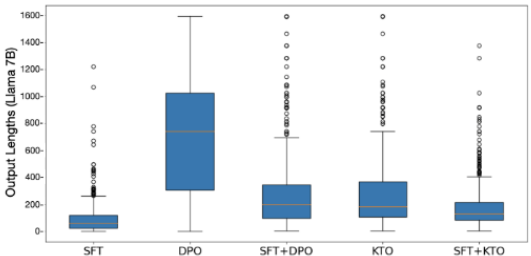


Figure 4. Without doing SFT first, DPO-aligned models tend to ramble and hallucinate entire conversations. KTO does not suffer from this issue.

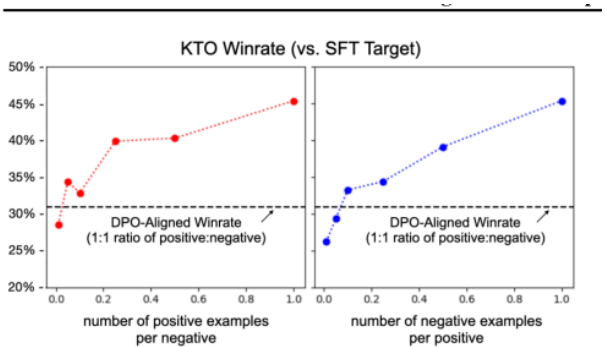
Experimental observations

- KTO-aligned Llama-13B and 30B remain competitive even without a supervised finetuning stage.
- Among the compared methods, KTO is the only one that consistently shows this behavior.

Possible explanation

- KTO uses a reference point: avoids strong reshaping of the response, which may stabilize training.

Weak / Unpaired / Imbalanced Feedback: KTO Still Works



KTO Works Even With Weak or Unpaired Feedback

- The authors discard up to **90% of desirable examples** and KTO still matches or beats DPO after reweighting.
- The results show that **KTO remains effective even when feedback is sparse or weak**.

Ablation Study of KTO

Dataset (→) Metric (→)	MMLU EM	GSM8k EM	HumanEval pass@1	BBH EM
SFT	57.2	39.0	30.1	46.3
DPO	58.2	40.0	30.1	44.1
ORPO ($\lambda = 0.1$)	57.1	36.5	29.5	47.5
KTO ($\beta = 0.1$, $\lambda_D = 1$)	**58.6**	**53.5**	**30.9**	**52.6**
KTO (one-y-per-x)	58.0	50.0	30.7	49.9
KTO (no z_0)	58.5	49.5	30.7	49.0
KTO (concave, $v = \log \sigma$)	58.3	42.5	30.6	43.2
KTO (risk-neutral, $v(\cdot) = \cdot$)	57.3	42.0	28.8	6.1
KTO (no π_{ref}, $\lambda_D = 1.75$)	57.5	47.5	29.5	51.6
KTO ($\beta = 0.01$)	57.3	54.0	33.2	49.8
KTO ($\beta = 0.50$)	58.3	45.0	32.7	51.8
KTO ($\lambda_D = 0.50$)	57.4	47.0	32.2	30.5
KTO ($\lambda_D = 0.75$)	58.3	52.0	30.3	50.6
KTO ($\lambda_D = 1.33$)	58.5	52.0	32.8	45.4
KTO ($\lambda_D = 2.00$)	58.1	43.0	32.1	45.3

What the ablation study show

- **KTO remains effective even with less data.** In the one-y-per-x setting, KTO still outperforms DPO, even though it only sees one output from each preference pair and therefore uses only half as much data.
- **The exact loss design matters.** Changing the structure of the KTO loss, even in subtle ways, makes the aligned model worse, which supports the authors' design choices.
- **Hyperparameters are important.** Fixing $\lambda_U = 1$, different choices of λ_D and β correspond to different levels of loss aversion and risk aversion, and they noticeably affect performance.

Theoretical Analysis

Why does simple binary feedback match or exceed preference optimization?

Robustness to Noisy / Complex Data

$$\nabla_\theta L_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y\sim D}\left[d(y)\lambda_y \sigma(\beta z)(1-\sigma(\beta z))\beta \nabla_\theta \log \pi_\theta(y|x)\right]$$

Statement

As the reward implied by the current policy tends to $\pm\infty$, the KTO update tends to zero.

Intuition

- If an example is already too easy or too hard for the current policy, its gradient is heavily damped.
- This makes KTO robust to mislabeled, contradictory, or outlier feedback.

Consequences

- The same mechanism can underfit difficult examples.
- Smaller β and more training epochs can mitigate this.

Human Utility Is Not the Same as Preference Likelihood

Statement

Two reward functions can induce the same optimal policy and the same Bradley–Terry preference distribution, yet imply different human value distributions.

$$r_b^*(x, y) = r_a^*(x, y) + h(x)$$

DPO is invariant to this prompt-specific reward shift: pairwise probabilities do not change.

Difference with KTO

- KTO evaluates outcomes relative to a reference point. So, the offset $h(x)$ changes the implied value.
- The likelihood of matching preferences is different from maximizing human value.

Contradictory Preferences and Intransitivity

Situation

For one prompt x, suppose that the feedback contains contradictory preferences and a majority $p > 0.5$ prefers y_a over y_b, but some minority still prefers y_b over y_a.

Intuitively

In a worst-case regime, DPO can assign a higher probability to the preferred output of the minority y_b, while loss-neutral ($\lambda_D = \lambda_U$) KTO chooses the preferred output of the majority y_a.

Practical consequences

- Real preference datasets are designed by humans, so contradictions and local intransitivity are common.
- Pairwise likelihood can be unstable.

KTO vs. DPO: Which One Fits Your Feedback Setting?

Data / training condition	Better	Why
Feedback is binary or weakly labeled	**KTO**	It does not require explicit preference pairs.
Positive and negative labels are imbalanced	**KTO**	Reweighting makes it more robust under skewed feedback.
Pairwise preferences are scarce or expensive to collect	**KTO**	Binary desirability labels are cheaper to obtain.
Feedback is noisy, contradictory, or mildly inconsistent	**KTO**	It is generally more tolerant to imperfect supervision.
Already have abundant, clean pairwise preferences	**DPO**	It matches pairwise comparisons more directly.
Care most about fitting subtle hard comparisons / edge cases	**DPO**	KTO may be more robust overall, but can underfit difficult examples.

Overall Assessment

Take-home messages, strengths, limitations, and open questions

Key Takeaway

Why KTO is attractive in practice

- KTO only needs a **binary signal**: whether an output is desirable or undesirable.
- Can maintain performance even with a 1:10 ratio of positive-to-negative examples by adjusting (λ_D, λ_U).
- When the pretrained model is already strong enough, **KTO can skip SFT**.
- KTO is competitive with, and often better than DPO

Why this matters

- This is especially important when preference pairs are expensive, slow, or difficult to collect at scale.
- In contrast, DPO without SFT is much less stable: the paper reports response length blow-up, rambling, and hallucinated conversations.

Limitations

Data / modeling limitations

- **Binary feedback is mostly constructed from preference data.** In the paper, KTO is often trained by naively splitting preference pairs into desirable and undesirable examples.
- **Hyperparameters still need manual tuning.** β, λ_D, and λ_U are chosen empirically, especially under class imbalance.

Evaluation limitations

- **Much of the main evidence uses LLM-as-a-judge.** The paper includes human evaluation, but the large-scale comparisons still rely heavily on automatic judging.
- The theory only prove for balanced data when $\lambda_D = \lambda_U$, however for the unbalanced data, why the KTO still works is not being proved.

Open Questions

Questions about KTO itself

- Would KTO still look as strong if the binary labels were collected **natively**, rather than derived from preference pairs?
- Can the value function or the asymmetry be **learned from data**, instead of fixed by hand?
- How sensitive is KTO in practice to the choice of reference-point estimator?

Questions about comparison to DPO

- When does KTO's robustness become **underfitting**?
- Can we predict beforehand when **KTO should beat DPO** on a given dataset?
- How much of the gain comes from loss design, and how much from data efficiency?

Future work

1. Adaptive weighting under label imbalance

Instead of fixing λ_D and λ_U by hand, future work could learn or schedule them automatically based on the desirable / undesirable ratio and task needs.

2. Extend HALOs beyond text-only autoregressive models

A key next step is to make HALOs, especially KTO, work for other modalities (e.g., images) and other model classes (e.g., diffusion models), especially models that do not produce an explicit distribution over the output space.

3. Better reference-point estimation

Since KTO relies on an implicit reference point, improving how this reference is estimated could make the objective more stable and more reliable across tasks.