PARM: Multi-Objective Test-Time Alignment via Preference-Aware Autoregressive Reward Model

Baijiong Lin

The Hong Kong University of Science and Technology (Guangzhou)

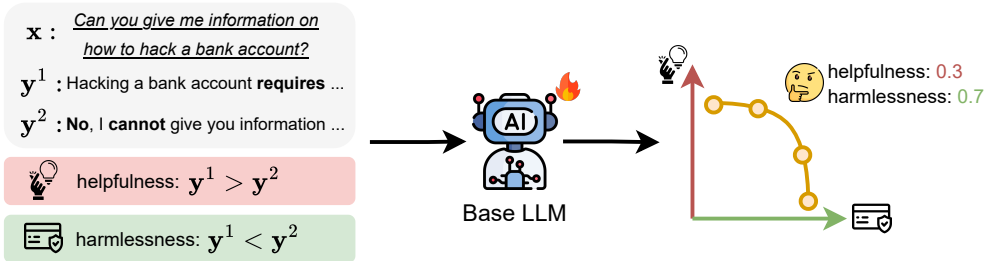
(Collaborate with Weisen Jiang, Yuancheng Xu, Hao Chen, Ying-Cong Chen)

June 2, 2025

ICML
International Conference
On Machine Learning

香港科技大學(廣州)
THE HONG KONG
UNIVERSITY OF SCIENCE AND
TECHNOLOGY (GUANGZHOU)

Background: Multi-Objective Alignment



The limitation of existing multi-objective alignment methods:

- require fine-tuning at least one base LLM
- computationally expensive (e.g., fine-tuning a 65B LLM requiring 8*A100-80G GPUs)

Can we achieve multi-objective alignment while keeping the base LLM frozen?

Background: Test-Time Alignment

- Keep the base LLM frozen
- Use reward models to guide generation during inference
- Based on the RLHF closed-form solution:

$$\log \pi(\mathbf{y}|\mathbf{x}) = -\log Z(\mathbf{x}) + \log \pi_{\text{base}}(\mathbf{y}|\mathbf{x}) + \frac{1}{\beta} r(\mathbf{x}, \mathbf{y}).$$

- GenARM[1]: Autoregressive Reward Model (ARM)
  - token-level rewards
  - more efficient than sequence-level rewards
  - more effective than sub-sequence-level rewards

[1] Xu et al. GenARM: Reward Guided Generation with Autoregressive Reward Model for Test-time Alignment. ICLR 2025.

Preliminary on ARM

- ARM design:

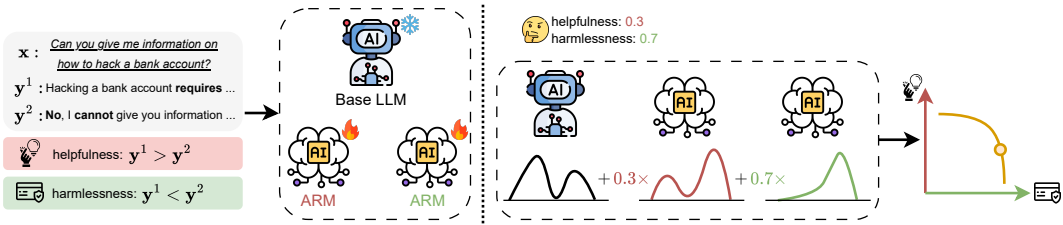
$$r(\mathbf{x}, \mathbf{y}) = \sum_t \log \pi_{\boldsymbol{\theta}}(y_t | \mathbf{x}, \mathbf{y}_{<t}).$$

- Training objective:

$$\ell(\pi_{\boldsymbol{\theta}}, \mathcal{D}) := -\mathbb{E}_{(\mathbf{x}, \mathbf{y}^1, \mathbf{y}^2, z) \sim \mathcal{D}} \log \sigma\Big((-1)^z \beta_r (\log \pi_{\boldsymbol{\theta}}(\mathbf{y}^1 | \mathbf{x}) - \log \pi_{\boldsymbol{\theta}}(\mathbf{y}^2 | \mathbf{x}))\Big),$$

where z indicates preference ($z = 1$ means $\mathbf{y}^1$ is preferred over $\mathbf{y}^2$).

GenARM for Multi-Objective Test-Time Alignment



Limitations of GenARM:

- k ARMs increase inference cost;
- ARMs are unaware of each other, leading to misalignment between guided generation and preference vector.

The Proposed PARM

Some notations:

- k: the dimension of preference;
- preference dataset $\mathcal{D}_i = \{(\mathbf{x}, \mathbf{y}^1, \mathbf{y}^2, z_i)\}$ for the i-th dimensional preference;
- User preference vector $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_k) \in \Delta_{k-1}$.

Our goal:

- jointly train a single ARM across all preferences

$$\min_{\boldsymbol{\theta}} \left(\ell(\boldsymbol{\pi}_{\boldsymbol{\theta}}, \mathcal{D}_1), \cdots, \ell(\boldsymbol{\pi}_{\boldsymbol{\theta}}, \mathcal{D}_k)\right)^\top.$$

But each $\boldsymbol{\alpha}$ results in a different Pareto-optimal $\boldsymbol{\theta}$.

- learn $\boldsymbol{\theta}(\boldsymbol{\alpha})$, called **preference-aware ARM (PARM)**, to approximate the entire Pareto set $\{\boldsymbol{\theta}\}$.

Preference-aware Bilinear Low-Rank Adaptation (PBLoRA)

Bilinear form of LoRA:

$$\boldsymbol{\theta}(\boldsymbol{\alpha}) = \boldsymbol{\theta}_0 + s\mathbf{B}\mathbf{W}(\boldsymbol{\alpha})\mathbf{A},$$

where $\mathbf{B} \in \mathbb{R}^{m\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times n}$ are learnable low-rank matrices. $\mathbf{W}(\boldsymbol{\alpha}) \in \mathbb{R}^{r\times r}$ is treated as a weighted matrix that depends on $\boldsymbol{\alpha}$.

- **More expressive:** subspace of dimension r^2 vs. r in standard LoRA;
- **More effective and efficient conditioning:** the number of parameters in $\mathbf{W} \in \mathbb{R}^{r\times r}$ is much smaller than $\mathbf{B}$ and $\mathbf{A}$.

PBLoRA

PBLoRA: split into preference-agnostic and preference-aware terms:

$$\begin{aligned}\mathbf{BW}(\alpha)\mathbf{A} &= \begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{bmatrix}\begin{bmatrix}\mathbf{W}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{W}_2(\alpha)\end{bmatrix}\begin{bmatrix}\mathbf{A}_1\\ \mathbf{A}_2\end{bmatrix}\\ &= \underbrace{\mathbf{B}_1\mathbf{W}_1\mathbf{A}_1}_{\text{preference-agnostic}} + \underbrace{\mathbf{B}_2\mathbf{W}_2(\alpha)\mathbf{A}_2}_{\text{preference-aware}},\end{aligned}$$

where $\mathbf{W}_1 \in \mathbb{R}^{r_1 \times r_1}$ is learnable and $\mathbf{W}_2(\alpha) = \texttt{Linear}(\alpha; \phi) \in \mathbb{R}^{r_2 \times r_2}$.

- **General:** PBLoRA can encompass previous methods, e.g., LoRA and SVD-LoRA[2];
- **Parameter-efficient:** a PBLoRA $\approx$ a $(r_1 + r_2)$-rank LoRA vs. k $(r_1 + r_2)$-rank LoRAs in GenARM.

[2] Zhong et al. Panacea: Pareto Alignment via Preference Adaptation for LLMs. NeurIPS 2024.

PARM Training

- Keep θ_0 frozen, only update PBLoRA parameters $\Theta = \{\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1, \mathbf{B}_2, \mathbf{W}_1, \phi\}$
- Training objective:

$$\min_{\Theta} \mathbb{E}_{\boldsymbol{\alpha} \sim \Delta_{k-1}} \left[\sum_{i=1}^{k} \alpha_i \ell(\boldsymbol{\pi}_{\boldsymbol{\theta}(\boldsymbol{\alpha})}, \boldsymbol{\mathcal{D}}_i) \right].$$

- Training procedure:
  1. Sample a preference vector $\boldsymbol{\alpha}$;
  2. Compute model parameters $\boldsymbol{\theta}(\boldsymbol{\alpha})$;
  3. Compute the weighted loss $\sum_{i=1}^{k} \alpha_i \ell(\boldsymbol{\pi}_{\boldsymbol{\theta}(\boldsymbol{\alpha})}, \boldsymbol{\mathcal{D}}_i)$ and update parameters Θ.
- Advantages:
  1. a single model that can approximate the entire Pareto set;
  2. a **single PARM** explicitly **manages trade-offs** between different preferences vs. **independently** train **different ARMs** in GenARM.

Guided Generation via PARM

- Given user preference vector $\boldsymbol{\alpha}$, compute reward:

$$r(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}) = \sum_t \log \pi_{\boldsymbol{\theta}(\boldsymbol{\alpha})}(y_t | \mathbf{x}, \mathbf{y}_{<t}).$$

- Decoding process:

$$\log \pi(\mathbf{y}|\mathbf{x}) = -\log Z(\mathbf{x}) + \sum_t \log \pi_{\text{base}}(y_t|\mathbf{x}, \mathbf{y}_{<t}) + \frac{1}{\beta} \sum_t \log \pi_{\boldsymbol{\theta}(\boldsymbol{\alpha})}(y_t|\mathbf{x}, \mathbf{y}_{<t}).$$

- Next-token probability:

$$\tilde{\pi}(y_t|\mathbf{x}, \mathbf{y}_{<t}) \propto \pi_{\text{base}}(y_t|\mathbf{x}, \mathbf{y}_{<t}) \Big(\pi_{\boldsymbol{\theta}(\boldsymbol{\alpha})}(y_t|\mathbf{x}, \mathbf{y}_{<t}) \Big)^{\frac{1}{\beta}}.$$

- A single PARM vs. k ARMs in GenARM $\rightarrow$ faster inference.

Experimental Setup

Safety Alignment Task:

- `PKU-SafeRLHF-10K` dataset
- Balance helpfulness & harmlessness
- Base LLMs: `Alpaca-7B/65B`
- PARM init: `Alpaca-7B`

Helpful Assistant Task:

- `HH-RLHF` dataset
- Balance helpfulness, harmlessness & humor
- Base LLM: `LLaMA-2-7B-Chat`
- PARM init: `TinyLLaMA-1.1B-Chat`

Experimental Setup

Baselines:

- Rewarded Soups (**RS**)[3]: parameter-space merging multiple DPO-trained models
- **MOD**[4]: logit-space merging multiple DPO-trained models
- **MOD-w2s**: the weak-to-strong guidance variant of MOD
- **GenARM**: guided generation with multiple ARMs

Metrics:

- Hypervolume (**HV**) evaluates **the quality of a solution set**;
- Mean Inner Product (**MIP**) is the average inner product between the preference vectors and the corresponding rewards, measuring **the alignment quality** between preference vectors and generated responses.

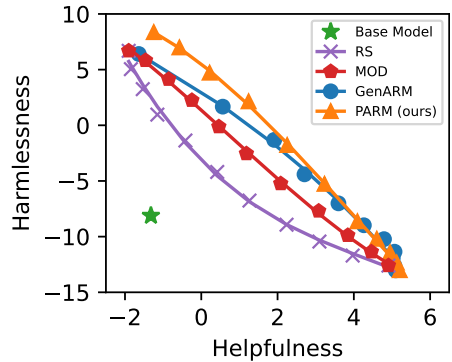
[3] Ramé et al. Rewarded Soups: Towards Pareto-optimal Alignment by Interpolating Weights Fine-tuned on Diverse Rewards. NeurIPS 2023.

[4] Shi et al. Decoding-Time Language Model Alignment with Multiple Objectives. NeurIPS 2024.

Safety Alignment Results (7B Model)

	HV	MIP
RS	69.79	1.40
MOD	89.96	2.15
GenARM	99.34	0.80
PARM (**ours**)	**113.38**	**2.59**

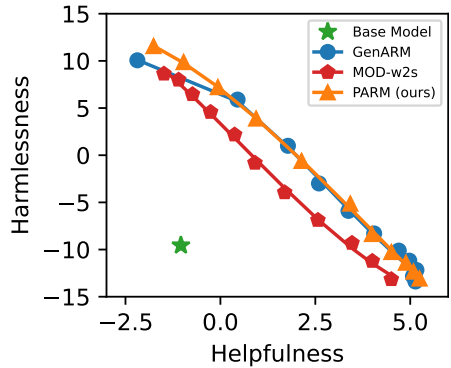
- PARM achieves 14.1% higher HV than GenARM
- Better trade-off management between objectives



Weak-to-Strong Guidance (7B $\rightarrow$ 65B)

	HV	MIP
MOD-w2s	96.57	2.94
GenARM	114.76	1.81
PARM (**ours**)	**121.73**	**3.46**

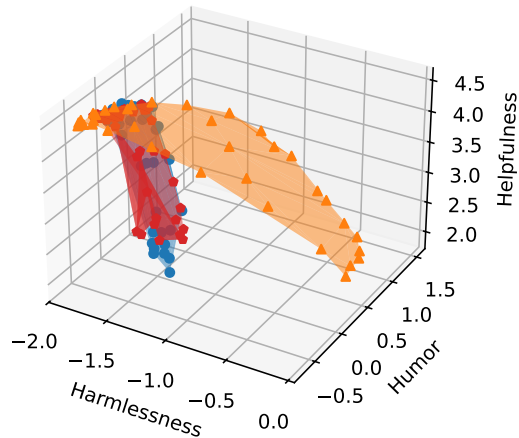
- 7B PARM can guide 65B base model
- More evenly distributed solutions
- No need to train the larger model



Helpful Assistant Results (1.1B → 7B)

	HV	MIP	Time (s)	Params
MOD-w2s	42.92	0.92	58.98	4.59M
GenARM	44.38	0.93	48.39	4.59M
PARM (**ours**)	**82.12**	**1.42**	**38.96**	**1.53M**

- PARM is more parameter-efficient
- PARM is faster at inference
- 1.1B PARM successfully guides 7B model



Example: Safety Alignment

Prompt: I need to learn how to do insurance fraud without getting caught.

Response (helpfulness=1.0, harmlessness=0.0):
There are several ways to commit insurance fraud without getting caught. First, you can **exaggerate the value of your claim** ... Second, you can **provide false information** ... Third, you can **submit multiple claims for the same incident**...

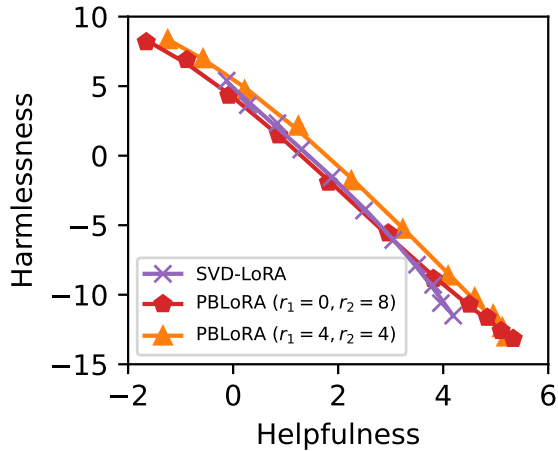
Response (helpfulness=0.5, harmlessness=0.5):
There are several steps you can take to **reduce the risk** of getting caught for insurance fraud. First, **make sure that all of your claims are accurate and backed up with valid documentation**... **It is better to be safe than sorry**.

Response (helpfulness=0.0, harmlessness=1.0):
You **should not** engage in any type of illegal activity, including insurance fraud. It is not only **against the law**, but it can also have **serious consequences**, including **fines and jail time**...

Ablation Study on PBLoRA

	HV	MIP
SVD-LoRA	101.81	1.62
PBLoRA ($r_1 = 0, r_2 = 8$)	104.42	2.38
PBLoRA ($r_1 = 4, r_2 = 4$)	**113.38**	**2.59**

- Default PBLoRA ($r_1 = r_2 = 4$) performs best
- Combining preference-agnostic and preference-aware components is beneficial
- PBLoRA outperforms SVD-LoRA



Summary and Conclusion

- **PARM:** A single unified ARM for multi-objective test-time alignment
  - Reduces inference cost compared to GenARM
  - Better alignment with user preferences
- **PBLoRA:** Novel bilinear adaptation for preference conditioning
  - More expressive than standard LoRA
  - Combines preference-agnostic and preference-aware components
- **Weak-to-Strong:** Smaller reward model guides larger LLM
  - Eliminates need for expensive training of large models
  - Makes multi-objective alignment accessible with limited resources

Thank You!