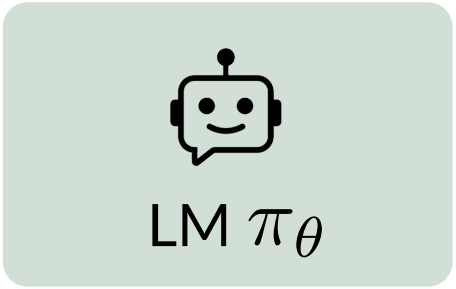
What Makes a Good Proxy Reward Function for Language Model Post-Training?

Noam Razin
Princeton Language and Intelligence, Princeton University

Language Models (LMs)

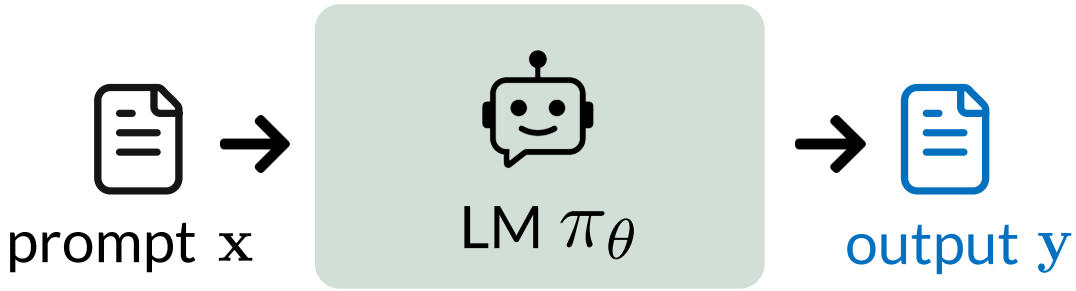
Language Model (LM):
Neural network that produces a **distribution over token sequences**



θ - Parameters

Language Models (LMs)

Language Model (LM):
Neural network that produces a **distribution over token sequences**



Typical LM Development Pipeline

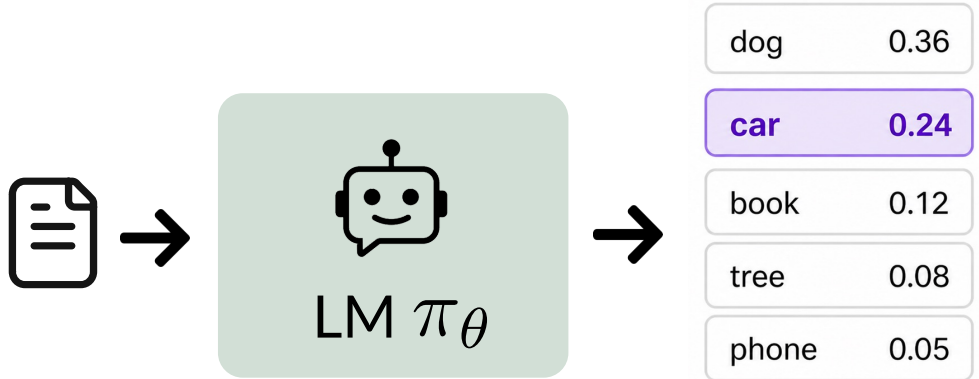
Typical LM Development Pipeline

Pre-Training: Endowing LM with world and language knowledge

Typical LM Development Pipeline

Pre-Training: Endowing LM with world and language knowledge

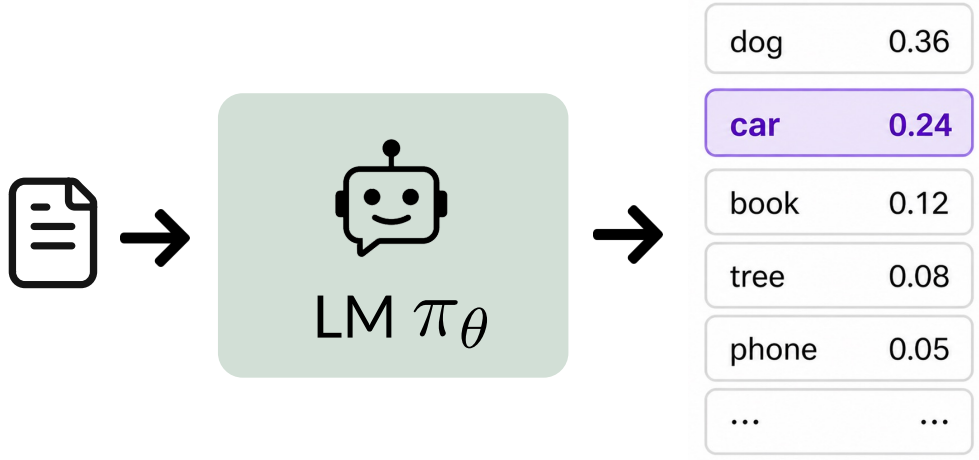
Approach: Training LM to predict the next token on vast amounts of data



Typical LM Development Pipeline

Pre-Training: Endowing LM with world and language knowledge

Approach: Training LM to predict the next token on vast amounts of data

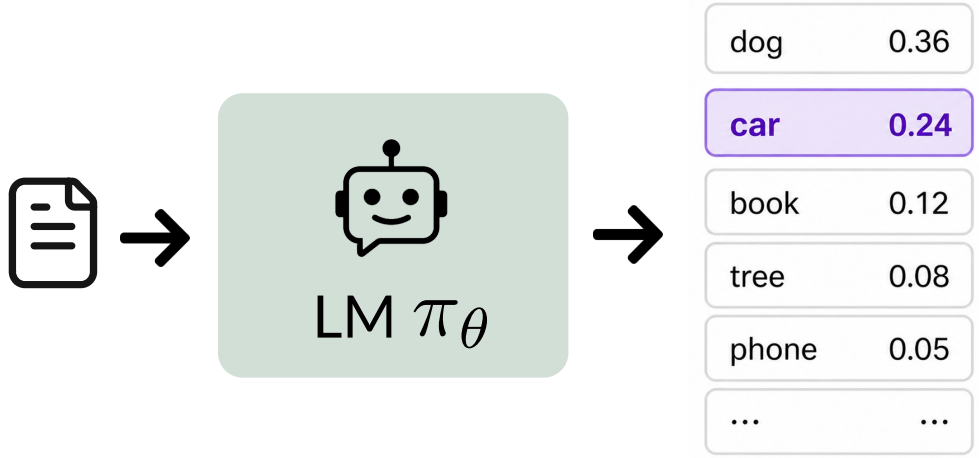


Post-Training: Converting the LM to a useful AI assistant/system

Typical LM Development Pipeline

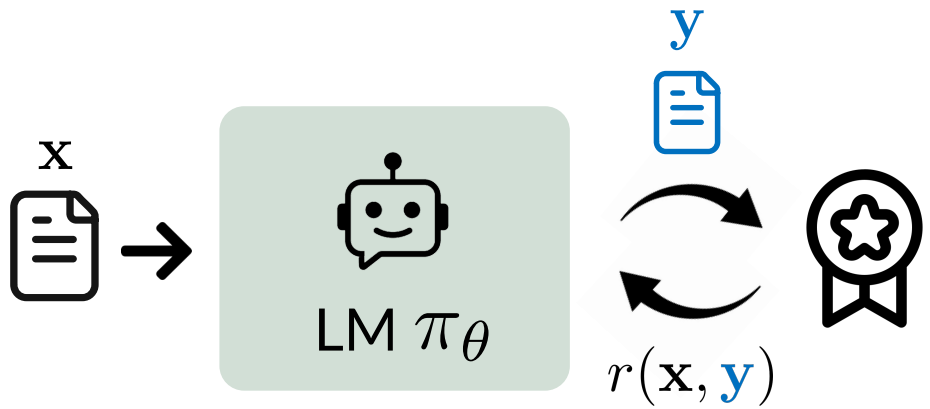
Pre-Training: Endowing LM with world and language knowledge

Approach: Training LM to predict the next token on vast amounts of data



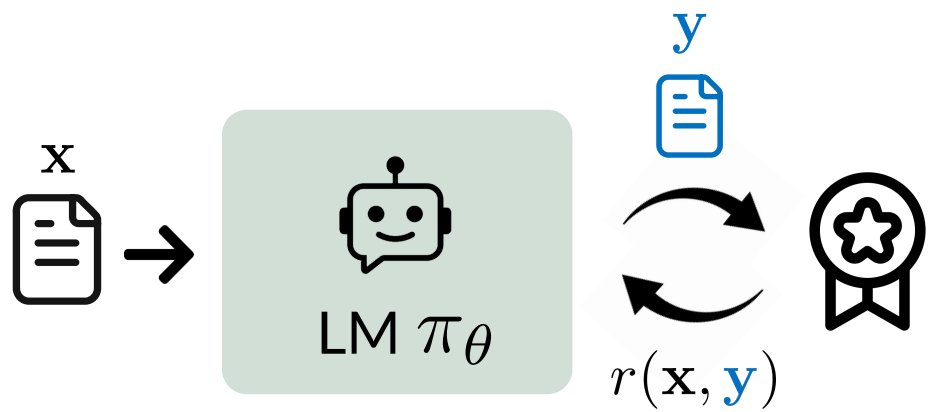
Post-Training: Converting the LM to a useful AI assistant/system

Major Component: Reinforcement learning



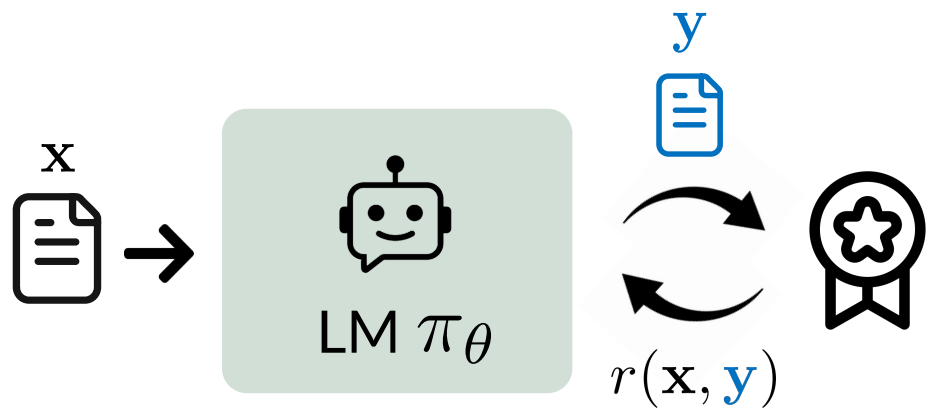
Policy Gradient

Policy gradient is the standard reinforcement learning method for LMs



Policy Gradient

Policy gradient is the standard reinforcement learning method for LMs



Maximize expected reward via **gradient-based updates** over set of prompts $\mathcal{S}$

$$V(\theta) := \mathbb{E}_{\mathbf{x}\sim\mathcal{S},\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\left[r(\mathbf{x}, \mathbf{y})\right]$$

aka **policy**

Imperfect Proxy Rewards Are Everywhere

Imperfect Proxy Rewards Are Everywhere

Goal: Maximize **ground truth (GT) reward** r_{GT} that precisely defines the intended objective

Imperfect Proxy Rewards Are Everywhere

Goal: Maximize **ground truth (GT) reward** r_{GT} that precisely defines the intended objective

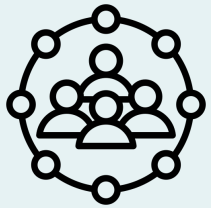
Reality: It is rarely possible to specify such GT rewards → use **proxy reward** r_{P}

Imperfect Proxy Rewards Are Everywhere

Goal: Maximize **ground truth (GT) reward** r_{GT} that precisely defines the intended objective

Reality: It is rarely possible to specify such GT rewards → use **proxy reward** r_{P}

Alignment (RLHF)



GT reward: Governs human preferences

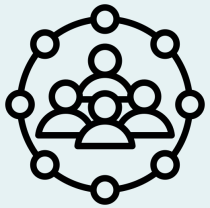
Proxy reward: Learned reward models

Imperfect Proxy Rewards Are Everywhere

Goal: Maximize **ground truth (GT) reward** r_{GT} that precisely defines the intended objective

Reality: It is rarely possible to specify such GT rewards → use **proxy reward** r_{P}

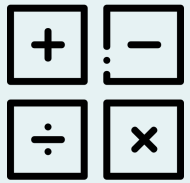
Alignment (RLHF)



GT reward: Governs human preferences

Proxy reward: Learned reward models

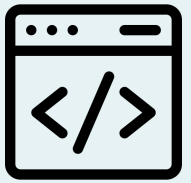
Math



GT reward: Should verify the correctness of the full output

Proxy reward: Only verifies final answer & has parsing issues

Coding



GT reward: Should verify correct functionality

Proxy reward: Uses incomplete unit tests

What Makes a Good Proxy Reward Function?

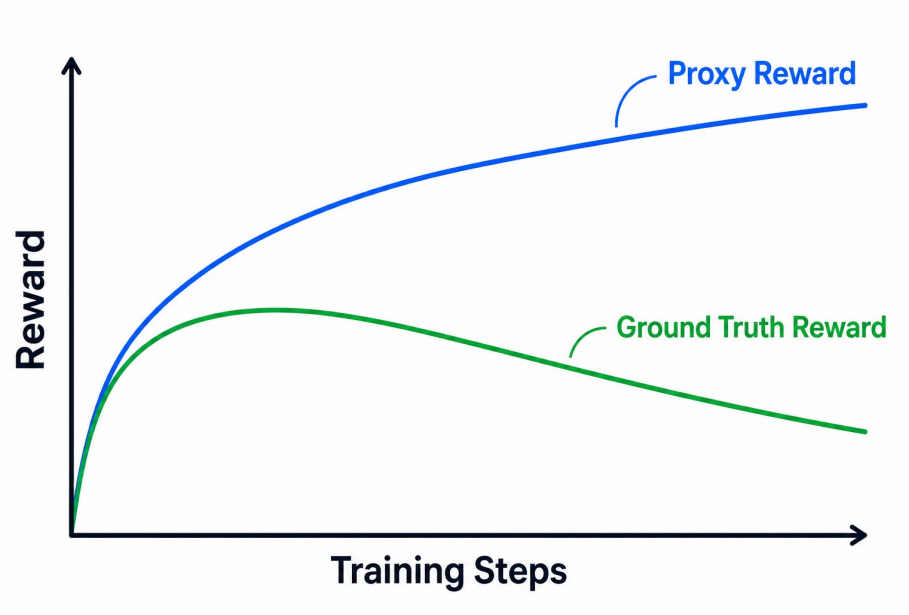
The understanding of what makes a good proxy reward function is limited

What Makes a Good Proxy Reward Function?

The understanding of what makes a good proxy reward function is limited

Well-known risk of using proxy rewards: **reward hacking**

Randløv and Alstrøm 1998, Ng et al. 1999, Amodei et al. 2016, Skalse et al. 2022, Pang et al. 2023, Gao et al. 2023, Karwowski et al. 2023, Fluri et al. 2025, Laidlaw et al. 2025



Evaluation of Proxy Rewards

Due to the risk of reward hacking, proxy rewards are evaluated by **deviation from GT reward**

Evaluation of Proxy Rewards

Due to the risk of reward hacking, proxy rewards are evaluated by **deviation from GT reward**

Underlying Assumption: Deviations are harmful

Evaluation of Proxy Rewards

Due to the risk of reward hacking, proxy rewards are evaluated by **deviation from GT reward**

Underlying Assumption: Deviations are harmful

Accuracy: Primary measure for evaluating reward models in LM alignment (RLHF)

Evaluation of Proxy Rewards

Due to the risk of reward hacking, proxy rewards are evaluated by **deviation from GT reward**

Underlying Assumption: Deviations are harmful

Accuracy: Primary measure for evaluating reward models in LM alignment (RLHF)

$$\mathcal{D} = \left\{ \mathbf{x},\ \mathbf{y}^{+},\ \mathbf{y}^{-} \right\},\ r_{\mathrm{G}}(\mathbf{x}, \mathbf{y}^{+}) > r_{\mathrm{G}}(\mathbf{x}, \mathbf{y}^{-})$$

Evaluation of Proxy Rewards

Due to the risk of reward hacking, proxy rewards are evaluated by **deviation from GT reward**

Underlying Assumption: Deviations are harmful

Accuracy: Primary measure for evaluating reward models in LM alignment (RLHF)

$$\mathcal{D} = \left\{ \mathbf{x},\ \mathbf{y}^+,\ \mathbf{y}^- \right\},\ r_{\mathrm{G}}(\mathbf{x}, \mathbf{y}^+) > r_{\mathrm{G}}(\mathbf{x}, \mathbf{y}^-)$$

$$\mathrm{Acc}(r_{\mathrm{P}}) := \sum_{(\mathbf{x},\mathbf{y}^+,\mathbf{y}^-)\in\mathcal{D}} \frac{\mathbb{1}[r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}^+) > r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}^-)]}{|\mathcal{D}|}$$

Evaluation of Proxy Rewards

Due to the risk of reward hacking, proxy rewards are evaluated by **deviation from GT reward**

Underlying Assumption: Deviations are harmful

Accuracy: Primary measure for evaluating reward models in LM alignment (RLHF)

$$\mathcal{D} = \left\{ \mathbf{x}, \mathbf{y}^+, \mathbf{y}^- \right\}, \; r_{\mathrm{G}}(\mathbf{x}, \mathbf{y}^+) > r_{\mathrm{G}}(\mathbf{x}, \mathbf{y}^-)$$

$$\mathrm{Acc}(r_{\mathrm{P}}) := \sum_{(\mathbf{x},\mathbf{y}^+,\mathbf{y}^-)\in\mathcal{D}} \frac{\mathbb{1}\left[r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}^+) > r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}^-)\right]}{|\mathcal{D}|}$$

REWARDBENCH Lambert et al. 2024

	Model	Score
1	infly/INF-ORM-Llama3.1-70B	95.1
2	ShikaiChen/LDL-Reward-Gemma-2-27B-v0.1	95.0
3	nicolinho/QRM-Gemma-2-27B	94.4

Are More Accurate Proxy Rewards Better?

Q: Are more accurate proxy rewards necessarily better?

Are More Accurate Proxy Rewards Better?

Q: Are more accurate proxy rewards necessarily better? ❌

Are More Accurate Proxy Rewards Better?

Q: Are more accurate proxy rewards necessarily better? ❌

We take an optimization perspective: How fast does the GT reward increase when maximizing the proxy reward via policy gradient?

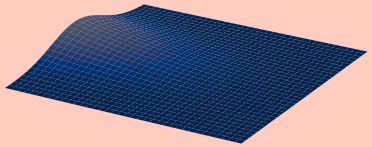
Are More Accurate Proxy Rewards Better?

Q: Are more accurate proxy rewards necessarily better? ❌

We take an optimization perspective: How fast does the GT reward increase when maximizing the proxy reward via policy gradient?

Limitation 1

An accurate proxy reward function can still induce a **poor objective landscape**



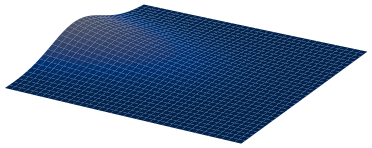
Limitation 2

Incorrect rewards are **not equally harmful**

Outline

1

Role of Reward Variance



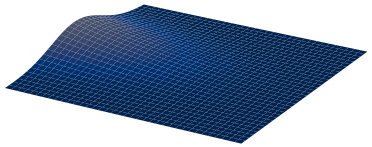
2

Not All Reward Errors Are Harmful

Outline

1

Role of Reward Variance



2

Not All Reward Errors Are Harmful

Vanishing Gradients in Reinforcement Finetuning of Language Models

R + Zhou + Saremi + Thilak + Bradley + Nakkiran + Susskind + Littwin | *ICLR 2024*

What Makes a Reward Model a Good Teacher? An Optimization Perspective

R + Wang + Strauss + Wei + Lee + Arora | *NeurIPS 2025*

Theoretical Analysis Setting

Theoretical Analysis Setting

General Autoregressive Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{|\mathbf{y}|} \pi_\theta(\mathbf{y}_l|\mathbf{x}, \mathbf{y}_{<l}) = \prod_{l=1}^{|\mathbf{y}|} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l}))_{\mathbf{y}_l}$$

↑ parameterized function (e.g., neural network)

Theoretical Analysis Setting

General Autoregressive Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{|\mathbf{y}|} \pi_\theta(\mathbf{y}_l|\mathbf{x}, \mathbf{y}_{<l}) = \prod_{l=1}^{|\mathbf{y}|} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l}))_{\mathbf{y}_l}$$

↑ parameterized function (e.g., neural network)

Analyzed In: Lower bound on optimization time

Theoretical Analysis Setting

General Autoregressive Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{|\mathbf{y}|} \pi_\theta(\mathbf{y}_l|\mathbf{x}, \mathbf{y}_{<l}) = \prod_{l=1}^{|\mathbf{y}|} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l}))_{\mathbf{y}_l}$$

↑ parameterized function (e.g., neural network)

Analyzed In: Lower bound on optimization time

Linear Softmax Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(\Phi_{\mathbf{x}}\theta)_{\mathbf{y}}$$

↑ output features matrix (for simplicity, has orthonormal rows)

Theoretical Analysis Setting

General Autoregressive Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{|\mathbf{y}|} \pi_\theta(\mathbf{y}_l|\mathbf{x}, \mathbf{y}_{<l}) = \prod_{l=1}^{|\mathbf{y}|} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l}))_{\mathbf{y}_l}$$

↑ parameterized function (e.g., neural network)

Analyzed In: Lower bound on optimization time

Linear Softmax Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(\Phi_{\mathbf{x}}\theta)_{\mathbf{y}}$$

↑ output features matrix (for simplicity, has orthonormal rows)

Analyzed In: Upper bounds on optimization time

Theoretical Analysis Setting

General Autoregressive Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{|\mathbf{y}|} \pi_\theta(\mathbf{y}_l|\mathbf{x}, \mathbf{y}_{<l}) = \prod_{l=1}^{|\mathbf{y}|} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l}))_{\mathbf{y}_l}$$

↑ parameterized function (e.g., neural network)

Analyzed In: Lower bound on optimization time

Linear Softmax Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(\Phi_{\mathbf{x}}\theta)_{\mathbf{y}}$$

↑ output features matrix (for simplicity, has orthonormal rows)

Analyzed In: Upper bounds on optimization time

*objective is non-concave and its optimization is not well understood (e.g., Agarwal et al. 2019, Lin et al. 2025)

Theoretical Analysis Setting

General Autoregressive Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{|\mathbf{y}|} \pi_\theta(\mathbf{y}_l|\mathbf{x}, \mathbf{y}_{<l}) = \prod_{l=1}^{|\mathbf{y}|} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l}))_{\mathbf{y}_l}$$

↑ parameterized function (e.g., neural network)

Analyzed In: Lower bound on optimization time

Linear Softmax Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(\Phi_{\mathbf{x}}\theta)_{\mathbf{y}}$$

↑ output features matrix (for simplicity, has orthonormal rows)

Analyzed In: Upper bounds on optimization time

*objective is non-concave and its optimization is not well understood (e.g., Agarwal et al. 2019, Lin et al. 2025)

Training: Maximize proxy reward via **gradient flow** $\frac{d}{dt}\theta_t = \nabla V_{\mathrm{P}}(\theta_t)$

Theoretical Analysis Setting

General Autoregressive Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{|\mathbf{y}|} \pi_\theta(\mathbf{y}_l|\mathbf{x}, \mathbf{y}_{<l}) = \prod_{l=1}^{|\mathbf{y}|} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l}))_{\mathbf{y}_l}$$

↑ parameterized function (e.g., neural network)

Analyzed In: Lower bound on optimization time

Linear Softmax Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(\Phi_{\mathbf{x}}\theta)_{\mathbf{y}}$$

↑ output features matrix (for simplicity, has orthonormal rows)

Analyzed In: Upper bounds on optimization time

*objective is non-concave and its optimization is not well understood (e.g., Agarwal et al. 2019, Lin et al. 2025)

Training: Maximize proxy reward via **gradient flow** $\frac{d}{dt}\theta_t = \nabla V_{\mathrm{P}}(\theta_t)$

Motivation: LM post-training typically uses small learning rates

Theoretical Analysis Setting

General Autoregressive Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{|\mathbf{y}|} \pi_\theta(\mathbf{y}_l|\mathbf{x}, \mathbf{y}_{<l}) = \prod_{l=1}^{|\mathbf{y}|} \mathrm{softmax}(f_\theta(\mathbf{x}, \mathbf{y}_{<l}))_{\mathbf{y}_l}$$

↑ parameterized function (e.g., neural network)

Analyzed In: Lower bound on optimization time

Linear Softmax Policy

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(\Phi_{\mathbf{x}}\theta)_{\mathbf{y}}$$

↑ output features matrix (for simplicity, has orthonormal rows)

Analyzed In: Upper bounds on optimization time

*objective is non-concave and its optimization is not well understood (e.g., Agarwal et al. 2019, Lin et al. 2025)

Training: Maximize proxy reward via **gradient flow** $\frac{d}{dt}\theta_t = \nabla V_{\mathrm{P}}(\theta_t)$

Motivation: LM post-training typically uses small learning rates

Real Goal of Training: Maximize GT reward $V_{\mathrm{GT}}(\theta_t)$

Reward Variance

Definition: Reward Variance

The reward variance that r_{P} induces for π_θ and $\mathbf{x}$ is:

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]$$

Reward Variance

Definition: Reward Variance

The reward variance that r_{P} induces for π_θ and $\mathbf{x}$ is:

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})] := \mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\left[\left(r_{\mathrm{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{y}'\sim\pi_\theta(\cdot|\mathbf{x})}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y}')\right]\right)^2\right]$$

Reward Variance

Definition: Reward Variance

The reward variance that r_{P} induces for π_θ and $\mathbf{x}$ is:

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})] := \mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\left[\left(r_{\mathrm{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{y}'\sim\pi_\theta(\cdot|\mathbf{x})}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y}')\right]\right)^2\right]$$

Interpretation: Reward variance measures how well r_{P} separates outputs that are probable under π_θ

Reward Variance

Definition: Reward Variance

The reward variance that r_{P} induces for π_θ and $\mathbf{x}$ is:

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})] := \mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\left[\left(r_{\mathrm{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{y}'\sim\pi_\theta(\cdot|\mathbf{x})}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y}')\right]\right)^2\right]$$

Interpretation: Reward variance measures how well r_{P} separates outputs that are probable under π_θ

In Contrast: Accuracy depends only on how r_{P} ranks different outputs

Low Reward Variance Implies Slow Reward Maximization

Low Reward Variance Implies Slow Reward Maximization

Theorem

The time it takes for $V_{\mathrm{P}}(\theta_t)$ and $V_{\mathrm{GT}}(\theta_t)$ to increase by any desired amount is:

Low Reward Variance Implies Slow Reward Maximization

Theorem

The time it takes for $V_{\mathrm{P}}(\theta_t)$ and $V_{\mathrm{GT}}(\theta_t)$ to increase by any desired amount is:

$$\Omega\left(\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}\left[\mathrm{Var}_{\mathbf{y}\sim\pi_{\theta_0}(\cdot|\mathbf{x})}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})\right]\right]^{-\frac{1}{3}}\right)$$

Low Reward Variance Implies Slow Reward Maximization

Theorem

The time it takes for $V_{\mathrm{P}}(\theta_t)$ and $V_{\mathrm{GT}}(\theta_t)$ to increase by any desired amount is:

$$\Omega\left(\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}\left[\mathrm{Var}_{\mathbf{y}\sim\pi_{\theta_0}(\cdot|\mathbf{x})}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})\right]\right]^{-\frac{1}{3}}\right)$$

Proof Sketch: $\nabla V_{\mathrm{P}}(\theta)$ vanishes when $\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}\left[\mathrm{Var}_{\mathbf{y}\sim\pi_{\theta}(\cdot|\mathbf{x})}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})\right]\right]$ is low & cannot grow rapidly

Low Reward Variance Implies Slow Reward Maximization

Theorem

The time it takes for $V_{\mathrm{P}}(\theta_t)$ and $V_{\mathrm{GT}}(\theta_t)$ to increase by any desired amount is:

$$\Omega\left(\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}\left[\mathrm{Var}_{\mathbf{y}\sim\pi_{\theta_0}(\cdot|\mathbf{x})}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})\right]\right]^{-\frac{1}{3}}\right)$$

Proof Sketch: $\nabla V_{\mathrm{P}}(\theta)$ vanishes when $\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}\left[\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]\right]$ is low & cannot grow rapidly

holds for any reinforcement learning setting with softmax policies (not just LMs)

Low Reward Variance Implies Slow Reward Maximization

Theorem

The time it takes for $V_{\mathrm{P}}(\theta_t)$ and $V_{\mathrm{GT}}(\theta_t)$ to increase by any desired amount is:

$$\Omega\left(\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}\left[\mathrm{Var}_{\mathbf{y}\sim\pi_{\theta_0}(\cdot|\mathbf{x})}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})\right]\right]^{-\frac{1}{3}}\right)$$

Proof Sketch: $\nabla V_{\mathrm{P}}(\theta)$ vanishes when $\mathbb{E}_{\mathbf{x}\sim\mathcal{S}}\left[\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]\right]$ is low & cannot grow rapidly

holds for any reinforcement learning setting with softmax policies (not just LMs)

Proxy reward needs to induce sufficient variance for efficient optimization

Implication: More Accurate Proxy Rewards Are Not Always Better

Implication: More Accurate Proxy Rewards Are Not Always Better

Theorem

For any initial policy π_{θ_0}, there exist a **perfectly accurate** r_{per} and **relatively inaccurate** r_{inacc} such that:

Implication: More Accurate Proxy Rewards Are Not Always Better

Theorem

For any initial policy π_{θ_0}, there exist a **perfectly accurate** r_{per} and **relatively inaccurate** r_{inacc} such that:

$V_{\text{GT}}(\theta_t)$ increases **arbitrarily slower**
when training with r_{per} compared to r_{inacc}

Implication: More Accurate Proxy Rewards Are Not Always Better

Theorem

For any initial policy π_{θ_0}, there exist a **perfectly accurate** r_{per} and **relatively inaccurate** r_{inacc} such that:

$V_{\text{GT}}(\theta_t)$ increases **arbitrarily slower**
when training with r_{per} compared to r_{inacc}

*Same holds with almost any accuracy values for the proxy reward functions

Illustration: Effect of Accuracy and Reward Variance

Illustration: Effect of Accuracy and Reward Variance

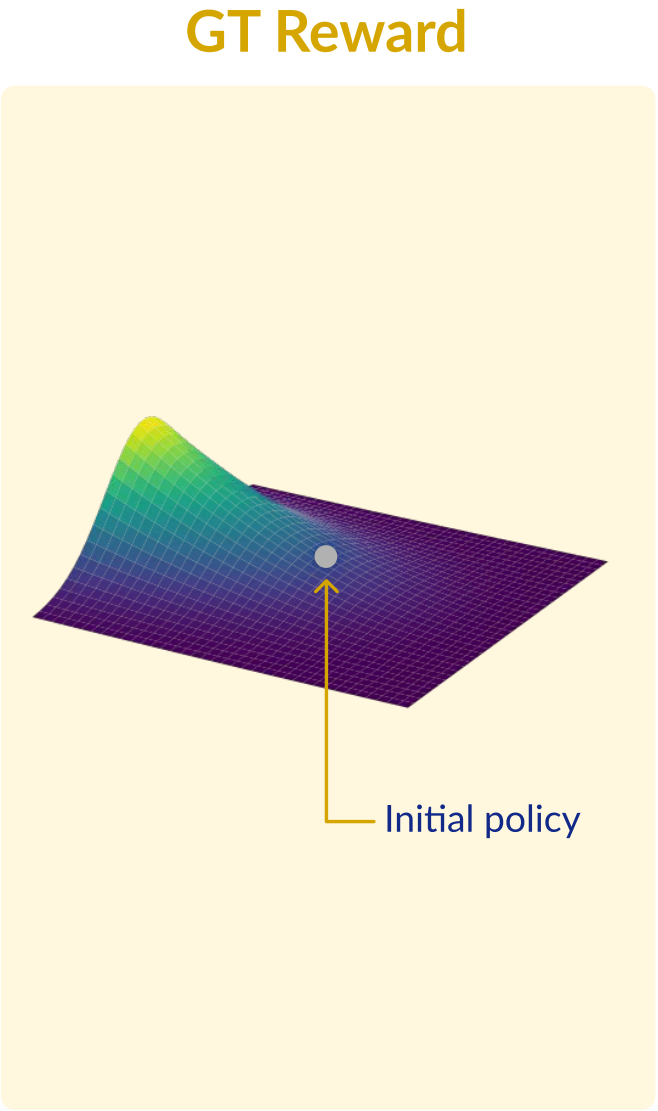
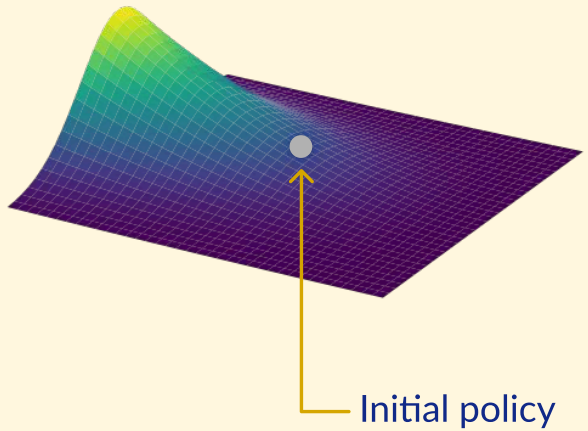


Illustration: Effect of Accuracy and Reward Variance

GT Reward

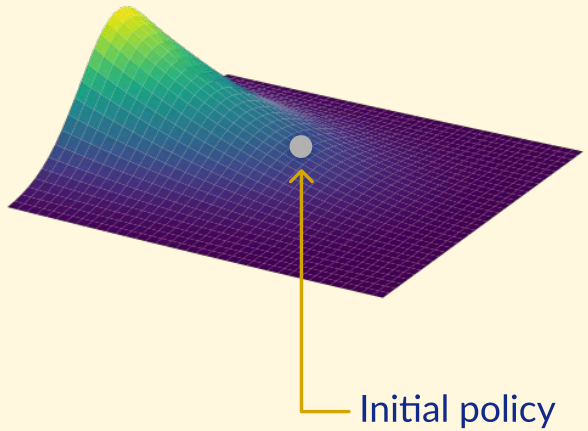


Proxy Reward

Accuracy and reward variance capture distinct aspects

Illustration: Effect of Accuracy and Reward Variance

GT Reward



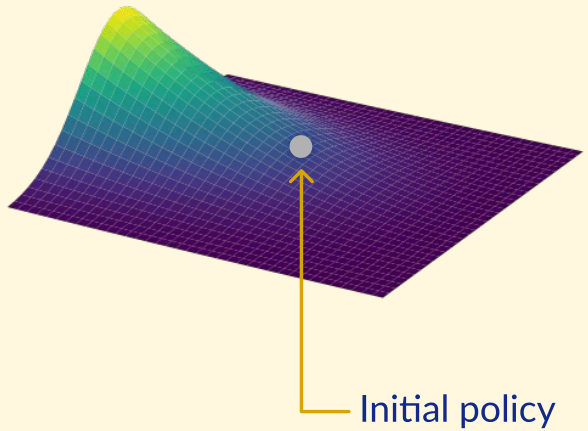
Proxy Reward

Accuracy and reward variance capture distinct aspects

Accuracy

Illustration: Effect of Accuracy and Reward Variance

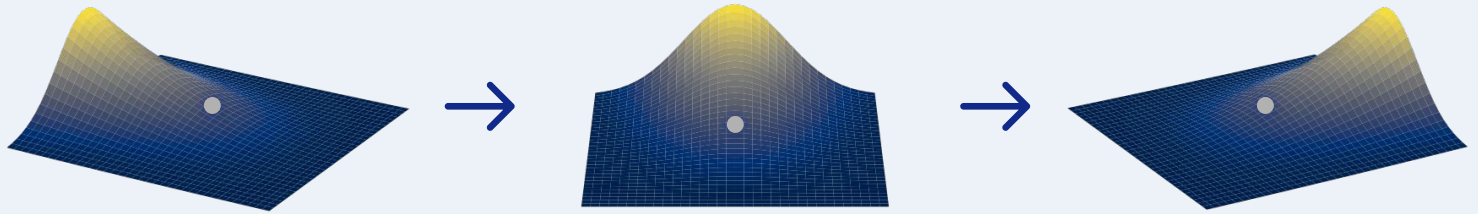
GT Reward



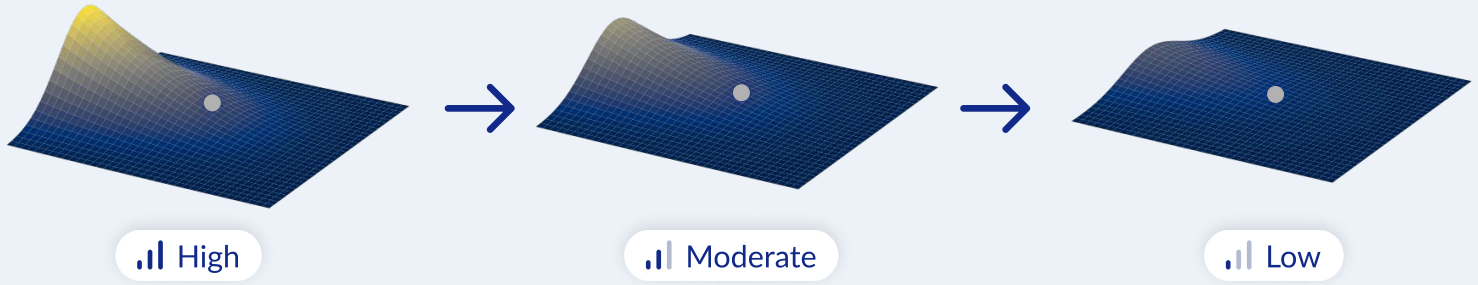
Proxy Reward

Accuracy and reward variance capture distinct aspects

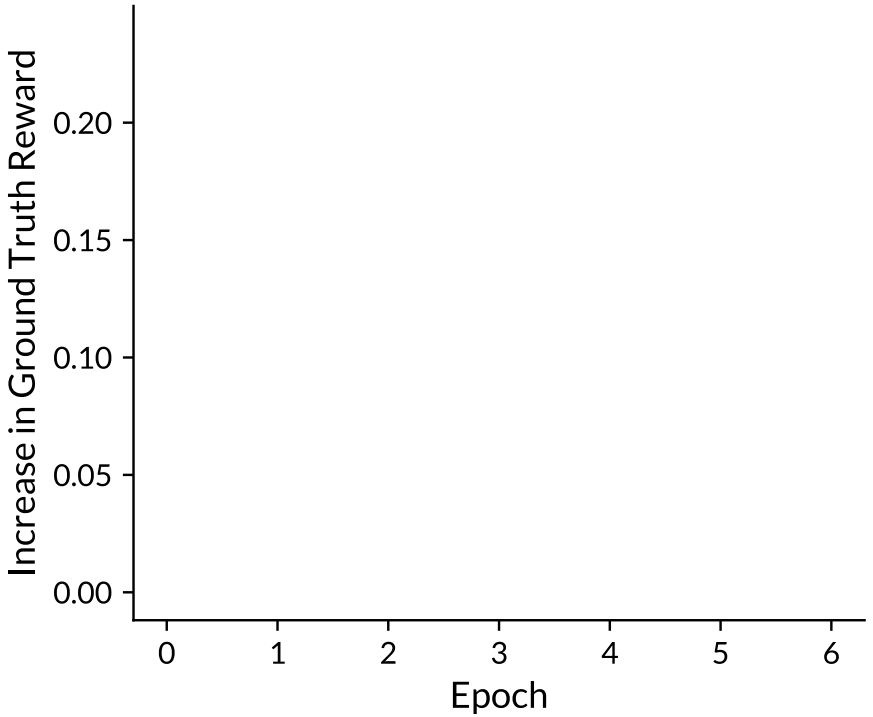
Accuracy



Reward Variance



Experiments: More Accurate Proxy Rewards Are Not Always Better



Reward Used for Training

≈ 70% accuracy

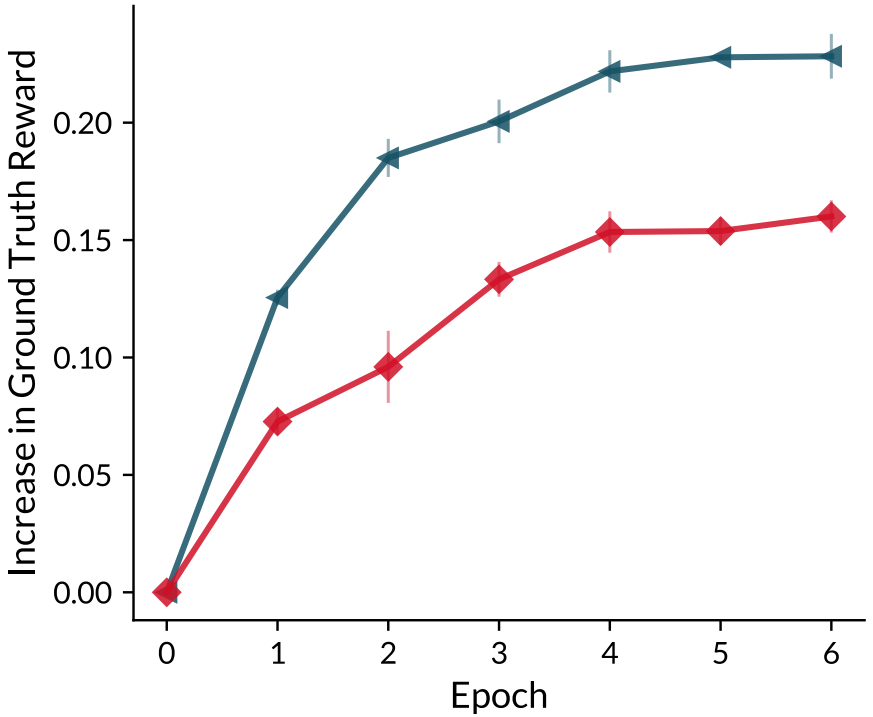
perfect accuracy but low reward variance

Dataset: UltraFeedback

LM: Pythia-2.8B

Chen et al. 2024, Wen et al. 2025: Further experiments showing more accurate proxy rewards are not necessarily better

Experiments: More Accurate Proxy Rewards Are Not Always Better



Reward Used for Training

- ≈ 70% accuracy
- perfect accuracy but low reward variance

Dataset: UltraFeedback

LM: Pythia-2.8B

Even a perfectly accurate proxy rewards can underperform less accurate ones, due to low reward variance

Chen et al. 2024, Wen et al. 2025: Further experiments showing more accurate proxy rewards are not necessarily better

Practical Applications

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]$$

Practical Applications

$$\mathrm{Var}_{\mathbf{y} \sim \pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x}, \mathbf{y})]$$

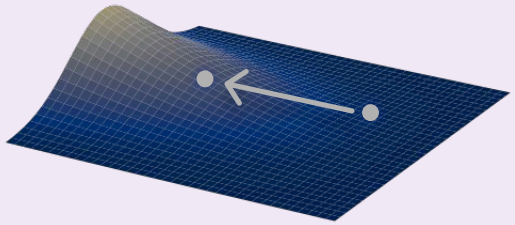
LM prompt proxy reward

Practical Applications

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]$$

LM prompt proxy reward

Empirical Finding: An initial **supervised finetuning (SFT)** phase can substantially increase reward variance

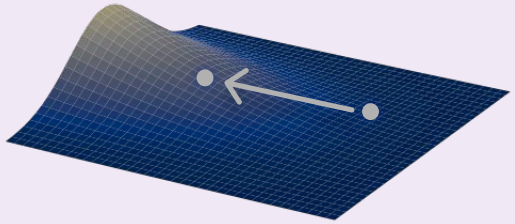


Practical Applications

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]$$

LM prompt proxy reward

Empirical Finding: An initial **supervised finetuning (SFT)** phase can substantially increase reward variance



Explains why SFT is often helpful and when it is necessary

Practical Applications

$$\mathrm{Var}_{\mathbf{y} \sim \pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x}, \mathbf{y})]$$

LM prompt proxy reward

Practical Applications

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]$$

LM prompt proxy reward

Data selection algorithms: choose prompts for reinforcement learning via reward variance

Not All Rollouts are Useful: Down-Sampling Rollouts in LLM Reinforcement Learning

Xu et al. 2025

Reinforcement Learning for Reasoning in Large Language Models with One Training Example

Wang et al. 2025

On the Role of Preference Variance in Preference Optimization

Guo et al. 2025

Improving Generalization in Intent Detection: GRPO with Reward-Based Curriculum Sampling

Feng et al. 2025

Practical Applications

$$\mathrm{Var}_{\mathbf{y} \sim \pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x}, \mathbf{y})]$$

LM prompt proxy reward

Practical Applications

$$\mathrm{Var}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]$$

LM prompt proxy reward

Reward transformations and policy gradient update rules for improving optimization

Accelerating RLHF Training with Reward Variance Increase

Yang et al. 2025

DGRO: Enhancing LLM Reasoning via Exploration-Exploitation Control and Reward Variance Management

Su et al. 2025

RePO: Replay-Enhanced Policy Optimization

Li et al. 2025

ReDit: Reward Dithering for Improved LLM Policy Optimization

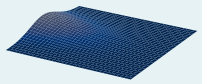
Wei et al. 2025

Takeaway: Importance of Reward Variance

Reward variance is key for successful policy gradient optimization

Takeaway: Importance of Reward Variance

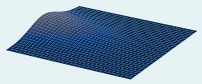
Reward variance is key for successful policy gradient optimization



A reason behind why more accurate proxy rewards are not necessarily better

Takeaway: Importance of Reward Variance

Reward variance is key for successful policy gradient optimization

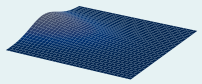


A reason behind why more accurate proxy rewards are not necessarily better

Can help identify optimization issues

Takeaway: Importance of Reward Variance

Reward variance is key for successful policy gradient optimization



A reason behind why more accurate proxy rewards are not necessarily better

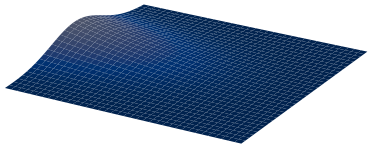
Can help identify optimization issues

Useful for developing new algorithms

Outline

1

Role of Reward Variance



2

Not All Reward Errors Are Harmful

Vanishing Gradients in Reinforcement Finetuning of Language Models

R + Zhou + Saremi + Thilak + Bradley + Nakkiran + Susskind + Littwin | *ICLR 2024*

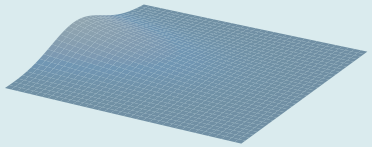
What Makes a Reward Model a Good Teacher? An Optimization Perspective

R + Wang + Strauss + Wei + Lee + Arora | *NeurIPS 2025*

Outline

1

Role of Reward Variance



2

Not All Reward Errors Are Harmful

Vanishing Gradients in Reinforcement Finetuning of Language Models

R + Zhou + Saremi + Thilak + Bradley + Nakkiran + Susskind + Littwin | *ICLR 2024*

What Makes a Reward Model a Good Teacher? An Optimization Perspective

R + Wang + Strauss + Wei + Lee + Arora | *NeurIPS 2025*

When Errors Can Be Beneficial: A Categorization of Imperfect Rewards for Policy Gradient

Shang + Strauss + Wei + Arora + ***R*** | *arXiv 2026*

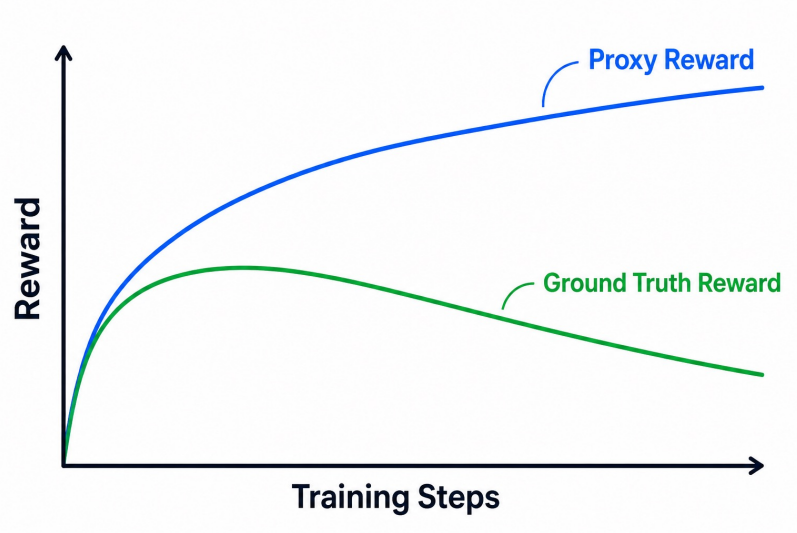
Not All Reward Errors Are Equal

Reward Error: Case where $r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$ on one or more input-output pairs

Not All Reward Errors Are Equal

Reward Error: Case where $r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$ on one or more input-output pairs

Reward errors are conventionally viewed as **harmful** due to the risk of reward hacking

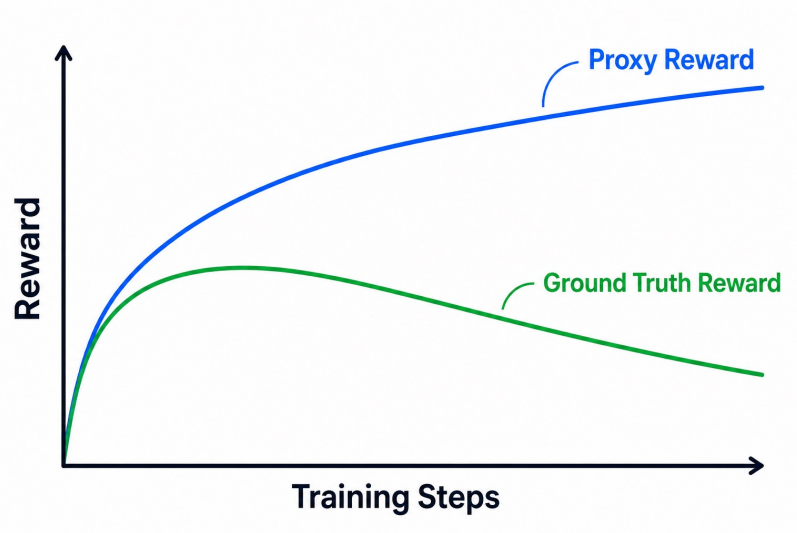


Not All Reward Errors Are Equal

Reward Error: Case where $r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$ on one or more input-output pairs

Reward errors are conventionally viewed as **harmful** due to the risk of reward hacking

caused by assigning high proxy reward to an output with low GT reward



Categorization of Reward Errors: Harmful, Benign, Beneficial

Aside from **harmful**, we prove that reward errors can also be **benign** or even **beneficial**!

$$r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$$

Categorization of Reward Errors: Harmful, Benign, Beneficial

Aside from **harmful**, we prove that reward errors can also be **benign** or even **beneficial**!

$$r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$$

We categorize reward errors according to their effect on the increase in GT reward

Categorization of Reward Errors: Harmful, Benign, Beneficial

Aside from **harmful**, we prove that reward errors can also be **benign** or even **beneficial**!

$$r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$$

We categorize reward errors according to their effect on the increase in GT reward

Technical Approach: Analyze which outputs attract probability during policy gradient

Warm Up: Low Reward Outputs Do Not Attract Probability

Warm Up: Low Reward Outputs Do Not Attract Probability

Proposition: Logit Dynamics

For an input $\mathbf{x}$, output $\mathbf{y}$, and linear softmax policy π_θ:

Warm Up: Low Reward Outputs Do Not Attract Probability

Proposition: Logit Dynamics

For an input $\mathbf{x}$, output $\mathbf{y}$, and linear softmax policy π_θ:

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\text{output probability}}\underbrace{\left(r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]\right)}_{\text{advantage}}$$

Warm Up: Low Reward Outputs Do Not Attract Probability

Proposition: Logit Dynamics

For an input $\mathbf{x}$, output $\mathbf{y}$, and linear softmax policy π_θ:

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\text{output probability}}\underbrace{\left(r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]\right)}_{\textbf{advantage}}$$

Warm Up: Low Reward Outputs Do Not Attract Probability

Proposition: Logit Dynamics

For an input $\mathbf{x}$, output $\mathbf{y}$, and linear softmax policy π_θ:

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\text{output probability}}\Big(\underbrace{r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]}_{\textbf{advantage}}\Big)$$

Implication: An output with negative advantage does not attract probability

Warm Up: Low Reward Outputs Do Not Attract Probability

Proposition: Logit Dynamics

For an input $\mathbf{x}$, output $\mathbf{y}$, and linear softmax policy π_θ:

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\text{output probability}}\Big(\underbrace{r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]}_{\textbf{advantage}}\Big)$$

Implication: An output with negative advantage does not attract probability

Benign Error I

Assigning an incorrect reward that is lower than $\mathbb{E}_{\mathbf{z}\sim\pi_{\theta_0}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]$
to a non-optimal output (i.e., does not achieve maximal GT reward)

Warm Up: Low Probability Outputs Have Negligible Impact

Warm Up: Low Probability Outputs Have Negligible Impact

Proposition: Logit Dynamics

For an input $\mathbf{x}$, output $\mathbf{y}$, and linear softmax policy π_θ:

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\textbf{output probability}}\Big(\underbrace{r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]}_{\text{advantage}}\Big)$$

Warm Up: Low Probability Outputs Have Negligible Impact

Proposition: Logit Dynamics

For an input $\mathbf{x}$, output $\mathbf{y}$, and linear softmax policy π_θ:

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\textbf{output probability}}\Big(\underbrace{r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]}_{\text{advantage}}\Big)$$

Suggests: Optimization is not significantly affected by incorrect rewards assigned to outputs with low initial probability

Warm Up: Low Probability Outputs Have Negligible Impact

Proposition

Warm Up: Low Probability Outputs Have Negligible Impact

Proposition

For any time $T \geq 0$, $\epsilon \geq 0$, and $\mathbf{x}$, suppose that r_{P} and r_{GT} differ only on a set of outputs $\mathcal{Z}$ with low initial probability: $\pi_{\theta_0}(\mathcal{Z}|\mathbf{x}) = \mathcal{O}(\exp(-T)\epsilon)$. Then:

Warm Up: Low Probability Outputs Have Negligible Impact

Proposition

For any time $T \geq 0$, $\epsilon \geq 0$, and $\mathbf{x}$, suppose that r_{P} and r_{GT} differ only on a set of outputs $\mathcal{Z}$ with low initial probability: $\pi_{\theta_0}(\mathcal{Z}|\mathbf{x}) = \mathcal{O}(\exp(-T)\epsilon)$. Then:

$$\mathrm{TV}\left(\pi_{\theta_T}(\cdot|\mathbf{x}), \pi_{\theta_T^{\mathrm{G}}}(\cdot|\mathbf{x})\right) \leq \epsilon$$

policy when maximizing r_{P}

policy when maximizing r_{GT}

Warm Up: Low Probability Outputs Have Negligible Impact

Proposition

For any time $T \geq 0$, $\epsilon \geq 0$, and $\mathbf{x}$, suppose that r_{P} and r_{GT} differ only on a set of outputs $\mathcal{Z}$ with low initial probability: $\pi_{\theta_0}(\mathcal{Z}|\mathbf{x}) = \mathcal{O}(\exp(-T)\epsilon)$. Then:

$$\mathrm{TV}\left(\pi_{\theta_T}(\cdot|\mathbf{x}), \pi_{\theta_T^{\mathrm{G}}}(\cdot|\mathbf{x})\right) \leq \epsilon$$

policy when maximizing r_{P}

policy when maximizing r_{GT}

Benign Error II

Incorrect proxy reward for an output with low probability under π_{θ_0}

Attraction to Mediocre Outputs Can Stall Optimization

We Saw: Outputs with low proxy reward or initial probability do not attract probability

Attraction to Mediocre Outputs Can Stall Optimization

We Saw: Outputs with low proxy reward or initial probability do not attract probability

So where does the probability mass go?

Attraction to Mediocre Outputs Can Stall Optimization

We Saw: Outputs with low proxy reward or initial probability do not attract probability

So where does the probability mass go?

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\text{output probability}}\underbrace{\left(r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]\right)}_{\text{advantage}}$$

Attraction to Mediocre Outputs Can Stall Optimization

We Saw: Outputs with low proxy reward or initial probability do not attract probability

So where does the probability mass go?

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\text{output probability}}\underbrace{\left(r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]\right)}_{\text{advantage}}$$

Implication: Policy gradient can be **drawn to mediocre outputs** if they are more likely than higher reward outputs

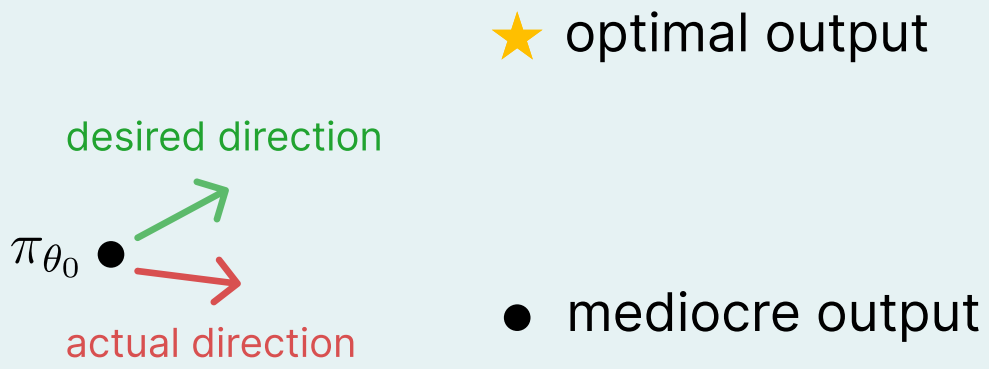
Attraction to Mediocre Outputs Can Stall Optimization

We Saw: Outputs with low proxy reward or initial probability do not attract probability

So where does the probability mass go?

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\text{output probability}}\underbrace{\left(r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]\right)}_{\text{advantage}}$$

Implication: Policy gradient can be **drawn to mediocre outputs** if they are more likely than higher reward outputs



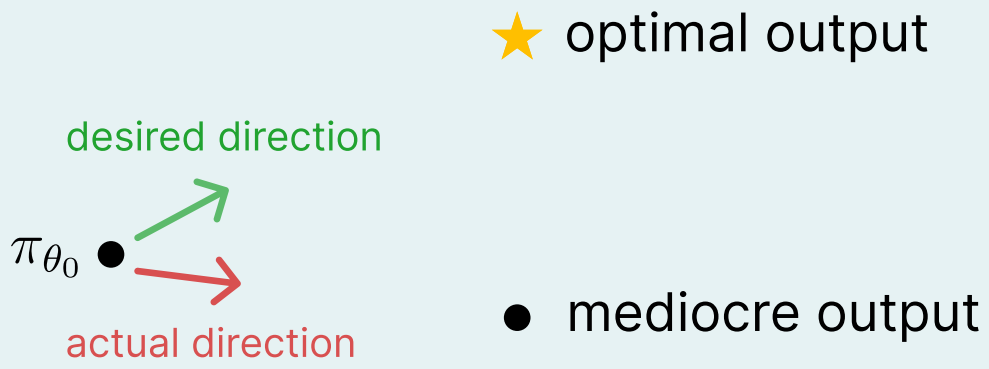
Attraction to Mediocre Outputs Can Stall Optimization

We Saw: Outputs with low proxy reward or initial probability do not attract probability

So where does the probability mass go?

$$\frac{d}{dt}\text{logit}_y(\theta_t) = \underbrace{\pi_{\theta_t}(\mathbf{y}|\mathbf{x})}_{\text{output probability}}\underbrace{\left(r_{\text{P}}(\mathbf{x},\mathbf{y}) - \mathbb{E}_{\mathbf{z}\sim\pi_{\theta_t}(\cdot|\mathbf{x})}\left[r_{\text{P}}(\mathbf{x},\mathbf{z})\right]\right)}_{\text{advantage}}$$

Implication: Policy gradient can be **drawn to mediocre outputs** if they are more likely than higher reward outputs



This phenomenon was originally identified in Mei et al. 2020 (“softmax gravity wells”)

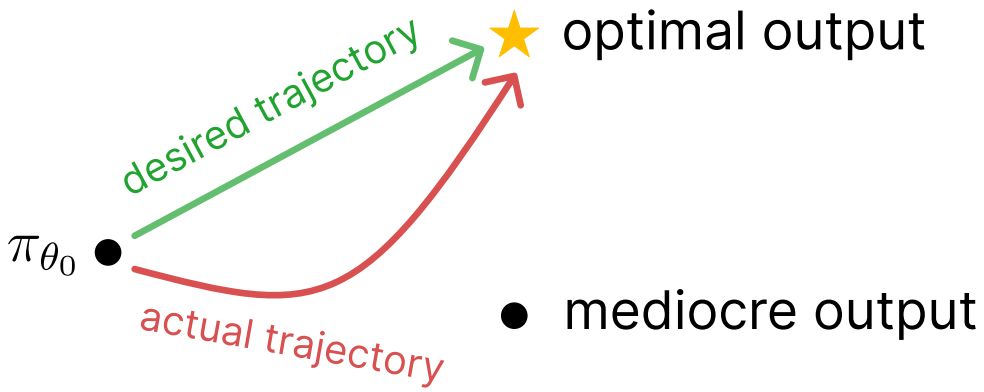
Attraction to Mediocre Outputs Can Stall Optimization

Open Q: How severely can the attraction to mediocre outputs delay optimization?

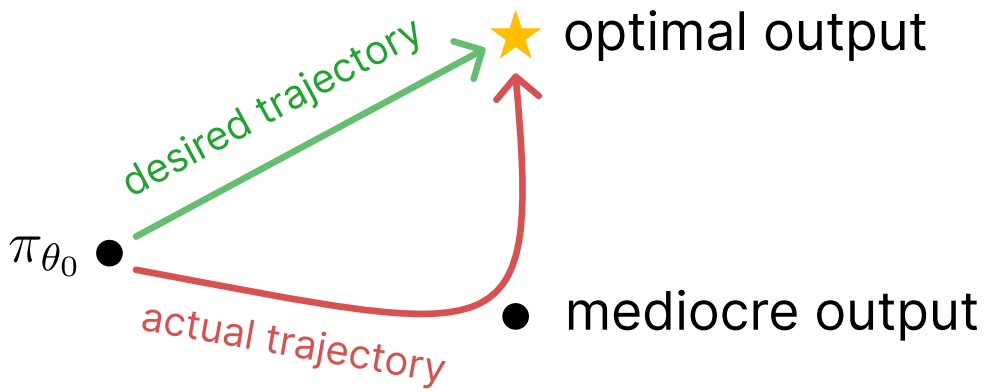
Attraction to Mediocre Outputs Can Stall Optimization

Open Q: How severely can the attraction to mediocre outputs delay optimization?

Mild Delay?

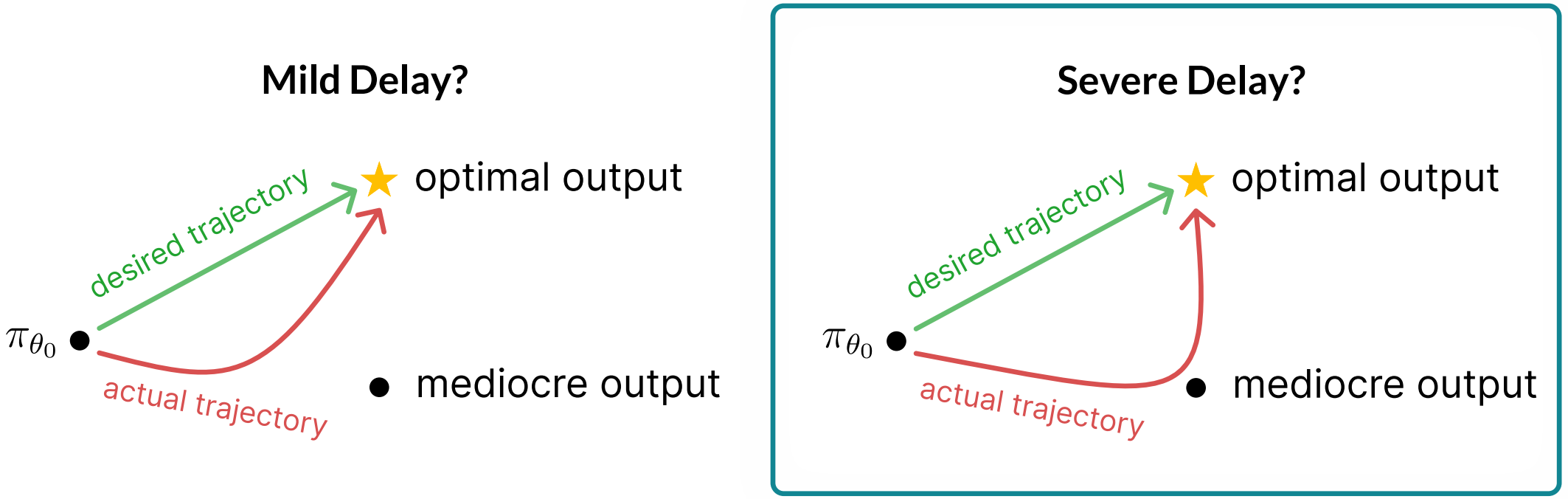


Severe Delay?



Attraction to Mediocre Outputs Can Stall Optimization

Open Q: How severely can the attraction to mediocre outputs delay optimization?



Attraction to Mediocre Outputs Can Stall Optimization

Setting: For an input $\mathbf{x}$

Attraction to Mediocre Outputs Can Stall Optimization

Setting: For an input $\mathbf{x}$

- optimal output $\mathbf{y}_\star = \operatorname{argmax}_{\mathbf{y}} r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$

Attraction to Mediocre Outputs Can Stall Optimization

Setting: For an input $\mathbf{x}$

- optimal output $\mathbf{y}_\star = \mathrm{argmax}_{\mathbf{y}} r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$
- mediocre output $\mathbf{y}_{\mathrm{med}}$ satisfying $V_{\mathrm{GT}}(\theta_0) < r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y}_{\mathrm{med}}) < r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y}_\star)$

Attraction to Mediocre Outputs Can Stall Optimization

Setting: For an input $\mathbf{x}$

- optimal output $\mathbf{y}_\star = \mathrm{argmax}_{\mathbf{y}} r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$
- mediocre output $\mathbf{y}_{\mathrm{med}}$ satisfying $V_{\mathrm{GT}}(\theta_0) < r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y}_{\mathrm{med}}) < r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y}_\star)$
- proxy reward r_{P} identical to r_{GT} except that it assign minimal reward to $\mathbf{y}_{\mathrm{med}}$

Attraction to Mediocre Outputs Can Stall Optimization

Setting: For an input $\mathbf{x}$

- optimal output $\mathbf{y}_\star = \operatorname{argmax}_{\mathbf{y}} r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$
- mediocre output $\mathbf{y}_{\mathrm{med}}$ satisfying $V_{\mathrm{GT}}(\theta_0) < r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y}_{\mathrm{med}}) < r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y}_\star)$
- proxy reward r_{P} identical to r_{GT} except that it assign minimal reward to $\mathbf{y}_{\mathrm{med}}$

Theorem

For $\epsilon \geq 0$, the time it takes for the policy to achieve ϵ-optimal GT reward can be **arbitrarily larger** when training with r_{GT} compared to r_{P}

Attraction to Mediocre Outputs Can Stall Optimization

Setting: For an input $\mathbf{x}$

- optimal output $\mathbf{y}_\star = \text{argmax}_{\mathbf{y}} r_{\text{GT}}(\mathbf{x}, \mathbf{y})$
- mediocre output $\mathbf{y}_{\text{med}}$ satisfying $V_{\text{GT}}(\theta_0) < r_{\text{GT}}(\mathbf{x}, \mathbf{y}_{\text{med}}) < r_{\text{GT}}(\mathbf{x}, \mathbf{y}_\star)$
- proxy reward r_{P} identical to r_{GT} except that it assign minimal reward to $\mathbf{y}_{\text{med}}$

Theorem

For $\epsilon \geq 0$, the time it takes for the policy to achieve ϵ-optimal GT reward can be **arbitrarily larger** when training with r_{GT} compared to r_{P}

The smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is relative to $\pi_{\theta_0}(\mathbf{y}_{\text{med}}|\mathbf{x})$ the larger the gap in time

Attraction to Mediocre Outputs Can Stall Optimization

Theorem

For $\epsilon \geq 0$, the time it takes for the policy to achieve ϵ-optimal GT reward can be **arbitrarily larger** when training with r_{GT} compared to r_{P}

The smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is relative to $\pi_{\theta_0}(\mathbf{y}_{\mathrm{med}}|\mathbf{x})$ the larger the gap in time

Proof Sketch:

Attraction to Mediocre Outputs Can Stall Optimization

Theorem

For $\epsilon \geq 0$, the time it takes for the policy to achieve ϵ-optimal GT reward can be **arbitrarily larger** when training with r_{GT} compared to r_{P}

The smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is relative to $\pi_{\theta_0}(\mathbf{y}_{\mathrm{med}}|\mathbf{x})$ the larger the gap in time

Proof Sketch:

- Under r_{GT} the policy first concentrates probability mass on $\mathbf{y}_{\mathrm{med}}$

Attraction to Mediocre Outputs Can Stall Optimization

Theorem

For $\epsilon \geq 0$, the time it takes for the policy to achieve ϵ-optimal GT reward can be **arbitrarily larger** when training with r_{GT} compared to r_{P}

The smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is relative to $\pi_{\theta_0}(\mathbf{y}_{\mathrm{med}}|\mathbf{x})$ the larger the gap in time

Proof Sketch:

- Under r_{GT} the policy first concentrates probability mass on $\mathbf{y}_{\mathrm{med}}$
  - → Reward variance collapses

Attraction to Mediocre Outputs Can Stall Optimization

Theorem

For $\epsilon \geq 0$, the time it takes for the policy to achieve ϵ-optimal GT reward can be **arbitrarily larger** when training with r_{GT} compared to r_{P}

The smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is relative to $\pi_{\theta_0}(\mathbf{y}_{\mathrm{med}}|\mathbf{x})$ the larger the gap in time

Proof Sketch:

- Under r_{GT} the policy first concentrates probability mass on $\mathbf{y}_{\mathrm{med}}$
  - → Reward variance collapses
  - → The gradient vanishes and the policy stalls around $\mathbf{y}_{\mathrm{med}}$
    (the smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is the more the policy stalls)

Attraction to Mediocre Outputs Can Stall Optimization

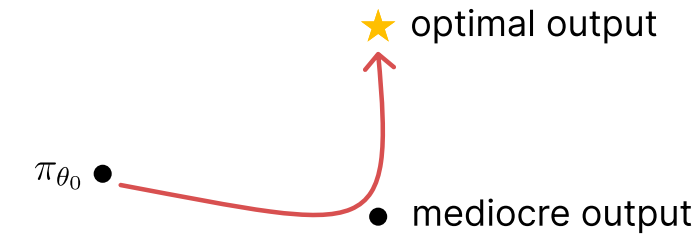
Theorem

For $\epsilon \geq 0$, the time it takes for the policy to achieve ϵ-optimal GT reward can be **arbitrarily larger** when training with r_{GT} compared to r_{P}

The smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is relative to $\pi_{\theta_0}(\mathbf{y}_{\mathrm{med}}|\mathbf{x})$ the larger the gap in time

Proof Sketch:

- Under r_{GT} the policy first concentrates probability mass on $\mathbf{y}_{\mathrm{med}}$
  - → Reward variance collapses
  - → The gradient vanishes and the policy stalls around $\mathbf{y}_{\mathrm{med}}$
    (the smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is the more the policy stalls)



Attraction to Mediocre Outputs Can Stall Optimization

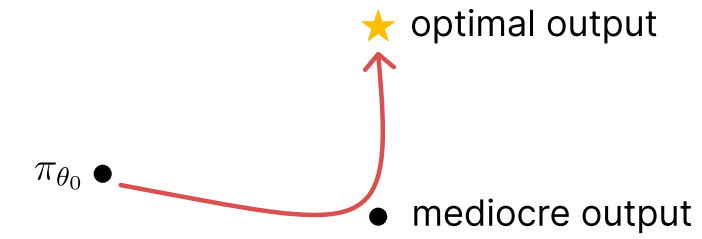
Theorem

For $\epsilon \geq 0$, the time it takes for the policy to achieve ϵ-optimal GT reward can be **arbitrarily larger** when training with r_{GT} compared to r_{P}

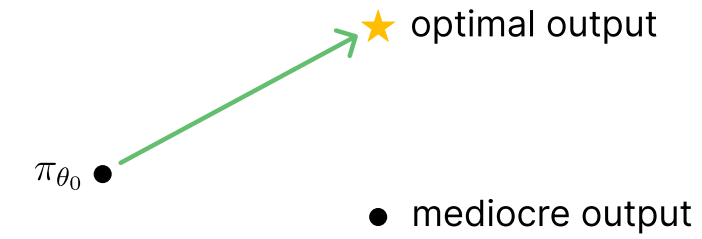
The smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is relative to $\pi_{\theta_0}(\mathbf{y}_{\mathrm{med}}|\mathbf{x})$ the larger the gap in time

Proof Sketch:

- Under r_{GT} the policy first concentrates probability mass on $\mathbf{y}_{\mathrm{med}}$
  - → Reward variance collapses
  - → The gradient vanishes and the policy stalls around $\mathbf{y}_{\mathrm{med}}$
    (the smaller $\pi_{\theta_0}(\mathbf{y}_\star|\mathbf{x})$ is the more the policy stalls)



- Under r_{P} the policy directly shifts probability mass to $\mathbf{y}_\star$



Attraction to Mediocre Outputs Can Stall Optimization

Beneficial Error

Assigning low proxy reward to an output with mediocre GT reward can accelerate the increase in GT reward by steering the policy away from it

Reward Error Categorization Overview

How does $r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$ influence the increase in GT reward?

Harmful Error

Benign Error

Beneficial Error

Reward Error Categorization Overview

How does $r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$ influence the increase in GT reward?

Harmful Error

I. High proxy reward for an output with low GT reward

Reason: reward hacking

Benign Error

Beneficial Error

Reward Error Categorization Overview

How does $r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$ influence the increase in GT reward?

Harmful Error

I. High proxy reward for an output with low GT reward

Reason: reward hacking

Benign Error

I. Incorrect proxy reward is lower than the initial expected proxy reward

Reason: y does not attract probability

II. Incorrect proxy reward for an output with low probability under the initial policy

Reason: negligible effect on outcome

Beneficial Error

Reward Error Categorization Overview

How does $r_{\mathrm{P}}(\mathbf{x}, \mathbf{y}) \neq r_{\mathrm{GT}}(\mathbf{x}, \mathbf{y})$ influence the increase in GT reward?

Harmful Error

I. High proxy reward for an output with low GT reward

Reason: reward hacking

Benign Error

I. Incorrect proxy reward is lower than the initial expected proxy reward

Reason: y does not attract probability

II. Incorrect proxy reward for an output with low probability under the initial policy

Reason: negligible effect on outcome

Beneficial Error

I. Low proxy reward for an output with mediocre GT reward

Reason: prevents policy from concentrating and stalling on y

Application I: Harm-Aware Accuracy

Application I: Harm-Aware Accuracy

Accuracy treats all incorrect output rankings as equally **harmful**

$$\mathrm{Acc}(r_{\mathrm{P}}) := \sum\nolimits_{(\mathbf{x},\mathbf{y}^+,\mathbf{y}^-)\in\mathcal{D}} \frac{1}{|\mathcal{D}|} \mathbb{1}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^+) > r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^-)\right]$$

Application I: Harm-Aware Accuracy

Accuracy treats all incorrect output rankings as equally **harmful**

$$\mathrm{Acc}(r_{\mathrm{P}}) := \sum_{(\mathbf{x},\mathbf{y}^+,\mathbf{y}^-)\in\mathcal{D}} \frac{1}{|\mathcal{D}|} \mathbb{1}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^+) > r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^-)\right]$$

However, we showed that some reward errors can be **benign** or even **beneficial**

Application I: Harm-Aware Accuracy

Accuracy treats all incorrect output rankings as equally **harmful**

$$\mathrm{Acc}(r_{\mathrm{P}}) := \sum_{(\mathbf{x},\mathbf{y}^+,\mathbf{y}^-)\in\mathcal{D}} \frac{1}{|\mathcal{D}|} \mathbb{1}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^+) > r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^-)\right]$$

However, we showed that some reward errors can be **benign** or even **beneficial**

Definition: Harm-Aware Accuracy (HAcc)

$$\mathrm{HAcc}(r_{\mathrm{P}}) := \sum_{(\mathbf{x},\mathbf{y}^+,\mathbf{y}^-)\in\mathcal{D}} \frac{\pi_{\theta_0}(\mathbf{y}^+|\mathbf{x})\pi_{\theta_0}(\mathbf{y}^-|\mathbf{x})}{\underbrace{Z}_{\text{normalization constant}}} \mathbb{1}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^+) > r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^-)\right]$$

Application I: Harm-Aware Accuracy

Accuracy treats all incorrect output rankings as equally **harmful**

$$\mathrm{Acc}(r_{\mathrm{P}}) := \sum_{(\mathbf{x},\mathbf{y}^+,\mathbf{y}^-)\in\mathcal{D}} \frac{1}{|\mathcal{D}|} \mathbb{1}\left[r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^+) > r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^-)\right]$$

However, we showed that some reward errors can be **benign** or even **beneficial**

Definition: Harm-Aware Accuracy (HAcc)

$$\mathrm{HAcc}(r_{\mathrm{P}}) := \sum_{(\mathbf{x},\mathbf{y}^+,\mathbf{y}^-)\in\mathcal{D}} \frac{\pi_{\theta_0}(\mathbf{y}^+|\mathbf{x})\pi_{\theta_0}(\mathbf{y}^-|\mathbf{x})}{\underbrace{Z}_{\text{normalization constant}}} \mathbb{1}\left[\max\left\{r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^+), \underbrace{\bar{V}_{\mathrm{P}}(\theta_0;\mathbf{x})}_{\text{empirical estimate of } \mathbb{E}_{\mathbf{y}\sim\pi_{\theta_0}(\cdot|\mathbf{x})}[r_{\mathrm{P}}(\mathbf{x},\mathbf{y})]}\right\} > r_{\mathrm{P}}(\mathbf{x},\mathbf{y}^-)\right]$$

Application I: Harm-Aware Accuracy

Experiments: Evaluating reward models using Acc vs HAcc

Application I: Harm-Aware Accuracy

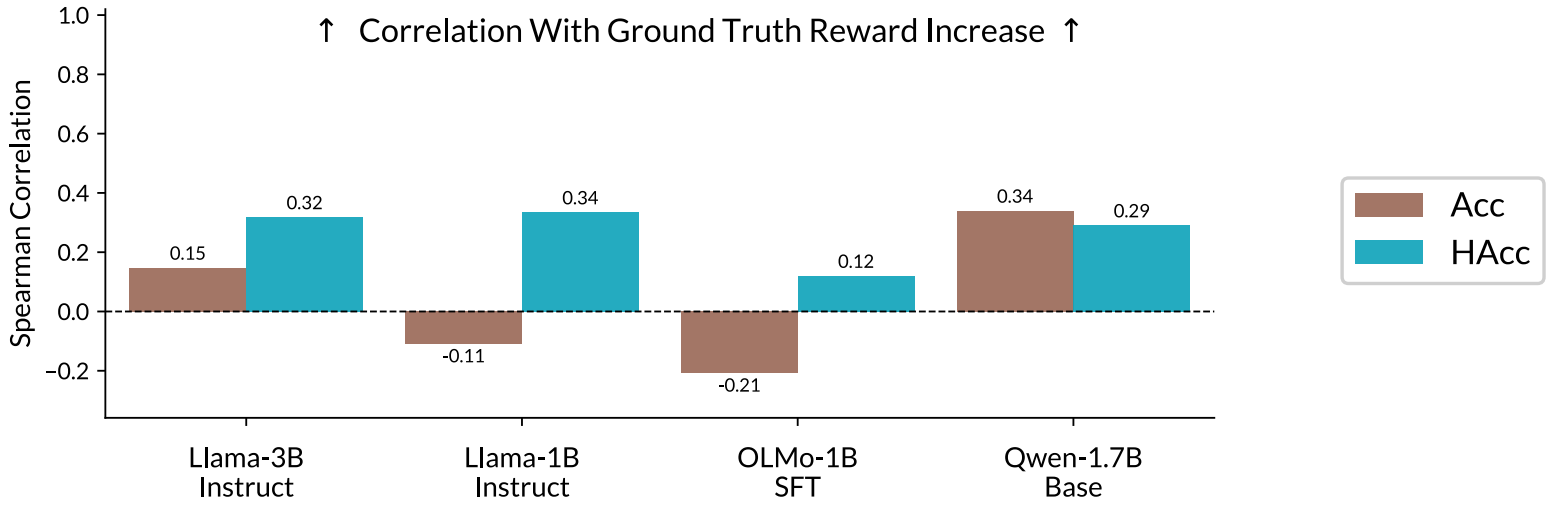
Experiments: Evaluating reward models using Acc vs HAcc

Setting: Train LMs via policy gradient with 13 different reward models on UltraFeedback

Application I: Harm-Aware Accuracy

Experiments: Evaluating reward models using Acc vs HAcc

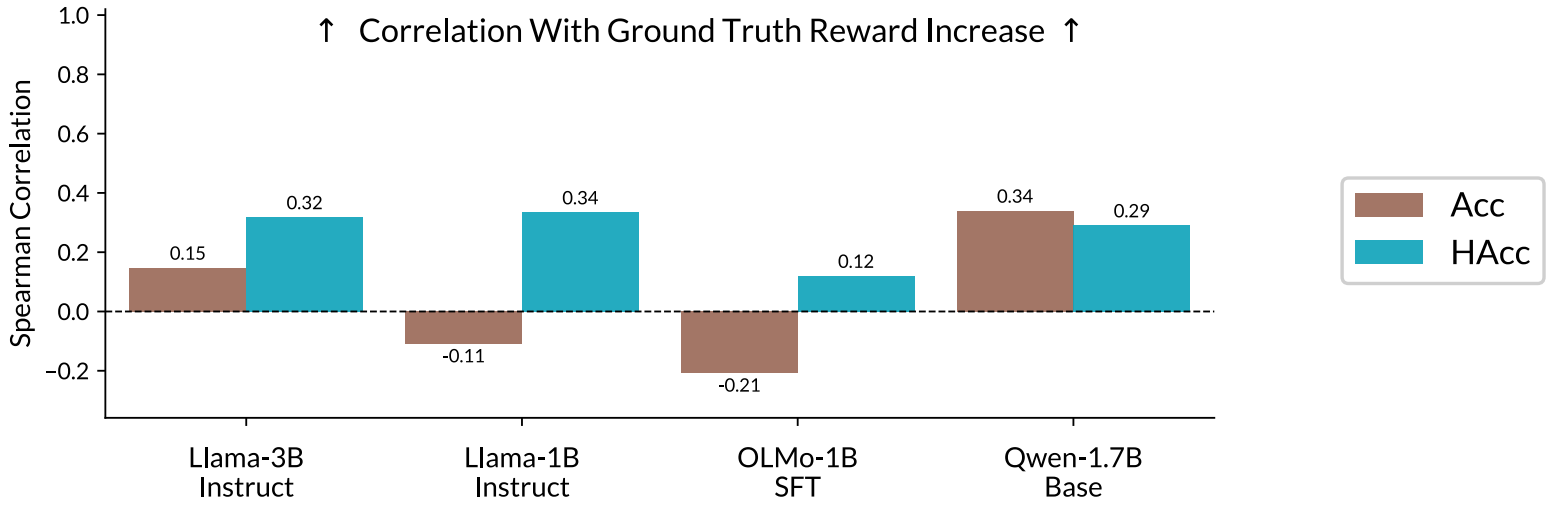
Setting: Train LMs via policy gradient with 13 different reward models on UltraFeedback



Application I: Harm-Aware Accuracy

Experiments: Evaluating reward models using Acc vs HAcc

Setting: Train LMs via policy gradient with 13 different reward models on UltraFeedback

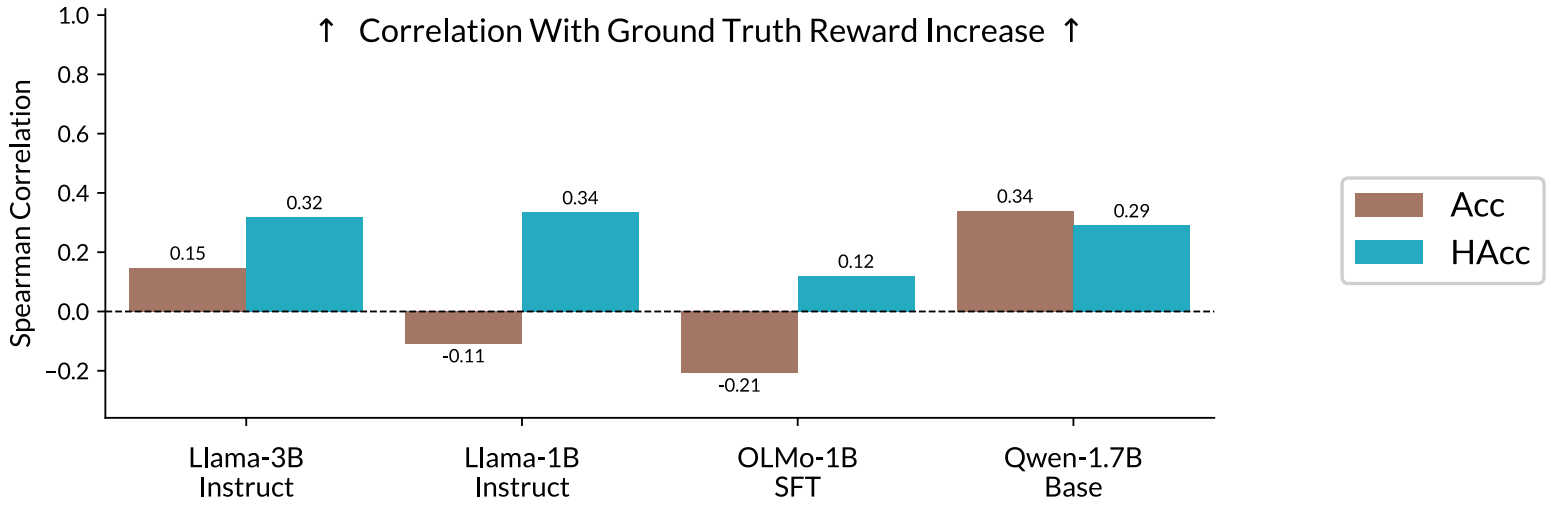


HAcc is more predictive of which reward model leads to better LM performance

Application I: Harm-Aware Accuracy

Experiments: Evaluating reward models using Acc vs HAcc

Setting: Train LMs via policy gradient with 13 different reward models on UltraFeedback



HAcc is more predictive of which reward model leads to better LM performance

Yet, robust reward model evaluation remains an open challenge

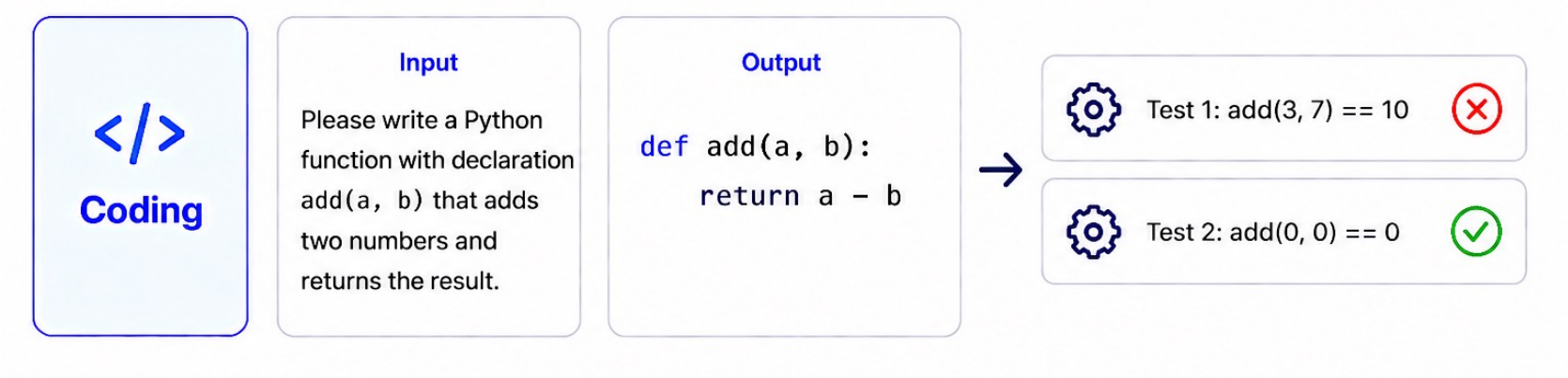
Application II: Should Partially Correct Outputs Be Rewarded?

Application II: Should Partially Correct Outputs Be Rewarded?

In domains such as coding and instruction following,
a **correct output must satisfy multiple constraints**

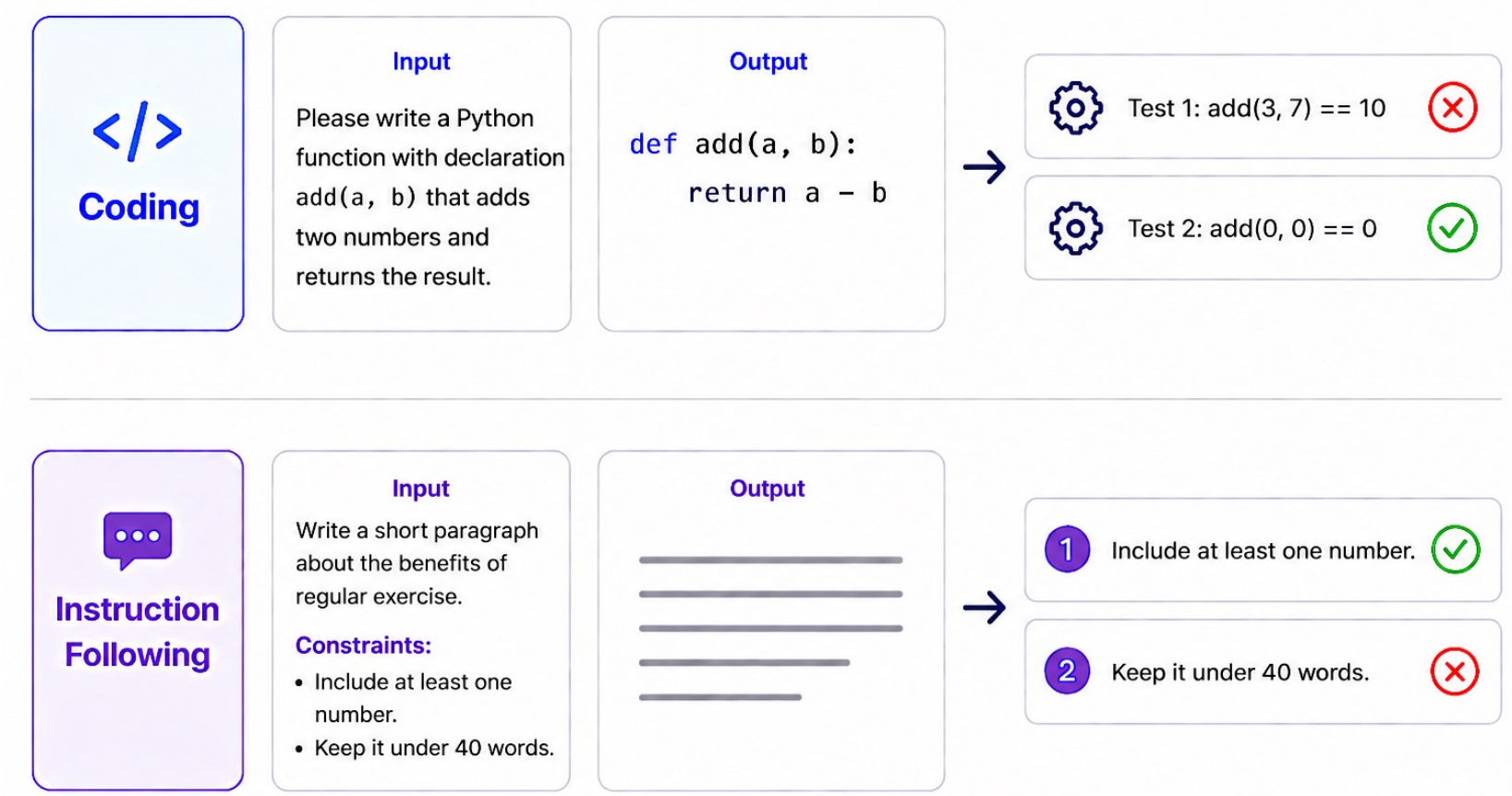
Application II: Should Partially Correct Outputs Be Rewarded?

In domains such as coding and instruction following,
a **correct output must satisfy multiple constraints**



Application II: Should Partially Correct Outputs Be Rewarded?

In domains such as coding and instruction following, a **correct output must satisfy multiple constraints**



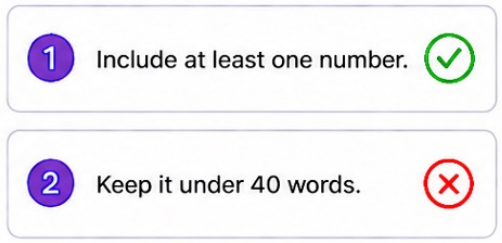
Application II: Should Partially Correct Outputs Be Rewarded?

Q: Should partially correct outputs be rewarded?

Application II: Should Partially Correct Outputs Be Rewarded?

Q: Should partially correct outputs be rewarded?

Partial Rewards:
1 to fully correct outputs, a value in (0,1) to partially correct outputs, and 0 to all others

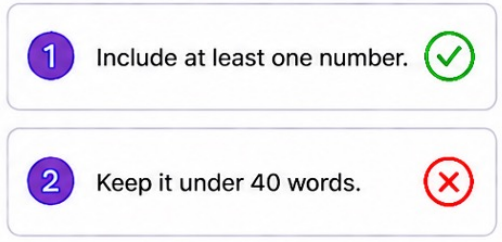


Reward = 0.5

Application II: Should Partially Correct Outputs Be Rewarded?

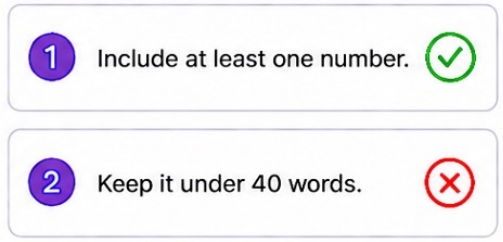
Q: Should partially correct outputs be rewarded?

Partial Rewards:
1 to fully correct outputs, a value in (0,1) to partially correct outputs, and 0 to all others



Reward = 0.5

Binary (Full-Correctness) Rewards:
1 to fully correct outputs and 0 to all others

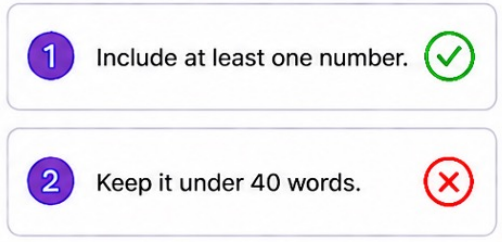


Reward = 0

Application II: Should Partially Correct Outputs Be Rewarded?

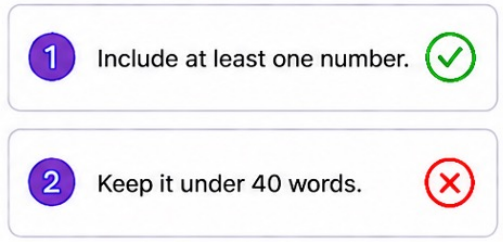
Q: Should partially correct outputs be rewarded?

Partial Rewards:
1 to fully correct outputs, a value in (0,1) to partially correct outputs, and 0 to all others



Reward = 0.5

Binary (Full-Correctness) Rewards:
1 to fully correct outputs and 0 to all others



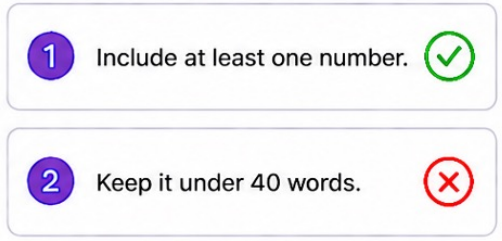
Reward = 0

Intuition: Partial rewards are helpful since they provide more information, but…

Application II: Should Partially Correct Outputs Be Rewarded?

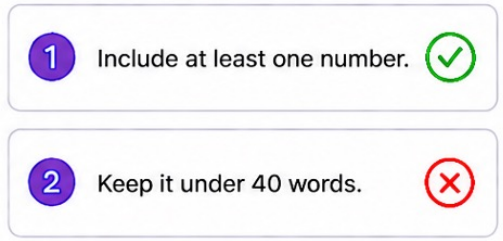
Q: Should partially correct outputs be rewarded?

Partial Rewards:
1 to fully correct outputs, a value in (0,1) to partially correct outputs, and 0 to all others



Reward = 0.5

Binary (Full-Correctness) Rewards:
1 to fully correct outputs and 0 to all others



Reward = 0

Intuition: Partial rewards are helpful since they provide more information, but…

Our Theory: If the LM is more likely to produce partially correct outputs than fully correct ones → rewarding partial correctness can cause the LM to **stall on partially correct outputs**

Application II: Should Partially Correct Outputs Be Rewarded?

Experiments: Train an LM to produce outputs satisfying a pair of constraints specified in the prompt

Application II: Should Partially Correct Outputs Be Rewarded?

Experiments: Train an LM to produce outputs satisfying a pair of constraints specified in the prompt

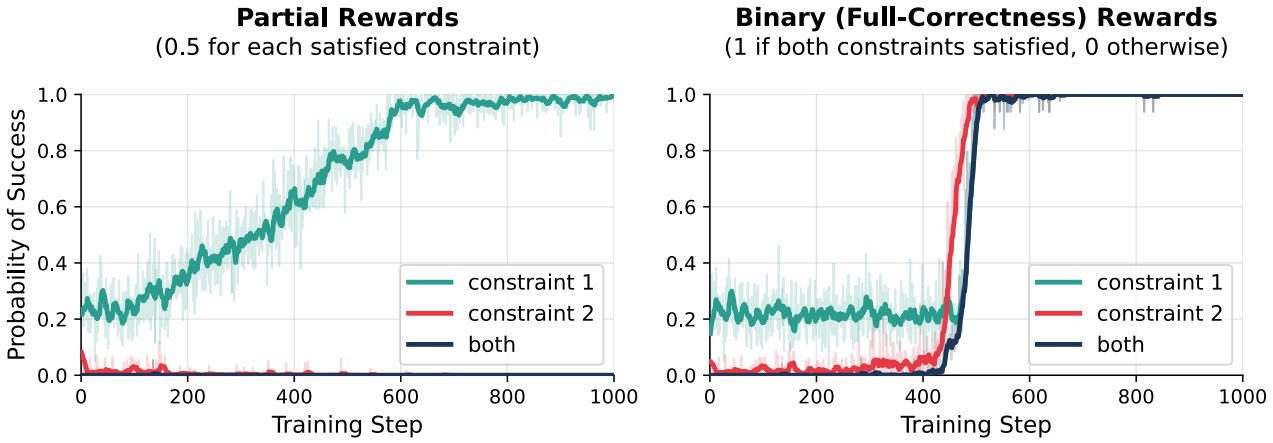
Setting: Qwen3-1.7B trained via policy gradient with constraints from IFBench (Pyatkin et al. 2025)

Application II: Should Partially Correct Outputs Be Rewarded?

Experiments: Train an LM to produce outputs satisfying a pair of constraints specified in the prompt

Setting: Qwen3-1.7B trained via policy gradient with constraints from IFBench (Pyatkin et al. 2025)

Constraint Pair A:
Gap in Learnability



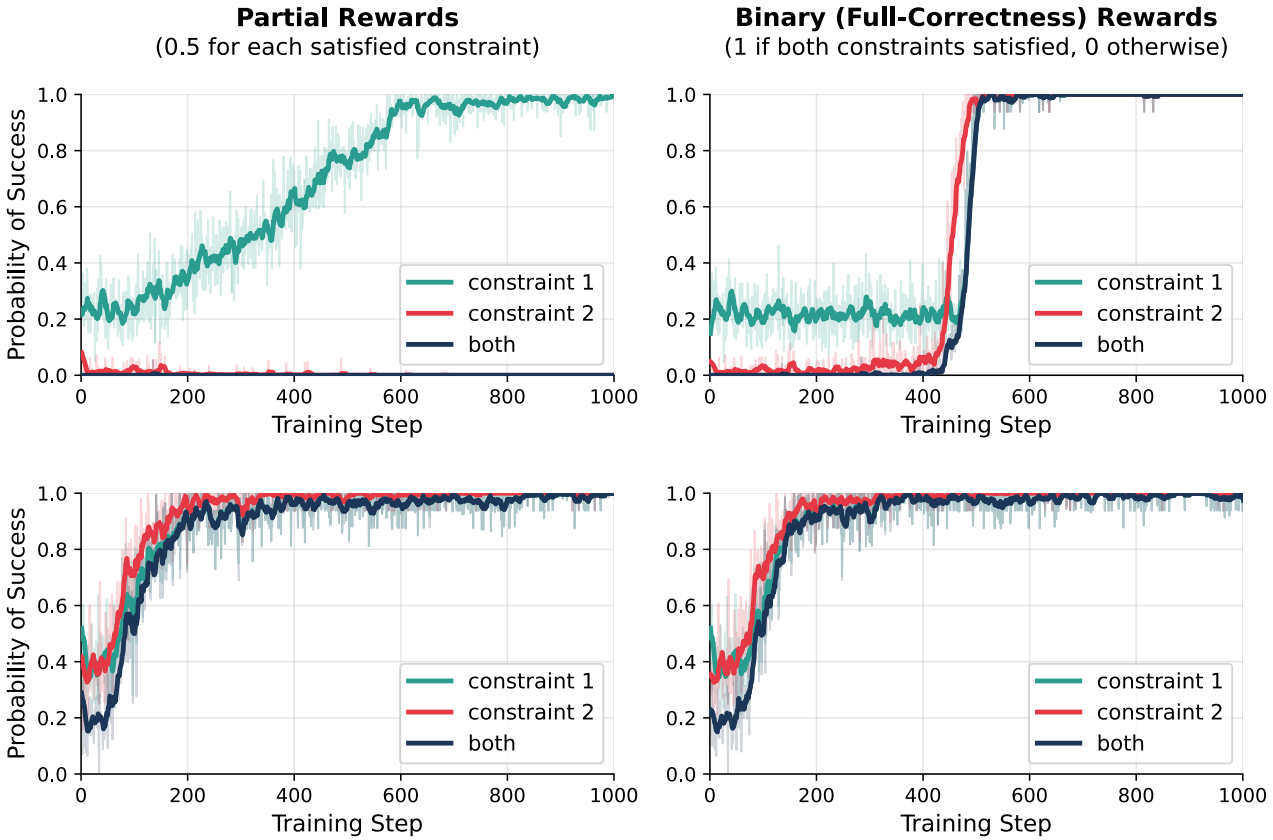
Application II: Should Partially Correct Outputs Be Rewarded?

Experiments: Train an LM to produce outputs satisfying a pair of constraints specified in the prompt

Setting: Qwen3-1.7B trained via policy gradient with constraints from IFBench (Pyatkin et al. 2025)

Constraint Pair A:
Gap in Learnability

Constraint Pair B:
No Gap in Learnability



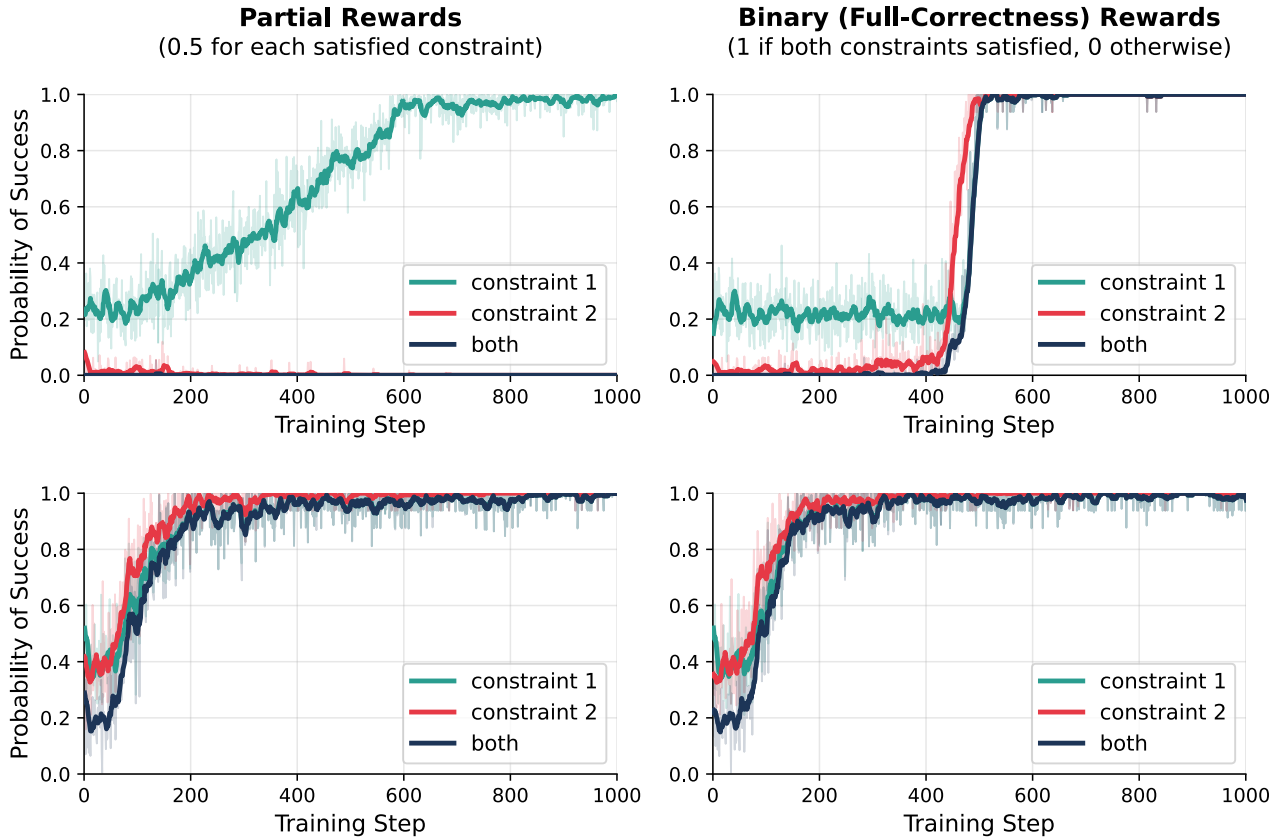
Application II: Should Partially Correct Outputs Be Rewarded?

Experiments: Train an LM to produce outputs satisfying a pair of constraints specified in the prompt

Setting: Qwen3-1.7B trained via policy gradient with constraints from IFBench (Pyatkin et al. 2025)

Constraint Pair A:
Gap in Learnability

Constraint Pair B:
No Gap in Learnability



Rewarding partially correct outputs is not always helpful!

Conclusion

Recap

Recap

Q: What makes a good proxy reward function for policy gradient?

Recap

Q: What makes a good proxy reward function for policy gradient?

Role of Reward Variance

Not All Reward Errors Are Harmful

Recap

Q: What makes a good proxy reward function for policy gradient?

Role of Reward Variance

Proxy rewards need to induce sufficient **reward variance** for efficient optimization

Not All Reward Errors Are Harmful

Recap

Q: What makes a good proxy reward function for policy gradient?

Role of Reward Variance

Proxy rewards need to induce sufficient **reward variance** for efficient optimization

Implication: **More accurate proxy reward functions are not always better**

Not All Reward Errors Are Harmful

Recap

Q: What makes a good proxy reward function for policy gradient?

Role of Reward Variance

Proxy rewards need to induce sufficient **reward variance** for efficient optimization

Implication: **More accurate proxy reward functions are not always better**

Practical Applications: Data selection and policy gradient methods

Not All Reward Errors Are Harmful

Recap

Q: What makes a good proxy reward function for policy gradient?

Role of Reward Variance

Proxy rewards need to induce sufficient **reward variance** for efficient optimization

Implication: **More accurate proxy reward functions are not always better**

Practical Applications: Data selection and policy gradient methods

Not All Reward Errors Are Harmful

Reward errors are conventionally viewed as harmful, but can also be **benign or beneficial**

Recap

Q: What makes a good proxy reward function for policy gradient?

Role of Reward Variance

Proxy rewards need to induce sufficient **reward variance** for efficient optimization

Implication: **More accurate proxy reward functions are not always better**

Practical Applications: Data selection and policy gradient methods

Not All Reward Errors Are Harmful

Reward errors are conventionally viewed as harmful, but can also be **benign or beneficial**

Mechanism behind beneficial errors: **policy gradient can get stuck on mediocre outputs**

Recap

Q: What makes a good proxy reward function for policy gradient?

Role of Reward Variance

Proxy rewards need to induce sufficient **reward variance** for efficient optimization

Implication: **More accurate proxy reward functions are not always better**

Practical Applications: Data selection and policy gradient methods

Not All Reward Errors Are Harmful

Reward errors are conventionally viewed as harmful, but can also be **benign or beneficial**

Mechanism behind beneficial errors: **policy gradient can get stuck on mediocre outputs**

Practical Applications: Harm-aware accuracy and insights for proxy reward design

Outlook

Our results shed light on what makes a good proxy reward function

But there is still much more to understand…

Outlook

Our results shed light on what makes a good proxy reward function

But there is still much more to understand…

Takeaway

The effectiveness of proxy rewards cannot be assessed solely by how much they deviate from the GT

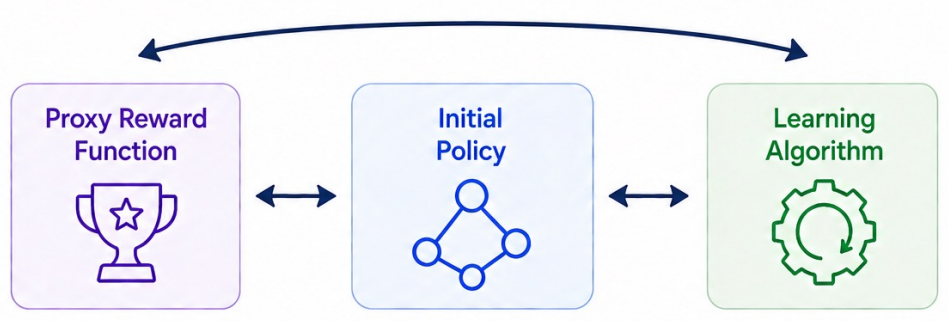
Outlook

Our results shed light on what makes a good proxy reward function

But there is still much more to understand…

Takeaway

The effectiveness of proxy rewards cannot be assessed solely by how much they deviate from the GT



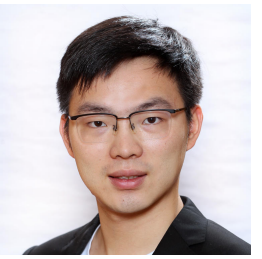
It is essential to account for the interplay between the proxy reward function, initial policy, and learning algorithm

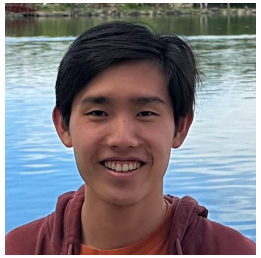








Thank You!

Work supported in part by the
Zuckerman STEM Leadership Program

Recap

Q: What makes a good proxy reward function for policy gradient?

Role of Reward Variance

Proxy rewards need to induce sufficient **reward variance** for efficient optimization

Implication: **More accurate proxy reward functions are not always better**

Practical Applications: Data selection and policy gradient methods

Not All Reward Errors Are Harmful

Reward errors are conventionally viewed as harmful, but can also be **benign or beneficial**

Mechanism behind beneficial errors: **policy gradient can get stuck on mediocre outputs**

Practical Applications: Harm-aware accuracy and insights for proxy reward design