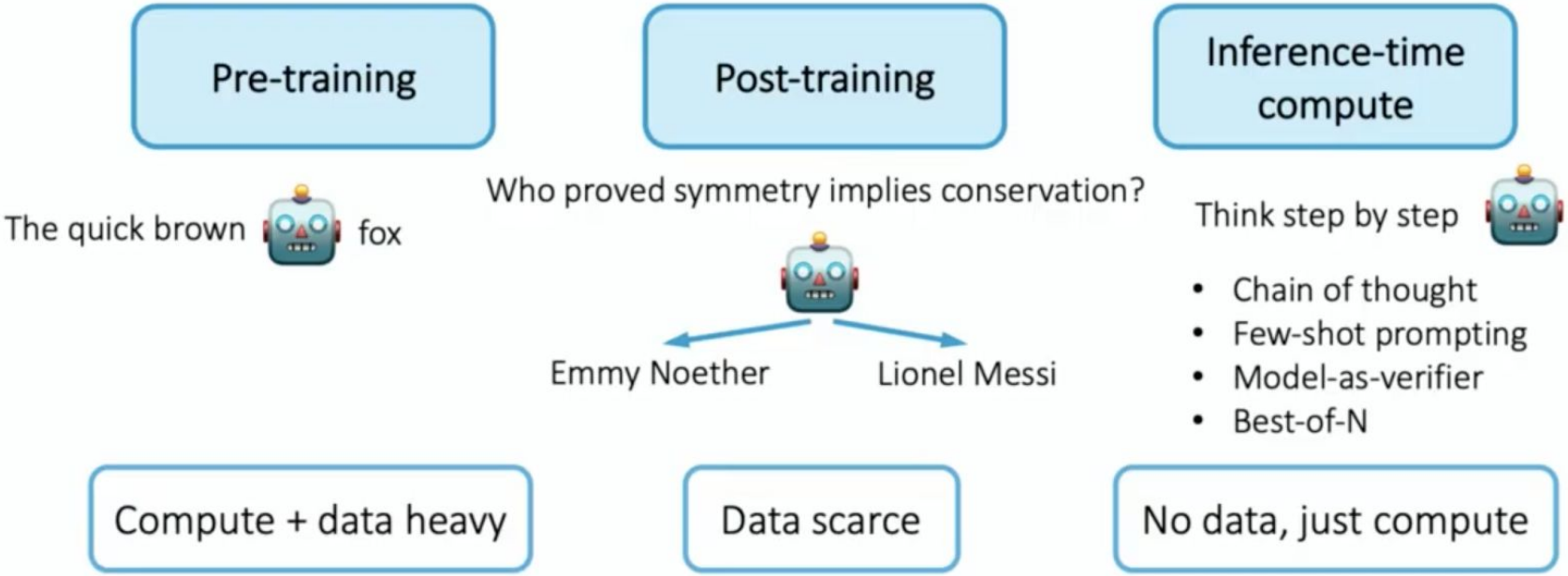
Self-Improvement in Language Models: The Sharpening Mechanism

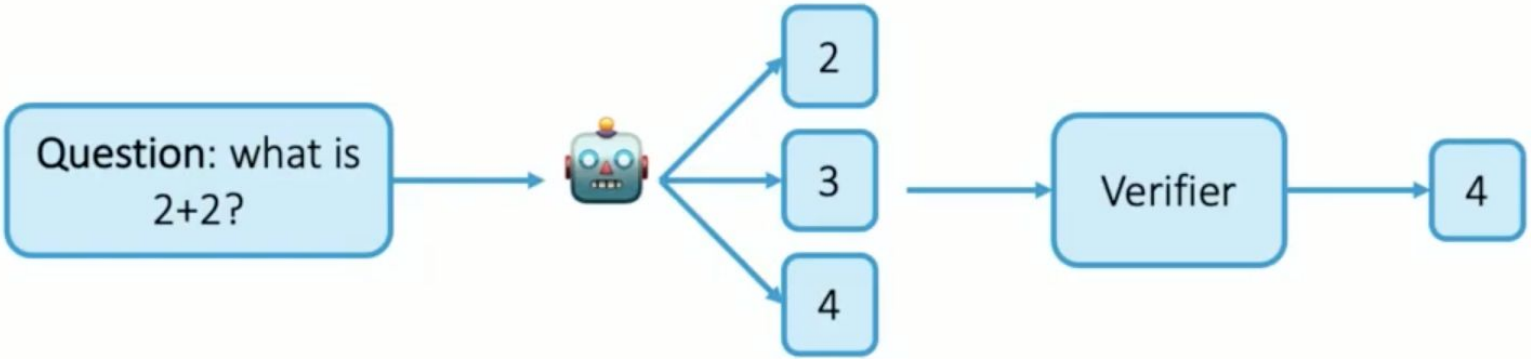
Huang, Block, Foster, Rohatgi, Zhang, Simchowitz, Ash, Krishnamurthy

Some slides & pictures adapted from Akshay Krishnamurthy

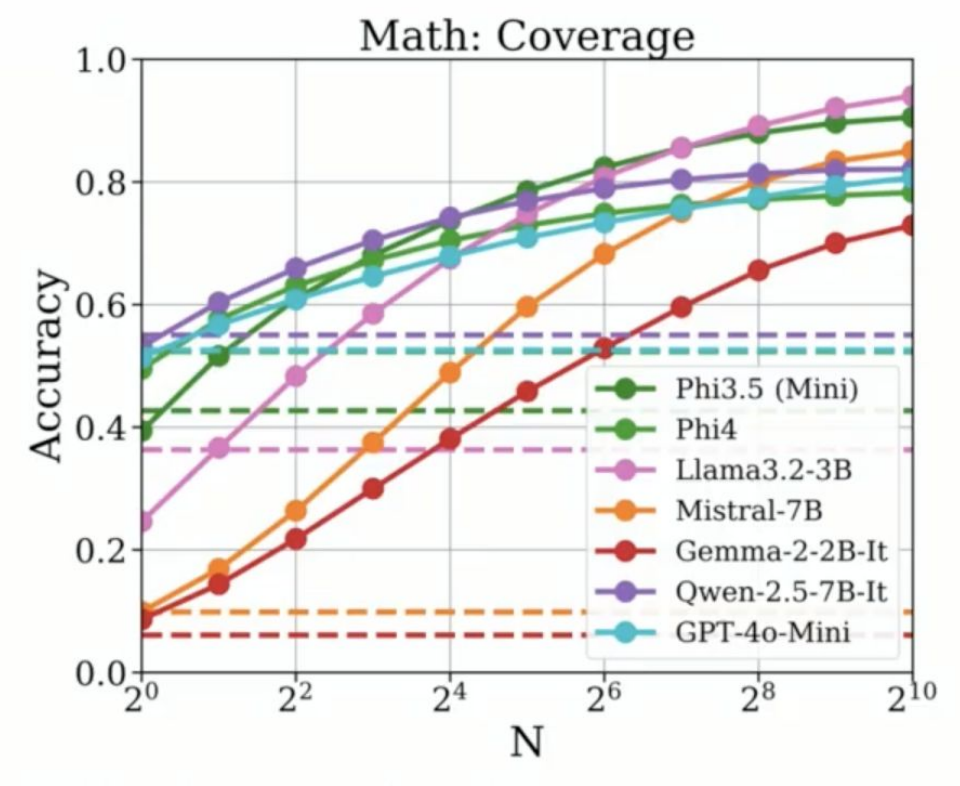
LLM Training



Inference Time Compute - BoN



Inference Time Compute - BoN



~32 samples for good performance

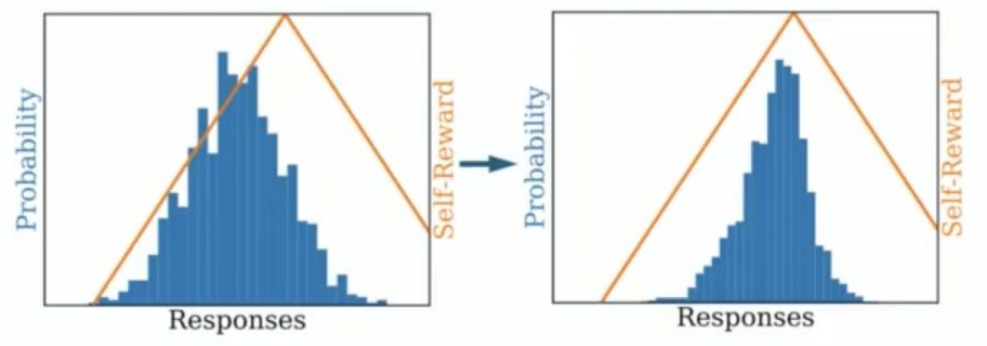
Self Improvement

- Model is improving performance using its own outputs
- Post-processing doesn’t provide new information
- Using compute to uncover answer it already knows

Sharpening

Hypothesis: using LLM-as-verifier improves downstream performance

Mechanism: LLM-as-verifier tilts the distribution to output better responses



Self-reward and sharpening

A language model is a distribution π_base(y | x) over responses given a prompt

Self reward r_self(y | x, πbase) is a function of response, prompt, and LM, examples:

- **Sequence level log-probabilities: log π_base(y | x)**
- Length-normalized log-probabilities: log π_base(y | x) / |y|
- Majority voting
- Prompting LM to score its own response (LM-as-judge)

Self-reward and sharpening

A language model is a distribution π_base(y | x) over responses given a prompt

Self reward r_self(y | x, πbase) is a function of response, prompt, and LM, examples:

- **Sequence level log-probabilities: log π_base(y | x)**
- Length-normalized log-probabilities: log π_base(y | x) / |y|
- Majority voting
- Prompting LM to score its own response (LM-as-judge)

Sharpening: π̂(x) ≈ argmax_y r_self(y; x, π_base)

Note: Theory is developed for log π_base(y | x) only as this admits a clean closed-form optimal policy π*(y|x) ∝ π_base (y|x) ^ (1+1/β). Length-normalized and other rewards are tested empirically but lack theoretical guarantees.

(ε, δ)-Sharpened Model

A model $\hat{\pi}$ is (ε, δ)-sharpened if

$$\mathbb{P}_{x \sim \mu}\left[\, \hat{\pi}(y^*(x) \mid x) \;\geq\; 1 - \delta \,\right] \;\geq\; 1 - \varepsilon$$

where $y^*(x) = \text{argmax}_y \log \pi_{\text{base}}(y \mid x)$

Key: if $\delta < \frac{1}{2}$, then $\hat{\pi}(y^*(x) \mid x) > \frac{1}{2} \;\Rightarrow$ greedy decoding recovers y^*

This is NP hard! Convert computational problem to statistical problem

Inference Time Sharpening with BoN

- Generate N responses
- Return the response with the highest rewards

Training Time Sharpening with BoN

- Generate N responses
- Return the response with the highest rewards
- **Amortize cost by SFT on the best of the N responses**

Sample & Evaluate Framework

- Sample n prompts from the base model
- For each prompt sample N responses
- Query the reward model
- Sample complexity = n.N

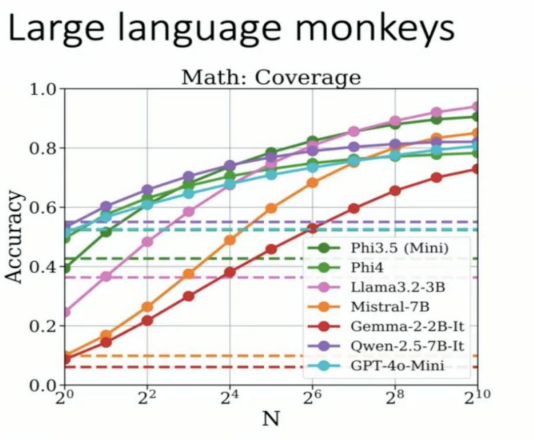
Coverage

Coverage Coefficient

$$C_{\text{cov}} = \mathbb{E}_{x \sim \mu}\left[\left(\pi_{\text{base}}(y^*(x) \mid x)\right)^{-1} \right]$$

Small C_{cov}: π_{base} often samples y^* ⇒ easy to sharpen

Large C_{cov}: π_{base} rarely samples y^* ⇒ exponentially hard



In practice CoV << Y = |V|^length

SFT Sharpening Guarantees

Upper Bound (Theorem 3.2)

SFT-Sharpening achieves (ε, δ)-sharpening with sample complexity

$$m = O\left(\frac{C_{\text{cov}} \cdot \log|\Pi| \cdot \log(\delta^{-1})}{\delta\, \varepsilon^2}\right)$$

Lower Bound (Theorem 3.3)

Any algorithm requires

$$m \gtrsim \frac{C_{\text{cov}}}{\varepsilon^2} \text{ samples}$$

Greedy Sufficiency (Prop. 3.1)

If $\delta < \frac{1}{2}$, then $\hat{\pi}(y^*(x) \mid x) > \frac{1}{2}$

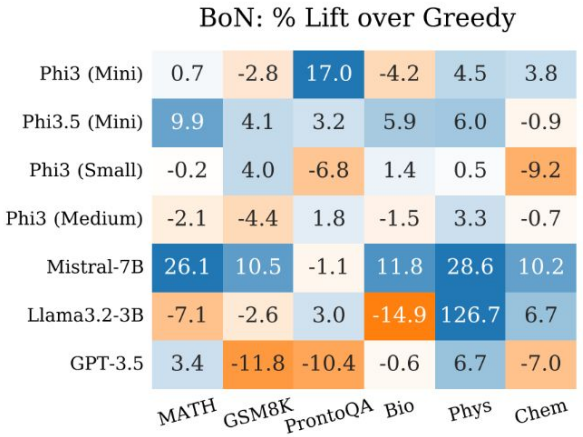
$\Rightarrow$ greedy decoding recovers y^* exactly

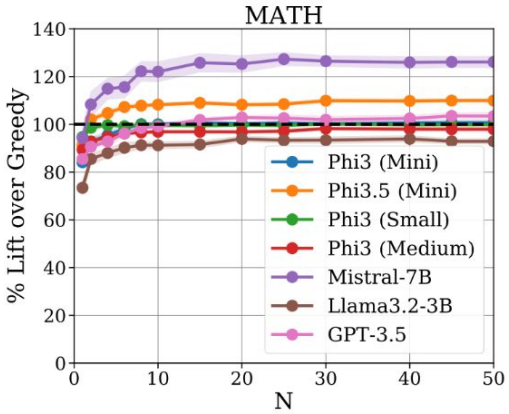
SFT-Sharpening is minimax optimal

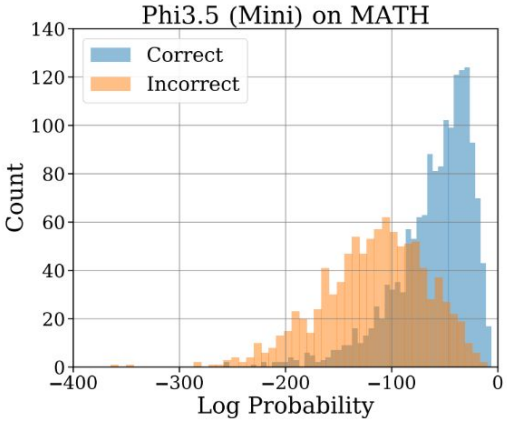
Matches the lower bound up to log factors — but cannot escape dependence on C_{cov}

$C_{\text{cov}} = \mathbb{E}_{x \sim \mu}[(\pi_{\text{base}}(y^*(x) \mid x))^{-1}]$ — *small when base model easily samples the best response*

Results on Maximum Likelihood Sharpening







Power of Exploration

Optimization Objective at Iteration t

$$\pi^{(t+1)} \leftarrow \arg\min_{\pi} \; -\sum_{D^{(t)}} \left(\beta \log \frac{\pi(y|x)}{\pi_{\text{base}}(y|x)} - \beta \log \frac{\pi(y'|x)}{\pi_{\text{base}}(y'|x)} - (r(x,y) - r(x,y')) \right)^2 + \alpha \sum_{D^{(t)}} \log \pi(y'|x)$$

DPO squared loss — *optimism bonus*

Dataset Construction

Chosen: $y \sim \pi^{(t)}(\cdot|x)$ (current policy)

Rejected: $y' \sim \pi_{\text{base}}(\cdot|x)$ (base model)

Reward: $r(x,y) = \log \pi_{\text{base}}(y|x)$

What Each Term Does

DPO loss: push π toward high-reward sequences

Optimism: suppress π_{base} samples $\Rightarrow$ explore new regions

Together: find high-reward sequences π_{base} never generates

Power of Exploration

XPO Sample Complexity (Theorem 4.3)

XPO achieves (ε, δ)-sharpening with sample complexity

$$m = \widetilde{O}\left(\frac{\mathrm{SEC}(\Pi) \cdot \log|\Pi|}{\gamma^2_{\mathrm{margin}}\, \delta^2\, \varepsilon^2}\right) \quad — \quad \text{no dependence on } C_{\mathrm{cov}}$$

SFT-Sharpening vs XPO

SFT-Sharpening

Complexity: $\dfrac{C_{\mathrm{cov}} \cdot \log|\Pi|}{\delta\, \varepsilon^2}$

Linear softmax: $C_{\mathrm{cov}} = \exp(\Omega(d))$

Offline — stuck with coverage of base model

Cannot escape coverage barrier

XPO

Complexity: $\dfrac{\mathrm{SEC}(\Pi) \cdot \log|\Pi|}{\gamma^2_{\mathrm{margin}}\, \delta^2\, \varepsilon^2}$

Linear softmax: $\mathrm{SEC}(\Pi) = \widetilde{O}(d)$

Online — actively explores beyond base model

Exponential improvement over SFT

For linear softmax models: $\mathrm{SEC}(\Pi) = \widetilde{O}(d)$ **vs** $C_{\mathrm{cov}} = \exp(\Omega(d))$ — **an exponential separation**

SEC measures model class complexity from an exploration perspective, not base model quality