Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan
Timothy Lillicrap, Kenji Kawaguchi, Michael Shieh

Follow me on X 👉 @sigrid_xie

Paper 👉

NUS National University of Singapore

Google DeepMind

Motivation and Contribution

Background

- **Iterative Improvement**: An iterative approach proposes a dynamic and continuous refinement process. It involves a cycle that begins with the current policy, progresses through the collection and analysis of data to generate new (preference) data, and uses this data to update the policy.
- **AlphaZero**: It combines the strengths of neural networks, RL techniques, and Monte Carlo Tree Search (MCTS). The integration of MCTS as a policy improvement operator that transforms the current policy into an improved policy.

Challenges in Applying MCTS to LLM Reasoning

- **Granularity**: The instance-level approach employs sparse supervision, which can lose important information and may not optimally leverage the potential of MCTS in improving the LLMs.
- **Critic/Reward Function**: A reliable function is crucial for providing meaningful feedback on different rollouts generated by MCTS, thus guiding the policy improvement process.

🤔 Research Question

With step-level MCTS providing a more granular supervision, plus self-evaluation to enhance step-level assessment, can we collect fine-grained preference data of better quality to enhance model reasoning?

Part One: MCTS for Step-Level Data Collection

Select. To navigate the trade-off between exploring new nodes and exploiting visited ones, we employ the Predictor + Upper Confidence bounds applied to Trees (PUCT).

$$s_{t+1}{}^{*} = \arg\max_{s_t}\left[Q(s_t,a) + c_{\text{puct}} \cdot p(a \mid s_t)\frac{\sqrt{N(s_t)}}{1+N(s_{t+1})}\right]$$

Expand. Expansion occurs at a leaf node during the selection process to integrate new nodes and assess rewards. Reward computation merges outcome correctness with self-evaluation.

$$R(s_t) = \mathcal{O}(s_t) + \mathcal{C}(s_t)$$

$$\mathcal{C}(s_t) = \pi_\theta(\mathrm{A} \mid \text{prompt}_{\text{eval}}, x, s_t)$$

Backup. Once a terminal state is reached, we carry out a bottom-up update from the terminal node back to the root.

$$Q(s_t,a) \leftarrow r(s_t,a) + \gamma V(s_{t+1})$$

$$V(s_t) \leftarrow \sum_a N(s_{t+1})Q(s_t,a) / \sum_a N(s_{t+1})$$

$$N(s_t) \leftarrow N(s_t) + 1$$

Part Two: Iterative Preference Learning

Data Selection. We select the candidate steps of highest and lowest Q values as positive and negative samples at each tree depth, respectively. The parent node selected at each tree depth has the highest value calculated by multiplying its visit count and the range of its children nodes' visit counts, indicating both the quality and diversity of the generations.

Conservative DPO. We use the visit counts simulated in MCTS to apply adaptive label smoothing on each preference pair.

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

$$\ell_i(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_i}\left[(1-\alpha_{x,y_w,y_l})\log\sigma(\beta\, h_{\pi_\theta}^{y_w,y_l}) + \alpha_{x,y_w,y_l}\log\sigma(-\beta h_{\pi_\theta}^{y_w,y_l})\right]$$

$$\alpha_{x,y_w,y_l} = \frac{1}{N(x,y_w)/N(x,y_l)+1}$$

MCTS-Enhanced Iterative Preference Learning

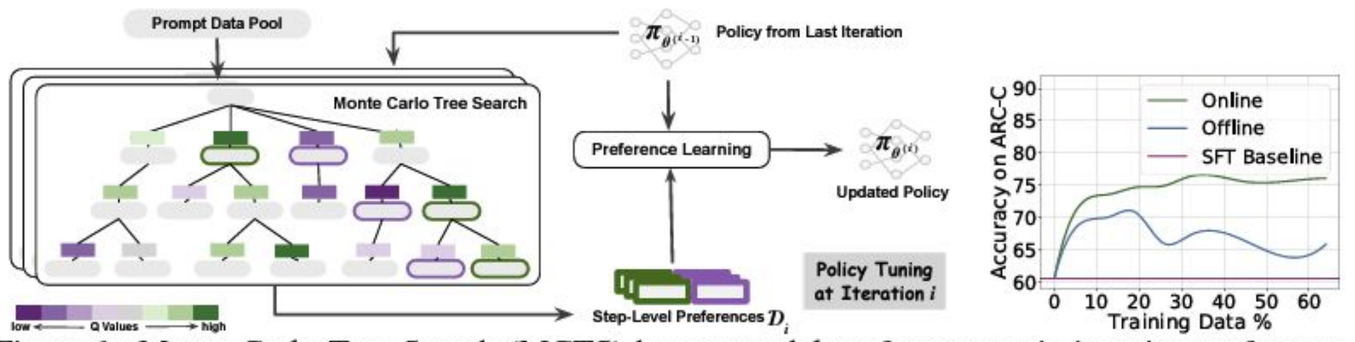

Figure 1: Monte Carlo Tree Search (MCTS) boosts model performance via iterative preference learning. Each iteration of our framework (on the left) consists of two stages: MCTS to collect step-level preferences and preference learning to update the policy. Specifically, we use action values Q estimated by MCTS to assign the preferences, where steps of higher and lower Q values will be labeled as positive and negative data, respectively. The scale of Q is visualized in the colormap. We show the advantage of the online manner in our iterative learning framework using the validation accuracy curves as training progresses on the right. The performance of ARC-C validation illustrates the effectiveness and efficiency of our proposed method compared to its offline variant.

Experiments

Commonsense Reasoning: assess the **generalizability** of our method in learning various reasoning tasks through self-distillation.

Approach	Base Model	Conceptual Comparison				ARC-c	AI2Sci-m	CSQA	SciQ	Train Data Used (%)
		NR	OG	OF	NS					
CoT Tuning	GPT-3-curie (6.7B)	✓	✗	✗	✗	–	–	56.8	–	100
Direct Tuning	GPT-J (6B)	✓	✗	✗	✗	–	–	60.0	–	100
STaR		✓	✓	✓	✗	–	–	72.5	–	86.7
Direct TUning	T5-11B	✓	✗	✗	✗	72.9	84.0	82.0	83.2	100
Crystal		✗	✓	✓	✓	73.2	84.8	**82.3**	85.3	100
SFT Base (Arithmo)	Mistral-7B	–	–	–	–	60.6	70.9	54.1	80.8	–
Direct Tuning		✓	✗	✗	✗	73.9	85.2	79.3	86.4	100
MCTS Offline-DPO		✓	✗	✗	✓	70.8	82.6	68.5	87.4	19.2
Instance-level Online-DPO		✓	✓	✓	✓	75.3	87.3	63.1	87.6	45.6
Ours		✓	✓	✓	✓	**76.4**	**88.2**	74.8	**88.5**	47.8

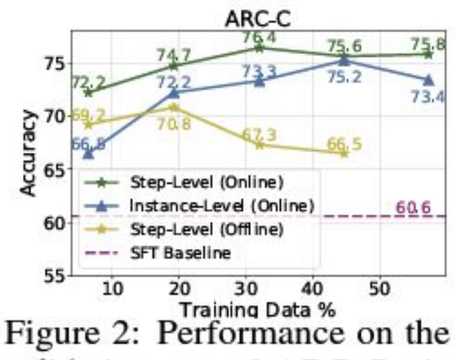

Figure 2: Performance on the validation set of ARC-C via training with different settings.

📌 Our method achieves significant improvement compared to baselines across various datasets, including *ARC-C*, *AI2Sci-mid*, and *SciQ*.

🤔 The modest performance gain on *CSQA* may attribute to two reasons:

- *CSQA* questions tend to rely more on System 1;
- The base model forgets specific knowledge or ways to utilize it for CSR after SFT training.

Arithmetic Reasoning: evaluate whether our method enhances reasoning abilities on **specific** arithmetic tasks.

Approach	Base Model	Conceptual Comparison				GSM8K	MATH
		NR	OG	OF	NS		
LMSI	PaLM-540B	✓	✓	✗	✗	73.5	–
SFT (MetaMath)	Mistral-7B	–	–	–	–	77.7	28.2
Math-Shepherd		✗	✓	✗	✓	**84.1**	33.0
SFT (Arithmo)		–	–	–	–	75.9	28.9
MCTS Offline-DPO		✓	✗	✗	✓	79.9	31.9
Instance-level Online-DPO		✓	✓	✓	✓	79.7	32.9
Ours		✓	✓	✓	✓	80.7	32.2
Ours (w/ G.T.)		✓	✓	✓	✓	81.8	**34.7**

- **NR**: w/o reward model
- **OG**: on-policy generation
- **OF**: online (on-policy) feedback
- **NS**: w/ negative samples

📌 When base model can produce reasoning chains with high quality, the online and offline settings perform comparably.

📌 Further tuning on arithmetic data may cause overfitting, leading to
- weaker exploration ability → lower diversity in generated data
- stronger restriction from the KL regularization for not deviating a lot from the base SFT distributions.

Oracle-Guided Self-Evaluation. The G.T. (example) answer is crucial to ensure the reliability of self-evaluation.

Approach	GSM8K		MATH		ARC-C	
	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
w/ example answer	74.7	72.5	76.6	48.8	65.2	57.5
w/o example answer	62.0	69.5	48.1	42.3	55.8	48.4