

Training Large Language Models to Reason via EM Policy Gradient

Tianbing Xu

Learning to Reason Goal

$$J(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] . \quad (3)$$

Our goal is to optimize the expected return with respect to rationale trajectories sampled from the distribution $P_{\theta}(\tau)$, where the latent rationales are generated by the model π_{θ} .

The policy gradient is given by:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [\nabla \log P_{\theta}(\tau) R(\tau)] . \quad (4)$$

Learning to Reason in the EM framework

We begin by deriving a lower bound for the logarithm of the objective function (3):

$$\begin{aligned}\log J(\theta) &= \log \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] \\ &\geq \mathbb{E}_{\tau \sim q(\tau)} [\log R(\tau) + \log P_{\theta}(\tau) - \log q(\tau)] .\end{aligned}$$

where the inequality follows from Jensen's inequality by introducing a variational distribution $q(\tau)$.

E-Step: Sample Rationale Trajectory

The lower bound becomes:

$$L(\theta^*, q) = -D_{KL}(q(\tau) || P_{\theta^*}(\tau)R(\tau)) \quad (7)$$

To minimize the KL divergence (and thereby maximize the lower bound $L(\theta^*, q)$), we seek to match the variational distribution $q(\tau)$ with the optimal distribution $q^*(\tau)$, defined as:

$$q^*(\tau) \propto P_{\theta^*}(\tau)R(\tau) \quad (8)$$

M-Step: Optimize the Rationale Trajectory Model

We can rewrite the lower bound from Eq. (9) with respect to θ (ignoring constants, or up to an additive constant) as a surrogate off-policy objective:

$$L(\theta) = \mathbb{E}_{\tau \sim D} [\log P_{\theta}(\tau) R(\tau)] , \quad (10)$$

with the corresponding off-policy policy gradient given by:

$$\nabla_{\theta} L(\theta) = \mathbb{E}_{\tau \sim D} [\nabla_{\theta} \log P_{\theta}(\tau) R(\tau)] , \quad (11)$$

Algorithm

Algorithm 1 EM Policy Gradient Algorithm

- 1: **Initialize:** $\theta_0 \leftarrow$ base LLM parameter
 - 2: **for** $t = 1$ to T **do**
 - 3: Reset buffer: $\mathcal{D} \leftarrow \emptyset$
 - 4: Set old model parameter: $\theta^* \leftarrow \theta_{t-1}$
 - 5: **E-Step: Sample Rationale Trajectories from** $P_{\theta^*}(\tau)$
 - 6: **for** each input prompt $x^i \sim Q(x)$ **do**
 - 7: Sample multiple intermediate rationales and generate responses:
 $y^i, Z^i \sim P_{\theta^*}(Z, y \mid x = x^i)$
 - 8: Store trajectory: $\mathcal{D} \leftarrow \mathcal{D} \cup \{x^i, Z^i, y^i\}$
 - 9: Evaluate reward: $r^i = R(x^i, Z^i, y^i)$
 - 10: **end for**
 - 11: **M-Step: Optimize Rationale Trajectory Model**
 - 12: Update θ_t : Fine-tune $P_\theta(Z, y \mid x)$ using the off-policy policy gradient (Eq. 12) on samples from \mathcal{D} .
 - 13: **end for**
-

Experiment

Base Model

QWen2.5-1.5B, QWen2.5-7B, QWen2.5-Math-7B, QWen2.5-14B-Instruct

4 A100 GPU (8 A100 for 14B)

Test Score Comparison

| Base Model | EMPG Test Score | GRPO Test Score |
|----------------------|-----------------|-----------------|
| QWen2.5-1.5B | 0.744 | 0.739 |
| QWen2.5-7B | 0.888 | 0.867 |
| QWen2.5-Math-7B | 0.918 | 0.004* |
| QWen2.5-14B-Instruct | 0.952 | 0.940 |

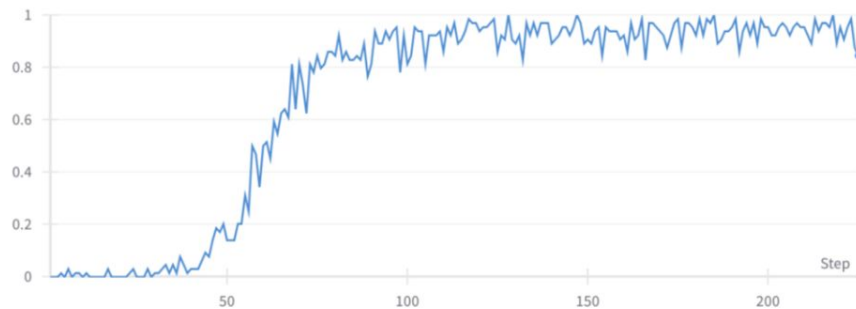
Table 1: EMPG and GRPO test scores on for GSM8K Dataset for QWen2.5 base models. (* GRPO failed to learn in this setting.)

Test Score Comparison

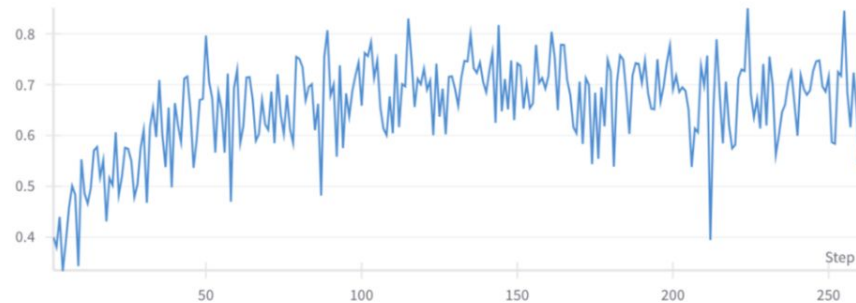
| Base Model | EMPG Test Score | GRPO Test Score |
|----------------------|-----------------|-----------------|
| QWen2.5-1.5B | 0.575 | 0.568 |
| QWen2.5-7B | 0.742 | 0.730 |
| QWen2.5-Math-7B | 0.794 | 0.755 |
| QWen2.5-14B-Instruct | 0.808 | 0.816 |

Table 2: EMPG and GRPO test scores on the MATH HARD dataset for QWen2.5 base models.

Learning Curves



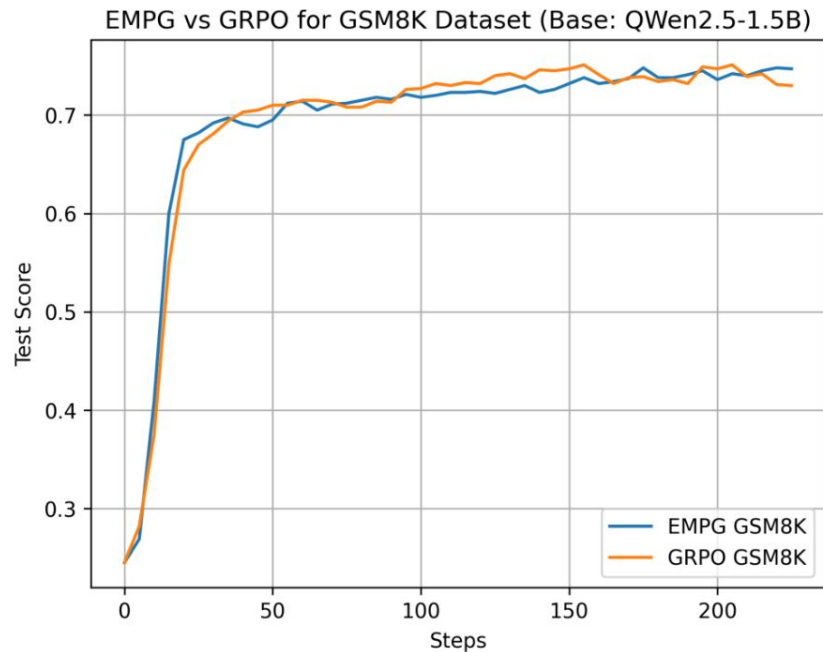
(a) GSM8K



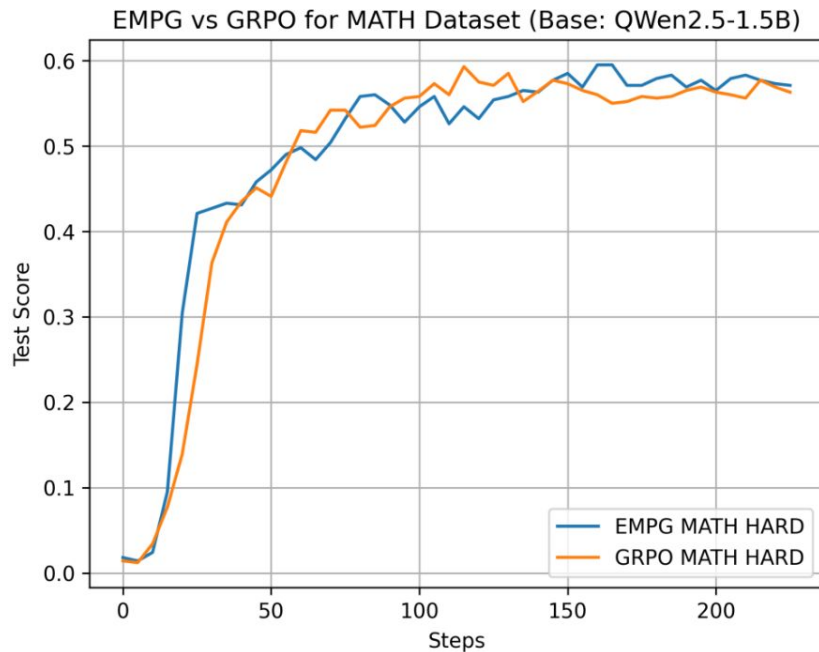
(b) MATH HARD

Figure 2: EMPG Learning Curves (Base: QWen2.5-Math-7B)

Learning from Weak Models (1.5B)



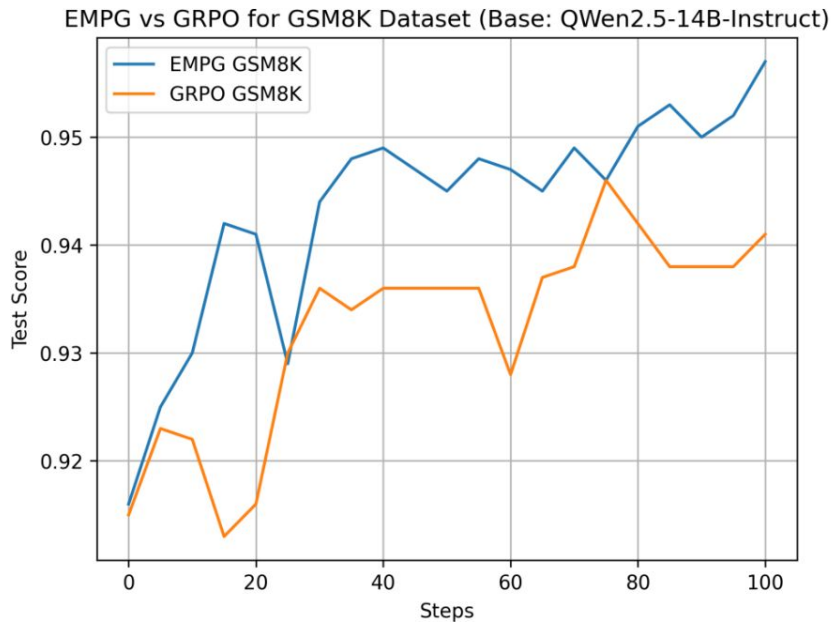
(a) GSM8K



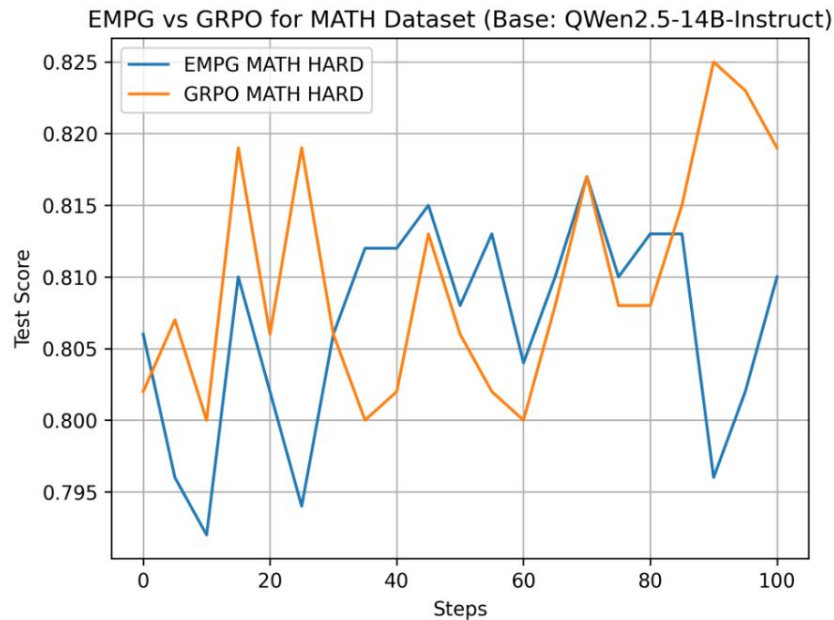
(b) MATH HARD

Figure 3: EMPG vs GRPO (Base: QWen2.5-1.5B)

Learning from Strong Models (14B)



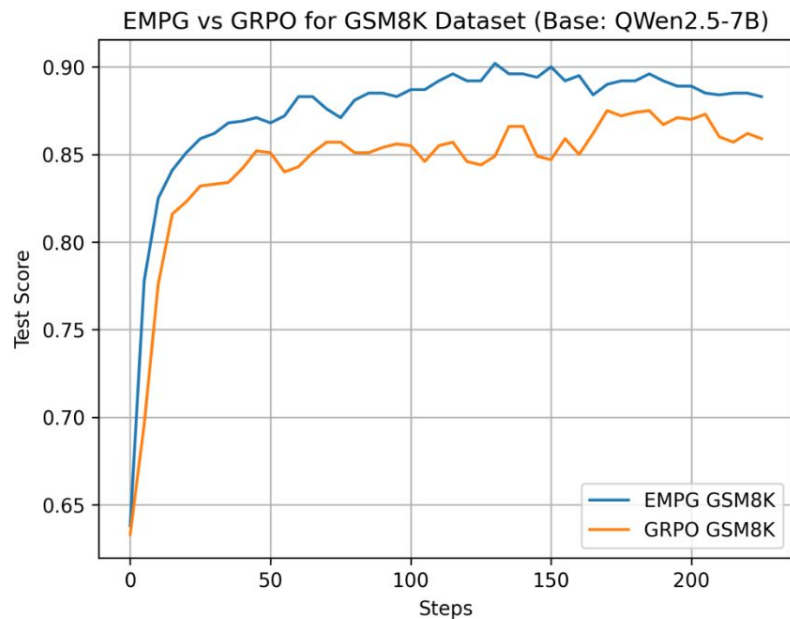
(a) GSM8K



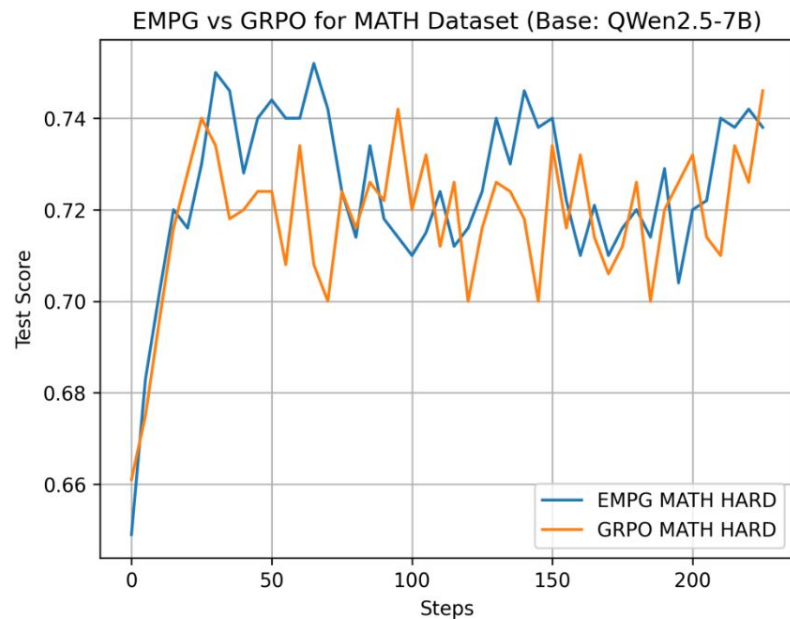
(b) MATH HARD

Figure 6: EMPG vs GRPO (Base: QWen2.5-14B-Instruct)

Learning from Medium-Sized Models (7B)



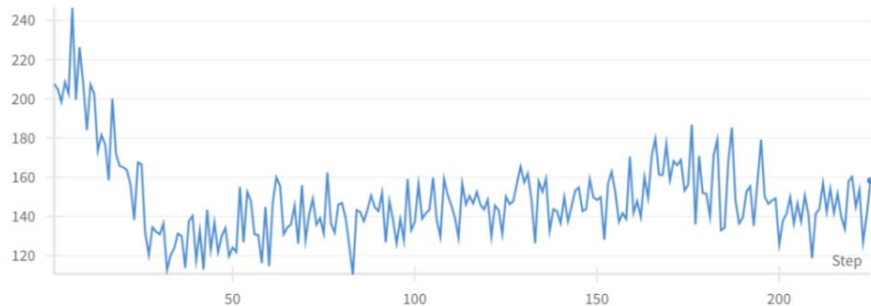
(a) GSM8K



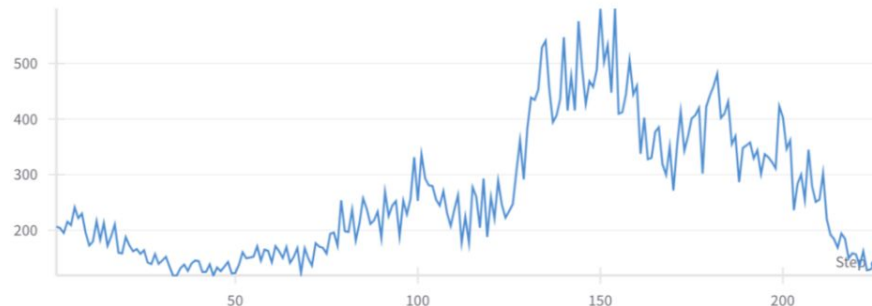
(b) MATH HARD

Figure 4: EMPG vs GRPO (Base: QWen2.5-7B)

Reasoning Length: Short or Long CoT (1.5B)



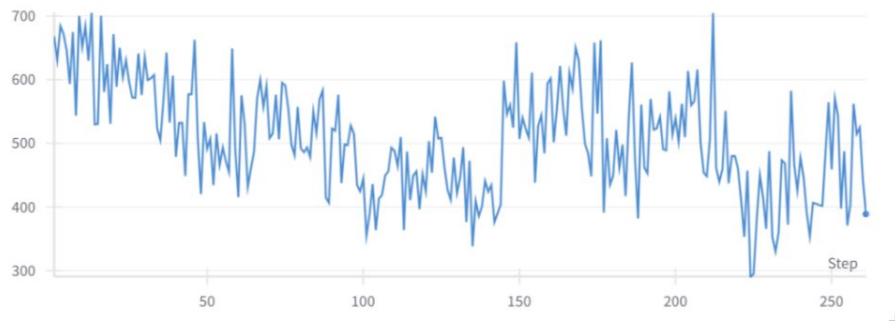
(a) EMPG



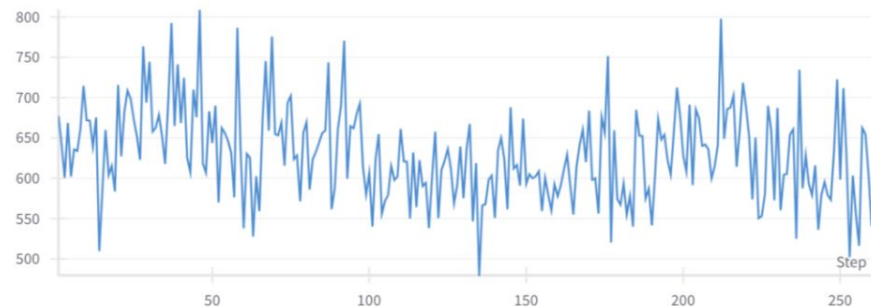
(b) GRPO

Figure 7: EMPG vs GRPO Reasoning Length (GSM8K, Base: QWen2.5-1.5B)

Reasoning Length: Short or Long CoT (7B)



(a) EMPG



(b) GRPO

Figure 8: EMPG vs GRPO Reasoning Length (MATH, Base: QWen2.5-7B)

Case Study: Learned Cognitive Reasoning Behaviors

Problem (HMMT 2009):

A torus (donut) having inner radius 2 and outer radius 4 sits on a flat table. What is the radius of the largest spherical ball that can be placed on top of the center torus so that the ball still touches the horizontal plane? (skip ...)

COT and Response:

Let's solve the problem step-by-step. We'll first understand the geometry and then use Python and `sympy` to find the radius of the largest spherical ball that can be placed on top of the center of the torus while still touching the horizontal plane.

1. Understanding the Geometry:

- The torus has an inner radius of 2 and an outer radius of 4. The center of the tube of the torus is located at $(3, 0, 1)$, and the radius of the tube is 1. The torus is formed by revolving this circle around the z -axis. (skip ...)

2. Formulating the Problem:

- The distance from the center of the spherical ball (at $(0, 0, h)$) to the center of the tube of the torus (at $(3, 0, 1)$) must be equal to $r - 1$ (because the ball touches the inner surface of the torus). (skip ...)

3. Solving the Equation:

- Square both sides of the equation to remove the square root: $9 + (r - 1)^2 = (r - 1)^2$.

- This simplifies to $9 = 0$, which is **incorrect**. We need to **re-evaluate** the distance condition.

Let's correct and solve this step-by-step in Python using `texttt{sympy}`: (skip ...)

Output: `IndexError: list index out of range`

It seems there was an issue with the previous approach. Let's re-evaluate the problem more carefully. The correct approach should be to solve the equation derived from the geometry of the problem correctly.

1. Re-evaluate the Geometry:

- The center of the spherical ball is at $(0, 0, r)$.

- The distance from the center of the spherical ball to the center of the tube of the torus (which is at $(3, 0, 1)$) is $\sqrt{3^2 + (r - 1)^2} = \sqrt{9 + (r - 1)^2}$.

- This distance must be equal to $r + 1$.

2. Formulate the Equation:

- We have $\sqrt{9 + (r - 1)^2} = r + 1$.

- This simplifies to $9 = 4r$, so $r = \frac{9}{4}$.

Let's solve this using Python and `sympy`:

```
import sympy as sp

# Define the radius of the spherical ball
r = sp.symbols('r')

# The equation we derived
distance_eq = sp.sqrt(9 + (r-1)**2) - (r + 1)

# Solve the equation
solution = sp.solve(distance_eq, r)

# Filter out the valid solution
valid_solution = [sol.evalf() for sol in solution if sol > 0]

print(valid_solution[0])
```

Output: $9/4$

Final Answer: The radius of the largest spherical ball that can be placed on top of the center of the torus and still touch the horizontal plane is $\frac{9}{4}$.

Figure 14: MATH Problem Example: Step-by-Step, Backtracking, Reflection, Self-Verification

Reference

Tianbing XU, [Training Large Language Models to Reason via EM Policy Gradient](#), 2025

Schulman, et. al, [Proximal Policy Optimization Algorithms](#), 2017

OpenAI, [Learning to Reason with LLMs](#), 2024

DeepSeekAI, [DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning](#), 2025