LLM-jp FT-LLMコンペにおける 数学推論能力向上の取り組み

尾崎 大晟[1], 力岡 友和[1], 渡部 泰樹[1], Jeong Seong Cheol[2]
[1]株式会社松尾研究所 , [2]東京大学 松尾・岩澤研究室

1. はじめに

LLM-jp FT-LLMコンペの概要

タスク llm-jp-4-8bをベースモデルとし、中高レベルの数学問題500問の正答率を競う

制約 推論時はベースモデル由来のモデルのみ使用可能
（外部LLMやインターネットアクセスは禁止）

我々のチームの取り組みと戦略

1. **コンテキスト長の拡張**
長い推論過程を途切れずに扱えるように**YaRN**を用いてコンテキスト長を拡張

2. **合成推論データの構築**
gpt-oss-120bを活用し、CoTおよびTIRの推論データを合成

3. **SFT・GRPOによる学習**
合成データを用いた**SFT**および**GRPO**による強化学習で推論能力を更に底上げ

2. 予備調査

ベースライン実験（ llm-jp-4-8bの精度調査）

llm-jp-4-8bの推論性能は「使用するプロンプト」と「使用言語」に大きく依存することが判明

モデル	検証データに対する正答率
[llm-jp-4-8b] 単純な指示文プロンプト	0.08
[llm-jp-4-8b] SFT用プロンプト（日本語）	0.32
[llm-jp-4-8b] SFT用プロンプト（英語）	**0.40**
（参考）gpt-oss-120b（日本語）	0.81
（参考）Qwen3-235B-A22B-Thinking（英語）	0.74

3. 合成推論データの構築

1. 既存の数学 QAデータセットを前処理

GSM8K　StackMathQA

2. CoT（Chain-of-Thought）拡張

- 問題ごとに推論過程を3候補生成
- 「直接性・明瞭性・完全性・効率性」をLLM-as-a-Judge方式で評価し、最高スコアの候補を選択

3. TIR（Tool-Integrated Reasoning）拡張

- 推論過程 + 実行可能なPythonコードを3候補生成
- 「コード正確性・推論明瞭性・完全性」をLLM-as-a-Judge方式で評価し、最高スコアの候補を選択、後処理で実行検証を実施

4. 品質管理とマージ

- 品質をLLM-as-a-Judge方式で再評価し、総合スコア7以上を有効データとして採用

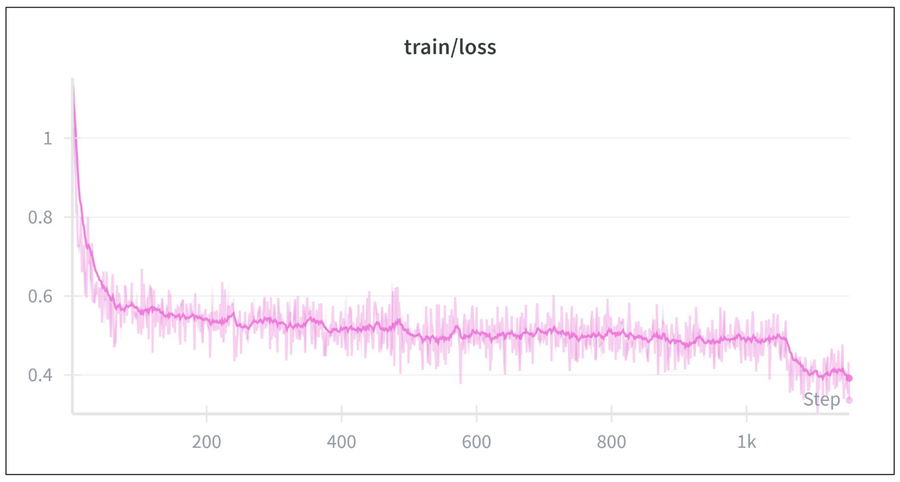
約36,000件のCoT, TIR形式のSFTデータを作成

※ TIRはコード生成が安定しなかったため、後続の検証は割愛する

4. SFT・GRPO

SFT

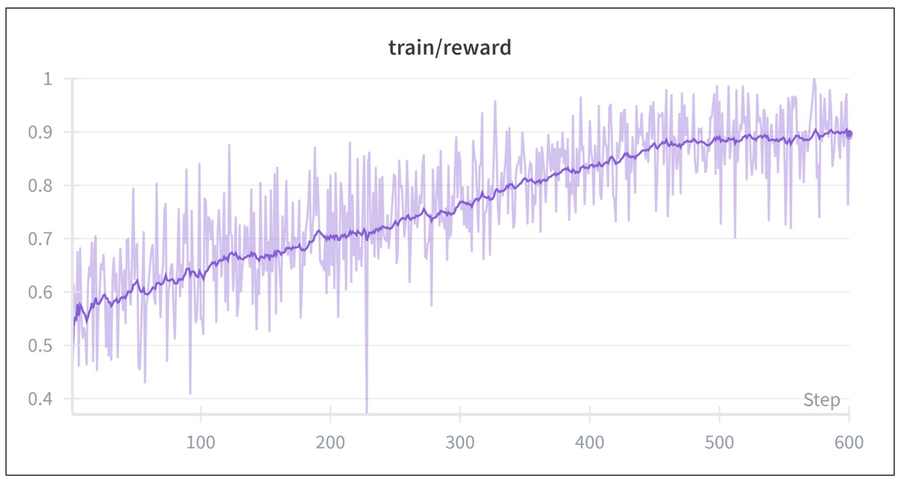
- 合成パイプラインで生成した**CoTデータ約3.4万件**を用いて教師ありファインチューニング（SFT）を実施
- 最大系列長 **8,192トークン** で **3エポック** 学習



SFTにおける訓練損失の推移

GRPO

- 手作業で作成した**数学問題180件**を用いてGRPOを実施
- 最大系列長 **16,384トークン** で **600ステップ** 学習
- 数学的な正誤判定 + LLM部分点 を組み合わせた報酬設計を採用



GRPOにおける訓練報酬の推移

5. 推論パイプライン

1. 問題の日英翻訳

- 日本語の数学問題を独自モデルで英語に翻訳（1問につき4つの翻訳候補を生成）
- 理由：ベースモデルが英語の数学問題に対して高い推論性能を発揮するため

2. マルチエージェント並列推論

- 8つの独立エージェントで並列推論を実行
- 汎用モデルと高難度特化モデルの2種を混在させ、アンサンブル効果を狙う
- 1問あたり 4（翻訳候補）×8（エージェント）×5（サンプル）= 160 個の回答候補を生成

3. 回答抽出と多数決投票

- math-verify ライブラリを用いた「数学的等価性判定」に基づいた多数決を実施
- 最大グループを最終回答とする

検証データにおいて、正答率　0.70を達成