Math Fine-Tuning in 2025: SFT is Dead, Long Live GRPO

If you are still fine-tuning 7B models on math datasets using standard Supervised Fine-Tuning (SFT), you are essentially teaching the model to *mimic* the appearance of a solution rather than the act of solving it. The signal from 2024/2025 is clear: **inference-time compute** is not just a prompting trick; it is a training objective.

The breakthrough for local labs is that we no longer need the massive infrastructure required for PPO (Proximal Policy Optimization). We have moved to **GRPO (Group Relative Policy Optimization)**, which removes the need for a critic model and fits reasoning training onto a single consumer GPU.

The Shift: From Imitation (SFT) to Incentive (GRPO)

In SFT, the model learns to predict the next token in a "Chain of Thought" (CoT) trace. The problem? If the trace contains a logical jump or a hallucination, the model memorizes the error just as firmly as the truth.

GRPO changes the game by sampling a *group* of outputs for the same prompt and optimizing based on a reward function (correctness of the final answer). The policy is updated to maximize the probability of outputs that score higher than the group average.

Crucially, the baseline is computed from the group mean, not a separate value network. This cuts VRAM usage nearly in half, making it possible to train reasoning behaviors on a 24GB card.

The Stack: Unsloth + Qwen-2.5-Math

The most viable path for a home lab right now involves **Unsloth**. They have integrated GRPO directly into their training pipeline. You don't need a complex reward model for math; you just need a deterministic way to check the final answer.

Here is the stripped-down logic for a local GRPO run:

```
from unsloth import FastLanguageModel, PatchFastRL
PatchFastRL("GRPO", FastLanguageModel)

# 1. Load a strong base model (Qwen-2.5-Math-7B is excellent)
model, tokenizer = FastLanguageModel.from_pretrained(
    "Qwen/Qwen2.5-Math-7B",
    load_in_4bit = True
)

# 2. Define a verifiable reward function
def reward_correctness(prompts, completions, answer, **kwargs):
    # Extract answer from completion and match with ground truth
   return [1.0 if extract_answer(c) == a else 0.0 for c, a in zip(completions, answer)]

# 3. Train with Group Sampling (GRPO)
trainer = GRPOTrainer(
    model = model,
    reward_funcs = [reward_correctness],
    args = GRPOConfig(
```

```
        per_device_train_batch_size = 1,
        gradient_accumulation_steps = 4,
        num_generations = 4, # Group size
        max_prompt_length = 512,
        max_completion_length = 1024,
    ),
)
trainer.train()
```

Why this matters for Mathematicians

This approach allows us to train models that **self-correct**. By rewarding only the final answer, we allow the model to "learn" that spending more tokens to double-check its work (or backtrack) leads to a higher reward. You aren't forcing a specific proof style; you are incentivizing the *process* of being right.

Contributor: Alessandro Linzi