Assignment on Reinformate Learning

Nasiruddin Kabir

QLoRA: Efficient Finetuning of Quantized LLMs

1. **What is QLoRA and how does it differ from standard LoRA?**

   QLoRA (Quantized Low-Rank Adaptation) is a memory-efficient fine-tuning method that combines low-rank adaptation (LoRA) with 4-bit quantization of base model weights. Unlike standard LoRA which uses full-precision weights, QLoRA stores the base model in 4-bit quantized form (e.g., using NF4), enabling fine-tuning of large models on low-resource hardware.

2. **Explain the role and benefits of the NormalFloat4 (NF4) quantization format.**

   NormalFloat4 (NF4) is a 4-bit quantization format that allocates more precision around zero, aligning better with the normal distribution of model weights. It leads to lower quantization error, better preservation of weight information, and improved model accuracy compared to uniform quantization.

3. **What is "double quantization" and why is it useful?**

   Double quantization refers to first quantizing model weights to 4-bit and then compressing the quantization constants (like scale and zero points) using an 8-bit quantizer. It reduces memory usage further, saving around 0.37 bits per parameter with negligible performance degradation.

4. **How do paged optimizers help in memory management during training?**

   Paged optimizers manage optimizer states (e.g., momentum) by offloading them to CPU memory or disk, loading them into GPU only when needed. This allows training large models on GPUs with limited memory by significantly reducing active VRAM usage.

5. **Why is QLoRA significant in enabling large model finetuning on limited hardware?**

QLoRA enables fine-tuning large models like LLaMA-65B on consumer GPUs (e.g., 24GB) by combining 4-bit quantization, LoRA layers, double quantization, and paged optimizers. This democratizes LLM fine-tuning by reducing memory and compute requirements.

6. **Suggest one possible improvement or variation to the QLoRA method. Why might it help?**

   One improvement could be adaptive LoRA: dynamically adjusting the rank of LoRA adapters per layer based on gradient importance. This could improve efficiency and performance by allocating more resources to layers that matter most.

PokerGPT: Lightweight Solver for Multi-Player Poker

1. **Describe the overall pipeline of PokerGPT from raw data to trained model.**

   The pipeline includes: (1) Collecting expert-level or simulated poker hands, (2) supervised fine-tuning of a pretrained LLM, (3) reinforcement learning with human feedback (RLHF) to optimize strategic decisions, and (4) evaluation using simulated matches.

2. **What makes PokerGPT different from traditional poker solvers like CFR or DeepStack?**

   PokerGPT treats poker as a language modeling task, unlike CFR or DeepStack which rely on game-theoretic or search-based techniques. It does not use abstraction or tree search, enabling data-driven, scalable strategy learning using LLMs.

3. **Why is RLHF used in PokerGPT and how does it affect decision-making quality?**

   RLHF fine-tunes the model based on win rates or human preferences. It enhances decision quality by encouraging strategies aligned with successful outcomes, going beyond supervised imitation to strategic optimization.

4. **What are the advantages of using LLMs for multiplayer poker games?**

   Advantages include: understanding of complex contexts, scalability to new game types, natural generalization, and potentially interpretable decisions through language-like reasoning.

5. **What challenges might arise when scaling PokerGPT to more complex games like Omaha?**

Challenges include: larger state space, longer action sequences, difficulty collecting quality data, and increased variance in outcomes, making training more complex and unstable.

6. **Suggest one possible extension or improvement to PokerGPT and explain its benefit.**

   Incorporating a belief module to model opponent strategies could enhance adaptability and bluff detection, improving performance in dynamic multi-agent environments.

GRPO (Group Relative PPO) – DeepSeekMath 7B

1. **What is the main idea behind GRPO and how does it differ from traditional PPO?**

   GRPO uses relative ranking of outputs within a batch for optimization, rather than absolute reward values like PPO. This makes it more stable and suited for preference-based or subjective reward scenarios.

2. **How does GRPO eliminate the need for a separate value network?**

   GRPO relies on intra-batch ranking rather than estimating future rewards. Since there's no need for value prediction, the value network is not required, simplifying the training process.

3. **What is the significance of using z-score normalization for reward signals?**

   Z-score normalization centers and scales rewards, making training more stable and consistent across batches. It prevents issues caused by varying reward magnitudes and helps avoid gradient instability.

4. **Why might GRPO be better suited for math reasoning tasks than PPO?**

   In math reasoning, outputs may be partially correct. GRPO's relative reward framework captures quality differences more effectively than binary or scalar rewards used in PPO, improving learning from nuanced outputs.

5. **What are potential weaknesses of GRPO when all sampled outputs are poor?**

   GRPO may still reinforce the "least bad" output, even if it's incorrect, leading to learning of flawed patterns and model drift in low-quality environments.

6. **Suggest a modification to GRPO to make it more robust to low-quality output groups.**

   A hybrid approach combining GRPO with minimum quality thresholds or absolute rewards could help. Alternatively, filtering low-quality groups before training updates can prevent reinforcing incorrect outputs.