Poker Zero Lightning Presentation

Problem Statement and Motivation

Objective

Develop a reasoning model optimized for **no-limit hold'em poker**. Success in poker extends beyond monetary gain—it signifies advancements in:

- **Reasoning under incomplete information**
- **Adversarial decision-making**
- **Strategic adaptation**

Applications

Poker AI has significant implications in:

- **Game Theory**
- **Economic Modeling**
- **Negotiation**

Prior Work

Several notable AI-driven poker models include:

- **Poker Bench** – Trained LLMs to become professional poker players
- **PokerGPT** – Lightweight solver leveraging a large language model
- **Pluribus** – Demonstrated multiplayer, near-GTO strategies

Data

Instruction	Output
You are a specialist in playing 6-handed No Limit Texas Hold'em. The following will be a game scenario, and you need to make the optimal decision.	Bet 18
Game Summary:	
- Small Blind: **0.5 chips**	
- Big Blind: **1 chip**	
- Everyone started with **100 chips**	
- Player positions: **UTG, HJ, CO, BTN, SB, BB**	
- **Your Position:** HJ	
- **Your Hand:** [King of Diamond, Jack of Spade]	

Technical Approach

Unsloth

Unsloth is an open-source Python framework that speeds up the process of fine-tuning and using LLMs.

- Optimized PyTorch code
- Handwriting GPU kernels to speed up inference
- Better memory utilization via typecasting
- Allows us to fine-tune 8B parameter models on just a Colab T4 GPU

GRPO

GRPO (Group Relative Policy Optimization) is a reinforcement learning algorithm developed by DeepSeek.

- Trains a model to optimize a reward function instead of training a model solely on next-token prediction (which simply teaches it to mimic data)
- Uses group-based comparisons to improve performance instead of after every trial
- Calculates advantage and updates policy to increase likelihood of better actions
- Uses KL Divergence constraint to prevent drastic changes in policy
- Overall objective maximizes cumulative reward with stable policy updates

LoRA

Low Rank Adaptation (LoRA) is a method for fine-tuning large models efficiently.

- It works by introducing low-rank matrices into pretrained model layers.
- This approach allows significant performance gains with a minimal number of additional parameters.

Mathematical Formulation of LoRA

- **Pretrained Weight Matrix:** Let $W \in \mathbb{R}^{d \times k}$ be a pretrained weight matrix.
- **Low-Rank Decomposition:** LoRA approximates the weight update ΔW as:
  $\Delta W = BA$
  where:
- $B \in \mathbb{R}^{d \times r}$
- $A \in \mathbb{R}^{r \times k}$
- $r \ll \min(d, k)$
- **Adapted Weights:** The updated weight matrix becomes: $W' = W + BA$

Implementation Updates

Training Infrastructure

- **Unsloth Optimization for T4 GPUs** - Successfully migrated training from high-end A100 GPUs to more accessible T4 GPUs
- More cost-effective while maintaining training quality
- Makes project more accessible with consumer-grade hardware (though training takes longer)

Response Structure Revisions

Implemented structured output format to encourage explicit reasoning:

- Encourages clear chain of thought before decision-making
- Makes reasoning process transparent and evaluable
- Critical for both training and human verification

Enhanced Reward Functions

Redesigned reward functions to reinforce structured reasoning:

- **Structure Compliance**: Rewards for properly using defined tag structure
- **Reasoning Quality**: Evaluates content within reasoning tags for depth and logical consistency
- **Decision Alignment**: Ensures final decision logically follows from reasoning
- More lenient criteria to allow for more frequent positive reinforcement

Reinforcement Learning Implementation

Self-Play with PPO

- Implemented Proximal Policy Optimization for self-play training
- 6-player poker games where model instances compete against each other
- Progressive refinement through competitive environments

GRPO Training Challenges

- Computationally intensive reward distribution
- Complex processing of numerous variables and potential outcomes
- Extended runtime for complete training cycles

Next Steps

Training Efficiency

- Decrease fine-tuning and self-play training time
- Explore smaller models or less compute-intensive algorithms
- Refine reward functions to prevent reward hacking
- Convert pypokerengine outputs into data format similar to Poker Bench data

Evaluation Methods

- Current: tracking average chip difference over many games
- Potential: evaluate on test-split of PokerBench dataset
- Explore more objective Game Theory Optimal (GTO) result comparison

Thank You!

Questions?