Intro to AI

Phase 2: Term project mid-status report

Development results

📌 Project Title: Solving the 2×2×2 Rubik's Cube with Feature-Based Reinforcement Learning

📂 GitHub Repository: https://github.com/RL-Rubiks-Solver/Rubiks-Cube-Solver

Team Members: Stuti Bimali, Binay Dhakal

Research Questions:

- Can an agent learn to solve a Rubik's Cube using **model-free reinforcement learning**?
- How does the performance vary across **Q-Learning**, **Policy Gradient (REINFORCE)**, and **PPO** (Proximal Policy Optimization)?

Introduction:

Rubik's Cube is a classic artificial intelligence challenge that demands strategic reasoning and sequential decision-making. In this project, we investigate how reinforcement learning (RL) agents can learn to solve the cube through trial and error, without prior knowledge or heuristics. To manage complexity and speed up training, we focus on the 2x2x2 variant of the cube. Our primary objective is to evaluate and compare the performance of various model-free RL algorithms—both classical and modern—on this task. Specifically, we analyze their learning efficiency, convergence behavior, and solution accuracy, with the broader goal of understanding how different RL strategies perform in solving discrete, combinatorial problems.

Related Literature:

1. **Solving the Rubik's cube with deep reinforcement learning and search ->** https://www.nature.com/articles/s42256-019-0070-z
2. **Solving the Rubik's cube with deep reinforcement learning and search ->** https://deepcube.igb.uci.edu/static/files/SolvingTheRubiksCubeWithDeepReinforcementLearningAndSearch_Final.pdf
3. **Solving a Rubik's Cube with Approximate Q-Learning ->** https://medium.com/analytics-vidhya/solving-a-rubiks-cube-with-reinforcement-learning-part-1-4f0405dd07f2

Our Work Differs By:

- Solving the 2x2x2 cube to maintain tractability and comparing multiple model-free RL algorithms:
  1. Q-Learning (Tabular)
  2. Policy Gradient (REINFORCE with NN)
  3. PPO (Clipped Policy Optimization with Actor-Critic)

Achievements and Challenges:

✅ Achievements:

1. Developed a fully functional 2x2x2 Rubik's Cube environment in Python from scratch and implemented and trained a tabular Q-Learning agent with feature-based state vectors.
2. Added Policy Gradient and PPO agents per instructor's feedback.
3. Created tools to: Log training statistics, comparing agent performance, Visualize learning curves and success rates

⚠ Challenges Faced:

1. Learning and coding Q-Learning, Policy Gradient, and PPO from scratch without using libraries like Stable Baselines was challenging.
2. Creating a reversible, valid 2x2x2 cube model and a compact, effective feature representation required careful design.
3. PPO and PG were unstable and hard to debug due to sparse rewards and sensitivity to hyperparameters.
4. Built custom tools for training analysis and managed scope expansion after feedback without relying on external frameworks.

Preliminary Results:

PPO showed the most stable learning and highest success rate, outperforming both Q-Learning and Policy Gradient. Q-Learning was inconsistent, while Policy Gradient performed moderately but was sensitive to tuning.