Reinforcement Learning

Eureka For Car Racing

Implemented by SI5-IA/ID-FISA&FISE Mariia BERDNYK & Sayf Eddine HALMI

Introduction to Eureka

On April 30, 2024, at ICLR, a new paper titled "Eureka: Human-Level Reward Design via Coding Large Language Models" by Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi "Jim" Fan, and Anima Anandkumar was published, attracting attention from 191 sites by October 31, 2024.

The Eureka algorithm leverages state-of-the-art large language models (LLMs) like GPT-4 for zero-shot generation, code-writing, and in-context improvement, applying these capabilities to evolutionary optimization over reward code. The generated rewards help train agents to acquire complex skills via reinforcement learning, outperforming expert human-engineered rewards without any task-specific prompting or predefined reward templates.

Context and Goal

The authors suggest this algorithm can extend to various environments to enhance agent performance by generating new reward functions. Our objective was to validate this claim by applying the Eureka algorithm to a complex stochastic gym environment, **CarRacing**, where the agent's primary goal is to complete the track as quickly and steadily as possible.

The CarRacing environment is considered inherently stochastic due to the following factors:

- **Track Variability**: The environment generates procedurally different tracks for each episode, requiring the agent to adapt its driving skills to handle various terrains and turns instead of memorising a single track.
- **Physics and Control Variability**: Small changes in physics calculations (e.g., friction, acceleration) and control precision can yield different trajectories, especially at high speeds.

Designing a reward function for this environment is non-trivial, as the reward depends on multiple factors, including the agent's *speed*, *position*, *acceleration*, *friction*, and *steering*. Effective reward construction is essential to guide the agent in completing the track efficiently and stably - something not feasible with manual reward engineering.

Our Work

To adapt the Eureka algorithm for this specific task, we began by designing task-specific prompts and system descriptions (The easiest part of the project). We also replaced OpenAI's GPT model with the self-hosted, free-to-use Ollama Llama 3.1. Although Ollama requires a lot of local resources, especially GPU, utilising an NVIDIA RTX 4060 GPU (6GB memory) made it even more efficient than sending the request through the network on remote OpenAI services and receiving the response back.

We also tailored the reward reflection component of the Eureka algorithm by calculating two performance metrics in this environment: **average reward** and **success rate**. The success rate, S, measures how often an agent's total reward for an episode, R_i), meets or exceeds a predefined threshold T (set to 100):

$$S = \frac{\sum_{i=1}^{N} 1(R_i \geq T)}{N}$$

where N is the number of episodes, and $1(R_i \geq T)$ is an indicator function that equals 1 if $(R_i \geq T)$ (success) and 0 otherwise. Thus, S represents the proportion of episodes where the reward surpassed the threshold T. (This part can be improved. The predefined threshold T can be tuned better according to the rewards received, for example)

Reinforcement Learning Implementation

For training, we used the Proximal Policy Optimization (PPO) algorithm implemented in the ***rl-baseline3-zoo*** library, due to:

1. The availability of pretrained models using base reward functions, enabling performance comparisons in identical conditions (excluding reward function).
2. PPO's use in the original Eureka paper, allowing us to replicate their conditions closely.

PPO is particularly effective here due to:

- **Continuous Action Space Handling**: CarRacing involves steering, acceleration, and braking - continuous actions well-suited to PPO's capabilities.

- **Stable and Sample-Efficient Learning**: PPO stabilises learning by constraining policy updates, reducing abrupt policy changes that could lead to crashes (e.g. where an agent is just stuck in one place and does not perform any action as it is too scared to get a negative reward).
- **Efficient Exploration**: Car racing requires exploration of optimal strategies for navigating tracks. PPO's exploration of long-term rewards helps agents achieve better control and speed.

Results

Overall, the Eureka algorithm enhanced agent performance in the CarRacing environment, achieving better results than baseline models. It has discovered a bug in a gym environment, which allows it to complete all the ties without following the full track and without being penalised by exiting the track. However, the trajectory of an agent is still not ideal, indicating that applications like autonomous vehicles may benefit from additional approaches beyond PPO - such as integrating PPO with human feedback for more refined outcomes (e.g. REvolve).