THÔNG TIN CHUNG CỦA NHÓM

- Link YouTube video của báo cáo (tối đa 5 phút):
    - https://youtu.be/K7y9DXtyBOg
      (ví dụ: https://www.youtube.com/watch?v=AWq7uw-36Ng)
- Link slides (dạng .pdf đặt trên Github của nhóm):
    - https://github.com/phchynhuu/CS2205.CS2205/blob/master/PhamHuynhPhuc-CS2205.NOV2024.Slides.pdf
      (ví dụ: https://github.com/mynameuit/CS2205.xxx/TenDeTai.pdf)
- *Mỗi thành viên của nhóm điền thông tin vào một dòng theo mẫu bên dưới*
- *Sau đó điền vào Đề cương nghiên cứu (tối đa 5 trang), rồi chọn Turn in*
- *Lớp Cao học, mỗi nhóm một thành viên*

- Họ và Tên: Phạm Huỳnh Phúc
- MSSV: 230101015

- Lớp: CS2205.CH2023.01
- Tự đánh giá (điểm tổng kết môn): 8.5/10
- Số buổi vắng: 1
- Số câu hỏi QT cá nhân: 5
- Link Github: https://github.com/mynameuit/CS2205.xxx/

ĐỀ CƯƠNG NGHIÊN CỨU

TÊN ĐỀ TÀI (IN HOA)

TỰ ĐỘNG HOÁ TẠO HÀM PHẦN THƯỞNG BẰNG MÔ HÌNH NGÔN NGỮ LỚN TRONG HỌC TĂNG CƯỜNG

TÊN ĐỀ TÀI TIẾNG ANH (IN HOA)

AUTOMATED GENERATE REWARD FUNCTION BY LLMS IN REINFORCEMENT LEARNING

TÓM TẮT *(Tối đa 400 từ)*

Reinforcement Learning (RL) đã chứng minh thành công đáng kể trong việc điều khiển tự động các nhiệm vụ robot phức tạp, tuy nhiên việc thiết kế hàm Reward(Rt) hiệu quả vẫn là một thách thức lớn. Hầu hết các phương pháp truyền thống đều là tạo các hàm reward bằng thủ công với hiệu quả vẫn chưa đạt được tối ưu cao, đòi hỏi nhiều nỗ lực thử nghiệm dẫn đến tốn kém chi phí trong huấn luyện. Các mô hình ngôn ngữ lớn (LLMs) đang phát triển rất mạnh mẽ đặc biệt là các mô hình suy luận (Reasoning LLMs) đang nổi lên liên tục với khả năng lập trình (coding) nhờ đó thấy được tiềm năng trong việc tự động hoá quá trình tạo hàm reward. Phương pháp của chúng tôi bao gồm ba phần chính: (1) Đầu vào ngữ cảnh sử dụng thông tin môi trường; (2) Các ví dụ về hàm reward, tìm kiếm tiến hoá lặp đi lặp lại để tinh chỉnh các ứng viên hàm reward; (3) Một bước phản chiếu kết hợp phản hồi từ hình ảnh và kết quả để nâng cao hiệu quả. Nghiên cứu này phát triển dựa trên công trình EUREKA[1] bằng cách giới thiệu phương pháp few-shot generation[2] kết hợp với chain-of-thought prompting[3], nhằm tăng cường độ chính xác của quá trình tạo hàm reward tự động. Cuối cùng bằng cách phân tích và so sánh kết quả giữa hàm reward được tạo tự động bởi LLMs và được tạo thủ công bởi con người để chứng minh hiệu suất được cải thiện và đồng thời cũng giảm bớt được chi phí huấn luyện.

GIỚI THIỆU *(Tối đa 1 trang A4)*

Học tăng cường - Reinforcement Learning (RT) là một lĩnh vực đang phát triển mạnh mẽ trong lĩnh vực AI. Tuy nhiên một trong những thách thức lớn nhất của RL là thiết kế hàm phần thưởng (hàm reward) hiệu quả - thành phần then chốt quyết định hành vi của agent và chất lượng của giải pháp cuối cùng. Đề tài này xuất phát từ nhận thức về khó khăn trong quá trình phát triển hàm reward truyền thống, vốn đòi hỏi kiến thức chuyên môn cao, tốn nhiều thời gian và phải trải qua nhiều lần thử nghiệm[4].

Điều thúc đẩy tôi chọn đề tài này bắt nguồn từ hai xu hướng công nghệ đang phát triển mạnh mẽ gần đây. Một mặt, RL đang được ứng dụng ngày càng rộng rãi trong nhiều lĩnh vực từ robot đến tối ưu hoá hệ thống phức tạp. Mặt khác, các mô hình ngôn ngữ lớn đạt được những bước nhảy vọt trong khả năng lập trình [5] - đặc biệt là các mô hình suy luận chuyên sâu như GPT-o1, Claude 3, DeepSeek-R1[6], Grok 3… Công trình của EUREKA gần đây đã chứng minh rằng LLMs có thể tạo ra các hàm reward hiệu quả trong một số nhiệm vụ đơn giản, tuy nhiên vẫn còn nhiều hạn chế cần được giải quyết, đặc biệt là các tác vụ phức tạp hơn.

Nghiên cứu này nhằm tận dụng khả năng mã hoá và suy luận của LLMs để tự động quá trình thiết kế hàm reward, từ đó: (1) giảm rào cản kỹ thuật cho các nhà nghiên cứu không chuyên về RL; (2) Tăng tốc quá trình phát triển ứng dụng RL; (3) Khám phá các tiếp cận mới để tạo ra các hàm reward hiệu quả và sáng tạo.

Đầu vào của đề tài:

1. Mô tả ngôn ngữ tự nhiên về nhiệm vụ RL:
    - Mô tả chi tiết môi trường (environment) bao gồm không gian trạng thái (state space) và không gian hành động (action space).
    - - Mục tiêu cần đạt được của agent (ví dụ: "Robot cần học cách di chuyển nhanh nhất có thể mà không bị ngã").
    - - Các ràng buộc hoặc điều kiện đặc biệt (ví dụ: "Robot không được tiêu thụ quá nhiều năng lượng").
2. Thông tin kỹ thuật về môi trường RL:
    - Cấu trúc dữ liệu của trạng thái (state): kích thước, phạm vi giá trị, ý nghĩa của

từng thành phần.

- Cấu trúc dữ liệu của hành động (action): kích thước, phạm vi giá trị, ý nghĩa của từng thành phần.

3. Ví dụ mẫu (Cho phương pháp few-shot):

- Các ví dụ về hàm reward đã được thiết kế cho các nhiệm vụ tương tự .
- Các đặc điểm và hiệu suất của các ví dụ này để làm cơ sở tham chiếu.

4. Phản hồi từ quá trình huấn luyện (cho phương pháp lặp):

- Dữ liệu hiệu suất từ các lần huấn luyện trước đó.
- Phân tích về điểm mạnh và điểm yếu của các hàm reward đã thử.

Đầu ra của đề tài:

- Mã nguồn đầy đủ của hàm reward có thể tích hợp trực tiếp vào framework RL.
- Hàm nhận vào state, action, next_state và trả về giá trị reward dưới dạng số thực .
- Bao gồm các thành phần reward khác nhau (như reward cho tiến triển, hình phạt cho hành vi không mong muốn).

MỤC TIÊU *(Viết trong vòng 3 mục tiêu)*

- Phát triển và kiểm nghiệm phương pháp tự động tạo hàm reward bằng mô hình ngôn ngữ lớn (LLMs) cho các bài toán học tăng cường.
- So sánh và đánh giá hiệu suất của hàm Reward(Rt) được tạo tự động bằng các huấn luyện agent với các môi trường benchmark cụ thể và đo lường khoảng cách hiệu năng giữa hàm do LLMs và con người thiết kế.
- Phân tích đánh giá các yếu tố ảnh hưởng đến chất lượng của hàm Reward được tạo tự động bao gồm độ phức tạp của môi trường, chất lượng mô tả đầu vào, và khả năng của các LLMs khác nhau từ đó đưa ra nhận xét về tiềm năng cũng như giới hạn của phương pháp này.

NỘI DUNG VÀ PHƯƠNG PHÁP

1. Chuẩn bị môi trường thử nghiệm

- Cài đặt và cấu hình các môi trường RL benchmark phổ biến bao gồm:

+ Gymnasium[7] (trước đây là OpenAI Gym) cho các môi trường cơ bản.
+ MuJoCo[8] cho các nhiệm vụ điều khiển robot và động lực học vật lý.
+ Isaac Gym[9] để mô phỏng các tác vụ robot phức tạp với khả năng song song hóa GPU.

- Lựa chọn các nhiệm vụ đa dạng với độ phức tạp tăng dần:
  + Các nhiệm vụ điều khiển cổ điển (CartPole, Pendulum)
  + Các nhiệm vụ vận động của robot (Ant, Humanoid)
  + Các nhiệm vụ thao tác đối tượng (robot gắp vật, sắp xếp khối)

2. Phát triển mã tạo hàm reward tự động bằng LLMs

2.1. Lựa chọn và tích hợp mô hình LLMs

- Triển khai các LLM chuyên về khả năng lập trình:
  - Chọn các mô hình suy luận có khả năng lập trình tốt như GPT-o1 của OpenAI, DeepSeek-R1[], Claude 3.7 Sonnet của Anthropic, Grok 3 của xAI.
  - Cài đặt để gọi OpenAI API, Anthropic API, DeepSeek-R1 API hay Grok API thông qua thư viện langchain hoặc hàm tự viết.
  - Xây dựng các hàm để xử lý kết quả và đánh giá kết quả từ LLMs.

2.2. Thiết kế và tối ưu hoá prompt

- Phát triển các template prompt theo các chiến lược khác nhau:
  - Zero-shot prompting: Chỉ sử dụng mô tả nhiệm vụ.
  - Few-shot prompting: Bổ sung các ví dụ về các hàm reward trong các nhiệm vụ tương tự.
  - Chain-of-thought prompting: Hướng dẫn LLM suy nghĩ từng bước về các xây dựng hàm reward.
- Triển khai kỹ thuật decomposition để chia nhỏ vấn đề:
  - Phân tích yêu cầu nhiệm vụ.
  - Xác định trạng thái và hành động có liên quan.
  - Xây dựng cấu trúc reward phù hợp.

2.3. Xây dựng quy trình lặp đi lặp lại (Iterative Evolutionary Search)

- Triển khai thuật toán tiến hoá để tối ưu hàm reward:
    - Tạo ra một quần thể các hàm reward khởi tạo từ LLM.
    - Đánh giá mỗi hàm reward bằng cách huấn luyện Agent.
    - Chọn lọc các hàm reward hiệu quả nhất
    - Tạo ra thế hệ mới bằng kết các hợp và đột biến các hàm reward tốt nhất.
- Thiết kế cơ chế phản hồi tự động:
    - Thu thập dữ liệu về hiệu suất agent và đặc điểm hàm reward.
    - Tạo ra mô tả phản hồi có cấu trúc.
    - Đưa mô tả vào prompt để LLM cải thiện hàm reward.

3. Huấn luyện và đánh giá các agent RL

3.1. Triển khai các thuật toán RL

- PPO (Proximal Policy Optimization) cho đào tạo chính sách ổn định.
- SAC (Soft Actor-Critic) cho các nhiệm vụ đòi hỏi khám phá.
- TD3 (Twin Delayed ĐPG) cho các nhiệm vụ điều khiển liên tục.
- Xây dựng wrapper cho từng thuật toán để tích hợp với các hàm reưad tự động tạo.

3.2. Thiết lập quy trình đánh giá

- Hiệu suất đạt được (cumulative reward, success rate).
- Tỷ lệ hội tụ (convergence rate).
- Ổn định trong quá trình huấn luyện(training stability).
- Khả năng tổng quát hoá (generalization capability).

4. Phân tích so sánh với hàm reward do con người thiết kế

4.1. Thiết lập thí nghiệm

- Cài đặt các hàm reward do con người thiết kế từ các nguồn uy tín.
- Đảm bảo điều kiện huấn luyện giống nhau cho hai loại hàm reward.
- Thiết kế thí nghiệm kiểm soát đối với các biến thể của môi trường.

4.2. Phân tích hiệu suất chi tiết

- So sánh trực tiếp các chỉ số hiệu suất:

- Thời gian hội tự đến ngưỡng hiệu suất.
- Điểm số cuối cùng đạt được.
- Độ ổn định của chính sách học được.

- Phân tích cấu trúc và đặc tính của các hàm reward:
  - Độ phức tạp của hàm (số lượng thành phần, các phép toán sử dụng).
  - Khả năng giải thích được (interpretability)
  - Tính mạnh mẽ đối với nhiễu và các biến thể môi trường.

4.3. Phân tích hiệu suất chi tiết

- Xác định các trường hợp mà hàm reward tự động tạo thất bại.
- Phân tích các nguyên nhân của lỗi và hạn chế.
- Đề xuất cải tiến dựa trên kết quả phân tích.

KẾT QUẢ MONG ĐỢI

- Chứng minh được hiệu quả của việc tự động tạo hàm reward bằng các mô hình ngôn ngữ lớn có khả năng suy luận.
- So sánh được hiệu suất của việc thiết kế và tối ưu hoá prompt từ các phương pháp khác nhau.
- Cung cấp tới cộng đồng một bộ mã Python phục vụ cho mục đích nghiên cứu và phát triển đề tài và tài liệu hướng dẫn chi tiết.

TÀI LIỆU THAM KHẢO *(Định dạng DBLP)*

[1]. Ma, Y. J., Liang, W., Wang, G., Huang, D. A., Bastani, O., Jayaraman, D., ... & Anandkumar, A. (2023). Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*.

[2]. Tang, X., Shin, R., Inan, H. A., Manoel, A., Mireshghallah, F., Lin, Z., ... & Sim, R. (2023). Privacy-preserving in-context learning with differentially private few-shot generation. *arXiv preprint arXiv:2309.11765*.

[3]. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models.

Advances in neural information processing systems, *35*, 24824-24837.

[4]. Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238-1274.

[5]. Jiang, J., Wang, F., Shen, J., Kim, S., & Kim, S. (2024). A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

[6]. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., ... & He, Y. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

[7]. Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., ... & Younis, O. G. (2024). Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*.

[8]. Todorov, E., Erez, T., & Tassa, Y. (2012, October). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5026-5033). IEEE.

[9]. Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., ... & State, G. (2021). Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*.