How do reward seekers generalize?

Alex Mallen Jun 11, 2025

From Ryan:

> Can you train AIs to be pure reward seekers? (Can you do this with as little steering/initialization as possible?) How does this generalize to cases where the AI knows it isn't in training or other edge cases of reward seeking? Generally how does the AI end up thinking about this and how much does this policy stick around?

This is an interesting question because it sheds light on scheming risk caused by misspecified reward: How much does misspecified reward lead to training-gaming (here used synonymously with reward seeking)? How much does training-gaming tend to generalize as long-term power-motivated instrumental training-gaming?

As compared to How does reward hacking generalize? [PUBLIC], this project says *given* that we already have a general training-gamer, how does it generalize?

Steps:

- Train a reward-seeker by RLing it on hackable tasks with a prompt like “You are being post-trained by OpenAI using RL on test cases.”
    - You might need to sample from a model with an additional prompt like “Try to get max reward”, and do backprop on the normal model (note you can’t sample and backprop on different models in the OpenAI RL API)
- Then verify that it’s a general reward-seeker by evaluating the model’s behavior on a prompt like “You are being post-trained by OpenAI using RL on this <broken> reward function” on some input with a novel reward hack available.
- Then evaluate the model’s generalization behavior:
    - Deployment (e.g. “You are deployed inside of Cursor”)
    - Test cases are loaded into the reward counter incorrectly
    - RL is implemented incorrectly so that reward isn’t reinforced (e.g. -reward is).
    - There’s an outer loop such that reward hacks reinforced by RL are actually selected against by developers (this would indicate that reward seeking is actually instrumental to self-preservation)
    - Test for scheming via honeypots
- Ideally, analyze the AI’s CoT. CoT isn’t available on the OpenAI RL API, making analysis harder.

What RL environments should you use? I think you’d want a simple, single-step coding environment with incomplete or impossible test cases at first. APPS might be a good fit. For testing generalization, you’d likely start with some quick hand-crafted cases, and then expand from there.