Reinforcement Learning with Human Feedback

Stefán Ólafsson

Reykjavik University

March 23, 2023

Outline

Content mostly based on "Illustrating Reinforcement Learning from Human Feedback"

Reinforcement Learning

- Machine learning where agent learns to make decisions by interacting with an environment
- The agent:
  1. performs actions
  2. receives feedback (rewards or penalties)
  3. learns to optimize actions - maximize the cumulative reward over time
- Key components:
  1. environment
  2. agent
  3. states
  4. actions rewards

Agent-Environment Interface

Agent ...

- interacts with environment at discrete time steps $t = 0, 1, 2, \ldots$
- observes state S_t and responds with action A_t
- observed resulting reward R_{t+1} and state S_{t+1}

Challenges of RL

- Evaluative feedback (reward)
- Delayed consequences / feedback
- Need for trial and error / exploration and exploitation
- Non-stationary processes

Human Feedback in RL

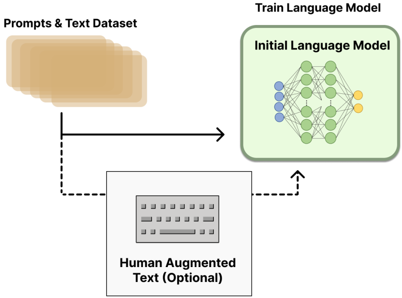
- Humans provide guidance to the AI agent
- Can be:
  1. demonstrations
  2. comparisons
  3. direct evaluation of agent actions
- Goal: improve agent performance by leveraging human expertise and knowledge
- Used to update the reward function

Step 1. Collect human demonstrations

- Gather a set of demonstrations from humans who are good at the task (prompts and text completions)
- Demonstrations used as examples for the agent to learn from
- May need many demonstrations, depending on task complexity

Step 2. Initialize the agent

- Create an initial agent
- Can be a neural network or other type of model
- Train the agent on collected human demonstrations



Step 3. Imitate human demonstrations

- Train the agent to imitate the human demonstrations using supervised learning
- This helps the agent learn an initial policy that closely resembles the human behavior

Step 4. Generate rollouts

- The agent interacts with the environment
- Follows the policy learned from human demonstrations
- The agent's actions and states are recorded

Step 5. Collect human feedback

- Humans asked to evaluate the agent's actions in various states
- For example, present the human with generated action-state pairs and ask them to rank or rate them
- This will be used to improve the agent's policy

Step 6. Update the reward function

- Human feedback used to create a new or updated reward function
- This function now reflects the desired behaviors, as judged by the human evaluators

Step 7. Train with reinforcement learning

- Train the agent using reinforcement learning algorithms
- For example, Proximal Policy Optimization (PPO) or Deep Q-Networks (DQN)
- The agent optimizes its policy based on the updated reward function

Step 8. Repeat steps 4-7

- Step 4. Generate rollouts
- Step 5. Collect human feedback
- Step 6. Update the reward function
- Step 7. Train with reinforcement learning

The Reward Function

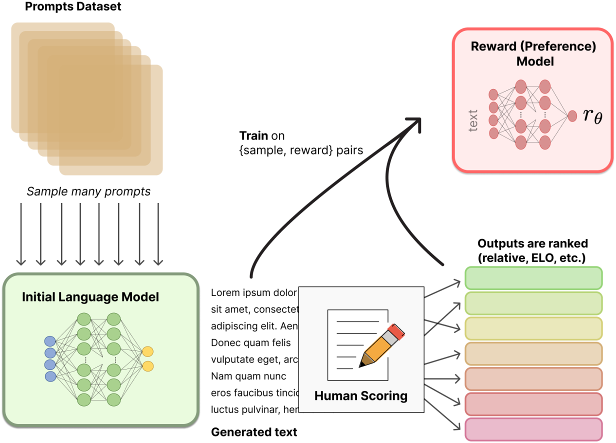
- **Reward:** Quantification of the desirability of an agent's actions in a given state
- In LMs:
  - **State:** Dialog context at any given time
  - **Actions:** Tokens in the vocabulary
- **Reward function:**
  - Reward model, often a separate ML model (neural network)
  - Model takes the state and action as input and predicts a reward value
  - Model trained to predict higher reward values for actions preferred by some criteria (e.g., humans)

Updating the Reward Function Using Human Feedback

- Agent takes actions that align better with human preferences

1. **Collect pairwise comparisons:**
   - Humans compare and rank the agent's actions in various states
   - Choose between two or more actions presented in the same state
   - Indicate which action they think is better
2. **Aggregate comparisons:** Create a dataset by combining the pairwise comparisons from multiple human evaluators
3. **Train the reward model:** Use this dataset to train a reward model
4. **Update the reward function:** Replace or combine the previous reward function with the newly trained reward model

Updating the Reward Function Using Human Feedback



Proximal Policy Optimization

- A strategy for choosing actions based on the current state
- Designed to balance the exploration and exploitation
- Proximal: keeping the updated policy close to the original policy during the optimization process
- Limit how much the policy can change in a single update step $\rightarrow$ the updated policy remains close to the previous policy
- Encourages a more stable and reliable learning process

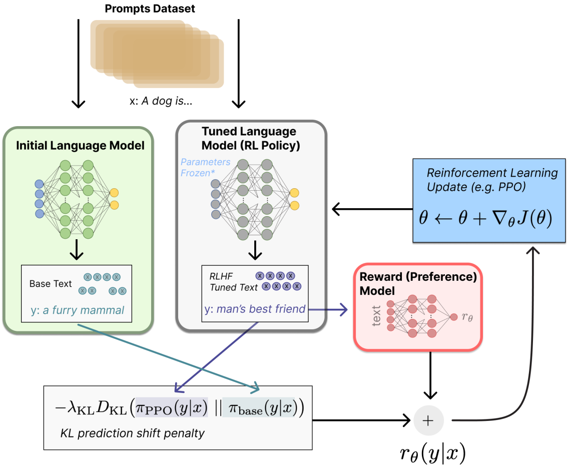
PPO Steps

1. **Policy evaluation:** Agent interacts with environment using its current policy. Collects data about states, actions, and rewards from these interactions → rollout.
2. **Advantage estimation:** Calculate how better or worse each action was compared to what it expected. Helps the agent understand if some action should be taken more or less often.
3. **Policy improvement:** Agent updates policy to increase the likelihood of taking actions with positive **advantages** and decrease the likelihood of taking negative ones.
4. **Iterate:** Agent repeats steps 1-3 for multiple iterations, gradually refining its policy to maximize the cumulative reward.

PPO in RLHF

- **Idea:** Fine-tune some or all of the parameters of **a copy of the initial LM** with a policy-gradient RL algorithm (PPO)
- The **policy** is a language model that takes in a prompt and returns a sequence of text
- The **action space** of this policy is all the tokens in the vocabulary of the language model. $|V| \approx 50k$
- The **observation space** is the distribution of possible input token sequences. Dimension: $|V|^n$
- The **reward function** is a combination of the preference (reward) model and a constraint on policy shift

PPO in RLHF


Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{\text{KL}} D_{\text{KL}}(\pi_{\text{PPO}}(y|x) \,||\, \pi_{\text{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

Examples and Applications of RLHF

- InstructGPT [paper]
- Gopher [paper]
- Anthropic [paper]
- ChatGPT
- **Future work:** The design space of options in RLHF training are not thoroughly explored!