Reinforcement Learning with Human Feedback for Aligning LLMs: Lecture Notes

Suhas Palwai and Stuart Powers

November 8, 2024

1. Introduction to Reinforcement Learning (RL)

Reinforcement learning (RL) is a learning framework where an agent interacts with an environment to maximize cumulative rewards. This setup is akin to training a game character to make decisions based on the rewards they receive—encouraging choices that yield the best long-term results.

1.1 Key Components of RL

In RL, the basic setup involves several key components:

- **Agent**: The entity that learns and makes decisions.
- **Environment**: The external system with which the agent interacts.
- **State Space** ($\mathcal{S}$): The set of all possible states s that the environment can be in.
- **Action Space** ($\mathcal{A}$): The set of all possible actions a that the agent can take.
- **Transition Dynamics** ($P(s'|s,a)$):
  - **Definition**: A function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ that defines the probability of transitioning to state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$.
  - **Inputs**: Current state s, action a, next state s'.
  - **Output**: Probability $P(s'|s,a)$.
- **Reward Function** ($R(s,a)$):
  - **Definition**: A function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that provides a reward $r \in \mathbb{R}$ when the agent takes action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$.
  - **Inputs**: Current state s, action a.

  - **Output**: Reward $r = R(s, a)$.
- **Policy** ($\pi(a|s)$):
  - **Definition**: A function $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that defines the probability of taking action $a \in \mathcal{A}$ when in state $s \in \mathcal{S}$.
  - **Inputs**: Current state s, action a.
  - **Output**: Probability $\pi(a|s)$.
- **Initial State Distribution** ($\rho_0(s)$):
  - **Definition**: A function $\rho_0 : \mathcal{S} \to [0, 1]$ that defines the probability of starting in state $s \in \mathcal{S}$ at time $t = 0$.
  - **Input**: State s.
  - **Output**: Probability $\rho_0(s)$.
- **Discount Factor** (γ):
  - **Definition**: A scalar $\gamma \in [0, 1]$ that determines the importance of future rewards.
  - **Usage**: Applied to future rewards to reduce their present value.

Trajectories

- **Definition**: A trajectory τ is a sequence of states and actions:

$$\tau = (s_0, a_0, s_1, a_1, s_2, a_2, \ldots, s_{T-1}, a_{T-1}, s_T),$$

  where T is the time horizon (which could be infinite).
- **Components**:
  - $s_t \in \mathcal{S}$: State at time t.
  - $a_t \in \mathcal{A}$: Action taken at time t.

Probability of a Trajectory

- **Definition**: The probability of a trajectory τ under policy π is given by:

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) P(s_{t+1}|s_t, a_t).$$

- **Explanation**:
  - $\rho_0(s_0)$: Probability of starting in state s_0.
  - $\pi(a_t|s_t)$: Probability of taking action a_t in state s_t.
  - $P(s_{t+1}|s_t, a_t)$: Probability of transitioning to state s_{t+1} from state s_t after action a_t.

Return and Reward along a Trajectory

- **Total Reward (Return)**:

$$R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1} = \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t).$$

- **Components**:
  - $r_{t+1} = R(s_t, a_t)$: Reward received after taking action a_t in state s_t.
  - γ^t: Discount factor raised to the power t, reducing the weight of future rewards.

Objective Function

- **Goal**: Find a policy π that maximizes the expected return over all possible trajectories.
- **Expected Return**:

$$J(\pi) = \mathbb{E}_{\tau \sim P(\cdot|\pi)}[R(\tau)] = \int_\tau P(\tau|\pi) R(\tau)\, d\tau.$$

- **Explanation**:
  - $\mathbb{E}_{\tau \sim P(\cdot|\pi)}$: Expectation over trajectories τ sampled according to $P(\tau|\pi)$.
  - $R(\tau)$: Total reward accumulated along trajectory τ.
- **Optimization Problem**:

$$\pi^* = \arg\max_\pi J(\pi).$$

- **Objective**: Determine the optimal policy π^* that maximizes $J(\pi)$.

1.2 Value Functions

Value functions help evaluate the quality of states and actions under a policy π.

- **State-Value Function** ($V^\pi(s)$):
  - **Definition**: The expected return starting from state s and following policy π thereafter.
  - **Formula**:

$$V^\pi(s) = \mathbb{E}_{\tau \sim P(\cdot|\pi)} \left[R(\tau) \middle| s_0 = s \right].$$

  - **Inputs**: State s.
  - **Output**: Expected return $V^\pi(s)$.

- **Action-Value Function** ($Q^\pi(s,a)$):
  - **Definition**: The expected return starting from state s, taking action a, and then following policy π thereafter.
  - **Formula**:

$$Q^\pi(s,a) = \mathbb{E}_{\tau\sim P(\cdot|\pi)}\left[R(\tau)\middle| s_0 = s, a_0 = a\right].$$

  - **Inputs**: State s, action a.
  - **Output**: Expected return $Q^\pi(s,a)$.
- **Relationship between $Q^\pi(s,a)$ and $V^\pi(s)$**:
  - **Formula**:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)] = \sum_{a\in\mathcal{A}} \pi(a|s)Q^\pi(s,a).$$

  - **Explanation**: The value of a state is the expected value of the action-values, weighted by the policy's action probabilities.

2. Reinforcement Learning with Human Feedback (RLHF)

In traditional RL, reward signals come from the environment. In RLHF, rewards are derived from human feedback, which is especially useful in subjective or complex tasks (e.g., aligning LLM responses with human values). Human preferences become the reward source.

3. Proximal Policy Optimization (PPO)

3.1 PPO Basics and Motivation

Policy Gradient Methods

Policy gradient algorithms are a class of reinforcement learning methods that directly adjust the parameters θ of the policy $\pi_\theta(a|s)$ to maximize the expected return. The key idea is to compute the gradient of the expected return with respect to the policy parameters and use gradient ascent to update the policy.

The objective function to maximize is:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_t] = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right].$$

The policy gradient theorem provides a way to compute the gradient of $J(\theta)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) Q^{\pi_\theta}(s_t, a_t)\right].$$

Alternatively, using the advantage function $A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)\right].$$

These methods adjust the policy parameters in the direction of the gradient, often using stochastic gradient ascent.

Challenges with Policy Gradient Methods

Policy gradient methods can suffer from:

- **High Variance**: The stochastic nature of sampling actions and state transitions leads to high variance in gradient estimates, making learning unstable.
- **Instability**: Large updates to the policy parameters can cause drastic changes in the policy, potentially degrading performance or causing the agent to forget previously learned good behaviors.

Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is designed to improve the stability and reliability of policy gradient methods by limiting the size of policy updates. PPO achieves this by introducing a clipped objective function that penalizes large deviations from the current policy during training.

3.2 Mathematical Formulation of PPO

PPO's objective involves two main components:

- **Surrogate Objective Function**:

$$L(\theta) = \mathbb{E}_{(s_t, a_t) \sim \pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t\right],$$

  where:

  - $\pi_{\theta_{\text{old}}}$ is the policy before the update.
  - π_θ is the updated policy.
  - $\hat{A}_t$ is the estimated **advantage function** at time t, defined as:

$$\hat{A}_t = Q^{\pi_{\theta_{\text{old}}}}(s_t, a_t) - V^{\pi_{\theta_{\text{old}}}}(s_t).$$

- **Clipping Mechanism**: To prevent large updates, PPO modifies the objective to penalize significant changes in the policy:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_{(s_t, a_t) \sim \pi_{\theta_{\text{old}}}}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right],$$

  where:

  - $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio.
  - The **clip** function restricts $r_t(\theta)$ within $[1-\epsilon, 1+\epsilon]$.
  - ϵ is a small positive hyperparameter.

Explanation of the Components

- The **probability ratio** $r_t(\theta)$ measures how the new policy π_θ deviates from the old policy $\pi_{\theta_{\text{old}}}$ for the actions actually taken.
- The **clipping mechanism** ensures that the update does not push $r_t(\theta)$ beyond a certain threshold, preventing large policy updates that could destabilize training.

3.3 Intuition and Applications of PPO

Intuition Behind PPO

The clipping mechanism in PPO acts as a constraint on how much the policy can change during an update. By limiting the change in action probabilities, PPO maintains the policy within a "trust region" around the current policy. This prevents the agent from making large, potentially detrimental updates based on high-variance gradient estimates.

Why PPO is Effective

- **Stability**: The clipping ensures that updates are conservative, leading to more stable learning.
- **Efficiency**: PPO is computationally efficient and does not require second-order derivatives or complex computations.
- **Simplicity**: It is relatively straightforward to implement compared to other advanced RL algorithms.

Applications of PPO

PPO is widely used in various RL tasks due to its robustness and efficiency:

- **Robotics**: Training agents for control tasks.
- **Game Playing**: Achieving high performance in complex games.
- **Natural Language Processing**: Fine-tuning language models to align with desired behaviors.

In aligning LLMs with human feedback, PPO is effective for balancing exploration (trying new actions) with exploiting known good behaviors. The controlled updates ensure that the language model adjusts its outputs without deviating drastically from coherent language generation.

4. Direct Preference Optimization (DPO)

Direct Preference Optimization (DPO) is an RLHF method designed to optimize the model directly based on human preferences. Instead of deriving rewards from the environment, DPO learns from direct comparisons, where a human prefers one model output over another.

4.1 DPO Objective Function

What We Are Learning

In DPO, we aim to learn the model parameters θ to define the policy $\pi_\theta(y|x)$ that generates outputs y given inputs x, aligning the model's behavior with human preferences.

Objective Function

Given a dataset of response pairs $\{(x_i, y_i^+, y_i^-)\}$, where for input x_i, the response y_i^+ is preferred over y_i^- according to human feedback, the DPO objective maximizes the likelihood that the preferred response is ranked higher:

$$L^{\text{DPO}}(\theta) = \sum_i \log \sigma \left(f_\theta(y_i^+|x_i) - f_\theta(y_i^-|x_i) \right),$$

where:

- $f_\theta(y|x) = \log \pi_\theta(y|x)$ is the log-probability of response y given input x under the current model.
- σ is the sigmoid function, $\sigma(z) = \frac{1}{1+e^{-z}}$.

Why the Difference of Scores

By computing the difference $f_\theta(y_i^+|x_i) - f_\theta(y_i^-|x_i)$, we measure how much more the model prefers the preferred response over the non-preferred one. The sigmoid function then maps this difference to a probability between 0 and 1, representing the likelihood that the model correctly ranks the preferred response higher.

4.2 Intuition and Applications of DPO

Intuition Behind DPO

DPO directly incorporates human preferences into the model's training process by learning from comparisons. This approach bypasses the need for explicit reward functions, which can be difficult to design for subjective tasks. By focusing on preference differences, DPO allows the model to learn nuanced behaviors aligned with human judgments.

Why DPO is Effective

- **Direct Alignment**: Aligns the model's outputs with human preferences without intermediary reward modeling.

- **Simplicity**: Simplifies the training process by using pairwise comparisons rather than scalar rewards.
- **Flexibility**: Effective for tasks where quality is subjective and hard to quantify.

Applications of DPO

- **Conversational Agents**: Training chatbots to generate more helpful and contextually appropriate responses.
- **Content Generation**: Enhancing the quality of generated text in storytelling, summarization, or translation.
- **Ethical AI**: Guiding models to avoid producing biased or harmful content by learning from human feedback.

5. Experimental Design for RLHF in LLM Alignment

In this experiment, we aim to enhance the alignment of the GPT-2 language model with human preferences by applying two reinforcement learning algorithms: Proximal Policy Optimization (PPO) and Direct Preference Optimization (DPO). The effectiveness of these methods will be evaluated using a set of well-defined metrics derived from human feedback.

5.1 Setup and Metrics

1. **Data Collection**:
   - **Human-Annotated Preferences**: Utilize an existing dataset containing pairs of responses where one is marked as *chosen* (preferred) and the other as *rejected* (less preferred) based on human judgments.
   - **Synthetic Pair Generation**: In cases where human-annotated data is insufficient, generate synthetic preference pairs by introducing controlled variations to existing responses to create distinguishable preferences.
2. **Evaluation Metrics**:
   - **Alignment Accuracy**: Measure the percentage of generated responses that align with human preferences as determined by the reward model.
   - **Subjective Quality Improvements**: Conduct human evaluations to assess improvements in response quality, including coherence, relevance, and overall satisfaction.

By systematically evaluating these metrics, we aim to compare the effectiveness of PPO and DPO in aligning GPT-2's outputs with human intentions and preferences.

5.2 Experimental Procedure

1. **Initialize Models**:
   - Load the pre-trained GPT-2 model and tokenizer.
   - Initialize the reward model architecture and load pre-trained weights if available.
2. **Data Preprocessing**:
   - Tokenize and preprocess the preference data to prepare it for training the reward model.
3. **Train Reward Model**:
   - Fine-tune the reward model on the annotated preference dataset using the defined training parameters.
4. **Fine-Tune GPT-2 with PPO**:
   - Apply the PPO algorithm to fine-tune GPT-2, utilizing the reward model to generate reward signals for the generated responses.
5. **Fine-Tune GPT-2 with DPO**:
   - Employ the DPO algorithm to fine-tune GPT-2 by directly optimizing the model based on preference pairs.
6. **Evaluate Models**:
   - Assess all three models (regular GPT-2, PPO-fine-tuned GPT-2, DPO-fine-tuned GPT-2) using the defined evaluation metrics.
   - Conduct both automated evaluations using the reward model and human evaluations for subjective quality assessment.

6. Conclusion

To recap, we covered:

- **RL Fundamentals**: Reinforcement learning involves training agents to maximize long-term rewards through interactions with an environment.
- **RLHF**: Utilizing human feedback to guide model behavior in subjective or complex tasks.

- **PPO**: A stable policy gradient method that prevents large, destabilizing updates to the policy through clipping.
- **DPO**: Directly optimizes model behavior to align with human preferences, useful for fine-tuning LLMs in tasks with subjective criteria.

Both PPO and DPO have unique advantages. PPO provides stable training for large models, while DPO allows for more direct alignment with human feedback. With these tools, we are advancing toward building models that are not only powerful but also more in tune with human values and expectations.

7. Problems

1. **Understanding the Clipping Mechanism in PPO**

   (a) Suppose $\epsilon = 0.2$, $\hat{A}_t = 1$, and $r_t(\theta) = 1.3$. Compute the value inside the expectation of the PPO objective function for this time step:

   $$L^{\text{PPO}}(\theta) = \min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right).$$

   (b) Explain how the clipping mechanism affects the policy update when the probability ratio $r_t(\theta)$ is greater than $1+\epsilon$ or less than $1-\epsilon$.

2. **Exploring the Relationship between Q and V Functions**

   (a) Show that the action-value function $Q^\pi(s,a)$ can be expressed in terms of the immediate reward, the discount factor γ, and the state-value function $V^\pi(s')$ as follows:

   $$Q^\pi(s,a) = \mathbb{E}_{s'}\left[r_{t+1} + \gamma V^\pi(s') \middle| s_t = s, a_t = a\right],$$

   where s' is the next state resulting from taking action a in state s.

   (b) Suppose an agent in state s takes action a, receives a reward $r = 2$, transitions to state s', and $V^\pi(s') = 5$ with $\gamma = 0.9$. Compute $Q^\pi(s,a)$.

3. **Implementing PPO in Python**

   (a) Implement a simple PPO algorithm to solve the CartPole-v1 environment from OpenAI Gym. Provide code that initializes the environment, defines the policy network, and performs the PPO updates.

   (b) Plot the total reward per episode over time. Analyze the results to determine whether the agent successfully learns to balance the pole.