# Reinforcement Learning with Human Feedback for Aligning Large Language Models: 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 $s \in \mathcal{S}$ that the environment can be in.
- **Action Space** ($\mathcal{A}$): The set of all possible actions $a \in \mathcal{A}$ 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** ($R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$):
  - **Definition**: A function that provides a reward $r \in \mathbb{R}$ when the agent takes action $a$ in state $s$.

  - **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 $a$ when in state $s$.
  - **Inputs**: 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 $s$ at time $t = 0$.
  - **Input**: State $s$.
  - **Output**: Probability $\rho_0(s)$.

- **Discount Factor** ($\gamma$):
  - **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 $\tau$ 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 $\tau$ under policy $\pi$ 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 $s_t$.
  - $\gamma^t$: Discount factor raised to the power $t$, reducing the weight of future rewards.

**Objective Function**

- **Goal**: Find a policy $\pi$ 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 $\tau$ sampled according to $P(\tau|\pi)$.
  - $R(\tau)$: Total reward accumulated along trajectory $\tau$.
- **Optimization Problem**:

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

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

## 1.2 Value Functions

Value functions help evaluate the quality of states and actions under a policy $\pi$.

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

**Derivation of the Policy Gradient Theorem**

The objective function to maximize is:

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim P(\cdot|\pi_\theta)}[R(\tau)].$$

We can express $J(\pi_\theta)$ as:

$$J(\pi_\theta) = \int_\tau P(\tau|\pi_\theta) R(\tau)\, d\tau.$$

The gradient of $J(\pi_\theta)$ with respect to $\theta$ is:

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \int_\tau P(\tau|\pi_\theta) R(\tau)\, d\tau = \int_\tau \nabla_\theta P(\tau|\pi_\theta) R(\tau)\, d\tau.$$

Using the likelihood ratio trick, we write:

$$\nabla_\theta P(\tau|\pi_\theta) = P(\tau|\pi_\theta) \nabla_\theta \log P(\tau|\pi_\theta).$$

But since $P(\tau|\pi_\theta)$ depends on $\pi_\theta$, and transitions $P(s_{t+1}|s_t, a_t)$ are independent of $\theta$, we have:

$$\log P(\tau|\pi_\theta) = \log \rho_0(s_0) + \sum_{t=0}^{T-1} \left[\log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t, a_t)\right].$$

Only $\pi_\theta(a_t|s_t)$ depends on $\theta$, so:

$$\nabla_\theta \log P(\tau|\pi_\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t).$$

Therefore,

$$\nabla_\theta J(\pi_\theta) = \int_\tau P(\tau|\pi_\theta) \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right] R(\tau)\, d\tau.$$

This simplifies to:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim P(\cdot|\pi_\theta)} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)\right].$$

Since $R(\tau)$ is the total return from time $t = 0$, we can express the return from time $t$ onward as $G_t$:

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_{k+1}.$$

Substituting, we get:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim P(\cdot|\pi_\theta)} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right].$$

This is the policy gradient theorem.

**Challenges with Policy Gradient Methods**

Policy gradient methods can suffer from:

- **High Variance**: Sampling actions and state transitions leads to high variance in gradient estimates.
- **Instability**: Large updates to policy parameters can cause drastic changes in the policy.

**Proximal Policy Optimization (PPO)**

PPO aims to improve stability by limiting the size of policy updates through a clipped objective function.

## 3.2 Mathematical Formulation and Derivation of PPO

**Surrogate Objective Function**

The PPO objective function is designed to maximize a surrogate objective that approximates the true objective but is easier to optimize.

**Derivation**

The starting point is the ratio of the new policy to the old policy:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}.$$

We consider the following surrogate objective function:

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

where $\hat{A}_t$ is an estimator of the advantage function.

**Explanation**

- When $r_t(\theta)\hat{A}_t$ is within the clipping range, we use $r_t(\theta)\hat{A}_t$. - When $r_t(\theta)$ goes beyond $[1-\epsilon, 1+\epsilon]$, the objective is penalized by using the clipped value.

**Justification**

This approach prevents the policy from changing too much in a single update, ensuring that the new policy stays close to the old one, thus maintaining stability.

## 3.3 Intuition and Applications of PPO

**Intuition Behind PPO**

The clipping mechanism acts as a constraint, keeping the policy updates within a "trust region." This helps in preventing large deviations that could harm performance.

**Why PPO is Effective**

- Balances exploration and exploitation.
- Reduces variance and improves stability.
- Easy to implement and tune.

# 4. Direct Preference Optimization (DPO)

## 4.1 DPO Objective Function and Derivation

**Objective Function**

Given a dataset $\mathcal{D} = \{(x_i, y_i^+, y_i^-)\}$ where $y_i^+$ is preferred over $y_i^-$ for input $x_i$, the DPO objective aims to maximize the probability that the model assigns higher preference to $y_i^+$.

**Derivation**

The probability that the model prefers $y_i^+$ over $y_i^-$ is given by:

$$P_\theta(y_i^+ \succ y_i^- | x_i) = \frac{\exp(f_\theta(x_i, y_i^+))}{\exp(f_\theta(x_i, y_i^+)) + \exp(f_\theta(x_i, y_i^-))}.$$

This is similar to a softmax over two items. Taking the log-likelihood over the dataset:

$$L^{\text{DPO}}(\theta) = \sum_i \log P_\theta(y_i^+ \succ y_i^- | x_i) = \sum_i \log \left( \frac{\exp(f_\theta(x_i, y_i^+))}{\exp(f_\theta(x_i, y_i^+)) + \exp(f_\theta(x_i, y_i^-))} \right).$$

Simplifying, we get:

$$L^{\text{DPO}}(\theta) = \sum_i \left[ f_\theta(x_i, y_i^+) - \log \left( \exp(f_\theta(x_i, y_i^+)) + \exp(f_\theta(x_i, y_i^-)) \right) \right].$$

Alternatively, using the logistic sigmoid function:

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

where $\sigma(z) = \dfrac{1}{1 + e^{-z}}$.

**Explanation**

- The objective encourages the model to assign higher scores to preferred responses. - By maximizing this likelihood, we directly optimize the model to align with human preferences.

## 4.2 Intuition and Applications of DPO

**Intuition Behind DPO**

DPO leverages pairwise comparisons, which are often easier to obtain than absolute scores. It simplifies the training process by focusing on preference rankings.

**Applications**

- Fine-tuning language models to produce outputs that are more aligned with human expectations. - Situations where defining an explicit reward function is challenging.

# 5. 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.

# 6. 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 $Q^\pi(s,a)$ and $V^\pi(s)$**

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

   $$Q^\pi(s,a) = \mathbb{E}_{s'}\left[R(s,a) + \gamma V^\pi(s') \middle| s_t = s, a_t = a\right],$$

   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.