Reinforcement Learning from Human Feedback (RLHF): From Concept to Implementation

Here is a glossary of the key terms and symbols used in the PPO equations.

- State (st) The input to the model at a specific timestep, t. For a language model, this is the sequence of text generated so far.
- Action (at) The output from the model at timestep t. For a language model, this is the specific token it chooses to generate next.
- Policy ($\pi\phi$) The model itself, which is a function that takes a state (st) and outputs a probability distribution over possible actions. The symbol ϕ represents the model's trainable parameters (weights).
- Old Policy ($\pi\phi$old) A fixed, read-only copy of the policy from before a training update begins. It's used as a stable baseline for comparison.
- Reward (Rt) The scalar feedback signal received from the environment (or a reward model) after the model takes an action. A positive reward indicates a good outcome.
- Value Function (V(st)) The output of the critic or value model. It's a prediction of the total expected future reward from a given state, st. Its parameters are often denoted by ψ.
- Advantage Function (A^t) A measure of how much better a specific action was compared to the average action at a given state. It's often calculated as the actual reward received minus the value function's prediction ($Rt - V(st)$).
- Probability Ratio (rt(ϕ)) The ratio of the probability of an action under the new policy versus the old policy, $\pi\phi\text{old}(at \mid st)\pi\phi(at \mid st)$. It measures how much the policy has changed.
- Discount Factor (γ) A number between 0 and 1 that determines the importance of future rewards. A smaller gamma makes the agent prioritize immediate rewards more heavily.
- Clipping Parameter (ϵ) A small hyperparameter (e.g., 0.2) in the PPO objective that defines the range within which the probability ratio (rt) is "clipped." This ensures updates are small and stable.
- Entropy (S or H) A measure of the randomness or uncertainty in the policy's output distribution. A higher entropy encourages exploration.
- KL Divergence (DKL) The Kullback-Leibler Divergence. A measure of how different one probability distribution is from another, used to ensure the policy doesn't change too drastically.
- Loss Function (L) The function that is minimized during training to update the model's parameters.
- Expectation (Et) The statistical expectation or, in practice, the average value of a quantity calculated over a batch of data collected at different timesteps, t.

Reinforcement Learning from Human Feedback (RLHF): From Concept to Implementation

The Three-Phase RLHF Pipeline

At its core, RLHF is a process that fine-tunes a language model by first learning a model of human preferences and then using that model as a reward function to guide further training with reinforcement learning. The process is universally broken down into three distinct phases.

Phase 1: Supervised Fine-Tuning (SFT)

Conceptual Goal

The journey begins not with reinforcement learning, but with standard supervised learning. The primary goal of the SFT phase is to adapt a general-purpose, pre-trained base language model (e.g., Llama 3, Mistral) to the specific domain and style of the desired task. If the end goal is to create a helpful chatbot, this phase teaches the model the format of a dialogue. If the goal is summarization, it teaches the model how to produce a good summary.

This is achieved by training the base model on a high-quality, (expert as a labeller created), curated dataset of (prompt, ideal_response) pairs. This creates Reward Model(initial).

Advanced Mechanics & Mathematics

- **Data Curation:** The target SFT dataset, used to train for a specific desired output, or "policy" can have several different data types for each purpose, eg:
    - **Instruction Following:** General datasets (e.g., Alpaca) to teach the model how to respond to commands.
    - **Domain-Specific Examples:** High-quality (prompt, response) pairs for the target task (e.g., summarization, code generation).
    - **Safety & Ethics Data**

Parameter-Efficient Fine-Tuning (PEFT): Fully fine-tuning a 70B+ parameter model is computationally infeasible for most.

In practice, techniques like **Low-Rank Adaptation (LoRA)** are standard. LoRA freezes the original weights of the pre-trained model and injects small, trainable low-rank matrices into the transformer layers. The SFT loss is minimized only with respect to these LoRA parameters, dramatically reducing the memory and compute requirements.

Reinforcement Learning from Human Feedback (RLHF): From Concept to Implementation

Phase 2: Reward Model (RM) Training

Conceptual Goal

The goal of this phase is to explicitly train a model to replicate given (dataset of) human preferences.

Human labelers then rank responses (either from data-set or AI created) from best to worst. This creates a preference dataset of (prompt, chosen_response, rejected_response) tuples.

A new model, the Reward Model (RM), is then trained on this dataset. The RM learns to take a (prompt, response) pair and output a single scalar score that represents the degree of human preference. A higher score should indicate a better response. Reward outputs will then be put into a comparative loss function, with the aim being eg R1 / Preferred Response R.V > R2 / Rejected Response R.V. This RM will serve as an automated, scalable proxy for human feedback in the final phase.

Advanced Mechanics & Mathematics

- **RM Architecture:** The RM typically uses the SFT model as its base, but its final language modeling head is replaced with a **scalar regression head** (a single linear layer) that outputs a single number (the reward).

- **Reward Calibration & Normalization:** The absolute scale of the RM's output is arbitrary. To stabilize the PPO training that follows, the rewards for each batch are typically **whitened / normalized**—centered to have a mean of 0 and scaled to have a standard deviation of 1.
  This prevents exploding gradients caused by excessively large reward values.

- **The Bradley-Terry Loss Function:** The RM is trained to maximize the difference between the scores of chosen and rejected responses. It does this using a pairwise ranking (comparative) loss from the Bradley-Terry model, which models the probability that a response y_w is preferred over y_l.

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log(\sigma(r_\theta(x,y_w) - r_\theta(x,y_l)))\right]$$

  Here, r_θ is the scalar output from the reward model, and σ is the sigmoid function. This loss maximizes the log-odds that the chosen response scores higher than the rejected.

Reinforcement Learning from Human Feedback (RLHF): From Concept to Implementation

- **Critical Issue: Reward Hacking:** The PPO algorithm in the next phase is a powerful optimizer. It ensures accuracy by exploiting and maximising any flaw or spurious correlation in the RM. For example, if the RM inadvertently learns that longer responses are slightly better, the PPO policy will learn to generate verbose, unhelpful text to maximize its score. This is known as "reward hacking." Mitigating this issue requires high-quality, diverse preference data and careful monitoring.

Phase 3: PPO-based Fine-Tuning

Conceptual Goal (MSc Level)

This is the core reinforcement learning phase. The SFT model / pair creation model is now treated as a "policy" that needs to be improved. For a given prompt, the summary response policy generates a response. This response is then "judged" by the frozen Reward Model from Phase 2, which provides a reward score.

The PPO algorithm uses this reward signal to update the policy's weights, encouraging it to generate responses that the RM will score highly. To prevent the policy from deviating too far from the sensible language it learned during SFT, a penalty (a "KL penalty") is applied, which measures how much the active policy has diverged from its original SFT version. This balances optimizing for preference with maintaining coherent language generation.

Advanced Mechanics & Mathematics

- **The RL Setup:**
  - **Policy (π_ϕ):** The SFT model, whose weights (ϕ) are actively being trained. This is the "actor."
  - **Reference Policy (π_{ref}):** A frozen copy of the SFT model, used to calculate the KL penalty.
  - **Critic (V_ψ):** A value head added to the policy model that learns to predict the expected future reward. It helps reduce the variance of the policy updates.

- **Generalized Advantage Estimation (GAE**): To get a stable estimate of how much better a generated token is than the baseline, GAE is used. It provides a low-variance advantage estimate $\hat{A}_t$ by blending one-step TD error with a multi-step Monte Carlo estimate.

Reinforcement Learning from Human Feedback (RLHF): From Concept to Implementation

The Full PPO Objective Function:

The final objective is a composite function that is maximized with respect to the policy parameters ϕ and minimized with respect to the critic parameters ψ.

$$L(\phi, \psi) = \hat{\mathbb{E}}_t \left[L_t^{\text{CLIP}}(\phi) - c_1 L_t^{\text{VF}}(\psi) + c_2 S[\pi_\phi](s_t) \right]$$

- This single objective elegantly combines:
    1. **The Clipped Surrogate Policy Objective**
       This is the core of PPO. It updates the policy (the language model in this context, treated as an "actor") using an advantage estimate. The updates are "clipped" to prevent them from being too large, which helps maintain stability. This ensures the policy doesn't deviate too drastically from its previous version, preventing destabilizing changes and effectively creating a trust region.

    2. **The Value Head / Value Function Loss**
       This component trains a "critic" (a value head added to the policy model) to predict the expected reward value. This is a standard mean-squared error loss, and its purpose is to make the critic a more accurate predictor, which in turn helps reduce the variance of the policy updates.

    3. **The Entropy Bonus (S):** An entropy term is added to the policy's output distribution. This variable encourages exploration by rewarding the policy for being less certain in its choices. This prevents the model from prematurely settling on a single, potentially suboptimal, strategy and encourages it to explore a wider range of possible responses.

By optimizing this composite objective, the language model learns to generate responses that are highly ranked by the learned human preference model, while remaining stable, coherent, and exploratory.

Reinforcement Learning from Human Feedback (RLHF): From Concept to Implementation

A Worked Example of a Single PPO Update Step

Let's use a very simple "toy problem" to see how the numbers flow.

Setup

- **State (s_t):** You are at a crossroads.
- **Action (a_t):** You chose to turn **right**.
- **Old Policy's Probability (pi_phi_textold):** Your policy before this update gave a **60%** chance of turning right.
- **Critic's Prediction (V(s_t)):** Your critic predicted that being at this crossroads would yield a future score of **1.5**.
- **Hyperparameters:** Discount factor gamma=0.9, Clipping parameter epsilon=0.2.

Step 1: Collect Experience (Rollout)

You take the action (turn right) and the environment responds.

- **Reward Received (R_t):** You find a treasure chest! The immediate reward is **+5**.
- **Next State (s_t+1):** You have reached the end of the road (a terminal state). The value of any terminal state is **0**.

Step 2: Calculate the Advantage (hatA_t)

Now, we calculate how much better the action was than the critic expected.

- **Formula:** hatA∗t=R_t+gammaV(s∗t+1)−V(s_t)
- **Calculation:** hatA_t=5+(0.9times0)−1.5=mathbf3.5
- **Conclusion:** The advantage is **+3.5**. This positive number tells the model that turning right was a much better action than it predicted.

Step 3: Propose a Policy Update

The optimizer suggests a new set of weights (phi). We check the probability of taking the same action (turning right) with this new policy.

- **New Policy's Probability (pi_phi):** The updated policy now gives a **75%** chance of

turning right.

Step 4: Calculate the Clipped Surrogate Objective (LCLIP)

This is the core of PPO. We first find the probability ratio.

- **Probability Ratio (r_t):**
  r_t=fracpi_phi(a_t|s_t)pi_phi_textold(a_t|s_t)=frac0.750.60=mathbf1.25

Now, we calculate the two terms inside the `min` function:

1. **Unclipped Objective:** r_ttimeshatA_t=1.25times3.5=mathbf4.375
2. **Clipped Objective:**
   - First, clip the ratio:
     textclip(r_t,1−epsilon,1+epsilon)=textclip(1.25,0.8,1.2)=mathbf1.2
   - Then multiply by advantage: 1.2times3.5=mathbf4.2

Finally, PPO takes the **minimum** of these two values.

- **LCLIP Contribution:** min(4.375,4.2)=mathbf4.2

The final update is based on the value **4.2**, not 4.375. The clipping prevented an overly aggressive update, ensuring the policy changes in a small, stable step, even though the action was very good. This is the mechanism that makes PPO so robust.