

## Lecture 24: RLHF as Game Solving

*Lecturer:* Gokul Swamy

*Scribe:* Apurva Gandhi, Jason Wei, Jehan Yang

### 24.1 Outline

We will cover two points in today's lecture:

1. When is the Bradley-Terry assumption inaccurate and what happens to online/offline PFT as a result?

*A: BT is violated when a reward function can't explain (aggregate) preferences, leading to mode collapse in RLHF.*

2. What is a more robust criterion for preference aggregation and how can we efficiently optimize it?

*A: The minimax winner doesn't assume transitivity of preferences. We can use a self-play algorithm to compute it.*

### 24.2 When is Bradley-Terry assumption inaccurate?

#### 24.2.1 Preference Matrix

Given some set of human preferences recorded in a dataset  $\mathcal{D}$ , we can transform them into a **preference matrix**  $\mathcal{P}$  where each entry  $\mathcal{P}(\xi_i \succ \xi_j)$  is the empirical probability that generation  $\xi_i$  is preferred over  $\xi_j$ :

$$\mathcal{P}(\xi_i \succ \xi_j) \triangleq \mathbb{P}_{\mathcal{D}}(\xi_i \succ \xi_j). \quad (24.1)$$

From this definition, we can back out a few key properties of  $\mathcal{P}$ :

1. All elements on the diagonal must be 0.5.
2. Elements across the diagonal must sum to 1 (as an event and its complement have a probability that sums to 1). This property will be key later.



Figure 24.1: Observe how, without loss of generality, we can translate human preferences (left) into a matrix (right). The same is not true for reward functions.

Above, we give an example of the translation from dataset  $\mathcal{D}$  to matrix  $\mathcal{P}$ . For this particular preference matrix, we can back out a reward function that *rationalizes* it (i.e. that the raters could have had in their heads to guide the choices they made). Assuming our raters make choices according to the Bradley-Terry model of preferences, we know that the underlying  $r$  must satisfy the following set of constraints:

$$r(\xi_1) > r(\xi_2) \quad \& \quad r(\xi_1) > r(\xi_3) \quad \& \quad r(\xi_2) = r(\xi_3). \quad (24.2)$$

One such reward function is  $r = [1, 0, 0]$ . Unfortunately, it is not always possible to collapse a preference *matrix* into a reward *vector*. For example, if we instead had the following three constraints, there is no scalar reward function that simultaneously satisfies them all:

$$r(\xi_1) > r(\xi_2) \quad \& \quad r(\xi_1) < r(\xi_3) \quad \& \quad r(\xi_2) > r(\xi_3). \quad (24.3)$$

This is known as *intransitivity* – a lack of a global ordering over completions. We will now discuss where intransitivity comes from in practice and how to handle it.

## 24.2.2 Violations of Bradley-Terry

Recall that the Bradley-Terry model states that the probability of preferring  $\xi_i$  over  $\xi_j$  is:

$$P(\xi_i \succ \xi_j) = \frac{\exp(r(\xi_i))}{\exp(r(\xi_i)) + \exp(r(\xi_j))} \quad (24.4)$$

BT assumes that population-level preferences can be explained by a single reward, shared function. However, this assumption breaks when preferences are **intransitive** (e.g.  $\xi_1 \succ \xi_2 \succ \xi_3 \succ \xi_1$ ). This happens mainly due to the following reasons:

- **Intransitivity in Individual Preferences:** An individual may judge different pairs of samples on different features/criteria (e.g. we compare apples based on color but apples and oranges based on sweetness).

- **Intransitivity from Preference Aggregation:** Even if each individual has consistent preferences, aggregating preferences across a group of individuals can lead to intransitivity (e.g. rock-paper-scissors-like scenarios).

For the rest of the lecture, we will focus on the second case, where the intransitivity arises from the aggregation of preferences across multiple individuals.

### 24.2.3 Intransitivity from Preference Aggregation

In real-world scenarios, when collecting a preference dataset, for any single individual, we usually only have preferences over a small subset of pairs of prompt completions.

We can turn this data into a *partial preference matrix*  $\mathcal{P}$ , which has unknown entries:

  
 $\mathcal{D}$


$\xi_1 \succ \xi_2$

$\mathcal{P}$	$\xi_1$	$\xi_2$	$\xi_3$
$\xi_1$	0.5	1	?
$\xi_2$	0	0.5	?
$\xi_3$	?	?	0.5


Figure 24.2: A limited number of comparisons from a single user can lead to a partially completed preference matrix. The question marks indicate pairs for which we have no data.

Even if the known entries are consistent with a reward function (e.g.,  $r(\xi_1) > r(\xi_2)$ ), the full matrix might still exhibit intransitivity once the missing entries are filled in with data from other users. We will walk through an example of this now, inspired by Rock Paper Scissors.


Rock  $\succ$  Scissors



Paper  $\succ$  Rock



Scissors  $\succ$  Paper



$\mathcal{P}_1$	R	P	S
R	0.5	?	1
P	?	0.5	?
S	0	?	0.5

$\mathcal{P}_2$	R	P	S
R	0.5	0	?
P	1	0.5	?
S	?	?	0.5

$\mathcal{P}_3$	R	P	S
R	0.5	?	?
P	?	0.5	0
S	?	1	0.5

Figure 24.3: Consider aggregating the preferences of three raters, each of which only compares two of the three options for us. This will lead to the appearance of intransitivity.

Aggregating these preferences can result in the following matrix:

$\mathcal{P}_{agg}$	Rock	Paper	Scissors
Rock	0.5	0	1
Paper	1	0.5	0
Scissors	0	1	0.5

$\mathcal{P}_{agg}$  exhibits intransitivity: Rock  $\succ$  Scissors, Scissors  $\succ$  Paper, and Paper  $\succ$  Rock. Thus, there can be no reward function  $r^*$  that can explain these preferences simultaneously.

#### 24.2.4 No $r^*$ Leads to Mode Collapse in RLHF

When the underlying aggregated preferences  $\mathcal{P}$  are intransitive, no single reward function  $r^*$  can perfectly capture them. Standard RLHF methods attempt to find the Maximum Likelihood Estimate (MLE) reward function  $\hat{r}_{mle}$  under the Bradley-Terry model. However, this  $\hat{r}_{mle}$  will be a poor fit for the true population preferences, leading to mode collapse.

Consider a slightly different intransitive preference matrix (rows preferred over columns):

$\mathcal{P}$	Rock	Paper	Scissors
Rock	0.5	0.7	0
Paper	0.3	0.5	1
Scissors	1	0	0.5

Figure 24.4: Another set of intransitive preferences. Row sums are 1.2, 1.8, 1.5.

Assume we have a uniform reference policy – i.e.  $\hat{\pi}_{ref} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ . In such a situation, loosely speaking, we would expect fitting a BT reward model via MLE would lead to higher values on the choices corresponding to rows with higher sums (i.e. the choices that have higher probabilities of being preferred by the raters on average). For simplicity, let’s assume  $\hat{r}_{mle} = [1.2, 1.8, 1.5]$ . Next, recall that the soft RL policy has the form

$$\hat{\pi}_{RLHF} \propto \pi_{ref} \cdot \exp\left(\frac{1}{\beta} \hat{r}_{mle}\right). \quad (24.5)$$

This means that even if the reward model puts a slightly higher value on something (e.g. paper), the resulting soft-optimal policy will choose paper significantly more often. This leads to **mode collapse**: the RLHF policy the flawed  $\hat{r}_{mle}$  and primarily generates outputs corresponding to the highest estimated reward (Paper), neglecting other valid or even superior options in certain contexts (like Rock, which beats Scissors, or Scissors, which beats Paper). The policy fails to capture the cyclic nature of the true preferences. In practice, this

can correspond to ignoring minority preferences and instead only generating completions preferred by the majority, which can be undesirable in a variety of situations.

We conclude by noting that beyond the toy instance above, this problem becomes increasingly severe as the size of the output space  $\Xi$  grows larger, as it becomes increasingly unlikely that there is a global ordering over all natural language prompt completions all raters agree with.

## 24.3 Beyond Bradley-Terry in RLHF

Given *any* preference matrix, e.g.:

$\mathcal{P}$	Rock	Paper	Scissors
Rock	0.5	0.7	0
Paper	0.3	0.5	1
Scissors	1	0	0.5

We can transform this matrix into an *anti-symmetric matrix* (i.e. one where elements across the diagonal are negations) by multiplying each element by 2 and subtracting by 1:

$\mathcal{P}$	Rock	Paper	Scissors
Rock	0	0.4	-1
Paper	-0.4	0	1
Scissors	1	-1	0

Observe that this is now the *payoff matrix* for a two-player zero sum game. Furthermore, it is a *symmetric* game, which means swapping strategies with your opponent leads to the opposite outcome for both players. We can now consider solving the corresponding game:

$$\pi_1^*, \pi_2^* = \arg \max_{\pi_1 \in \Pi} \arg \min_{\pi_2 \in \Pi} \mathbb{E}_{\xi_1 \sim \pi_1, \xi_2 \sim \pi_2} [2\mathcal{P}(\xi_1 \succ \xi_2) - 1]. \quad (24.6)$$

the solutions to which are known as *Von Neumann / Minimax winners*. Before we discuss how to solve this game, we first note that this solution concept has appealing properties:

- Does not assume we can collapse the preference matrix into a reward function and therefore doesn't assume transitivity / a shared reward function.
- Learns a strategy that is robust against the worst-case comparator. More formally, MWs are preferred to any other policy with probability at least  $\frac{1}{2}$ .

For the preceding game the MW is  $[5/12, 5/12, 1/6]$ , which does not suffer from mode collapse. This reflects the fact that equilibrium strategies are often randomized. Also observe that the above strategy is not merely a scaled version of the policy we’d get out of soft RL – we’re instead learning something qualitatively different and more robust.

## 24.4 SPO: Self-Play Preference Optimization

Unfortunately, doing adversarial training at the scale of LLMs can be challenging in practice. In response, we will now describe how a simple, *self-play* strategy can be used instead to compute an approximate equilibria, leveraging the symmetry of the above game.

**Proof:** First, let us consider solving the above game by running two no-regret algorithms against eachother. As usual, we can define a sequence of losses for each player:

$$\ell_t^1(\pi) = \mathbb{E}_{\xi \sim \pi, \xi' \sim \pi_2^t} [2\mathcal{P}(\xi \succ \xi') - 1], \quad \ell_t^2(\pi) = \mathbb{E}_{\xi \sim \pi_1^t, \xi' \sim \pi} [-(2\mathcal{P}(\xi \succ \xi') - 1)]. \quad (24.7)$$

Without loss of generality, assume that  $\pi_1^0 = \pi_2^0$ . Then, we have

$$\ell_0^1(\pi) = \mathbb{E}_{\xi \sim \pi, \xi' \sim \pi_2^0} [2\mathcal{P}(\xi \succ \xi') - 1] \quad (24.8)$$

$$= \mathbb{E}_{\xi \sim \pi, \xi' \sim \pi_1^0} [2\mathcal{P}(\xi \succ \xi') - 1] \quad (24.9)$$

$$= \mathbb{E}_{\xi \sim \pi_1^0, \xi' \sim \pi} [-(2\mathcal{P}(\xi \succ \xi') - 1)] \quad (24.10)$$

$$= \ell_0^2(\pi), \quad (24.11)$$

where we use the anti-symmetry of the payoff matrix in the second to last equality. If we feed two (deterministic) no-regret algorithms the same sequence of loss functions and start them from the same initial strategy, they will produce identical iterates. Thus, an implication of the above is that  $\forall t \in [T], \pi_1^t = \pi_2^t$ . ■

Practically, this means that rather than needing to have two separate policy networks and memory and dealing with the instabilities of adversarial training, we can instead merely treat a second sample from the same policy as a sample from a fictitious second player.

To implement SPO, one first trains a pairwise *preference* model  $\hat{\mathcal{P}}$  that maps from pairs of completions to the probability the former is preferred to the latter. This can be done via standard maximum likelihood estimation (MLE) but doesn’t assume transitivity like a reward model. Observe that this is just using a function approximator to model the ground-truth preference matrix,  $\mathcal{P}$ . One then samples multiple times from the policy, queries the preference model on each pair, and computes a *win rate* for each sample – i.e., an empirical

estimate of the above loss function. One can then use these win-rates as rewards for a downstream RM-based policy optimization procedure. In short, the changes from standard RLHF are (1) training a preference rather than a reward model and (2) sampling multiple completions per prompt (which one often does anyway in practice). We visualize the full process below. One can also use another LLM as the preference model, often referred to as LLM-as-a-judge. Because such models are trained on internet data, they often exhibit intransitivities in their judgments due to the implicit preference aggregation caused by scale.

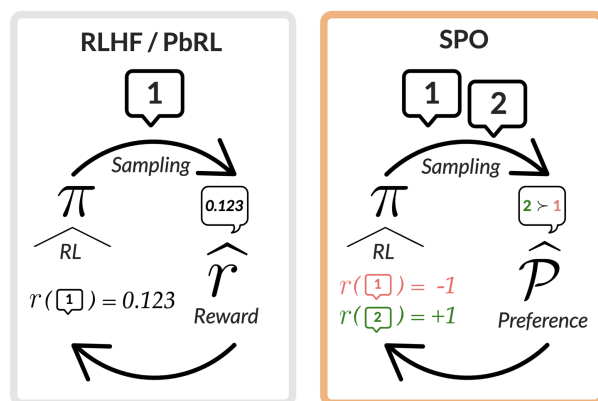


Figure 24.5: In SPO (Self-Play Preference Optimization), one compares a policy to itself via the use of a learned preference model or “judge” LLM. In particular, one samples multiple completions from the policy, asks the preference model to pick a winner for each pairwise comparison, and then uses the win rate of each completion as the reward for an RL procedure.