CIS 6200: Emerging Theory of LLMs **Fall 2026**

Topic 5C: Learning from Preferences

Last updated: October 22, 2026 *Surbhi Goel*

Acknowledgements: *The preference model follows Bradley and Terry [1]. The RLHF setup draws on Ouyang et al. [4] and Lambert's exposition [7]. The policy–reward correspondence and the DPO derivation follow Rafailov et al. [5]. The discussion of heterogeneous preferences draws on Gölz, Haghtalab, and Yang [9] and Haghtalab's exposition [10]. Nash learning follows Munos et al. [13]; the minimax exposition draws on Syrgkanis [11] and Roth [12].*

AI Disclosure: *These notes were drafted and revised with AI assistance. If you find an error, please let me know.*

In the last lecture, we used policy gradients to optimize a reward that we could evaluate on each generated response. For some tasks, an automatic verifier can check correctness. For summarization, there may be many good responses and no simple correctness test. A person can often compare two summaries more easily than assign either one a numerical score. We want to use these comparisons to train the policy.

Reinforcement learning from human feedback (RLHF) addresses this problem in two stages: learn a reward model from comparisons, then optimize that reward. This approach predates recent reasoning models: comparisons were used to train agents in 2017 [2], summarization models in 2020 [3], and InstructGPT in 2022 [4].

1 Learning rewards from comparisons

Who supplies the feedback? Early RLHF datasets were collected by hiring human annotators. For InstructGPT, annotators were shown a prompt and several model responses and asked to rank the responses using written guidelines. These rankings supplied pairwise comparisons [4]. Companies also collect feedback from users: a chat interface may show two answers and ask which one the user prefers. Another option is an **AI judge**: a model receives the prompt, two responses, and instructions about what makes a good answer, then selects the preferred response [6].

Let $x \sim \mathcal{D}$ be a prompt and $\mathcal{U}$ a finite set of complete responses. As in the previous lecture, finiteness follows by bounding response length over a finite vocabulary. We assume finitely many prompts, and all logarithms are natural.

For each prompt, a labeler compares two generated responses and selects the one they prefer. We write y_w for the selected response and y_ℓ for the other, giving the dataset

$$\mathcal{P} = \{(x_i, y_{w,i}, y_{\ell,i})\}_{i=1}^{n}.$$

The label records a preference, not whether either response is correct; both may be reasonable, and different labelers may choose differently.

1.1 The Bradley–Terry model

The data give us choices between responses, but no numerical scores. Since the same pair can receive different labels, we model the probability of each choice. Assign a real-valued score to each response

and let the score difference determine this probability. The **Bradley–Terry model** [1] assumes an unknown score $r^*(x, y) \in \mathbb{R}$ such that

$$\mathbb{P}(y_1 \succ y_2 \mid x) = \frac{e^{r^*(x,y_1)}}{e^{r^*(x,y_1)} + e^{r^*(x,y_2)}} = \sigma(r^*(x, y_1) - r^*(x, y_2)) \,. \tag{1}$$

Here $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid function. This is a softmax over two scores, or a logistic model of their difference. Equal scores give either response probability $1/2$ of being selected; a larger score difference makes y_1 more likely to be preferred. The difference is the log odds of that preference, so binary labels can reveal real-valued score differences.

For each prompt, one score function r^* must explain the population preference probabilities for every response pair. This common-score assumption allows labelers to disagree on individual comparisons.

To learn the scores, parameterize a reward model r_ϕ. It assigns probability $\sigma(r_\phi(x, y_w) - r_\phi(x, y_\ell))$ to a recorded comparison. Holding $\mathcal{P}$ fixed, fit the model by minimizing the negative log likelihood of these choices:

$$\mathcal{L}_{\mathrm{RM}}(\phi) = -\frac{1}{n} \sum_{i=1}^{n} \log \sigma(r_\phi(x_i, y_{w,i}) - r_\phi(x_i, y_{\ell,i})) \,. \tag{2}$$

Writing each pair in the order (winner, loser) gives this same expression for either label. The fitted reward r_ϕ estimates r^*; the recorded winner is one sample from (1) and need not win every repeated comparison.

Only score differences appear in (1). Adding the same constant $c(x)$ to every response score for a prompt leaves all comparison probabilities unchanged. The scores are therefore identifiable only up to a prompt-dependent constant: comparisons can determine relative scores, but not their absolute level.

2 From a learned reward to a policy

Once we have learned r_ϕ, we can hold it fixed and apply REINFORCE as in the previous lecture. But optimizing a learned reward can favor responses that it scores incorrectly. We also want to retain the useful behavior of the model from which we start. RLHF therefore adds a penalty for departing from a fixed reference policy [4]. The comparison model specifies how to fit a reward; we must also choose an objective for training the policy.

Write $q_\theta(y \mid x)$ for the trainable policy, with differentiable, positive probabilities. Let $q_{\mathrm{ref}}(y \mid x) > 0$ be a fixed reference policy, typically obtained by supervised fine-tuning. Initialize q_θ from q_{ref}, then update θ while holding both the reward model and reference fixed.

2.1 The KL-regularized objective

Fix a prompt and suppress x. Write r for the fixed reward used in policy optimization; in RLHF this is the fitted reward r_ϕ. For a candidate response distribution q, define

$$\begin{aligned} J_\beta(q) &= \mathbb{E}_{y \sim q}[r(y)] - \beta \, \mathrm{KL}(q \| q_{\mathrm{ref}}), \qquad \beta > 0, \\ \mathrm{KL}(q \| q_{\mathrm{ref}}) &= \sum_{y \in \mathcal{U}} q(y) \log \frac{q(y)}{q_{\mathrm{ref}}(y)}. \end{aligned} \tag{3}$$

We use the convention $0 \log(0/p) = 0$. The KL divergence is nonnegative, with equality exactly when $q = q_{\mathrm{ref}}$. The coefficient β controls how much deviation from the reference we penalize. The penalty

measures change relative to the reference probabilities, so moving substantial mass to previously rare responses is costly. It discourages large changes, but does not make the learned reward accurate.

Training maximizes (3), averaged over prompts drawn from $\mathcal{D}$. The policy-gradient method still applies, now accounting for the KL penalty as well as the reward.

Using the policy gradient. The KL penalty also depends on θ. Applying the product rule and the log-derivative identity to (3) gives

$$\begin{aligned}\nabla_\theta J_\beta(q_\theta) &= \sum_y \left(r(y) - \beta \log \frac{q_\theta(y)}{q_{\text{ref}}(y)} - \beta \right) \nabla_\theta q_\theta(y) \\ &= \mathbb{E}_{y \sim q_\theta} \left[\left(r(y) - \beta \log \frac{q_\theta(y)}{q_{\text{ref}}(y)} \right) \nabla_\theta \log q_\theta(y) \right].\end{aligned} \tag{4}$$

The constant term vanishes because $\sum_y \nabla_\theta q_\theta(y) = \nabla_\theta 1 = 0$. As in the previous lecture, sample responses from the current policy and weight their likelihood gradients. The weight is now reward minus the log-ratio penalty. Hold its numerical value fixed when evaluating this expression; the derivation already accounts for its dependence on θ. The reward model scores the new responses without further human labels.

2.2 The Gibbs optimal policy

For a fixed prompt, we can solve the regularized optimization problem exactly over all response distributions. The formula for the optimum will let us express a reward through its optimal policy and train that policy directly from comparisons [5, Section 4 and Appendix A.1].

Theorem 2.1 (Gibbs optimum)**.** *For a finite response set $\mathcal{U}$, a reference $q_{\text{ref}}(y) > 0$, a reward $r : \mathcal{U} \to \mathbb{R}$, and $\beta > 0$, the unique maximizer of J_β is*

$$q_r^*(y) = \frac{q_{\text{ref}}(y) e^{r(y)/\beta}}{Z_r}, \qquad Z_r = \sum_{z \in \mathcal{U}} q_{\text{ref}}(z) e^{r(z)/\beta}. \tag{5}$$

Proof. Taking logarithms of the Gibbs formula and solving for $r(y)$ gives

$$r(y) = \beta \log \frac{q_r^*(y)}{q_{\text{ref}}(y)} + \beta \log Z_r. \tag{6}$$

Substitute this into the objective. For any distribution q,

$$\begin{aligned}J_\beta(q) &= \beta \sum_y q(y) \left[\log \frac{q_r^*(y)}{q_{\text{ref}}(y)} - \log \frac{q(y)}{q_{\text{ref}}(y)} \right] + \beta \log Z_r \\ &= \beta \log Z_r - \beta \, \mathrm{KL}(q \| q_r^*) \le \beta \log Z_r.\end{aligned} \tag{7}$$

KL divergence is nonnegative, and it is zero exactly when $q = q_r^*$. Thus q_r^* attains the upper bound and is the unique maximizer. □

This optimum is over all response distributions. A neural policy may not represent it for every prompt, and the theorem makes no claim about convergence of a training algorithm.

Preference probability is not generation probability. Suppose there are two responses, a and b. The reference generates a with probability $1/10$, but labelers prefer it to b with probability $3/4$. A Bradley–Terry reward fitting these comparisons has $r(a) - r(b) = \log 3$. Choose $r(a) = \log 3$ and $r(b) = 0$; for $\beta = 1$, the Gibbs optimum is

$$q_r^*(a) = \frac{(1/10) \cdot 3}{(1/10) \cdot 3 + 9/10} = \frac{1}{4}.$$

The preferred response becomes more likely, from $1/10$ to $1/4$. Its comparison probability describes labelers' choices when both responses are shown. Its generation probability also depends on the reference and the KL penalty.

3 Direct preference optimization

RLHF first fits a reward to comparisons, then trains a policy to optimize it. **Direct preference optimization (DPO)** [5] combines these stages into one optimization: use the Gibbs relation to parameterize the reward through its optimal policy, then fit the comparisons by changing the policy parameters. There is no separate reward model or subsequent reinforcement-learning stage.

3.1 Using the policy to represent the reward

The identity (6) from the Gibbs proof expresses a reward through its optimal policy. Restoring the prompt x, its only term independent of the response is $\beta \log Z_r(x)$. That term cancels when we compare two responses to the same prompt. Thus the log ratio to the reference contains all the reward differences needed by Bradley–Terry, without computing the normalizer.

For any positive trainable policy q_θ, define its **implicit reward** by

$$\widetilde{r}_\theta(x, y) = \beta \log \frac{q_\theta(y \mid x)}{q_{\text{ref}}(y \mid x)}. \tag{8}$$

To check that this policy is optimal for the reward it defines, substitute into the Gibbs formula:

$$q_{\text{ref}}(y \mid x) e^{\widetilde{r}_\theta(x,y)/\beta} = q_\theta(y \mid x).$$

These probabilities already sum to one, so $Z_{\widetilde{r}_\theta}(x) = 1$ and the Gibbs policy is exactly q_θ. Each choice of θ therefore supplies both a reward and a policy optimal for it. By (6), allowing all positive policies represents every reward up to a prompt-dependent constant, which does not affect comparisons.

3.2 The preference likelihood

The implicit reward is defined for every candidate policy. To choose θ, we fit this reward to the comparisons, just as we fit r_ϕ in Section 1. Under Bradley–Terry, the policy predicts the preference probability

$$\mathbb{P}_\theta(y_w \succ y_\ell \mid x) = \sigma(\widetilde{r}_\theta(x, y_w) - \widetilde{r}_\theta(x, y_\ell)) \,.$$

Call the score difference inside the sigmoid the **DPO margin**. Using the definition of $\widetilde{r}_\theta$, it is

$$m_\theta(x, y_w, y_\ell) = \beta \left[\log \frac{q_\theta(y_w \mid x)}{q_{\text{ref}}(y_w \mid x)} - \log \frac{q_\theta(y_\ell \mid x)}{q_{\text{ref}}(y_\ell \mid x)} \right]. \tag{9}$$

Each record tells us that y_w was preferred, so its likelihood is $\sigma(m_\theta)$. Maximizing the likelihood of the recorded choices is equivalent to minimizing the **DPO loss**

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log \sigma(m_\theta(x_i, y_{w,i}, y_{\ell,i}))\,. \tag{10}$$

At initialization $q_\theta = q_{\mathrm{ref}}$, all margins are zero and the predicted preference probabilities are $1/2$.

For an LLM, evaluate both recorded sequences under q_θ and the fixed reference, summing token log probabilities as in SFT. Form the margin and differentiate the loss through q_θ only. These *offline* updates reuse the comparison dataset; policy-gradient RLHF instead generates new responses from the current policy [7].

The role of β. Gibbs divides a reward difference by β to obtain the change in policy log odds. DPO inverts this relation: multiplying that change by β recovers the reward difference. Thus fitting the same preference probability with a larger β requires a smaller change from the reference odds, consistent with a stronger KL penalty.

3.3 When DPO recovers the Gibbs optimum

Every candidate policy is optimal for its own implicit reward. We now show when fitting comparisons recovers the Gibbs policy for r^*, the score that generates the feedback.

Consider the following sampling process: draw a prompt, draw a pair from a fixed distribution of response pairs for that prompt, and then draw the winner according to Bradley–Terry with score r^*. The *population DPO loss* replaces the sample average in (10) by an expectation over this process. For a fixed pair, it averages both possible winners with their true probabilities.

Proposition 3.1 (Population correspondence)**.** *Suppose feedback follows Bradley–Terry with score r^*. For every prompt with positive probability, assume every distinct response pair has positive probability of being compared. If the policy class contains $q^*_{r^*}$, then every global minimizer of the population DPO loss equals $q^*_{r^*}$ on these prompts.*

Proof. Fix a prompt and name the pair (a, b) before drawing its winner. Let $p = \mathbb{P}(a \succ b \mid x)$ and write m for the model's margin in favor of a. The observed winner's margin is m if a wins and $-m$ if b wins. Averaging the two losses gives

$$C_p(m) = -p \log \sigma(m) - (1-p)\log \sigma(-m), \qquad C_p'(m) = \sigma(m) - p.$$

The derivative is negative when $\sigma(m) < p$ and positive when $\sigma(m) > p$. Thus the unique minimum predicts the true probability, with margin $m = \log(p/(1-p)) = r^*(x, a) - r^*(x, b)$.

The Gibbs policy $q^*_{r^*}$ attains these minima for all pairs simultaneously, by (6). Every pair has positive weight in the population loss. A global minimizer must therefore attain every pair's minimum: a larger loss on one pair cannot be offset by going below the minimum on another. Its margins consequently match those of $q^*_{r^*}$. Since β and the reference are fixed, equating the margins in (9) gives

$$\frac{q_\theta(a \mid x)}{q_\theta(b \mid x)} = \frac{q^*_{r^*}(a \mid x)}{q^*_{r^*}(b \mid x)} \qquad \text{for every } a, b.$$

Fixing b shows that the two policies differ only by a common multiplicative factor. Both sum to one, so that factor is one. □

Under these assumptions, population DPO and KL-regularized reward optimization have the same optimal policy, although their objectives and gradients differ. Figure 1 illustrates this for the two-response example.

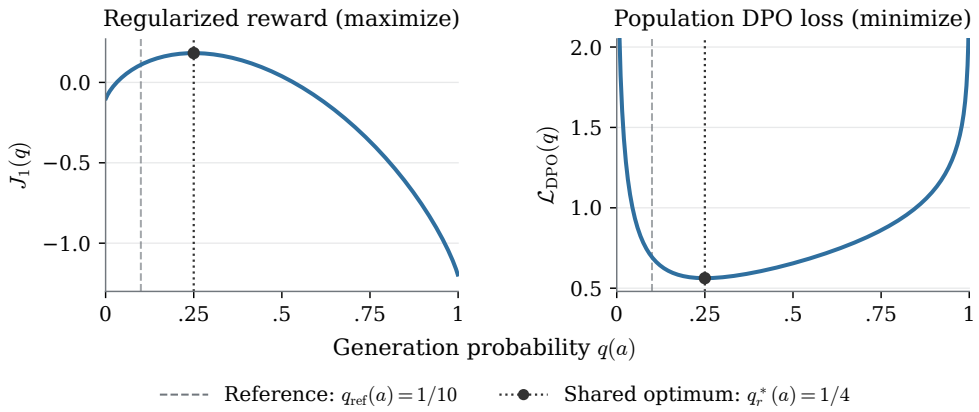

Figure 1: Two responses with $q_{\mathrm{ref}}(a) = 1/10$, $r(a) = \log 3$, $r(b) = 0$, and $\beta = 1$. Regularized reward is maximized and population DPO loss is minimized at $q(a) = 1/4$. The curves show objective values as the policy varies, not training trajectories.

The correspondence above concerns the population loss. For a pair observed only once, its empirical loss keeps decreasing as the recorded winner's margin grows; as a function of a free margin, it has no finite minimizer for any fixed $\beta > 0$ [8, Section 4.2]. Other pairs and policy restrictions can constrain that margin. The theorem alone therefore does not prevent empirical overfitting.

4 When labelers have different preferences

DPO removes the separate reward-training stage, but still represents preferences through one Bradley–Terry score function. The population theorem assumes this function can fit all comparison probabilities. Different groups may each follow Bradley–Terry while their pooled comparisons do not. Fitting a common reward can then reverse a preference, and optimizing it can produce a policy that loses to another policy under the same KL constraint.

4.1 Why fitting one reward can reverse a preference

Fix one prompt with three responses a, b, c, and let $P(y, z)$ be the probability that a random labeler chooses y over z, breaking ties uniformly. Assume these probabilities are known exactly. Consider the following example from Gölz, Haghtalab, and Yang [9, Section 4.1], also presented in Haghtalab's slides [10].

The majority favors a over b, but almost always chooses c over either. A minority strongly favors b over both. This minority does not overturn the majority preference for a over b. But it can make b win more often than a against c, since the majority almost always chooses c in both comparisons. The resulting comparisons satisfy

$$P(a, b) > \frac{1}{2}, \qquad P(a, c) < P(b, c) < \frac{1}{2}.$$

Thus a majority prefers a to b, yet b wins more often against c. A single Bradley–Terry reward cannot express both facts. The first comparison requires $r(a) > r(b)$; an increasing sigmoid would then predict that a also beats c more often than b does.

The reward model must therefore compromise. Which comparisons it fits best depends on how often each response pair is compared. If comparisons with c receive sufficiently more weight than the a, b comparison, the fitted reward can satisfy

$$\hat{r}(c) > \hat{r}(b) > \hat{r}(a).$$

This fits the ordering of win probabilities against c, but reverses the direct preference for a.

4.2 Why the reward optimizer can lose

We compare policies under a fixed KL budget around the same reference. The reward optimizer chooses the allowed policy with the highest expected fitted reward. Choose a reference that gives a, b comparable probability and c very little probability. The budget allows policies concentrated on either a or b, while severely restricting how much probability they can put on c.

Under these choices, the fitted-reward optimizer q_{R} puts almost all its probability on b: substantial probability on c is too costly, and b has higher fitted reward than a. The policy q_a that always produces a is also feasible. Because $P(a, b) > 1/2$, it beats q_{R} more than half the time when q_{R} is sufficiently concentrated on b. The paper's construction chooses the comparison weights, reference, and budget so that all these properties hold.

Both policies satisfy the same KL constraint. The optimizer correctly maximizes its fitted reward, but that reward ranks b above a even though labelers prefer a in a direct comparison. Limiting access to c makes this mistaken ranking matter for the policy.

DPO can have the same problem. With unrestricted policies and a matching regularization coefficient, its population optimum is the Gibbs policy for the fitted reward [9, Appendices E.3–E.4], so it can produce the same q_{R}. This does not contradict the earlier population theorem, which assumed a correct common-score model.

4.3 Nash learning from pairwise comparisons

The challenger q_a beats the reward optimizer, but another allowed policy might beat q_a. We therefore use the comparison probabilities P directly to seek a policy that no allowed challenger beats. Nash learning changes this optimization objective while retaining the same KL constraint.

More generally, let $\mathcal{U}$ be a finite response set for the fixed prompt. A policy q is a probability distribution on $\mathcal{U}$. We retain the known comparison probabilities $P(y, z)$ and assume $P(y, z) + P(z, y) = 1$, with ties broken uniformly. No Bradley–Terry representation is required. Given a reference policy $q_{\mathrm{ref}}(y) > 0$ and a KL budget $\tau \geq 0$, the allowed policies are

$$\mathcal{Q}_\tau = \{q \in \Delta(\mathcal{U}) : \mathrm{KL}(q \| q_{\mathrm{ref}}) \leq \tau\},$$

where $\Delta(\mathcal{U})$ denotes all response distributions. Both policies use this same set. The budget τ bounds KL divergence; β in the earlier objective was a penalty coefficient.

Choose two policies q, q', then draw one response independently from each and ask a random labeler to compare them. The first policy's win probability is

$$W(q, q') = \mathbb{E}_{y \sim q,\, z \sim q'}[P(y, z)].$$

The challenger chooses its policy before either response is drawn. Its win probability is $W(q', q) = 1 - W(q, q')$, so maximizing its own wins is equivalent to minimizing ours.

For a proposed policy q, the inner minimum below finds the allowed challenger against which it wins least often. The outer maximum chooses our policy to make that worst comparison as good as possible:

$$\max_{q \in \mathcal{Q}_\tau} \min_{q' \in \mathcal{Q}_\tau} W(q, q'). \tag{11}$$

This is the KL-constrained **Nash learning from human feedback** objective [13, 9]. The worst case is over competing policies; labelers are still averaged according to their population frequencies.

4.4 What does the objective guarantee?

Two independent draws from the same policy give neither side an advantage, so $W(q, q) = 1/2$. The responses need not be identical. Since the challenger can copy our policy, no policy can guarantee more than $1/2$. The following result shows that this value is attainable.

Theorem 4.1 (A policy that no feasible opponent beats)**.** *Let $\mathcal{U}$ be finite, $q_{\mathrm{ref}}(y) > 0$ for every y, and $\tau \geq 0$. Suppose $P : \mathcal{U} \times \mathcal{U} \to [0, 1]$ satisfies $P(y, z) + P(z, y) = 1$. Then (11) has value $1/2$. In particular, some $q_{\mathrm{N}} \in \mathcal{Q}_\tau$ satisfies*

$$W(q_{\mathrm{N}}, q') \geq \frac{1}{2} \qquad \textit{for every } q' \in \mathcal{Q}_\tau.$$

Proof. The set $\mathcal{Q}_\tau$ is nonempty (it contains q_{ref}), compact, and convex. The last property follows from convexity of KL divergence in its first argument. The win probability W is continuous and linear in either policy when the other is fixed. Under these conditions, the minimax theorem says that the max–min and min–max values agree. In words, choosing our policy after seeing the challenger's policy does not improve the best worst-case guarantee:

$$v := \max_{q \in \mathcal{Q}_\tau} \min_{q' \in \mathcal{Q}_\tau} W(q, q') = \min_{q' \in \mathcal{Q}_\tau} \max_{q \in \mathcal{Q}_\tau} W(q, q').$$

Symmetry gives $W(q, q') + W(q', q) = 1$, hence $W(q, q) = 1/2$. In the first expression, the challenger can choose $q' = q$, so $v \leq 1/2$. In the second, we can choose $q = q'$, so $v \geq 1/2$. Thus $v = 1/2$, and any maximizing policy has the stated guarantee. Using the same feasible set on both sides allows both copying steps. □

Equivalently, every allowed challenger wins against q_{N} with probability at most $1/2$. This guarantee rules out q_{R} in the example because q_a beats it. Showing that q_a is optimal would require checking it against *every* feasible challenger.

Why this is a Nash equilibrium. Consider two players, each choosing a feasible policy and trying to win the comparison. A *Nash equilibrium* is a pair of policies for which neither player can improve its win probability by changing its own policy while the other's stays fixed. If both use q_{N}, each wins half the time. By the guarantee and symmetry, no feasible change can win more often against the unchanged q_{N}. Thus $(q_{\mathrm{N}}, q_{\mathrm{N}})$ is a Nash equilibrium. The labelers judge the responses; they are not the two players choosing policies.

Whose preferences does this guarantee concern? The comparison averages over the labeler population defining P. With AI feedback, P describes the AI labeler's choices. A policy can be unbeaten on average and still be disfavored by a particular group [14]. The guarantee also concerns only challengers satisfying the same KL constraint; it does not assert that q_{N} maximizes average utility. The theorem establishes existence for known P; it does not establish convergence of a training algorithm.

References

[1] R. A. Bradley and M. E. Terry. *Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons.* Biometrika, 39(3–4):324–345, 1952. Paper.

[2] P. F. Christiano et al. *Deep Reinforcement Learning from Human Preferences.* NeurIPS, 2017. Paper.

[3] N. Stiennon et al. *Learning to Summarize from Human Feedback.* NeurIPS, 2020. Paper.

[4] L. Ouyang et al. *Training Language Models to Follow Instructions with Human Feedback.* NeurIPS, 2022. Paper.

[5] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. *Direct Preference Optimization: Your Language Model Is Secretly a Reward Model.* NeurIPS, 2023. Paper.

[6] Y. Bai et al. *Constitutional AI: Harmlessness from AI Feedback.* arXiv:2212.08073, 2022. Paper.

[7] N. Lambert. *RLHF and Post-Training.* Online book, chapters on reward modeling and direct alignment. Accessed October 17, 2026.

[8] M. G. Azar et al. *A General Theoretical Paradigm to Understand Learning from Human Preferences.* AISTATS, PMLR 238:4447–4455, 2024. Paper.

[9] P. Gölz, N. Haghtalab, and K. Yang. *Distortion of AI Alignment: Does Preference Optimization Optimize for Preferences?* NeurIPS, 2025. Paper.

[10] N. Haghtalab. *Distortion of AI Alignment: Does Preference Optimization Optimize for Preferences?* STOC workshop on Understanding Large Language Models via a TCS Lens, 2026. Slides.

[11] V. Syrgkanis. *Applications of Zero-Sum Games to ML and AI.* MS&E 233, Lecture 4, Stanford University, Spring 2026. Slides.

[12] A. Roth. *Maximal Lotteries: Minimax and Self-Play.* CIS 7000: AI Alignment, Lecture 5, 2026. Lecture slides.

[13] R. Munos et al. *Nash Learning from Human Feedback.* ICML, 2024. Paper.

[14] T. Sorensen et al. *Position: A Roadmap to Pluralistic Alignment.* ICML, PMLR 235:46280–46302, 2024. Paper.