17-740 Spring 2025
Algorithmic Foundations of Interactive Learning

Lecture 22: The Value of RL in Fine-Tuning

Lecturer: Gokul Swamy
Scribe: Harshita D., Michael C., Zichun Y., Yiming Z.

22.1 Outline

1. What assumption on the preference dataset did we make in the DPO derivation and what happens when it breaks?
   A: Full coverage of $\mathcal{D}$. Without it, we can't control the RKL.

2. When are two-stage RLHF and DPO equivalent?
   A: When Π and $\mathcal{R}$ are isomorphic and all projections are exact.

3. Why does two-stage RLHF work much better in practice?
   A: RLHF only has to search over policies (generators) that are optimal for simple rewards (verifiers) rather than over all of Π, which requires less data.

22.2 Recap

Recall the formulas for *forward* and *reverse* KL divergence.

- FKL: $\min_p \mathbb{D}_{KL}(q||p) = \min_p \sum_x q(x) \log\left(\frac{q(x)}{p(x)}\right)$ (MLE)
- RKL: $\min_p \mathbb{D}_{KL}(p||q) = \min_p \sum_x p(x) \log\left(\frac{p(x)}{q(x)}\right)$ (Soft RL)

FKL is mode-covering while RKL is mode-seeking. To see why, imagine we tried fitting a single Gaussian distribution to a bimodal sum of Gaussians. FKL doesn't want $\frac{q(x)}{p(x)}$ to get too big, so it ensures $p(x)$ is never too small when $q(x)$ is large. This results in it roughly fitting an "average" of the sum of Gaussians. However, RKL doesn't want $\frac{p(x)}{q(x)}$ to get too big, so it ensures $p(x)$ is never too large when $q(x)$ is close to zero. This results in it just trying to fit to one of the modes. Note that both of these extremes (average of the two modes from FKL vs. just one mode from RKL) result in a very high divergence in the other direction.

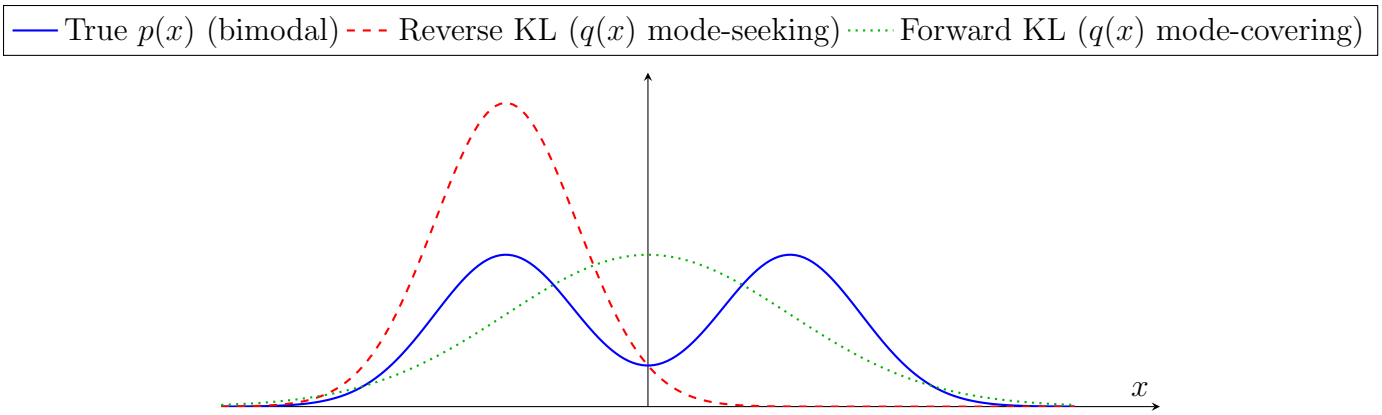

Figure 22.1: Forward KL covers both modes while reverse KL selects one when trying to fit a bimodal mixture of Gaussians with a single Gaussian.

In RLHF, we often care about mode selection (mode-seeking) more than mode covering (i.e. we'd prefer to do one thing well rather than two things poorly).

Last lecture, we described two methods for learning from preference feedback:

1. Two-stage RLHF: learn a reward model $\hat{r}_{\text{mle}}$ via logistic regression over $\mathcal{D}$. MLE corresponds to minimizing the FKL. Then, learn a policy via soft RL on the RM:

$$\hat{\pi}_{\text{rlhf}} = \arg\max_{\pi \in \Pi} \mathbb{E}_{\xi \sim \pi} \left[\hat{r}_{\text{mle}}(\xi) \right] + \mathbb{D}_{KL}(\pi || \pi_{\text{ref}}). \tag{22.1}$$

   As discussed last time, soft RL corresponds to minimizing the RKL.

2. One-step Direct Alignment (e.g. DPO): directly learn a policy over $\mathcal{D}$ via MLE ($\approx$ logistic regression over the space of policies).

In practice, most if not all frontier models use the two-stage procedure. However, it is significantly more computationally intensive as it requires sampling from the policy and performing finicky policy gradient updates. The key question we will focus on for the rest of this lecture is exploring when this extra complexity in implementation is beneficial.

22.3 DPO Breaks under Partial Coverage

As we will now discuss, DPO implicitly assumes full coverage of the preference dataset $\mathcal{D}$, an assumption that is rarely if ever true in practice. Without this assumption, DPO cannot control the RKL. Intuitively, this is because RKL involves an *on-policy* expectation, which can only be estimated if $\mathcal{D}$ covers all places the learner might visit.

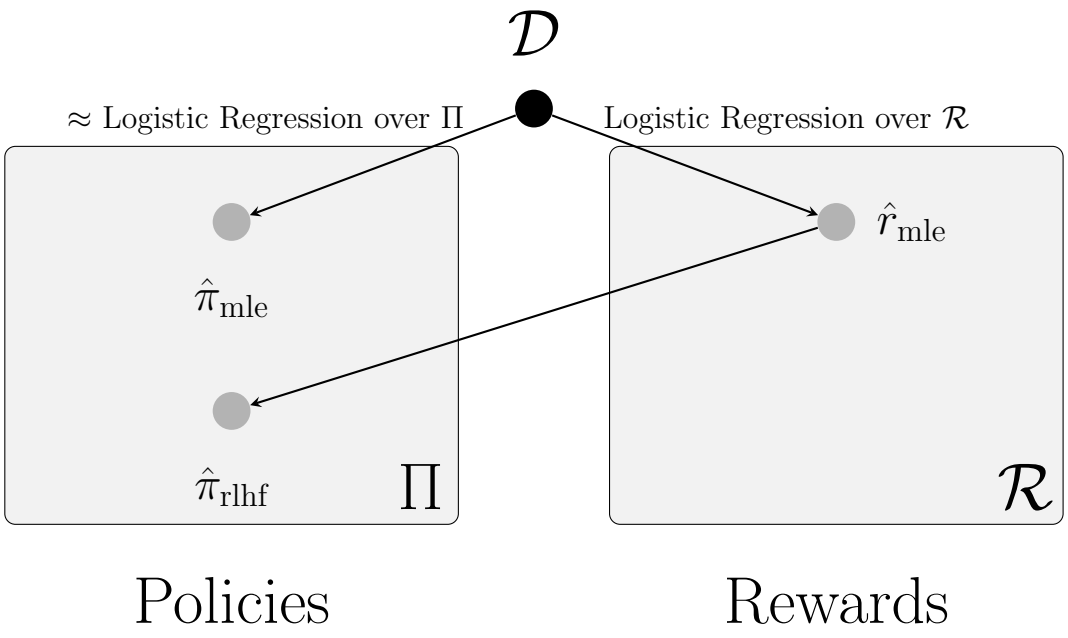

Figure 22.2: We visualize the two-stage and one-stage approaches to PFT.

More explicitly, suppose we had three possible completions ξ_1, ξ_2, ξ_3 with probabilities $0.5, 0.5, 0$ under π_{ref}. Furthermore, assume that our human labeler always prefer $\xi_1 \succ \xi_2$.

First, let's consider the two-stage procedure. When fitting $\hat{r}_{\text{mle}}$ on $\mathcal{D}$, we would want to assign a much higher score to ξ_1 than to ξ_2. However, because we never see any preference data with ξ_3, we could erroneously assign it a high reward. For example, $\hat{r}_{\text{mle}} = [1, 0, 10]$ has a high Bradley-Terry likelihood. However, the RKL term blows up if we place any probability mass on ξ_3, so $\hat{\pi}_{\text{rlhf}}$ wouldn't generate ξ_3 and therefore awards *reward hacking*.

Now, let us consider applying DPO to this problem. Because π_{ref} has the same probability of generating either ξ_1 or ξ_2, we can drop the reference probabilities from the DPO loss and end up with a vanilla logistic regression objective:

$$\arg\max_{\pi \in \Pi} \mathbb{E}_{\mathcal{D}} \left[\log \sigma \left(\sum_h^H \log \frac{\pi(a_h^+|s_h^+)}{\pi_{\mathsf{ref}}(a_h^+|s_h+)} - \log \frac{\pi(a_h^-|s_h^-)}{\pi_{\mathsf{ref}}(a_h^-|s_h^-)} \right) \right] \tag{22.2}$$

$$= \arg\max_{\pi \in \Pi} \mathbb{E}_{\mathcal{D}} \left[\log \sigma \left(\sum_h^H \log \pi(a_h^+|s_h^+) - \log \pi(a_h^-|s_h^-) \right) \right] \tag{22.3}$$

$$= \arg\max_{\pi \in \Pi} \log \sigma \left(\sum_h^H \log \pi(a_h^1|s_h^1) - \log \pi(a_h^2|s_h^2) \right). \tag{22.4}$$

Observe this loss function does not constrain the probability of generating ξ_3 in any way and there is no on-policy RKL correction to fix it. In summary, DPO doesn't regularize to π_{ref}

properly in the partial coverage setting, and can therefore produce OOD responses.

22.4 When are two-stage RLHF and DPO equivalent?

As we will now argue, when the space of policies Π is isomorphic to the space of reward models $\mathcal{R}$ and all projections are exact, DPO and RLHF will produce the same policy (i.e. $\hat{\pi}_{\text{mle}} = \hat{\pi}_{\text{rlhf}}$). For simplicity, we will assume there is no prior regularization, but a more general version of the following claim can be proved under slightly stronger assumptions.

Proof: First, we observe that because we are minimizing the same functional over isomorphic classes, we know that $\hat{\pi}_{\text{mle}}$ and $\hat{r}_{\text{mle}}$ must achieve the same loss value:

$$\mathbb{E}_{\mathcal{D}}\left[\log\sigma\left(\sum_{h}^{H}\left(\log\hat{\pi}_{\text{mle}}(a_h^+ \mid s_h^+) - \log\hat{\pi}_{\text{mle}}(a_h^- \mid s_h^-)\right)\right)\right] = \mathbb{E}_{\mathcal{D}}\left[\log\sigma\left(\hat{r}_{\text{mle}}(\xi^+) - \hat{r}_{\text{mle}}(\xi^-)\right)\right]. \tag{22.5}$$

Assuming uniqueness of minimizers for simplicity, we then know that

$$\forall\,\xi \in \Xi, r_{\hat{\pi}_{\text{mle}}}(\xi) = \sum_{h}^{H}\log\hat{\pi}_{\text{mle}}(a_h \mid s_h) = \hat{r}_{\text{mle}}(\xi). \tag{22.6}$$

Next, we recall that soft RL can be written as an RKL projection onto Π and substitute $\hat{r}_{\text{mle}}(\xi)$ for $r_{\hat{\pi}_{\text{mle}}}(\xi)$:

$$\begin{aligned}\hat{\pi}_{\text{rlhf}} &= \arg\min_{\pi\in\Pi}\mathbb{D}_{KL}\left(\mathbb{P}_\pi \,\|\, \mathbb{P}^\star_{\hat{r}}\right)\\ &= \arg\min_{\pi\in\Pi}\mathbb{D}_{KL}\left(\mathbb{P}_\pi \,\|\, \mathbb{P}_{r_{\hat{\pi}}}\right)\\ &= \hat{\pi}_{\text{mle}}\end{aligned}$$

The last step above uses the DPO isomorphism we discussed last lecture. ■

If you've taken a statistics class before, you might have heard this stated more formally as *MLE is invariant to reparameterization*. Interestingly enough, we're roughly in the isomorphic classes setting for practical applications of RLHF to LLMs. This is because both the policy and the reward model are fine-tuned from the same SFT checkpoint. Specifically, to get an RM, one removes the final softmax from the SFT policy and then adds a linear layer.

The above result should be surprising to you – we've proved theoretically that there should be no benefit to RL in fine-tuning but practically, we see everyone using RL-based methods. We will now focus on reconciling this theory-practice gap via the use of controlled experiments.

22.5 Why is RLHF > DPO in Practice?

We will focus on the task of summarization of Reddit posts, using models from the Pythia family pre-trained on the Pile. To eliminate confounders, we will try to equalize as many things as possible between offline and online PFT:

1. We will use the **same** dataset to train both policies and reward models.
2. We will start from the **same** SFT checkpoint to train both.
3. We will use the **same** optimizer (DPO) for both online and offline PFT with the **same** hyperparameters.

By online DPO, we mean sampling from the policy, ranking the samples with the reward model, and using the top and bottom of the list as a new preference pair for a DPO procedure. Thus, the *only* difference between the offline and online PFT procedures is the data passed to them – the offline algorithm learns directly from the human preferences, while the online procedure uses the inputed preferences generated by reward model rankings.

Key Takeaways While maintaining the same experimental configuration (same initial checkpoint, training data, objective function, same gradient steps) online PFT robustly out-performs offline PFT. See [1] for full results.

22.5.1 Hypotheses for the Online-Offline Gap

We now explore and refute several hypotheses for the online-offline performance gap – see [1] for the complete list and more comprehensive experiments.

1. **Hypothesis 1: Intrinsic Value of On-Policy Feedback.** If the RM was able to give us new information during on-policy sampling, we might hope to out-perform the offline algorithm. This is the heart of the classic separations between online and offline RL. However, we do not observe any ground-truth reward labels during on-policy sampling – the labels are merely imputed by an RM trained on the same data as the offline policy. Furthermore, due to the data processing inequality, we cannot create any new information (i.e. bona fide new human preferences) via on-policy sampling.
2. **Hypothesis 2: Failure of Offline Regularization.** As we discussed above, offline algorithms like DPO need strong assumptions to properly regularize to the prior. While this is true in general, we used DPO as our optimizer for both the online and offline experiments. Thus, this fact cannot explain the preceding results.

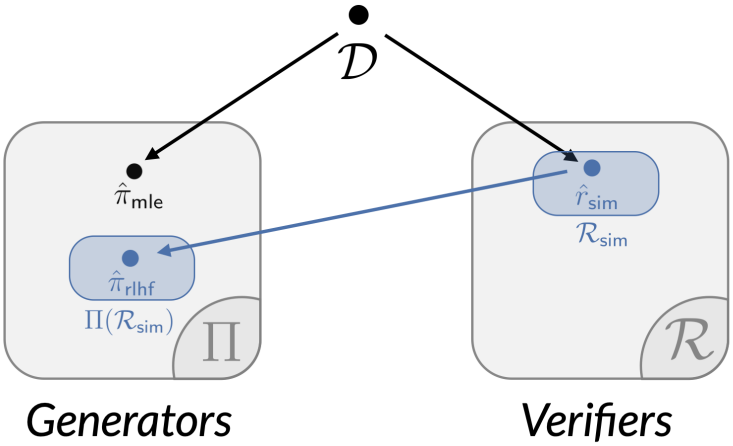

Figure 22.3: On problems with a *generation-verification gap*, the first stage of RLHF can be thought of picking a relatively simple reward model $\hat{r}_{\text{sim}} \in \mathcal{R}_{\text{sim}}$. The secondary RL step will pick a policy that is soft-optimal for this relatively simple verifier. Thus, end to end, the two-stage RLHF procedure only needs to look at policies that are soft-optimal for relatively simple verifiers $\Pi(\mathcal{R}_{\text{sim}})$, rather than across all of Π like vanilla maximum likelihood estimation. This provides a *statistical* benefit (i.e., fewer samples required for learning).

3. **Hypothesis 5: RMs generalize better OOD than Policies.** Intuitively, if our RM generalizes better OOD than the *implicit* RM r_π we train in DPO-like approaches, we might expect online PFT perform better. This is true (as measured in terms of Best-of-N performance), but is best explained by the fact that RMs also generalize better *in-distribution* (i.e. they have better validation BT likelihoods). Intutively, having better margins in distribution is often correlated with better OOD generalization, similar to the arguments for maximizing the margin of an SVM. However, this just kicks the can down the road – we haven't explained *why* RMs should have better BT validation likelihoods ID than implicit RMs. It isn't obvious they should, given we're minimizing the same loss function over similar function classes using the same dataset.

22.5.2 Generation-Verification Gaps in RLHF

We now discuss an alternative hypothesis grounded in the phenomenon of *generation-verification gaps*: that for many problems, it is easier to *check* that one has the right answer rather than to generate it in the first place. For example, checking if a Sudoku puzzle is correctly solved is much easier (just check every row, column and square) rather than generating a complete solution (requires an exponentially large search). A bit more formally, this is what problems in NP but not P are like. In RL, policies are *generators* while reward models are *verifiers*. If

we believe that the underlying reward function for a problem is simpler than the corresponding soft-optimal policy, this means that it should take less data to learn the former. Then, the second RL stage of RLHF only picks policies that soft-optimal for the relatively simple learned verifier. Thus, end to end, two-stage RLHF only has to look at the *subset* of policies that are soft-optimal for relatively simple verifiers, rather than across *all* policies, which can require much more preference data to search across. This hypothesis is saying there is a *statistical* separation between online and offline PFT. We note in passing that this hypothesis is analogous to the statistical benefit of inverse RL we discussed where the learner has a reduction in search space to just those policies that are on the Pareto frontier. We visualize this reduction in effective hypothesis class size in Figure 22.3. Observe that this hypothesis explains why there should still be an online-offline performance gap in the isomorphic classes setting. Furthermore, it explains why RMs generalize better ID than implicit RMs, as one is attempting to learn a simpler function in the former case and therefore should require less data to do so (via one of many arguments from statistical learning theory).

[1] provide evidence that the summarization problem we consider has a GV-gap by noting relatively small reward models can accurately rank samples from relatively large policy. Furthermore, they observe that using an RM that is much larger than the generating policy does not improve upon the performance of an RM of the same size as the generating policy. They also discuss how this hypothesis explains all other experimental results in their paper. To further stress-test the hypothesis, they then perform experiments where they eliminate the GV-gap and see that offline and online PFT collapse to the same performance, as predicted by the information-theoretic picture we started off the lecture with.

22.6 *

References

[1] Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025.