17-740 Spring 2025
Algorithmic Foundations of Interactive Learning

Lecture 23: RL via Regressing Relative Rewards

Lecturer: Wen Sun
Scribe: Pranjal Aggarwal

23.1 Recap

Recall that in RLHF, we're trying to solve the following KL-regularized RL problem:

$$J(\pi) = \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{\tau\sim\pi(\cdot|x)} r(x,\tau) - \beta \mathrm{KL}(\pi(\cdot|x) \parallel \pi_{\mathrm{ref}}(\cdot|x))\right], \tag{23.1}$$

where

- x is the input context (prompt),
- τ is the sequence of actions / tokens or response generated by the policy π,
- $r(x,\tau)$ is the reward associated with the prompt-response pair (x,τ),
- β is a scalar factor controlling KL penalty strength,
- π_{ref} is the reference policy.

As we discussed previously, the optimal policy $\pi^\star$ has the following closed form:

$$\hat{\pi}(\tau|x) \propto \pi_{\mathrm{ref}}(\tau|x) \cdot \exp\left(\frac{r(x,\tau)}{\beta}\right). \tag{23.2}$$

23.1.1 Recap: DPO

DPO takes the closed-form solution above and reparameterizes the reward using the policy. It models the reward difference in terms of the policy. The DPO training objective is:

$$\arg\max_\theta \sum_{x,\tau,\tau',z} \ln \frac{1}{1+\exp\left(-\beta\left(\ln\frac{\pi_\theta(\tau|x)}{\pi_{\mathrm{ref}}(\tau|x)} - \ln\frac{\pi_\theta(\tau'|x)}{\pi_{\mathrm{ref}}(\tau'|x)}\right)\right)}, \tag{23.3}$$

where

- x is a prompt,
- τ, τ' are responses to x,
- $z \in \{-1, 0, 1\}$ labels preference: $z = 1$ if τ is preferred to τ',
- $\pi_\theta(\tau|x)$ is the probability of generating τ given x under policy π_θ,
- $\pi_{\theta_{\text{ref}}}(\tau|x)$ is the probability under the reference policy π_{ref}.

DPO's performance is often weaker than Reward Model (RM) + PPO due to the *generation-verification gap*: evaluating a response is often easier than generating one. Furthermore, online RL approaches allow one to use state-of-the-art RMs (e.g., from RewardBench) trained by the community and train the policy on prompts unseen in the preference data.

Main Question: PPO is computationally expensive, requiring storing four large models in memory: π, π_{ref}, the reward model r, and the critic / value function V. Can we develop a more efficient and potentially more effective RL algorithm?

23.2 Mirror Descent

Mirror Descent is a well-studied no-regret algorithm that most, if not all, popular RL algorithms can be thought of as approximating. [1] We will begin by discussing this idealized algorithm before describing how best to approximate it. Given some reward function $r(x, \tau)$, RL focuses on finding some policy π that maximizes expected reward:

$$\max_\pi \mathbb{E}_{x,\tau\sim\pi(.|x)}[r(x, \tau)]. \tag{23.4}$$

At each iteration, t, *mirror descent* attempts to solve the above via the following update:

$$\pi_{t+1} = \arg\max_\pi \mathbb{E}_{x,\tau\sim\pi_t(.|x)} \left[r(x, \tau) - \beta \text{KL}(\pi(\cdot|x) \parallel \pi_t(\cdot|x))\right]. \tag{23.5}$$

Observe that this is nothing but the KL-regularized / MaxEnt RL problem we've been studying, but with $\pi_{\text{ref}} = \pi_t$ (i.e. we regularize to the last policy we learned). Pattern matching from the above, we know that the closed-form solution to the above problem is

$$\pi_{t+1}(\tau|x) = \frac{\pi_t(\tau|x) \exp(r(x, \tau)/\beta)}{Z(x)}, \tag{23.6}$$

[1] In fact, the Follow The Regularized Leader algorithm we discussed extensively in earlier lectures is equivalent to mirror descent for a particular choice of the regularizer.

where $Z(x)$ is the normalization constant. We can expand out this recursion over T iterations:

$$\pi_{t+T}(\tau|x) \propto \pi_t(\tau|x) \exp(\sum_{i=1}^{T} r(x,\tau)/\beta) \propto \pi_{\text{ref}}(\tau|x) \exp(\sum_{i=1}^{t+T} r(x,\tau)/\beta) \tag{23.7}$$

This update should be intuitive: we're steadily up-weighting high-reward trajectories and down-weighting the prior π_{ref}. Observe that this down-weighting implies that we may eventually drift away from the reference policy, but this is acceptable if we trust our reward model. We don't prove in lecture but exact mirror descent has a *fast* rate of $\mathcal{O}(1/T)$.

While conceptually elegant, exact mirror descent is difficult to implement if π and r are neural networks (transformers) – exactly computing the above product for *all* τ (i.e. all prompt completions) is computationally infeasible. Thus, we will now discuss how to approximate this idealized update via a simple, square-loss regression.

23.3 Reparameterization Trick and REBEL

Taking a log on both sides of the above expression and re-arranging terms, we get:

$$r(x,\tau) = \beta \left(\ln \frac{\pi_{t+1}(\tau|x)}{\pi_t(\tau|x)} \right) + \ln Z(x) \tag{23.8}$$

where $Z(x) = \mathbb{E}_{\tau \sim \pi_t(.|x)}[\exp(r(x,\tau)/\beta)]$. Akin to DPO, to eliminate the partition function, we can consider the *relative reward* between two completions to the same prompt x:

$$r(x,\tau) - r(x,\tau') = \beta \left(\ln \frac{\pi_{t+1}(\tau|x)}{\pi_t(\tau|x)} - \ln \frac{\pi_{t+1}(\tau'|x)}{\pi_t(\tau'|x)} \right). \tag{23.9}$$

For a small, tabular problem, we could attempt to solve the above equation exactly. Unfortunately, this is infeasible at the scale of language models. Thus, we will minimize the squared difference between the LHS and RHS of the above expression, giving us the following:

$$\pi_{t+1} = \arg\min_{\pi} \mathbb{E}_{x,\tau,\tau' \sim \pi_t(.|x)} \left[\underbrace{\beta \left(\ln \frac{\pi(\tau|x)}{\pi_t(\tau|x)} - \ln \frac{\pi(\tau'|x)}{\pi_t(\tau'|x)} \right)}_{\text{Regressor}} - \underbrace{(r(x,\tau) - r(x,\tau'))}_{\text{Relative reward}} \right]^2, \tag{23.10}$$

where τ, τ' are sampled independently from the latest policy $\pi_t(\cdot|x)$. [2] We call this the **REBEL** (REgression to RElative REward Based RL) update. Observe that if we were able

[2] One can instead compare the current policy to samples from some *baseline* distribution, as discussed in the full paper. This could be suboptimal data from the internet or SFT data.

to perfectly minimize this objective to zero and π_t has full support for all x, we would recover the exact mirror descent update. One can still prove strong guarantees with *approximate* minimization via reduction to supervised learning, as is discussed further in the paper.

23.4 Differences between REBEL, DPO, and PPO

It is worth pausing for a moment to consider the differences between the three RLHF algorithms we've discussed in this course.

1. **DPO:** Uses finite, offline preference data. Can't take advantage of reward models or generation-verification gaps. Simple, supervised learning-based update.
2. **PPO:** Can use an arbitrary RM and optimize on unseen prompts. Complex update, requiring attention to detail in terms of implementation (e.g. clipping, baselines, GAE) and having four separate models in memory.
3. **REBEL:** Achieves the best of both worlds: simple, regression based update akin to DPO. Doesn't require critic network like PPO. Can take advantage of arbitrary RMs.

Intuitively, one can think of REBEL as DPO but with a reward model (that may or may not be learned from human preference data).

23.5 Connecting REBEL to Other Algorithms

To recap, REBEL minimizes the following least squares loss problem at each iteration:

$$\ell_t(\theta) = \mathbb{E}_{x,\tau,\tau' \sim \pi_{\theta_t}(.|x)} \left[\beta \left(\ln \frac{\pi_\theta(\tau|x)}{\pi_{\theta_t}(\tau|x)} - \ln \frac{\pi_\theta(\tau'|x)}{\pi_{\theta_t}(\tau'|x)} \right) - (r(x,\tau) - r(x,\tau')) \right]^2 \tag{23.11}$$

Let us consider what happens when we *approximately* minimize the above loss function.

23.5.1 One Step of Gradient Descent

Consider taking a single step of gradient descent in the above objective, i.e.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta l_t|_{\theta_t} \tag{23.12}$$

For some fixed (x, τ, τ'), the gradient at $\theta = \theta_t$ is

$$\nabla_\theta l(\theta_t) = \beta \left(\nabla_\theta \ln \pi_{\theta_t}(\tau|x) - \nabla_\theta \ln \pi_{\theta_t}(\tau'|x)\right)(r(x,\tau) - r(x,\tau')). \tag{23.13}$$

Observe that this recovers the REINFORCE Leave One Out (RLOO) update (i.e. REINFORCE, but using the reward of the other sample in the batch as a baseline).

23.5.2 One Step of Gauss-Newton

To take a Gauss-Newton (GN) step, we first approximate the non-linear part inside the square via a first-order Taylor expansion at θ_t:

$$\ln \frac{\pi_\theta(\tau|x)}{\pi_{\theta_t}(\tau|x)} - \ln \frac{\pi_\theta(\tau'|x)}{\pi_{\theta_t}(\tau'|x)} \approx \left(\nabla \ln \pi_{\theta_t}(\tau|x) - \nabla \ln \pi_{\theta_t}(\tau'|x)\right)^T (\theta - \theta_t). \tag{23.14}$$

We then plug this linear approximation back into the least squares problem:

$$\theta_{t+1} = \arg\min_\theta \mathbb{E}_{x,(\tau,\tau')\sim\pi_{\theta_t}(\cdot|x)} \left[\left(\left(\nabla \ln \pi_{\theta_t}(\tau|x) - \nabla \ln \pi_{\theta_t}(\tau'|x)\right)^T (\theta - \theta_t) - (r(x,\tau) - r(x,\tau'))\right)^2\right]$$

This is precisely the "regression view" of the Natural Policy Gradient (NPG) we discussed many lectures ago! Thus, NPG can be seen as an approximation of the REBEL update.

23.5.3 Other Domains

REBEL can be applied to other RL problems, such as continuous control or fine-tuning image diffusion models. While we assumed deterministic dynamics in the above derivation, if we expand the probability of a trajectory as

$$\pi(\zeta|s_0) = \prod_{h=0}^{H-1} \pi(a_h|s_h) P(s_{h+1}|s_h, a_h), \tag{23.15}$$

observe that

$$\ln \frac{\pi(\zeta|s_0)}{\pi_{\text{old}}(\zeta|s_0)} = \sum_{h=0}^{H-1} \ln \frac{\pi(a_h|s_h)}{\pi_{\text{old}}(a_h|s_h)}, \tag{23.16}$$

i.e. that the dynamics cancel out as they are evaluated along the same trajectory. This could be useful in a multi-turn dialog setting, where the person's responses to what the agent says are stochastic. For such a problem, each action may more naturally correspond to an entire response, rather than a token. See the follow-up paper, REFUEL, for more information.

The REBEL algorithm can also be adapted to use a critic Q (rather than a reward model), which leads to a similar update to Soft Actor Critic (SAC):

$$\min_{\pi} \mathbb{E}_{s,a\sim\pi_t(\cdot|s)} \left[\left(\beta \left(\ln \frac{\pi(a|s)}{\pi_t(a|s)} - \ln \frac{\pi(a'|s)}{\pi_t(a'|s)} \right) - (Q(s,a) - Q(s,a')) \right)^2 \right], \tag{23.17}$$

where a and a' are sampled independently from the current policy $\pi_t(\cdot|s)$. This can help for longer horizon problems for which more precise credit assignment is required. If one doesn't want to explicitly train a critic, one can instead simply roll out the policy from the state s after taking some action a and observe the sum of rewards accumulated over the rest of the episode. This is an unbiased, Monte-Carlo estimate of the desired $Q^\pi(s,a)$ value. Recall that these sorts of "resets" are free in text problems as they are just generating from a prefix.