TEXAS
The University of Texas at Austin

DIRECT PREFERENCE OPTIMIZATION

2026F INF385T: Post-Training of LLMs | Week 9

CHAN YOUNG PARK
Assistant Professor, iSchool, School of Computing, The University of Texas at Austin

Announcements

- **HW2** will be released this Friday
- Next week's class (Oct 28)
  - **Guest lecture by Hamish Ivison (UW)**
    "Post-training open agents for long-horizon tasks"
  - **Fully remote (zoom only)** / no in-person attendance required
    Zoom link will be posted on Canvas before the class
  - After the lecture, HW2 walkthrough → Group Project sprint
    - You can choose to meet in-person or virtually for the sprint
    - Teams will submit mid-checkpoint slides at the end of class

Project logistics for the remaining classes

- Oct 28: Project sprint (submit project checkpoint slides)
- Nov 4: Project checkpoint presentation
- Nov 18: Project Presentation
- Dec 7: Final report due

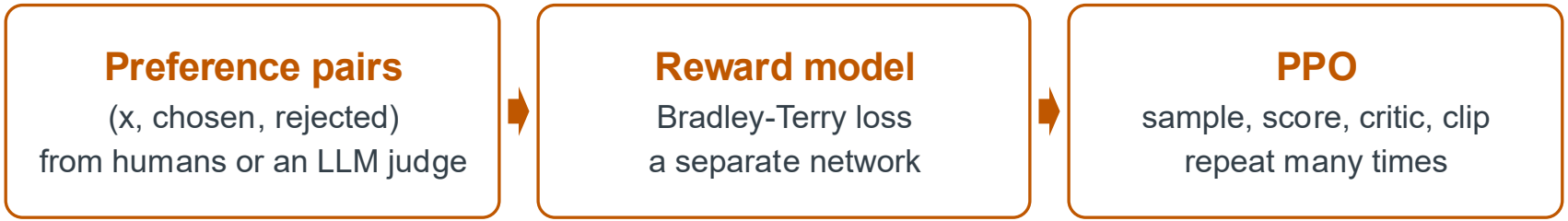
Recap

Week 7: which behaviors receive demonstrations? → examples

Week 8: what if we can only compare? → Reward model, PPO

Week 9: can we skip the reward model and RL? → DPO

Last week: the full RLHF pipeline



- Four models in memory (policy, critic, reference, reward)
- PPO on-policy generation at every step (rollout gen → score → advantage → update)
- Many knobs: β, clip ε, critic hyper-paremeters, reward normalization

Can we go straight from preference pairs to a better policy?

Week 9 overview

- **The DPO trick**
  How can preference pairs train the policy directly? (Rafailov et al., 2023)
- **Inside the update**
  What does the gradient push, and what can go wrong? (Razin et al., 2025)
- **In practice**
  Offline vs. on-policy pairs; does DPO vs. PPO matter? (Lambert et al., 2024; Ivison et al., 2024)

Can we skip the reward model?

THE DPO TRICK

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

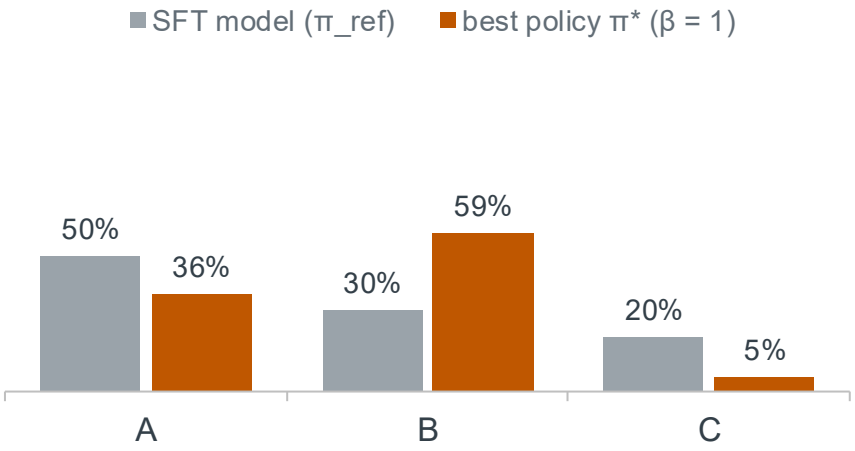
Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Recall: the best policy has a closed form

$$\pi^*(y \mid x) = (1 / Z(x)) \cdot \pi_{ref}(y \mid x) \cdot \exp(r(x, y) / \beta)$$



- Same toy world as last week: scores A = 1, B = 2, C = 0
- Z(x) sums over every possible response → can't compute → we needed RL

Last week this was a dead end. Today it is the starting point

- **Today: solve the same equation for r instead**

Flip it: a reward hides inside every policy

$$r(x, y) = \beta \cdot \log(\pi^*(y \mid x) / \pi_{ref}(y \mid x)) + \beta \cdot \log Z(x)$$

Response	π_ref	π* (β = 1)	log(π* / π_ref)	+ log Z	= r
A	0.50	0.36	-0.33	+1.33	1.0
B	0.30	0.59	+0.67	+1.33	2.0
C	0.20	0.05	-1.33	+1.33	0.0

Log-ratios alone already give the right gaps (B − A = 1, A − C = 1)
log Z just shifts all three

Reward = how much more π* likes y than π_ref does

Plug it into Bradley-Terry

We only observe pairs
Week 8 linked pairs to rewards via Bradley-Terry

$$P(y_w \succ y_l \mid x) = \sigma(\, r(x, y_w) - r(x, y_l)\,)$$

$$= \sigma(\ \beta \log \pi^*(y_w)/\pi_{ref}(y_w) + \beta \log Z(x)$$
$$- \beta \log \pi^*(y_l)/\pi_{ref}(y_l) - \beta \log Z(x)\ )$$

$\beta \log Z(x) - \beta \log Z(x) = 0$

The uncomputable term disappears because both responses share the same prompt

Week 8: only differences matter
(each prompt has a free zero point)

The DPO loss

$$L_{DPO}(\theta) = -\log \sigma\left(\beta \log \pi\theta(y_w|x)/\pi_{ref}(y_w|x) - \beta \log \pi\theta(y_l|x)/\pi_{ref}(y_l|x)\right)$$

implicit reward of chosen
$r_{DPO}(y_w)$

implicit reward of rejected
$r_{DPO}(y_l)$

	Score inside $\sigma(\cdot - \cdot)$
Reward model (Week 8)	$r_\varphi(x, y)$: a separate network's output
DPO (today)	$\beta \log \pi\theta(y|x) / \pi_{ref}(y|x)$: the policy's own log-ratio

Treat $\pi\theta$ as a candidate for π^*
pick θ that makes the observed preferences most likely (= minimize the BT NLL)

Your language model is secretly a reward model

$$r_{DPO}(x, y) = \beta \cdot \log(\pi\theta(y \mid x) / \pi_{ref}(y \mid x))$$

Rank any pair
higher r_DPO = the trained policy favors it more than the SFT model does

Evaluate pair ranking
Does it rank held-out pairs correctly?

No extra network
just two forward passes (policy and reference)

However, does not check generated-answer quality

Lab: compare implicit DPO reward and RM rankings
evaluate generations with an API judge

What DPO removes

Policy $\pi\theta$	Value $V\psi$	Reference π_{ref}	Reward $r\varphi$
player	commentator	anchor	judge
trained	removed	frozen	removed

- No sampling during training: fixed (x, chosen, rejected) pairs, like SFT
- The reference's log-probs never change, so they can be computed once before training

One model to train and one frozen copy → training looks like SFT

DPO in code

```
pi_w = seq_logprob(policy, x, y_w)      # sum of log-probs of response tokens
pi_l = seq_logprob(policy, x, y_l)
with torch.no_grad():                   # reference is frozen, set to eval()
    ref_w = seq_logprob(ref, x, y_w)
    ref_l = seq_logprob(ref, x, y_l)

r_w = beta * (pi_w - ref_w)             # implicit reward, chosen
r_l = beta * (pi_l - ref_l)             # implicit reward, rejected
loss = -F.logsigmoid(r_w - r_l).mean()  # Week 8 BT loss, new score
```

A few lines on top of an SFT setup
(you will write this in the lab)

What does the gradient push?

INSIDE THE DPO UPDATE

Worked example (β = 0.1)

1. Chosen: log πθ = −40, log πref = −42 → +2
2. Rejected: log πθ = −45, log πref = −44 → −1
3. **Implicit rewards: 0.2 and −0.1**
4. Gap: 0.3 → σ(gap) ≈ 0.57
5. Loss: −log σ(0.3) ≈ 0.55
6. **$|\partial L/\partial \text{gap}| = 1 - \sigma \approx 0.43$**

multiplied by β, log-prob weight ≈ 0.043

Log-probs are sums over all response tokens, so they are large negative numbers

Only the change vs. the reference matters

Same numbers as RM training once you read β × log-ratios as scores

The gradient: same shape as RM training

$$L = -\log(\sigma(gap)),\ gap = \beta\,(r(w) - r(l)) \rightarrow \frac{dL}{d\theta} = \frac{dL}{dgap} * \frac{dgap}{d\theta}$$

$$\nabla L = -(1 - \sigma(gap)) \cdot \beta \cdot [\nabla \log \pi\theta(y_w|x) - \nabla \log \pi\theta(y_l|x)]$$

Weight
$\beta[1 - \sigma(gap)]$: largest for confidently misranked pairs

Push up
the chosen response's log-likelihood

Push down
the rejected response's log-likelihood

Confidently correct pairs receive small weights
barely correct pairs still move the policy

SFT vs. REINFORCE vs. DPO

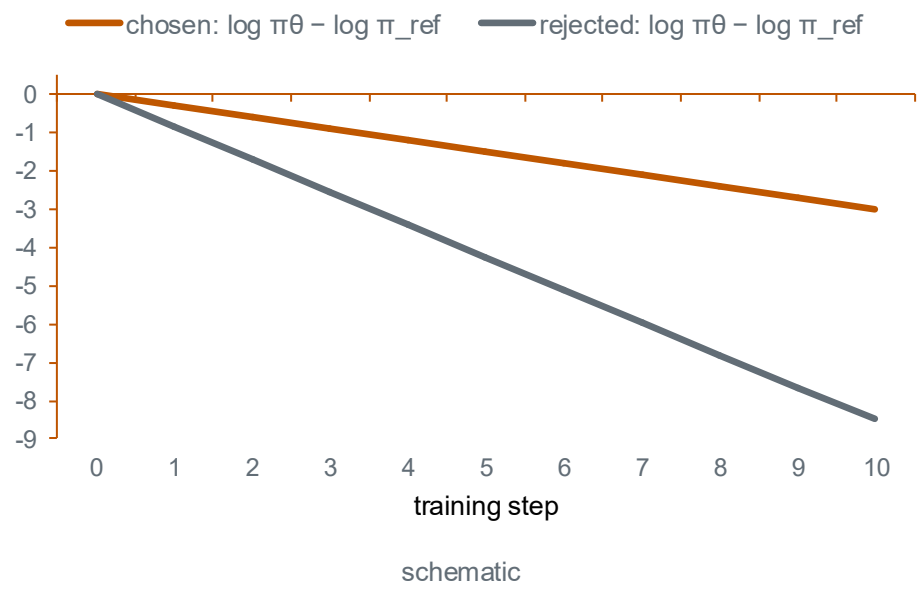
Method	Which text	Weight on $\nabla \log \pi_\theta$
SFT (Week 5)	dataset answer y*	always +1
REINFORCE / PPO (Week 8)	the model's own samples	advantage (score − baseline)
DPO (today)	**fixed dataset pair**	**+w on chosen, −w on rejected**

On fixed data, DPO = weighted SFT on chosen + weighted unlearning on rejected

$w = \beta[1 - \sigma(\text{gap})]$ (w range: (0, 1))

β = leash: smaller β → the policy must move further from π_ref to fit the same preferences

Watch out: chosen can get less likely



schematic

- The loss only needs the gap to grow: both likelihoods can fall if rejected falls faster
- Shared parameters couple the two updates; a positive chosen term does not guarantee a net increase
- Razin et al.: Training to refuse lowered the refusal rate: 74.4% → 33.4%
- Mitigations include
  - model-aware pair filtering (filter out too similar pairs)
  - Adding an SFT term on chosen $L = L_DPO - \alpha \cdot \log \pi\theta(yw \mid x)$

Lab: check the four fixed pairs for likelihood displacement

Razin et al. (2025). Unintentional Unalignment: Likelihood Displacement in Direct Preference Optimization (ICLR)

Length is an easy way to win the margin

- Preference data often favors longer answers
- DPO then learns “longer = better” and outputs grow
- Fixes
  - Score fix: length-normalized log-probs (Tülu 3 (Lambert et al., 2024), SimPO (Meng et al., 2024))
  - Data fix: use length-balanced pairs
- Olmo 3: length control during DPO gave shorter outputs **and** a stronger model after the later RL stage

Check response length every time you compare a DPO model with its SFT start

Lab: same data, sum vs. mean scores. Does normalization change how much answers grow?

Where do the pairs come from?

DPO IN PRACTICE

Unpacking DPO and PPO: Disentangling Best Practices for Learning from Preference Feedback

Hamish Ivison[♣♠] Yizhong Wang[♣♠] Jiacheng Liu[♣♠]
Zeqiu Wu[♠] Valentina Pyatkin[♣♠] Nathan Lambert[♣]
Noah A. Smith[♣♠] Yejin Choi[♣♠] Hannaneh Hajishirzi[♣♠]

[♣]Allen Institute for AI [♠]University of Washington
hamishiv@cs.washington.edu

Response source vs. data refresh

Response source
- Other models: e.g., UltraFeedback
- Starting policy: sample once, then label

Data refresh
- Offline: keep a fixed preference file
- Online / iterative: sample and label again as the policy changes

Potential third axis: label source (human or AI)

+Policy-sourced responses can still be used in offline DPO

What makes good preference pairs?

Contrast

Olmo 3 finds that high contrast between chosen and rejected is critical for DPO gains

Delta, not perfection

weak chosen vs. weaker rejected still teaches (delta learning from week 7)

Judge + pool

Tülu 3: many models generate; GPT-4o picks chosen and rejected

= Seek meaningful quality contrast

a judge probability or confidence is not that contrast

DPO vs. PPO: does the algorithm matter?

One SFT model (Tülu 2 13B), change one component at a time, 11 benchmarks

Data first. PPO helps when you can afford it

1. **Preference data**
   14 datasets, up to +8% (instruction following, truthfulness); per-aspect labels and diverse prompts matter more than size or generator strength
2. **Algorithm**
   PPO > DPO by up to 2.5% (math), 1.2% (general)
3. **Reward model**
   bigger RM helps math, little else
4. **On-policy prompts**
   unlabeled prompts for PPO

Ivison et al. (2024). Unpacking DPO and PPO: Disentangling Best Practices for Learning from Preference Feedback (NeurIPS, 2024)

What DPO removed and what it kept

Removed
explicit training RM
critic
rollouts during each update
actor–critic training loop

Kept
the reference leash (β)
Bradley–Terry assumptions
dependence on data quality

Remaining risks
data coverage and label errors
likelihood displacement
length shortcuts

A simpler training loop still needs data checks and separate generation evaluation

Conclusion

- Reparameterize reward with a policy/reference log-ratio → Z cancels
- Fit πθ to observed preferences using the Bradley-Terry likelihood
- The weighted chosen/rejected gradient terms interact through shared parameters
- Track chosen likelihood, margins, and length (lower loss is not proof of better answers)
- held-out pair ranking != API-judged generation quality

Train RM and DPO

LAB 8

Lab 8

Time	Part
0:00–0:10	setup
0:10–0:25	Task 1. Train a scalar reward model. read the BT update
0:25–0:35	Task 2. Implement and check DPO loss
0:35–1:05	Task 3. Train DPO twice (sum vs. length-normalized)
1:05–1:20	Task 4a. Held-out rankings (RM, DPO, length)
1:20–1:30	Task 4b. Generated answer quality (LLM judge)
1:30–1:40	Interpret and submit the form