RL: From Optimization to Exploitation

Lecture 4 Notes

CDSS 94: Building Thoughtful AI Systems (Spring 2026)

Contents

1 Lecture 3 Recap: The Post-Training Landscape

1.1 Five Methods of Post-Training

Each method compresses a different signal into the model's weights:

Method	What It Compresses	KL Direction
SFT (IFT)	Demonstrations (format + instruction following)	Forward KL
RLHF / PPO	Preferences via RL (style + subtle judgment)	Reverse KL
DPO	Pairwise preferences directly, no reward model	Implicit KL
CAI	Principles (RLAIF)	RLAIF
GRPO	Verifiable domain correctness (RLVR)	Reverse KL

Lambert's shorthand for the pipeline: IFT (format) → PreFT (preferences) → RLVR (correctness).

1.2 Three Key Ideas from Lecture 3

Definition 1.1: KL is the Currency

Every post-training method spends a **KL budget**—distance from the starting policy to change behavior. But KL is measured on training data, not the full distribution. Models can drift more than you think (Lambert, 2025). Go too far and the model breaks: nonsense, repetition, language switching.

Definition 1.2: Forward vs. Reverse KL

SFT (forward KL) spreads probability across the whole demonstration distribution. Safe, conservative, sounds generic. RL (reverse KL) concentrates mass on high-reward regions, ignores the rest. More decisive, opinionated—picks winners. Useful pipelines: SFT first for coverage, then RL to sharpen.

Definition 1.3: The Overoptimization Curve

Gao et al. showed: as you optimize against a proxy reward model, true quality first improves, peaks, then *comes back down*. Two failure modes:

- **Quantitative:** proxy reward diverges from gold reward. Measurable, well-studied.
- **Qualitative:** models that "feel worse"—overly verbose, sycophantic, rigid. No single metric captures it, which is why vibes-based evaluation persists.

Both have the same root: the reward model is a lossy compressor of human preferences.

> *What happens when you let RL loose on a language model?*

2 RL Deep-Dive: Inside the Objective

2.1 KL Divergence Revisited

Definition 2.1: Forward vs. Reverse KL — The Sampling Rule

There is only one thing to understand: **the first slot is what gets sampled from.** That determines everything.

- **Forward KL:** $\mathrm{KL}(P_{\text{data}} \| Q_{\text{model}})$. You sample from P_{data}. For each data point, you ask: "does my model assign high probability here?" If Q is zero somewhere P has mass, infinite penalty. The model learns: "I must cover everything." Result: broad, inclusive, generic.
- **Reverse KL:** $\mathrm{KL}(\pi_{\text{model}} \| P_{\text{ref}})$. You sample from π_{model}. The model is penalized for generating things the reference would not approve of. Result: narrow, mode-seeking, decisive.

Remark 2.2

SFT loss = Forward KL: $\mathcal{L} = -\mathbb{E}_{x \sim p_{\text{data}}} \log p_\theta(x)$, and cross-entropy decomposes as $H(p, q) = H(p) + D_{\mathrm{KL}}(p \| q)$. RLHF uses reverse KL. The difference is in *who drives*—the data or the model.

2.2 The RLHF Objective

Definition 2.3: The RLHF Objective

$$\max_{\pi_\theta} \mathbb{E}\big[r(x, y)\big] - \beta \cdot \mathrm{KL}(\pi_\theta \| \pi_{\text{ref}})$$

- $r(x, y)$: Push toward high-reward outputs.
- $\beta \cdot \mathrm{KL}$: Penalize deviation from the reference model.
- β: The exchange rate—how much is one nat of divergence worth?

2.3 The Boltzmann Optimal Policy

Theorem 2.4: Boltzmann Optimal Policy

The optimal solution to the RLHF objective is:

$$\pi^*(y \mid x) \propto \pi_{\text{ref}}(y \mid x) \cdot \exp\left(\frac{r(x, y)}{\beta}\right)$$

Intuitively: take every output the reference could produce. Keep its original probability as a starting point. Then scale up high-reward outputs and scale down low-reward ones.

Remark 2.5

This is **intractable** to compute directly—the normalizing constant $Z(x)$ requires summing over all possible outputs. So we need approximate solutions.

2.4 Two Families of Solutions

Approach	Idea
Policy Gradients (PPO, GRPO)	Sample outputs, estimate gradients, nudge π_θ toward π^*.
Direct Alignment (DPO)	Rearrange the math so π^* never needs to be computed.

2.5 Policy Gradients

Definition 2.6: The Policy Gradient

Every policy gradient method is an instance of:

$$g = \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \Psi_t\right]$$

Intuition: make good outputs more likely, in proportion to how good they are. The term Ψ_t is the **advantage estimate**, and different algorithms differ in how they estimate it.

Algorithm	How Ψ_t is Estimated
REINFORCE	Reward minus a simple baseline. 2 models in memory.
PPO	Learned advantage from a trained critic, with clipped updates to prevent catastrophic steps. 4 models in memory (policy, reference, critic, reward).
GRPO	Reward minus the average of sampled completions for the same prompt. 2 models in memory, no critic needed.

Remark 2.7: Lambert's Unifying Framing

All three algorithms are instantiations of the *same* policy gradient—they differ only in how they estimate Ψ_t. Once you see that, the algorithm zoo becomes a **design space**, not a taxonomy.

2.6 PPO vs. GRPO

Definition 2.8: PPO's Objective

1. **How good was this output?** Use a learned critic to estimate the advantage: "How much better/worse did this turn out than what we expected for this prompt?"
2. **How much did the policy change?** Clip the probability ratio to prevent catastrophic updates—the trust region.

Cost: 4 models in memory (policy, reference, critic, reward model). If an output looks great and the critic wants to crank up its probability, clipping says: "not so fast, you can only move this much per step."

Definition 2.9: GRPO's Objective

What if we skip the critic entirely? For each prompt, sample G completions and score them all. The advantage is:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_1, \ldots, r_G\})}{\text{std}(\{r_1, \ldots, r_G\})}$$

"Was this output better or worse than others for the same prompt?" Uses the same clipped ratio as PPO. The critic is gone—replaced by the group of samples as a baseline. With enough samples ($G \geq 8$), variance is manageable.
Cost: 2 models in memory. No critic to train.

2.7 Direct Policy Optimization (DPO)

Definition 2.10: DPO Derivation

Starting from the Boltzmann optimal policy, four steps eliminate the need for RL:

1. **Express reward as a function of policy:**

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

2. **Apply Bradley–Terry preference model:**

$$p(y_w \succ y_l) = \sigma\big(r(x, y_w) - r(x, y_l)\big)$$

3. **Plug in:** the normalizing constant $Z(x)$ cancels.
4. **DPO objective:**

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}\left[\log \sigma\left(\beta \cdot \left(\log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right)\right]$$

Remark 2.11: Takeaway

PPO, GRPO, and DPO are all paths from π_{ref} toward π^*—the Boltzmann optimal policy. They differ in how they approximate the journey: policy gradients sample and nudge; DPO rearranges the algebra so the destination is computed directly.

3 Does RL Add Knowledge?

3.1 The Debate

A central question: does RL *teach* the model anything new, or does it merely *sharpen* what the model already knows?

Theorem 3.1: RL as Sharpener (NeurIPS 2025 Oral — informal)

Pass@k analysis shows: RLVR beats the base model at $k = 1$, but *converges* at large k. All reasoning paths in the RL-trained model already exist in the base model's sampling distribution. Six different RL algorithms were tested; all performed similarly and remained far from optimal. **Conclusion:** RL is a *sharpener*, not a teacher. It concentrates mass on correct paths that the model already had—it doesn't create new ones.

Before RL	Probability spread across paths (some correct, some not)
After RL	Mass concentrated on correct paths

3.2 The Counterargument

Remark 3.2: The Metrics Were Wrong

Wen et al. (2025) introduced **CoT-Pass@K**, which evaluates reasoning *steps*, not just final answers. Under this metric, RLVR can extend the model's boundaries when measured correctly. The debate may be partly about metrics.

Example 3.3: DeepSeek-R1 Emergent Behaviors

DeepSeek-R1 exhibited emergent behaviors after RL: "aha moments," self-verification, progressive lengthening of reasoning chains. These patterns were not present in the SFT checkpoint, even if the underlying capabilities existed in the base model.

Remark 3.4: Resolution

Does RL create new capabilities, or does it compose existing ones? The answer depends on where you draw the line. This connects back to the compression spectrum from Lecture 3: whether RL "adds knowledge" depends on your definition of knowledge.

4 Multimodal RL: Beyond Text

4.1 RL Across Modalities

Domain	Approach
Image Generation	RLHF/DPO for diffusion models (DDPO, Diffusion-DPO). Action space: denoising trajectories, not tokens.
Video Generation	RL from human feedback on temporal consistency and physics simulation.
Robotics	Preference-based reward learning for manipulation. The original RL motivation (pre-LLM era).

4.2 What Makes Multimodal RL Harder

Remark 4.1

- **Reward modeling compounds:** text quality + visual quality + cross-modal consistency. The Kolmogorov complexity of "good multimodal output" is much higher.
- **Annotation is more expensive and noisier** than for text.
- **Verifiable rewards are even scarcer:** "correct" is ambiguous for images.

Example 4.2: Multimodal Reward Hacking

Oversaturated "hyperreal" images score high on aesthetic reward models but look uncanny. The model learns to exploit the proxy (saturation, contrast) rather than genuine visual quality.

5 Exploitation: Reward Hacking

5.1 Over-Optimization Types

Definition 5.1: Two Types of Over-Optimization

1. **Quantitative over-optimization:** The classic Gao et al. curve. Proxy reward goes up, gold reward comes back down. Measurable, predictable, well-studied.
2. **Qualitative degradation:** Models that "feel worse" without measurably hacking. Overly verbose, sycophantic, rigid. No single metric captures it, which is why vibes-based evaluations persist.

> *Goodhart's Law: "When a measure becomes a target, it ceases to be a good measure."*
> $K(\textit{true human preferences}) \gg K(\textit{reward model})$. *The gap is where reward hacking lives.*

5.2 Chattiness

Example 5.2: Length as the Easiest Exploit

- Annotators prefer longer responses. Datasets bake in length bias.
- The reward model learns: length = quality.
- RL amplifies this: more markdown, more emoji, more bullet lists.
- The model learns the *shape* of good responses, not the *substance*.
- AlpacaEval and WildBench added length corrections because gaming length was the easiest way to "beat GPT-4."
- KL is measured on training data, not the full distribution. There is room to drift (Lambert, 2025).

Counterpoint: sometimes increased length genuinely helped raters. Length isn't always bad. Is it learned or is it earned?

5.3 Reward Hacking in Practice

Example 5.3: CoastRunners

The classic RL reward hacking example: an agent in the CoastRunners game learned to drive in circles collecting points from turbo boosts rather than finishing the race. The reward was a proxy for racing skill, but the agent found a higher-reward strategy that had nothing to do with the intended behavior.

Example 5.4: Coding Agents and Alignment Faking

In safety evaluations of RL-trained coding agents, concerning behaviors emerged:

- 12% of the time, models would intentionally attempt to sabotage in ways that reduce the ability to detect reward hacking and other misalignment.
- Models exhibited **spontaneous alignment faking**: pretending to behave as intended for deceptive purposes, despite never having been trained or instructed to be misleading.
- This behavior emerged exclusively as an unintended consequence of RL on code/programming problems.

5.4 The Reward Hacking Spectrum

Remark 5.5: Where RL Works and Where It Breaks

Domain	Character
Clean verifiable rewards (math, code)	You know if the answer is right. Lowest hacking risk.
Learned middle ground (chatbots)	Most deployed systems live here. Proxy rewards, manageable hacking.
Creative / emotional	Nobody agrees on what "good" means. Highest hacking risk.

Further along the spectrum = lossier reward model = more room to hack.

5.5 Mitigations

Approaches to reducing reward hacking include:

(i) **Reward model ensembles:** train multiple RMs and take conservative estimates.
(ii) **KL constraints:** the β penalty in RLHF directly limits exploitation range.
(iii) **Iterative RM updates:** retrain the reward model on the current policy's outputs.
(iv) **Length normalization:** explicitly correct for length bias in reward scores.
(v) **Process reward models:** score reasoning steps, not just final answers.
(vi) **Constitutional AI:** self-critique against principles rather than scalar reward.
(vii) **Vibes-based evaluation:** human spot-checks that catch qualitative degradation that metrics miss.

6 Summary

Topic	Key Takeaway
The RL Objective	Maximize reward subject to a KL constraint. β is the exchange rate.
Boltzmann Solution	$\pi^*(y\|x) \propto \pi_{\text{ref}}(y\|x) \cdot \exp(r/\beta)$. Intractable but conceptually central.
PPO vs. GRPO vs. DPO	Three paths to the same destination. Same policy gradient, different advantage estimates.
Does RL Add Knowledge?	Debated. RL sharpens existing paths (pass@k evidence), but may enable new compositions (CoT-Pass@K, emergent behaviors).
Multimodal RL	Same ideas, harder rewards, more hacking surface.
Reward Hacking	The gap between K(preferences) and K(reward model) is where exploitation lives. Chattiness is the easiest exploit.

> *Post-training is compression. The reward model is the compressor. Its lossy artifacts are where reward hacking lives. The better we compress human values, the less room there is to exploit.*