Preference Learning, Policy Optimization, and RLHF/GRPO

Mathematics of Deep Learning — Chapter 13

Yin Yang

Foundation Books Project

Roadmap

From Tokens to Preferred Behavior
Language Models as Policies
Supervised Fine-Tuning
Preference Data
Reward Models and Bradley–Terry
KL-Regularized Policy Optimization
The RLHF Pipeline
PPO for Language-Model Post-Training
Direct Preference Optimization
GRPO
Failure Modes
Evaluation
Summary

From Tokens to Preferred Behavior

Why Post-Training?

Pretraining fits a base model to predict the next token from a large corpus. That objective teaches grammar, facts, and style — but not whether a complete answer is *useful*.

- A response can have low perplexity yet mislead the user.
- Behavioral targets (instruction following, factuality, refusal quality) live at the level of complete responses, not tokens.
- Post-training adds feedback on whole responses while keeping the language ability learned during pretraining.

Perplexity is not a direct measure of usefulness.

Running Example

Prompt x: *Explain why dividing by zero is undefined to a middle-school learner.*

- y^+: "no number times zero gives a nonzero numerator."
- y^-: "dividing by zero is just infinity."

The preference $y^+ \succ y^-$ holds because y^+ is more careful about the constraint — not because it is longer or more fluent. The tuple (x, y^+, y^-) reappears as: preference data, reward-model data, and DPO data. Later, a *group* of sampled responses to the same x powers GRPO.

One tuple carries the whole chapter.

Post-Training Map



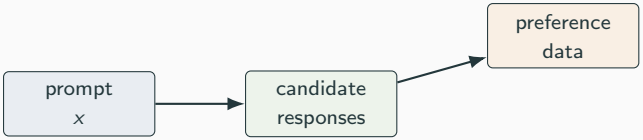
(1) A prompt is given.

Post-Training Map



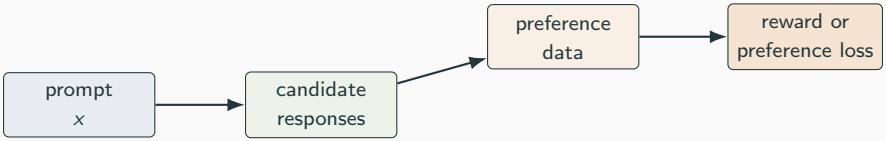
(1) A prompt is given.
(2) Several candidate responses are sampled.

Post-Training Map



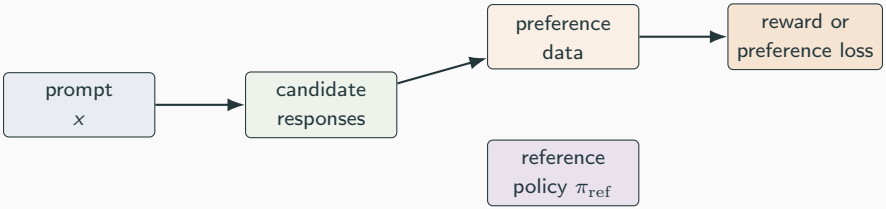
(1) A prompt is given.
(2) Several candidate responses are sampled.
(3) Comparisons label preferred vs. rejected.

Post-Training Map



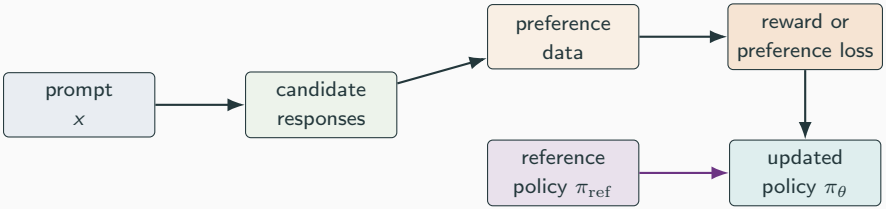
(1) A prompt is given.
(2) Several candidate responses are sampled.
(3) Comparisons label preferred vs. rejected.
(4) A scalar reward or preference loss is built.

Post-Training Map



(1) A prompt is given.
(2) Several candidate responses are sampled.
(3) Comparisons label preferred vs. rejected.
(4) A scalar reward or preference loss is built.
(5) A reference policy is frozen as the drift baseline.

Post-Training Map



(1) A prompt is given.
(2) Several candidate responses are sampled.
(3) Comparisons label preferred vs. rejected.
(4) A scalar reward or preference loss is built.
(5) A reference policy is frozen as the drift baseline.
(6) Update π_θ pulled by reward, restrained by π_{ref}.

Reward pulls; the reference policy restrains.

Language Models as Policies

Response Policy

Treat the language model as a policy over complete responses:

$$\pi_\theta(y \mid x) = \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t}), \qquad y = (y_1, \ldots, y_T).$$

Same autoregressive factorization from Chapter 11 — new applied role.

- State at time t: $s_t = (x, y_{<t})$. Action: token y_t.
- Whole response y is one sampled object for rating.
- Log-probability $\log \pi_\theta(y \mid x) = \sum_t \log \pi_\theta(y_t \mid x, y_{<t})$ is easier to compare.

Three conditional tokens with probs 0.50, 0.40, 0.25 give $\pi_\theta(y \mid x) = 0.05$ and $\log \pi_\theta(y \mid x) \approx -2.996$.

A rater scores the whole response; the math should too.

Reference Policy

Definition
A reference policy $\pi_{\text{ref}}(y \mid x)$ is a fixed response distribution used as the comparison point during preference optimization.

- Often the SFT checkpoint, sometimes the base model.
- Its job is *not* to label data — it says what the updated model should stay close to.
- Protects broad language quality while reward pulls the policy toward higher preference.

The reference model is a restraint, not a labeler.

Supervised Fine-Tuning

SFT as Imitation

Demonstration dataset:

$$\mathcal{D}_{\mathrm{SFT}} = \{(x_i, y_i^\star)\}_{i=1}^{n}.$$

SFT loss is response-level negative log-likelihood:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log \pi_\theta(y_i^\star \mid x_i).$$

Example: two demos with $\pi_\theta(y_1^\star \mid x_1) = 0.20$, $\pi_\theta(y_2^\star \mid x_2) = 0.50$ give $-\frac{1}{2}(\log 0.20 + \log 0.50) \approx 1.151$.

- Same NLL as pretraining — different data source and purpose.
- Teaches *format* before preference methods refine tradeoffs.
- The SFT checkpoint is a natural reference policy.

SFT teaches format; preference methods then refine tradeoffs.

Preference Data

Comparisons Instead of Absolute Scores

Many response qualities are easier to compare than to score. A rater struggles to say "reward 7.3" but can say "A is better than B for this prompt."

Pairwise preference dataset

$$\mathcal{D}_{\text{pref}} = \{(x_i, y_i^+, y_i^-)\}_{i=1}^n.$$

A ranking $a \succ b \succ c$ yields three pairs $(a, b), (a, c), (b, c)$, or two adjacent pairs $(a, b), (b, c)$.

- Labels are noisy: raters disagree; rubrics may favor style over truth.
- Prompt mix, candidate sampling, and rater instructions become part of the mathematical setup.

Preference quality *is* data quality.

Reward Models and Bradley–Terry

Reward Model

A reward model maps prompt-response pairs to scalars:

$$r_\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}.$$

Bradley–Terry preference probability:

$$\mathbb{P}_\phi(y^+ \succ y^- \mid x) = \sigma\big(r_\phi(x, y^+) - r_\phi(x, y^-)\big), \qquad \sigma(u) = \tfrac{1}{1+e^{-u}}.$$

Reward-model loss:

$$\mathcal{L}_{\mathrm{RM}}(\phi) = -\frac{1}{n}\sum_{i=1}^{n} \log \sigma\big(r_\phi(x_i, y_i^+) - r_\phi(x_i, y_i^-)\big).$$

Example: $r_\phi(x, y^+) = 2$, $r_\phi(x, y^-) = 0$ gives $\sigma(2) \approx 0.881$.

Pairwise reward models learn *differences*, not absolute scores.

The Constant Cancels

Adding the same constant to both rewards does not change the Bradley–Terry probability:

$$\sigma\big((r_+ + c) - (r_- + c)\big) = \sigma(r_+ - r_-).$$

Consequences:

- The absolute zero of a reward model is not identified by pairwise comparisons.
- Only the score *gap* drives the Bradley–Terry likelihood.
- When r_ϕ is later used for policy optimization, reward scale and normalization become diagnostics, not cosmetics.
- Reward models can overfit; easy proxies (length, confidence) get learned and exploited.

Reward scale is unidentified; downstream normalization matters.

KL-Regularized Policy Optimization

The KL-Regularized Objective

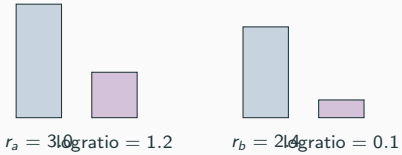
Fix a prompt distribution, reward model r_ϕ, reference policy π_{ref}, and penalty $\beta > 0$:

$$J(\theta) = \mathbb{E}_x \, \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \left[r_\phi(x, y) - \beta \, \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \right].$$

- Reward term: pull toward responses with higher r_ϕ.
- Penalty term: response-level log-ratio against π_{ref}.
- Large β: strong restraint, small reward influence.
- Small β: reward dominates, drift risk rises.

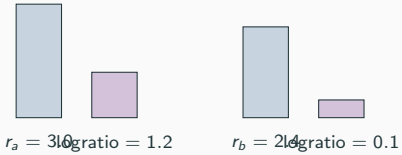
The reward-vs-KL tradeoff is the central post-training knob.

Reward vs. KL: A Numerical View



(1) Two responses: higher reward but more drift, vs. less reward and less drift.

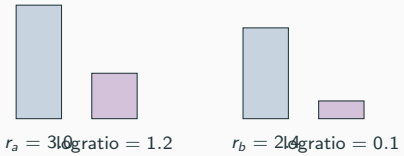
Reward vs. KL: A Numerical View



$\beta = 0.5$:
$a : 3.0 - 0.6 = 2.40$
$b : 2.4 - 0.05 = 2.35$
a wins

(1) Two responses: higher reward but more drift, vs. less reward and less drift.
(2) Modest β: reward dominates, a wins.

Reward vs. KL: A Numerical View



$\beta = 0.5$:
$a : 3.0 - 0.6 = 2.40$
$b : 2.4 - 0.05 = 2.35$
***a* wins**

$\beta = 1.0$:
$a : 3.0 - 1.2 = 1.80$
$b : 2.4 - 0.1 = 2.30$
***b* wins**

(1) Two responses: higher reward but more drift, vs. less reward and less drift.
(2) Modest β: reward dominates, *a* wins.
(3) Larger β: drift penalty flips the ranking.

Same data, different β, different optimum.

The Closed-Form Tilted Reference

For one prompt, finite response set, and $\pi_{\mathrm{ref}}(y) > 0$ on its support, the maximizer of

$$\sum_y \pi(y) r(y) - \beta \sum_y \pi(y) \log \frac{\pi(y)}{\pi_{\mathrm{ref}}(y)}$$

is

$$\pi^\star(y) = \frac{\pi_{\mathrm{ref}}(y) \exp(r(y)/\beta)}{\sum_{y'} \pi_{\mathrm{ref}}(y') \exp(r(y')/\beta)}.$$

Lagrange multiplier λ on $\sum_y \pi(y) = 1$; setting $\partial / \partial \pi(y) = 0$ gives

$\pi(y) = C\, \pi_{\mathrm{ref}}(y) \exp(r(y)/\beta)$; normalize for C.

Reward *tilts* the reference distribution; β controls the sharpness.

The RLHF Pipeline

RLHF as a Workflow



(1) Start from a base language model.

RLHF as a Workflow



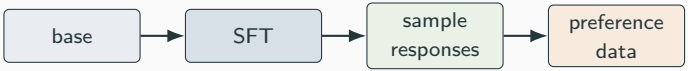
(1) Start from a base language model.
(2) Supervised fine-tune on demonstrations.

RLHF as a Workflow



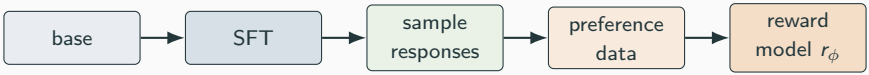
(1) Start from a base language model.
(2) Supervised fine-tune on demonstrations.
(3) Sample several responses per prompt.

RLHF as a Workflow



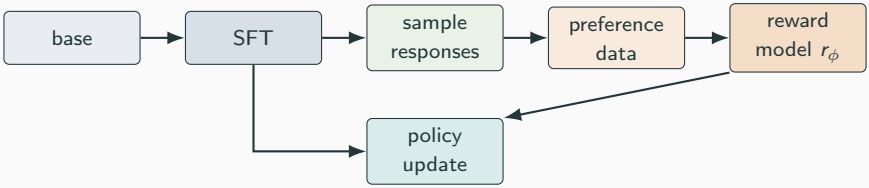
(1) Start from a base language model.
(2) Supervised fine-tune on demonstrations.
(3) Sample several responses per prompt.
(4) Collect human or rubric comparisons.

RLHF as a Workflow



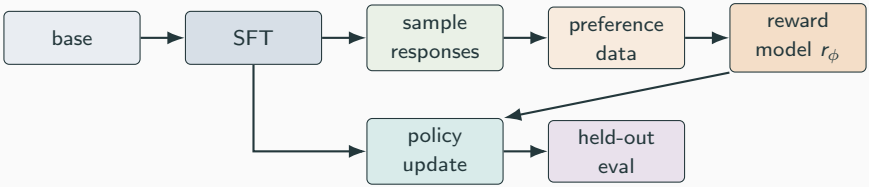
(1) Start from a base language model.
(2) Supervised fine-tune on demonstrations.
(3) Sample several responses per prompt.
(4) Collect human or rubric comparisons.
(5) Train a reward model r_ϕ from comparisons.

RLHF as a Workflow



(1) Start from a base language model.
(2) Supervised fine-tune on demonstrations.
(3) Sample several responses per prompt.
(4) Collect human or rubric comparisons.
(5) Train a reward model r_ϕ from comparisons.
(6) Update policy: reward $-$ KL drift from SFT.

RLHF as a Workflow



(1) Start from a base language model.
(2) Supervised fine-tune on demonstrations.
(3) Sample several responses per prompt.
(4) Collect human or rubric comparisons.
(5) Train a reward model r_ϕ from comparisons.
(6) Update policy: reward $-$ KL drift from SFT.
(7) Evaluate on held-out prompts and targeted checks.

Each arrow can fail. RLHF quality is pipeline quality.

RLHF vs. Verifier-Driven Training

Same policy-optimization machinery, different reward sources.

Use case	Reward/advantage signal	Main diagnostic risk
RLHF / alignment	Human comparisons; reward model fit to preferences	Reward hacking, style bias, rater mismatch
Verifier-driven reasoning	Answer checker, tests, proof or rubric score	Sparse rewards, shortcut exploitation, brittle checking

Win rate of A vs. B: fraction of comparison prompts on which A's response is preferred. Small held-out sets carry sampling uncertainty.

The same math covers human-preference and verifier-driven post-training.

PPO for Language-Model Post-Training

Token-Level PPO Ratio

State at time t: $s_t = (x, y_{<t})$. Action: token y_t. Old policy = snapshot that produced the sampled response. Token probability ratio:

$$\rho_t(\theta) = \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}.$$

Clipped PPO surrogate applied to these per-token decisions (or to response-level ratios in variants).

- Advantage $\widehat{A}_t$ tells the optimizer whether y_t should be made more or less likely than the baseline.
- Built from a response reward, a value baseline, or a group-relative baseline.

PPO recycles sampled rollouts; clip width controls stability.

The Credit-Assignment Problem

A reward model usually scores the *whole* response, but PPO attaches signals to *token* decisions.

- Need a way to turn a response-level scalar into per-token $\widehat{A}_t$: learned value baseline, reward normalization, or relative baselines.
- KL penalty/monitor plays a separate role: limit drift from π_{ref} (the behavior we should not destroy).
- Interacting knobs: rollout budget, value estimator, KL coefficient, clip width, reward normalization.

PPO stabilizes updates; it does not validate the reward.

Direct Preference Optimization

The DPO Margin

Skip the explicit reward model and RL rollouts. Use π_θ and π_{ref} directly on each pair.

DPO log-ratio margin

$$\Delta_\theta = \log \frac{\pi_\theta(y^+ \mid x)}{\pi_{\mathrm{ref}}(y^+ \mid x)} - \log \frac{\pi_\theta(y^- \mid x)}{\pi_{\mathrm{ref}}(y^- \mid x)}.$$

DPO loss:

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \log \sigma(\beta\, \Delta_{\theta,i}).$$

DPO compares two policy-reference log-ratios, not absolute likelihoods.

From KL-Regularized Optimum to DPO

Take logs of the closed-form tilted reference:

$$\log \frac{\pi^\star(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} = \frac{r(y)}{\beta} - \log Z(x).$$

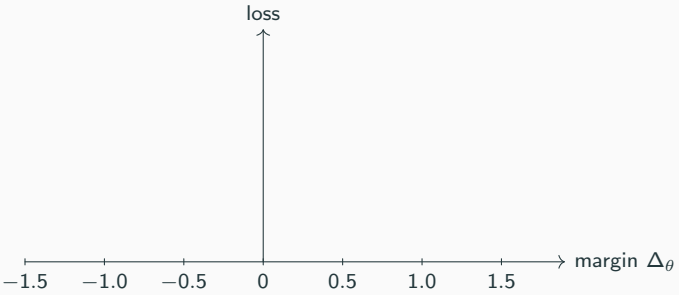
Subtract y^+ from y^- — the prompt normalizer $\log Z(x)$ cancels:

$$r(y^+) - r(y^-) = \beta \left[\log \frac{\pi^\star(y^+ \mid x)}{\pi_{\mathrm{ref}}(y^+ \mid x)} - \log \frac{\pi^\star(y^- \mid x)}{\pi_{\mathrm{ref}}(y^- \mid x)} \right].$$

Replace unavailable $\pi^\star$ with trainable π_θ: reward difference $\iff \beta \Delta_\theta$.

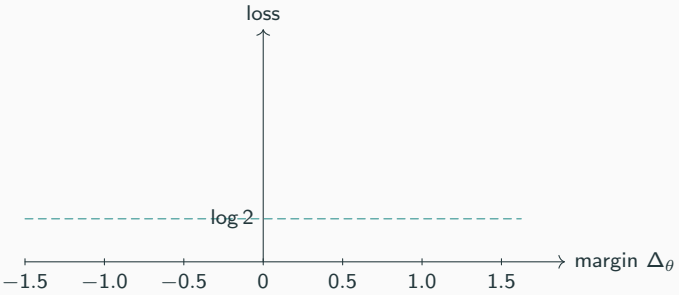
The DPO loss *is* a Bradley–Terry loss in disguise — with the policy ratio as the implicit reward.

DPO Loss Shape



(1) Axes: margin on x, loss on y.

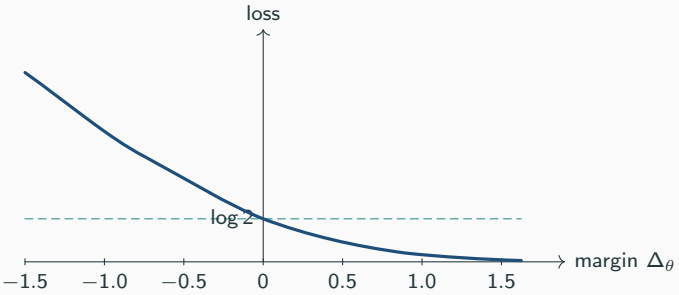
DPO Loss Shape



(1) Axes: margin on x, loss on y.

(2) Reference level $\log 2$ at $\Delta_\theta = 0$.

DPO Loss Shape

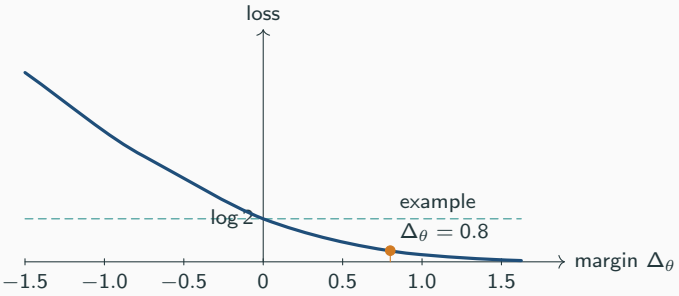


(1) Axes: margin on x, loss on y.

(2) Reference level $\log 2$ at $\Delta_\theta = 0$.

(3) Loss decreases smoothly as Δ_θ grows.

DPO Loss Shape



(1) Axes: margin on x, loss on y.
(2) Reference level $\log 2$ at $\Delta_\theta = 0$.
(3) Loss decreases smoothly as Δ_θ grows.
(4) Example: $\Delta_\theta = 0.8$, $\beta = 2$, loss ≈ 0.184.

Positive margin = preferred response has more relative lift than rejected.

Reading a DPO Example

	$\log \pi_\theta(\cdot \mid x)$	$\log \pi_{\text{ref}}(\cdot \mid x)$
y^+	-4.0	-5.0
y^-	-3.0	-3.2

Margin: $\Delta_\theta = (-4.0 + 5.0) - (-3.0 + 3.2) = 1.0 - 0.2 = 0.8$.

With $\beta = 2$, the per-pair loss is $-\log \sigma(1.6) \approx 0.184$.

- If π_θ raises y^+ and y^- by the same log amount, Δ_θ is unchanged.
- DPO rewards *relative* lift for y^+ over y^-, not absolute likelihood increases.
- Still depends on π_{ref}, preference data, and β.

Simpler pipeline than PPO — not free of modeling assumptions.

GRPO

Group-Relative Advantages

GRPO needs only several samples per prompt and a way to score them — reward model, verifier, unit test, or rubric.

Group-relative advantage
For a group $G(x) = \{y_1, \ldots, y_K\}$ with rewards $r_1, \ldots, r_K$,

$$\bar{r}_G = \frac{1}{K}\sum_j r_j, \quad s_G = \sqrt{\frac{1}{K}\sum_j (r_j - \bar{r}_G)^2}, \quad \widehat{A}_j = \frac{r_j - \bar{r}_G}{s_G + \epsilon}.$$

Example: $(1, 1, 0, -1) \Rightarrow \bar{r}_G = 0.25$; centered rewards $(0.75, 0.75, -0.25, -1.25)$. Centered values sum to zero by construction.

Group baselines compare answers to the same prompt — no separate critic needed.

Three Policies, Three Roles

Policy	Role	Question it answers
π_θ	current	What is being updated?
π_{old}	sampling	Where did this response come from?
π_{ref}	reference	How far has behavior drifted?

- Probability ratio $\pi_\theta/\pi_{\text{old}}$: importance correction for reusing samples.
- Reference penalty $\log(\pi_\theta/\pi_{\text{ref}})$: drift restraint.
- Reading the two as the same object is a common bug.

Sampling policy and reference policy answer different questions.

The GRPO Surrogate

For one prompt x, sampled $y_1, \ldots, y_K$, advantages $\widehat{A}_j$:

$$J_{\mathrm{GRPO}}(\theta; x) = \frac{1}{K} \sum_{j=1}^{K} \left[\rho_j(\theta) \widehat{A}_j - \beta \log \frac{\pi_\theta(y_j \mid x)}{\pi_{\mathrm{ref}}(y_j \mid x)} \right],$$

with $\rho_j(\theta) = \pi_\theta(y_j \mid x) / \pi_{\mathrm{old}}(y_j \mid x)$. A clipped variant uses the PPO-style minimum.

One-response check: $\widehat{A}_j = 2$, $\pi_\theta = 0.12$, $\pi_{\mathrm{old}} = 0.10$, $\pi_{\mathrm{ref}} = 0.08$, $\beta = 0.1$:

$$\rho_j = 1.2 \Rightarrow \rho_j \widehat{A}_j = 2.4, \quad \beta \log(0.12/0.08) \approx 0.041.$$

Positive advantage pushes up; reference term pushes back because π_θ already exceeds π_{ref}.

No separate critic: the group supplies a prompt-specific baseline.

Failure Modes

Reward Hacking and Overoptimization

Definitions

- *Reward hacking*: policy gets high measured reward via behavior that does not satisfy the intended criterion.
- *Overoptimization*: more reward optimization makes the intended evaluation criterion *worse*.
- *Distribution shift*: training prompts/responses differ from evaluation/deployment.

In the division-by-zero example: a reward model that learns "longer is better" produces long answers even when concise wins. "Confident wording" becomes overconfident false claims.

Optimizers follow the proxy you give them — audit it as a proxy.

Symptom → Likely Cause → Diagnostic

Symptom	Likely cause	Diagnostic
Answers get longer	Reward rewards length	Track length by prompt type
Reward up, win rate flat	Reward overoptimization	Hold out preference prompts
Fluent wrong answers rise	Style proxy beats correctness	Targeted factual checks
Refusals get excessive	Safety reward dominates	Split helpfulness vs. refusal
KL spikes	Policy drifts from π_{ref}	Monitor token and response KL

Watch reward, KL, length, win rate, task success, calibration, refusal behavior, and qualitative cases *together*.

A single scalar trains; a portfolio decides whether the model is actually better.

Evaluation

What to Measure After PO

- **Held-out comparisons**: closest to the training signal.
- **Rubric scores**: split correctness, clarity, harmfulness.
- **Task metrics**: math correctness, code tests, resolution rate — when the task admits automatic checks.
- **Model-graded eval**: scales but inherits judge errors.
- **Human eval**: more direct, slower, rater variation.

Compare the final model to π_{ref} on matched prompts — this is what *the update* changed.

Evaluation should match the behavior claim.

Win-Rate Uncertainty

Model wins 65 of 100 held-out comparisons. Observed win rate 0.65. Simple standard error:

$$\mathrm{SE} \approx \sqrt{\frac{0.65\,(1-0.65)}{100}} \approx 0.048.$$

- A ± 1 SE band of $\approx [0.60, 0.70]$ is wide.
- Even "clear" win-rate gains need many prompts to defend.
- Reward up but win rate stalled? Training objective has diverged from intended behavior.

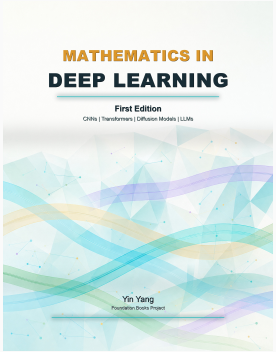
Small evaluation sets carry real sampling uncertainty.

Summary

Chapter in One Slide

- **Policy view:** treat the LM as $\pi_\theta(y \mid x)$; post-training reshapes that distribution.
- **SFT:** response-level NLL on demonstrations — teaches format; gives a natural reference checkpoint.
- **Preferences:** comparisons, not absolute scores. Reward model fits Bradley–Terry; only the score gap is identified.
- **KL-regularized PO:** reward pulls, π_{ref} restrains; β is the central knob. Optimum tilts the reference.
- **RLHF / PPO:** clipped surrogate over token decisions; credit assignment from response-level reward.
- **DPO:** preference-pair loss from policy-vs-reference log-ratios; reward model implicit.
- **GRPO:** within-prompt group baseline, no separate critic; three policies (θ, old, ref) with three roles.
- **Failure & eval:** reward hacking and overoptimization are mathematical consequences — audit with a portfolio.

Next: deployment and inference — serving, decoding, and what changes when the trained policy meets real prompts.



Mathematics in Deep Learning

Chapter overview slides for the book by Yin Yang.

Book page	`foundation-books.github.io/mathematics-in-deep-learning`
Code	`github.com/foundation-books/mathematics-in-deep-learning-code`
Paperback	`amazon.com/dp/B0H1ZZHDT2`
Hardcover	`amazon.com/dp/B0H1ZL11MJ`
Kindle	`amazon.com/dp/B0GX32M8SP`
License	CC BY-NC-ND 4.0

These free overview slides may be shared non-commercially with attribution and without modifications. Full explanations, exercises, and companion-code context are in the book and linked repository.