Post-training I

Lecture 11: learning from demonstrations and feedback

Efficient Generative Language Models

MS&E 319
Stanford University
Autumn 2026



The course so far

Component	Purpose
Architecture	Defines the model's computation
Pretraining	Learns language and knowledge from text
Inference	Generates responses efficiently

What should post-training add?



GPT-3 and InstructGPT

"Explain the moon landing to a 6 year old in a few sentences."

GPT-3 excerpt

"Explain the theory of gravity to a 6 year old."

InstructGPT excerpt

"People went to the moon, and they took pictures of what they saw…"

InstructGPT answers the request.

OpenAI (2022), InstructGPT announcement



What did pretraining optimize?

$$\mathcal{L}(\theta) = -\sum_t \log p_\theta(x_t \mid x_{<t})$$

Predict the continuation of observed text.

A plausible continuation may not fulfill the request.

Why might GPT-3 continue with another question?

Brown et al. (2020), GPT-3 · Ouyang et al. (2022), InstructGPT



Can prompting close the gap?

Illustrative prompt

A science teacher gives clear explanations to children.

Child: How did people land on the Moon?

Teacher:

Context can guide the continuation.

Brown et al. (2020) · Figure 1.2



In-context learning

Demonstrations in the prompt

c#a!t $\rightarrow$ cat
d@o#g $\rightarrow$ dog
b!i@r#d $\rightarrow$ bird

New query

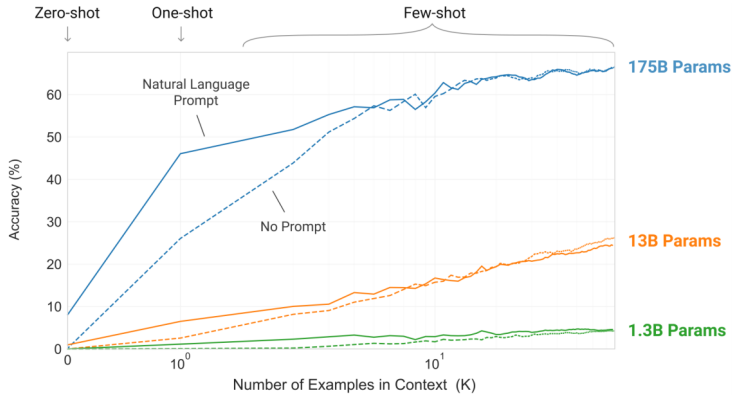
h%o!r@s#e $\rightarrow$ **horse**

Illustrative completion. Model weights stay fixed.

Brown et al. (2020) · Figure 1.2



Few-shot prompting in GPT-3



Recover words by removing symbols. Weights stay fixed.
K = prompt examples. Solid: with instruction. Dashed: without.

Brown et al. (2020) · Figure 1.2

Post-training versus prompting

Compare two 175B models on user prompts:

GPT-3 with task examples in the prompt

InstructGPT, trained on *demonstrations* and *human feedback*

Evaluators preferred InstructGPT in $71 \pm 4\%$ of comparisons (95% confidence interval).

Ouyang et al. (2022), InstructGPT



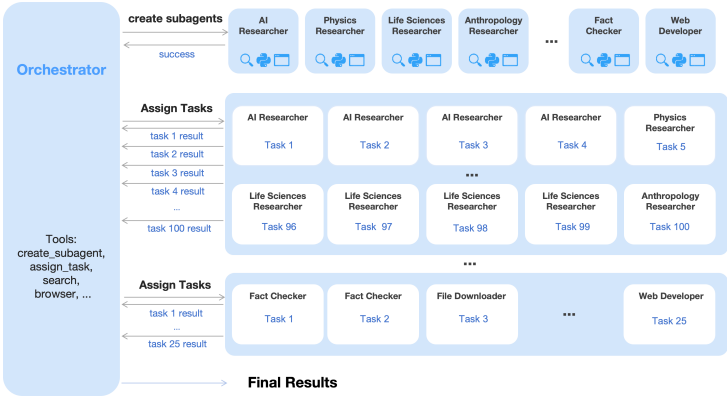
Post-training beyond conversation

Target	Examples
Assistant behavior	Useful, truthful, safe responses
Problem solving	Math, code, reasoning, checking
Agentic capability	Tool use, delegation, coordination

Ouyang et al. (2022), InstructGPT · DeepSeek-AI (2025), DeepSeek-R1 · Kimi Team (2026), Kimi K2.5



Agent coordination in Kimi K2.5



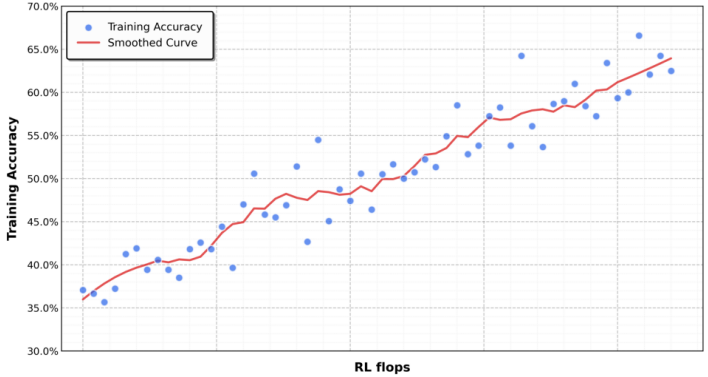
The coordinator delegates subtasks and combines the results.

Kimi Team (2026), Kimi K2.5

Coordinator training accuracy

Train the coordinator. Keep helper models fixed.

Horizontal axis: RL training compute.

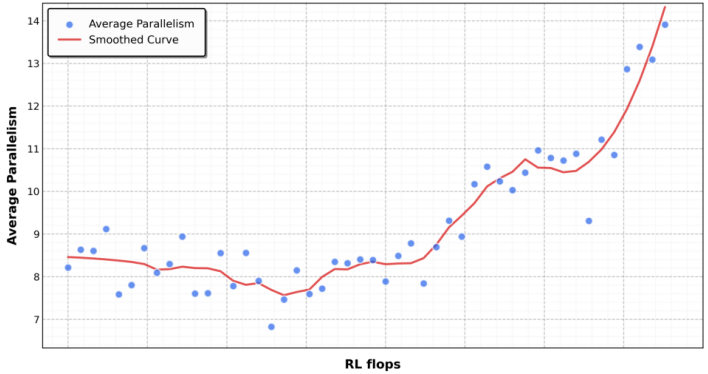


Kimi Team (2026) · Figure 4

Parallel work during training

Average parallelism measures concurrent agent work.

Horizontal axis: RL training compute.



Kimi Team (2026) · Figure 4

Rewarding useful coordination

Suppose we reward only the number of agents launched.

What behavior might that encourage?

More agents need not mean better results.

Kimi rewards task performance and completed subtasks.

Auxiliary reward weights decay to zero.

Kimi Team (2026) · Figure 4



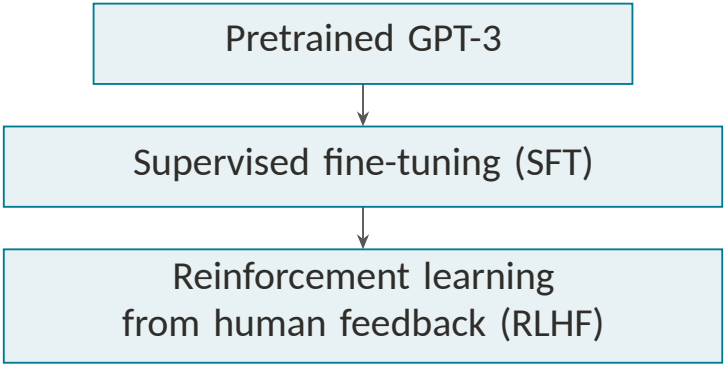
Post-training signals

Available information	A way to learn from it
Demonstrations	Supervised fine-tuning (SFT)
Response comparisons	Preference learning
Task outcomes	Reinforcement learning (RL)
Teacher-model outputs	Learn from a teacher (distillation)

Ouyang et al. (2022), InstructGPT · Rafailov et al. (2023), DPO · Agarwal et al. (2024), on-policy distillation



Example recipe: InstructGPT



Evaluate response quality and retained capabilities.

Ouyang et al. (2022), InstructGPT

Post-training in this course

Learning from demonstrations (SFT)

Efficient adaptation (LoRA)

Learning from human feedback (RLHF)

Later: direct preferences and distillation



Supervised fine-tuning

Data: prompts paired with desired responses.

Goal: learn to produce this behavior on new prompts.

Start from a pretrained model.
Train it to predict demonstrated response tokens.

Ouyang et al. (2022), InstructGPT



The SFT objective

For one prompt x and demonstrated response y:

$$\mathcal{L}(\theta; x, y) = -\sum_{t=1}^{T} \log p_\theta(y_t \mid x, y_{<t})$$

Predict each response token using its demonstrated prefix.

Average these losses across the training data.

Ouyang et al. (2022), InstructGPT



An SFT example

Prompt

Where did Apollo 11 land?

Demonstrated response

On the Moon <end>

Toy tokenization: four response tokens.

Ouyang et al. (2022), InstructGPT



Which predictions enter the loss?

Prompt x: Where did Apollo 11 land?

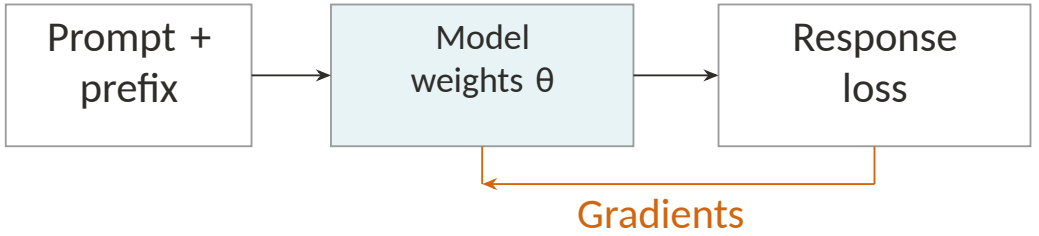
Available context	Prediction target	Mask
Earlier prompt tokens	Next prompt token	0
x	On	1
x + On	the	1
x + On the	Moon	1
x + On the Moon	<end>	1

Loss mask: 0 = exclude this target; 1 = include it.



Loss masking and gradient flow

Mask 0 removes the prompt's own prediction loss.



Yes. Response loss can update parameters that process the prompt.

Lambert et al. (2024), Tülu 3

SFT with tool calls

Schematic tool-call syntax

Role	Recorded message	Loss?
User	Mean of [2, 4, 9]?	No
Assistant	calculator((2+4+9)/3)	Yes
Tool	5	No
Assistant	The mean is 5.	Yes

Here: train assistant outputs; observe tool results.



Pretraining and SFT

	Pretraining	SFT
Data	Broad text	Demonstrations
Target	Next token	Next token
Loss mask	Often all tokens	Often assistant tokens
Emphasis	Broad capability	Desired behavior



Building the demonstration dataset

Dataset decision	Effect on training
Example sources	Behaviors available to imitate
Filter and verify	Errors that enter training
Choose the mixture	Exposure to tasks and difficulty
Separate evaluation	Whether gains generalize

Hold out evaluation examples and check for exact and near-duplicate overlap with training data.

Lambert et al. (2024), Tülu 3



Weighting short and long responses

Two demonstrations: 10 tokens and 1,000 tokens.

Weights on each response's average token loss:

Average over	Short response	Long response
All 1,010 tokens	≈1%	≈99%
Two response means	50%	50%

Normalization changes the influence of long answers.

Lambert et al. (2024), Tülu 3



Demonstrations during training

Simplified schedules

Strategy	Use of training data
Staged	General text first, then demonstrations
Mixed	General text and demonstrations in the same training phase

Mixed pretraining can still be followed by a separate SFT stage.

Cheng et al. (2024), Instruction Pre-Training



Instructions during pretraining

Instruction Pre-Training (Cheng et al., 2024)

Source passage (classroom illustration)

Aster Observatory opens at 18:00.
Its last entry is at 22:00.

Generated instruction and response

Q: When is the observatory's last entry?
A: 22:00.

Pretrain on text enriched with generated examples.
Further SFT also helped in the paper's experiments.

Cheng et al. (2024), Instruction Pre-Training



Loss versus response quality

InstructGPT's SFT experiments

Validation loss rose after about one epoch.

More training improved human preference ratings.

Check response quality when selecting a checkpoint.

Ouyang et al. (2022), InstructGPT



How do SFT updates work?

Start from pretrained weights.
Full fine-tuning makes all weights trainable.

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{SFT}}(\theta)$$

The same backpropagation and optimizer machinery applies.

What are the costs and risks of updating all weights?

Ouyang et al. (2022), InstructGPT



Preserving pretrained capabilities

InstructGPT study: 175B models, zero-shot evaluation.

Benchmark	GPT-3	SFT
HellaSwag accuracy	78.1%	75.3%
SQuADv2 F1	64.30	57.67

Evaluate retained skills as well as the target behavior.

Ouyang et al. (2022), InstructGPT



Learning rate and duration

GPT-3 and InstructGPT, both with 175B parameters.

Training stage	Reported learning rate
GPT-3 pretraining peak	6.0×10^{-5}
InstructGPT SFT initial	5.03×10^{-6}

Tune rate and duration. A small rate alone cannot ensure retention.

Brown et al. (2020), GPT-3 · Ouyang et al. (2022), InstructGPT



Post-training in this course

Learning from demonstrations (SFT)

Efficient adaptation (LoRA)

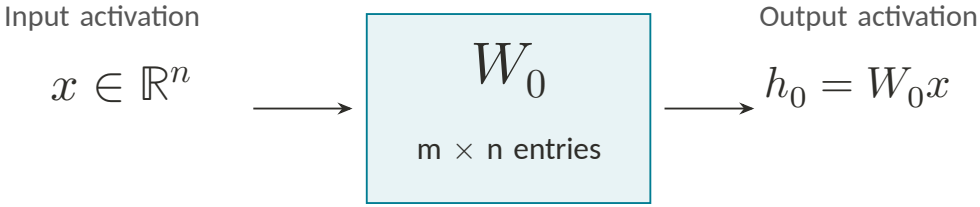
Learning from human feedback (RLHF)

Later: direct preferences and distillation



Adapting one weight matrix

x is the activation vector entering one projection layer.



Full fine-tuning allows every entry of this matrix to change.

Hu et al. (2021), LoRA, Sections 4 and 7

The cost of a full update

$$W = W_0 + \Delta W$$

ΔW records the accumulated change to all $m \times n$ entries.

For this matrix	Stored scalar values
Trainable weights	mn
Weight gradients	mn
Adam first + second moments	2mn

Parameter state: $O(mn)$. Dense training work: $O(Tmn)$.

T = total token positions processed in the training step.

Can useful changes live in a smaller space?

Hu et al. (2021), LoRA, Sections 4 and 7



What low rank means

Rank counts independent output directions of a linear map.

One direction

$$\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$$

Second column = 2 $\times$ first column

Two directions

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

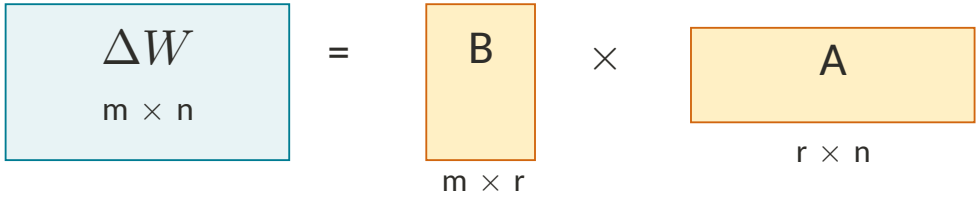
Neither column is a multiple of the other



LoRA learns a factored change

Low-rank adaptation (LoRA)

$$W = W_0 + BA$$



Freeze W_0. Train only A and B.

The change $\Delta W = BA$ has rank at most r.

Hu et al. (2021), LoRA, Sections 4 and 7



Rank one: one output direction

Input x. Base output $h_0 = W_0 x$. Adapted output h.

$$\Delta h = h - h_0 = b(a^\top x)$$

$a^\top x$ is one scalar: how strongly this input activates a learned feature.

b turns that scalar into a change along one learned output direction.

Hu et al. (2021), LoRA, Sections 4 and 7



A rank-one example

$$a = (1, -2)^\top, \qquad b = (1, 2)^\top$$

Input x	Coefficient $a^\top x$	Output change Δh
(1, 0)	1	(1, 2)
(0, 1)	−2	(−2, −4)

Can a rank-one update produce (1,0) for one input and (0,1) for another?



Combining r output directions

$$\Delta h = BAx = \sum_{j=1}^{r} b_j (a_j^\top x)$$

Row $a_j^\top$ of A computes coefficient $a_j^\top x$.

Column b_j of B is the corresponding output direction.

Larger r allows a richer change and trains more parameters.

Hu et al. (2021), LoRA, Sections 4 and 7



Gradients for the factors

Backpropagation gives G, the loss gradient for W.

$$G = \frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial (\Delta W)}$$

$$\mathrm{d}(\Delta W) = (\mathrm{d}B)A + B(\mathrm{d}A)$$

$$\nabla_A \mathcal{L} = B^\top G, \qquad \nabla_B \mathcal{L} = G A^\top$$

$$A \leftarrow A - \eta \nabla_A \mathcal{L}, \qquad B \leftarrow B - \eta \nabla_B \mathcal{L}$$

Hu et al. (2021), LoRA, Sections 4 and 7 · Thinking Machines Lab (2025), LoRA Without Regret



Initializing the factors

Standard initialization: A random, $B = 0$.

$BA = 0$: the initial forward pass matches the base.

Random A lets B receive a gradient first.

$$\nabla_A \mathcal{L} = B^\top G, \qquad \nabla_B \mathcal{L} = G A^\top$$

Both zero would give zero gradients.
After B changes, A can receive gradients too.
Hu et al. (2021), LoRA, Sections 4 and 7



Trainable parameter counts

$W_0 : \ m \times n \qquad B : \ m \times r \qquad A : \ r \times n$

Trainable array	Number of entries
Full matrix W	mn
Factors B and A	mr + rn = r(m + n)

m = n = 4096, r = 8: 16,777,216 versus 65,536

256× fewer trainable entries for this matrix.

Hu et al. (2021), LoRA, Sections 4 and 7



Computing the LoRA branch

$$h = W_0 x + B(Ax)$$

First $z = Ax$, then $\Delta h = Bz$.

Added branch: $O(rn + mr)$ work per token.

The base multiplication still costs $O(mn)$.

No dense BA is needed during training.

Hu et al. (2021), LoRA, Sections 4 and 7



Merging LoRA for inference

After training, fix A and B.

$$W_{\text{merged}} = W_0 + BA$$

Then $h = W_{\text{merged}} x$.

One dense multiplication, with no extra LoRA branch.

Hu et al. (2021), LoRA, Sections 4 and 7



What LoRA saves

Reduced by LoRA	Still required
Trainable weight gradients	Frozen base weights
Adam states for A and B	Base forward computation
Task-specific saved weights	Activations and gradient flow

Actual memory and speed depend on the full training system.

QLoRA also stores the frozen base in 4-bit precision.

Hu et al. (2021), LoRA, Sections 4 and 7 · Dettmers et al. (2023), QLoRA, Section 3



Choosing the LoRA rank

GPT-3 175B on WikiSQL: question + table schema $\rightarrow$ SQL

Metric: SQL logical-form validation accuracy. LoRA on Q and V.

Full fine-tuning: 73.8%. Typical seed variation: $\approx$ 0.5 points.

Rank r	Trainable parameters relative to rank 1	Logical-form accuracy
1	1$\times$	73.4%
2	2$\times$	73.3%
4	4$\times$	73.7%
8	8$\times$	73.8%
64	64$\times$	73.5%

Here, raising the rank gives no clear accuracy gain.

Hu et al. (2021), LoRA, Sections 4 and 7 · Thinking Machines Lab (2025), LoRA Without Regret



Post-training in this course

Learning from demonstrations (SFT)

Efficient adaptation (LoRA)

Learning from human feedback (RLHF)

Later: direct preferences and distillation



Getting feedback more easily

Human demonstrations require complete responses.

Long or specialized answers can be costly to produce.

Ask people to compare proposed answers.

Christiano et al. (2017) · Ouyang et al. (2022), InstructGPT



Comparing two answers

Explain the Moon landing to a six-year-old.

A: Apollo 11 used lunar-orbit rendezvous, with separate command and lunar modules.

B: A rocket took astronauts to the Moon. Two landed in a small spacecraft, walked around, and then came home.

Which answer better fits the requested audience?



Why use comparisons?

A numerical rating needs a shared scale.

Illustrative rater	Score for A	Score for B
Rater 1	4 / 10	7 / 10
Rater 2	2 / 10	5 / 10

Both judgments say: B fits this request better.

A shared rubric is still necessary.

Christiano et al. (2017) · Ouyang et al. (2022), InstructGPT



Learning from comparisons

Demonstration	Comparison
Prompt + target answer	Prompt + two answers + choice
One response to imitate	An ordering to learn from

How do relative judgments improve a generator?

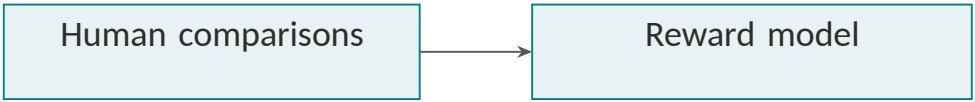
Ouyang et al. (2022), InstructGPT · Rafailov et al. (2023), DPO



RLHF with a reward model

Reinforcement learning from human feedback

1. Learn a reward model from comparisons.



2. Use its scores to improve the assistant.

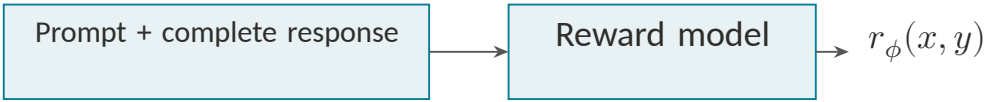
Ouyang et al. (2022), InstructGPT



Why learn a reward model?

A reward model assigns a learned scalar score to a response.

x: prompt. y: complete response. ϕ: scorer parameters.



Transformer representation + learned scalar output head

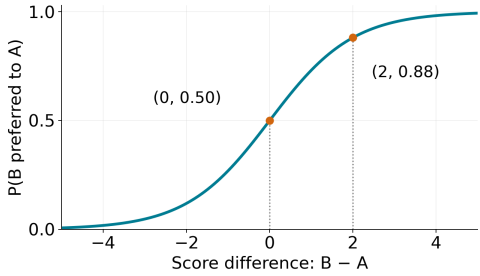
Score new responses without a new human label.

Ouyang et al. (2022), InstructGPT

From score differences to choices

Model assumption: predicted preference probability

$$P(B \succ A \mid x) = \sigma\big(r_\phi(x, B) - r_\phi(x, A)\big)$$



$$\sigma(z) = \frac{1}{1+e^{-z}}$$

A: score 1

B: score 3

Predicted probability
B is preferred: ≈ 0.88

Ouyang et al. (2022), InstructGPT



Training the reward model

Observed label: B is preferred to A.

$$\ell(\phi) = -\log \sigma\big(r_\phi(x, B) - r_\phi(x, A)\big)$$

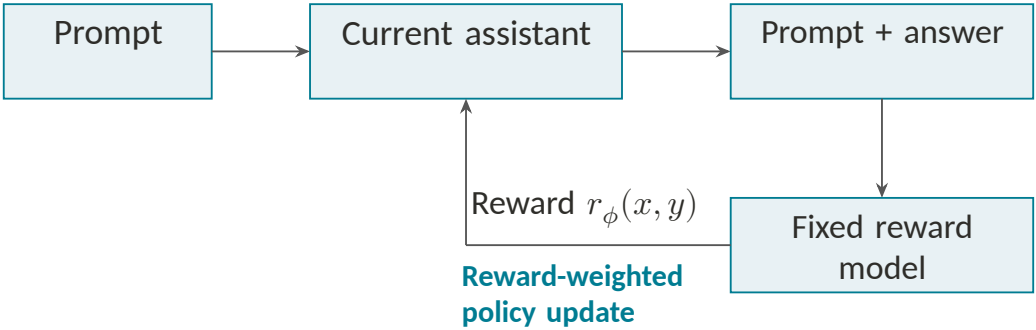
Increase B's score relative to A's to reduce the loss.

Update reward model ϕ. Keep assistant θ fixed.

Ouyang et al. (2022), InstructGPT



Training with the reward model



Then sample new answers from the updated assistant.

Ouyang et al. (2022), InstructGPT · OpenAI Spinning Up, policy gradients

Favoring higher-reward responses

Moon answer	Reward	Before	Illustrative after
A: technical	1	0.50	0.20
B: for a child	3	0.50	0.80

Expected reward: $2.0 \longrightarrow 2.6$

How can we choose parameter updates systematically?



Maximizing expected reward

For a fixed prompt x, the language model is a policy π_θ.

$\pi_\theta(y \mid x)$: probability of generating response y.

$$J(\theta) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}[r_\phi(x, y)]$$

Keep reward parameters ϕ fixed. Update assistant θ.

Williams (1992) · Ouyang et al. (2022), InstructGPT



Differentiating the objective

$$J(\theta) = \sum_y \pi_\theta(y \mid x)\, r_\phi(x, y)$$

$$\nabla_\theta J(\theta) = \sum_y r_\phi(x, y)\, \nabla_\theta \pi_\theta(y \mid x)$$

There are too many responses to sum over explicitly.

Williams (1992) · OpenAI Spinning Up, policy gradients



Estimating the gradient by sampling

$$\nabla_\theta \log \pi_\theta(y \mid x) = \frac{\nabla_\theta \pi_\theta(y|x)}{\pi_\theta(y|x)}$$

$$\nabla_\theta \pi_\theta(y \mid x) = \pi_\theta(y \mid x) \nabla_\theta \log \pi_\theta(y \mid x)$$

$$\nabla_\theta J = \sum_y \pi_\theta(y \mid x)\, r_\phi(x, y) \nabla_\theta \log \pi_\theta(y \mid x)$$

$$\nabla_\theta J = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \left[r_\phi(x, y) \nabla_\theta \log \pi_\theta(y \mid x) \right]$$

Sample answers. Differentiate their weighted log probabilities.

Williams (1992) · OpenAI Spinning Up, policy gradients



The connection back to SFT

SFT: a demonstrated answer receives weight 1.

$$\ell_{\text{SFT}}(\theta) = -\log \pi_\theta(y^* \mid x)$$

Policy gradient: a sampled answer receives weight w.

$$\ell_{\text{PG}}(\theta) = -w \log \pi_\theta(y \mid x), \quad w = r_\phi(x, y)$$

For either loss: $\theta \leftarrow \theta - \eta \nabla_\theta \ell$

Hold the sampled response and its reward fixed for this update.

OpenAI Spinning Up, policy gradients · Schulman et al. (2017), PPO



Centering rewards around a baseline

Update weight $= r_\phi(x, y) - b(x)$

$b(x)$ does not depend on the sampled response.

Starting probabilities: P(A) = P(B) = 0.5. Mean reward = 2.

Response	Reward	Baseline	Weight
A: technical	1	2	−1
B: for a child	3	2	+1

Same expected gradient, potentially lower variance.

OpenAI Spinning Up, policy gradients



Keeping a reference policy

Start from the SFT model and keep a frozen copy.

$$\max_\theta \ \mathbb{E}_{y\sim\pi_\theta}[r_\phi(x,y)] - \beta D_{\mathsf{KL}}(\pi_\theta(\cdot \mid x)\|\pi_{\mathsf{ref}}(\cdot \mid x))$$

$\beta > 0$ sets the penalty strength.

The reward term favors higher-scoring answers.

The KL penalty discourages large changes in the distribution.

Ouyang et al. (2022), InstructGPT



What if the reward model is wrong?

A controlled experiment with two reward models

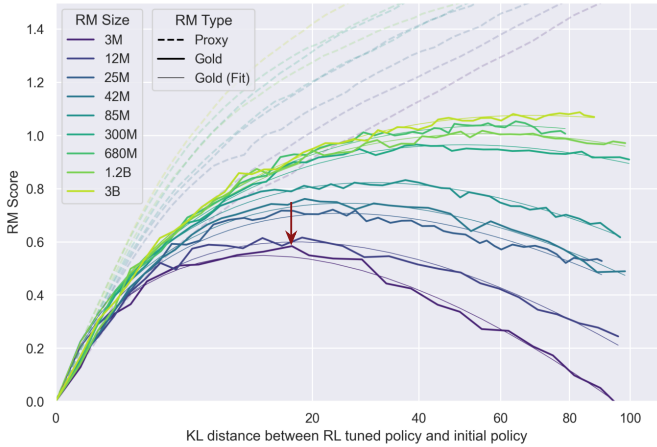
Model	Role
Proxy reward model	Its score trains the assistant
Separate 6B evaluator	Its score measures the result

Synthetic setup: the evaluator also supplies comparison labels.

Gao, Schulman & Hilton (2022)



Overoptimizing the reward model



Gao, Schulman & Hilton (2022)

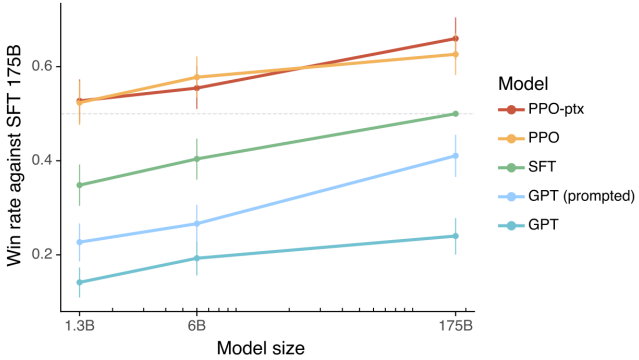
Purple: 3M-parameter proxy reward model.

Dashed: optimized proxy score rises.

Solid: evaluator score first rises, then falls.

X: KL from initial policy (square-root scale).

Returning to InstructGPT



Every point uses the same 175B SFT comparator.

PPO: Proximal Policy Optimization (RL update method)

PPO-ptx adds pretraining-data likelihood updates to RLHF, helping retain pretrained capabilities.

Ouyang et al. (2022)



SFT, LoRA, and RLHF

Method	Training change
SFT	Imitate demonstrated responses
LoRA	Train a low-rank change to the weights
RLHF	Favor responses with higher learned reward

Evaluate response quality and retained capabilities.



Where we go from here

PPO (Proximal Policy Optimization):
control policy-gradient updates

DPO (Direct Preference Optimization):
train directly on preference pairs

Distillation: learning from a teacher's outputs

Reasoning rewards and iterative improvement

Schulman et al. (2017), PPO · Rafailov et al. (2023), DPO · Agarwal et al. (2024), on-policy distillation

