LLM Intro Course: 4 – Post-training: objectives, PEFT, and preferences

SFT, post-training, RL

Simon Vary

Mathematical Institute, University of Oxford

March 25, 2026

The plan for today

- **From pretraining to post-training:** why next-token prediction is not enough for chat behavior

The plan for today

- **From pretraining to post-training:** why next-token prediction is not enough for chat behavior
- **Instruction tuning / SFT:** objective, datasets, and why it gives instruct behavior

The plan for today

- **From pretraining to post-training:** why next-token prediction is not enough for chat behavior
- **Instruction tuning / SFT:** objective, datasets, and why it gives instruct behavior
- **PEFT:** LoRA / QLoRA and where we place the trainable degrees of freedom

The plan for today

- **From pretraining to post-training:** why next-token prediction is not enough for chat behavior
- **Instruction tuning / SFT:** objective, datasets, and why it gives instruct behavior
- **PEFT:** LoRA / QLoRA and where we place the trainable degrees of freedom
- **Limits of SFT:** hallucinations, knowledge extraction, and when fixed targets are not enough

The plan for today

- **From pretraining to post-training:** why next-token prediction is not enough for chat behavior
- **Instruction tuning / SFT:** objective, datasets, and why it gives instruct behavior
- **PEFT:** LoRA / QLoRA and where we place the trainable degrees of freedom
- **Limits of SFT:** hallucinations, knowledge extraction, and when fixed targets are not enough
- **Preference learning and RL:** REINFORCE, PPO, DPO, and GRPO at a high level

The plan for today

- **From pretraining to post-training:** why next-token prediction is not enough for chat behavior
- **Instruction tuning / SFT:** objective, datasets, and why it gives instruct behavior
- **PEFT:** LoRA / QLoRA and where we place the trainable degrees of freedom
- **Limits of SFT:** hallucinations, knowledge extraction, and when fixed targets are not enough
- **Preference learning and RL:** REINFORCE, PPO, DPO, and GRPO at a high level
- **Reasoning scaffolds:** chain-of-thought and verifier-based post-training

Today: Distribution of language $\Rightarrow$ task-oriented model

Pretraining

- Distribution of language
- Syntax, style, world knowledge
- but only to *predict the next token*

Today: Distribution of language ⇒ task-oriented model

Pretraining

- Distribution of language
- Syntax, style, world knowledge
- but only to *predict the next token*

Post-training

- Teaches behaviour
- **SFT:** teach the format
  - answer questions
  - follow instructions
  - imitate demonstrations
- **RL:**
  - human preferences
  - reward models / verifiers
  - RLHF, DPO, RLVR

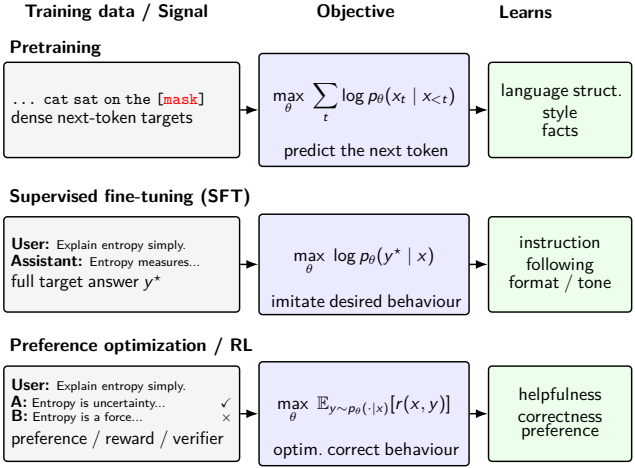
Today: Distribution of language ⇒ task-oriented model

Pretraining

- Distribution of language
- Syntax, style, world knowledge
- but only to *predict the next token*

Post-training

- Teaches behaviour
- **SFT:** teach the format
  - answer questions
  - follow instructions
  - imitate demonstrations
- **RL:**
  - human preferences
  - reward models / verifiers
  - RLHF, DPO, RLVR



SFT + Instruction Tuning

Instruction tuning: objective and optimization

Data and model

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

with instructions $x^{(i)}$ and desired responses $y^{(i)}$.
The autoregressive model defines

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

Instruction tuning: objective and optimization

Data and model

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

with instructions $x^{(i)}$ and desired responses $y^{(i)}$.
The autoregressive model defines

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

Objective

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

- Negative log-likelihood on the **assistant response**.
- **Next-token prediction** conditioned on instruction.

Instruction tuning: objective and optimization

Data and model

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

with instructions $x^{(i)}$ and desired responses $y^{(i)}$.
The autoregressive model defines

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

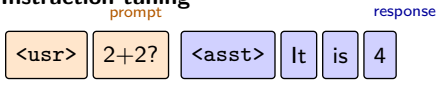
Objective

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

- Negative log-likelihood on the **assistant response**.
- **Next-token prediction** conditioned on instruction.

Pretraining

Instruction tuning



Feed the concatenated sequence $[x_i; y_i]$ into the causal LM, but mask out prompt tokens in the loss.

Instruction Tuning: Data

Dataset	Schema	Size	Typical use	Link
FLAN Collection	task mixtures	large	broad academic IT mixture	FLAN
Super-Natural Inst.	task definitions + examples	1616 tasks	natural-language task instructions with positive / negative examples	SNI
Alpaca	instruction / input / output	52k	canonical open single-turn SFT format	Alpaca
Dolly-15k	prompt / response	15k	human-written open instruction-response pairs	Dolly
OpenAssistant (OASST1)	multi-turn chat tree	161k msgs	chat-style supervision with roles, branches, and rankings	OASST1

Example of SFT record

```
{ "instruction": "Explain why the sky is blue.",
"output": "Sunlight scatters in the atmosphere; shorter blue wavelengths scatter more strongly." }
```

Raw text $\Rightarrow$ **structured demonstrations**: instruction $\rightarrow$ desired assistant response, often with explicit roles and formatting.

FLAN: Instruction Tuning ⇒ Zero-shot Learning **(Wei et al., 2021)**

What FLAN showed

- IT improves **zero-shot** generalization
- gains transfer to **unseen tasks**
- FLAN 137B beats **zero-shot GPT-3 175B** on many benchmarks

FLAN: Instruction Tuning ⇒ Zero-shot Learning **(Wei et al., 2021)**

What FLAN showed

- IT improves **zero-shot** generalization
- gains transfer to **unseen tasks**
- FLAN 137B beats **zero-shot GPT-3 175B** on many benchmarks

Takeaways

- SFT can create **instruct behavior**
- diverse supervised data gives **cross-task generalization**

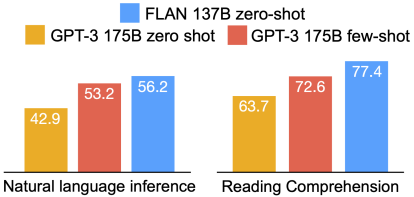
FLAN: Instruction Tuning ⇒ Zero-shot Learning **(Wei et al., 2021)**

What FLAN showed

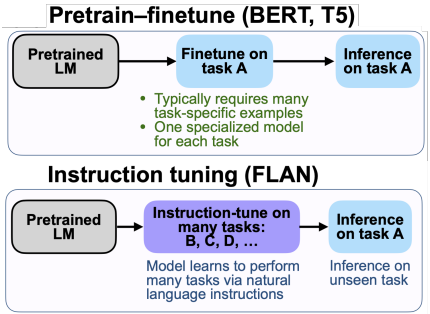
- IT improves **zero-shot** generalization
- gains transfer to **unseen tasks**
- FLAN 137B beats **zero-shot GPT-3 175B** on many benchmarks

Takeaways

- SFT can create **instruct behavior**
- diverse supervised data gives **cross-task generalization**



Performance on unseen tasks



PEFT: adapt a large model with a small trainable update

Low-Rank Adaptation (LoRA)

Freeze the pretrained matrix W_0 and learn only a low-rank update:

$$W = W_0 + \Delta W, \qquad \Delta W = BA, \qquad r \ll d.$$

So instead of training all of W, we train only the small factors A and B.

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

PEFT: adapt a large model with a small trainable update

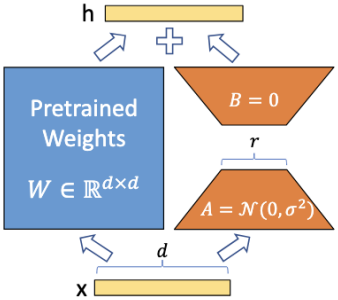
Low-Rank Adaptation (LoRA)

Freeze the pretrained matrix W_0 and learn only a low-rank update:

$$W = W_0 + \Delta W, \qquad \Delta W = BA, \qquad r \ll d.$$

So instead of training all of W, we train only the small factors A and B.

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



Why this is useful

- few trainable parameters (lower memory / optimizer cost
- one frozen base model + small adapters
- reduces forgetting since W_0 stays fixed

PEFT: adapt a large model with a small trainable update

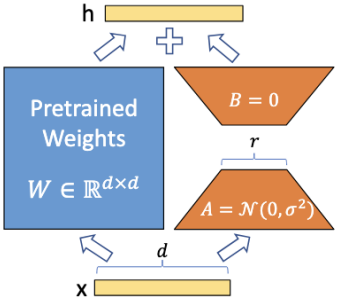
Low-Rank Adaptation (LoRA)

Freeze the pretrained matrix W_0 and learn only a low-rank update:

$$W = W_0 + \Delta W, \qquad \Delta W = BA, \qquad r \ll d.$$

So instead of training all of W, we train only the small factors A and B.

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



Why this is useful

- few trainable parameters (lower memory / optimizer cost
- one frozen base model + small adapters
- reduces forgetting since W_0 stays fixed

Other PEFT

- **Adapters:** bottleneck modules
- **QLORA:** Quantized LoRA

Knowledge extraction, alignment, and hallucinations

- When the model doesn't have the knowledge,
  SFT will just *imitate the style*

Knowledge extraction, alignment, and hallucinations

- When the model doesn't have the knowledge,
  SFT will just *imitate the style*
- Cross-entropy loss:
  a confident hallucination $\gg$ no answer at all

Knowledge extraction, alignment, and hallucinations

- When the model doesn't have the knowledge, SFT will just *imitate the style*
- Cross-entropy loss:
  a confident hallucination $\gg$ no answer at all

Counterintuitive phenomenon

- *Rich and correct* instruction data will still encourage hallucinations
- A weaker student may learn the *format* of expert answers, not the underlying ability
- In distillation, a much stronger teacher can make this worse

Knowledge extraction, alignment, and hallucinations

- When the model doesn't have the knowledge, SFT will just *imitate the style*
- Cross-entropy loss:
  a confident hallucination $\gg$ no answer at all

Counterintuitive phenomenon

- *Rich and correct* instruction data will still encourage hallucinations
- A weaker student may learn the *format* of expert answers, not the underlying ability
- In distillation, a much stronger teacher can make this worse

Takeaway

- SFT is good at **extracting present knowledge**
- It is less reliable for **adding new capabilities**

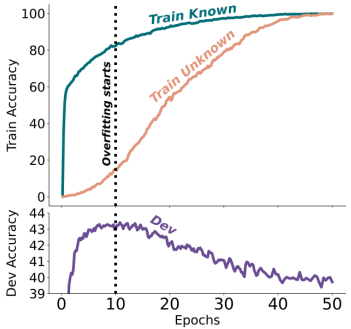
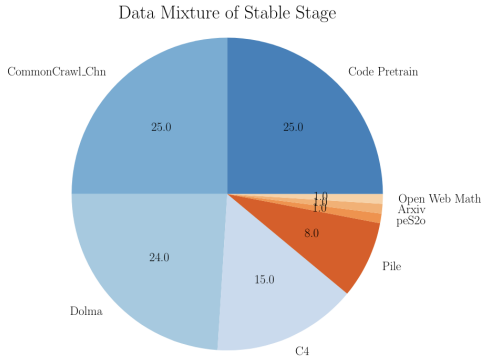
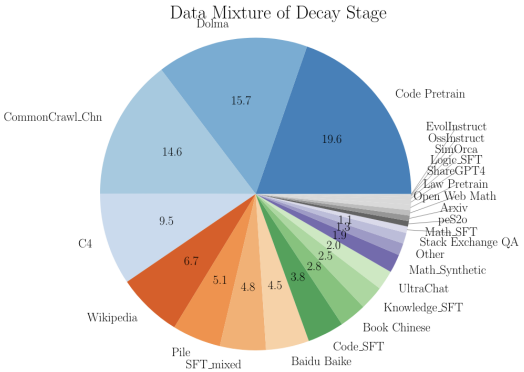


Figure 1: Train and development accuracies as a function of the fine-tuning duration, when fine-tuning on 50% `Known` and 50% `Unknown` examples. `Unknown` examples are fitted substantially slower than `Known`. The best development performance is obtained when the LLM fits the majority of the `Known` training examples but only few of the `Unknown` ones. From this point, fitting `Unknown` examples reduces the performance.

Midtraining / Two-phase training

- train on a **mixture** of web-scale text and instruction data
- **injects instruct behaviour** without leaving the pretraining regime entirely
- preserves **general knowledge and language modelling ability**
- reduces **catastrophic forgetting:** the model loosing previously learned capabilities.



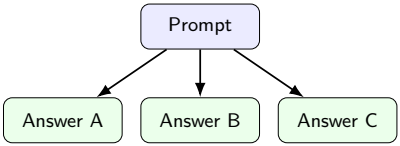


When instruction tuning is not sufficient

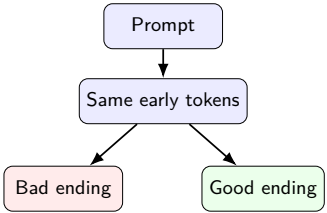
Instruction tuning is limited

- **There is no single "correct" path**
- **Quality is judged by outcome**
- **The important signal appears only after generation**
- **Different mistakes have different severity**
- **The model may exploit superficial patterns**
- **Preferences are often relative, not absolute**

In short: instruction tuning learns from fixed targets, but many post-training signals are comparative or outcome-based.



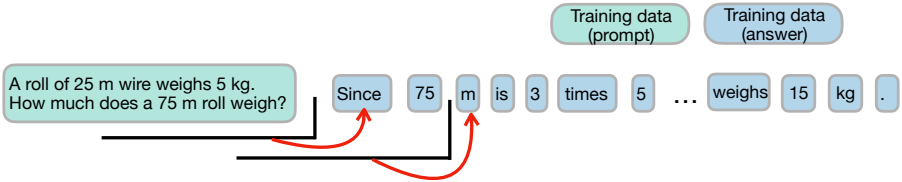
Many plausible responses: instruction tuning reinforces one recorded target, even when several answers are acceptable.



Outcome visible late: two responses may look similar at first, but only the full completion reveals whether it is correct, safe, or useful.

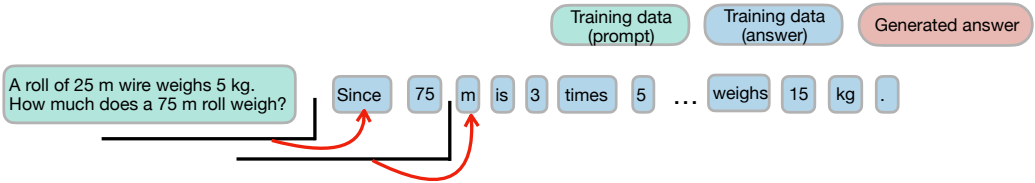
RL

SFT $\Rightarrow$ RL

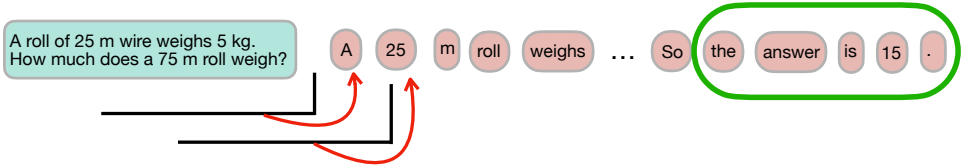


Compute loss of $p_\theta(\text{"m"} \mid \text{"Since"}, \text{"75"}, \text{prompt}) \Rightarrow$ gradient update on θ to increase the prob. of observed training distribution

SFT ⇒ RL



Compute loss of $p_\theta(\text{"m"} \mid \text{"Since"}, \text{"75"}, \text{prompt}) \Rightarrow$ gradient update on θ to increase the prob. of observed training distribution



Generate from "m" $\sim \pi_\theta(\cdot \mid \text{"A"}, \text{"25"}, \text{prompt}) \Rightarrow$ check correctness $\Rightarrow$ push probability of path up

SFT vs RL

Supervised Fine-Tuning (SFT)

$$\max_{\theta} \mathbb{E}_{(x,y^\star)\sim\mathcal{D}}\big[\log \pi_\theta(y^\star \mid x)\big]$$

- **Target:** *"do this"*
- Full desired output $y^\star$ is given
- Strong, structured training signal

$$x \longrightarrow y^\star$$

Increase probability of the target response.

SFT vs RL

Supervised Fine-Tuning (SFT)

$$\max_{\theta} \mathbb{E}_{(x,y^\star)\sim\mathcal{D}}\big[\log \pi_\theta(y^\star \mid x)\big]$$

- **Target:** *"do this"*
- Full desired output $y^\star$ is given
- Strong, structured training signal

$$x \longrightarrow y^\star$$

Increase probability of the target response.

Reinforcement Learning (RL)

$$\max_{\theta} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[r(x,y)\big]$$

- **Reward:** *"what you did worked well"*
- Model must first sample an output y
- Feedback is weaker: a reward score

$$x \longrightarrow y \longrightarrow r(x,y)$$

Increase probability of higher-reward responses.

SFT vs RL

Supervised Fine-Tuning (SFT)

$$\max_{\theta} \mathbb{E}_{(x,y^\star)\sim\mathcal{D}}\big[\log \pi_\theta(y^\star \mid x)\big]$$

- **Target:** *"do this"*
- Full desired output $y^\star$ is given
- Strong, structured training signal

$$x \longrightarrow y^\star$$

Increase probability of the target response.

Reinforcement Learning (RL)

$$\max_{\theta} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[r(x,y)\big]$$

- **Reward:** *"what you did worked well"*
- Model must first sample an output y
- Feedback is weaker: a reward score

$$x \longrightarrow y \longrightarrow r(x,y)$$

Increase probability of higher-reward responses.

Challenges of using rewards

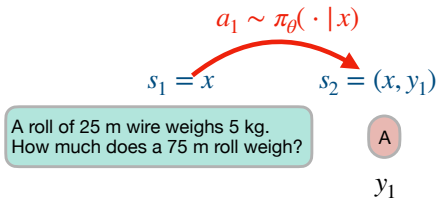
- **Scalar vs structured:** a label specifies the desired output; a reward is one number.
- **Credit assignment:** the reward arrives after the whole response, so it is unclear which part of reasoning are responsible for it.

RL formulation for an LLM

action a_t : sampled token $y_t \sim p_\theta(\cdot \mid x, y_{<t})$

state s_t : prompt + partial completion $(x, y_{<t})$

policy π_θ : LLM next tok. prediction

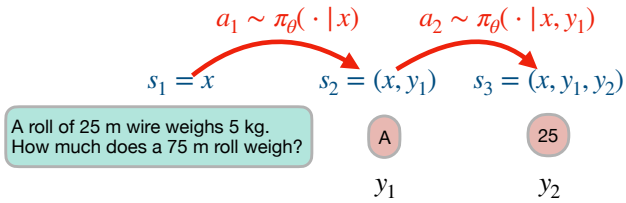


RL formulation for an LLM

action a_t : sampled token $y_t \sim p_\theta(\cdot \mid x, y_{<t})$

state s_t : prompt + partial completion $(x, y_{<t})$

policy π_θ : LLM next tok. prediction

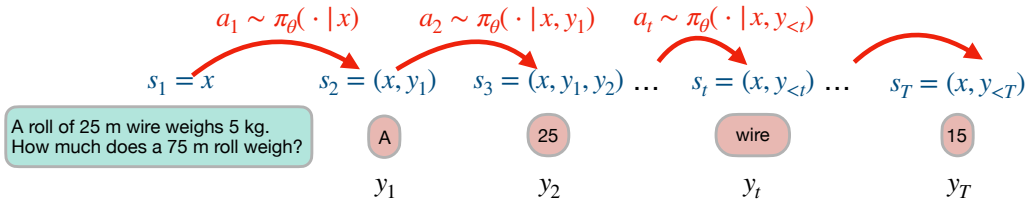


RL formulation for an LLM

action a_t : sampled token $y_t \sim p_\theta(\cdot \mid x, y_{<t})$

state s_t : prompt + partial completion $(x, y_{<t})$

policy π_θ : LLM next tok. prediction



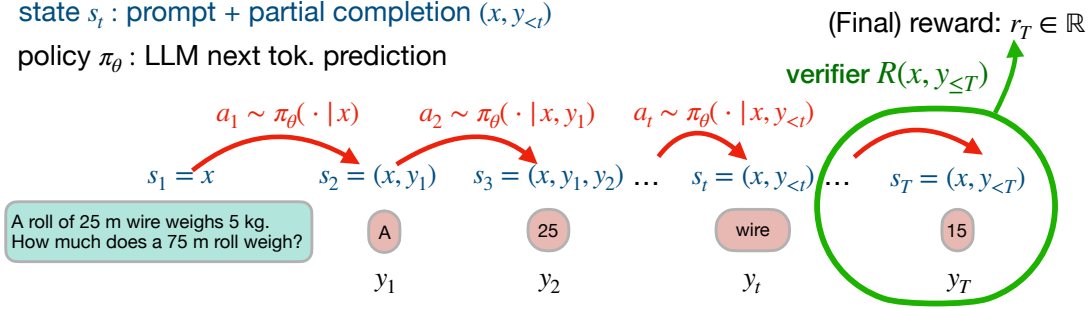
rollout $y_{\leq T}$: fully AR sampled answer

RL formulation for an LLM

action a_t : sampled token $y_t \sim p_\theta(\cdot \mid x, y_{<t})$

state s_t : prompt + partial completion $(x, y_{<t})$

policy π_θ : LLM next tok. prediction



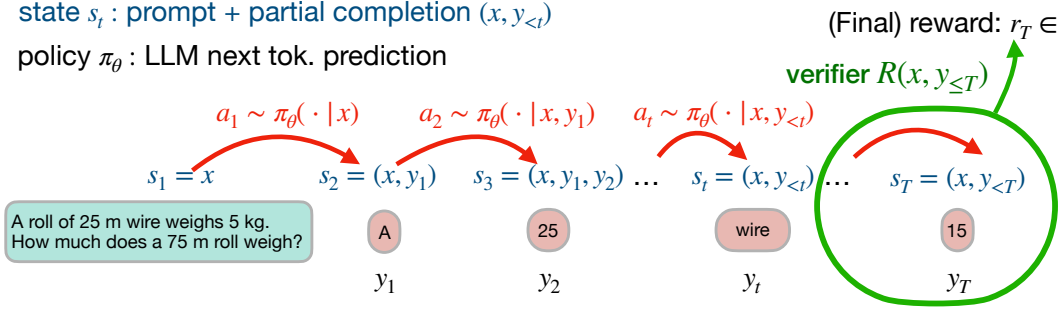
rollout $y_{\le T}$: fully AR sampled answer

RL formulation for an LLM

action a_t : sampled token $y_t \sim p_\theta(\cdot \mid x, y_{<t})$

state s_t : prompt + partial completion $(x, y_{<t})$

policy π_θ : LLM next tok. prediction



rollout $y_{\le T}$: fully AR sampled answer

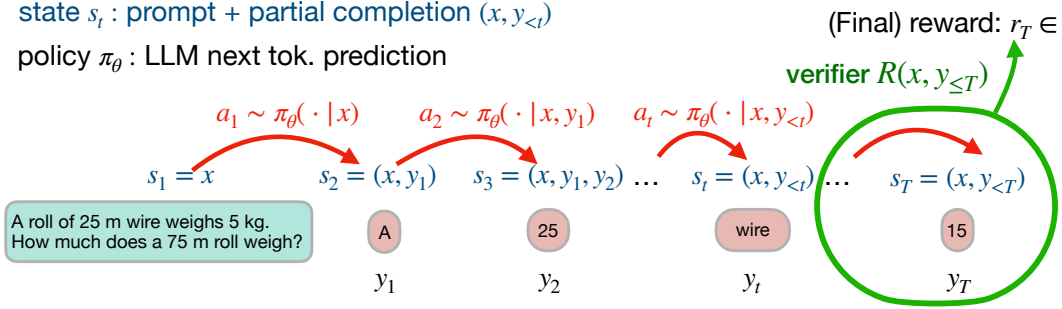
total future reward: $G_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, where $\gamma \in (0,1)$ is fixed discounting

RL formulation for an LLM

action a_t : sampled token $y_t \sim p_\theta(\cdot \mid x, y_{<t})$

state s_t : prompt + partial completion $(x, y_{<t})$

policy π_θ : LLM next tok. prediction



rollout $y_{\le T}$: fully AR sampled answer

total future reward: $G_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, where $\gamma \in (0,1)$ is fixed discounting

Value function: $V^\pi(s_t) = \mathbb{E}_{a \sim \pi_\theta(\cdot \mid s_t)} \left[G_t \mid s_t \right]$ $\Longleftarrow$ Expected future reward under π_θ from s_t

REINFORCE: Gradient Update on the Policy (LLM)

Objective:

$$\max_{\theta} J(\theta) := \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[R(x,y)\right]$$

REINFORCE: Gradient Update on the Policy (LLM)

Objective:

$$\max_{\theta} J(\theta) := \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[R(x,y)\right]$$

Idea: Use gradient update:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha \nabla_\theta J(\theta)$$

REINFORCE: Gradient Update on the Policy (LLM)

Objective:

$$\max_{\theta} J(\theta) := \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[R(x,y)\right]$$

Idea: Use gradient update:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha \nabla_\theta J(\theta)$$

How to compute $\nabla_\theta J(\theta)$?

$$\nabla_\theta \pi_\theta = \pi_\theta \nabla_\theta \log \pi_\theta$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{x,y}[R(x,y)] = \mathbb{E}_{x,y}[R(x,y)\,\nabla_\theta \log \pi_\theta(y\,|\,x)]$$

REINFORCE: Gradient Update on the Policy (LLM)

Objective:

$$\max_{\theta} J(\theta) := \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[R(x,y)\right]$$

Idea: Use gradient update:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha \nabla_\theta J(\theta)$$

How to compute $\nabla_\theta J(\theta)$?

$\nabla_\theta \pi_\theta = \pi_\theta \nabla_\theta \log \pi_\theta$

$$\nabla_\theta J(\theta) = \mathbb{E}_{x,y}[R(x,y)] = \mathbb{E}_{x,y}[R(x,y)\,\nabla_\theta \log \pi_\theta(y\,|\,x)]$$

$$\pi_\theta(y \mid x) = \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t}) \qquad = \mathbb{E}_{x,y}\left[R(x,y)\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right].$$

REINFORCE: Gradient Update on the Policy (LLM)

Objective:

$$\max_\theta J(\theta) := \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[R(x,y)\right]$$

Idea: Use gradient update:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha \nabla_\theta J(\theta)$$

How to compute $\nabla_\theta J(\theta)$?

$\nabla_\theta \pi_\theta = \pi_\theta \nabla_\theta \log \pi_\theta$

$$\nabla_\theta J(\theta) = \mathbb{E}_{x,y}[R(x,y)] = \mathbb{E}_{x,y}[R(x,y)\,\nabla_\theta \log \pi_\theta(y\,|\,x)]$$

$$\pi_\theta(y \mid x) = \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t}) \qquad = \mathbb{E}_{x,y}\left[R(x,y)\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right].$$

REINFORCE:

- sample prompt $x \sim D$
- answer $y \sim \pi_\theta(\,\cdot\,|\,x)$
- score reward $R(x,y) \in \mathbb{R}^d$

$$\theta^{(t+1)} = \theta^{(t)} + \alpha\, R(x,y) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})$$

From REINFORCE to PPO (1/2): Advantages and value functions

Problem: Sparse rewards $\Rightarrow$ high-variance

The *same score* applied to every token in the response.

- which tokens actually mattered for the answer?
- long completions $\Rightarrow$ noisy gradient
- some trajectories more informative than others

From REINFORCE to PPO (1/2): Advantages and value functions

Problem: Sparse rewards ⇒ high-variance

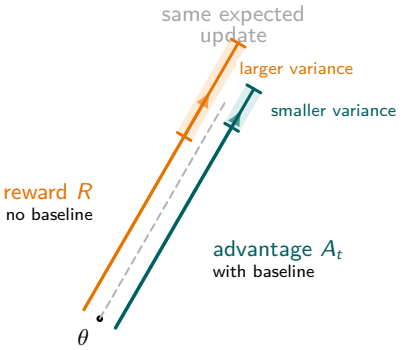
The *same score* applied to every token in the response.

- which tokens actually mattered for the answer?
- long completions ⇒ noisy gradient
- some trajectories more informative than others

Idea: subtract expected reward as a baseline

$$\sum_{t=1}^{T} \underbrace{\left(R(x,y) - V^{\pi_\theta}(x)\right)}_{A_t = \text{Advantage}} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}).$$

- Reduces variance by centering the reward signal
- Same expected value of the gradient estimate
- *Advantage:* better or worse than expected?
- **Challenge:** Unknown $V^{\pi_\theta}(x)$ ⇒ need to estimate



$$\mathbb{E}_{y\sim\pi_\theta}[R(x,y)\, g_\theta] = \mathbb{E}_{y\sim\pi_\theta}[A_t\, g_\theta]$$

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

From REINFORCE to PPO (2/2): Trust region and clipping

Problem: Rollouts are expensive

- take several gradient steps with the same rollout
- generate multiple rollouts in parallel

But then, the updated policy $\pi_\theta \neq \pi_{\text{old}}$ will differ:

- the gradient estimate becomes less reliable
- large policy changes can hurt language quality

From REINFORCE to PPO (2/2): Trust region and clipping

Problem: Rollouts are expensive

- take several gradient steps with the same rollout
- generate multiple rollouts in parallel

But then, the updated policy $\pi_\theta \neq \pi_{\text{old}}$ will differ:

- the gradient estimate becomes less reliable
- large policy changes can hurt language quality

Idea: trust region + importance sampling

Reuse the same rollouts and correct the change:

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

$$\widehat{J(\theta)} = \mathbb{E}_{x,y}\left[\sum_t \min\Big(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t\Big)\right]$$

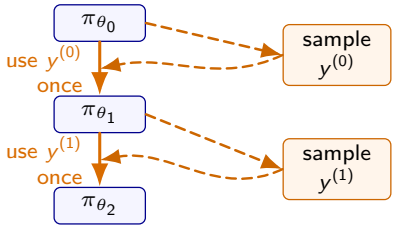
From REINFORCE to PPO (2/2): Trust region and clipping

Problem: Rollouts are expensive

- take several gradient steps with the same rollout
- generate multiple rollouts in parallel

But then, the updated policy $\pi_\theta \neq \pi_{\text{old}}$ will differ:

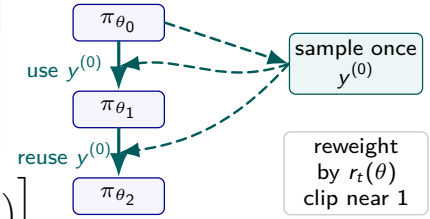
- the gradient estimate becomes less reliable
- large policy changes can hurt language quality



Idea: trust region + importance sampling

Reuse the same rollouts and correct the change:

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



$$\widehat{J(\theta)} = \mathbb{E}_{x,y}\left[\sum_t \min\Big(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t\Big)\right]$$

PPO-Clip psuedocode (OpenAI Spinning Up, 2018)

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

Direct Preference Optimization (DPO): Pairwise preferences

Goal: Simplify RLHF

DPO trains on *preference* pairs

$$(x, y_w, y_l), \quad \text{where } y_w \text{ is preferred over } y_l.$$

Direct Preference Optimization (DPO): Pairwise preferences

Goal: Simplify RLHF

DPO trains on *preference* pairs

$$(x, y_w, y_l), \quad \text{where } y_w \text{ is preferred over } y_l.$$

Idea: Policy defines an implied reward

For KL-regularized RLHF, we can write the reward as

$$r(x, y) = \beta \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x).$$

Relative probability vs. the reference model $\Rightarrow$ reward.

Direct Preference Optimization (DPO): Pairwise preferences

Goal: Simplify RLHF

DPO trains on *preference* pairs

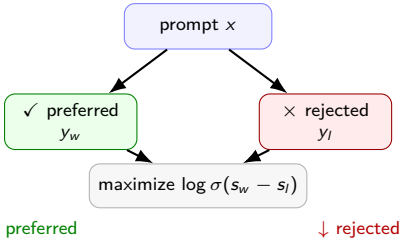
$$(x, y_w, y_l), \quad \text{where } y_w \text{ is preferred over } y_l.$$

Idea: Policy defines an implied reward

For KL-regularized RLHF, we can write the reward as

$$r(x, y) = \beta \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x).$$

Relative probability vs. the reference model $\Rightarrow$ reward.

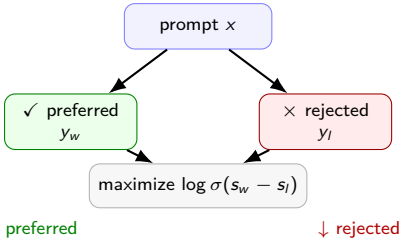


Direct Preference Optimization (DPO): Pairwise preferences

Goal: Simplify RLHF

DPO trains on *preference* pairs

$$(x, y_w, y_l), \quad \text{where } y_w \text{ is preferred over } y_l.$$



Idea: Policy defines an implied reward

For KL-regularized RLHF, we can write the reward as

$$r(x, y) = \beta \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x).$$

Relative probability vs. the reference model $\Rightarrow$ reward.

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

$$\nabla_\theta \mathcal{L}_{\text{DPO}} = -\beta\, \mathbb{E}\Big[w(x, y_w, y_l) \left(\nabla_\theta \log \pi_\theta(y_w \mid x) - \nabla_\theta \log \pi_\theta(y_l \mid x) \right) \Big],$$

where $w(x, y_w, y_l)$ tells how badly the current model violates the preference.

Group Relative Policy Optimization (GRPO): PPO without a critic

Goal: simplify PPO-style RL

GRPO keeps PPO's *clipped* policy update, but removes the learned value function / critic.

- for each prompt q, sample a group $o_1, \ldots, o_G$
- score them with a verifier / reward $r_1, \ldots, r_G$
- compare each response *relative to the group*

Group Relative Policy Optimization (GRPO): PPO without a critic

Goal: simplify PPO-style RL

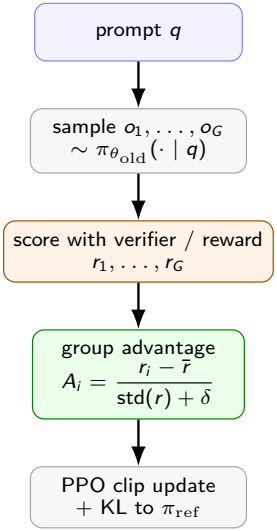
GRPO keeps PPO's *clipped* policy update, but removes the learned value function / critic.

- for each prompt q, sample a group $o_1, \ldots, o_G$
- score them with a verifier / reward $r_1, \ldots, r_G$
- compare each response *relative to the group*

Idea: use a group-normalized advantage

$$A_i = \frac{r_i - \bar{r}}{\mathrm{std}(r_1, \ldots, r_G) + \delta}, \qquad \bar{r} = \frac{1}{G} \sum_{j=1}^{G} r_j.$$

- $A_i > 0$: better than the group average
- $A_i < 0$: worse than the group average



Chain-of-Thought Ellicits Reasoning **(Wei et al., 2022)**

Idea:

- Reasoning tasks are hard to solve as $x \to y$.
- Instead learn $x \to z \to y$ (z is "chain of thought")

This *elicit* much better performance.

Chain-of-Thought Ellicits Reasoning **(Wei et al., 2022)**

Idea:

- Reasoning tasks are hard to solve as $x \to y$.
- Instead learn $x \to z \to y$ (z is "chain of thought")

This *elicit* much better performance.

CoT prompting:

- showed few CoT examples in context
- improves arithmetic and symbolic reasoning
- the effect is stronger in large models
- produced reasoning traces useful for post-training

Chain-of-Thought Ellicits Reasoning **(Wei et al., 2022)**

Idea:

- Reasoning tasks are hard to solve as $x \to y$.
- Instead learn $x \to z \to y$ (z is "chain of thought")

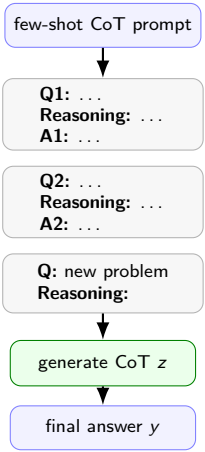
This *elicit* much better performance.

CoT prompting:

- showed few CoT examples in context
- improves arithmetic and symbolic reasoning
- the effect is stronger in large models
- produced reasoning traces useful for post-training

Takeaway

Giving the model the ability to generate (longer) reasoning traces allows it to decompose complex problems in simpler steps and improves reasoning.



Takeaways

Takeaways: Post-training

- **Supervised fine-tuning (SFT):** SFT is still next-token prediction, but now as *conditional* probability fitting on instruction–response pairs

Takeaways: Post-training

- **Supervised fine-tuning (SFT):** SFT is still next-token prediction, but now as *conditional* probability fitting on instruction–response pairs
- **Parameter-efficient adaptation (PEFT):** Methods such as LoRA / QLoRA make adaptation cheap by fine-tuning only a small number of trainable parameters.

Takeaways: Post-training

- **Supervised fine-tuning (SFT):** SFT is still next-token prediction, but now as *conditional* probability fitting on instruction–response pairs
- **Parameter-efficient adaptation (PEFT):** Methods such as LoRA / QLoRA make adaptation cheap by fine-tuning only a small number of trainable parameters.
- **SFT vs. RL:** SFT is *off-policy*: it learns from fixed target data. RL is *on-policy*: it trains on outputs sampled from the current policy and rewards.

Takeaways: Post-training

- **Supervised fine-tuning (SFT):** SFT is still next-token prediction, but now as *conditional* probability fitting on instruction–response pairs
- **Parameter-efficient adaptation (PEFT):** Methods such as LoRA / QLoRA make adaptation cheap by fine-tuning only a small number of trainable parameters.
- **SFT vs. RL:** SFT is *off-policy*: it learns from fixed target data. RL is *on-policy*: it trains on outputs sampled from the current policy and rewards.
- **Limits of SFT:** SFT imitates the *format* of good answers without adding the capability $\rightarrow$ may encourage hallucinations.

Takeaways: Post-training

- **Supervised fine-tuning (SFT):** SFT is still next-token prediction, but now as *conditional* probability fitting on instruction–response pairs
- **Parameter-efficient adaptation (PEFT):** Methods such as LoRA / QLoRA make adaptation cheap by fine-tuning only a small number of trainable parameters.
- **SFT vs. RL:** SFT is *off-policy*: it learns from fixed target data. RL is *on-policy*: it trains on outputs sampled from the current policy and rewards.
- **Limits of SFT:** SFT imitates the *format* of good answers without adding the capability $\rightarrow$ may encourage hallucinations.
- **Preference learning and RL:** Use richer feedback than a single target. Optimize relative preferences or outcome-based rewards.

Takeaways: Post-training

- **Supervised fine-tuning (SFT):** SFT is still next-token prediction, but now as *conditional* probability fitting on instruction–response pairs
- **Parameter-efficient adaptation (PEFT):** Methods such as LoRA / QLoRA make adaptation cheap by fine-tuning only a small number of trainable parameters.
- **SFT vs. RL:** SFT is *off-policy*: it learns from fixed target data. RL is *on-policy*: it trains on outputs sampled from the current policy and rewards.
- **Limits of SFT:** SFT imitates the *format* of good answers without adding the capability $\rightarrow$ may encourage hallucinations.
- **Preference learning and RL:** Use richer feedback than a single target. Optimize relative preferences or outcome-based rewards.
- **Final takeaway:** Post-training turns a pretrained language model into a useful assistant by choosing the right data, objective, and feedback signal.

Not covered

- Mixture of Experts
- Scaling laws
- Theory + Training dynamics
- Data-mixtures
- Technical: Inference vLLM, PagedAttention, FlashAttention.