Generative Lecture 7: Fine-tuning

Asrl · 2026年11月20日 · generative

This note gives a brief introduction to fine-tuning, including instruction fine-tuning and reinforcement fine-tuning, and introduces REINFORCE algorithm.

原文：https://asrl.top/posts/generative-fine-tuning



文章封面

Fine-tuning

What is Fine-tuning?

Pre-training and Fine-tuning

After pre-training, the model may perform well on next-token prediction but struggle with QA tasks, which are more common in everyday interactions. So we need to adapt the LLM into a chat model.

Training a separate model from scratch for every task requires substantial labeled data. Pre-training provides a shared starting point for fine-tuning on specific downstream tasks.

Instruction Fine-tuning

Training on QA Interactions

Starting from the pre-trained parameters, we minimize the next-token prediction loss on instruction–response pairs using a small learning rate.

What is the task? Learn to follow instructions in QA interactions.

- Reminder: Instruction fine-tuning continues training with a next-token prediction loss, but uses instruction–response pairs.

Aligning with practical tasks

Misalignment can arise from noisy training data and the fact that good next-token prediction does not guarantee good sequence-level performance.

To align the model with practical tasks, one approach is **reinforcement fine-tuning**: The model generates an answer sequence. A human evaluator scores the whole sequence. The model uses the score to update its parameters.

Reinforcement Fine-tuning

Framework

Given a prompt q and a response s, suppose we have a human reward score $R(q, s) \in \mathbb{R}$, where higher is better:

$$q \xrightarrow{\text{input}} \text{LLM } p(\cdot \mid q; \theta) \xrightarrow{\text{sampling}} s \xrightarrow{\text{label}} R(q, s)$$

To update θ, we need $\nabla_\theta \mathbb{E}_{s\sim p(\cdot|q;\theta)}[R(q, s)]$, but we cannot backpropagate directly through discrete sampling. We therefore use the following algorithms.

REINFORCE algorithm

REINFORCE

For simplicity, fix a prompt q and omit it from the notation. We maximize the expected reward:

$$\max_\theta J(\theta), \qquad J(\theta) = \mathbb{E}_{s\sim p(\cdot;\theta)}[R(s)].$$

The **log-derivative trick** gives:

$$\nabla_\theta p(s;\theta) = p(s;\theta)\nabla_\theta \log p(s;\theta).$$

Treating the reward function as fixed, we have:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_s p(s;\theta) R(s) \\
&= \sum_s R(s) \nabla_\theta p(s;\theta) \\
&= \sum_s p(s;\theta) R(s) \nabla_\theta \log p(s;\theta) \\
&= \mathbb{E}_{s\sim p(\cdot;\theta)} \left[R(s) \nabla_\theta \log p(s;\theta) \right].
\end{aligned}$$

We only differentiate the log-probability, so no gradient through the reward or discrete sampling is needed.

It is difficult to compute the expectation exactly, so we estimate it by sampling. At iteration t, sample K responses $s_k \sim p(\cdot;\theta_t)$ and compute their rewards. Estimate the gradient by:

$$\Delta\theta_t = \frac{1}{K} \sum_{k=1}^{K} R(s_k)\ \nabla_\theta \log p(s_k;\theta)|_{\theta=\theta_t}.$$

This is an unbiased Monte Carlo estimate of $\nabla_\theta J(\theta_t)$. Update the parameters by gradient ascent with learning rate α:

$$\theta_{t+1} = \theta_t + \alpha \Delta\theta_t.$$

Intuition: Positive rewards encourage the model to generate the sampled responses; negative rewards discourage it.

In implementation, minimize the following loss, treating the sampled responses and rewards as constants:

$$L_t(\theta) = -\frac{1}{K} \sum_{k=1}^{K} R(s_k) \log p(s_k;\theta).$$

The minus sign converts gradient ascent into gradient descent. Sample new responses from the updated model for the next iteration.

REINFORCE with a baseline

In the original REINFORCE algorithm, both small and large positive rewards give an upward contribution. We compare rewards with a fixed baseline b, independent of the sampled response s:

$$A(s) = R(s) - b.$$

Positive $A(s)$ gives an upward contribution; negative $A(s)$ gives a downward contribution.

Notice that the baseline contributes zero to the expected gradient:

$$\begin{aligned}\mathbb{E}_{s\sim p(\cdot;\theta)}\left[b\nabla_\theta \log p(s;\theta)\right] &= b\sum_s p(s;\theta)\nabla_\theta \log p(s;\theta) \\ &= b\sum_s \nabla_\theta p(s;\theta) \\ &= b\nabla_\theta \sum_s p(s;\theta) \\ &= b\nabla_\theta 1 = 0.\end{aligned}$$

Replace $R(s_k)$ with $A(s_k)$ in the gradient estimate and loss above.

REINFORCE with a baseline + reuse samples

Still GUNMU

PPO algorithm

Still GUNMU