07-280: LLM Post-Training

Nihar B. Shah

Pre-training

Unsupervised training. Self-supervised by masking tokens and training the model to predict them from previous tokens. Very good autocomplete.

E.g., Prompt: *"What is the capital of France?"*
Response: *"What is the capital of Germany?"*

(See `https://brainly.com/question/59632956`, `https://stackoverflow.com/questions/69227995/is-there-an-easier-way-to-make-a-randomised-quiz` for examples of such data)

E.g., Prompt: *"Can you teach me ⟨something harmful⟩?"*
Response: *"Yeah, . . . "*

Post-training

Improve the pre-trained (base) model to follow instructions, have better alignment, better performance overall. We will discuss four methods:

- Instruction tuning
- Supervised fine tuning (SFT)
- RL with human feedback (RLHF)
- RL with verifiable rewards (RLVR)

Instruction Tuning

Fine tune to teach question–answer format. Have samples of the form:
x = "⟨system⟩ You are a helpful assistant who will answer questions. ⟨user⟩ What is the capital of USA? ⟨assistant⟩"
and
y = "Washington DC"

The specific question and answer are different across samples.

When deployed, the model internally receives the prompt in a format identical to the x samples.

SFT

Discussed in earlier lecture. *Note:* instruction tuning is a special case of SFT.

RLHF

Incorporate human feedback on model's outputs. Two stages:

1. Train a "reward model" that can take as input a prompt + response and output a rating of how good that output is for that prompt.
2. Use this reward model to improve pretrained / fine-tuned model using techniques from RL.

Stage 1

Dataset: Many samples of the form

- Prompt (x)
- Two responses from the language model
- A human annotates which of the two responses is better.

Let us denote the chosen response as y_C ($c =$ chosen) and the other as y_R ($R =$ rejected).

Model: We will now create the reward model. It is a neural net with parameters denoted by ϕ.

To start off, we will need a model that can process language. We don't want to create one from scratch, as that would be a massive effort. Instead, we take our existing language model and modify it to make a reward model, e.g., by removing the last language layer and instead adding a linear layer with trainable parameters.

The actual training is done via MLE. The probabilistic assumption ("Bradley–Terry assumption") is that for "true" parameters ϕ,

$$\mathbb{P}(y_C \text{ is chosen over } y_R \mid x) \;=\; \frac{e^{r_\phi(y_C|x)}}{e^{r_\phi(y_C|x)} + e^{r_\phi(y_R|x)}} \;=\; \frac{1}{1 + e^{-(r_\phi(y_C|x) - r_\phi(y_R|x))}}.$$

Given dataset $\{(x_i, y_{c,i}, y_{R,i})\}_{i=1}^n$, compute ϕ as

$$\arg\max_\phi \; \frac{1}{n} \sum_{i=1}^{n} \log\left(\frac{1}{1 + e^{-(r_\phi(y_{c,i}|x_i) - r_\phi(y_{R,i}|x_i))}} \right).$$

This gives a reward model. (*Exercise:* identify connection to logistic regression.)

Stage 2

Using the trained reward model, use techniques from RL to update the language model. Here is the mapping to RL:

- **States:** Prompt + output so far.
- **Action:** Next token.
- **Reward:** $r_\phi(y \mid x)$ from the reward model, obtained at the end of the full response.
- **Policy:** Our language model, denoted as π_θ, where $\theta =$ parameters.

If we transport standard RL objectives, we would look to optimize

$$\arg\max_\theta \; \mathbb{E}_{x \sim \text{prompt distribution},\, y \sim \pi_\theta}[r_\phi(y \mid x)].$$

Problem: this can take the language model far away from the pretrained language model. The language model can forget a lot of pretrained knowledge and overfit to the human feedback data.

Main idea: Denote the language model obtained at the end of pretraining + fine tuning as π_{ref}. Optimize:

$$\arg\max_{\theta} \ \mathbb{E}_{x\sim\text{prompt distribution},\, y\sim\pi_\theta}[r_\phi(y \mid x) \ - \ \beta \cdot \text{Deviation}(\pi_\theta(\cdot \mid x),\ \pi_{\text{ref}}(\cdot \mid x))]\,,$$

where β is a hyperparameter.

Challenges: computational cost, high variance in estimates of gradients.

Many algorithms: REINFORCE, PPO, GRPO.

There is also a line of work that bypasses reward model creation and directly optimizes the language model from human feedback, e.g., Direct Preference Optimization (DPO).

Finally, here is an exercise. Post-training can help improve alignment. For instance, annotators in RLHF can flag down compliant responses to harmful requests. Can you think of scenarios where post-training can make the model worse?