12. Aligning Language Models (Advanced)

Guest lecture by Hyung Won Chung

Instruction Finetuning and Reinforcement Learning with Human Feedback (RLHF)

Outline

Rule-based systems

Classical machine learning

- Automatic learning

Deep learning: (self-)supervised learning

- Hand-designed features → Learned features

Deep learning: RLHF

- Hand-designed loss function → Learned loss function

Pretraining (general knowledge) → Instruction finetuing (Ability to respond to instructions)

→ Reward model training

→ Policy model training → Instruction finetuing (as a loop)

Instruction finetuing

- Instruction finetuing on a mixture of academic tasks
  - Example: Flan
  - Scaling with number of tasks and task diversity
- Instruction finetuing on user prompts of language model APIs
  - Example: InstructGPT
  - Academic tasks aren't reflected of how models are used in an API setting

Reward model training

Limitation of instruction finetuning: the target is the single correct answer. In RL, this is called "behavior cloning".

RL provides one way to use a learned objective.

Reward modeling (RM): ranking → score

Let p_{ij} be the probability that completion y_i is better than completion y_j

Bradley-Terry model: $\log \frac{p_{ij}}{1-p_{ij}} = r(x, y_i; \phi) - r(x, y_j; \phi)$

Pairwise ranking loss for K responses

$\text{loss}(\theta) = -\frac{1}{\binom{k}{2}} E_{x,y_w,y_l \sim D} [\log(\sigma(r_\theta(x, y_w)) - r_\theta(x, y_l))]$

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ, y_w is the preferred completion out of the pair of y_w and y_l, and D is the dataset of human comparisons.

Policy model training

Once we have a RM, we maximize the expected reward

$\max_\theta J(\theta) = \max_\theta \mathbb{E}_{(X,Y) \sim D_{\pi_\theta}} [r(X, Y; \phi)]$

We use iterative algorithms such as gradient ascent to solve this

$\theta := \theta + \alpha \nabla J(\theta)$

We add KL penalty to prevent over-optimization of the RM

Proximal Policy Optimization (PPO)

$\text{objective}(\phi) = E_{(x,y) \sim D_{\pi_\phi^{\text{RL}}}} [r_\theta(x, y) - \beta \log(\pi_\phi^{\text{RL}}(y \mid x) / \pi^{\text{SFT}}(y \mid x))] +$
$\gamma E_{x \sim D_{\text{pretrain}}} [\log(\pi_\phi^{\text{RL}}(x))]$

where π_ϕ^{RL} is the learned RL policy, π^{SFT} is the supervised trained model, and D_{pretrain} is the pretraining distribution.

- $r_\theta(x, y)$: expected reward for the new model
- $\log(\pi_\phi^{\text{RL}}(y \mid x) / \pi^{\text{SFT}}(y \mid x))]$: KL divergence to avoid going too far away from the original model
- $E_{x \sim D_{\text{pretrain}}} [\log(\pi_\phi^{\text{RL}}(x))]$: objective for GPT3 on the original data