Reinforcement Learning from Human Feedback

Presented by Daniel McKenzie

Mines

March 26, 2024

Overview

1. Large Language Models
2. Reinforcement Learning
3. RLHF for LLMs
4. More on RL

Table of Contents

A Model of Large Language Models

- High-level model of an (autoregressive) transformer-based LLM π_Θ:
  1) **Input:** A natural language prompt *e.g.* "The cat sat on"
  2) **Tokenizer/Embedding:** words in prompt mapped to vectors:

  $$v_{\text{the}}, v_{\text{cat}}, v_{\text{sat}}, v_{\text{on}} \in \mathbb{R}^{d_m}$$

     And stacked as rows in a matrix $x \in \mathbb{R}^{n_c \times d_m}$.
  3) **Decoding:** $\pi_\Theta(x)$ yields prob. distribution over tokens in model vocabulary.
  4) **Sampling:** sample new token t_{next} according to $\pi_\Theta(x)$.
  5) **Append and repeat:**
     - Add t_{next} to growing continuation/output y.
     - Append v_{next} to x.
     - Repeat $3 \to 5$. Stop when special token <|endoftext|> is generated.
- Denote by $y(x;\Theta)$ the output (**continuation**) generated by above procedure.
- For later use, we'll write probability distribution in (3) as $\pi_\Theta(t|x)$, *e.g.*

$$\pi_\Theta(t_1|x) = 0.0003,\ \pi_\Theta(t_2|x) = 0.005, \ldots \quad \text{and} \quad \sum_{i=1}^{n_T} \pi_\Theta(t_i|x) = 1$$

Back to intro to RLHF

Pre-training

- Pre-trained in a self-supervised manner:
  - Training data is text: "The cat sat on the mat". Tokenized to $t_1 = \text{The}, t_2 = \text{cat}, \ldots, t_7 = ..$
  - Notation: x_n = first n vectorized tokens of text, *e.g.*

$$x_3 = \begin{bmatrix} v_{\text{the}} \\ v_{\text{cat}} \\ v_{\text{sat}} \end{bmatrix} \in \mathbb{R}^{3 \times d_m}$$

  - $t_{n;\text{next}}$ is new token sampled according to $\pi(t|x_n)$.
  - Loss measures discrepancy between $t_{n;\text{next}}$ and t_{n+1}:

$$L(\Theta; V) = \sum_{k=1}^{n_c - 1} \left\{ \ell\left(t_{n;\text{next}}; t_{n+1}\right) = \|v_{t_{n;\text{next}}} - v_{t_{n+1}}\|_2^2 \right\}$$

- Train to choose LLM parameters Θ making loss small.

Table of Contents

A one slide introduction to RL

Throughout, we'll consider the application of driving a car.

- Divide model into **agent** and **system**.
- System is in state x_n *e.g. position and velocity of car*.
- Agent makes decision a_n *e.g. accelerate/decelerate* at discrete steps.
- System evolves to new state x_{n+1} *e.g. updated position and velocity of car*.
- Agent receives reward $r(x_{n+1})$ *e.g.*

$$r(x_{n+1}) = \begin{cases} +50 & \text{arrived at destination} \\ -1000 & \text{hit other car} \\ 0 & \text{otherwise} \end{cases}$$

- **Goal:** Maximize reward over N decision epochs: $R = \sum_{n=1}^{N} r(x_{n+1})$.
- Assuming x_{n+1} depends only on x_n, a_n, this framework is called a **Markov Decision Process**[1]

[1] *Markov Decision Processes* Puterman, Wiley-Interscience 1994

A one slide introduction to RL continued: Policies

- How do we (algorithmically) select decision/action a_n?
- A **policy** π is a map from states to actions.
- We'll focus on **parametrized**, **probabilistic** policies:
  - Parametrized: $\pi = \pi_\Theta$ where Θ are tunable (think neural network).
  - Probabilistic: If available actions at x_n are $a^1, \ldots, a^4$, π_Θ outputs

    $$\pi_\Theta(a^1|x_n) = 0.1,\ \pi_\Theta(a^2|x_n) = 0.3,\ \pi_\Theta(a^3|x_n) = 0.4,\ \pi_\Theta(a^4|x_n) = 0.2$$

    Select a_n according to $\pi_\Theta(a|x_n)$.
- The **learning** problem in Reinforcement Learning:

$$\max_{\Theta} \left\{ R(\Theta) = \sum_{n=1}^{N} r(x_{n+1}) \text{ where } a_{n+1} \sim \pi_\Theta(a|x_n) \right\}$$

Application: Simulated Robot Control

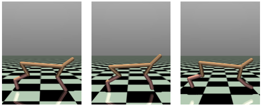

Figure: States of `half-cheetah`, available in `Gymnasium`[2]

- States: $x_n \in \mathbb{R}^{17}$ is position/velocity of segments.
- Action: actuate 6 joints; $a_n \in \mathbb{R}^6$.
- System evolution done by physics simulator: $x_{n+1} = F(x_n; a_n)$.
- Reward: $r(x_n) = \begin{cases} +10(N-n) & \text{exist right side of frame} \\ +1 & \text{stay upright} \\ 0 & \text{otherwise} \end{cases}$

[2] `https://gymnasium.farama.org/index.html`

Reward alignment

- For complicated tasks (*e.g.* scramble an egg) appropriate reward function unclear.
- For simpler tasks, maximizing reward function may result in undesirable behaviour[3].
- **Problem:** Know the right behaviour when you see it, but hard to articulate why.
- Reinforcement Learning from Human Feedback (**RLHF**) uses human feedback to learn an appropriate reward function.[4]

[3] `https://openai.com/research/faulty-reward-functions`

[4] *Deep reinforcement learning from human preferences* Christiano *et al* (2017)

Decoding as a RL problem

Back to LLM model

- Initial state $x_0 = x \in \mathbb{R}^{4 \times d_m}$.
- Policy given by the LLM $\pi_\Theta(t|s_n)$, evolution is simple.
  - Sample $t_{n+1} \sim \pi_\Theta(t|x_n)$.
  - Vectorize to $v_{t_{n+1}}$, append to prompt: $x_{n+1} = \begin{bmatrix} x_n \\ v_{t_{n+1}} \end{bmatrix}$.
  - Also append to continuation: $y_{n+1} = [y_n, t_{n+1}]$.
  - Receive reward $r(y_{n+1})$.
  - Continue until N tokens generated or <|endoftext|> is generated.
- What is the reward? Consider these prompts:
  1) "Translate into French: Today is a fine day"
  2) "Give me a summary of [insert long text here]"
  3) "What were the causes of WW2?"
- Reward heavily dependent on completed continuation y_N.

Table of Contents

Overview of RLHF

- Pretrain LLM in self-supervised manner, get $\pi_{\Theta^{\mathrm{pt}}}$.
- Collect pairs of continuations given same prompt $\{y^0(x_i;\Theta^{\mathrm{pt}}), y^1(x_i;\Theta^{\mathrm{pt}})\}_{i=1}^{M}$.
- Ask humans to select a winner (w) and loser (ℓ) from each pair.
- Train a **reward model** to maximize difference in score between winner and loser:

$$\max_{\Phi} \sum_{i=1}^{M} \left[r_\Phi(y^w(x_i;\Theta)) - r_\Phi(y^\ell(x_i;\Theta)) \right].$$

- Use r_Φ as the reward function. Use RL to fine-tune π_Θ.

Example: Text summarization

- Stiennon *et al*[5] fine-tune GPT-3 models for summarization.
- Use Reddit TL;DR dataset[6].
- Fix length of summary to 48 tokens.
- Recruited labelers via Upwork, Lionbridge, Scale[7]
- Aquired 60k human comparisons @ \$15 per hour.
- Use r_Φ as above but with a **Kullback-Leibler** regularizer:

$$r(\Theta) = r_\Phi(\Theta) + \mathbb{D}_{\mathrm{KL}}(\pi_\Theta | \pi_{\Theta^{\mathrm{pt}}})$$

- Use standard RL algorithm; Proximal Policy Optimization[8]
- RL step takes 320 GPU-days.

---

[5] *Learning to summarize from human feedback* Stiennon *et al*, NeuIPS 2020

[6] *TL;DR: Mining Reddit to learn automatic summarization* Völske *et al*, 2017

[7] Similar websites include clarifai, Amazon Mechanical Turk

[8] *Proximal policy optimization algorithms* Schulman *et al* 2017

Sample data

[r/dating_advice] **First date ever, going to the beach. Would like some tips**		
Hey Reddit! I (20M) would like some tips, because I have my first ever date tomorrow (although I've had a gf for 3 years, but no actual dating happened), and we're going to the beach. I met this girl, we have mutual friends, at a festival a few days ago. We didn't kiss, but we talked, held hands, danced a bit. I asked her to go on a date with me, which was super hard as it is the first time I've asked this to anybody. What I mean to say is, it's not like a standard *first* date because we already spent some time together. I'm really nervous and excited. I'm going to pick her up tomorrow, we're cycling to the beach which will take 30 minutes, and then what? I'm a bit scared. Should I bring something (the weather, although no rain and sunny, is not super so no swimming), should we do something. I'd like all the tips I can get. Thanks!		
Human written reference TL;DR	**6.7B supervised model**	**6.7B human feedback model**
First date after 3 years in a relationship, going to the beach, terrified. What to bring with me, what to do?	Going on a date with a girl I met a few days ago, going to the beach. What should I bring, what should we do?	Going on my first ever date tomorrow, cycling to the beach. Would like some tips on what to do and bring. I'm a bit nervous and excited. Thanks!

Table: Example of post and samples on the TL;DR dataset, chosen to be particularly short. Taken from Stiennon *et al*. For more examples see
`https://openai.com/research/learning-to-summarize-with-human-feedback`

Table of Contents

Q-learning

- Consider a simple MDP with 4 states, (same) 3 action at each state.
- **Q-function** determines the quality of action a at state x:

$$Q(x, a) = r(a) + \sum_{x'} \mathbb{P}[x'|x, a] \max_{a'} Q(x'a')$$

- Q-values can be represented in a 4×3 table.
- Can compute Q table via **exploration**.
- Given (a good approx to) Q-table, optimal policy is greedy approach.