20260519
Eric Jang

AlphaGo Podcast Errata

In the podcast, I suggested that multi-step LLM RL has higher variance than single-step LLM RL due to "variance growing quadratically with length". This statement turns out to be not quite correct; I would like to issue an errata to make that section of the tutorial more clear, and apologize for the error on my end.

Let sequence $y = (y_1, y_2, \ldots, y_T)$, prompt context x, and an autoregressive transformer in which the probability of the sequence is the product of conditional probabilities (each token y_t conditioned on the prompt x and prior tokens $y_{<t}$).

Then we have

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

Applying a log and gradient operator to both sides, we have

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

If we treat the whole sequence $y = (y_1, y_2, \ldots, y_T)$ as one action, the "single-step" policy gradient (REINFORCE) estimator is:

$$\hat{g}_{\text{single}} = R(y) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})$$

If we treat each token $y_1, y_2, \ldots, y_T$ as a separate "action", the "multi-step" policy gradient estimator is:

$$\hat{g}_{\text{multi}} = \sum_{t=1}^{T} G_t \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})$$

Where G_t is the score for each action, which is the "return-to-go" from time t (sum of rewards starting from time step t).

Scenario A: one sparse terminal reward for the whole sequence

If the reward is just a final binary 0 or 1 at the end (pass or fail, as is common in LLM RL environments), then $G_t = R(y)$ for all t and therefore $\hat{g}_{\text{single}} = \hat{g}_{\text{multi}}$, so treating

LLMs as single-step or multi-step is irrelevant here. That being said, in practice one uses slightly better RL algorithms than a naive policy gradient (e.g. baselines)

Scenario B: per-token rewards

What happens if we have a way to score intermediate rewards, $r_1, r_2, \ldots, r_T$, as mentioned ("process supervision") in the podcast? In practice for LLMs, this is hard as we don't have a great way to score individual tokens, so we usually score entire sentences.

The single-step estimator multiplies every "gradient of log-prob" term with a summation of the rewards for the episode.

$$\hat{g}_{\text{single}} = (\sum_{k=1}^{T} r_k)(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}))$$

The multi-step estimator has a similar "summation of rewards" term for each action at time t, but only starting from k=t, not from k=1.

$$\hat{g}_{\text{multi}} = \sum_{t=1}^{T} (\sum_{k=t}^{T} r_k) \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})$$

Under per-token rewards, the single-step estimator $\hat{g}_{\text{single}}$ actually has *higher* variance than $\hat{g}_{\text{multi}}$. Past rewards $r_{k<t}$ do not depend on current action y_t, so one makes the credit assignment problem *harder* when adding past rewards to the score G_t of action at time t.

This was my critical mistake in the podcast: with per-step rewards, multi-step should have lower variance than naive whole-sequence single-step REINFORCE, not higher

Okay, if multi-step RL actually has lower variance than single-step RL, why do LLM labs do single-step RL?

It comes down to the fact that in LLM RL, it's hard to compute per-token rewards, so they end up scoring the whole trajectory, i.e. we are in the "sparse terminal reward" scenario. In LLMs, per-token rewards are often misspecified, so we don't gain much by trying to do the "only score each token with its reward-to-go G_t" .

"Was the answer helpful / a valid solution / did the code tests pass" are often sequence-level properties, i.e. sparse terminal conditions that can be computed only after the full sequence is generated.

If we knew of a way to set up per-token rewards for LLM tasks we care about, then multi-step RL would greatly help with reducing variance.

How does credit assignment work then across tokens, if we do single-step RL?

The environment gives one score per full answer, and then computes and advantage $A(y)$ on that sequence and this advantage is broadcasted to all of the score function updates across every token in the sequence. In LLM RL, there *are* some per-token terms like KL constraints (don't drift too far from the reference policy).

You mentioned quadratic variance with sequence length in the podcast. What was that about?

Recall the basic identity $\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y)$

Let's use shorthand $S_t = \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})$ for the "gradient of log-prob of token" term. We have

$$\hat{g}_{\text{single}} = R \sum_{t=1}^{T} S_t$$

$$\text{Var}(R \sum_{t=1}^{T} S_t) = \sum_{t=1}^{T} \text{Var}(RS_t) + 2 \sum_{i<j}^{T} \text{Cov}(RS_i, RS_j)$$

There are T variance terms and T(T-1)/2 covariance terms, which is $O(T^2)$ in the worst case (e.g. every token has positive covariance with other terms in the sequence).

Note that this depends on the sequence length $y = (y_1, y_2, \ldots, y_T)$, *not* whether we are using multi-step or single-step RL formulation to achieve a policy gradient.

The "plain English" explanation here is that credit assignment is quadratically hard in the worst case for long LLM generations.