Study Material - Youtube

Document Information

- **Generated:** 2025-08-26 08:05:18
- **Source:** https://www.youtube.com/watch?v=Y2KZ_zDGhKw
- **Platform:** Youtube
- **Word Count:** 1,787 words
- **Estimated Reading Time:** ~8 minutes
- **Number of Chapters:** 3
- **Transcript Available:** Yes (analyzed from video content)

Table of Contents

Video Overview

This video lecture, titled "Expressing an AR-LM as RL policy," is a segment from the "Mathematical Foundations of Generative AI" course. The instructor, Prof. Prathosh A P, provides a detailed explanation of how to conceptualize and formally define an Autoregressive Language Model (AR-LM) within the framework of Reinforcement Learning (RL). This re-framing is a critical prerequisite for applying powerful RL optimization techniques, such as policy gradients, to fine-tune language models based on complex objectives, a cornerstone of modern systems like ChatGPT.

Learning Objectives

Upon completing this study module, you will be able to: - Understand the conceptual mapping between an autoregressive language model and a reinforcement learning policy. - Formally define the key components of an RL problem—**states**, **actions**, and **trajectories**—in the context of text generation by a language model. - Recognize that the language model's conditional probability distribution is equivalent to the RL policy function, $\pi_\theta(a_t|s_t)$. - Appreciate why this framework is essential for applying policy gradient methods to optimize language models.

Prerequisites

To fully grasp the concepts in this lecture, students should have a foundational understanding of: - **Reinforcement Learning (RL):** Basic concepts including agent, environment, state, action, policy, and trajectory. - **Policy Gradient Methods:** Familiarity with the core idea of policy gradient theorems, particularly the objective function and its gradient as seen in algorithms like REINFORCE. - **Autoregressive Language Models (AR-LMs):** Knowledge of how these models generate text sequentially, by predicting one token at a time based on the preceding sequence.

Key Concepts Covered in This Video

- Policy Gradient Theorem
- Autoregressive Language Model (AR-LM) as a Policy
- State Definition in Language Generation
- Action Definition in Language Generation
- Trajectory Construction for Text Generation

Expressing an Autoregressive Language Model (AR-LM) as an RL Policy

This section details the central theme of the lecture: how to interpret and express an autoregressive language model as a reinforcement learning policy. This allows us to leverage the powerful optimization tools from RL to enhance the performance of language models.

The Bridge: The Policy Gradient Theorem

At the beginning of the lecture (00:11), the instructor presents the policy gradient theorem, which is the mathematical foundation for this discussion. It provides a way to update the parameters of a policy to maximize the expected reward.

Mathematical Formulation

The gradient of the objective function $J(\theta)$ with respect to the policy parameters θ is given by:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{W} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t) \right]$$

Intuitive Breakdown of the Formula

- $\nabla_\theta J(\theta)$: This is the **direction of steepest ascent** for our objective function. We want to adjust our policy parameters θ in this direction to get more rewards.
- $\mathbb{E}_{\tau \sim p_\theta(\tau)}$: This denotes the **expectation over many possible trajectories** (τ) that are generated by following our current policy π_θ. In practice, we approximate this by sampling a batch of trajectories and averaging the results.
- $\sum_{t=0}^{W}$: This is the sum over all the steps in a single trajectory, from the start ($t = 0$) to the end ($t = W$).
- $\pi_\theta(a_t|s_t)$: This is the **policy**. It's a function, parameterized by θ, that gives the probability of taking action a_t when in state s_t.
- $\nabla_\theta \log \pi_\theta(a_t|s_t)$: This is the **score function**. It's a vector that tells us how to change the parameters θ to make the action a_t (which we actually took) more likely.
- $A^{\pi_\theta}(s_t, a_t)$: This is the **Advantage function**. It measures how much better the action a_t was compared to the average action one could have taken from state s_t.
  - If $A^{\pi_\theta} > 0$, the action was **better than average**. The update will "encourage" this action by increasing its log-probability.
  - If $A^{\pi_\theta} < 0$, the action was **worse than average**. The update will "discourage" this action by decreasing its log-probability.

  Key Takeaway: The policy gradient theorem provides a recipe for improving a policy: run the policy to generate some experiences (trajectories), see which actions led to better-than-average outcomes, and adjust the policy to make those actions more probable in the future.

Intuitive Foundation: Why Frame an LM as a Policy?

To apply the policy gradient theorem to a language model, we must first establish a clear analogy between the two. The instructor begins this process at 00:46.

- **An AR-LM's Task:** An autoregressive language model's fundamental job is to predict the next token in a sequence, given all the previous tokens. At each step, it produces a probability distribution over the entire vocabulary and we sample a token from it.
- **An RL Policy's Task:** A policy's job is to decide which action to take from a given state. It does this by defining a probability distribution over all possible actions.

The parallel is striking: - The **current sequence of text** is the **state (s_t)**. - The **choice of the next token** is the **action (a_t)**. - The **language model itself**, which provides the probability of choosing each possible next token, is the **policy (π_θ)**.

This conceptual link is powerful because it allows us to think about "good" and "bad" sequences of text in terms of RL rewards. If a generated sentence is helpful, coherent, and safe, it should receive a high reward. The policy gradient algorithm can then be used to adjust the language model's parameters (θ) to make it more likely to generate such high-reward sentences.

Formalizing the Analogy: Defining RL Components for LMs

The instructor systematically formalizes this analogy from 01:09 onwards.

The Policy π_θ is the Language Model P_θ

The core of the mapping lies in equating the language model's probability distribution with the RL policy.

- **AR-LM Probability (01:44):** An AR-LM models the conditional probability of the next token x_t given the history of previous tokens $x_{<t}$:

$$P_\theta(x_t|x_{<t}) = P_\theta(x_t|x_0, x_1, \dots, x_{t-1})$$

- **RL Policy (02:22):** An RL policy models the probability of an action a_t given a state s_t:

$$\pi_\theta(a_t|s_t)$$

By setting the LM's output as the policy, we establish the equivalence:

$$\pi_\theta(a_t|s_t) \equiv P_\theta(x_t|x_{<t})$$

where the action a_t is the chosen token x_t, and the state s_t is the history of tokens $x_{<t}$. The parameters θ of the neural network are the same for both the LM and the policy.

Constructing a Trajectory τ

To use the policy gradient formula, we need to generate trajectories. A trajectory is a sequence of states and actions, $\tau = (s_0, a_0, s_1, a_1, \dots)$. The instructor details this process from 02:59.

The generation process can be visualized as follows:

```
flowchart TD
    A["Start with an Initial Prompt<br/>(e.g., 'Write a poem about...')"] --> B{"This prompt is the ini
    B --> C["Feed <b>s<sub>0</sub></b> into the LM (Policy  <sub> </sub>)"];
    C --> D["LM outputs a probability distribution over the next token"];
    D --> E["Sample the first token from the distribution<br/>This is the first action: <b>a<sub>0</sub></b>"];
    E --> F{"Form the new state <b>s<sub>1</sub></b> by concatenating:<br/><b>s<sub>1</sub> = [s<sub>0</sub>
    F --> G{"Feed <b>s<sub>1</sub></b> into the LM (Policy  <sub> </sub>)"};
    G --> H["Sample the second token<br/>This is the second action: <b>a<sub>1</sub></b>"];
    H --> I{"Form the new state <b>s<sub>2</sub></b>:<br/><b>s<sub>2</sub> = [s<sub>1</sub>, a<sub>1</sub></b>
    I --> J["...repeat until an end-of-sequence token is generated or max length is reached."];
```

This flowchart illustrates the autoregressive process of generating a trajectory by iteratively using the language model as a policy to select actions (tokens) and update the state (text history).

Mapping Summary Table

This table provides a clear, side-by-side comparison of the terminology.

Reinforcement Learning (RL) Term	Language Model (LM) Counterpart	Description in LM Context
State (s_t)	Prompt + Generated History	The sequence of tokens fed into the LM at timestep t. (e.g., `[prompt, x_0, ..., x_{t-1}]`)
Action (a_t)	Next Generated Token (x_t)	The token sampled from the LM's output distribution at timestep t.
Policy (π_θ)	The Language Model Itself (P_θ)	The function that takes a state (history) and outputs a probability for each action (next token).
Trajectory (τ)	Full Generated Sequence	The complete interaction: `(s_0, a_0, s_1, a_1, ..., s_W, a_W)`.
Parameters (θ)	Model Weights	The neural network weights of the language model that we want to optimize.

Self-Assessment for This Video

Test your understanding of the concepts covered in this lecture.

1. **Question:** In the context of framing a language model as an RL agent, what constitutes the "state" s_t?
   - **Answer:** The state s_t is the sequence of all tokens up to that point, which includes the initial prompt and all tokens generated by the model from timestep 0 to $t-1$.
2. **Question:** What is considered an "action" a_t?
   - **Answer:** The action a_t is the specific token that the language model chooses or samples to be the next token in the sequence at timestep t.
3. **Question:** How is the policy $\pi_\theta(a_t|s_t)$ mathematically defined in terms of the language model?
   - **Answer:** The policy $\pi_\theta(a_t|s_t)$ is defined as the conditional probability assigned by the language model to the next token, i.e., $\pi_\theta(a_t|s_t) \equiv P_\theta(x_t|x_{<t})$.
4. **Application Exercise:** Describe the first three steps (from s_0 to s_2) of generating a trajectory for the prompt "The capital of France is".
   - **Step 1 (State s_0):** The state is the prompt itself: `("The", "capital", "of", "France", "is")`.
   - **Step 2 (Action a_0):** The model is fed s_0 and generates a distribution. We sample from it and get the token "Paris". So, a_0 = "Paris".
   - **Step 3 (State s_1):** The new state is the concatenation of the previous state and action: `("The", ..., "is", "Paris")`. This becomes the input for the next step. The process continues to generate the next action, a_1.

Key Takeaways from This Video

- **Primary Insight:** An autoregressive language model can be directly interpreted as a reinforcement learning policy. This is the foundational step for applying RL-based optimization.
- **State and Action Mapping:** In this framework, states are the history of generated text, and actions are the next tokens to be generated.
- **Trajectory Generation:** A trajectory is built by iteratively feeding the current text sequence (state) into the LM, sampling a new token (action), and appending it to the sequence to form the next state.
- **Path to Optimization:** By framing the LM as a policy, we can use the policy gradient theorem to update the model's parameters (θ) to maximize a reward function, enabling advanced fine-tuning techniques like Reinforcement Learning from Human Feedback (RLHF).