Introduction to Hyperparameters in Large Language Models

Haoyang Han

2025-02-13

When working with large language models (LLMs) like OpenAI's GPT, Google's PaLM, Anthropic's Claude, or DeepMind's Gemini, understanding how **hyperparameters** affect model behavior is crucial for fine-tuning outputs. Among the various hyperparameters, **temperature** is one of the most significant as it controls the **randomness** or **creativity** of the model's responses.

This document explains key hyperparameters used in LLMs, including **temperature**, **top-k**, **top-p**, and **penalties** (frequency and presence). We'll provide a mathematical foundation for each parameter and discuss what their values mean in practice.

1 1. Temperature (T)

Temperature is a hyperparameter that controls the **randomness** in the model's predictions. A higher temperature makes the model's responses more random, while a lower temperature makes the output more deterministic and focused.

Mathematical Formula for Adjusting Logits:

$$\text{Logits}_{\text{adjusted}} = \frac{\text{Logits}}{T}$$

Softmax with Temperature:

$$P(w_i) = \frac{e^{\frac{\text{Logits}_i}{T}}}{\sum_j e^{\frac{\text{Logits}_j}{T}}}$$

What Does Temperature Mean?

- **Low Temperature (T ¡ 1.0)**: The model's output becomes more **deterministic**, focusing on the most likely predictions. It's ideal for tasks requiring factual, less creative responses.

- **High Temperature (T ¿ 1.0)**: The output becomes more **random** and creative, producing diverse but potentially less coherent text. This is useful for tasks like brainstorming or creative writing.
- **T = 1.0**: The default setting, offering a balance between creativity and coherence.

2 2. Top-k Sampling

Top-k sampling restricts the possible next tokens to the top **k** most likely candidates. This helps in focusing the output on a smaller, more probable set of options, avoiding excessive randomness.

Formula for Top-k Sampling:

$$P(w_i) = \frac{e^{\text{Logits}_i}}{\sum_{j \in \text{Top-}k} e^{\text{Logits}_j}}$$

This ensures the model generates tokens from only the most likely set of words, improving the quality and coherence of the output.

3 3. Top-p (Nucleus) Sampling

Top-p sampling, also known as nucleus sampling, works by considering a subset of tokens whose cumulative probability exceeds a specified threshold p.

Formula for Top-p Sampling:

$$P(w_i) = \frac{e^{\text{Logits}_i}}{\sum_{j \in \text{Top-p subset}} e^{\text{Logits}_j}}$$

Top-p sampling dynamically adjusts the number of tokens considered, balancing diversity and quality in a more adaptive way compared to top-k.

4 4. Frequency Penalty (F)

The frequency penalty discourages the model from repeating words it has already generated, thus improving the diversity of the generated text.

Formula for Frequency Penalty:

$$\text{Logits}_{\text{penalized}} = \text{Logits}_i - F \cdot \text{count}(w_i)$$

A higher frequency penalty makes the model more likely to avoid repeating words.

5 5. Presence Penalty (P)

The presence penalty encourages the model to introduce new words into the generated text, helping avoid over-repetition of the same concepts or tokens.

Formula for Presence Penalty:

$$\text{Logits}_{\text{penalized}} = \text{Logits}_i - P \cdot \mathbf{1}[\text{token exists in sequence}]$$

This penalty ensures the model explores new tokens and ideas, contributing to more varied outputs.

6 Summary Table: Comparison of Hyperparameters

Hyperparameter	**Formula**	**Effect**
Temperature (T)	$\frac{\text{Logits}}{T}$	Controls random
Top-k Sampling	$\frac{e^{\text{Logits}_i}}{\sum_{j \in \text{Top-}k} e^{\text{Logits}_j}}$	Restricts the mo
Top-p Sampling	$\frac{e^{\text{Logits}_i}}{\sum_{j \in \text{Top-p subset}} e^{\text{Logits}_j}}$	Ensures diversity
Frequency Penalty (F)	$\text{Logits}_{\text{penalized}} = \text{Logits}_i - F \cdot \text{count}(w_i)$	Reduces the prob
Presence Penalty (P)	$\text{Logits}_{\text{penalized}} = \text{Logits}_i - P \cdot \mathbf{1}[\text{token exists in sequence}]$	Encourages new

Hyperparameter	**Formula**	**Effect**
Temperature (T)	$\text{Logits}_{\text{adjusted}} = \frac{\text{Logits}}{T}$	Controls random
Top-k Sampling	$\text{Logits}_{\text{adjusted}} = \frac{e^{\text{Logits}_i}}{\sum_{j \in \text{Top-}k} e^{\text{Logits}_j}}$	Restricts the mo
Top-p Sampling	$P(w_i) = \frac{e^{\text{Logits}_i}}{\sum_{j \in \text{Top-p subset}} e^{\text{Logits}_j}}$	Ensures diversity
Frequency Penalty (F)	$\text{Logits}_{\text{penalized}} = \text{Logits}_i - F \cdot \text{count}(w_i)$	Reduces the prob
Presence Penalty (P)	$\text{Logits}_{\text{penalized}} = \text{Logits}_i - P \cdot \mathbf{1}[\text{token exists in sequence}]$	Encourages new

Table 1: Comparison of Hyperparameters

Conclusion

Understanding and tuning these hyperparameters can drastically influence the behavior of language models. **Temperature** stands out as one of the most important parameters for controlling output creativity versus coherence. By manipulating **top-k**, **top-p**, and the penalty parameters, we can further refine and shape the model's behavior to suit specific use cases.

For **creative tasks**, higher temperatures and a relaxed top-k/top-p setting may be preferred. For **factual or structured outputs**, lower temperatures and higher penalties on repetition are often better.

By adjusting these hyperparameters, you can fine-tune models to meet the needs of a wide variety of applications, from creative writing to technical documentation.