Evaluating Inference-Time Adaptive Temperature for Mathematical Reasoning in LLMs

Mehul Bafna

Yeshiva University

December 21, 2024

Overview

Softmax Function

Definition:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

where:

- z_i is the logit (raw output) for token i
- n is the vocabulary size

Properties:

- Converts logits to probabilities (0 to 1)
- Sum of probabilities equals 1
- Preserves relative ordering of inputs

Temperature in Language Models

Temperature-Scaled Softmax:

$$\text{softmax}(z_i, T) = \frac{e^{z_i/T}}{\sum_{j=1}^{n} e^{z_j/T}}$$

Role of Temperature (T):

- Controls the "sharpness" of probability distribution
- $T > 1$: Makes distribution more uniform (more random)
- $T < 1$: Makes distribution more peaked (more deterministic)
- $T = 1$: Standard softmax behavior

Beta and Temperature Relationship

Beta Definition:

$$\beta = \frac{1}{T}$$

Properties:

- β is inverse temperature
- Higher β means lower temperature (more focused)
- Lower β means higher temperature (more diverse)
- Using β simplifies control calculations

In Practice:

- $\beta > 1$: More conservative predictions
- $\beta < 1$: More exploratory predictions
- $\beta = 1$: Standard softmax behavior

Why Adaptive Temperature?

Key Insights:

- Mathematical reasoning requires precise control over token generation
- Static temperature scaling lacks context awareness
- Token-level entropy indicates model uncertainty
- Dynamic adaptation could improve reasoning precision

Goals:

- Dynamic temperature adjustment based on prediction uncertainty
- Maintain coherence in mathematical operations
- Balance between exploration and exploitation

System Architecture

Key Components:

- TokenMetrics: Stores per-token generation statistics
- AdaptiveEntropyTemperature: Custom LogitsProcessor
- ModelManager: Handles model lifecycle
- Evaluation System: Analyzes mathematical correctness

Core Features:

- Real-time entropy monitoring
- Dynamic temperature scaling
- Comprehensive metric tracking
- Threading for non-blocking generation

TokenMetrics Implementation

Tracked Metrics Per Token:

- Token value and timestamp
- Entropy of prediction distribution
- Applied beta (inverse temperature)
- Generation timestamps for performance analysis

Data Management:

- Pandas DataFrame integration
- Real-time metric accumulation
- Statistical analysis capabilities

Control System Design

Beta Control Function:

$$\beta(H) = \begin{cases} 1.0 & \text{if } H \leq H_{\text{threshold}} \\ \max(p(H), \beta_{\text{min}}) & \text{if } H > H_{\text{threshold}} \end{cases}$$

where:

- H is token distribution entropy
- $H_{\text{threshold}} = 0.6$
- $\beta_{\text{min}} = 0.5$ ($T = 2$)

Polynomial Control Function

Fourth-Order Polynomial:

$$p(H) = -1.791H^4 + 4.917H^3 - 2.3H^2 + 0.481H - 0.037$$

Implementation Details:

- Direct numpy polyval implementation
- Puts and upper bound at β_{min}

GSM8K Benchmark Setup

Dataset Configuration:

- High school math word problems
- Sample size: 200 problems
- Evaluation metrics tracked per problem

Model Configuration:

- Model: Gemma 2-2B Instruction-tuned
- Max new tokens: 200
- Top-p: 0.9
- Top-k: 40

Evaluation Metrics

Problem-Level Metrics:

- Numerical accuracy with tolerance
- Solution step presence detection
- Answer completeness check
- Response length analysis

Token-Level Analysis:

- Per-token entropy tracking
- Per-token Beta value tracking
- Generation time per sample

Performance Results

Metric	Baseline	Adaptive
Numerical Accuracy	50.0%	45.5%
Solution Steps	100.0%	100.0%

n = 200

Token Statistics:

- 21% tokens used adaptive temperature
- Average entropy: 0.327
- Average beta: 0.993

Future Directions

Methodology Improvements:

- Advanced entropy moment analysis
- Dynamic threshold adaptation
- Integration with reward modeling
- Polynomial coefficient optimization
- Multi-token lookahead strategies

Implementation Goals:

- Improved benchmark performance
- Enhanced evaluation metrics
- Task-specific adaptation strategies
- Alternative control functions

Conclusion

Key Contributions:

- Implemented complete entropy-based temperature control
- Developed comprehensive evaluation framework
- Provided detailed token-level analysis

Lessons Learned:

- Current implementation shows mixed results
- Token-level adaptation affects 21% of generations
- Further optimization needed for better performance