LLMs in the Nesa Ecosystem

LLMs, such as OpenAI's GPT (Generative Pre-trained Transformer), are advanced AI systems designed to understand, generate, and interact with human language in a way that is both contextually aware and highly nuanced.

These models are trained on vast amounts of text data, allowing them to perform a wide range of language-related tasks such as translation, summarization, question-answering, and creative content generation.

Mathematical Formulation and Notations

We now introduce the formulation and notations we will be using to describe our LLM inference algorithm. In essence, LLMs model a language sequence (w1 , w2 , · · · wN) by computing the conditional distributions

The initial phase of deploying an LLM on Nesa involves offline training. This process is conducted independently from Nesa to ensure optimal train- ing conditions and efficiency. The model constructor focuses on creating a robust and accurate LLM, utilizing extensive datasets and computational resources. This phase culminates in a fully trained model ready for integration with Nesa's infrastructure.

$$\log p(w_1, w_2, \cdots w_N) = \sum_{i=1}^{N} \log p(w_i | w_1, w_2 \cdots w_{i-1}) \tag{5.1}$$

For convenience, we write $p^i(w_i) = p(w_i | w_1, w_2 \cdots w_{i-1})$, we also write context $C^i = (w_1, w_2 \cdots, w_i)$ for $i < N$. In an LLM task, the inference code will be given C^K, and the model is asked to predict continuations $(w_{K+1} \cdots, C_N)$. The prediction is done by ancestral sampling from each $p^{(}w_i)$, where $i = K+1, K+2 \cdots N$.

Uploading Model Parameters

Once the LLM is trained, Nesa provides an upload pipeline to upload its model parameters to decentralized storage solutions IPFS or Arweave. This makes the model accessible to Nesa nodes. The uploaded package includes not only the model parameters but also a comprehensive configuration file detailing the model's architecture and running environment requirements.

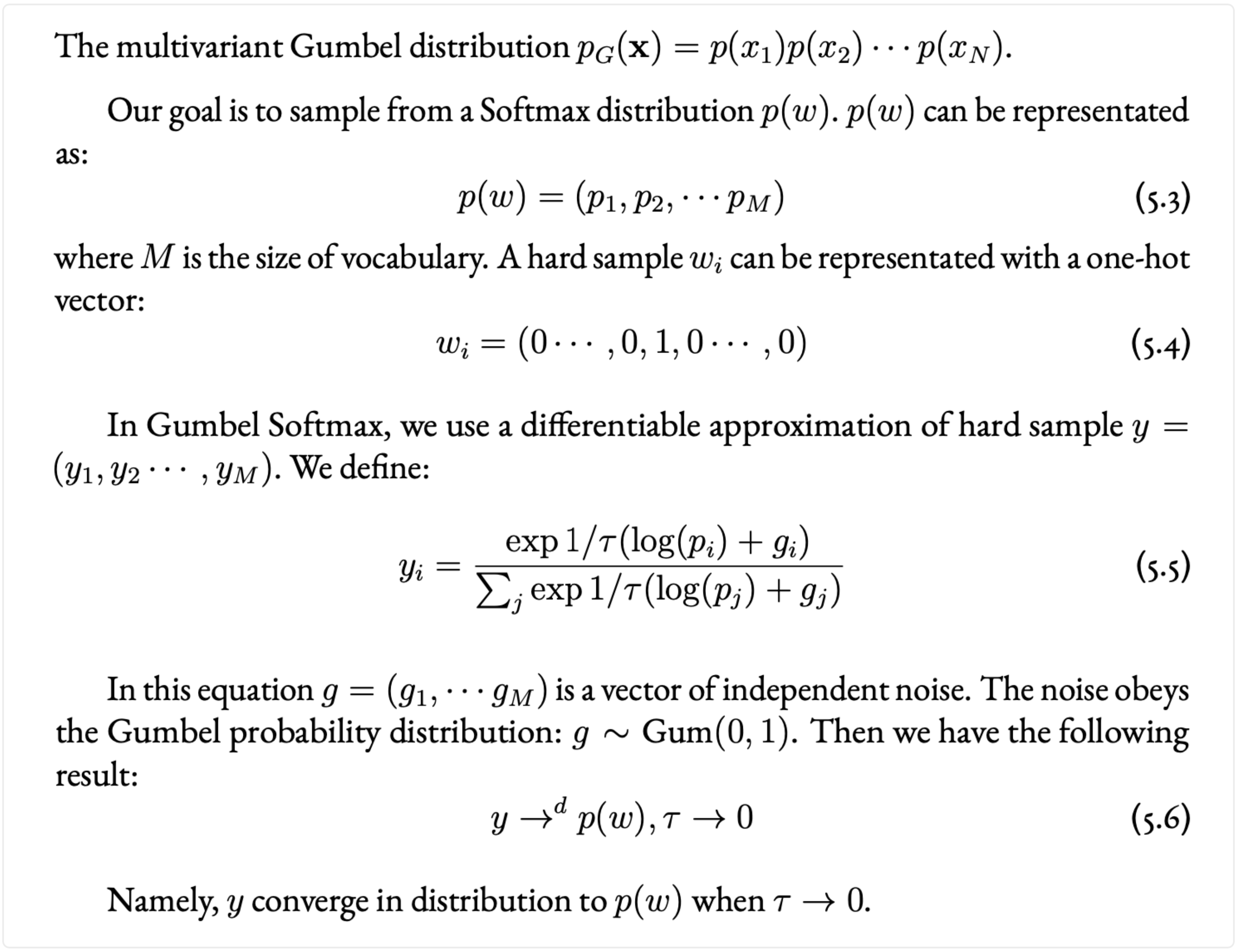
Inference Code Integration

The core of the on-chain LLM functionality lies in the inference code. This code is responsible for generating natural language responses based on given prompts. To achieve this, the code employs a sequence of Gumbel noises as a deterministic factor in the Softmax sampling process. By using these noises, the model can produce consistent outputs across different nodes, given the same prompt. This consistency is crucial for maintaining the integrity and reliability of the decentralized network.

Given a prompt P = w1, w2, · · · wK where K < N, the model generates a continuation by sampling from distribution p(wK+1, · · · wN |P) = pK+1(wK+1) · · · pN (wN). In LLMs, the sampling of each pi(wi) is through Softmax sampling. However, it is well known that the softmax distribution can be approximately reparameterized using Gumbel distribution.

We briefly introduce Gumbel softmax and Gumbel distribution. The standard Gumbel distribution Gum(0, 1) is defined with probability density function:

$$p_G(x) = \exp -(x + e^{-x}) \tag{5.2}$$



The multivariant Gumbel distribution $p_G(\mathbf{x}) = p(x_1)p(x_2)\cdots p(x_N)$.

Our goal is to sample from a Softmax distribution $p(w)$. $p(w)$ can be representated as:

$$p(w) = (p_1, p_2, \cdots p_M) \tag{5.3}$$

where M is the size of vocabulary. A hard sample w_i can be representated with a one-hot vector:

$$w_i = (0 \cdots, 0, 1, 0 \cdots, 0) \tag{5.4}$$

In Gumbel Softmax, we use a differentiable approximation of hard sample $y = (y_1, y_2 \cdots, y_M)$. We define:

$$y_i = \frac{\exp 1/\tau(\log(p_i) + g_i)}{\sum_j \exp 1/\tau(\log(p_j) + g_j)} \tag{5.5}$$

In this equation $g = (g_1, \cdots g_M)$ is a vector of independent noise. The noise obeys the Gumbel probability distribution: $g \sim \mathrm{Gum}(0, 1)$. Then we have the following result:

$$y \to^d p(w), \tau \to 0 \tag{5.6}$$

Namely, y converge in distribution to $p(w)$ when $\tau \to 0$.

Now looking back to our problem. We want to sample from wi ~pi(w). Roughly speaking,there is a deterministic function f as above, such that wi′ =f(w1,w2···,wi−1,gi) when wi ~ pi(w) when τ → 0, where gi is an independent Gumbel noise at timestep i.

a In this way, if the sequence of Gumbel noises (εK+1, · · · , εN) is predetermined, the sequence (wK+1, · · · , wN) will be determined. So each node can generate the continuation given the pre-given Gumbel noises, making them easy to reach a consensus.

Consensus Among Nodes

To achieve consensus among nodes, the consensus code is included by the model constructor alongside the inference code. A series of presets for default majority response is provided by Nesa, with customizable consensus mechanisms available to the model constructor based on preference and nature of the model and query request.

Enhancing Deterministic Output Generation

The use of Gumbel noises in the Softmax sampling process is a key innovation of Nesa's inference system. It allows the LLM to generate deterministic outputs, which is a critical requirement for consensus in a decentralized environment. This approach ensures that all nodes, given the same input prompt and noise sequence, will produce identical outputs and the deterministic nature of Nesa's inference here is vital for the consistency and reliability of LLM responses across the network.

Nesa's unique combination of offline training, decentralized storage, on-chain deployment, Gumbel noise-based inference for the reconstruction of the exact sequence of inferential steps by any participating node, and unique method of consensus allows LLM inference requests on the network to enjoy a unique level of robustness, reliability, security, and execution speed.

The adaptability of consensus protocols on Nesa is further strengthened with threshold cryptography that enables a subset of nodes to generate a valid group signature, optimizing network latency, and throughput and protecting against threats such as Sybil attacks and node manipulations.

Our system is bolstered by the concept of ‘Convergence through Divergent Paths’, entailing that even though the nodes in the system may execute the inference process independently, the application of Gumbel noise ensures that the divergent paths converge to a single, deterministic result. This sets a new paradigm for how decentralized artificial intelligence systems can operate with high availability and determinism.

Previous
Composite 2: Cryptography

Next
Overview of $NES

Last updated 1 month ago