RESUMO - ENGENHARIAS, TECNOLOGIAS E CIÊNCIAS EXATAS

POLAR QUANTIZATION APPLIED TO KEY-VALUE VECTOR COMPRESSION IN CACHE MEMORIES OF LARGE LANGUAGE MODELS

Igor Duarte (igorduarte318@gmail.com)

Hugo Hoffmann Borges (hugo.borges@afya.com.br)

The quantization of KV cache vectors is a promising strategy for reducing memory overhead within context windows, preserving the integrity of the model architecture while reducing vector length. Conventional approaches, such as block-wise normalization, require the storage of auxiliary parameters that can exceed 1 bit per quantized value, thereby diminishing compression effectiveness.

The integration of Large Language Models (LLMs) into applications presents intrinsic challenges, particularly regarding computational efficiency, including memory utilization. In self-attention architectures such as Transformers, the sequential token generation process necessitates the continuous storage of Key-Value (KV) cache vectors for all previously processed tokens. This storage grows proportionally with both the length of the context and the size of the model, representing a significant memory usage bottleneck in LLM deployments. Minimizing its size increases the system-supported context window within the same hardware constraints.

To seek efficient alternatives for handling this issue, the present research implements the PolarQuant algorithm proposed by Han et al. (2025), which provides a theoretical approach for this challenge. The method substitutes the

Cartesian representation of vectors with polar coordinates. The algorithm is structured into three phases: (i) random preconditioning via Hadamard rotation, which preserves the norms and inner products of the original vectors while uniformly redistributing the signal energy; (ii) recursive polar transformation, applied to randomized vectors; and (iii) Lloyd-Max quantization, derived from the analytical distribution of angles following pre- and post-conditioning.

A core characteristic of the method is that, after preconditioning, the data vectors follow well-established, mutually independent Gaussian distributions, with a progressive concentration around p/4 as the level advances. This obviates the necessity for explicit data normalization, thereby reducing the memory overhead typically associated with conventional approaches.

As a practical contribution, a Python implementation of the algorithm was developed to reproduce the compression and reconstruction pipeline described in the Han et al. paper. Experiments with Gaussian vectors of dimension d = 32,768, using 8 bits for level 0 and 4 bits for the remaining levels, demonstrate that the compression algorithm achieved a reduction exceeding 80% on a 131,072-byte document. During reconstruction, a mean squared error of approximately 0.0022 was observed.

Palavras-chave: inteligência artificial; eficiência de ia; otimização; algoritmo.