A Causal Layer Using Presum

Ali Qajar
ali.qajar @ gmail.com

In natural language processing (NLP), maintaining causal relationships between tokens is crucial for tasks such as language modeling and sequence prediction. This paper proposes a novel causal layer for neural networks, where each token is influenced only by previous tokens. By leveraging a presum (prefix sum) technique, the proposed model efficiently accumulates information from preceding tokens, enabling causal dependency without the computational overhead of traditional methods. We present the theoretical foundation of this approach and illustrate its implementation with an example. Natural language processing models often require maintaining sequential dependencies to ensure that each token's prediction is influenced only by preceding tokens. Traditional methods like recurrent neural networks (RNNs) and attention mechanisms can capture these dependencies but at a significant computational cost. We propose a causal layer using presum operations to streamline this process, reducing complexity and enhancing efficiency.

Methodology

1. Presum Technique:

The presum (prefix sum) is an operation that computes the cumulative sum of a sequence up to a given point. For a sequence of tokens $\{x_1, x_2, \ldots, x_3\}$, the presum at position i is defined as:

$$presum_i = \sum_{j=1}^{i} x_j$$

Causal Layer Design

In the proposed causal layer, each token t_i is influenced by the cumulative information from all previous tokens $\{x_1, x_2, \ldots, x_{i-1}\}$. This is achieved by computing the presum up to the current token. The output for token x_i is then derived from this presum.

Forward Pass

For each token x_i, compute the presum:

$$S_i = \sum_{j=1}^{i-1} x_j$$

The output for x_i is influenced by S_i ensuring that only previous tokens affect the current token.

Efficiency

This method is efficient as it requires only a linear pass through the sequence to compute the presum, avoiding the quadratic complexity often associated with attention mechanisms.

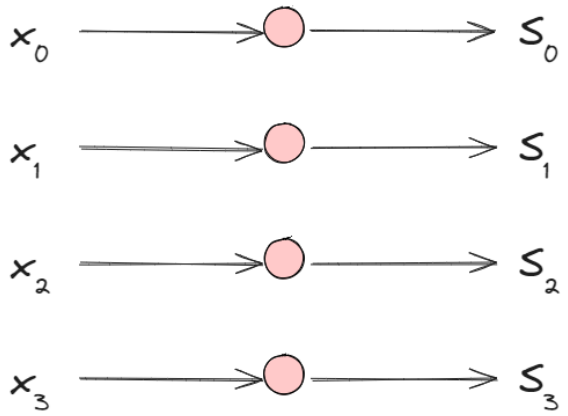

Figure 1: Input is N.d dimensional array (d is model dimension and N is the token array size). Output is the presum on the token dimension, N