Assignment 1: The softmax function

Luca Lombardo

March 9, 2025

Contents

1 Introduction

The softmax function is a mathematical function that takes a vector of real numbers as input and transforms it into a probability distribution. The mathematical definition of the softmax function is given by:

$$\sigma : \mathbb{R}^K \to \left\{ z \in \mathbb{R}^K \mid z_i \geq 0, \sum_{i=1}^{K} z_i = 1 \right\}$$

$$\sigma(\mathbf{z}_j) = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}} \tag{1}$$

2 Implementations

In the following sections, given a scalar implementation (to witch we will refer as `softmax_plain`), we will show how to auto-vectorize it and then how to manually vectorize the code using AVX intrinsics and FMA. Then we will compare the results of the three implementations.

2.1 Auto-Vectorized implementation

In implementing[1] the autovectorized version of the softmax function, I made several key modifications compared to the plain implementation. First, I added `#pragma omp simd` directives to explicitly instruct the compiler to vectorize the three main computational loops, allowing parallel processing of multiple array elements with SIMD instructions. For the first two loops, I included appropriate reduction clauses (i.e., `reduction(max : max_val)` and `reduction(+ : sum)`) to ensure correct calculation of the maximum value and sum while maintaining vectorization. I replaced `std::exp()` with the single-precision `expf()` function, which is specifically optimized

[1] This version is implemented in the file `softmax_auto.cpp`.

for floating-point operations, offers better performance with SIMD instructions, and avoids unnecessary double-precision calculations that would be performed by `std::exp()` before converting back to float. Rather than using repeated divisions in the normalization step, I precomputed the inverse of the sum (`inv_sum = 1.0f / sum`) and used multiplication operations, which are generally more efficient in vectorized code. Instead of using `std::max()`, I implemented an explicit comparison with an `if`-statement that might be more amenable to autovectorization for the compiler.