Making softmax safe

$$\mathbf{S} = \overset{(N,d)}{\mathbf{Q}}\underset{(d,N)}{\mathbf{K}^\top} \in \mathbb{R}^{N\times N}, \quad \mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N\times N}, \quad \mathbf{O} = \overset{(N,N)}{\mathbf{P}}\overset{(N,d)}{\mathbf{V}} \in \mathbb{R}^{N\times d},$$

$$\underset{(5,5)}{\begin{bmatrix} q_1^T k_1 & q_1^T k_2 & q_1^T k_3 & q_1^T k_4 & q_1^T k_5 \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ q_5^T k_1 & q_5^T k_2 & q_5^T k_3 & q_5^T k_4 & q_5^T k_5 \end{bmatrix}} \xrightarrow{\text{SOFTMAX}} \underset{(5,5)}{\begin{bmatrix} 0.1 & 0.05 & 0.5 & 0.15 & 0.2 \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ 0.3 & 0.1 & 0.35 & 0.2 & 0.05 \end{bmatrix}} \begin{array}{l} \Rightarrow \Sigma = 1 \\ \\ \\ \\ \Rightarrow \Sigma = 1 \end{array}$$

Given a vetor $x \in \mathbb{R}^N$, the softmax is defined as:

$$\mathrm{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

But there's a problem! If the values of the vector are large, the exponential will explode!

Numerically unstable = cannot be represented with a float 32 or float 16

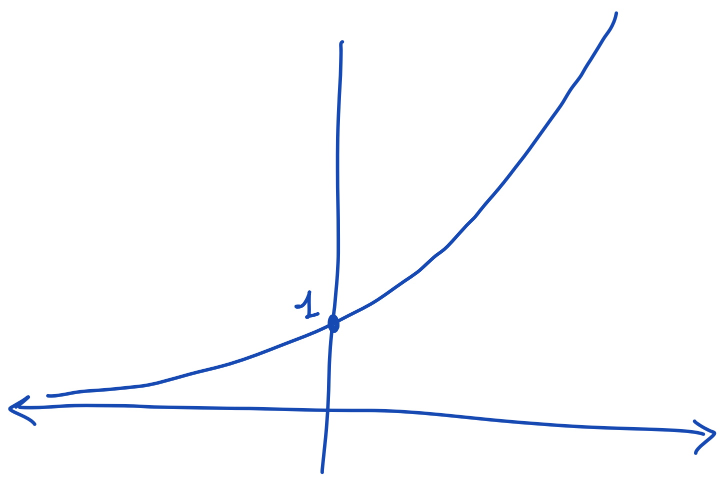
Luckily, we have a solution:

$$\frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} = \frac{c \cdot e^{x_i}}{c \cdot \sum_{j=1}^{N} e^{x_j}} = \frac{c e^{x_i}}{\sum_{j=1}^{N} c e^{x_j}} = \frac{e^{\log(c)} e^{x_i}}{\sum_{j=1}^{N} e^{\log(c)} e^{x_j}} =$$

$$= \frac{e^{x_i + \log(c)}}{\sum_{j=1}^{N} e^{x_j + \log(c)}} = \frac{e^{x_i - k}}{\sum_{j=1}^{N} e^{x_j - k}} \quad \text{where } k = -\log(c)$$

So we can "sneak in" a constant in the exponential to decrease its argument and make it numerically stable.

We will choose $k = \max_i (x_i)$



Let's review the algorithm:

$$\text{softmax}(x_i) = \frac{e^{x_i - x_{max}}}{\sum_{j=1}^{N} e^{x_i - x_{max}}}$$

given a $N \times N$ matrix, for each row.

1) Find the max value among all elements

Time complexity: $O(N)$

Memory reads : $O(N)$

2) Calculate the normalization factor

Time complexity: $O(N)$

Memory reads: $O(N)$

3) Apply the softmax to each element of the vector

Time Complexity: $O(N)$

Memory reads: $O(N)$

Pseudocode:

$m_0 = -\infty$

for $i = 1$ to N

$\quad m_i = \max(m_{i-1}, x_i)$

$\ell_0 = 0$

for $j = 1$ to N

$\quad \ell_j = \ell_{j-1} + e^{x_j - m_N}$

for $k = 1$ to N

$\quad x_k \leftarrow \frac{e^{x_k - m_N}}{\ell_N}$

$$\text{softmax}(x_i) = \frac{e^{x_i - x_{max}}}{\sum_{j=1}^{N} e^{x_i - x_{max}}}$$

Let's see a practical example

$X = [3, \; 2, 5, \; 1]$ $\qquad \text{softmax}(x_i) = \frac{e^{x_i - x_{max}}}{\sum_{j=1}^{N} e^{x_j - x_{max}}}$

1) $X_{max} = 5$

2) $e^{3-5} + e^{2-5} + e^{5-5} + e^{1-5} = e^{-2} + e^{-3} + e^{0} + e^{-4} = \ell$

3) $X_1 = \frac{e^{3-5}}{\ell} \quad X_2 = \frac{e^{2-5}}{\ell} \quad X_3 = \frac{e^{5-5}}{\ell} \quad X_4 = \frac{e^{1-5}}{\ell}$

To apply the softmax to a $N \times N$ matrix, we need to load each of its elements 3 times, and it must be done sequentially...

Is there a better way?