Online Softmax

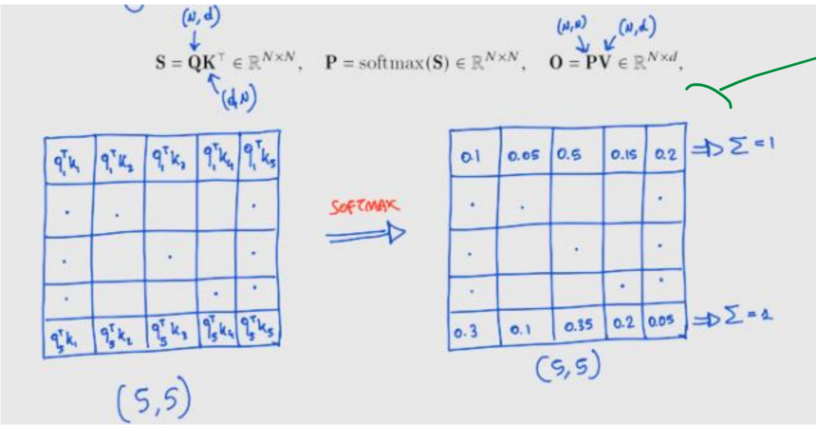
Thursday, October 23, 2025 4:12 PM

Process of turning: Unsafe Naive Softmax --> Safe Naive Softmax --> Online Softmax

Making Softmax "safe":

1

Considerations:



Full Softmax output (S matrix)

If x_i is too large, e**x_i explodes! (eg: e**100, e**200)

If e**x_i explodes, the softmax output cannot be stored.
Reason: outside representable range of standard formats (fp16, fp32)

Normalization Factor: sum of e**x_j, where j is each row vector element

Making Softmax "safe" (numerically stable)

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

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

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 $k = \max_i (x_i)$

Largest x_i among all x_i elements making up THAT row vector on the S matrix
(S : the full softmax output matrix)

- So effectively, we can have two cases:
  - x_i == k:
    - Causes x_i - k = 0, causing e**0 = 1

~~Causes x_i - k = 0, causing e**0 = 1~~

~~'1' : easily representable in standard formats (fp16, fp32)~~

~~x_i < k:~~

~~Causes x_i - k < 0, causing e**(negative value)~~

~~'e**(negative value)' : easily representable in standard formats (fp16, fp32)~~

~~Finally, the Softmax output can be represented safely! (does not become infinity or zero)~~

Safe Softmax:

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

- So effectively, we can have two cases:
  - x_i == k:
    - Causes x_i - k = 0, causing e**0 = 1
    - '1' : easily representable in standard formats (fp16, fp32)
  - x_i < k:
    - Causes x_i - k < 0, causing e**(negative value)
    - 'e**(negative value)' : easily representable in standard formats (fp16, fp32)
- Finally, the Softmax output can be represented safely! (does not explode to infinity)

2

But, this is still "naïve"
Why? Look on ahead:

Let's review the algorithm:

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

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

STEP 1) Find the max value among all elements
Time complexity: $O(N)$
Memory reads : $O(N)$

STEP 2) Calculate the normalization factor
Time complexity: $O(N)$
Memory reads: $O(N)$

STEP 3) Apply the softmax to each element of the vector
Time Complexity: $O(N)$
Memory reads: $O(N)$

Pseudocode:

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

$m_0 = -\infty$
for $i = 1$ to N
  $m_i = \max(m_{i-1}, x_i)$

STEP 1: Finding x_max by looping over all x_i in the corresponding row vector, and accumulating max

Takes O(n) time

$\ell_0 = 0$
for $J = 1$ to N
  $\ell_J = \ell_{J-1} + e^{x_J - m_N}$

STEP 2: Computing Normalization Factor (l_J)
(can only be done AFTER Step 1, as we need it for STEP 2)

Again, takes O(n) time

for $k = 1$ to N
  $x_k \leftarrow \frac{e^{x_k - m_N}}{\ell_N}$

STEP 3: Compute Softmax over each row vector element x_i
(can only be done AFTER STEPS 1 & 2, as we need calculations from both for STEP 3)

One more time!
Takes O(n) time

TOO SLOW!!!
3 sequential O(n) operations here

3

So, what do we do to make this FASTER
(cutting time complexity without exploding exponential, i.e. keeping it "safe")?

New pseudocode (Online Softmax)

$m_0 = -\infty$
$\ell_0 = 0$
for $i = 1$ to N
  $m_i = \max(m_{i-1}, x_i)$
  $\ell_i = \ell_{i-1} \cdot e^{m_{i-1} - m_i} + e^{x_i - m_i}$

- Fusing STEP 1 and STEP 2 (by using a "Correction Factor")
- So, as we loop over each row vector element, we calculate max x_i (SO FAR) AND the Normalization Factor (SO FAR), in the same loop.
- This is opposed to Safe Naïve Softmax, where we went looping over row vector to first get max x_i, then again looping over row vector to get Normalization Factor

Correction Factor
{m_i-1 : max x_i until the 'i-1'th row vector element (looking at all x_i from the start, the max x_i so far),
m_i : max x_i at the 'i'th row vector element}

for $k = 1$ to N
  $x_k \leftarrow \frac{e^{x_k - m_N}}{\ell_N}$

STEP 3 (remains as it was)

2 sequential O(n) operations vs the 3 sequential O(n) earlier

$$X = [3, 2, 5, 1]$$

We can take this row vector X and run the Fused Steps 1 & 2 from the above to see if the math works out, turns out it does!!
Finally, we can do a 'Proof by Induction' for this, turns out that works as well.