Flash Attention v1 原理

Attention 公式:

$$O = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$

其中, Q, K, V 都是 $N \times d$ 大小的矩阵, $d_k = d$

算法动机

原始版本的 Attention 计算中有多次矩阵乘法, 需要多次从较慢的 HBM 中换入换出数据进行运算, 未能充分利用更快的 SRAM. 但是, 由于 SRAM 大小要小于 HBM 的大小, 若直接将所有数据都放入 SRAM, 就很容易出现爆缓存. 但是, 若只加载一部分数据, 原有的算法不能正确的更新结果得到正确结果.

Attention By Online Softmax

原始的 attention 计算是 3-pass 的, 而利用 online softmax 可以进一步减少迭代次数.

$$\begin{aligned}
&\#\ \text{2-pass} \\
&\quad for\ i \to 1 \ldots N \\
&\qquad x_i \leftarrow Q[k, :]K^T[:, i] \\
&\qquad m_i \leftarrow \max(m_{i-1}, x_i) \\
&\qquad l'_i \leftarrow l'_{i-1} e^{m_{i-1} - m_i} + e^{x_i - m_i} \\
&\quad for\ i \to 1 \ldots N \\
&\qquad a_i \leftarrow \frac{e^{x_i - m_N}}{l'_N} \\
&\qquad o_i \leftarrow o_{i-1} + a_i V[i, :]
\end{aligned}$$

在上面的式子中, 下标只代表迭代次数, 而不代表元素具体的位置.

类似之前 l 这样原本需要依据更新完成后的 m 才能更新, 但是后面经过代换也可以与 m 同步更新, 我们也可以对 o 做公式代换, 尝试同步更新 m, l, o.

由

$$\begin{aligned}
o_i &= o_{i-1} + a_i V[i, :] \\
&= o_{i-1} + \frac{e^{x_i - m_N}}{l'_N} V[i, :] \\
&= \sum_{j=1}^{i} \frac{e^{x_j - m_N}}{l'_N} V[j, :]
\end{aligned}$$

有

$$o_N = \sum_{j=1}^{N} \frac{e^{x_j - m_N}}{l'_N} V[j, :].$$

对 o_N 做代换

$$
\begin{aligned}
o'_N &= \sum_{j=1}^{N-1} \frac{e^{x_j - m_N}}{l'_N} V[j,:] + \frac{e^{x_N - m_N}}{l'_N} V[N,:] \\
&= \sum_{j=1}^{N-1} \frac{e^{x_j - m_{N-1}}}{l'_{N-1}} \frac{(e^{m_{N-1} - m_N}) l'_{N-1}}{l'_N} V[j,:] + \frac{e^{x_N - m_N}}{l'_N} V[N,:] \\
&= o'_{N-1} \frac{(e^{m_{N-1} - m_N}) l'_{N-1}}{l'_N} + \frac{e^{x_N - m_N}}{l'_N} V[N,:] \\
&= \left(o'_{N-2} \frac{(e^{m_{N-2} - m_{N-1}}) l'_{N-2}}{l'_{N-1}} + \frac{e^{x_{N-1} - m_{N-1}}}{l'_{N-1}} V[N-1,:] \right) \frac{(e^{m_{N-1} - m_N}) l'_{N-1}}{l'_N} + \frac{e^{x_N - m_N}}{l'_N} V[N,:] \\
&= \ldots
\end{aligned}
$$

也就是说, 这样就找到了利用还没更新完的 m, l 来更新 o 的公式

$$
\begin{aligned}
o'_i &= o'_{i-1} \frac{(e^{m_{i-1} - m_i}) l'_{i-1}}{l'_i} + \frac{e^{x_i - m_i}}{l'_i} V[i,:] \\
&= \frac{o'_{i-1} l'_{i-1} (e^{m_{i-1} - m_i}) + e^{x_i - m_i} V[i,:]}{l'_i}.
\end{aligned}
$$

写成向量形式, 就得到了 FlashAttention 原论文中的算法给出的更新公式

$$
\mathbf{O}_i \leftarrow \operatorname{diag}\left(l_i^{new}\right)^{-1} \left(\operatorname{diag}(l_i) e^{m_i - m_i^{new}} \mathbf{O}_i + e^{\tilde{m}_{ij} - m_i^{new}} \tilde{\mathbf{P}}_{ij}^{dropped} \mathbf{V}_j \right).
$$

此时就得到了 1-pass 的迭代公式

$$
\begin{aligned}
&\#\ \text{1-pass} \\
&\quad for\ i \rightarrow 1 \ldots N \\
&\quad\quad x_i \leftarrow Q[k,:] K^T[:,i] \\
&\quad\quad m_i \leftarrow \max(m_{i-1}, x_i) \\
&\quad\quad l'_i \leftarrow l'_{i-1} e^{m_{i-1} - m_i} + e^{x_i - m_i} \\
&\quad\quad o'_i \leftarrow o'_{i-1} \frac{(e^{m_{i-1} - m_i}) l'_{i-1}}{l'_i} + \frac{e^{x_i - m_i}}{l'_i} V[i,:] \\
&\quad O[k,:] \leftarrow o'_N
\end{aligned}
$$

Tiling

flash attention 一个很重要的思想是将 Q, K, V 分块计算, 在 seq_len 维度上将 N 切分, 一次处理一块, 这样, 在矩阵乘法的时候不能一次性得到 K, V 在这一列的所有数据, 最后 O 上的结果经过多次迭代才得到最后结果, 这就利用了上面的同步更新 o'_i 的算法.