Softmax 优化原理

原始版本

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

```
# 2-pass
l = 0
for i in range(N):
    x[i] = exp(x[i])
    l += x[i]
for i in range(N):
    x[i] = x[i] / l
```

Safe Softmax

在原始版本中, 在极端情况下存在一些问题. 若x_i本身已经很大, 指数运算后容易上溢出. 若每个都是很小的负数, 那么就容易导致分母为零. 故可做如下改进:

记$x_{(N)} = \max x_i$, 则

$$\begin{aligned} Softmax(x_i) &= \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \\ &= \frac{e^{x_i} e^{-x_{(N)}}}{(\sum_{j=1}^{N} e^{x_j}) e^{-x_{(N)}}} \\ &= \frac{e^{x_i - x_{(N)}}}{\sum_{j=1}^{N} e^{x_j - x_{(N)}}} \end{aligned}$$

也就是说, 在计算 e^{x_i} 前先把 x_i 减去最大值最后的结果不变, 同时, 这样也解决了上面的问题.

```
# 3-pass
m = -inf
l = 0
for i in range(N):
    m = max(m, x[i])
for i in range(N):
    x[i] = exp(x[i] - m)
    l += x[i]
for i in range(N):
    x[i] = x[i] / l
```

Online Softmax

Safe Softmax虽然解决了原始版本的溢出问题, 但同时也导致循环次数变多了, 从 2-pass 变为 3-pass, 一行的求和l需要求出了最大值m后才能得到. 可以考虑能否一边迭代更新m的同时也更新l.

记第 i 次迭代得到的 m 为 m_i, 第i次迭代得到的 l 为 l_i, 则 $l_N = \sum_{i=1}^{N} \mathrm{e}^{\mathrm{x_i - m_N}}$
对 l_N 做如下变换:

$$
\begin{aligned}
l'_N &= \sum_{i=1}^{N} e^{x_i - m_N} \\
&= \sum_{i=1}^{N-1} e^{x_i - m_N} + e^{x_N - m_N} \\
&= \left(\sum_{i=1}^{N-1} e^{x_i - m_{N-1}} \right) e^{m_{N-1} - m_N} + e^{x_N - m_N} \\
&= l'_{N-1} e^{m_{N-1} - m_N} + e^{x_N - m_N} \\
&= \left(l'_{N-2} e^{m_{N-2} - m_{N-1}} + e^{x_{N-1} - m_{N-1}} \right) e^{m_{N-1} - m_N} + e^{x_N - m_N} \\
&= \ldots
\end{aligned}
$$

也就是说, 通过上面的代换, 找到了利用还没更新完的 m 来更新 l 的公式:

$$
l'_i = l'_{i-1} e^{m_{i-1} - m_i} + e^{x_i - m_i}
$$

```
# 2-pass
m = -inf
l = 0
for i in range(N):
    new_m = max(m, x[i])
    l = l * exp(m - new_m) + exp(x[i] - new_m)
    m = new_m
for i in range(N):
    x[i] = (x[i] - m) / l
```

Block Online Softmax

虽然上面的 Online Softmax 可以很好的解决问题, 但是每次迭代只能处理一个数据, 有些过于浪费 CUDA 资源, 于是可以想到将 x 分块, 一次处理一块, 在块内做 online softmax, 然后依据每个 block 的结果来更新全局的 l 和 m.

记将 x 分成 T 个块, 每个块的大小为 $B = N/T$, 第 t 个块为 $x^{(t)}$, 第 t 个块得到的局部 l 和 m 分别记为 $l^{(t)}$ 和 $m^{(t)}$. 则全局 $m = \max m^{(t)}$, 但是全局 l 并不能直接相加更新, 注意到

$$
\begin{aligned}
l &= \sum_{i=1}^{N} e^{x_i - m} \\
&= \sum_{t=1}^{T} \sum_{i=(t-1)*B+1}^{t*B} e^{x_i - m} \\
&= \sum_{t=1}^{T} \sum_{i=(t-1)*B+1}^{t*B} e^{x_i - m^{(t)}} e^{m^{(t)} - m} \\
&= \sum_{t=1}^{T} l^{(t)} e^{m^{(t)} - m}
\end{aligned}
$$

于是, 全局的 l 可通过上面式子的结论 $l = \sum_{t=1}^{T} l^{(t)} e^{m^{(t)} - m}$ 来更新

```python
x_blocks = x.chunk(T)
m = [-inf for _ in range(T)]
l = [0 for _ in range(T)]
l_global = 0
for t in range(T):
    x_t = x_blocks[t]
    online_softmax(x_t, m[t], l[t])
m_global = max(m)
l_global = 0
for t in range(T):
    l_global = l[t] * exp(m[t] - m_global)
for i in range(N):
    x[i] = exp(x[i] - m_global) / l_global
```