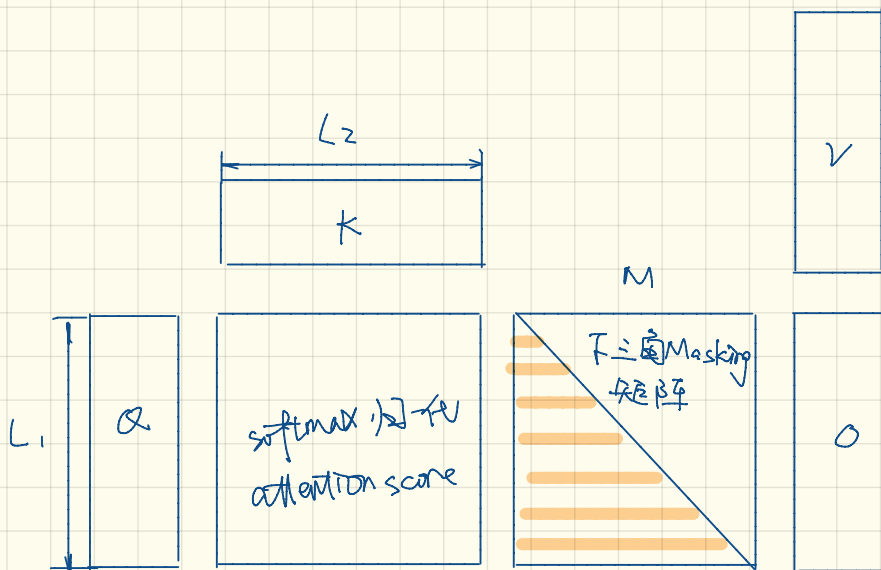


训练时 attention 以全并行方式进行计算，也就是矩阵化公式

$$Q \ K \ V = x W_Q, x W_K, x W_V$$

$$O = (\text{softmax}(Q K^T) \odot M) V$$



中间结果的时间和空间复杂度是
序列长度的二次方

全并行但 I/O 复杂度高

预测可以 recurrent-form 计算

$$q_t, k_t, v_t = x_t W_q, x_t W_k, x_t W_v$$

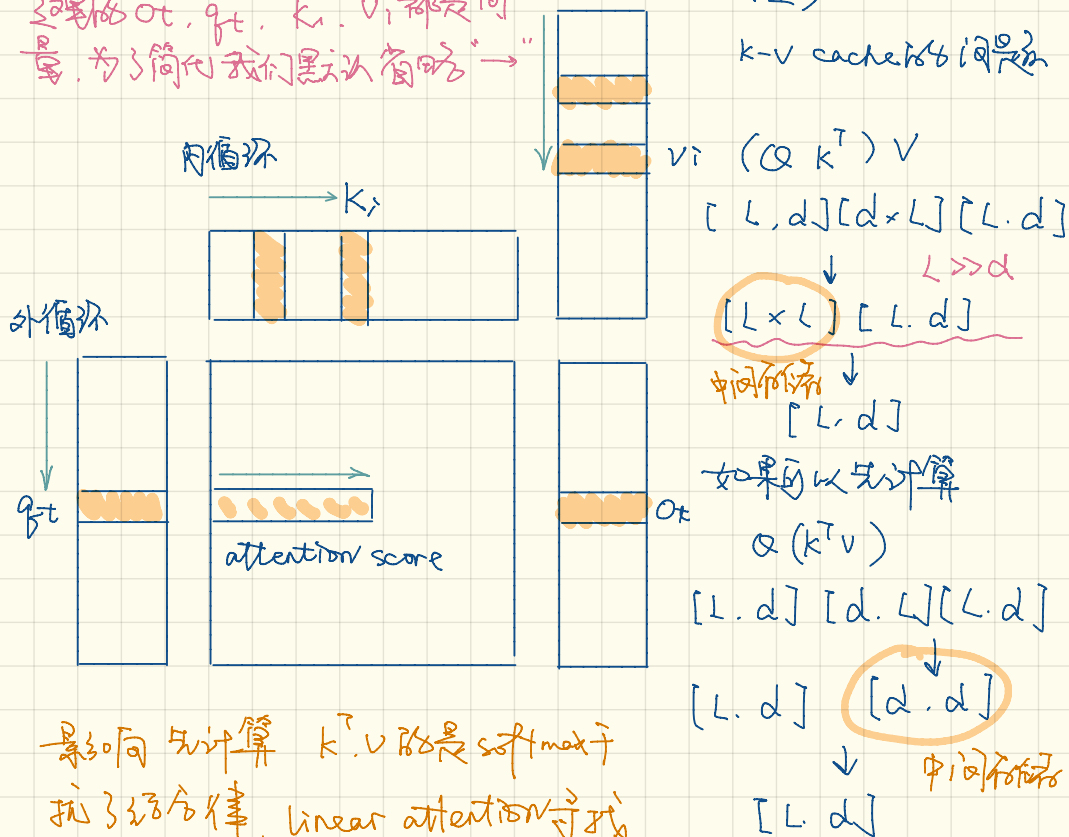
将这里的 softmax 计算看作是相似性计算

$$O_t = \frac{\sum_{i=1}^t \exp(q_t \cdot \underline{k_i^T}) \underline{v_i}}{\sum_{i=1}^t \exp(q_i \cdot \underline{k_i^T})}$$

随着外循环长度的增加如 $\{k_i\} \{v_i\}$ (都是向量) 的长度在增加。

k-v cache 的问题

这里的 q_t, k_t, v_i 都是向量, 为了简化我们默认省略 "→"



影响先计算 $k^T \cdot v$ 的是 softmax 违反了结合律, linear attention 寻找对 softmax 的改造, 使得可以

全并行和串行 RNN 式计算 (这里串行可以以较少的并行性 parallel scan 进行并行性增强)

$$Q_t = \frac{\sum_{i=1}^T \exp(q_t, k_i^T) v_i}{\sum_{i=1}^T \exp(q_t, k_i^T)}$$

内循环

用 kernel 替代 $\exp(q, k)$

$\exp(\vec{q}, \vec{k}) = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle$. 让 Q_t 的计算被简化.

$$Q_t = \frac{\sum_{i=1}^T \phi(q_t) \phi(k_i)^T v_i}{\sum_{i=1}^T \phi(q_t) \phi(k_i)^T}$$

$$= \frac{\phi(q_t) \sum_{i=1}^T \phi(k_i) v_i}{\sum_{i=1}^T \phi(k_i)} \triangleq S_t$$

$$\triangleq Z_t$$

$$\begin{cases} Q_t = \frac{\phi(q_t) S_t}{\phi(q_t) Z_t} \\ S_t = S_{t-1} + \phi(k_t)^T v_t \\ Z_t = Z_{t-1} + \phi(k_t) \end{cases}$$

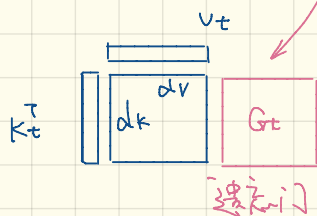
这是在 attention score 的计算中是一个归一化项, 一些研究验证了可以省略, 于是 Q_t 可以被简化为

这是 linear attention 的基本框架

$$\begin{cases} Q_t = \phi(q_t) S_t \\ S_t = S_{t-1} + \phi(k_t)^T v_t \end{cases}$$

架. SSM/RWKV 这样的新型 RNN 从另一个理论视角得到这种形式的形式

$$S_t = S_{t-1} + K_t^T V_t, \quad O_t = g(S_t)$$



Linear attention在学习效果建模上有一个缺点，类似于LSTM中遗忘门的衰减机制对学习效果帮助很大，这里没有decay
 ← 这是一个外积操作，Linear attention是一个隐状态是2D的并行RNN

$$K_t^T V_t = (x_t W_k)^T (W_v x_t)$$

同学习参数

$$O = ((K^T) \odot M) V$$

但由于M的引入破坏了矩阵乘法的结合性，O的计算还是需要从左到右，因此依然是 O^2 空间复杂度

Gated Linear Unit

RetNet引入静态衰减

Mamba / Mamba-2 data-dependent的decay 并行难度高

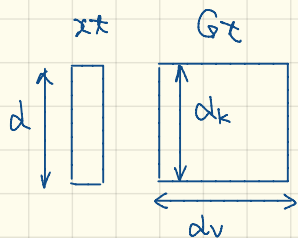
GLA的基本形式: $S_t = G_t \odot S_{t-1} + K_t^T V_t$ 这个公式可以统一许多模型的计算过程

为隐状态引入遗忘门，随时间衰减

$$G_t = [d_t \times d_v] \in [0, 1]$$

在算法设计上要着重考虑如何得到 G_t

$x_t \rightarrow G_t$ 求解一个参数的映射



如果通过某种映射矩阵得到 G_t , 需要 $d * d_k * d_v$ 的参数量. 这是低效的

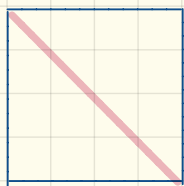
在 Mamba 中, 使用向量 α 与矩阵 A 之间的运算得到 G_t . 缺点是没法使用 tensor core

$$G_t = \alpha_t^T$$

$$S_t = (\alpha_t^T) \odot S_{t-1} + K_t^T V_t$$

$$= \text{Diag}(\alpha_t) S_{t-1} + K_t^T V_t$$

这个公式的通用性很好
并且对硬件友好的并行实现



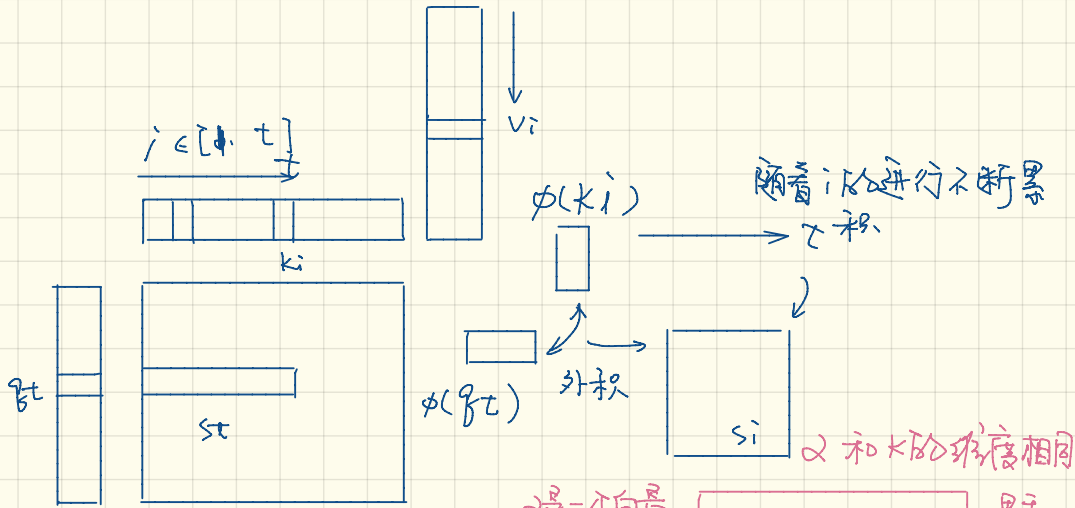
α_t



S_{t-1}

$K_t^T V_t$

通过 x_t 上的低秩线性层 + sigmoid 得到 α_t



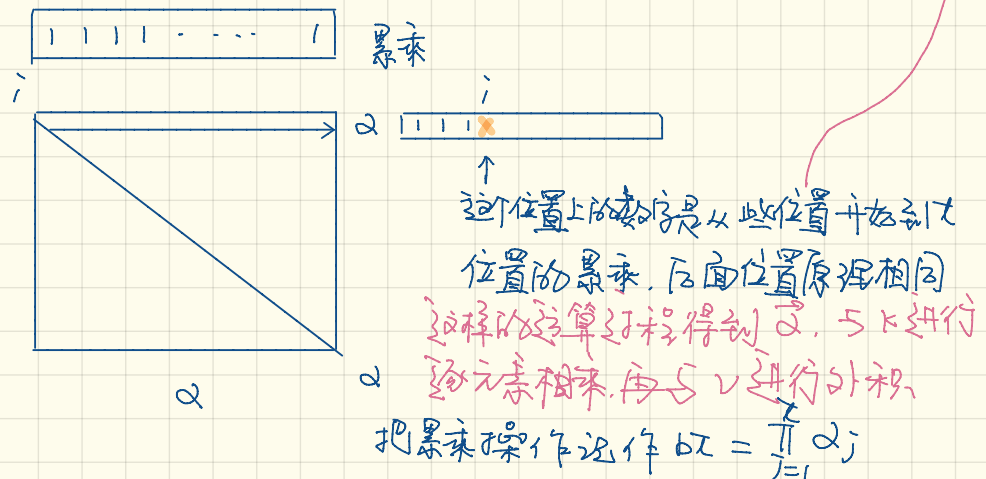
这是一个向量， α 和 k 的维度相同

通过某种低秩方法得到

$$S_T = \sum_{i=1}^t \left(\phi(k_i)^T v_i \right)$$

$$= \sum_{i=1}^t \left(\left(\begin{matrix} \phi(k_i)^T \\ \vdots \end{matrix} \right) \alpha^T \right) \odot k_i^T v_i$$

这个公式是GLA中的具体形式



$$\begin{aligned}
 a_t &= q_t S_t = q_t \sum_{i=1}^t \left(\begin{pmatrix} \frac{b_t}{b_i} & 1 \end{pmatrix}^T \odot k_i \right) u_i \\
 &= \sum_{i=1}^t (q_t \odot b_t) \left(\frac{k_i}{b_i} \right)^T u_i
 \end{aligned}$$

$$0 = \left(\left(Q \odot B \left(\frac{K}{B} \right)^T \right) \odot M \right) v$$