arXiv:2602.07242v1 [cs.CC] 6 Feb 2026

张量提示矩阵向量乘积猜想

Zhao Song*

摘要

Brand、Nanongkai 和 Saranurak [BNS19] 提出了一个被称为提示矩阵向量乘积猜想的猜想。尽管它最初是针对矩阵情形提出的，我们在此将其推广到张量设定。

1 引言

在 [BNS19], 他们 p 提出了提示矩阵向量乘积猜想。在这里，我们将其推广到张量设定。

定义 1（提示矩阵向量乘积猜想的张量版本，类型I）。考虑一个布尔半环和参数 $0 < \tau \leq 1$。令 k 表示一个正整数。我们通过以下三个阶段定义张量提示矩阵向量乘法问题：

1. 输入 $k+1$ $n \times n$ 矩阵 M和 $V_1, V_2, \cdots, V_k$。
2. 输入 k $n \times n$ 矩阵 $P_1, P_2, \cdots, P_k$，其中非零项至多有 n^τ 个。
3. 输入一个索引 $i \in [n]$。
   - 我们需要回答 $M (\oslash_{j=1}^k P_j)^\top (\oslash_{j=1}^k V_j)_{*,i}$，这里 $V_{*,i} \in \mathbb{R}^n$ 是矩阵$V = \oslash_{j=1}^k V_j$的第 i 列。

给定矩阵 $K_j \in \mathbb{R}^{n_\ell \times d}$对所有 $\ell \in [k]$，我们定义矩阵 $K := K_1 \oslash K_2 \oslash \cdots \oslash K_k \in \mathbb{R}^{n_1 n_2 \cdots n_k \times d}$如下 $K_{i,j} := \prod_{\ell=1}^k (K_\ell)_{i_\ell, j}$, $\forall i_\ell \in [n_\ell]$，其中 $i := \sum_{\ell=1}^{k-1} (i_\ell - 1) \prod_{\ell'=\ell+1}^k n_{\ell'} + i_k$。我们记 $\oslash_{j=1}^k V_j := V_1 \oslash V_2 \oslash \cdots \oslash V_k$。对于 $k = 1$，定义 1 退化为 [BNS19]（该问题在其原论文中被称为矩阵向量乘积提示矩阵向量乘积）。乍一看，该问题似乎要求对 k呈指数级依赖。然而，通过采用张量技巧，我们证明多项式级依赖就足够了。运行时间分析的核心思想是张量技巧（引理 3），即 $(\oslash_{j=1}^k P_j)^\top (\oslash_{j=1}^k V_j) = \odot_{j=1}^k (P_j^\top V_j)$。我们用 $\odot$ 表示哈达玛积，即逐元素乘积 $(A \odot B)_{i,j} = A_{i,j} B_{i,j}$。我们记 $\odot_{j=1}^k V_j := V_1 \odot V_2 \odot \cdots \odot V_k$。

假设多项式预处理时间成立，如何有效利用 M 和 $\{V_\ell\}_{\ell \in [k]}$ 矩阵所提供的结构信息仍不明确。此外，我们无法预先确定 M 中的哪些特定项会与最终乘积 V中的项发生交互。受限于多项式时间，我们无法穷举评估 P所有可能配置构成的指数级庞大空间。因此，我们考虑后续阶段的两种不同算法策略：

方法 1. 在阶段 2 中，我们只需计算 $M(\oslash_{j=1}^k P_j)^\top (\oslash_{j=1}^k V_j)$. 根据引理 3，它可以写成 $M(\odot_{j=1}^k (P_j^\top V_j))$. 对于每个 $j \in [k]$，我们计算 $P_j^\top V_j$这需要 $n^{1+\tau}$的时间，注意

*magic.linuxkde@gmail.com。作者谨此感谢 Jan van den Brand 在构思这一猜想时提供的有益讨论。

其中 $P_j^\top V_j$ 有 n^τ 个非零行。对于所有 k 项，我们需要花费 $k \times n^{1+\tau}$ 的时间。计算$\odot_{j=1}^k (P_j^\top V_j)$需要 $k \cdot n^{1+\tau}$的时间。矩阵 $\odot_{j=1}^k (P_j^\top V_j)$最多也只有 n^τ个非零行。最后，我们可以计算 $M(\odot_{j=1}^k (P_j^\top V_j))$，这需要 $n^{\omega(1,1,\tau)}$。总共需要 $O(n^{\omega(1,1,\tau)} + kn^{1+\tau})$的时间。在阶段3中，我们只需输出所需的列向量。

方法2。我们在阶段 2 不进行任何计算，只记住 P_j的非零集合。在阶段3中，我们计算 $M(\oslash_{j=1}^k P_j)^\top (\oslash_{j=1}^k V_j)_{*,i_\circ}$ 对于每个 j，我们用 v_j 表示矩阵 V_j 的第 i 列。接下来需要关注的量是 $M(\oslash_{j=1}^k P_j)^\top (\oslash_{j=1}^k v_j)$。根据引理 3，目标量可以写成 $M(\odot_{j=1}^k P_j^\top v_j)$。对于每个 $j \in [k]$，我们计算 $P_j^\top v_j$，这需要n^τ时间。由于共有 k 项，因此需要 $k \cdot n^\tau$。接下来，我们将计算 $\odot_{j=1}^k (P_j^\top v_j)$，这需要$k \cdot n^\tau$ 时间。最后，使用该 n^τ 稀疏向量计算 M 需要 $n^{1+\tau}$ 时间。总的来说，整个过程可以在 $O(n^{1+\tau} + kn^\tau)$ 时间内完成。

我们的猜想如下：

猜想2. 对于任何求解 *Tensor Hinted Mv*（定义 *1*，类型*I*）的算法，若在阶段*1*进行多项式预处理，则以下下界至少适用于一个阶段：

- 阶段 *2* 需要 $\Omega(n^{\omega(1,\ 1,\ \tau)-\delta} + kn^{1+\tau-\delta})$.
- 阶段*3*需要 $\Omega(n^{1+\tau-\delta} + kn^{\tau-\delta})$。

对于每个 $\delta > 0$。

对于 $k = 1$ 的情况，上述猜想退化为 [BNS19] 中的猜想5.7。这里我们给出本文所用张量技巧的证明。

引理 3. 对于每个 $\ell \in [k]$，我们定义 $A_\ell \in \mathbb{R}^{n_\ell \times d_a}$。令 $n := \prod_{\ell=1}^k n_\ell$。令 $A := (\oslash_{\ell=1}^k A_\ell) \in \mathbb{R}^{n \times d_a}$。令 $B := (\oslash_{\ell=1}^k B_\ell) \in \mathbb{R}^{n \times d_b}$。我们将 $C \in \mathbb{R}^{d_a \times d_b}$ 定义为 $C := A^\top B$。我们定义 $C_\ell := A_\ell^\top B_\ell$。于是，我们有 $\odot_{\ell=1}^k C_\ell = C$。

证明。对于每个 $i_\ell \in [n_\ell]$，令 $a_{\ell,i_\ell}^\top$ 表示 A_ℓ 的第 i_ℓ 行。类似地，我们为 B 定义 b 。于是，我们可以将 $C \in \mathbb{R}^{d_a \times d_b}$写作 $C = A^\top B = \sum_{i=1}^n A_{i,*} (B_{i,*})^\top = \sum_{i_1,i_2,\cdots,i_k} (\odot_{\ell=1}^k a_{\ell,i_\ell}) \cdot (\odot_{\ell=1}^k b_{\ell,i_\ell})^\top = \sum_{i_1,i_2,\cdots,i_k} \odot_{\ell=1}^k (a_{\ell,i_\ell} b_{\ell,i_\ell}^\top) = \odot_{\ell=1}^k (\sum_{i_\ell=1}^{n_\ell} a_{\ell,i_\ell} b_{\ell,i_\ell}^\top) = \odot_{\ell=1}^k C_\ell$。因此，我们可以得出 $C = \odot_{\ell=1}^k C_\ell$。 □

除定义1外，我们还考虑另一种张量表述。

定义 4 (提示矩阵向量乘法的张量版本，II 型)。 考虑一个布尔半环和参数 $\tau > 0$。我们通过以下三个阶段定义 *Tensor Hinted Mv* 问题：

1. 输入 k $n \times d$ 矩阵 $V_1, V_2, \cdots, V_k$。

2. 输入一个至多包含 n^τ 个非零项的 $d \times d \times \cdots \times d$ （对角）张量 P。

3. 输入一个索引集 $\{\ell_1, \cdots, \ell_s\} \subseteq [k]$ 和一个索引多重集 $\{i_1, \cdots, i_s\}$，其中每个元素均来自 $[n]$。

- 我们需要回答 $[P(V_1, V_2, \cdots, V_k)]_{\cdots,\ i_1,\ \cdots,\ i_s,\ \cdots}$，其中 i_1 位于第 ℓ_1 个方向。这同样适用于其他 $s-1$ 对。

这里 $P(V_1, \cdots, V_k)$ 是一个 n^k 大小的张量，其中第 $(i_1, \cdots, i_k)$ 个元素为 $\sum_{j_1,\cdots,j_k} P_{j_1,\cdots,j_k} (V_1)_{i_1,j_1} \cdots (V_k)_{i_k,j_k}$。 根据我们的定义，不难验证 $[P(V_1, V_2, \cdots, V_k)]_{\cdots,i,\cdots}$ 是 $P(V_1, \cdots, (V_\ell)_{*,i}, \cdots, V_k)$。对于 $k = 2$，定义 4 退化为 [BNS19]。现在，我们来介绍一些直接的方法。

方法 1. 在阶段 2 中，我们将计算 $P(V_1, V_2, \cdots, V_k)$。这实际上与$U_1 \otimes U_2 \otimes \cdots \otimes U_k$[1] 相同，其中 U_i的大小为 $n \times n^\tau$。 U_1 是通过选取 n^τ列生成的矩阵

[1]该张量的第 $(i_1, \cdots, i_k)$ 个元素为 $\sum_{j_1,\cdots,j_k} \prod_{\ell=1}^k U_{i_\ell,j_\ell}$。

在 V_1 中，将 V_1 重缩放 P倍。其他 U_ℓ 只需从 V_ℓ中选取 n^τ 列即可构成。我们先计算 $W_1 = \oslash_{\ell=1}^{k-\lfloor k/2 \rfloor} U_\ell$，然后计算 $W_2 = \oslash_{\ell=k-\lfloor k/2 \rfloor+1}^{k} U_\ell$。这一步需要$O(n^{k-\lfloor k/2 \rfloor})$时间。接下来计算 $W_1 W_2^\top$，这需要 $n^{\omega(k-\lfloor k/2 \rfloor, \tau, \lfloor k/2 \rfloor)}$ 时间。在阶段3中，我们将直接输出各项，因为 $W_1 W_2^\top$只是原始张量的矩阵视图。

方法2。 在阶段 2 中，我们什么也不做。在阶段3中，我们直接计算答案。不失一般性，我们可以假设 $\{\ell_1, \cdots, \ell_s\}$ 正是 $[k]$中最后的 s 个指标。我们关注的量本质上是 $U_1 \otimes \cdots \otimes U_{k-s}$，其中 U_1 是从 V_1 中选取n^τ 列（依据 P），并通过 P 进行重缩放后生成的矩阵，以及这些 $(U_{\ell_t})_{i_t, *}$（对所有 $t \in [s]$而言）。其他 U_ℓ 则只需针对 $\ell \in \{2, \cdots, k-s\}$，从 V_ℓ 中选取列（依据 P）即可构成。我们先计算 $W_1 = \oslash_{\ell=1}^{k-s-\lfloor (k-s)/2 \rfloor} U_\ell$，再计算 $W_2 = \oslash_{\ell=k-s-\lfloor (k-s)/2 \rfloor+1}^{k-s} U_\ell$。这两个步骤耗时 $O(n^{k-s-\lfloor (k-s)/2 \rfloor})$。接下来，我们只需计算 $W_1 W_2^\top$。这需要$n^{\omega(k-s-\lfloor (k-s)/2 \rfloor, \tau, \lfloor (k-s)/2 \rfloor)}$ 时间。我们将输出 $W_1 W_2^\top$，因为它就是目标张量的矩阵视图。

猜想5。 对于任何解决 *Tensor Hinted Mv*（定义 *4*，类型*II*）的算法，在阶段*1*进行多项式预处理的情况下，以下下界至少适用于一个阶段：

- 阶段 *2* 需要$\Omega(n^{\omega(k-\lfloor k/2 \rfloor,\ \lfloor k/2 \rfloor,\ \tau)-\delta})$。
- 阶段 *3* 需要$\Omega(n^{\omega(k-s-\lfloor (k-s)/2 \rfloor,\ \lfloor (k-s)/2 \rfloor,\ \tau)-\delta})$。

对于每个 $\delta > 0$。

对于 $k = 2$ 和 $s = 1$的情况，上述猜想退化为 [BNS19]中的猜想5.7。

参考文献

[BNS19] Jan van den Brand、Danupon Nanongkai 和 Thatchaphol Saranurak。动态矩阵求逆：改进的算法与匹配的条件下界。载于 *2019*年*IEEE*第*60*届计算机科学基础年会（*FOCS*），第456–480页。IEEE，2019年。