arXiv:2304.02207v1 [cs.DS] 5 Apr 2023

大语言模型中 DynamicAttention 维护的算法与困难性

Jan van den Brand* Zhao Song† Tianyi Zhou‡

摘要

大语言模型（LLM）从根本上改变了人类生活。注意力机制是所有大语言模型中的关键组成部分之一，例如 BERT、GPT-1、Transformer、GPT-2、3、3.5 和 4。受此前关于静态版本注意力乘法问题的理论研究启发 [Zandieh、Han、Daliri 和 Karbasi，arXiv 2023；Alman 和 Song，arXiv2023]。在这项工作中，我们正式定义了注意力矩阵乘法问题的动态版本。给定矩阵 Q，K，$V \in \mathbb{R}^{n \times d}$，它们分别表示大语言模型中的查询、键和值。在每次迭代中，我们更新 K 或 V 中的一个条目。在查询阶段，我们接收 $(i, j) \in [n] \times [d]$ 作为输入，并希望得到 $(D^{-1}AV)_{i,j}$，其中 $A := \exp(QK^\top) \in \mathbb{R}^{n \times n}$ 是一个方阵，$D := \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$ 是一个对角矩阵。这里 $\mathbf{1}_n$ 表示一个长度为 n 且所有元素均为 1 的向量。

我们给出两项结果：一种算法和一个条件下界。

- 一方面，受 [Demetrescu 和 Italiano（FOCS 2000）、Sankowski（FOCS 2004）、Cohen、Lee 和 Song（STOC 2019）以及 Brand（SODA 2020], ）提出的惰性更新思想启发，我们给出一种数据结构，其 $O(n^{\omega(1,1,\tau)-\tau})$ 摊还更新时间为 $O(n^{1+\tau})$，最坏情况查询时间为 $O(n^{1+\tau})$。
- 另一方面，我们证明，除非提示矩阵向量乘法猜想[Brand、Nanongkai 和 Saranurak（FOCS 2019] ）不成立，否则不存在一种算法能同时达到 $O(n^{\omega(1,1,\tau)-\tau-\Omega(1)})$ 摊还更新时间和 $O(n^{1+\tau-\Omega(1)})$ 最坏情况查询时间。

总之，除非提示矩阵向量乘法猜想不成立，否则我们的算法结果在条件意义下是最优的。

先前工作 [Alman and Song arXiv 2023]与我们工作的一个显著区别是，他们的技术来自细粒度复杂性领域，而我们的技术则不是。我们的算法技术来自凸优化的近期工作，例如求解线性规划。我们的困难性技术来自动态算法领域。

*vdbrand@gatech.edu. 佐治亚理工学院。
†zsong@adobe.com. Adobe 研究院。
‡t8zhou@ucsd.edu. 加利福尼亚大学圣迭戈分校。

1 引言

大语言模型（LLMs）如 Transformer [VSP+17], BERT [DCLT18], GPT-3 [BMR+20],PaLM [CND+22], 以及 OPT [ZRG+22]在处理自然语言时比小型模型或传统技术提供更好的结果。这些模型具备理解和生成复杂语言的能力，这对语言翻译、情感分析和问答等广泛应用有益。大语言模型可以调整用于多种目的，而无需从头构建。一个主要例子是ChatGPT，这是由OpenAI开发的聊天软件，充分利用了GPT-3的潜力。GPT-4 [Ope23], 最新迭代版本，有潜力超越GPT-3已经令人印象深刻的能力，包括语言翻译、问答和文本生成等任务。因此，GPT-4对自然语言处理的影响可能是显著的，新应用可能出现在虚拟助手、聊天机器人和自动化内容创作等领域。

大语言模型背后的主要技术基础是注意力矩阵 [VSP+17，RNS+18，DCLT18，BMR+20]。本质上，注意力矩阵是一个方阵，其对应的行和列分别表示单词或“词元”，矩阵中的元素则表示它们在给定文本中的相关性。随后，利用该矩阵评估序列中每个词元相对于期望输出的重要程度。在注意力机制中，系统会根据每个输入词元对当前输出的重要性或相关性为其分配一个分数或权重；这一重要性或相关性通过相似度函数比较当前输出状态与输入状态来确定。

更正式地说，注意力矩阵可表示如下：假设有两个矩阵，Q 和 K，分别由查询词元和键词元组成，其中 $Q \in \mathbb{R}^{n\times d}$ 和 $K \in \mathbb{R}^{n\times d}$。注意力矩阵是一个方形 $n\times n$ 矩阵，记作 A，用于关联序列中的输入词元。使用 Softmax 函数进行归一化后，该矩阵中的每个元素都量化了特定输入词元（查询词元 Q）与输出词元（键词元K）之间的注意力权重或分数。值得注意的是，对角线上的元素反映了自注意力分数，表示每个词元与自身相关的重要性。

使用大型 n对长序列进行建模时，加速大语言模型运算的最大障碍是执行注意力矩阵计算所需的时间 [KKL20，WLK+20]。这些计算涉及将注意力矩阵 A 与另一个值令牌矩阵相乘 $V \in \mathbb{R}^{n\times d}$。在 [WLK+20]中，他们证明，自注意力机制可以用低秩矩阵近似。他们提出了一种新的自注意力机制，并将其用于 Linformer 模型。在 [KKL20], 中，他们用基于局部敏感哈希的注意力取代点积注意力，这也降低了时间复杂度。

此外， [AS23]从算法和困难性两个角度研究了静态注意力计算及其近似。然而，在实际应用中，注意力矩阵需要经过训练，并且会不断变化。在这项工作中，我们研究注意力计算问题的动态版本。通过采用动态方法，随着新信息的引入，可以实时更新注意力权重，使模型能够更有效地适应输入的变化。这在输入数据高度动态且频繁变化的情况下尤其有益，例如自然语言处理应用中，词语和短语的含义及上下文会受到周围文本的影响。

沿用此前的工作 [ZHDK23、AS23], 我们将标准的注意力计算问题正式定义如下。为区别于其标准模型与本文研究的动态版本，我们将 [ZHDK23、AS23] 中定义的问题称为注意力乘法的“静态”版本。与此前工作 [ZHDK23、AS23] 相比，我们工作的另一项主要区别在于：它们研究的是近似版本，而我们研究的是精确版本。

定义 1.1（静态注意力乘法）。给定三个矩阵 Q、K、$V \in \mathbb{R}^{n \times d}$，我们定义注意力计算

$$\mathrm{Att}(Q, K, V) = D^{-1}AV$$

其中，方阵 $A \in \mathbb{R}^{n \times n}$ 和对角矩阵 $D \in \mathbb{R}^{n \times n}$ 为

$$A := \exp(QK^\top), \quad D := \mathrm{diag}(A\mathbf{1}_n)$$

这里我们逐元素应用$\exp(\cdot)$ 函数[1]。我们用 $\mathbf{1}_n$ 表示一个长度为n 的向量，其中所有元素均为 *1*。函数$\mathrm{diag}()$ 以长度为n 的向量作为输入，并通过将该向量复制到输出矩阵的对角线上，输出一个 $n \times n$对角矩阵。示意图见图*1*和图*2*。

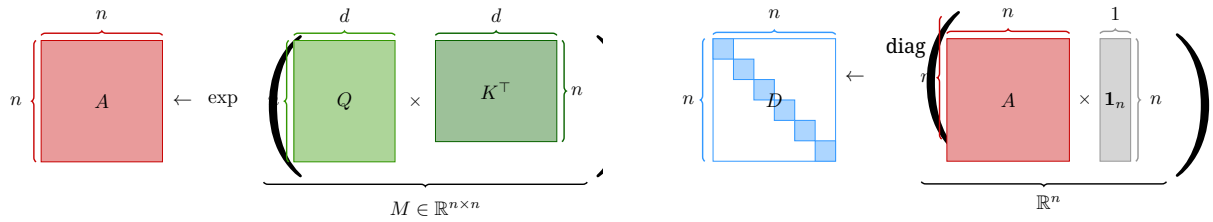

图 1：注意力矩阵的计算 $A = \exp(QK^\top)$以及对角矩阵 $D \in \mathbb{R}^{n \times n}$(定义见定义1.1)。这里的 exp() 是逐元素函数。

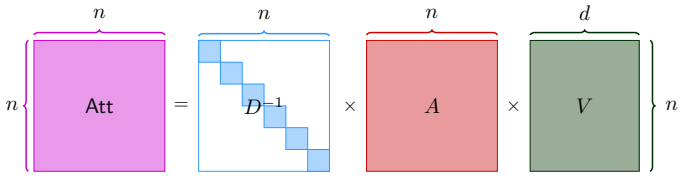

图 2：目标矩阵的计算$\mathrm{Att}(Q,\ K,\ V) = D^{-1}AV$(定义见定义1.1)

在应用场景中的大语言模型训练里，模型参数会在训练过程中缓慢变化 [CLP+21]。因此，值得考虑注意力乘法问题的动态版本。接下来，我们将正式定义注意力乘法问题的“动态”或“在线”版本，称之为ODAMV[2]。为保持讨论一致，本文其余部分将使用“在线”一词。

定义 1.2 ($\mathrm{ODAMV}(n, d)$)。 ***Online Diagonal-based normalized Attention Matrix Vector multiplication problem*** $\mathrm{ODAMV}(n, d)$ 的目标是设计一种数据结构，以支持以下操作：

1. Init：在三个 $n \times d$ 矩阵 Q、K、V上执行 Init初始化。
2. Update：更改 K或 V中的任意一个条目。
3. Query：对于任意给定的 $i \in [n]$、$j \in [d]$，返回$(D^{-1}\exp(QK^\top)V)_{i,\ j}$。

[1]对于一个矩阵 $M \in \mathbb{R}^{n \times n}$，遵循 Transformer 文献中的惯例，我们使用 $\exp(M)_{i,\ j} := \exp(M_{i,\ j})$ 。我们的 $\exp(\cdot)$ 并非矩阵 Chernoff 界文献中的矩阵指数 [GLSS18]。[2]我们的问题名称受到理论计算机科学中一个著名问题的启发，该问题称为 Online Matrix Vector multiplication problem（OMV） [HKNS15、LW17、CKL18]。

- 这里 $D := \mathrm{diag}(\exp(QK^\top)\mathbf{1}_n) \in \mathbb{R}^{n\times n}$ 是一个正对角矩阵。
- 这里 $[n]$ 表示集合 $\{1, 2, \cdots, n\}$。

本文首先提出一种数据结构，通过使用延迟更新技术，高效地解决 ODAMV 问题（定义 1.2）。随后，我们通过一个条件下界补充了这一结果。积极的一面是，我们在动态算法领域使用延迟更新技术来给出一个上界。在理论计算机科学领域，证明下界时假设某个复杂性猜想是非常常见的做法。例如，P $\neq$ NP、（强）指数时间假设、正交向量等。为了证明我们的条件下界，我们使用了一个称为 **H**inted **M**atrix **V**ector 乘法（HMV）猜想的猜想 [BNS19]。消极的一面是，在假设 ODAMV 的 HMV猜想成立的前提下，我们证明了计算求解

1.1 我们的结果

我们首先介绍利用延迟更新策略得到的上界结果 y.

定理 1.3（上界，定理 4.1 的非正式表述）。对于任意常数 $a \in (0, 1]$。令$d = O(n)$。存在一种动态数据结构，使用 $O(n^2)$ 空间，并支持以下操作：

- Init(Q, K, V)。其运行时间为 $O(\mathcal{T}_{\mathrm{mat}}(n, n, n))$。[3]
- UpdateK$(i \in [n], j \in [d], \delta \in \mathbb{R})$。此操作更新 Q 中的一个条目，摊还时间为$O(\mathcal{T}_{\mathrm{mat}}(n, n^a, n)/n^a)$。
- UpdateV$(i \in [n], j \in [d], \delta \in \mathbb{R})$。此操作的摊还时间与 Q 更新相同。
- Query$(i \in [n], j \in [d])$。此操作输出 $(D^{-1}(\exp(QK^\top))V)_{i,j}$ 并且最坏情况下的运行时间为 $O(n^a)$。

我们的第二个结果利用了广为人知的在线矩阵向量乘法（OMV）猜想的一种变体，称为带提示的矩阵向量乘法猜想（见定义5.2和 [BNS19]）。接下来，我们给出动态维护注意力计算问题的一个下界 Att(Q, K, V)。

引理 1.4（下界，引理 5.9 的非正式版本）。假设 HMV 猜想成立。对于任意常数 $0 < \tau \leq 1$，不存在能解决 ODAMV(n, d) 问题的算法（正式版本见定义 *5.8*），其

- 多项式初始化时间，且
- 摊还更新时间 $O(\mathcal{T}_{\mathrm{mat}}(n, n^\tau, d)/n^{\tau+\Omega(1)})$，且
- 最坏查询时间 $O(n^{\tau-\Omega(1)})$。

[3]我们使 e $\mathcal{T}_{\mathrm{mat}}(n, d, m)$ 表示将一个 $n \times d$ 矩阵与另一个 $d \times m$ 矩阵相乘的时间。

1.2 相关工作

静态注意力计算 Zandieh、Han、Daliri 和 Karbasi 的近期工作 [ZHDK23]首次给出了具有可证明保证的近似注意力计算算法。他们的算法利用了局部敏感哈希（LSH）技术 [CKNS20]。他们表明，Softmax 函数分母中配分函数的计算可归约为核密度估计（KDE）问题的一种变体，并可通过基于子采样的快速矩阵乘积来使用高效的 KDE 求解器。他们提出了 KDEformer，可在亚二次时间内近似注意力，并给出了可证明的谱范数界。相比之下，早期结果仅获得逐项误差界。基于经验证据，已确认 KDEformer 在不同预训练模型中，于精度、内存和运行时间方面均优于其他注意力近似方法。

另一项近期工作 [AS23], 关注的是 $d = O(\log n)$下的长序列设置。作者证明，在给定 $\|Q\|_\infty \leq B$、 $\|K\|_\infty \leq B$和 $\|V\|_\infty \leq B$的保证条件下，能否找到一种快速算法来近似计算注意力，取决于 B的取值。他们的下界证明基于另一条研究路线，该路线研究了 KDE 问题的细粒度复杂性，此前已有工作对此进行探讨[BIS17, ACSS20]。他们的证明采用了从近似最近邻搜索问题 ANN 出发的细粒度归约。此外，他们的研究结果解释了如何通过假设矩阵元素有界，或可用少量比特良好近似，来加快大语言模型的计算；这一点此前已在 [ZBIW19], 第 2 节和 [KVPF20], 第 3.2.1 节中讨论。具体而言，他们 [AS23] 证明了一个下界：当 $B \geq \Omega(\sqrt{\log n})$时，不存在能在次二次时间内近似该计算的算法。然而，当$B < o(\sqrt{\log n})$时，他们提出了一种几乎线性时间内近似计算注意力的算法。

Transformer理论 尽管 Transformer 在各个领域取得的成就不容否认，但我们对其学习机制的准确理解仍存在显著不足。尽管这些模型已在涵盖众多结构化任务和推理活动的基准测试中接受评估，但我们对 Transformer 数学层面的理解仍远远滞后。早期研究认为，基于 Transformer 的模型（如 BERT[DCLT18], ）之所以成功，可归因于其组件中蕴含的信息，尤其是注意力头。这些组件被发现包含大量信息，有助于完成与句法和语义相关的各类探测任务，多个研究中的实证证据也支持这一观点 [HM19，CKLM19，TDP19，HL19，VB19，Bel22]。

近期有多项研究深入探讨了Transformer的表征能力，并尝试提供有力证据来证明其表达能力。这些研究从图灵完备性 [BPG20], 函数逼近 [YBR+20]、形式语言表征[BAG20, EGZ20, YPPN21], 、抽象代数运算学习 [ZBB+22]以及统计样本复杂度 [WCM21, EGKZ22] 等方面出发，采用了理论方法和受控实验方法。根据 [YBR+20]的研究，Transformer能够作为序列到序列运算的通用逼近器。同样， [PMB19, BPG20] 的研究表明，注意力模型能够有效地模拟图灵机。除这些近期研究外，此前也有多项研究通过测试神经网络模型在简单数据模型上的学习能力，来评估其能力 [SS92, YPPN21, ZBB+22]。此外， [LLR23] 对训练动态进行了形式化分析，以进一步了解模型从这类数据模型中学到的知识类型。根据一项

近期研究 [ZPGA23], 中等规模的掩码语言模型已展现出能够以令人满意的效果进行句法分析的能力。此外，该研究使用了类似BERT的模型，这些模型通过掩码语言建模损失函数，在利用概率上下文无关文法生成的合成文本上进行预训练。研究人员通过实证验证，这些模型能够识别有助于部分重建句法分析树的句法信息。 [LSZ23] 研究了正则化版本的指数回归问题的计算（不含归一化因子）。

动态维护 近年来，投影维护已成为一个重要的数据结构问题。若干前沿凸优化算法的有效性和效率，很大程度上取决于稳健且精简的投影维护数据结构 [CLS19, LSZ19, Bra20, JLSW20, BLSS20, JSWZ21, SY21, DLY21, Bra21, JKL+20, HJS+22, GS22]。优化领域的动态数据结构问题与我们的动态注意力矩阵维护问题之间存在两个主要差异。第一个显著差异是，在优化任务中，通常需要计算满秩方阵的逆；而在注意力问题中，我们关注的是正对角矩阵的逆，它在大语言模型中起归一化作用。第二个主要差异是，在标准优化任务中，所有矩阵乘法运算都是线性运算。然而，在大语言模型中，需要使用 softmax/指数函数等非线性运算，才能使模型取得良好性能。因此，我们需要对相应矩阵逐元素应用非线性函数。特别是，当 f 为线性函数时，为了计算$f(QK^\top)V$，我们可以预先计算 $K^\top V$。然而，当 f 为指数函数时，我们不能直接计算 $K^\top V$。

接下来，我们将更详细地回顾经典的优化动态矩阵维护问题。设 $B \in \mathbb{R}^{m \times n}$，考虑投影矩阵 $P = B^\top (BB^\top)^{-1} B$。投影维护问题提出了如下数据结构问题：该数据结构可以进行预处理并计算初始投影。在每次迭代中，B 会发生低秩或稀疏变化，数据结构需要更新 B 以反映这些变化。随后，需要对更新后的 P 与在线向量 h的矩阵-向量乘积进行近似计算。例如，在线性规划中，令 $B = \sqrt{W}A$，其中 $A \in \mathbb{R}^{m \times n}$ 是约束矩阵，W 是对角矩阵。在每次迭代中，W 都会受到相对较小的扰动。随后，数据结构需要针对在线向量 $h \in \mathbb{R}^n$，输出一个近似向量 $\sqrt{W}A^\top (AWA^\top)^{-1} A\sqrt{W}h$。

路线图 本文其余部分安排如下。在第 2 节，我们介绍一些预备知识。在第 3 节，我们阐述用于证明上界和下界结果的技术。在第4 节，我们介绍动态数据结构。我们的算法给出了上界结果。在第 5 节，在假设提示式矩阵向量猜想成立的情况下，我们给出条件下界结果。

2 预备知识

对于矩阵 A，我们用 $A^\top$ 表示其转置。对于非零对角矩阵 $D \in \mathbb{R}^{n \times n}$，我们用 $D^{-1} \in \mathbb{R}^{n \times n}$ 表示这样一个矩阵：对所有 $i \in [n]$，其第 (i, i) 个对角元素为 $(D_{i,i})^{-1}$。

对于向量 $x \in \mathbb{R}^n$，我们用 $\mathrm{diag}(x) \in \mathbb{R}^{n \times n}$ 表示一个 $n \times n$ 矩阵，其中对所有 $i \in [n]$，对角线上第 i，i 个元素为 x_i，其余元素均为零。

在许多理论计算机科学/机器学习文献中，$\exp(M)$ 表示矩阵指数，即 $\exp(M) = \sum_{i=0}^{\infty} \frac{1}{i!} M^i$。然而，在本文中，我们用 $\exp(M)$ 表示逐元素指数，即，

$$\exp(M)_{i,j} := \exp(M_{i,j}).$$

我们用 $\mathbf{1}_n$ 表示一个长度为 n 的向量，其所有元素均为 1。我们用 $\mathbf{0}_n$ 表示一个长度为 n 的向量，其所有元素均为 0。

我们给出一个在证明中会用到的标准事实。

事实 2.1（众所周知）。给定一组向量 $a_1, \cdots, a_k \in \mathbb{R}^n$ 和 $b_1, \cdots b_k \in \mathbb{R}^d$，则有

$$\sum_{i=1}^k a_i b_i^\top = AB^\top$$

其中，$A \in \mathbb{R}^{n \times k}$ 和 a_i 是 A的第 i列，且对于所有$i \in [k]$，$B \in \mathbb{R}^{d \times k}$ 和 b_i 是 B 的第 i列。

此外，我们有

- 第 *1* 部分。计算 $AB^\top$

  – 若采用朴素方法，需要 $O(nkd)$ 时间– 若使用快速矩阵乘法，需要 $\mathcal{T}_{\mathrm{mat}}(n, k, d)$ 时间

- 第 *2* 部分。对于任意矩阵 $C \in \mathbb{R}^{d \times d}$，计算 $AB^\top C$

  – 需要 $\mathcal{T}_{\mathrm{mat}}(n, k, d) + \mathcal{T}_{\mathrm{mat}}(n, d, d)$ 的时间；若使用快速矩阵乘法，先计算 $AB^\top$，再计算$(AB^\top)C$– 需要 $\mathcal{T}_{\mathrm{mat}}(k, d, d) + \mathcal{T}_{\mathrm{mat}}(n, k, d)$ 的时间；若使用快速矩阵乘法，先计算 $B^\top C$，再计算 $A(B^\top C)$

我们定义一种标准记法来描述矩阵乘法的运行时间，参见文献 [DI00,Zwi02, San04,San05, LG14, BN19,CLS19, LSZ19,BNS19, Bra20, GR21,JSWZ21,Bra21]例如。

定义 2.2。对于任意三个正整数，我们用 $\mathcal{T}_{\mathrm{mat}}(a, b, c)$ 表示将一个 $a \times b$ 矩阵与另一个 $b \times c$ 矩阵相乘所需的运行时间。

我们用 ω 表示 $n^\omega = \mathcal{T}_{\mathrm{mat}}(n, n, n)$ 的时间。目前 $\omega \approx 2.373$ [Wil12、LG14、AW21]。

定义 2.3。我们将 $\omega(\cdot, \cdot, \cdot)$ 函数定义如下：对于任意 a、b和 c，我们用 $\omega(a, b, c)$ 表示 $n^{\omega(a,b,c)} = \mathcal{T}_{\mathrm{mat}}(n^a, n^b, n^c)$。

3 技术概述

给定三个矩阵 $Q, K, V \in \mathbb{R}^{n \times d}$，我们需要计算由 $\mathrm{Att}(Q, K, V) = D^{-1}AV$ 给出的注意力，其中方阵 $A \in \mathbb{R}^{n \times n}$ 和对角矩阵 $D \in \mathbb{R}^{n \times n}$ 分别为 $A := \exp(QK^\top)$、$D := \mathrm{diag}(A\mathbf{1}_n)$。静态问题 [AS23] 只需针对给定的 Q、K 和 V计算 Att。在动态问题中，我们可以在每次迭代中获得 K 和 V 的更新。

对于 [AS23], 中的算法结果，他们采用了“算法设计中的多项式方法”。多项式方法是一种寻找注意力矩阵低秩近似的技术 A；如果矩阵元素有界，就可以高效地计算该近似。对于 [AS23], 中的困难性结果，他们假设强指数时间假设成立，并在归约中使用最近邻搜索困难性结果。

3.1 算法

问题表述 对于每次更新，我们接收 δ 作为输入，并更新矩阵 K 或 V中的一个元素。在查询函数中，我们接收索引 $i \in [n]$, $j \in [d]$ 作为输入，并返回目标矩阵 $B := D^{-1}AV$中的第$\{i, j\}$个元素。

令 $\widetilde{B}$ 表示更新后的目标矩阵 B。我们注意到，注意力的计算可以写为

$$\widetilde{B} = (D^{-1} + \Delta_D)(C + \Delta_C)$$

令 $\Delta^{(t)}$ 表示第 t 次迭代中的变化。采用延迟更新的方式，我们将 $\widetilde{B}$ 写成隐式形式

$$\widetilde{B} = (D^{-1} + \sum_{t=1}^{\mathrm{ct}} \Delta_D^{(t)})(C + \sum_{t=1}^{\mathrm{ct}} \Delta_C^{(t)})$$

其中，ct 表示自上次重新计算 D 和 C以来的更新次数。

延迟更新 我们提出一种延迟更新算法（算法 2），当键矩阵发生更新时，不计算注意力矩阵 K。我们还提出一种延迟更新算法（算法 3），当值矩阵发生更新时，不计算注意力矩阵 V。相反，我们维护一个数据结构（算法 1），使用列表$_C$、列表$_D$和列表$_V$，通过存储秩为 1 的矩阵来记录更新，直到迭代次数达到阈值 n^a，其中 a。在初始化阶段（算法1），我们计算精确的目标矩阵 $D^{-1}AV$和其他中间矩阵，耗时为 $O(\mathcal{T}_{\mathrm{mat}}(n, d, n))$（引理 4.3）。

重新计算 当迭代次数达到阈值 n^a 时，我们按如下方式重新计算数据结构中的所有变量（引理 4.8）。根据事实 2.1，我们首先将列表 $_C$ 中所有秩为 1 的矩阵堆叠起来，并仅进行一次矩阵乘法，使用 $\mathcal{T}_{\mathrm{mat}}(n, n^a, d)$ 的 $= n^{\omega(1,1,a)}$ 时间计算得到 $\sum_{t=1}^{\mathrm{ct}} \Delta_C^{(t)}$（引理 4.9）。然后，我们计算 $C + \sum_{t=1}^{\mathrm{ct}} \Delta_C^{(t)}$，得到重新计算后的 $\widetilde{C}$。类似地，为了重新计算 V，我们将列表 $_V$ 中所有秩为 1 的矩阵堆叠起来，并仅进行一次矩阵乘法，使用 $\mathcal{T}_{\mathrm{mat}}(n, n^a, d)$ 的 $= n^{\omega(1,1,a)}$ 时间计算得到 $\sum_{t=1}^{\mathrm{ct}} \Delta_V^{(t)}$。然后，我们计算 $V + \sum_{t=1}^{\mathrm{ct}} \Delta_V^{(t)}$，得到重新计算后的 $\widetilde{V}$。为了重新计算对角矩阵 D，我们将 $\sum_{t=1}^{\mathrm{ct}} \Delta_D^{(t)}$ 的所有更新相加，并将其加到旧的 D^{-1} 上（详见算法 5）。因此，我们的算法以 $n^{\omega(1,1,a)}/n^a$ 摊还时间更新 K 和 V（引理 4.4、引理 4.5）。

快速查询 回顾一下，查询函数以索引 $i \in [n], j \in [d]$ 为输入，并返回目标矩阵 $B := D^{-1}AV$中的第 $\{i, j\}$个元素。令 $\widetilde{D}^{-1}$ 表示从列表$_D$中获得的最新 D^{-1}。令 $\Delta_{V,1}$ 和 $\Delta_{V,2}$ 表示从 V中的列表获得的堆叠矩阵。我们可以将输出改写为

$$\begin{aligned}
&((\widetilde{D}^{-1}) \cdot (A) \cdot (V + \Delta_{V,1}\Delta_{V,2}))_{i,j} \\
= &((\widetilde{D}^{-1}) \cdot (A \cdot V))_{i,j} + ((\widetilde{D}^{-1}) \cdot A \cdot (\Delta_{V,1}\Delta_{V,2}))_{i,j} \\
= &(\widetilde{D})_i^{-1} C_{i,j} + (\widetilde{D})_i^{-1} A_{i,*} \Delta_{V,1} (\Delta_{V,2})_{*,j}.
\end{aligned}$$

请注意，我们在 C 重新计算函数中维护了 O（1）时间。由于$\Delta_{V\text{的每一列},1}$ 以及$\Delta_{V\text{的每一行},2}$ 都是 1-稀疏的，因此计算第二部分需要 O（n^a）时间。查询函数所需的总运行时间为 O（n^a）（引理 4.7，引理 4.6）。

3.2 困难性

现在我们转向下界结果，该结果受HMV 猜想 [BNS的启发。首先定义HMV 问题（形式化定义见定义 5.2）。

设计算在布尔半环上进行，并令 $m = n^{\tau}, \forall 0 < \tau \leq 1$。HMV问题包含以下三个阶段

- **阶段1.** 输入两个 $n \times n$ 矩阵 M 和 V
- **阶段2.** 输入一个 $n \times n$ 矩阵 P，其中至多有 n^{τ} 个非零元素
- **阶段3.** 输入一个单独的索引 $i \in [n]$– 要回答 $MPV_{*,}$
  $_{i}$– 这里 $V_{*,i} \in \mathbb{R}^{n}$ 是矩阵 V的第 i列

根据 [BNS19], ，上述问题被猜想在以下意义下是困难的,

猜想 3.1 (带提示的矩阵向量乘法 (HMV19])）。对于每个常数 $0 < \tau \leq 1$，不存在能够同时满足以下条件的带提示的矩阵向量乘法问题（定义 5.2）算法：

- 阶段1的多项式时间。
- $O(n^{\omega\ (1,\ 1,\ \tau)\ -\epsilon})$ 阶段2的时间复杂度，且
- $O(n^{1+\tau-\epsilon})$阶段3中

对于某个常数 $\epsilon > 0$。

我们的主要贡献在于展示如何将 OAMV（定义 5.4）和 ODAMV（定义 5.8）归约到 HMV问题（定义 5.2）。为此，我们采用了基于矛盾的方法。具体而言，我们首先假设存在一种算法，能够以多项式初始化时间解决 OAMV 问题，且摊还更新时间为 $O(\mathcal{T}_{\mathrm{mat}}(n, n^{\tau}, d)/n^{\tau+\Omega(1)})$，而最坏情况查询时间为 $O(n^{\tau-\Omega(1)})$，对所有 $\tau \in (0, 1]$ 均成立。我们的假设意味着存在一种比我们的结果（定理 4.1）更快的数据结构。随后，我们将证明，利用该算法还能过快地解决 HMV问题，从而与 HMV 猜想矛盾。

具体而言，我们取一个 HMV 问题（定义5.2）的实例

- 令 M，V $\in \{0,1\}^{n \times n}$ 表示来自 HMV 中**阶段1**的两个矩阵。

  ~我们创建一个新的实例 OAMV($\widetilde{n} = n$，$d = n$)，其中

$$\widetilde{Q} = \mathsf{M}, \quad \widetilde{K} = 0, \quad \widetilde{V} = \mathsf{V}$$

~ ~ ~在命题5.6 和命题5.7中，通过利用我们对 Q，K 和 V的构造，我们证明，对于每个 $i \in [n]$ 和 $j \in [n]$,

$$\text{If } ((\exp(\widetilde{Q}\widetilde{K}^{\top}) - \mathbf{1}_{n \times n})\widetilde{V})_{j,i} > 0, \text{ then } (\mathsf{MPV})_{j,i} = 1.$$
$$\text{If } ((\exp(\widetilde{Q}\widetilde{K}^{\top}) - \mathbf{1}_{n \times n})\widetilde{V})_{j,i} = 0, \text{ then } (\mathsf{MPV})_{j,i} = 0.$$

利用上述两个结论，我们知道，对于 HMV 问题（定义 5.2），exp($\widetilde{Q}\widetilde{K}^{\top})\widetilde{V}_{*,i}$ 足以重构 $\mathsf{MPV}_{*,i}$。然后，求解 $\mathsf{MPV}_{*,i}$ 需要多项式初始化时间，摊还更新时间为 $O(\mathcal{T}_{\mathrm{mat}}(n, n^{\tau}, d)/n^{\tau+\Omega(1)})$，而每个 $\tau \in (0, 1]$ 的最坏情况查询时间为 $O(n^{\tau-\Omega(1)})$。HMV 猜想所导致的矛盾表明，不存在这样的算法。类似地，对于归一化情形下的 ODAMV（定义 5.8）问题，我们展示如何重构 HMV 问题的另一个实例，并通过反证完成证明。

4 主要上界

在第4.1节中，我们展示初始化数据结构的运行时间。在第 4.2节中，我们展示更新 K 和 V的运行时间。在第 4.3节中，我们展示查询目标矩阵的正确性和运行时间。在第 4.4节中，我们展示重新计算数据结构中变量的正确性和运行时间。

我们将上界结果表述如下：

定理 4.1 （主要算法，定理 1.3 的正式版本）。 对于任意常数 $a \in (0, 1]$。令$d = O(n)$。存在一个动态数据结构，其空间使用量为 $O(n^2)$，并支持以下操作：

- Init(Q, K, V)。运行时间为 $O(\mathcal{T}_{\mathrm{mat}}(n, d, n))$。
- UpdateK$(i \in [n], j \in [d], \delta \in \mathbb{R})$。此操作会更新 Q 中的一个条目，摊还时间为$O(\mathcal{T}_{\mathrm{mat}}(n, n^a, n)/n^a)$。
- UpdateV$(i \in [n], j \in [d], \delta \in \mathbb{R})$。此操作的摊还时间与 Q 更新相同。
- Query$(i \in [n], j \in [d])$。此操作输出 $(D^{-1}(\exp(QK^\top))V)_{i,j}$ 操作的最坏情况运行时间为$O(n^a)$。

备注 4.2。 通过使用标准技术，例如参见 UpdateK 和 UpdateV 中的摊还时间可以转化为最坏情况时间，参见 *[BNS19]*的 B 节。

4.1 初始化

我们首先给出初始化过程的运行时间。

引理 4.3 (初始化)。 过程 Init *(算法 1)* 的运行时间为 $\mathcal{T}_{\mathrm{mat}}(n, d, n)$。

证明。 这显然可通过应用快速矩阵乘法得到。 □

4.2 更新

接下来，我们给出更新 K 的运行时间。

引理 4.4 (UpdateK**的运行时间)**。过程 UpdateK *(算法 2)* 的耗时为

- 第 *1* 部分。 $\mathcal{T}_{\mathrm{mat}}(n, n, n^a)$ 时间（最坏情况）
- 第 *2* 部分。 $\mathcal{T}_{\mathrm{mat}}(n, n, n^a)/n^a$ 时间（摊还情况）

证明。 **第 1 部分**。 这直接由引理4.9 得出

第 2 部分。 如果 $\mathsf{ct}_K < n^a$，我们需要花费 $O(n)$ 的时间。如果 $\mathsf{ct}_K = n^a$，我们需要花费 $n^{\omega(1,1,a)}$。因此，摊还时间为

$$\frac{n(n^a - 1) + n^{\omega(1,1,a)}}{n^a} = O(n^{\omega(1,1,a)-a})$$

注意，利用快速矩阵乘法以及 $d = O(n)$这一事实，我们有 $n^{\omega(1,1,a)} = \mathcal{T}_{\mathrm{mat}}(n, n^a, d)$。由此我们完成了证明。 □

现在，我们给出更新 V 的运行时间。

算法 1 动态数据结构 ▷ 定理4.1

数据结构 DynamicAttention
成员
  $Q \in \mathbb{R}^{n \times d}$ ▷ 查询令牌
  $K \in \mathbb{R}^{n \times d}$ ▷ 键令牌
  $V \in \mathbb{R}^{n \times d}$ ▷ 值令牌
  $M \in \mathbb{R}^{n \times n}$ ▷ logits 矩阵，$M = QK^\top$
  $A \in \mathbb{R}^{n \times n}$ ▷ 注意力矩阵，$A = \exp(QK^\top)$
  $D \in \mathbb{R}^{n \times n}$ ▷ 对角矩阵，
  $C \in \mathbb{R}^{n \times d}$ ▷ 中间矩阵，$C = \exp(QK^\top)V$
  $B \in \mathbb{R}^{n \times d}$ ▷ 目标矩阵，$B = D^{-1}AV$
  列表$_A$ ▷ 大小为 n^a 的列表
  列表$_C$ ▷ 大小为 n^a 的列表
  列表$_D$ ▷ 大小为 n^a 的列表
  ct_K，ct_V
末端成员
 ▷ 引理4.3
过程 Init(Q, K, V)
  $Q \leftarrow Q$，$K \leftarrow K$，$V \leftarrow V$
  $M \leftarrow QK^\top$，$A \leftarrow \exp(QK^\top)$
  $C \leftarrow \exp(QK^\top)V$
  $B \leftarrow D^{-1}AV$
  $\mathsf{ct}_K \leftarrow 0$
  $\mathsf{ct}_V \leftarrow 0$
结束过程
结束数据结构

引理 4.5 (运行时间 UpdateV)。过程 UpdateV *(算法 3)* 需要

- 第 *1* 部分。 $\mathcal{T}_{\text{矩阵}}(n, n, n^a)$ 最坏情况下的运行时间。
- 第 *2* 部分。 $\mathcal{T}_{\text{mat}}(n, n, n^a)/n^a$ 摊还情况下的时间。

证明。 **第 1 部分**。这可直接由引理4.9得出。

第 2 部分。如果 $\mathsf{ct}_K < n^a$，我们需要花费 $O(n)$ 的时间。如果 $\mathsf{ct}_K = n^a$，我们需要花费 $n^{\omega(1,1,a)}$。因此摊还时间为

$$\frac{n(n^a - 1) + n^{\omega(1,1,a)}}{n^a} = O(n^{\omega(1,1,a)-a})$$

注意，利用快速矩阵乘法以及 $d = O(n)$ 这一事实，我们有 $n^{\omega(1,1,a)} = \mathcal{T}_{\text{mat}}(n, n^a, d)$。由此完成证明。 □

4.3 查询

我们证明了我们的 Query 仅查询目标矩阵中的一个元素时的正确性。

引理 4.6 （ Query的正确性）。过程 Query （算法 *4*）输出

$$\begin{aligned} \widetilde{B}_{i,j} &= (D^{-1} \cdot A \cdot (V + \Delta_V))_{i,j} \\ &= (D^{-1}AV + D^{-1}A\Delta_V)_{i,j} \end{aligned}$$

算法 2 用于更新 K 并维护数据结构的算法

数据结构 DynamicAttention ▷ 定理4.1

过程 UpdateK($i \in [n], j \in [d], \delta$) ▷ 引理4.4

$\mathsf{ct}_K \leftarrow \mathsf{ct}_K + 1$

$K_{i,j} \leftarrow K_{i,j} + \delta$

$(\Delta_M)_{*,i} \leftarrow \delta \cdot \underbrace{Q}_{n \times d} \underbrace{e_j}_{d \times 1}$ ▷ Δ_M 仅在第 i列有条目

▷ 这里 $\circ$ 表示逐元素乘积

$(\Delta_A)_{*,\ i} \leftarrow (A_{*,\ i} \circ (\exp((\Delta_M)_{*,\ i}) - \mathbf{1}_n))$

$\widetilde{M} \leftarrow M + (\Delta_M)_{*,\ i} e_i^\top$ ▷ 我们只更新第 i 列 M

$\widetilde{A} \leftarrow A + (\Delta_A)_{*,\ i} e_i^\top$ ▷ 我们只更新第 i 列 A

获取对角向量 D_{tm} $_{\mathrm{p}}$ 从列表$_D[\mathsf{ct}_K - 1]$.GetB ▷ 耗时 $O(n)$

$\widetilde{D} \leftarrow D_{\mathrm{tmp}}^{-1} + \mathsf{diag}(\Delta_A)_{*,i}$

对于 $j = 1 \to n$**，执行**

$(\Delta_D)_{j,j} \leftarrow (D_{\mathrm{tmp}})_{j}^{-1}{}_{,j} - \widetilde{D}_{j\,j,}^{-1}$

结束循环

如果 $\mathrm{ct}_K < n^a$ **，则**

列表$_C[\mathsf{ct}_K - 1].(a, b) \leftarrow ((\Delta_A)_{*,i} \in \mathbb{R}^n, V^\top e_i \in \mathbb{R}^d)$

列表$_D[\mathsf{ct}_K - 1].(a,\ \ b) \leftarrow (\Delta_D \in \mathbb{R}^{n \times n}, \widetilde{D}^{-1} \in \mathbb{R}^{n \times n})$ ▷ 对角矩阵

else ▷ $\mathcal{T}_{\mathrm{mat}}(n, n^a, d) = n^{\omega(1,1,a)}$ 时间

ReCompute() ▷ 算法5。重新计算所有内容

结束 if

/*刷新内存*/

$K \leftarrow \widetilde{K}$

$A \leftarrow \widetilde{A}$

$M \leftarrow \widetilde{M}$

结束过程

结束数据结构

算法 3

数据结构 DynamicAttention ▷ 定理4.1

过程 UpdateV($i \in [n], j \in [d], \delta$) ▷ 引理4.5

$\mathsf{ct}_V \leftarrow \mathsf{ct}_V + 1$

如果 $\mathrm{ct}_V < n^a$ **那么**

列表$_V[\mathsf{ct}_V - 1].(a,\ \ b) \leftarrow (e_i \in \mathbb{R}^n,\ \ \delta e_j \in \mathbb{R}^d)$

else

ReCompute() ▷ 算法5. 重新计算所有内容

结束如果

结束过程

结束数据结构

证明。令 $\Delta_{V,\ 1}$ 表示从列表$_D[\mathsf{ct}_K]$.GetA中得到的向量。令 $\Delta_{V,\ 2}$ 表示从列表$_D[\mathsf{ct}_K]$.GetB中得到的向量。令 $(D_{\mathrm{tmp}})_i^{-1}$表示从列表$_D[\mathsf{ct}_K]$.GetB中得到的对角矩阵列表。我们知道

$$\widetilde{B} = ((D_{\mathrm{tmp}}^{-1}) \cdot (A) \cdot (V + \Delta_{V,1}\Delta_{V,2}))$$

算法 4 查询目标矩阵中第 $\{i, j\}$ 个元素的算法

数据结构 DynamicAttention ▷ 定理4.1
过程 Query($i \in [n], j \in [d]$) ▷ 引理4.7, 4.6
    设 $\Delta_{V,1}$ 和 $\Delta_{V,2}$ 是从 V获得的矩形矩阵 ▷ 这
    令 $(D_{\mathrm{tmp}})_i^{-1}$表示从列表$D[\mathsf{ct}_K]$.GetB获得的对角矩阵列表
耗时 $O(1)$
    /*以下是目标*/
    答案 $\leftarrow ((D_{\mathrm{tmp}}^{-1}) \cdot (A) \cdot (V + \Delta_{V,1}\Delta_{V,2}))_{i,j}$
    /*实际计算*/
    /*第 1 部分。答案，这很快，因为我们存储 $C = AV$*/
    答案$_1 \leftarrow (D_{\mathrm{tmp}})_i^{-1} C_{i,j}$ ▷ $O(1)$ 次
    /*第 2 部分。答案，这很快，因为 $\Delta_{V,1}$ 的每一列以及 $\Delta_{V,2}$ 的每一行都是 1-稀疏的*/
    答案$_2 \leftarrow (D_{\mathrm{tmp}})_i^{-1} A_{i,*} \Delta_{V,1} (\Delta_{V,2})_{*,j}$ ▷ n^a 时间
    答案 $\leftarrow \sum_{j=1}^{2}$答案$_j$
    返回 答案
结束过程
结束数据结构

算法 5 重新计算所有内容的算法

数据结构 DynamicAttention ▷ 定理4.1
过程 Recompute() ▷ 引理 4.9，引理4.8
    令 $\Delta_{C,1}$ 和 $\Delta_{C,2}$ 是由列表C得到的矩形矩阵
    令 $\Delta_{V,1}$ 和 $\Delta_{V,2}$ 是由列表V得到的矩形矩阵
    令 $\Delta_D(i)$表示由列表$D[i]$.GetA得到的对角矩阵列表
    $\widetilde{C} \leftarrow C + \Delta_{C,1} \cdot \Delta_{C,2}$ ▷ 需要 $\mathcal{T}_{\mathrm{mat}}(n, n^a, d)$ 的时间
    $\widetilde{V} \leftarrow V + \Delta_{V,1} \cdot \Delta_{V,2}$ ▷ 需要 $\mathcal{T}_{\mathrm{mat}}(n, n^a, d)$ 的时间
    $\Delta_D \leftarrow \sum_{i=1}^{\mathsf{ct}_K} \Delta_D(i)$ ▷ 耗时 n^{1+a}
    $\widetilde{D}^{-1} \leftarrow D^{-1} + \Delta_D$ ▷ 耗时 n
    $\widetilde{B} \leftarrow \widetilde{D}^{-1} \cdot \widetilde{C}$ ▷ 这需要 nd
    /*刷新内存*/
    $D \leftarrow \widetilde{D}, C \leftarrow \widetilde{C}, B \leftarrow \widetilde{B}, V \leftarrow \widetilde{V}$
    /*重置计数器*/
    $\mathsf{ct}_K \leftarrow 0, \mathsf{ct}_V \leftarrow 0$
结束过程
结束数据结构

$$= (D_{\mathrm{tmp}})^{-1}(AV) + (D_{\mathrm{tmp}})^{-1}(A\Delta_{V,1}\Delta_{V,2})$$

对于第 $\{i, j\}$ 个元素，通过简单的代数运算，我们有

$$\begin{aligned}
\widetilde{B}_{i,j} &= (D_{\mathrm{tmp}})_i^{-1}(AV)_{i,j} + (D_{\mathrm{tmp}})_i^{-1}(A\Delta_{V,1}\Delta_{V,2}) \\
&= (D_{\mathrm{tmp}})_i^{-1}(C)_{i,j} + (D_{\mathrm{tmp}})_i^{-1}(A\Delta_{V,1}\Delta_{V,2})_{i,j} \\
&= (D_{\mathrm{tmp}})_i^{-1}(C)_{i,j} + (D_{\mathrm{tmp}})_i^{-1} A_{i,*}\Delta_{V,1}(\Delta_{V,2})_{*,j}
\end{aligned}$$

我们知道

$$\mathrm{answer}_1 = (D_{\mathrm{tmp}})_i^{-1} C_{i,j}$$

and

$$\text{answer}_2 = (D_{\text{tmp}})_i^{-1} A_{i,*} \Delta_{V,1} (\Delta_{V,2})_{*,j}$$

将答案$_1$ 与答案$_2$,相加，我们得到

$$\widetilde{B}_{i,j} = (D^{-1}AV + D^{-1}A\Delta_V)_{i,j}.$$

现在，我们完成证明。 □

接下来，我们给出其运行时间。

引理 4.7 （ QUERY的运行时间）。 过程 QUERY （算法 4）的运行时间为O（n^a）。

证明。 我们首先将列表$_V$ 中的所有向量堆叠到 $\Delta_{V,\ 1} \in \mathbb{R}^{n \times n^a}$ 和 $\Delta_{V,\ 2} \in \mathbb{R}^{n^a \times d}$中，这需要 $O(1)$ 的时间。

- 计算 $(D_{\text{临时}})_i^{-1} C_{i,\ j}$ 需要 $O(1)$ 的时间。
- 计算 $(\Delta_{V,\ 1}\Delta_{V,\ 2})$ 需要 $O(n^a)$ 的时间，因为 $\Delta_{V,\ 1}$ 在列上是 1-稀疏的，而 $(\Delta_{V,\ 2})$ 在行上是 1-稀疏的。
- 计算 $(D_{\text{tmp}})_i^{-1} A_{i,\ *}(\Delta_{V,\ 1}\Delta_{V,\ 2})_{*,\ j}$ 需要 $O(n^a)$ 时间，与 $\text{nnz}((\Delta_{V,\ 1}\Delta_{V,\ 2})_{*,\ j})$ 相同 $\leq n^a$。

因此，所需的总运行时间为 $O(n^a)$ □

4.4 重新计算

我们证明了重新计算函数的正确性。

引理 4.8 (RECOMPUTE)。 过程 RECOMPUTE *(5)* 正确地重新计算 D， C， B， V。

证明。 **第 1 部分**。 重新计算 D

设 $\Delta_D(i)$ 表示从列表$_D[i]$.GETA获得的对角矩阵列表。然后，更新后的 $\widetilde{D}$ 和 D 之间的总差异为 $\sum_{i=1}^{\text{ct}_K} \Delta_D(i)$。

通过计算通过计算第对角矩阵的逆$\widetilde{\ }_D$, 我们得正确地得到更新后的 $\widetilde{\ }$

第2部分。 重新计算 V

我们首先将列表$_V$ 中的所有向量堆叠到 $\Delta_{V,1} \in \mathbb{R}^{n \times n^a}$ 和 $\Delta_{V,2} \in \mathbb{R}^{n^a \times d}$。

$\widetilde{\ }$利用事实2.1，我们有$V = V + \Delta_{V,1} \cdot \Delta_{V,2}$。

第 3 部分。重新计算 C

与重新计算 V的证明类似。

我们首先将列表$_C$ 中的所有向量堆叠起来，得到$\Delta_{C,1} \in \mathbb{R}^{n \times n^a}$ 和$\Delta_{C,2} \in \mathbb{R}^{n^a \times d}$。

利用事实2.1，我们有 $C = C + \Delta_{C,1} \cdot \Delta_{C,2}.$

第4部分。 重新计算 B根据 $B = D^{-1}C$的定义，我们可以使用 $\widetilde{B} = \widetilde{D}^{-1} \cdot \widetilde{C}$来更新 B 。现在，我们完成了证明。

□

接下来，我们给出它的运行时间。

引理 4.9 (运行时间：RECOMPUTE)。过程 RECOMPUTE *(算法 5)* 的运行时间为 $\mathcal{T}_{\mathrm{mat}}(n, n^a, d)$。

证明。我们首先将列表$_V$ 中的所有向量堆叠到 $\Delta_{V,1} \in \mathbb{R}^{n \times n^a}$ 和 $\Delta_{V,2} \in \mathbb{R}^{n^a \times d}$中，这需要 $O(1)$ 时间。

我们将列表$_C$ 中的所有向量堆叠到 $\Delta_{C,1} \in \mathbb{R}^{n \times n^a}$ 和 $\Delta_{C,2} \in \mathbb{R}^{n^a \times d}$中，这需要 $O(1)$ 时间。

- 计算 $C + \Delta_{C,1} \cdot \Delta_{C,2}$ 需要 $\mathcal{T}_{\mathrm{mat}}(n, n^a, d)$ 时间。
- 计算 $V + \Delta_{V,1} \cdot \Delta_{V,2}$ 需要 $\mathcal{T}_{\mathrm{mat}}(n, n^a, d)$ 时间。
- 计算 $\sum_{i=1}^{\mathrm{ct}_K} \Delta_D(i)$ 需要 $O(n^{a+1})$ 时间，因为 $\mathrm{nnz}(\Delta_D(i)) = O(n)$ 且 $\mathrm{ct}_K = O(n^a)$。
- 计算 $D^{-1} + \Delta_D$ 需要 $O(n)$ 时间，因为 $\mathrm{nnz}(\Delta_D) = O(n)$。
- 计算 $\widetilde{D}^{-1} \cdot \widetilde{C}$ 需要 $O(nd)$ 时间，因为 $\widetilde{D}^{-1}$ 是对角矩阵。因此，总运行时间为 $\mathcal{T}_{\mathrm{mat}}(n, n^a, d)$。

□

5 主要下界

在第 5.1 节中，我们给出在线矩阵向量乘法（OMV）问题的定义。在第 5.2 节中，我们介绍带提示的矩阵向量乘法的定义及其猜想（来自先前工作 [BNS19]）。在第5.3节中，我们展示了在没有归一化因子的情况下计算目标矩阵的困难性。在第 5.4 节中，我们展示了在有归一化因子的情况下计算目标矩阵的困难性。

5.1 在线矩阵向量乘法

在研究我们的问题的困难性之前，我们先回顾理论计算机科学中的一个著名问题，即在线矩阵向量乘法问题。以下是在线矩阵向量乘法的定义；它一直是许多基础优化问题中的一项关键任务。

定义 5.1 (在线矩阵向量（OMV） [HKNS15, LW17, CKL18])。给定一个矩阵 $A \in \{0,1\}^{n \times n}$，令 $T = O(n)$，存在一个在线向量序列 $u_1, \cdots, u_T \in \{0,1\}^n$。目标是设计一种数据结构，使其每当收到一个新向量时 u_t 并输出 Au_t。

学界普遍认为，不存在一种算法能以真正的次二次时间复杂度处理每个向量，也不存在一种算法能以真正的次立方时间复杂度处理所有向量。

5.2 先前工作的困难性

我们定义先前工作中提出的带提示的矩阵向量乘法问题 [BNS19]。

定义 5.2 （带提示的矩阵向量乘法（HMV） [BNS19]）。设计算在布尔半环上进行，并令 $m = n^\tau$，$0 < \tau \leq 1$。带提示的矩阵向量乘法问题包含以下几个阶段：

1. 输入两个 $n \times n$ 矩阵 M 和 V
2. 输入一个至多包含 n^τ 个非零元素的 $n \times n$ 矩阵 P

3. 输入单个索引 $i \in [n]$

- 我们需要得到答案 $MPV_{*,\ i}$
- 这里 $V_{*,\ i} \in \mathbb{R}^n$ 是矩阵 V 的第 i 列

我们给出源自先前工作的提示式矩阵向量猜想 [BNS19]。

猜想 5.3 (带提示的矩阵向量乘法 猜想 [BNS19], 猜想的重述 3.1)。 对于每个常数$0 < \tau \leq 1$，不存在能够同时满足以下条件的带提示的矩阵向量乘法问题算法（定义 *5.2*）

- 第 *1* 阶段的多项式时间
- $O(n^{\omega(1,\ 1,\ \tau)-\epsilon})$ 第 *2* 阶段的时间复杂度，且
- $O(n^{1+\tau-\epsilon})$在阶段 *3*

对于某个常数 $\epsilon > 0$。

5.3 在线注意力矩阵向量乘法

我们在此定义动态注意力矩阵向量问题。对于以下定义，我们忽略归一化因子的影响。我们将在后面的部分处理它。

定义 5.4 (OAMV(n, d))。 在线注意力矩阵向量乘法问题 ***Online Attention Matrix Vector Multiplication problem*** OAMV(n, d) 的目标是设计一个满足以下操作的数据结构：

1. Init：在 $n \times d$ 矩阵 Q, K, V 上初始化。 *2.* Update：更改 Q、 K 或 V 的任何条目。 *3.* Query：对于任何给定的 $i \in [n]$, $j \in [d]$，返回 $(\exp(QK^\top)V)_{i,j}$ 。

接下来，我们给出忽略归一化因子的下界结果。

引理 5.5. 假设提示的 Mv 猜想（猜想 *5.3*）：对于每个常数 $0 < \tau \leq 1$，不存在用于 OAMV(n, d) 问题（定义 *5.4*）的动态算法，其

- 多项式初始化时间，且
- 摊还更新时间 $O(\mathcal{T}_{\mathrm{mat}}(n, n^\tau, d)/n^{\tau+\Omega(1)})$，且
- 最坏查询时间 $O(n^{\tau-\Omega(1)})$。

证明. 假设存在一个动态算法，对于某个参数 τ，其速度快于引理 5.5 中的界限，即更新时间为 $O(\mathcal{T}_{\mathrm{mat}}(n, n^\tau, d)/n^{\tau+\epsilon})$，查询时间为 $O(n^{\tau-\epsilon})$，其中某个常数$\epsilon > 0$。我们将证明，这与提示的 Mv 猜想（猜想 5.3）矛盾。

考虑带提示的 v-提示的矩阵向量乘法问题（定义5.2），其中$\mathsf{M}, \mathsf{V} \in \{0,1\}^{n\times n}$。~我们构造一个新的实例OAMV（$\widetilde{n} = n,\ d = n$），其中

$$\widetilde{Q} = \mathsf{M}, \quad \widetilde{K} = 0, \quad \widetilde{V} = \mathsf{V}$$

在阶段 1，我们将此输入提供给用于 OAMV 问题的动态算法（定义 5.4）。在阶段 2，当我们收到 $n\times n$ 矩阵 P，其中包含 n^τ 个非零项时，我们对数据结构执行 n^τ 次更新，将 $\widetilde{K}^\top$ =P设置为该矩阵。这需要

$$O(\widetilde{n}^\tau \cdot (\mathcal{T}_{\mathrm{mat}}(\widetilde{n}, \widetilde{n}^\tau, \widetilde{d})/\widetilde{n}^{\tau+\epsilon})) = O(n^{\omega(1,1,\tau)-\epsilon})$$

时间。

最后，在阶段 3，我们执行 $\widetilde{n}$ 次查询以获得列 $\exp(\widetilde{Q}\widetilde{K}^\top)\widetilde{V}_{*,i}$ 其时间复杂度为 $O(\widetilde{n}\cdot\widetilde{n}^{\tau-\epsilon}) = O(n^{1+\tau-\epsilon})$。

根据命题 5.6 和命题 5.7，我们知道 $\exp(\widetilde{Q}\widetilde{K}^\top)\widetilde{V}_{*,i}$ 足以重构 $\mathsf{MPV}_{*,i}$ 用于该提示问题 Mv 。

□

命题 5.6。 对于每个 $i \in [n]$ 和 $j \in [n]$，如果 $((\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{n\times n})\widetilde{V})_{j,i}$ 是 > 0，那么 $(\mathsf{MPV})_{j,i} = 1$，

证明。 假设我们有

$$((\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{n\times n})\widetilde{V})_{j,i} > 0,$$

~ ~ ~我们定义了 $Q = \mathsf{M}$ $K = \mathsf{P}$ $V = \mathsf{V}$，因此可以将其改写为

$$((\exp(\mathsf{MP}) - \mathbf{1}_{n\times n})\mathsf{V})_{j,i} > 0.$$

根据矩阵乘法的定义，以及 $\exp(x) > 1$ 对所有 $x > 0$都成立这一事实，我们有某个 $k \in [n]$ 满足

$$\begin{aligned}((\exp(\mathsf{MP}) - \mathbf{1}_{n\times n})_{j,k}(\mathsf{V})_{k,i} &> 0\\ ((\exp(\mathsf{MP})_{j,k} - 1)(\mathsf{V})_{k,i} &> 0\end{aligned}$$

We 可以得出结论：对于每个 $i \in [n]$， $j \in [n]$，至少存在一个 $k \in [n]$ 使得 th at

- $\mathsf{V}_{k,i} > 0$
- $(\mathsf{MP})_{j,k} > 0$

因此，通过使用布尔半环的定义，我们可以得出 $(\mathsf{MPV})_{j,i} = 1$

□

断言 5.7. 对于每个 $i \in [n]$ 和 $j \in [n]$，如果 $((\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{n\times n})\widetilde{V})_{j,i}$ 为 0 则 $(\mathsf{MPV})_{j,i} = 0$.

证明. 我们有

$$\begin{aligned}&((\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{n\times n})\widetilde{V})_{j,k}\\ =\ &((\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{n\times n}))_{j,*}\widetilde{V}_{*,i}\\ =\ &((\exp(\mathsf{MP}) - \mathbf{1}_{n\times n}))_{j,*}\mathsf{V}_{*,i}\end{aligned}$$

其中第一步遵循矩阵乘法的定义，第二步遵循~ ~ ~来自 Q, K 和 V 的定义。

通过使用上述方程，如果 $((\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{n\times n})\widetilde{V})_{j,k} = 0$，我们有

$$(\exp(\mathsf{MP}) - \mathbf{1}_{n\times n})_{j,*}\mathsf{V}_{*,i} = 0 \tag{1}$$

等式 (1) 意味着，对于所有 $k \in [n]$ 使得 $\mathsf{V}_{k,i} = 1$，我们有 $(\exp(\mathsf{MP}) - \mathbf{1}_{n\times n})_{j,k} = 0$，这也意味着 $(\mathsf{MP})_{j,k} = 0$。

现在，我们可以得出结论：$(\mathsf{MPV})_{j,i} = 0$ 对于每个 $i \in [n]$ 和 $j \in [n]$。 □

5.4 在线对角归一化注意力矩阵向量乘法

接下来, 我们考虑归一化因子，并将问题定义如下 .

定义 5.8 (ODAMV(n,d)，定义 1.2 的重述。 目标是为 ***O**nline **D**iagonal-basednormalized **A**ttention **M**atrix **V**ector **M**ultiplication* 问题 ODAMV(n,d) 设计一种满足以下操作要求的数据结构：

1. Init：在 $n \times d$ 矩阵 Q、 K、 V 上进行初始化。
2. Update：更改 Q、 K 或 V 中的任意一个元素。
3. Query：对于任意给定的 $i \in [n]$、 $j \in [d]$，返回 $(D^{-1}\exp(QK^\top)V)_{i,\ j}$，其中 $D = \mathrm{diag}(\exp(QK^\top)\mathbf{1}_n)$。

接下来，我们给出带有归一化因子的下界结果。

引理 5.9. 假设提示的 Mv 猜想（猜想 5.3）：对于任意常数 $0 < \tau \leq 1$，不存在能够解决 ODAMV(n,d) 问题（定义 5.8）的算法，且其

- 多项式初始化时间，且
- 摊还更新时间 $O(\mathcal{T}_{\mathrm{mat}}(n, n^\tau, d)/n^{\tau+\Omega(1)})$，且
- 最坏情况下查询时间 $O(n^{\tau-\Omega(1)})$。

证明. 假设存在一种动态算法，对某个参数 τ，其速度快于引理 5.9 所述的算法，即更新时间为 $O(\mathcal{T}_{\mathrm{mat}}(n, n^\tau, d)/n^{\tau+\epsilon})$，查询时间为 $O(n^{\tau-\epsilon})$，其中某个常数$\epsilon > 0$。我们将证明，这将与提示的 Mv 猜想（猜想 5.3）矛盾。

取一个 v-带提示的矩阵向量乘法问题（定义5.2）的实例，其中 $M \in \{0,1\}^{n\times n}$， $V \in \{0,1\}^{n\times n}$.

我们可以如下构造矩阵 $\mathsf{M} \in \{0,1\}^{n\times 2n}$ 和 $\mathsf{V} \in \{0,1\}^{2n\times n}$

$$\mathsf{M} := \begin{bmatrix} M & \overline{M} \end{bmatrix} \quad \text{and} \quad \mathsf{V} := \begin{bmatrix} V \\ \mathbf{0}_{n\times n} \end{bmatrix}$$

其中 $\overline{M}$是一个矩阵，满足 $\overline{M_{i,\ j}} = 1 - M_{i,\ j}$。

注意，对于每个 $i \in [n]$，都有 $\|\mathsf{M}_{i,\ *}\|_1 = n$。

根据上述构造，我们将创建一个新的 ODAMV($\widetilde{n} = 2n$, $\widetilde{d} = 2n$)，其中e

$$\widetilde{Q} = \begin{bmatrix} \mathsf{M} \\ \mathbf{0}_{n\times 2n} \end{bmatrix}, \quad \widetilde{K} = \mathbf{0}_{2n\times 2n}, \quad \widetilde{V} = \begin{bmatrix} \mathsf{V} & \mathbf{0}_{2n\times n} \end{bmatrix}$$

在阶段1中，我们将此输入提供给用于解决ODAMV 问题的动态算法（定义5.8）。

令 $D \in \{0,1\}^{n\times n}$ 表示一个对角矩阵，其中 nnz(D) $= n^\tau$

在阶段2中，我们收到 $2n \times 2n$ 对角矩阵P，其中

$$\mathsf{P} = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix}$$

且 nnz(P) $= 2n^\tau$。

我们对数据结构执行 $2n^\tau$ 次更新，以设定 $\widetilde{K}^\top =$P。此操作需要

$$O(\widetilde{n}^\tau \cdot (\mathcal{T}_{\mathrm{mat}}(\widetilde{n}, \widetilde{n}^\tau, \widetilde{d})/\widetilde{n}^{\tau+\epsilon})) = O(n^{\omega(1,1,\tau)-\epsilon})$$

时间。

请注意

- $\|\widetilde{Q}_{i,*}\|_1 = n$，对每个 $i \in [n]$。 ~

- $\|Q_{i,*}\|_1 = 0$，对于每个 $i \in [n+1, 2n]$。

根据 P 的定义，我们知道，对于每个 $i \in [n]$

$$\widetilde{D}_{i,i} = n^\tau \exp(1) + n^\tau \exp(0) = n^\tau (e+1).$$

对于每个 $i \in [n+1, 2n]$

$$\widetilde{D}_{i,i} = n^\tau \exp(0) = n^\tau. \tag{2}$$

~因此，我们无需更新 D。

最后，在阶段 3，我们执行 $\widetilde{n}$ 次查询，以在 $O(\widetilde{n} \cdot \widetilde{n}^{\tau-\epsilon}) = O(n^{1+\tau-\epsilon})$ 时间内获得列 $\exp(\widetilde{Q}\widetilde{K}^\top)\widetilde{V}_{*,i}$。

根据断言 5.11 和断言 5.10，我们知道，对于任意 $i \in [n]$ 和任意 $j \in [n]$，如果存在一个算法能够找到 $(\widetilde{D}^{-1}\exp(\widetilde{Q}\widetilde{K}^\top)\widetilde{V})_{j,i}$，那么利用 $(\widetilde{D}^{-1}\exp(\widetilde{Q}\widetilde{K}^\top)\widetilde{V})_{j,i} - (\widetilde{D}^{-1}\widetilde{V})_{j,i}$ 就足以重构 $(\mathrm{MPV})_{j,i}$。这里，$\widetilde{D}^{-1}\widetilde{V}$ 可通过式 (2) 在仅 $O(1)$ 时间内计算得出。因此，我们可以在 $O(n^{1+\tau\epsilon})$ 时间内获知提示的 Mv 问题的 $(MDV)_{j,i}$，这与提示的 Mv 猜想矛盾。

□

命题 5.10. 对于每个 $i \in [n]$ 和 $j \in [n]$，如果 $(\widetilde{D}^{-1}(\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{\widetilde{n}\times\widetilde{n}})\widetilde{V})_{j,i}$ 为 > 0，那么 $(\mathrm{MPV})_{j,i} = 1$,

证明。 利用 $n^\tau(e+1) > 0$ 和 $n^\tau > 0$这一事实，我们得到

$$\widetilde{D}^{-1}(\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{\widetilde{n}\times\widetilde{n}})\widetilde{V})_{j,i} > 0$$
$$((\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{\widetilde{n}\times\widetilde{n}})\widetilde{V})_{j,i} > 0$$

我们知道

$$\widetilde{Q} = \begin{bmatrix} \mathsf{M} \\ \mathbf{0}_{n\times 2n} \end{bmatrix}, \quad \widetilde{K}^\top = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix}, \quad \widetilde{V} = \begin{bmatrix} \mathsf{V} & \mathbf{0}_{2n\times n} \end{bmatrix},$$

因此我们得到

对于 $k \in [n+1, 2n]$，由于 $\mathsf{V} = \begin{bmatrix} V \\ \mathbf{0}_{n\times n} \end{bmatrix}$，我们知道 $(\exp(\mathrm{MP}) - \mathbf{1}_{n\times 2n})_{j,k}(\mathsf{V})_{k,i} = 0$。 $((\exp(\mathsf{MP}) - \mathbf{1}_{n\times 2n})\mathsf{V})_{j,i} > 0.$

根据矩阵乘法的定义，以及对所有 $x > 0$，$\exp(x) > 1$ 这一事实，我们可以找到某个 $k \in [n]$，使得

$$(\exp(\mathsf{MP}) - \mathbf{1}_{n\times 2n})_{j,k}(\mathsf{V})_{k,i} > 0$$
$$(\exp(\mathsf{MP})_{j,k} - 1)(\mathsf{V})_{k,i} > 0$$

We 可以得出结论，对于每个 $i \in [n], j \in [n]$，至少存在一个 $k \in [n]$ 满足条件 at

- $\mathsf{V}_{k,i} > 0$

- $(\mathrm{MP})_{j,k} > 0$

因此，根据布尔半环的定义，我们可以得出结论：$(\mathrm{MPV})_{j,i} = 1$

□

命题 5.11。对于每个 $i \in [n]$ 和 $j \in [n]$，若 $(\widetilde{D}^{-1}(\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{\widetilde{n}\times\widetilde{n}})\widetilde{V})_{j,i}$ 为 0 则 $(\mathrm{MPV})_{j,i} = 0$。证明。利用以下事实：$n^\tau(e+1) > 0$ 和 $n^\tau > 0$，我们得到

$$\widetilde{D}^{-1}(\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{\widetilde{n}\times\widetilde{n}})\widetilde{V})_{j,i} = 0$$
$$((\exp(\widetilde{Q}\widetilde{K}^\top) - \mathbf{1}_{\widetilde{n}\times\widetilde{n}})\widetilde{V})_{j,i} = 0$$

我们知道

$$\widetilde{Q} = \begin{bmatrix} \mathsf{M} \\ \mathbf{0}_{n\times 2n} \end{bmatrix}, \quad \widetilde{K}^\top = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix}, \quad \widetilde{V} = \begin{bmatrix} \mathsf{V} & \mathbf{0}_{2n\times n} \end{bmatrix},$$

因此我们有

$$((\exp(\mathsf{MP}) - \mathbf{1}_{n\times 2n})\mathsf{V})_{j,i} = 0.$$

对于 $k \in [n+1, 2n]$，当 $\mathsf{V} = \begin{bmatrix} V \\ \mathbf{0}_{n\times n} \end{bmatrix}$ 时，我们知道 $(\exp(\mathrm{MP}) - \mathbf{1}_{n\times 2n})_{j,k}(\mathrm{V})_{k,i} = 0$。

对于所有满足 $\mathrm{V}_{k,i} = 1$ 的 $k \in [n]$，我们有 $(\exp(\mathrm{MP}) - \mathbf{1}_{n\times 2n})_{j,k} = 0$，这也意味着 $(\mathrm{MP})_{j,k} = 0$。

现在，我们可以得出结论：$(\mathrm{MPV})_{j,i} = 0$ 对于每个 $i \in [n]$ 和 $j \in [n]$。 □

参考文献

[ACSS20] Josh Alman、Timothy Chu、Aaron Schild 和 Zhao Song。《几何图上的线性代数算法与困难性》。载于*2020 年 IEEE 第 61 届年度计算机科学基础研讨会（FOCS）*，第 541–552 页。IEEE，2020 年。[AS23] Josh Alman 和 Zhao Song。快速注意力需要有界条目。*arXiv* 预印本 *arXiv:2302.13214*，2023 年。[AW21] Josh Alman 和 Virginia Vassilevska Williams。《改进的激光方法与更快的矩阵乘法》。载于*2021 年 ACM-SIAM 离散算法研讨会论文集（SODA）*，第 522–539 页。SIAM，2021 年。[BAG20] Satwik Bhattamishra、Kabir Ahuja 和 Navin Goyal。《Transformer 识别形式语言的能力与局限》。载于*2020 年自然语言处理经验方法会议论文集（EMNLP）*，第 7096–7116 页，线上，2020 年 11 月。计算语言学协会。[Bel22] Yonatan Belinkov。《探测分类器：前景、不足与进展》。计算语言学，48(1):207–219，2022 年 3 月。[BIS17] Arturs Backurs、Piotr Indyk 和 Ludwig Schmidt。《经验风险最小化的细粒度复杂性：核方法与神经网络》。神经信息处理系统进展，第 30 卷，2017 年。[BLSS20] Jan van den Brand、Yin Tat Lee、Aaron Sidford 和 Zhao Song。《近线性时间求解高而稠密的线性规划》。载于第 *52 届 ACM SIGACT 理论计算年度研讨会论文集*，第 775–788 页，2020 年。

[BMR+20] Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 等。《语言模型是少样本学习者》。神经信息处理系统进展，33:1877–1901，2020 年。[BN19] Jan van den Brand 和 Danupon Nanongkai。《动态近似最短路径及其他问题：次二次复杂度与最坏情况下的更新时间》。载于*2019* 年 *IEEE* 第 *60* 届年度计算机科学基础研讨会（*FOCS*），第 436–455 页。IEEE，2019 年。[BNS19] Jan van den Brand、Danupon Nanongkai 和 Thatchaphol Saranurak。《动态矩阵求逆：改进算法与匹配的条件下界》。载于*2019* 年 *IEEE* 第 *60* 届年度计算机科学基础研讨会（*FOCS*），第 456–480 页。IEEE，2019 年。[BPG20] Satwik Bhattamishra、Arkil Patel 和 Navin Goyal。《Transformer 的计算能力及其对序列建模的影响》。载于第 *24* 届计算自然语言学习会议论文集，第 455–475 页，线上，2020 年 11 月。计算语言学协会。[Bra20] Jan van den Brand。《以当前矩阵乘法时间实现的确定性线性规划求解器》。载于第十四届 *ACM-SIAM* 离散算法年度研讨会论文集（*SODA*），第 259–278 页。SIAM，2020 年。[Bra21] Jan van den Brand。《统一矩阵数据结构：简化并加速迭代算法》。载于算法简洁性研讨会（*SOSA*），第 1–13 页。SIAM，2021 年。[CKL18] Diptarka Chakraborty、Lior Kamma 和 Kasper Green Larsen。《紧确的单元探测界：简洁布尔矩阵-向量乘法》。载于第 *50* 届 *ACM SIGACT* 理论计算年度研讨会论文集（*STOC*），第 1297–1306 页，2018 年。[CKLM19] Kevin Clark、Urvashi Khandelwal、Omer Levy 和 Christopher D. Manning。《BERT 在关注什么？对 BERT 注意力的分析》。载于*2019* 年 *ACL* 自然语言处理黑箱分析与神经网络解读研讨会论文集，第 276–286 页，意大利佛罗伦萨，2019 年 8 月。计算语言学协会。[CKNS20] Moses Charikar、Michael Kapralov、Navid Nouri 和 Paris Siminelakis。《通过密度约束近邻搜索进行核密度估计》。载于*2020* 年 *IEEE* 第 *61* 届年度计算机科学基础研讨会（*FOCS*），第 172–183 页。IEEE，2020 年。[CLP+21] Beidi Chen、Zichang Liu、Binghui Peng、Zhaozhuo Xu、Jonathan Lingjie Li、Tri Dao、Zhao Song、Anshumali Shrivastava 和 Christopher Re。《Mongoose：用于高效神经网络训练的可学习 LSH 框架》。载于国际学习表征会议，2021 年。[CLS19] Michael B Cohen、Yin Tat Lee 和 Zhao Song。《以当前矩阵乘法时间求解线性规划》。载于$STOC$，2019 年。[CND+22] Aakanksha Chowdhery、Sharan Narang、Jacob Devlin、Maarten Bosma、Gaurav Mishra、Adam Roberts、Paul Barham、Hyung Won Chung、Charles Sutton、Sebastian Gehrmann 等。PaLM：借助 pathways 扩展语言建模。*arXiv* 预印本 *arXiv:2204.02311*，2022 年。

[DCLT18] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。BERT：用于语言理解的深度双向 Transformer 预训练。*arXiv* 预印本 *arXiv:1810.04805*，2018 年。[DI00] Camil Demetrescu 和 Giuseppe F Italiano。完全动态传递闭包：突破 o (n/sup 2/) 屏障。载于第 *41* 届计算机科学基础年会论文集，第 381–389 页。IEEE，2000 年。[DLY21] Sally Dong、Yin Tat Lee 和 Guanghao Ye。一种用于小树宽线性规划的近线性时间算法：鲁棒中心路径的多尺度表示。载于第 *53* 届 *ACM SIGACT* 计算理论年会论文集，第 1784–1797 页，2021 年。[EGKZ22] Benjamin L Edelman、Surbhi Goel、Sham Kakade 和 Cyril Zhang。自注意力机制中的归纳偏置与变量创建。载于 Kamilika Chaudhuri、Stefanie Jegelka、Le Song、Csaba Szepesvari、Gang Niu 和 Sivan Sabato 编，第 *39* 届国际机器学习会议论文集，第 162 卷，机器学习研究论文集，第 5793–5831 页。PMLR，2022 年 7 月 17–23 日。[EGZ20] Javid Ebrahimi、Dhruv Gelda 和 Wei Zhang。自注意力网络如何识别 Dyck-n 语言？载于计算语言学协会论文集：*EMNLP 2020*，第 4301–4306 页，线上会议，2020 年 11 月。计算语言学协会。[GLSS18] Ankit Garg、Yin Tat Lee、Zhao Song 和 Nikhil Srivastava。矩阵扩张器 Chernoff 界。载于第 *50* 届 *ACM SIGACT* 计算理论年会论文集，第 1102–1114 页，2018 年。[GR21] Yong Gu 和 Hanlin Ren。如何在 $o(n^{2.5794}m)$ 时间内构造距离敏感性预言机。*arXiv* 预印本 *arXiv:2102.08569*，2021 年。[GS22]Yuzhou Gu 和 Zhao Song。一种更快的小树宽半正定规划求解器。*arXiv* 预印本 *arXiv:2211.06033*，2022 年。[HJS+22] Baihe Huang、Shunhua Jiang、Zhao Song、Runzhou Tao 和 Ruizhe Zhang。更快地求解半正定规划：鲁棒内点法框架与高效实现。载于*2022* 年 *IEEE* 第 *63* 届计算机科学基础年会（*FOCS*），第 233–244 页。IEEE，2022 年。[HKNS15] Monika Henzinger、Sebastian Krinninger、Danupon Nanongkai 和 Thatchaphol Saranurak。通过在线矩阵向量乘法猜想统一并加强动态问题的困难性。载于第 *47* 届 *ACM* 计算理论年会论文集（*STOC*），第 21–30 页，2015 年。[HL19] John Hewitt 和 Percy Liang。利用控制任务设计和解释探针。载于*2019* 年自然语言处理经验方法会议暨第 *9* 届国际自然语言处理联合会议论文集（*EMNLP-IJCNLP*），第 2733–2743 页，中国香港，2019 年 11 月。计算语言学协会。[HM19] John Hewitt 和 Christopher D. Manning。用于在词表示中发现句法的结构探针。载于 2019 年北美计算语言学协会会议论文集

计算语言学协会：人类语言技术，第 1 卷（长篇与短篇论文），第 4129–4138 页，美国明尼苏达州明尼阿波利斯，2019 年 6 月。计算语言学协会。[JKL+20] Haotian Jiang、Tarun Kathuria、Yin Tat Lee、Swati Padmanabhan 和 Zhao Song。一种更快的半正定规划内点法。载于 2020 年 IEEE 第 61 届计算机科学基础年会（FOCS），第 910–918 页。IEEE，2020 年。[JLSW20] Haotian Jiang、Yin Tat Lee、Zhao Song 和 Sam Chiu-wai Wong。一种改进的凸优化与凸凹博弈切平面法及其应用。载于第 52 届 ACM SIGACT 计算理论年会论文集，第 944–953 页，2020 年。[JSWZ21] Shunhua Jiang、Zhao Song、Omri Weinstein 和 Hengjie Zhang。更快地进行动态矩阵求逆，以更快地求解线性规划。载于 STOC。arXiv 预印本 arXiv:2004.07470，2021 年。[KKL20] Nikita Kitaev、 ̵Lukasz Kaiser 和 Anselm Levskaya。Reformer：高效的 Transformer。arXiv 预印本 arXiv:2001.04451，2020 年。[KVPF20] Angelos Katharopoulos、Apoorv Vyas、Nikolaos Pappas 和 Fraņcois Fleuret。Transformer 即 RNN：具有线性注意力的快速自回归 Transformer。载于国际机器学习会议，第 5156–5165 页。PMLR，2020 年。[LG14] Fraņcois Le Gall。张量幂与快速矩阵乘法。载于第 39 届国际符号与代数计算研讨会论文集（ISSAC '14），2014 年。[LLR23] Yuchen Li、Yuanzhi Li 和 Andrej Risteski。Transformer 如何学习主题结构：迈向机制层面的理解。arXiv 预印本 arXiv:2303.04245，2023 年。[LSZ19] Yin Tat Lee、Zhao Song 和 Qiuyi Zhang。以当前矩阵乘法时间复杂度求解经验风险最小化问题。载于 COLT，2019 年。[LSZ23] Zhihang Li、Zhao Song 和 Tianyi Zhou。求解正则化 exp、cosh 和 sinh 回归问题。arXiv 预印本 2303.15725，2023 年。[LW17] Kasper Green Larsen 和 Ryan Williams。更快的在线矩阵向量乘法。载于第二十八届 ACM-SIAM 离散算法研讨会论文集（SODA），第 2182–2189 页，2017 年。[Ope23] OpenAI。GPT-4 技术报告，2023 年。[PMB19] Jorge P´erez、Javier Marinkovi´c 和 Pablo Barcel´o。关于现代神经网络架构的图灵完备性。arXiv 预印本 arXiv:1901.03429，2019 年。[RNS+18] Alec Radford、Karthik Narasimhan、Tim Salimans、Ilya Sutskever 等。通过生成式预训练提升语言理解能力。2018 年。[San04] Piotr Sankowski。通过动态矩阵求逆实现动态传递闭包。载于第 45 届 IEEE 计算机科学基础年会论文集，第 509–517 页。IEEE，2004 年。

[San05] Piotr Sankowski. 动态最短距离的次二次时间算法。载于《计算与组合：第 *11* 届年度国际会议 *COCOON 2005，2005* 年 *8* 月 *16–19* 日，中国昆明，会议论文集 *11*》，第 461–470 页。Springer，2005 年。[SS92] Hava T. Siegelmann 和 Eduardo D. Sontag。论神经网络的计算能力。载于《第五届年度计算学习理论研讨会论文集》，COLT '92，第 440–449 页，美国纽约州纽约市，1992 年。计算机协会。[SY21] Zhao Song 和 Zheng Yu。基于无偏草图的中心路径法，用于求解线性规划问题。载于第 *38* 届国际机器学习会议（*ICML*），2021 年。[TDP19] Ian Tenney、Dipanjan Das 和 Ellie Pavlick。BERT 重新发现了经典的自然语言处理流程。载于计算语言学协会第 *57* 届年会论文集，第 4593–4601 页，意大利佛罗伦萨，2019 年 7 月。计算语言学协会。[VB19] Jesse Vig 和 Yonatan Belinkov。分析 Transformer 语言模型中的注意力结构。载于《*2019* 年 *ACL* 黑盒自然语言处理研讨会论文集：分析和解释用于自然语言处理的神经网络》，第 63–76 页，意大利佛罗伦萨，2019 年 8 月。计算语言学协会。[VSP+17] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、 ˙Lukasz Kaiser 和 Illia Polosukhin。注意力就是一切。神经信息处理系统进展，第 30 卷，2017 年。[WCM21] Colin Wei、Yining Chen 和 Tengyu Ma。具有统计意义的近似：用 Transformer 近似图灵机的案例研究，2021 年。[Wil12] Virginia Vassilevska Williams。比 Coppersmith-Winograd 更快的矩阵乘法。载于第四十四届 *ACM* 年度计算理论研讨会（*STOC*）论文集，第 887–898 页。ACM，2012 年。[WLK+20] Sinong Wang、Belinda Z Li、Madian Khabsa、Han Fang 和 Hao Ma。Linformer：具有线性复杂度的自注意力。*arXiv* 预印本 *arXiv:2006.04768*，2020 年。[YBR+20] Chulhee Yun、Srinadh Bhojanapalli、Ankit Singh Rawat、Sashank Reddi 和 Sanjiv Kumar。Transformer 是序列到序列函数的通用逼近器吗？载于国际学习表征会议，2020 年。[YPPN21] Shunyu Yao、Binghui Peng、Christos Papadimitriou 和 Karthik Narasimhan。自注意力网络能够处理有界层次语言。载于计算语言学协会第 *59* 届年会暨第 *11* 届自然语言处理国际联合会议论文集（第 *1* 卷：长篇论文），第 3770–3785 页，线上举行，2021 年 8 月。计算语言学协会。[ZBB+22] Yi Zhang、Arturs Backurs、S´ebastien Bubeck、Ronen Eldan、Suriya Gunasekar 和 Tal Wagner。用乐高揭示 Transformer：一项合成推理任务，2022 年。[ZBIW19] Ofir Zafrir、Guy Boudoukh、Peter Izsak 和 Moshe Wasserblat。Q8bert：量化的 8 位 BERT。载于*2019* 年第五届能源高效机器学习与认知计算研讨会——*NeurIPS* 专场（*EMC2-NIPS*），第 36–39 页。IEEE，2019 年。

[ZHDK23] Amir Zandieh、Insu Han、Majid Daliri 和 Amin Karbasi。Kdeformer：通过核密度估计加速 Transformer。*arXiv* 预印本 *arXiv:2302.02451*，2023 年。[ZPGA23] Haoyu Zhao、Abhishek Panigrahi、Rong Ge 和 Sanjeev Arora。Transformer 在预测掩码词时会进行句法解析吗？*arXiv* 预印本 *arXiv:2303.08117*，2023 年。[ZRG+22] Susan Zhang、Stephen Roller、Naman Goyal、Mikel Artetxe、Moya Chen、Shuohui Chen、Christopher Dewan、Mona Diab、Xian Li、Xi Victoria Lin 等。OPT：开放式预训练 Transformer 语言模型。*arXiv* 预印本 *arXiv:2205.01068*，2022 年。[Zwi02] Uri Zwick。利用桥接集和矩形矩阵乘法求解所有点对最短路径。《*ACM* 学报》，49(3):289–317，2002 年。