arXiv:2303.03443v1 [cs.IT] 6 Mar 2023

改进隐马尔可夫模型的算法极化运行时间

Vincent Bian 和 Rachana Madhukara

Abstract. 我们改进了 Guruswami、Nakkiran 和 Sudan 在一篇 2018年论文中提出的隐马尔可夫源线性压缩方案的运行时间。在先前的方案中，压缩长度为 n 的消息需要 $O(n\log n)$ 的运行时间，而解压缩对任意固定的 $\delta > 0$. 需要 $O(n^{1+\delta})$ 的运行时间。我们介绍了如何通过缓存中间结果来避免重复计算，从而将解码方案的运行时间改进为 $O(n\log n)$。

1. Introduction

为随机信道设计编码方案是一个困难的问题。在 Guruswami、Nakkiran 和 Sudan 最近的一篇论文 [GNS18], 中，他们研究了隐马尔可夫信道，其中状态按照某个马尔可夫过程演化。“隐”是指无法从信道的输出中确定状态。这里需要注意的一点是，信道引入的误差仅取决于马尔可夫过程的状态。

在设计用于解决隐马尔可夫信道问题的编码和解码算法时，马尔可夫源变得十分重要，这源于一个众所周知的对应关系：可加马尔可夫链的纠错与马尔可夫源的压缩和解压缩算法相关（当压缩为线性时）。回顾一下，可加信道是指将某个字母表中的输入映射为 Σ 上的输出的信道， Σ其中在 Σ 上定义了一个阿贝尔群（这是进行加法所必需的）。信道会根据相应的马尔可夫源生成一个错误序列，该序列与输入序列相互独立。随后，信道的输出便是输入序列与错误序列按坐标相加的结果。注意，在[GNS18], 中，字母表 Σ取为基数是素数的有限域。更正式地说，

定义 1.1. 加性马尔可夫信道 $\mathcal{C}_{\mathcal{H}}$由字母表 $\mathbb{F}_q$上的马尔可夫源 $\mathcal{H}$指定，是一个随机映射 $\mathcal{C}_{\mathcal{H}} : \mathbb{F}_q^* \to \mathbb{F}_q^*$，其定义如下：对于信道输入 $X_1, \ldots, X_n$，信道输出 $Y_1, \ldots, Y_n$，其中 $Y_i = X_1 + Z_i$，其中 $Z = (Z_1, \ldots, Z_n) \sim \mathcal{H}_n$。

现在我们定义一个马尔可夫源。

定义 1.2. 字母表 Σ 上的（隐）马尔可夫源，是定义在有限状态空间上的马尔可夫链，其中每个状态 s 都关联着某个分布。更正式地说，序列 $Z_1, \ldots, Z_n$ 是马尔可夫源，当且仅当

(1) 存在一个 k状态、时间齐次的马尔可夫序列 $X_0, X_1, \ldots, X_n, \ldots$，由转移矩阵 M生成；(2) 存在一些 k分布 $P^{(1)} \ldots, P^{(k)} \in \Delta(\Sigma)$，使得 $Z_i \sim P^{(X_i)}$。

在 [GNS18], 给出了一种针对加性马尔可夫信道的编码构造。特别是，当压缩 ε 比特时，这些构造的信道容量表现接近 n，其中n 关于 $1/\varepsilon$以及马尔可夫链的混合时间是多项式的。此外，他们还表明，可以将 n 比特压缩至其熵，误差至多为 εn 的加性因子，其中 n 再次关于 $1/\varepsilon$是多项式的。尽管本文只考虑可加信道，但作者（以及我们）认为，只要仔细处理相关细节并考虑条件概率，本文的结果应当可以推广到更一般的对称信道。

本文讨论这些结果，并总结所给压缩和解压算法的主要思路。此外，我们将 Polar-Decompress 算法的运行时间改进为 $O(n\log\ n)$。

2. Preliminaries

2.1. 符号说明。我们将极化／混合矩阵记为 M（大小为 $k \times k$），并考虑将 M 应用于输入 t 次，因此极化结果为$P_m(Z) = M^{\otimes t} Z$. 我们将字母表 Σ 设为某个素数 q 对应的 $\mathbb{F}_q$，因此 $M \in \mathbb{F}_q^{k\times k}$. 此外，马尔可夫源记为 $\mathcal{H}$。

2.2. 前向算法。前向算法广泛应用于预处理、压缩和解压缩算法中。因此，本小节将概述该算法的功能及其工作原理。

给定一个具有 ℓ 个状态的马尔可夫源 $\mathcal{H}$，设前 j 个样本为 Y_1、……、$Y_j \sim \mathcal{H}_n$。对于设定 y_1、……、y_{j-1}，前向算法计算在 $(Y_1$、……、$Y_{j-1}) = (y_1$、……、$y_{j-1})$ 的条件下 Y_n 的分布。关于其实现细节，请参见 [GNS18], 中的算法 A.1, 但其主要思想是通过贝叶斯更新，依次计算 $\mathcal{H}$ 在每个样本之后的底层状态分布。每次更新耗时 $O(\ell^2)$（因为需要考虑的可能转移大致有这么多），而我们必须进行 j 次更新，因此总运行时间为 $O(j\ell^2)$。该过程通过动态规划实现。

3. Polar-Decompress Speed Up Process

以下定理与 [GNS18], 中的定理 2.9 基本相同，区别在于Polar-Decompress 被替换为 Fast-Polar-Decompress，其运行时间也从 $O(n^{3/2}\ell^2 + n\log\ n)$ 降至 $O(n\log n)$。

定理 3.1。对于每个素数 q 和混合矩阵 $M \in \mathbb{F}_q^{k\times k}$，存在一个预处理算法（Polar-Preprocess，即 *[GNS18]*中的算法 6.3）、一个压缩算法（Polar-Compress, 即 *[GNS18]*中的算法 4.1）、一个解压缩算法（Fast-Polar-Decompress，算法 *1*）以及一个多项式 $p(\cdot)$，使得对于每个 $\epsilon > 0$，以下性质成立：

(1) Polar-Preprocess 是一个随机化算法，其输入为具有 ℓ 个状态的马尔可夫源 $\mathcal{H}$ 以及 $t \in \mathbb{N}$，并在 poly$(n, \ell, 1/\varepsilon, q)$ 时间内运行，其中 $n = k^{2t}$，输出供压缩器和解压缩器使用的辅助信息（对于 $\mathcal{H}_n$）。(2) Polar-compress 的输入是一条序列 $Z \in \mathbb{F}_q^n$ 以及预处理器输出的辅助信息，在 $\mathcal{O}(n\log\ n)$ 时间内运行，并输出一个压缩字符串 $\tilde{U} \in \mathbb{F}_q^{\overline{H}(Z)+\epsilon n}$。此外，对于每个辅助输入，映射 $Z \to U$ 都是线性映射。(3) Fast-Polar-Decompress的输入为马尔可夫源 $\mathcal{H}$、压缩字符串 $\tilde{U} \in \mathbb{F}_q^{\overline{H}(Z)+\epsilon n}$以及预处理器输出的辅助信息，在 $\mathcal{O}(n \log\ n)$ 时间内运行，并输出 $Z \in \mathbb{F}_q^n$.

上述算法所提供的保证是：以至少$1 - \exp(-\Omega(n))$的概率，预处理算法会输出辅助信息 S，使得

$$\Pr_{Z\sim\mathcal{H}_n}[\textsc{Polar-Decompress}(\mathcal{H}, S; \textsc{Polar-Compress}(Z; S)) \neq Z] \leq \mathcal{O}(\frac{1}{n^2}),$$

满足 $n > p(\tau/\varepsilon)$，其中 τ 是 $\mathcal{H}$.的混合时间

完整描述请参阅附录，其中介绍了 Polar-Decompress以及前向算法。此外，请注意，Polar-decompress对 Fast-decoder 中的算法进行了黑盒使用 [BGN+22，算法 4]。

现在，我们介绍 Polar-Decompress的优化版本，该版本有效地缓存了在 ForwardInfer中计算的中间值。请注意， Polar-Decompress和 ForwardInfer 采用了不同的记法，我们将使用 Polar-Decompress中的记法。

现在，我们证明定理3.1。

证明。性质 (1) 和 (2) 的证明与 [GNS18]中的证明相同。要证明性质 (3)，只需证明 Fast-Polar-Decompress 的运行时间为 $O(n\log\ n)$。要证明最后的正确性断言，只需证明 Fast-Polar-Decompress产生的输出与 Polar-Decompress. 相同。在主 for 循环的每次迭代中，这两个算法版本都会将 $\hat{}Z^j$设为 Fast-Decoder$(\mathcal{D}_{z^j|z^{<j}}; U^j)$，因此只需证明两个算法对所有$\mathcal{D}_{z_i^j|z_i^{<j}}$，以及 i 和j.

我们对 j 使用归纳法证明以下两个陈述，其中 $j \leq (1-\varepsilon)m$:

（1）当子程序 ForwardInfer$(\mathcal{H}; 1, \hat{}Z_i^{<j})$ 在 Polar-Decompress 的第 4 行被调用时，ForwardInfer 中计算的 s_{j-1} 的值等于 Fast-Polar-Decompress 中计算的 r_i 的值, $_{j-1}$（2） Polar-Decompress 和 Fast-Polar-Decompress 都计算出 $\mathcal{D}_{z_i^j|z_i^{<j}}$ 的相同值

对于基础情形 $j = 1$， Polar-Decompress 将 $\mathcal{D}_{z_i^j|z_i^{<j}}$ 设置为 ForwardInfer$(\mathcal{H}; 1, \hat{}Z_i^{<1})$ 调用 ForwardInfer 时，它会将 s_0 设为平稳分布 π，这也是 $v_{i,0}$ 的初始值。这证明了基础情形的前半部分。

算法 1: FAST-POLAR-DECOMPRESS

常量: $M \in \mathbb{F}_q^{k\times k}$, $m = k^t$, $n = m^2$

输入: 马尔可夫源 $\mathcal{H}$，状态空间为 $[\ell]$，平稳分布为 π，以及压缩字符串 $U_{S_1}^1$、$U_{S_2}^2$、.....、$U_{S_m}^m \in \mathbb{F}_q^m$

输出: $\hat{Z} \in \mathbb{F}_q^{m\times m}$

过程 FAST-POLAR-DECOMPRESS($\mathcal{H}$; $U_{S_1}^1, U_{S_2}^2, \ldots, U_{S_m}^m$)
  对每个 $i \in [m]$ 执行
    初始化 $v_{i,\,0} \leftarrow \pi$
  对每个 $j \in [m]$ 执行
    如果 $j \le (1-\varepsilon)m$，则
      对每个 $i \in [m]$ 执行
        计算 $\mathcal{D}_{z_i^j|z_i^{<j}} := \mathbb{E}_{z\sim \Pi v_{i,\,j-1}}[\mathcal{S}_z]$
      通过填充 U^j 的未指定条目来定义 $U^j \in (\mathbb{F}_q \cup \{\perp\})^m$ 其中 $U^j_{S_j} \perp$
      Set $\hat{Z}^j \leftarrow$ FAST-DECODER($\mathcal{D}_{z^j|z^{<j}}$; U^j)
      对于每个 $i \in [m]$，执行

$$\text{定义 } v_{i,\,j} \in \Delta([\ell]),\ \text{通过 } v_{i,\,j}(z) \leftarrow \frac{(\Pi v_{i,j-1})_z \cdot \mathcal{S}_z(\hat{Z}_i^j)}{\sum_{y\in[\ell]} (\Pi v_{i,j-1})_y \cdot \mathcal{S}_y(\hat{Z}_i^j)},$$

        将 $v_{i,\,j-1}$ 视为 $\mathbb{R}^\ell$ 中的向量
    else
      令 $\hat{Z}^j \leftarrow (M^{-1})^{\otimes t} U^j_{S_j}$
  返回 $\hat{Z}$

初始化 s_0 后， FORWARDINFER 会完全跳过 for 循环，将 s_1 设为 $\Pi s_0 = \pi$，并返回 $\mathbb{E}_{i\sim s_1}[\mathcal{S}_i] = \mathbb{E}_{i\sim\pi}[\mathcal{S}_i]$.

同时，当 $j = 1$ 时， FAST-POLAR-DECOMPRESS 直接将 $\mathcal{D}_{z_i^j|z_i^{<j}}$ 设为 $\mathbb{E}_{z\sim\Pi v_{i,\,j-1}}[\mathcal{S}_z]$.。由于 $v_{i,\,j-1} = v_{i,\,0} = \pi$，且 $\Pi\pi = \pi$，我们有 $\mathcal{D}_{z_i^j|z_i^{<j}} = \mathbb{E}_{z\sim\pi}[\mathcal{S}_z]$，这与 POLAR-DECOMPRESS. 中的值相同。这就完成了基础情形。

在 POLAR-DECOMPRESS 中，$\mathcal{D}_{z_i^j|z_i^{<j}}$ 的条目已经设定便不会再改变。考虑 FORWARDINFER. 中 for 循环里的计算。s_t 的每个值都只取决于 s_{t-1}，而 y_t. 的值。当调用 FORWARDINFER 时， y 被设为 $\hat{Z}^{<j}$.。因此，当调用 FORWARDINFER($\mathcal{H}$; $j-1$, $\hat{Z}_i^{<j}$)时的输入 y_1、y_2、.....、y_{j-2}，与上一次迭代中调用 FORWARDINFER($\mathcal{H}$; $j-1$, $\hat{Z}_i^{<j}$)时相同。这意味着 s_0 到 s_{j-2} 的值也相同。根据归纳假设，这意味着$s_{j-2} = r_{i,\,j-2}$.

因此， s_{j-1} 是用

$$s_{j-1}(z) \leftarrow \frac{(\Pi s_{j-2})_z \cdot \mathcal{S}_z(y_t)}{\sum_{j\in[\ell]} (\Pi s_{j-2})_j \cdot \mathcal{S}_j(y_t)} = \frac{(\Pi s_{j-2})_z \cdot \mathcal{S}_z(\hat{Z}_i^{j-1})}{\sum_{j\in[\ell]} (\Pi s_{j-2})_j \cdot \mathcal{S}_j(\hat{Z}_i^{j-1})},$$

其中我们代入了输入值 y_t. 这与$v_{i,\,j-1}(z)$ 在 FAST-POLAR-DECOMPRESS 第 11 行中的计算表达式完全相同，

只是其中一些索引的命名有所不同。这证明了归纳假设中的第一个陈述。

为证明归纳假设的第二个陈述，注意到 FORWARDINFER($\mathcal{H}; j-1, \hat{}Z_i^{<j}$) 输出 $\mathbb{E}_{z\sim s_j}[\mathcal{S}_z]$，且 $s_j := \Pi s_{j-1} = \Pi v_{i,\ j-1}$. 然而，$\mathbb{E}_{z\sim \Pi v_{i,\ j-1}}[\mathcal{S}_z]$ 恰好是 FAST-POLAR-DECOMPRESS 在第 7 行赋给 $\mathcal{D}_{z_i^j|z_i^{<j}}$ 的值。至此，归纳步骤完成。因此，FAST-POLAR-DECOMPRESS 和 POLAR-DECOMPRESS 产生相同的输出。■

最后，我们证明 FAST-POLAR-DECOMPRESS 的运行时间确实快于 POLAR-DECOMPRESS。

引理 3.2。FAST-POLAR-DECOMPRESS 的运行时间为 $O(n\log\ n)$。

证明。第 8 行耗时 $O(m)$，第 9 行耗时 $O(m\log\ m)$ [BGN+22]。第 7 行和第 11 行的运行时间都与 ℓ^2 成正比，而相对于 m. 而言，ℓ^2 是常数。因此，内部的 for 循环每次运行分别贡献 $O(m)$。由于第 6 行至第 11 行运行了 $(1-\varepsilon)m$ 次，总运行时间为 $O(m^2\log\ m) = O(n\log\ n)$，如需证明。■

4. ACKNOWLEDGEMENTS

本研究始于哈佛大学一门关于计算机科学中的信息论的课程（CS229r, 2022年秋季）的期末项目。我们感谢 Madhu Sudan 教授对本项目的指导。我们也感谢 Chi-Ning Chou 和 R. Emin Berker 提供的宝贵建议。

REFERENCES

[BGN+22] Jaros law B lasiok、Venkatesan Guruswami、Preetum Nakkiran、Atri Rudra 和 Madhu Sudan。一般强极化。ACM《计算机协会学报》（JACM），69(2):1–67，2022年。[GNS18] Venkatesan Guruswami、Preetum Nakkiran 和 Madhu Sudan。隐马尔可夫模型的算法极化。arXiv 预印本 arXiv:1810.01969，2018年。[GX14] Venkatesan Guruswami 和 Patrick Xia。极化码：极化速度与距信道容量的多项式间隙。IEEE《信息论汇刊》，61(1):3–16，2014年。

APPENDIXA. POLAR DECOMPRESS

算法 2： POLAR-DECOMPRESS（算法 4.2，来自 [GNS18]）

常数：$M \in \mathbb{F}^{k\times k}_q$，$m = k^t$，$n = m^2$

输入：马尔可夫源 $\mathcal{H}$ 和 $U^1_{S_1}$、$U^2_{S_2}$、……、$U^m_{S_m} \in \mathbb{F}^m_q$

输出：$\hat{Z} \in \mathbb{F}^{m\times m}_q$

过程 POLAR-DECOMPRESS($\mathcal{H}$; $U^1_{S_1}, U^2_{S_2}, \ldots, U^m_{S_m}$)

  对于每个 $j \in [m]$，执行

    如果 $j \le (1-\varepsilon)m$，则

      计算 $\mathcal{D}_{z^j|z^{<j}}$ 的条件分布 $\overline{Z}^j$ 给定 $\overline{Z}^{<j} = \hat{Z}^{<j}$，使用前向算法计算马尔可夫源 $\mathcal{H}$

      通过填写 U^j 中的未指定条目来定义 $U^j \in (\mathbb{F}_q \cup \{\perp\})^m{}_{S_j}$ 其中 $\perp$

      Set $\hat{Z}^j \leftarrow$ FAST-DECODER($\mathcal{D}_{z^j|z^{<j}}$; U^j)

    else

      设 $\hat{Z}^j \leftarrow (M^{-1})^{\otimes t} U^j{}_{S_j}$

  返回 $\hat{Z}$

APPENDIXB. FORWARD ALGORITHM

算法 3： 前向算法（来自 [GNS18]的算法 A.1）

输入：$n \in \mathbb{N}$. 马尔可夫源 $\mathcal{H}$，其状态空间为 $[\ell]$，字母表为 Σ，平稳分布 $\pi \in \Delta([\ell])$、转移矩阵 $\Pi \in \mathbb{R}^{\ell\times\ell}$以及输出分布 $\{\mathcal{S}_i \in \Delta(\Sigma)\}_{i\in[\ell]}$. 以及 $y = (y_1, y_2, \ldots, y_{n-1})$，其中 $y_i \in \Sigma$.

输出：分布 $Y_n \in \Delta(\Sigma)$

过程 FORWARDINFER($\mathcal{H} = (\ell, \Sigma, \pi, \Pi, \{\mathcal{S}_i\})$; n, y)

  $s_0 \leftarrow \pi$

  对于每个 $t = 1, 2, \ldots, n-1$ 执行

    定义 $s_t \in \Delta([\ell])$ 为 $s_t(i) \leftarrow \frac{(\Pi s_{t-1})_i \cdot \mathcal{S}_i(y_t)}{\sum_{j\in[\ell]} (\Pi s_{t-1})_j \cdot \mathcal{S}_j(y_t)}$，将 s_{t-1} 视为一个向量，其中 $\mathbb{R}^\ell$.

  $s_n \leftarrow \Pi s_{n-1}$

  返回分布 $Y_n := \mathbb{E}_{i\sim s_n}[\mathcal{S}_i]$

V. BIAN: DEPARTMENT OF PHYSICS, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAM-BRIDGE, MA 02139电子邮箱：vinvinb@mit.edu

R. MADHUKARA: DEPARTMENT OF MATHEMATICS, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MA 02139电子邮箱：rachanam@mit.edu