arXiv:2301.09203v1 [cs.DS] 22 Jan 2023

对抗性流处理的宽松模型：建议模型与有界中断模型*

Moshe Shechner† Menachem Sadigurschi‡ Uri Stemmer§

2023年1月22日

摘要

流处理算法通常在无知设置下进行分析，即假设输入流预先固定。近年来，人们对设计对抗性鲁棒流处理算法的兴趣日益浓厚；这类算法必须在执行过程中，即使输入流被自适应地、以对抗方式选定，也能维持效用。尽管针对对抗性设置已有若干引人入胜的研究成果，但通常需要付出极高的空间代价。受此启发，本文旨在探索一些中间模型，使我们能够在无知模型与对抗模型之间进行插值。具体而言，我们提出以下两种模型：

- 建议模型允许流式算法偶尔请求一位建议信息。
- 在有界中断模型中，我们假设对手仅具有部分自适应性。

我们分别针对这两种模型给出了正面和负面的结果。特别是，我们给出了从这两种模型分别归约到非自适应模型的通用归约。借此，我们能够设计出鲁棒算法，其空间复杂度显著优于纯对抗模型中已知算法的空间复杂度。

1 引言

流处理算法用于处理数据流，输入以一系列数据项的形式呈现。一般而言，这些算法只能访问有限内存，其容量远小于存储整个数据流所需的内存。Alon、Matias 和 Szegedy [3], 对这一领域进行了形式化，此后该领域产生了大量研究成果，并与计算机科学中的许多其他领域相交。

在本文中，我们关注旨在跟踪输入流的某个函数，并持续报告该函数估计值的流处理算法。形式化地说，

*本项目部分由以色列科学基金会（资助项目 1871/19）以及 Len Blavatnik 和 Blavatnik 家族基金会资助。

†特拉维夫大学。moshe.shechner@gmail.com
‡本-古里安大学。menisadi@gmail.com§特拉维夫大学和谷歌研究院。 u@uri.co.il

定义 1.1（定义 2.1 的非正式版本）。设 X 是一个有限域，并设 $g : X^* \to \mathbb{R}$ 是一个函数，它将每个输入 $\vec{x} \in X^*$ 映射为一个实数 $g(\vec{x}) \in \mathbb{R}$。

设 $\mathcal{A}$ 是一种算法，在每一轮中，$i \in [m]$ 获取一个元素 $x_i \in X$ 并输出一个响应 $z_i \in \mathbb{R}$。如果对于每个输入序列 $\vec{x} = (x_1、x_2、...、x_m) \in X^m$，以下条件均成立，则称算法 $\mathcal{A}$ 是一个针对 g 、准确度为 α、失败概率为 β 且流长度为 m 的非自适应流处理算法。考虑 $\mathcal{A}$ 在输入流 $\vec{x}$ 上的执行。那么，

$$\Pr\left[\forall i \in [m] \ we\ have\ z_i \in (1 \pm \alpha) \cdot g(x_1, \ldots, x_i)\right] \geq 1 - \beta,$$

其中，概率取自算法 $\mathcal{A}$ 使用的随机币。

请注意，在定义 1.1 中，流式算法需要针对每个固定的输入流成功（以高概率）。特别地，这意味着假设数据流中元素的选择与流式算法的内部随机性相互独立。这一假设称为无知设置，对于大多数经典流式算法的正确性至关重要。在本文中，我们关注的是这一假设不成立的情形，称为对抗性设置。

1.1 （朴素）对抗模型

各种形式的对抗式流模型由 [27, 17, 18, 1, 2, 20, 9,8, 21, 30, 7, 4] 进行研究。对抗性设置被建模为（随机化） `StreamingAlgorithm` 与 `Adversary` 之间的双人博弈。开始时，我们固定一个函数 $g : X^* \to \mathbb{R}$。在整个博弈过程中，对手选择数据流中的更新，并且可以在其选择的 T 个时间步查询流式算法（称为“查询时刻”）。形式化地，

1. 对于第 $i = 1$、2、.....、m 轮

   (a) `Adversary` 选择一个更新 $x_i \in X$ 和一个查询请求 $q_i \in \{0, 1\}$，但须满足限制 $\sum_{j=1}^{i} q_j \leq T$。(b) `StreamingAlgorithm` 处理新更新 x_i。如果 $q_i = 1$ ，那么 `StreamingAlgorithm` 输出响应 z_i，并将其提供给 `Adversary`。

`Adversary` 的目标是在数据流中的某个查询时间 i ，使 `StreamingAlgorithm`输出错误的响应 z_i。设 g 为定义某个流式问题的函数，并假设存在一个用于 g 的无知流式算法 $\mathcal{A}$ ，其使用空间 s。显然，通过运行 $\mathcal{A}$ 的 T 个副本，并使每个副本至多用于一次查询，可以使用空间 $\approx s \cdot T$在对抗性设置下解决 g。问题是我们能否做得更好。事实上，Hassidim 等人 [21]证明了以下结果。

定理 1.2（[21],非正式表述）。如果存在一个用于函数 g 的无知流式算法，其使用空间 s，那么就存在一个用于 g 、支持 T 次查询的对抗鲁棒流式算法，其使用空间为 $\approx \sqrt{T} \cdot s$。

请注意，当查询数量 T 较大时，这种构造会导致空间大幅膨胀。应对这一问题的一种方法，是对函数 g 或输入流施加额外限制。事实上，自 Ben-Eliezer 等人 [8], 以来，大多数关于对抗性流处理的正面结果都假设输入流的翻转次数较小，其定义如下。

定义 1.3（翻转次数 [8]）。输入流 $\vec{x}$ 相对于函数g的 (α, m)-翻转次数，记作 $\lambda_{\alpha,\ m}(\vec{x}, g)$，或简记为 λ，是指在数据流 $\vec{x}$过程中， g 的值以 $(1+\alpha)$ 倍发生变化（增加或减少）的最大次数。

从 [8], [8, 21, 30, 4] 的早期研究开始，学者们提出了通用构造，将空间复杂度为 s 的非自适应流式算法转化为空间复杂度为 $\approx s \cdot \text{poly}(\lambda)$的对抗鲁棒流式算法。也就是说，在翻转次数有界的假设下，这些早期研究甚至能够支持 $T = m$ 次查询。这很有用，因为对于仅插入模型中的许多有趣流式问题（数据流中没有删除操作），已知参数 λ 很小。然而，一般情况下，该参数可能大到 $\Theta(m)$，此时[8, 21, 30, 4] 的转换在空间开销方面代价极高。

总而言之，当查询次数 T 较少，或假设翻转次数较小时，目前从非自适应模型到对抗性设置的转换才有用。

1.2 我们的结果

对抗模型的一种批评是，它（或许）过于悲观。确实，许多场景可能不属于非自适应模型，但又与“对抗性”设置相去甚远。受此启发，本研究旨在探索一些中间模型，使我们能够在非自适应模型与对抗模型之间进行插值。具体而言，我们研究了两种这样的模型，分别称为建议模型和有界中断模型。

1.2.1 带建议的对抗性流处理 (ASA)

我们提出一种模型，其中流式算法在整个执行过程中可以偶尔请求一位建议信息。令 $\eta \in \mathbb{N}$ 为控制查询/建议比率的参数。我们考虑以下在 `StreamingAlgorithm` 与 `Adversary` 之间进行的游戏，称为 ASA 游戏。

对于轮次 $i = 1,2,\ldots,m$：

1. `Adversary` 选择一次更新 $x_i \in X$ 和一个查询请求 $q_i \in \{0,1\}$，并受限于 $\sum_{j=1}^{i} q_j \leq T$。

2. `StreamingAlgorithm` 处理新的更新 x_i。

3. 如果 $q_i = 1$，则

(a) `StreamingAlgorithm`输出响应 z_i，并将其提供给 `Adversary`。(b) 如果 $\left(\sum_{j=1}^{i} q_j\right) \bmod \eta = 0$，那么 `StreamingAlgorithm` 指定一个谓词 $p_i: X^* \to \{0,1\}$，并获取 $p_i(x_1, x_2, \ldots, x_i)$。

也就是说，在 ASA 模型中，对手总共可以进行 T 次查询，而每进行 η 次查询，流式算法就可以获得一位建议信息；该信息是根据数据流中截至目前的元素计算得出的一个谓词。

研究这一模型的主要动机在于理论层面：它为我们提供了一种直观的方式，用来衡量流式算法要在对抗性设置中维持效用所需的额外信息量。从实践角度来看，在以下情境中，这一模型也可能颇具意义。考虑一种流式场景：输入流同时输入到一个（低空间开销的）流式算法 $\mathcal{A}$ 和一台服务器 $\mathcal{S}$。服务器拥有大量空间，可以存储整个输入流（因此原则上也能自行解决流式问题）。然而，假设服务器存在通信瓶颈，同时还忙于并行处理许多其他任务。因此，我们希望尽可能多地将通信工作委托给“廉价”的（低空间开销）流式算法 $\mathcal{A}$。ASA 模型允许进行这样的委托：流式算法自行处理大部分查询，每隔 η 次查询才向服务器请求一位建议信息。

我们证明以下一般性结果。

定理 1.4（定理 3.5 的非正式表述）。如果存在一个空间复杂度为 s、用于计算函数 $g\colon X^* \to \mathbb{R}$ 的无偏线性流式算法，那么对于每个 $\eta \in \mathbb{N}$，都存在一个针对 g 的 ASA 模型对抗鲁棒流式算法，其查询/建议比率为 η，空间复杂度为 $\approx \eta \cdot s^2$。

为得到这一结果，我们采用了 Hassidim 等人提出的一种技术 [21]，该技术利用差分隐私 [15] 保护的不是输入数据，而是流式算法的内部随机性。直观地说，这让我们能够确保算法的“鲁棒性”随着建议比率的增加而衰减得更慢，从而实现建议与鲁棒性权衡。

请注意，定理 1.4 中算法的空间复杂度不会随查询数量 T 呈多项式增长。例如，以下是在 F_2 估计问题中的定理直接应用（即估计输入流的频率向量的二阶矩）。

定理 1.5（ASA 模型中的 F_2 估计，非正式表述）。设 $\eta \in \mathbb{N}$。存在一种在 ASA 模型中具有对抗鲁棒性的 F_2 估计算法，其查询/建议比率为 η，能够保证达到α准确度（以高概率），所用空间为 $\tilde{O}\left(\eta/\alpha^4\right)$。

注 1.6。我们强调，ASA 模型在形式上确实介于无偏模型与（纯）对抗模型之间。显然，ASA 模型比纯对抗模型更简单，因为我们可以直接忽略建议比特。另一方面，简单的论证表明，ASA 模型（对于任意 $\eta > 1$）在定性上比无知设置更难。为此，设 $\mathcal{A}$ 是 ASA 模型中针对函数 g、查询/建议比率为 $\eta > 1$的算法。那么， $\mathcal{A}$可以转换为以下针对 g 的无偏算法 $\mathcal{A}_{\text{oblivious}}$（该算法在每个时间步都会返回一个估计值，且不接收任何建议）：

1. 实例化 $\mathcal{A}$。
2. 在每个时刻 $i \in [m]$:

   (a) 获取一次更新 $x_i \in X$。(b) 将 $\mathcal{A}$（包括其内部状态）复制为影子副本 $\mathcal{A}^{\text{shadow}}$。(c) 将更新 $(x_i, 0)$ 输入 $\mathcal{A}$，将更新 $(x_i, 1)$ 输入 $\mathcal{A}^{\text{shadow}}$，并从影子副本获取答案 z_i。注意，我们只查询影子副本。(d) 输出 z_i，并从内存中删除影子副本。

由于我们在每次查询后都会“回退” $\mathcal{A}$ ，因此它不会发出建议请求， $\mathcal{A}_{\text{oblivious}}$也就不会发出建议请求。也就是说， $\mathcal{A}_{\text{oblivious}}$ 是无视模型算法。此外，简单的论证表明，该算法在无知设置中仍能保持效用。[1]

注释 1.7。我们的构造有一个优点：整个交互过程中指定的谓词都很“简单”，也就是说，每个谓词都可以用流式方式计算。具体而言，给定谓词 p_i，可以在仅遍历一次 $x_1, x_2, \ldots, x_i$ 的情况下，使用小空间计算比特 $p_i(x_1, x_2, \ldots, x_i)$。

ASA 模型的一个否定结果。定理 1.4 针对由实值函数定义的流式问题，在 ASA 模型中给出了一个强有力的肯定结果。我们通过给出一个不由实值函数定义的简单流式问题的否定结果，补充这一成果。具体而言，我们考虑一个广为研究的 ℓ_0-采样问题（的变体）：流式算法必须从未删除元素集合中返回一个均匀随机元素。已知，在无知设置下， ℓ_0-采样问题很容易解决（例如参见 [22]），而在纯对抗设置下则很难（例如参见 [1]）。通过一个简单的计数论证，我们证明，即使查询/建议比率为 1， ℓ_0-采样问题在 ASA 模型中仍然很难；也就是说，即使流式算法每次查询都能获得一个建议比特，问题依然很难。

1.2.2 有界中断下的对抗性流处理 (ASBI)

回想一下，在纯对抗模型中，对手具有完全自适应性，也就是说，第 i 次更新可以根据截至此时对手掌握的所有信息来选择。我们考虑对这一设定的一种细化，其中对手仅具有部分自适应性。游戏开始时，对手指定一个完整的输入流。在整个执行过程中，对手（它能看到流式算法给出的所有输出）可以自适应地决定中断数据流，并替换其尚未由流式算法处理的后缀。为简单起见，这里假设流式算法在每个时间步都会接受查询（即 $T = m$）。

形式化地，令 $R \in \mathbb{N}$ 为限制中断次数的参数。我们考虑以下博弈，称为 ASBI 博弈，由 `StreamingAlgorithm` 和 `Adversary` 参与。

[1]为说明这一点，固定一个输入流 $\vec{x} = (x_1, x_2, \ldots, x_m)$，并固定 $j \in [m]$。注意，在运行于 $\vec{x}$ 时， $\mathcal{A}_{\text{oblivious}}$ 于时间 j 给出的输出分布，与 $\mathcal{A}$ 运行于数据流 $((x_1, 0) \ldots, (x_{j-1}, 0), (x_j, 1))$ 时的结果分布相同；根据 $\mathcal{A}$ 的效用保证，该输出以高概率准确（因为这个替代数据流中只有 1 次查询，因此 $\mathcal{A}$ 在其上运行时得不到任何建议）。现在对查询时刻应用并集界，即可得出该结论。

1. Adversary 选择一个数据流 $\vec{x} = (x_1, x_2, \ldots, x_m) \in X^m$。

2. 对于轮次 $i = 1,2, \ldots, m$

(a) StreamingAlgorithm获取更新 x_i，并输出响应 z_i。(b) Adversary 获取 z_i，并输出中断请求 $d_i \in \{0,1\}$，但须满足限制 $\sum_{j=1}^{i} d_j \leq R$。(c) 如果 $d_i = 1$，那么对手会输出一个新的流后缀 $(x'_{i+1}, \ldots, x'_m)$，并用 $(\leftarrow, \ldots, x'_m)$ 覆盖 $(x_{i+1}, \ldots, x_m)$。

也就是说，对手可以看到流式算法给出的所有输出，并自适应地决定在 R 个位置任意修改数据流的其余部分。重要的是，流式算法“并不知道”中断何时发生。该模型在无知设置（其中 $R = 0$）与完全对抗性设置之间提供了一种非常直观的插值（后者可通过设置 $R = m$ 获得；或者，更巧妙地，在查询数至多为 $T \leq m$ 时设置 $R = T$ 获得）。我们证明了以下一般性结果。

定理 1.8（定理 4.1 的非正式版本）。如果存在一个针对函数 $g\colon\ X^* \to \mathbb{R}$ 的无知流式算法，使用空间 s，那么对于每个 $R \in \mathbb{N}$，都存在一个针对 g 的对抗鲁棒流式算法，在 ASBI 模型中能够抵御 R 次中断，使用空间 $\approx R \cdot s$。

为得到这一结果，我们借助 Ben-Eliezer 等人 [8] 提出的草图切换技术。直观地说，我们维护无知流式算法 $\mathcal{A}$ 的 $2R$ 个副本，在任意时刻，恰好将其中两个指定为“活动”副本。只要这两个活动副本给出的估计值大致相同，它们就继续作为“活动”副本，而我们使用它们的估计值作为响应。一旦二者出现分歧，我们就将它们一并丢弃（不再使用），并指定两个（全新的）副本作为“活动”副本。我们证明，可以将这一构造形式化，从而得到定理 1.8。

请注意，定理 1.8 中算法的空间复杂度不会随时间步数 m 呈多项式增长。例如，以下是定理 1.8 在估计 F_2 时的直接应用。

定理 1.9（ASBI 模型中的 F_2 估计，非正式表述）。设 $R \in \mathbb{N}$。存在一种具备对抗鲁棒性的 F_2 估计算法，在 ASBI 模型中保证准确度为 α（以高概率成立），同时能够抵御 R 次中断，且使用空间 $\tilde{O}\left(R/\alpha^2\right)$。

ASBI 模型的一个否定性结果。请注意，我们在定理 1.8 中的构造，其空间膨胀随中断次数 R线性增长。回想一下，在完全对抗模型中（其中 $R = T$ 针对 T 个查询），已知空间膨胀达到 $\sqrt{T}$ 即足够（见定理 1.2）。因此，人们可能会猜测，在 ASBI 模型中， R 的正确依赖关系应为 $\sqrt{R}$。然而，我们证明一般情况下并非如此。具体而言，我们证明存在一个流式问题，在无知设置下可以用小空间轻松解决，但在 ASBI 模型中则需要线性于 R 的空间，前提是查询数量足够大（相对于 R为多项式级别）。

1.3 其他相关工作

对抗性流处理模型（在与我们类似的设置中）至少可追溯到 [1], , 他们对此进行了隐式研究，并证明了在次线性内存下进行鲁棒 ℓ_0 采样是不可能的。随后， [20], 对对抗性流处理模型进行了明确的形式化，并证明了线性草图方面的强不可能性结果。以 [8] 为开端、由 [21, 30, 4, 7]继续推进的一系列近期工作，在假设数据流的翻转次数有界的情况下，为许多重要问题取得了积极成果（即鲁棒算法）。另一方面， [8]还针对 AMS 草图的一个变体提出了一种攻击，该攻击包含 O（n）个自适应轮次，其中 n 是定义域的大小。之后， [25] 构造了一个流式问题，对于该问题，任何对抗鲁棒流算法都必须使用多项式空间，由此证明了非自适应模型与（普通）对抗模型之间存在分离。最近， [12] 针对一种具体算法—— `CountSketch`——提出了一种攻击；该攻击的长度与算法的空间呈线性关系，且仅使用两轮自适应性。

2 预备知识

本文考虑由实值函数定义的流式问题（此时目标是近似该函数的值），以及定义有效解集合的流式问题（此时目标是返回其中一个有效解）。以下定义统一了无知设置下的这两个目标。

定义 2.1（非自适应流处理）。设 X 是一个有限域，并设 $g\colon X^* \to 2^W$ 是一个函数，它将每个输入 $\vec{x} \in X^*$ 映射到一个有效解的子集 $g(\vec{x}) \subseteq W$（取自某个范围 W）。

设 $\mathcal{A}$ 是一个算法，它在 m 轮次中获取一个元素 $x_i \in X$ 并输出响应$z_i \in W$。若对于每个输入序列 $\vec{x} = (x_1, x_2, \ldots, x_m) \in X^m$，以下条件均成立，则称算法 $\mathcal{A}$ 是针对 g 的无知流式算法，其失败概率为β，流长度为 m。考虑 $\mathcal{A}$ 在输入流 $\vec{x}$ 上的一次执行。则，

$$\Pr\left[\forall i \in [m] \text{ we have } z_i \in g(x_1, \ldots, x_i)\right] \geq 1 - \beta,$$

其中的概率是针对算法 $\mathcal{A}$ 的随机币而言的。

例如，在估计数据流中不同元素数量的问题中，上述定义中的函数 g 返回区间 $g(x_1, \ldots, x_i) = (1 \pm \alpha) \cdot |\{x_1, \ldots, x_i\}|$，其中 α 是所需的近似参数。

2.1 差分隐私预备知识

差分隐私 [15] 是一种隐私的数学定义，旨在支持对数据库进行统计分析，同时提供强有力的保证，确保个体级信息不会泄露。考虑一个算法 $\mathcal{A}$ ，它作用于一个数据库，其中每一行代表一个个体的数据。如果算法 $\mathcal{A}$ 的结果分布对任意单个个体数据的任意变化都不敏感，则称该算法满足差分隐私。直观地说，这意味着算法 $\mathcal{A}$ 几乎不会泄露任何单个个体的数据，因为即使没有该个体的数据，其输出的分布也大致相同。形式化地说，

定义 2.2（[15]）。设 $\mathcal{A}$ 是一个作用于数据库的随机化算法。如果对于任意两个仅有一行不同的数据库 S、S' 以及任意事件T，都有

$$\Pr[\mathcal{A}(S) \in T] \leq e^{\varepsilon} \cdot \Pr\left[\mathcal{A}(S') \in T\right] + \delta.$$

2.1.1 对数据的中位数进行隐私保护近似

给定一个数据库 $S \in X^*$，考虑隐私保护地识别 S 的近似中位数这一任务。具体而言，对于误差参数 Γ，我们希望识别一个元素 $x \in X$，使得 S 中至少有 $|S|/2 - \Gamma$ 个元素大于或等于 x，并且 S 中至少有 $|S|/2 - \Gamma$ 个元素小于或等于 x。目标是使 Γ 尽可能小，并将其表示为隐私参数 ε、δ、数据库大小 $|S|$ 和定义域大小 $|X|$ 的函数。

文献中有若干高级构造，其误差随定义域大小增长得非常缓慢（仅随 $\log^* |X|$ 多项式增长）[6, 11, 10, 24, 13]。不过，在我们的应用中，更简单的构造就足够了（其误差随定义域大小呈对数增长）。以下定理可直接由指数机制[26] 推出。

定理 2.3. 存在一个 $(\varepsilon, 0)$-差分隐私算法，对于给定的数据库 $S \in X^*$，输出一个元素 $x \in X$，使得以至少 $1 - \beta$ 的概率，S 中至少有 $|S|/2 - \Gamma$ 个元素大于或等于 x，且 S 中至少有 $|S|/2 - \Gamma$ 个元素小于或等于 x，其中 $\Gamma = O\left(\frac{1}{\varepsilon}\log\left(\frac{|X|}{\beta}\right)\right)$。

2.1.2 差分隐私的组合定理

以下定理允许我们论证某个算法在通过若干差分隐私机制访问其输入数据库时所能提供的隐私保证。

定理 2.4 ([16])。令 $0 < \varepsilon, \delta' \leq 1$，且令 $\delta \in [0, 1]$。允许与保持 (ε, δ)-差分隐私的机制进行 k 次自适应交互（且不以其他方式访问数据库）的机制，可确保 $(\varepsilon', k\delta + \delta')$-差分隐私，其中 $\varepsilon' = \sqrt{2k\ln(1/\delta')} \cdot \varepsilon + 2k\varepsilon^2$。

2.1.3 差分隐私的泛化性质

Dwork 等人 [14] 和 Bassily 等人 [5] 证明，如果谓词 h 是对随机样本进行差分隐私计算的结果，那么 h 的经验均值与其在底层分布上的期望必定接近。形式上，

定理 2.5（[14, 5, 29]）。设 $\varepsilon \in (0, 1/3)$、$\delta \in (0, \varepsilon/4)$ 且 $n \geq \frac{1}{\varepsilon^2}\log(\frac{2\varepsilon}{\delta})$。设 $\mathcal{A}\colon X^n \to 2^X$ 是一个 (ε, δ)-差分隐私算法，它对大小为 n 的数据库进行操作，并输出一个谓词 $h\colon X \to \{0, 1\}$。设 $\mathcal{D}$ 是 X 上的一个分布，S 是一个包含来自 $\mathcal{D}$ 的 n 个独立同分布元素的数据库，且 $h \leftarrow \mathcal{A}(S)$。那么

$$\Pr_{\substack{S\sim\mathcal{D}^n \\ h\leftarrow\mathcal{A}(S)}}\left[\left|\frac{1}{|S|}\sum_{x\in S} h(x) - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[h(x)]\right| \geq 10\varepsilon\right] < \frac{\delta}{\varepsilon}.$$

3 带建议的对抗流处理（ASA）

本节介绍我们在 ASA 模型中的结果，该模型在第 1.2.1 节中定义。我们从通用变换开始。

3.1 ASA 模型的一种通用构造

我们针对 ASA 模型的通用构造，将一个无视性线性流处理算法$\mathcal{A}$ 转换为 ASA 模型中的鲁棒流处理算法。我们所需的线性性质如下。假设以相同的内部随机性 r 实例化算法 $\mathcal{A}$ 的三个副本，分别记为 $\mathcal{A}_1$、$\mathcal{A}_2$、$\mathcal{A}_3$；并假设 $\mathcal{A}_1$ 处理数据流 $\vec{x}_1$，$\mathcal{A}_2$ 处理数据流$\vec{x}_2$，而 $\mathcal{A}_3$ 处理数据流 $\vec{x}_1 \circ \vec{x}_2$。那么，存在一种运算（记为“+”），使我们能够获得一个内部状态（$\mathcal{A}_1 + \mathcal{A}_2$），该状态与 $\mathcal{A}_3$ 的内部状态完全相同。许多经典流式算法都具有这一性质（例如，经典的 F_2 AMS 草图就具有这一性质 [3]）。形式化地说，

定义 3.1（线性状态算法）。设 $\mathcal{A}$ 是一个算法，其内存状态分为三个部分。第一部分在算法开始时随机生成，并在整个执行过程中保持不变，记为 $\mathcal{S}_R$。第二部分是 $\mathbb{R}^d$ 中向量的编码，记为 $\mathcal{S}_v$。第三部分是其余的内存空间，记为 $\mathcal{S}_c$，用于其他计算。那么，如果对于任意两个数据流$\vec{u}_1 = ((x_1, q_1) \ldots, (x_l, q_l)) \in (X \times \{0,1\})^l$、$\vec{u}_2 = ((x_1, q_1) \ldots, (x_p, q_p)) \in (X \times \{0,1\})^p$，其长度为 l、$p \in \mathbb{N}$，且对 $\mathcal{A}$ 使用相同随机状态（$\mathcal{S}_R$）进行三次不同的执行时，下述条件成立，则称 $\mathcal{A}$ 关于其输入流具有线性状态：

$$\mathcal{S}_v(\vec{u}_1 \circ \vec{u}_2) = \mathcal{S}_v(\vec{u}_1) + \mathcal{S}_v(\vec{u}_2)$$

其中，$\mathcal{S}_v(\vec{u})$ 是由输入流 $\vec{u}$ 在相应内存状态中编码后得到的编码向量 $v \in \mathbb{R}^d$。

考虑算法 `RobustAdvice`。在外层循环的每次迭代开始时，算法 `RobustAdvice` 会采样 k 个新的随机字符串，并用它们为该次外层循环迭代实例化相应的下一个副本和影子副本算法 $\mathcal{A}$。记此类外层循环的数量为 τ（$\tau \le m$）。我们不妨设想，算法 `RobustAdvice` 不采样这些随机字符串，而是按以下格式将它们作为输入获取：

$$\vec{r} = \left(\vec{r^1}, \vec{r^2}, \ldots, \vec{r^\tau}\right) = \left((r_1^1, \ldots, r_k^1), (r_1^2, \ldots, r_k^2), \ldots, (r_1^\tau, \ldots, r_k^\tau)\right),$$

其中，对于某个 $j \in [\tau]$，$\vec{r^j} = (r_1^j, \ldots, r_k^j)$ 用作外层循环迭代中在步骤 $2a$ 初始化的 k 个实例及其在步骤 2b 的 k 个重复项所使用的随机字符串。记每次外层循环开始的时刻（步骤 $2a$）为 $(t_1 < \cdots < t_\tau)$，并记截至时刻 t 的输入流为 $\vec{x}_t$。那么，下述引理指出，在外层循环迭代开始时（例如，对于某个 $j \in [\tau]$，时刻为 t_j），$i \in [k]$ 的 $\mathcal{A}_i^{\text{active}}$ 实例由时刻 t_j 的输入流所对应的状态组成，$\vec{x}_{t_j}$，其中随机字符串为 $r_i^{t_{j-1}}$。也就是说，步骤 2e 已成功恢复算法 $\mathcal{A}^{\text{active}}$ 的实例 i 关于 $r_i^{t_{j-1}}$ 对输入 $\vec{x}_{t_j}$ 的状态。

引理 3.2（状态恢复）。记 τ为外层循环的次数，$(t_1 < \cdots < t_\tau) \in [m]^\tau$为每次外层循环（步骤 2a）开始的时刻，并以 $\vec{x}_t$ 表示截至时刻 t的输入流。那么，对于所有 $j \in \{1, \ldots, \tau - 1\}$，对于所有 $i \in [k]$，算法实例 $\mathcal{A}_i^{\text{active}}$由状态段 $\mathcal{S}_v(\vec{x}_{t_j})$ 组成，该状态段对应于状态段 $\mathcal{S}_R(r_i^{j-1})$。

算法 1 RobustAdvice(β, m, η)

输入：参数： β 为失败概率， m 为输入流的长度， η 为建议查询周期。

所用算法：一个空间为 s 、准确度为 α 的无偏线性流式算法 $\mathcal{A}$ 。常数计算：

1. $k = \Omega(\eta s \log(m/\beta) \log^2(m/(\beta\alpha)))$ 是集合“活动”、“下一个”和“影子”中每个集合的实例数量。

2. $\varepsilon_0 = \frac{\varepsilon}{\sqrt{8\eta k s \ln(1/\delta)}}$ 是 PrivateMed 次执行的隐私参数，其中 $\varepsilon = 1/100, \delta = O(\beta/m)$。

---

1. 初始化算法 $\mathcal{A}$的 k 个独立实例 $\mathcal{A}_1^{\text{active}}, \ldots, \mathcal{A}_k^{\text{active}}$。
2. 重复（外层循环）

(a) 初始化 k 个算法 $\mathcal{A}$ 的独立实例 $\mathcal{A}_1^{\text{next}}, \ldots, \mathcal{A}_k^{\text{next}}$。 (b) 令 $\mathcal{A}_1^{\text{shadow}}, \ldots, \mathcal{A}_k^{\text{shadow}}$ 为 $\mathcal{A}_1^{\text{next}}, \ldots, \mathcal{A}_k^{\text{next}}$ 的重复副本，其中每个 $\mathcal{A}_j^{\text{shadow}}$ 都以与 $\mathcal{A}_j^{\text{next}}$ 相同的随机性启动。 (c) 将当前时间步记为 t。（也就是说，到目前为止，我们已在数据流中看到 t 次更新。）(d) 重复（内层循环）i. 接收下一个更新 x_i 和一个查询请求 $q_i \in \{0, 1\}$。ii. 将更新 x_i 插入 $\mathcal{A}_1^{\text{active}}, \mathcal{A}_1^{\text{next}}, \ldots, \mathcal{A}_k^{\text{active}}, \mathcal{A}_k^{\text{next}}$ 中的每一个。iii. 如果 $q_i = 1$ ，则：

- 查询 $\mathcal{A}_{\text{活动}1}, \mathcal{A}_{\text{活动}2}, \ldots, \mathcal{A}_{\text{活动}k}$ ，并获得答案 $y_{i,1}, y_{i,2}, \ldots, y_{i,k}$
- 输出 $z_i \leftarrow \texttt{PrivateMed}(y_{i,1}, y_{i,2}, \ldots, y_{i,k})$，隐私参数为 ε_0。
- 如果 $\left(\sum_{j=1}^{i} q_j\right) \bmod \eta = 0$ ，则定义谓词 p_i ，它接收一个数据流 $\vec{u}$ （或其前缀），并返回处理完 $\vec{u}$中的前 t 次更新后，$(\mathcal{A}_{\text{shadow}1}, \mathcal{A}_{\text{shadow}2}, \ldots, \mathcal{A}_{\text{shadow}k})$ 内部状态中的下一个比特。更新相应 $\mathcal{A}_{\text{影子}j}$ 状态中的对应比特。
- 如果 $\left(\sum_{j=1}^{i} q_j\right) \bmod \eta k s = 0$ ，则退出内层循环。否则，继续内层循环。

(e) 对于 $j \in [k]$ ，令 $\mathcal{A}_j^{\text{active}} \leftarrow \mathcal{A}_j^{\text{next}} + \mathcal{A}_j^{\text{shadow}}$。

证明。固定 $j \in \{1, \ldots, \tau - 1\}$。在时间 t_{j-1} ，字符串 $r_i^{t_{j-1}}$ 被用于对 $\mathcal{A}_i^{\text{shadow}}$、$\mathcal{A}_i^{\text{next}}$ 进行随机化，以用于 $i \in [k]$。我们关注内层循环的执行，也就是时间步 $t \in [t_{j-1}, t_j)$ 。在时间步 $t \in [t_{j-1}, t_j)$ 的执行过程中，算法 $\mathcal{A}^{\text{next}}$ 的状态 $\mathcal{S}_v$ 会根据这些时间步的数据流更新而更新。此外，在时间步 $t \in [t_{j-1}, t_j)$ 期间，算法 RobustAdvice 会发出与输入流 $\vec{x}_{t_{j-1}}$ 相对应的建议查询。

，其随机化字符串为 r^{j-1}，其中每个查询都会从 $\mathcal{A}^{\text{shadow}}$ 个实例的 k $\mathcal{S}_v$ 状态中恢复一位额外的信息。现在，由于内层循环的大小至少为 $\eta \cdot k \cdot s$，其中 s 是状态 $\mathcal{S}_v$ 的位数，因此在时间步 $t \in [t_{j-1}, t_j)$ 期间，通过这些建议查询，所有 k 个 $\mathcal{A}^{\text{shadow}}$ 实例的状态 $\mathcal{S}_v$ 中的全部 s 位都会被恢复。也就是说，每个实例 $\mathcal{A}_i^{\text{shadow}}$ 都由与 $\vec{x}_{t_{j-1}}$ 对应的 $\mathcal{S}_v$ 组成，其随机化字符串为 r_i^{j-1}，其中 $i \in [k]$。回想一下，对于 $i \in [k]$，每一对（$\mathcal{A}_i^{\text{shadow}}$，$\mathcal{A}_i^{\text{next}}$）都具有状态 $\mathcal{S}_v$，并且对应于相同的随机化r_i^{j-1}。因此，根据算法 $\mathcal{A}$ 的线性性质，将步骤 2e 中每一对（$\mathcal{A}_i^{\text{shadow}}$，$\mathcal{A}_i^{\text{next}}$）的状态 $\mathcal{S}_v$ 相加，得到对应于输入流 $\vec{x}_{t_j}$ 的实例 $\mathcal{A}^{\text{active}}$ 的状态 $\mathcal{S}_v$。也就是说，我们知道所有 $\mathcal{A}^{\text{active}}$ 个实例都已恢复其在时间 t_j 的 $\mathcal{S}_v$。

□

引理 3.3。设 τ 为算法 *RobustAdvice*的外层循环次数，并将第 j个外层循环的随机字符串向量记为 r^j，其中$j \in [\tau]$。那么，对于每个 $j \in [\tau]$，算法*RobustAdvice* 相对于 r^j满足 (ε, δ)-差分隐私。

证明。将外层循环次数记为 τ，并将算法执行 2a 的时间步记为 t_1、...、t_τ。在每个这样的时间 $t_j \in \{t_1, ..., t_\tau\}$，算法 RobustAdvice 都会使用一个新的数据库：在时间步 $t \in [t_j, t_{j+1})$ 内，算法只使用 r^j。现在，固定$j \in [\tau]$。在时间步 $t \in [t_j, t_{j+1})$ 内，算法 RobustAdvice执行 ηks次 PrivateMed机制。根据组合定理（定理 2.4）和 PrivateMed保证（定理 2.3），取$\varepsilon_0 = \varepsilon/\sqrt{8\eta ks \log(1/\delta)}$可~~确保每个内~~层循环 $j \in [\tau]$ 相对于其专属数据库r^j满足 (ε, δ)-差分隐私。

□

定理 3.4（算法 RobustAdvice 具有鲁棒性）。记 $\vec{x}_t$ 为截至时间 t 的输入流。假设 $\mathcal{A}$ 是针对实值函数 g 的无自适应线性算法，并且以至少 $9/10$ 的概率，它在无知设置下对所有 $t \in [m]$ 都是准确的（即它给出的估计值为 $\hat{g}(\vec{x}_t) \in (1 \pm \alpha) \cdot g(\vec{x}_t)$），那么以至少 $1 - \beta$ 的概率，对所有 $t \in [m]$，我们都有：

$$z_t \in (1 \pm \alpha) \cdot g(\vec{x}_t).$$

证明。令 $t_1 < \cdots < t_\tau$ 表示外层循环开始的时刻（步骤 2a）。固定 $j \in [\tau]$，并关注时间段 $t \in [t_{j-1}, t_j)$，在此期间执行第 j 次外层循环迭代。令 $\vec{x}_t = (x_1, \ldots, x_t)$ 表示数据流中的前 t 次更新。根据引理 3.2，在时间 t_{j-1}，所有 k 个 $\mathcal{A}^{\text{active}}$ 实例均已根据输入流 $\vec{x}_{t_{j-1}}$ 完成更新，每个实例对应于 $i \in [k]$ 的随机字符串 r_i^{j-1}。现在我们来说明，在时间段 $t \in [t_{j-1}, t_j)$ 内，这些实例始终保持鲁棒；在此期间，它们接收该时间段内的输入流更新，而 RobustAdvice 的输出是这些实例估计值的函数。令 $\mathcal{A}(r, \vec{x}_t)$ 表示无偏算法 $\mathcal{A}$ 在使用随机字符串 r 实例化，并在看到数据流 $\vec{x}_t$ 后接受查询时的输出。考虑以下函数：

$$f_{\vec{x}_t}(r) = \mathbb{I}\left\{\mathcal{A}(r, \vec{x}_t) \in (1 \pm \alpha) \cdot g(\vec{x}_t)\right\}.$$

根据引理 3.3，算法 RobustAdvice关于字符串集合 r^{j-1}是 $\left(\varepsilon = \frac{1}{100}, \delta = \frac{\varepsilon\beta}{2m}\right)$-差分隐私的。此外，数据流 $\vec{x}_t$ 中的更新是由对手通过对 RobustAdvice返回的估计值进行后处理来选择的，且函数 $f_{\vec{x}_t}(\cdot)$由$\vec{x}_t$定义。由于差分隐私在后处理中具有封闭性，因此我们可以将函数 $f_{\vec{x}_t}(\cdot)$视为

$\vec{}$对字符串集合 r^{j-1}进行差分隐私计算的结果。因此，根据差分隐私的泛化性质（定理 2.5），假设 $k \geq \frac{1}{\varepsilon^2}\log(\frac{2\varepsilon}{\delta})$，则以至少 $(1-\frac{\delta}{\varepsilon})$ 的概率，对于每个 $t \in [t_{j-1}, t_j)$，都有

$$\left| \mathop{\mathbb{E}}_{r}[f_{\vec{x}_t}(r)] - \frac{1}{k}\sum_{i=1}^{k} f_{\vec{x}_t}(r_i^{j-1}) \right| \leq 10\varepsilon = \frac{1}{10}.$$

这对任意 $j \in [\tau]$ 都成立，因此通过选择 $\frac{\delta}{\varepsilon} = \frac{\beta}{2m}$，对所有 $t \in [m]$，以至少 $1-\beta/2$ 的概率成立（对应的随机字符串和函数为 $f_{\vec{x}_t}(r)$）。我们在假设这一情况成立的前提下继续分析。现在注意到，根据 $\mathcal{A}$ 的效用保证，$\mathbb{E}_r[f_{\vec{x}_t}(r)] \geq 9/10$（因为当数据流固定时，其答案以至少 $9/10$ 的概率在 $(1\pm\alpha)$ 的乘法误差范围内准确）。因此，对至少 $(\frac{9}{10}-10\varepsilon)k = 4k/5$ 比例的 $\mathcal{A}$ 执行，我们有 $f_{\vec{x}_t}(r_i) = 1$，这意味着 $y_{t,i} \in (1\pm\alpha)\cdot g(\vec{x}_t)$。也就是说，在每个时间步 $t \in [m]$，至少有 $4k/5$ 个 y_t 中的 $_i$ 满足 $y_{t,i} \in (1\pm\alpha)\cdot g(\vec{x}_t)$。回顾一下，当算法输出一个估计值时，该估计值是通过算法 `PrivateMed` 计算的，该算法在数据库 $(y_{t,1}, \ldots, y_{t,k})$ 上执行。根据定理 2.3，假设 [2]

$$k = \Omega\left(\frac{1}{\varepsilon_0}\log\left(\frac{m}{\beta}\frac{\log m}{\alpha}\right)\right) = \Omega\left(\sqrt{\eta k s \cdot \log\left(\frac{m}{\beta}\right)}\cdot\log\left(\frac{m}{\beta\alpha}\right)\right),$$

那么，以至少 $1-\frac{\beta}{2m}$ 的概率，算法 `PrivateMed` 返回估计值 $y_{t,\ 1}$、.....、$y_{t,\ k}$ 的近似中位数 $\tilde{g}$，满足

$$|\{i : y_{t,i} \geq \tilde{g}\}| \geq \frac{4k}{10} \qquad \text{and} \qquad |\{i : y_{t,i} \leq \tilde{g}\}| \geq \frac{4k}{10}.$$

由于 $y_{i,\ j}$ 中有 $4k/5$ 满足 $y_{t,\ i} \in (1\pm\alpha)\cdot g(\vec{x}_t)$，因此这样的近似中位数 z_t 也必定在范围 $(1\pm\alpha)\cdot g(\vec{x}_t)$ 内。对于每个在时间 $t \in [m]$ 计算出的估计值（近似中位数 z_t），上述结论分别以至少 $1-\frac{\beta}{2m}$ 的概率成立，因此，以至少 $1-\beta/2$ 的概率，它同时适用于整个执行过程中计算出的所有估计值。总而言之，对于所有 $t \in [m]$，以至少 $1-\beta/2$ 的概率具有鲁棒性；对于在其误差保证范围内执行的所有 $t \in [m]$，以至少 $1-\beta/2$ 的概率具有 `PrivateMed`。因此，以至少 $1-\beta$ 的概率，对于所有 $t \in [m]$，`RobustAdvice` 输出 z_t 满足：

$$z_t \in (1\pm\alpha)\cdot g(\vec{x}_t).$$

□

以下定理可由定理 3.4 推出。

定理 3.5。固定任意实值函数 g，并固定 α、$\beta > 0$ 和 $\eta \in \mathbb{N}$。设 $\mathcal{A}$ 是一个用于 g 的无偏线性流式算法，其使用空间为 s，并以失败概率 $1/10$ 保证准确度 α。那么，存在一个用于 g 的对抗鲁棒流式算法，在 ASA 模型中具有查询/建议比率 η、准确度 α 和失败概率 β，并使用空间 $O(\eta s^2\log(m/\beta)\log^2(m/(\beta\alpha)))$。

[2]我们假设 $\mathcal{A}$ 返回的估计值都在范围 $[-n^c, -1/n^c]\cup\{0\}\cup[1/n^c, n^c]$ 内（关于 n 呈多项式有界），其中 $c > 0$ 为某个常数。此外，在运行 `PrivateMed` 之前，我们可以将每个 $y_{t,i}$ 四舍五入到最接近的 $(1+\alpha)$ 的幂，这对误差的影响很小。该范围内至多有 $O(\frac{1}{\alpha}\log\ n)$ 个 $(1\pm\alpha)$ 的幂，因此，`PrivateMed` 以至少 $1-\delta/\lambda$ 的概率保证误差至多为 $\Gamma = O(\frac{1}{\varepsilon_0}\log\ (\frac{\lambda}{\alpha\delta}\log\ n))$。参见定理 2.3。还要回顾一下我们的假设：$\log\ m = \Theta(\log\ n)$。

3.2 ASA 模型的一个负面结果

本节将证明，在带建议的对抗性设置中， ℓ_0 采样这一经典流式问题无法使用次线性空间求解。考虑一个转向流 $\vec{u} = (u_1、 ...、 u_m)$，其中每个 $u_i = (a_i、 \Delta_i) \in [n] \times \{\pm 1\}$。一个 β 误差的 ℓ_0 采样器以至少 $1 - \beta$ 的概率返回一个均匀随机元素，该元素来自

$$\text{support}(u_1, \ldots, u_m) = \left\{ a \in [n] : \sum_{i:a_i=a} \Delta_i \neq 0 \right\},$$

前提是该支撑集非空。下一个定理由 Jowhari 等人 [23], 提出，表明ℓ_0 在无知设置中，采样很容易。

定理 3.6（[23]）。存在一种流式算法，存储空间为 $O\left(\log^2(n) \log(\frac{1}{\beta})\right)$ 位；它以至多 β 的概率报告 FAIL，以至多 $1/n^2$ 的概率报告任意答案，而在其他所有情况下，都会从 support($\vec{u}$) 中生成一个均匀样本。

然而，正如下文将要展示的，这在 ASA 设置下是一个难题。事实上，我们的负面结果甚至适用于更简单的 ℓ_0 采样问题变体：算法可以返回任意元素，而不是随机元素。形式化地说，

定义 3.7。设 X 为一个有限域， $\mathcal{A}$ 为一个处理更新流 $(u_1, \ldots, u_m) \in (X \times \{\pm 1\})$ 的算法，这些更新逐个提供给 $\mathcal{A}$。如果除概率至多为 β的情况外，每当查询 $\mathcal{A}$ 时，算法 $\mathcal{A}$ 都会输出一个相对于当前数据流具有非零频率的元素，则称算法 $\mathcal{A}$ 以失败概率 β 解决了 J_0 问题。也就是说，如果在时刻 i 查询 $\mathcal{A}$，那么它应当输出 support($u_1, \ldots, u_i$) 中的一个元素。

定理 3.8. 设 X 是一个有限域，并设 T 满足 $|X| = \Omega(T)$（足够大）。设 $\mathcal{A}$ 是一个算法，在带建议的对抗性设置下，使用 T 次查询，在 X 上求解 J_0 流式问题，且失败概率至多为 $3/4$。那么 $\mathcal{A}$ 使用的空间为 $\Omega(T)$。此外，即使 $\eta = 1$，也就是说，即使算法 $\mathcal{A}$ 在每次查询后都收到一条建议，这一结论仍然成立。

证明。设 $\mathcal{A}$ 是一个算法，在 ASA 设置下，对域 X 使用 T 次查询进行 J_0 采样，且失败概率至多为 $3/4$。考虑以下思想实验。

输入：大小为 $|Y| = T$ 的 $Y \subseteq X$

1. 对每个 $x \in Y$，将更新 $(x, 1)$ 输入算法 $\mathcal{A}$。

2. 初始化 $\hat{Y} = \emptyset$。

3. 重复 T 次：

   (a) 查询 $\mathcal{A}$ 并获得结果 $x \in X$(b) 如果 $\mathcal{A}$ 请求一条建议，则向其提供一个随机比特 b.$\hat{}$ (c) 将 x 添加到 Y
   (d) 将更新 $(x, -1)$ 输入给 $\mathcal{A}$

$\hat{}$4. 输出 Y。

ˆ 如果 $Y = Y$，我们就说该思想实验成功。根据关于 $\mathcal{A}$ 的假设，对于每个输入 Y，我们的思想实验成功的概率至少为 $2^{-T}/4$。这是因为，如果所有建议比特都正确，那么 $\mathcal{A}$ 成功的概率至少为 $1/4$，而所有建议比特均正确的概率至少为 2^{-T}。因此，必然存在一个 $\mathcal{A}$ 的随机币固定配置，以及一个建议字符串 $\vec{b}$ 的固定配置，使得我们的思想实验在所有可能输入 Y 中至少有 $2^{-T}/4$ 的比例能够成功。[3]

→也就是说，固定 $\mathcal{A}$的随机币和上述建议字符串 b 后，存在一个大小为 $|\mathfrak{B}| \geq \frac{2^{-T}}{4}\binom{|X|}{T}$ 的输入子集 $\mathfrak{B}$，使得对于每个 $Y \in \mathfrak{B}$，在 Y上执行时，我们的思想实验ˆ都会输出 $Y = Y$。最后请注意，算法 $\mathcal{A}$ 在步骤 1 结束时的内部状态决定了思想实验的结果。因此，由于至少有 $\frac{2^{-T}}{4}\binom{|X|}{T}$ 种不同的输出，算法 $\mathcal{A}$必定至少有 $\frac{2^{-T}}{4}\binom{|X|}{T}$ 种可能的不同内部状态，这意味着其空间复杂度（以比特为单位）至少为 $\log\left(\frac{2^{-T}}{4}\binom{|X|}{T}\right)$；只要 $|X| = \Omega(T)$（足够大），这个值就大于 T。

□

4 有界中断下的对抗性流处理（ASBI）

本节介绍我们针对 ASBI 模型的结果，该模型在第 1.2.2 节中定义。我们从通用变换开始。

4.1 ASBI 模型的通用构造

我们的构造在算法 `RobustInterruptions` 中给出。以下定理说明了它的性质。

定理 4.1。任取函数 g，并固定 α、$\beta > 0$。设 $\mathcal{A}$ 是针对 g 的无知流式算法，使用空间 s，并保证准确度为 α、失败概率为 β。那么，存在一种针对 g 的对抗鲁棒流式算法，能够抵御 R 次中断，并保证准确度为 5α、失败概率为 $O(R\beta)$，使用空间 $O(Rs)$。

固定一个对手 $\mathcal{B}$，并考虑算法 `RobustInterruptions` 与对手 $\mathcal{B}$ 之间的交互。对于 $r \in [R]$，令 i_r 表示返回 $z^{\text{check}}_{r\,i_r}$ 的时间步。

引理 4.2。固定 $r^* \in [R]$。以至少 $1-\beta$ 的概率，$\mathcal{A}^{\text{check}}_{r^*}$ 在时间 1、2、...、i_{r^*} 返回的答案都是 α-准确的。也就是说，对于每个 $1 \leq i \leq i_{r^*}$，都有 $z^{\text{check}}_{r^*,i} \in (1\pm\alpha)\cdot g(x_1, ..., x_i)$。

证明。为简单起见，我们假设对手 $\mathcal{B}$ 是确定性的（根据一个简单的平均论证，这样假设不失一般性）。固定算法 $\mathcal{A}$ 的所有副本的随机性，但不固定 $\mathcal{A}^{\text{check}}_{r^*}$ 的随机性。令 $\texttt{RI}_{r^*}$ 为算法 `RobustInterruptions` 的一个变体，它与 `RobustInterruptions` 完全相同，直到 r 变为 r^* 的时间步 i^*。在时间 $i \geq i^*$，算法 $\texttt{RI}_{r^*}$ 只输出 $z^{\text{answer}}_{r^*\,i}$，即由 $\mathcal{A}^{\text{answer}}_{r^*}$ 给出的答案。注意，算法 RI_{r^*} 中不存在 $\mathcal{A}^{check}_{r^*}$。

由于我们固定了 $\mathcal{A} \neq \mathcal{A}^{\text{check}}_{r^*}$ 各副本所用的随机币，$\mathcal{B}$ 与 $\texttt{RI}_{r^*}$ 之间的交互是确定性的。特别地，它会生成唯一的数据流 $\vec{x}_{r^*}$。根据算法的效用保证

[3]否则，考虑均匀地抽取一个输入 Y。我们有 $\frac{2^{-T}}{4} \leq \Pr_{r,\vec{b},Y}[\mathcal{A}_{r,\vec{b}}(Y)\text{成功}] = \sum_{r,\vec{b}} \Pr[r,\vec{b}] \cdot \Pr_Y[\mathcal{A}_{r,\vec{b}}(Y)\text{成功}] < \sum_{r,\vec{b}} \Pr[r,\vec{b}] \cdot \frac{2^{-T}}{4} = \frac{2^{-T}}{4}$，这就构成了矛盾。这里，$r$ 表示 $\mathcal{A}$ 的随机性，而 $\vec{b}$ 是建议字符串。

算法 **2** `RobustInterruptions`

输入：参数 R，用于限定可能发生的中断次数。
使用的算法：一个无知流式算法 $\mathcal{A}$，其空间为 s，准确度为 α，置信度为β。

1. 初始化 $2R$ 个算法 $\mathcal{A}$的独立实例，记为 $\mathcal{A}_1^{\text{answer}}$，…，$\mathcal{A}_R^{\text{answer}}$和$\mathcal{A}_1^{\text{check}}$，…，$\mathcal{A}_R^{\text{check}}$。设置 $r = 1$。
2. 对于 $i = 1,2,\ldots,m$:

   (a) 获取数据流 $x_i \in X$中的下一个项目。(b) 将 x_i 输入算法 $\mathcal{A}$的所有副本。(c) 令$z_{r\,i}^{\text{answer}}$,和$z_{r\,i}^{\text{check}}$,分别表示由 $\mathcal{A}_r^{\text{answer}}$和 $\mathcal{A}_r^{\text{check}}$返回的答案。(d) 如果$z_{r}^{\text{answer}}{}_{,i} \in (1\pm 2\alpha)\cdot z_{r\,i}^{\text{check}}$,，则输出$z_{r\,i}^{\text{answer}}$,。否则，输出$z_{r\,i}^{\text{check}}$,并设置 $r \leftarrow r+1$。(e) 如果 $r > R$，则失败。否则继续下一次迭代。

$\mathcal{A}_{r^*}^{\text{check}}$在此数据流上运行时，以至少 $1-\beta$ 的概率在整个数据流中保持 α 准确度。

观察到在时间步 i_{r^*} 之前，对手与算法 `RobustInterruptions` 交互所生成的数据流与数据流 $\vec{x}_{r^*}$ 完全相同，即可推出该引理。 □

引理 4.3. 以至少 $1 - R\beta$ 的概率， *`RobustInterruptions`* 给出的所有答案（返回 FAIL 之前）都具有 5α 准确度。

证明。由引理 4.2 的并集界以及 `RobustInterruptions` 的步骤 2d 可得。 □

引理 4.4. 算法 *`RobustInterruptions`* 返回 FAIL 的概率至多为 $2R\beta$。

证明。令 $j_1, j_2, \ldots, j_R$ 表示对手进行中断的时间步。也就是说，j_1 是对手首次切换数据流后缀的时间步，j_2 是此事第二次发生的时间步，依此类推。再令 $p_1, p_2, \ldots, p_R$ 表示算法 `RobustInterruptions` 执行期间参数 r 增大的时间步。具体而言，p_ℓ 是 i 时刻，此时 r 变为等于 $\ell + 1$。我们证明，对每个 $r \in [R]$，以至少 $1 - 2r\beta$ 的概率都有 $j_r \le p_r$。（也就是说，中断发生得比 r 增长“更快”。）

证明通过对 r 使用归纳法完成。对于基例，$r = 1$，令 $\vec{x}_1$ 表示对手选择的第一个数据流。根据 $\mathcal{A}$ 的效用保证，以至少 $1-2\beta$ 的概率，$\mathcal{A}_1^{\text{answer}}$ 和 $\mathcal{A}_1^{\text{check}}$ 对该数据流都是 α 准确的，此时 r 不会增加。因此，以至少 $1-2\beta$ 的概率有 $j_1 \le p_1$。

归纳步骤类似：固定 $r \in [R]$，并假设 $j_r \le p_r$，根据归纳假设，该情况发生的概率至少为 $(1 - 2r\beta)$。令 $\vec{x}_r$ 表示在时间 p_r 之前由对手指定的最后一个数据流。注意，$\mathcal{A}_{r+1}^{\text{answer}}$ 和 $\mathcal{A}_{r+1}^{\text{check}}$ 的内部随机性与该数据流相互独立。因此，根据 $\mathcal{A}$ 的效用保证，以至少 $1-2\beta$ 的概率，$\mathcal{A}_{r+1}^{\text{answer}}$ 和 $\mathcal{A}_{r+1}^{\text{check}}$ 相对于该数据流都是 α-准确的，在这种情况下，r 不会增加。综上，以至少 $1 - 2(r+1)\beta$ 的概率，$j_{r+1} \le p_{r+1}$ 成立。

回顾整个执行过程中至多发生 R 次中断，即可得出该引理。因此，以至少 $1 - 2R\beta$ 的概率，r 不会增加到超过 R，且算法不会失败。

□

定理 4.1 现在可由引理 4.2、4.3 和 4.4 合并得出。

4.2 ASBI 模型的一个否定性结果

定理 4.5. 对于每个 R，都存在一个定义域大小为 poly(R)、流长度为 poly(R) 的流式问题，在 ASBI 模型中，使用 R次中断且准确度达到常数级（足够小）时，至少需要 Ω(R) 的空间才能解决；但在无知设置下，使用 polylog(R) 的空间即可解决。

该定理可通过重新考察 Kaplan 等人关于（纯粹）对抗模型的否定性结果得出。 [25] 他们提出了一个称为 SADA 问题的流式问题：该问题在无知设置下很容易解决，但在对抗性设置下需要大量空间才能解决。为了得到他们的困难性结果， [25] 他们将学习理论中的一个困难问题（称为自适应数据分析（ADA）问题）归约到在对抗性设置下使用小空间解决 SADA 问题的任务。

在 ADA 问题中，目标是设计一种机制 $\mathcal{A}$，它首先获取一个数据集 D，其中包含来自某个未知分布 $\mathcal{P}$ 的 n 个独立同分布样本，然后针对 $\mathcal{P}$ 回答 k 个自适应选择的查询。重要的是， $\mathcal{A}$ 的答案必须针对底层分布 $\mathcal{P}$ 准确，而不仅仅是针对经验数据集 D 准确。Hardt、Ullman 和 Steinke [19, 28] 证明了 ADA 问题需要很大的样本复杂度。具体而言，他们证明了，对于这一问题，任何高效的 [4] 机制都必须具有 Ω($\sqrt{k}$) 的样本复杂度 $n \geq$。

定理 4.5 可由以下两个观察得出：一个关于 [25] 针对 SADA 问题的否定结果，另一个关于 [19, 28] 针对 ADA 问题的底层困难性结果：

1. 在针对 ADA 问题的困难性结果 [19, 28] 中，对手使用 O（n）轮自适应性来生成查询，其中 n 是样本量。具体而言，尽管对手在整个交互 [5]，过程中提出了 poly(n) 个查询，但这些查询是分成 O（n）个批次生成的，其中第 j 个批次中的查询仅依赖于对先前批次中的查询所给出的答案。

2. Kaplan 等人从 ADA 问题到 SADA 问题的归约保持了自适应轮次的数量。也就是说，Kaplan 等人的归约将一个为 ADA 问题分批次生成查询的对手，转换为一个为 SADA 问题使用中断的对手。

我们指出，定理 4.5 即使对于这样一种模型也成立：在该模型中，流式算法得到增强，并在每个中断轮次期间都会收到指示。这是因为，借助 [25], 的技术，流式算法可以识别新批次出现的确切轮次，而该轮次对应一个中断轮次。

在附录 A 中，我们概述了 [25, 19, 28], 中的必要细节，并更详细地说明了为获得定理 4.5 所需的修改。

[4] [19, 28] 的结果适用于所有计算高效机制；或者，也适用于一类他们称为 *natural* 机制的无界机制。[5] 对手在 [19] 中提出 $O(n^3)$，并在 [28] 中提出 $O(n^2)$。

参考文献

[1] K. J. Ahn、S. Guha 和 A. McGregor。通过线性测量分析图结构。载于 SODA，第 459–467 页，2012 年。[2] K. J. Ahn、S. Guha 和 A. McGregor。图草图：稀疏化、扩展器与子图。载于 PODS，第 5–14 页，2012 年。[3] N. Alon、Y. Matias 和 M. Szegedy。频率矩近似的空间复杂度。《计算机与系统科学杂志》，58(1):137–147，1999 年。[4] I. Attias、E. Cohen、M. Shechner 和 U. Stemmer。通过差分隐私和差值估计器实现对抗性流处理的框架。CoRR，abs/2107.14527，2021 年。[5] R. Bassily、K. Nissim、A. D. Smith、T. Steinke、U. Stemmer 和 J. Ullman。自适应数据分析的算法稳定性。载于 STOC，第 1046–1059 页，2016 年。[6] A. Beimel、K. Nissim 和 U. Stemmer。隐私学习与数据净化：纯差分隐私与近似差分隐私。载于 APPROX-RANDOM，第 363–378 页，2013 年。[7] O. Ben-Eliezer、T. Eden 和 K. Onak。通过稠密-稀疏权衡实现对抗鲁棒流处理。载于 SOSA@SODA，第 214–227 页，2022 年。[8] O. Ben-Eliezer、R. Jayaram、D. P. Woodruff 和 E. Yogev。对抗鲁棒流算法的框架。《ACM期刊》，69(2):17:1–17:33，2022 年。[9] O. Ben-Eliezer 和 E. Yogev。采样的对抗鲁棒性。载于 PODS，第 49–62 页，2020 年。[10] M. Bun、C. Dwork、G. N. Rothblum 和 T. Steinke。通过截断集中差分隐私实现可组合且多用途的隐私。载于 STOC，第 74–86 页，2018 年。[11] M. Bun、K. Nissim、U. Stemmer 和 S. P. Vadhan。阈值函数的差分隐私发布与学习。载于 FOCS，第 634–649 页，2015 年。[12] E. Cohen、X. Lyu、J. Nelson、T. Sarl´os、M. Shechner 和 U. Stemmer。关于 CountSketch 对自适应输入的鲁棒性。载于 K. Chaudhuri、S. Jegelka、L. Song、C. Szepesv´ari、G. Niu 和 S. Sabato 编，《国际机器学习大会》，ICML 2022，2022年7月17日至23日，美国马里兰州巴尔的摩，第 162 卷《机器学习研究论文集》，第 4112–4140 页。PMLR，2022 年。[13] E. Cohen、X. Lyu、J. Nelson、T. Sarl´os 和 U. Stemmer。阈值与拟凹优化的最优差分隐私学习。CoRR，abs/2211.06387，2022 年。[14] C. Dwork、V. Feldman、M. Hardt、T. Pitassi、O. Reingold 和 A. L. Roth。在自适应数据分析中保持统计有效性。载于 STOC，第 117–126 页，2015 年。[15] C. Dwork、F. McSherry、K. Nissim 和 A. Smith。在隐私数据分析中根据敏感度校准噪声。载于 TCC，第 265–284 页，2006 年。[16] C. Dwork、G. N. Rothblum 和 S. P. Vadhan。提升与差分隐私。载于 FOCS，第 51–60 页，2010 年。

[17] A. C. Gilbert、B. Hemenway、A. Rudra、M. J. Strauss 和 M. Wootters。恢复简单信号。载于信息论与应用研讨会（ITA），第 382–391 页，2012 年。[18] A. C. Gilbert、B. Hemenway、M. J. Strauss、D. P. Woodruff 和 M. Wootters。通过隐私保护实现可重复使用的低误差压缩采样方案。载于 IEEE 统计信号处理研讨会（SSP），第 536–539 页，2012 年。[19] M. Hardt 和 J. Ullman。交互式数据分析中防止假发现并非易事。载于 FOCS。IEEE，2014年10月19日至21日。[20] M. Hardt 和 D. P. Woodruff。线性草图对自适应输入的鲁棒性如何？载于 STOC，第 121–130 页，2013 年。[21] A. Hassidim、H. Kaplan、Y. Mansour、Y. Matias 和 U. Stemmer。通过差分隐私实现对抗鲁棒流处理算法。《ACM期刊》，2022 年。[22] H. Jowhari、M. Saglam 和 G. Tardos。lp 采样器、在数据流中查找重复项及相关问题的紧确界。载于 PODS，第 49–58 页。ACM，2011 年。[23] H. Jowhari、M. Saˇglam 和 G. Tardos。lp 采样器、在数据流中查找重复项及相关问题的紧确界。载于第三十届 ACM SIGMOD-SIGACT-SIGART 数据库系统原理研讨会论文集，第 49–58 页，2011 年。[24] H. Kaplan、K. Ligett、Y. Mansour、M. Naor 和 U. Stemmer。私密地学习阈值：弥合指数差距。载于 COLT，第 2263–2285 页，2020 年。[25] H. Kaplan、Y. Mansour、K. Nissim 和 U. Stemmer。利用有界存储模型区分自适应流处理与非自适应流处理。载于 CRYPTO，第 94–121 页，2021 年。[26] F. McSherry 和 K. Talwar。通过差分隐私实现机制设计。载于 FOCS，第 94–103 页，2007 年。[27] I. Mironov、M. Naor 和 G. Segev。对抗环境中的草图处理。《SIAM计算杂志》，40(6):1845–1870，2011 年。[28] T. Steinke 和 J. R. Ullman。交互式指纹码与防止假发现的难题。载于 2016年信息论与应用研讨会，ITA 2016，美国加利福尼亚州拉霍亚，2016年1月31日至2月5日，第 1–41 页。IEEE，2016 年。[29] U. Stemmer。数据分析视角下的个体与隐私。博士论文，内盖夫本-古里安大学，2016 年。导师——Amos Beimel 和 Kobbi Nissim。[30] D. P. Woodruff 和 S. Zhou。通过差值估计器获得对抗鲁棒数据流和滑动窗口的紧确界。载于 FOCS，第 1183–1196 页，2022 年。

A 定理 4.5 的详细说明

本节将详细阐述推导出定理 4.5 的各个组成部分：

1. [19, 28] 关于 ADA 问题的困难性结果。

2. Kaplan 等人 [25] 从 ADA 问题到 SADA 问题的归约。

目的是证明，尽管对手 [28] 发起的攻击中查询总数为 $O(R^2)$，而且在 [25], 的归约中，这些 $O(R^2)$ 个查询会被编码，并通过长度为 poly(R) 的较长数据流传递给流式算法，但归约的自适应轮次数量仍为 $O(R)$，其中 R是中断次数的上界。换言之，我们表明， [25] 针对解决 SADA 问题的算法所给出的否定性结果实际上更强：它排除的是 ASBI 模型中的算法，而不仅仅是纯对抗模型中的算法。

A.1 IFPC 自适应性级别

现在，我们将详细讨论困难性结果中的自适应轮次数量 [19, 28] （而非轮次总数）。具体而言， [28] 提出了一个由对手 $\mathcal{P}$ 和指纹码 $\mathcal{F}$参与的双人博弈协议（见下文定义）。在他们的论文中，[28] 结合这一博弈定义与一种编码（即交互式指纹码，记作 IFPA），证明了针对自适应分析者所能保证的准确查询数量上界。分析者的角色由 $\mathcal{F}$扮演。 [28] 为 $\mathcal{F}$ 提出了一种算法，保证 $\mathcal{P}$ 在经过 $O(n^2)$ 个自适应轮次后失败（即对分析者的某个查询返回不准确的答案），其中 n 是博弈中数据库的大小。博弈定义如下：

博弈协议：对手 $\mathcal{P}$ vs IFPC $\mathcal{F}$

1. $\mathcal{P}$选择一个子集 $S^1 \subseteq [N]$，且不让 $\mathcal{F}$知道。

2. 对于 $j = 1,\ldots,\ell$:

   (a) $\mathcal{F}$ 输出一个列向量 $c^j \in \{\pm 1\}^N$(b) 设 $c^j_{S^j} \in \{\pm 1\}^{|S^j|}$ 为 C^j 在坐标 S^j上的限制，并将其提供给 $\mathcal{P}$。(c) $\mathcal{P}$ 输出 $a^j \in \{\pm 1\}$，并将其提供给 $\mathcal{F}$。(d) $\mathcal{F}$指控一组（可能为空的）用户 $I^j \subseteq [N]$。设 $S^{j+1} = S^j \setminus I^j$。

在上述博弈中， $\mathcal{P}$ 定义为用户 S^1组成的一个联盟，每轮只接收部分码字 C^j。 $\mathcal{P}$ 的目标是保持一致。这意味着，每当查询 c_j 全为 $+1$ 或全为 -1时， $\mathcal{P}$ 必须分别回答 $+1$ 或 -1 。每轮中， $\mathcal{F}$ 选择要指控的某个子集 $I^j \subseteq [N]$ 。这意味着，下一轮中，这些条目也将受到 $\mathcal{P}$的限制。 $\mathcal{F}$ 的目标是使 $\mathcal{P}$ 变得不一致，同时不能指控太多不属于 $\mathcal{P}$ 的用户（即用户 S^1）。

在他们的论文中， [28] 展示了一种 $\mathcal{F}$ 算法，该算法能保证在经过 $O(n^2)$轮次后，使任意 $\mathcal{P}$ 出现不一致性（见算法 3）。在该算法（3）中，$D_{\alpha,\ \zeta}$ 是一个 $[0,1]$ 上的分布，从中抽取伯努利参数 p^j ，并用它生成第 j个码字$c^j_{1\ldots N} \in p^j$ 和ϕ^{p^j} ： $\{0,1\} \to \mathbb{R}$ 是一个函数，用于衡量用户 i的输入 c^j_i 与输出 a^j之间的相关量。其主要思想是在迭代过程中为每位用户累积这一相关量；一旦某位用户的该相关量越过某个阈值，算法便判定该用户属于联盟，并将其标记为已指控。

值得注意的是，就我们的应用场景而言，算法 3 中的所有码字 c^j 都可以从中抽取

算法 **3** IFPC(n, N, δ, β) [**28**]

输入：参数： N 是用户数， $1 \le n \le N$ 是联盟规模， $\delta \in (0, 1]$ 是失败概率， $\beta < 1/2$ 是允许的不一致轮次占比。

1. Set parameters $\alpha = (1/2 - \beta)/4n = \Omega(1/n)$, $\zeta = 3/8 - \beta/4 = 1/2 - 1/4(1/2 - \beta)$, $\sigma = O((n/(1/2 - \beta)^2)/\log(\delta^{-1}))$, $\ell = O((n^2/(1/2 - \beta)^4)\log(1/\delta))$

2. 对每个 $i \in [N]$, 令 $s_i^0 = 0$

3. 对于 $j = 1,\ldots,\ell$:

   (a) 绘制 $p_j \sim D_{\overline{\alpha,\ \zeta}}$和 $c_{1\ldots N}^j \sim p^j$。(b) 发出 $c^j \in \{\pm 1\}^N$ 作为挑战，并接收 $a^j \in \{\pm 1\}$ 作为响应。(c) 对于 $i \in [N]$, 令 $s_i^j = s_i^{j-1} + a_j \cdot \phi^{p^j}(c_i^j)$。(d) 指控 $I^j = \left\{ i \in [N] | s_i^j > \sigma \right\}$

提前。也就是说，它们独立于答案 a^j。码字不是一次性全部呈现给$\mathcal{P}$，而是逐个呈现，因此算法 3 是交互式的，但不是自适应的。

然而，博弈协议本身是自适应的。在协议步骤 2b 中，发送给 $\mathcal{P}$ 的码字部分仅限于联盟用户 S^j。也就是说，这些用户来自初始联盟 S^1，且尚未受到指控；这取决于所有先前的答案 a^j。因此，自适应性体现在确定向 $\mathcal{P}$ 提供码字的时刻。

现在我们得出结论：这类确定时刻的数量只有 $O(n)$：未被指控的联盟用户列表 S^j 是单调递减的，因此最多可以更新 n 次。现在回想一下，算法的长度为 $O(n^2)$ 次迭代。此外，在 S^j的连续两次修改之间，提供给 $\mathcal{P}$的码字集合是固定的。因此，记 $j_1, \ldots, j_k$为列表 S^j 被修改的时刻（对于某个 $k \le n$），那么在迭代$[j_i, \ldots, j_{i+1})$期间的 $i \in [k]$，博弈协议是不自适应的。

A.2 ADA 到 SADA 的归约保持自适应性级别

现在我们来看 Kaplan 等人 [25] 的困难性结果。在他们的论文中，Kaplan 等人展示了一个从 ADA 问题到 SADA 问题（流式自适应数据分析）的归约；其中 ADA 问题已被证明是困难的（在 [28], 中需要 $O(n^2)$ 个查询轮次）。概括来说，其思路包含以下几个部分：

1. 流生成与流式算法：数据流由 IFPC 的博弈协议确定，其中 $\mathcal{P}$ 是一个流式算法，用于回答查询（编码在输入流中）。

2. 压缩：已知空间较小（远小于其输入数据集大小）的算法具有很强的泛化性质。因此，如果一个小空间算法能够解决 SADA 问题（查询次数为 $\gg O(n^2)$，空间为 $\ll n$），那么它实际上也必须能够解决具有相同查询次数的底层统计 ADA 问题，这与 [28] 矛盾。

这两个组成部分表明，任何解决 SADA 问题的算法都必须使用 Ω(n) 的空间。由于在我们的下界范式中，有界中断的次数就是第一个组成部分所确定的自适应轮次数，因此我们只详细说明该组成部分。

该数据流是相对于 IFPC 博弈协议分两个阶段定义的。第一阶段通过数据流中的 n 次更新来设置集合 S^1。每次流更新都会在 S^1 中编码一个用户。接着，在第二阶段，对于 $O(n^2)$ 轮博弈协议，会向流式算法发送受限查询 $c^j_{S^j}$，并要求其给出答案。一个小问题是，这些查询的编码大小均为 poly(n)，因此流式算法需要接收 poly(n) 次流更新才能收到查询（因此，该攻击的流长度为 poly(n)）。每次收到这样的（编码后的）查询后，流式算法都必须作答（通过 [28], ，任何此类算法在收到 $O(n^2)$ 次查询后都必然失败）。

不过，由于归约使用的是 [28] 的博弈协议，我们仍然可以观察到只有 $O(n)$ 个自适应轮次。因此，在 ASBI 模型中，对于具有 $O(n)$ 轮中断的情形，也存在从 ADA 问题到 SADA 问题的类似归约。这意味着该模型的空间存在一个下界。