复杂时间戳事件流的快速多方面挖掘

Kota Nakamura，SANKEN，大阪大学，日本 kota88@sanken.osaka-u.ac.jp

Yasuko Matsubara，SANKEN，大阪大学，日本 yasuko@sanken.osaka-u.ac.jp

Koki Kawabata，SANKEN，大阪大学，日本 koki@sanken.osaka-u.ac.jp

Yuhei Umeda，人工智能实验室，富士通，日本 umeda.yuhei@fujitsu.com

Yuichiro Wada，人工智能实验室，富士通；AIP，RIKEN，日本 wada.yuichiro@fujitsu.com

Yasushi Sakurai SANKEN，大阪大学，日本 yasushi@sanken.osaka-u.ac.jp

arXiv:2303.03789v2 [cs.LG] 6 Jul 2023

摘要

对于包含多个属性、随时间演变的海量在线事件数据流，例如在线购物日志*(*商品，价格，品牌，时间*)*，我们如何汇总大型动态高阶张量流？又如何发现其中的隐藏模式、规则和异常？我们的解决方案是聚焦于高阶张量流中的两类模式，即“状态”和“组件”，并为此提出一种高效且有效的方法——CubeScope。具体而言，该方法能够识别突发的不连续性，并辨识不同的动态模式，即“状态”（例如工作日／周末／节假日模式）。在每种状态下，它还会对所有属性（例如商品、价格、品牌和时间）进行多维汇总，并发现代表潜在群组（例如商品／品牌群组）及其关系的隐藏“组件”。凭借简洁而有效的汇总，CubeScope 还能够检测异常的突然出现，并识别实践中出现的异常类型。

我们提出的方法具有以下特点：(a) 有效：能够捕捉动态多方面模式，即状态和组件，并对所有事件进行统计汇总；(b) 通用：适用于在各种类型的张量流上成功应用于数据压缩、模式发现和异常检测；(c) 可扩展：我们的算法不受数据流长度及其维度的影响。对真实数据集进行的大量实验表明，CubeScope 能够正确发现有意义的模式和异常，并且在准确性和执行速度方面始终优于最先进的方法。

CCS概念

• **信息系统** →**数据挖掘**。

ACM参考文献格式：

KotaNakamura、YasukoMatsubara、Koki Kawabata、Yuhei Umeda、Yuichiro Wada 和 Yasushi Sakurai。2023年。复杂时间戳事件流的快速多方面挖掘。载于《*ACM Web Conference 2023 (WWW '23)* 论文集》，*2023*年*4*月*30*日至*5*月*4*日，美国得克萨斯州奥斯汀。ACM，美国纽约州纽约市，12页。https://doi.org/10.1145/3543507.3583370

特此授权，允许为个人或课堂使用而免费制作本作品的全部或部分内容的数字副本或纸质副本，前提是这些副本不得用于营利或商业目的，也不得分发，并且副本首页须保留本声明及完整引文。对于本作品中归他人而非作者所有的组件，必须尊重其版权。注明出处后，允许摘录。若要以其他方式复制、转载、发布到服务器或分发至列表，则须事先获得具体许可和/或支付费用。请向 permissions@acm.org 申请许可。*WWW '23，2023*年*4*月*30*日至*5*月*4*日，美国得克萨斯州奥斯汀© 2023年版权归所有者/作者所有。出版权已授权给 ACM。ACM ISBN 978-1-4503-9416-1/23/04...$15.00 https://doi.org/10.1145/3543507.3583370

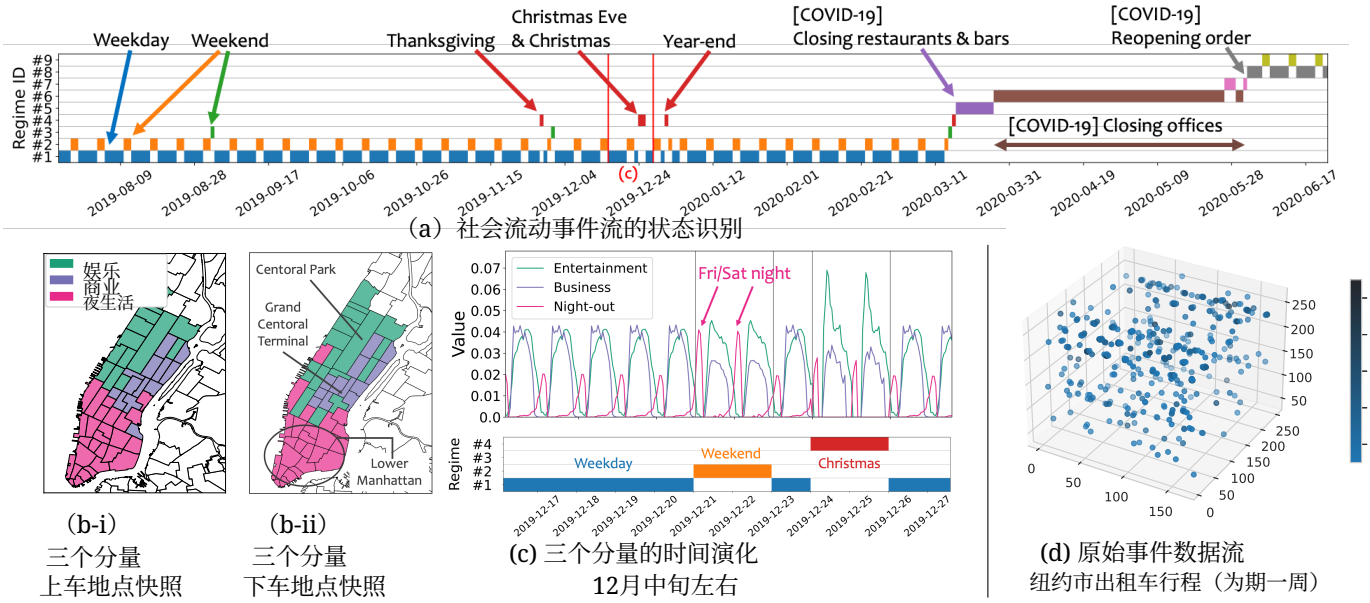
1 引言

面对规模庞大的带时间戳事件在线流，我们如何从统计角度概括所有事件流，并找出重要模式、规则和异常？许多实际应用都会生成并收集带时间戳的事件数据 [10, 17, 29, 84],，包括在线营销分析 [52, 75],、社交网络/基于位置的服务 [28, 71], 以及网络安全系统 [19, 80], ，其交易规模越来越大，速率也越来越快。例如，在线购物服务每秒可能生成数百万条日志条目，其中包含丰富的商品和用户信息。服务提供商希望通过调查在线购买模式及隐藏的用户/商品关系，投放定向广告并检测欺诈活动。

在此，假设我们拥有大量事件日志，这些日志包含多个属性，例如在线购物：（商品、价格、品牌、时间）以及本地出行活动：（接送地点、时间）。其中，大量事件条目以高比特率在线到达，我们将其称为“复杂时间戳事件流”。这些数据表示为高阶张量流，例如四阶商品-价格-品牌-时间张量流；这不同于先前研究的多变量时间序列 [37], 张量[50], 或元素流 [61], ，具体而言，正如第 3.1 节稍后所述，其高维、稀疏和半无限特性使现有方法失效，甚至影响我们对数据的理解。那么，复杂时间戳事件流的良好表示是什么？这正是本文所关注的问题。我们首先提出一种紧凑而强大的表示，用于概括张量流的半无限集合。具体而言，我们旨在捕捉两类模式，即“状态”和“组件”。

在实践中，真实数据流包含各种持续时间不同且彼此有别的时间动态模式，即“状态”，例如在线购物服务或出租车行程中的工作日／周末／节假日模式。在每种状态中，由多个属性构成的一组事件具有相似的行为和潜在交互。我们引入潜在“组件”的概念，用以捕捉每个属性中的隐藏群组（例如商品群组和典型的上车地点）及其关系。

一个重要的应用场景是网络安全。例如，拒绝服务或端口扫描攻击等多种类型的入侵/异常会突然发生，需要实时检测和分析，以尽量减少危害。因此，我们也希望回答这个问题：如何快速检测异常并识别其类型？然而，这极具挑战性，因为异常的迹象会出现在一个或多个属性中（例如，源



（a）社会流动事件流的状态识别

（b-i）三个分量上车地点快照

（b-ii）三个分量下车地点快照

(c) 三个分量的时间演化 12月中旬左右

(d) 原始事件数据流 纽约市出租车行程（为期一周）

图1：CubeScope 对纽约市出租车行程进行实时建模：(a) 它逐步识别出不同的随时间演变的模式（即状态）及其变换点。具体而言，状态#1（蓝色）对应工作日，而状态#2、#3 和 #4（橙色、绿色和红色）则表示周末和公共假日。此外，它还能自适应地识别出反映 COVID-19疫情期间社会状况的突发状态转变（状态#5、#6、···）。它发现了可解释的组件，能够概括所有属性（即上车地点、下车地点和时间），尤其是 (b) 上车/下车地点和 (c) 时间属性。(d) 原始数据是一个稀疏高维张量，没有明显的组件或状态。

IP地址、数据包大小、...），而事件流会随时间演变，新的异常类型可能出现，正常行为的概念也会发生变化。

本文提出了 CubeScope，这是一种高效且有效的挖掘方法，能够应对上述问题。CubeScope 会监控高阶张量流，并增量式地识别动态多方面模式，即状态和组件，以及异常，同时更新各自的信息。直观而言，我们希望解决的问题如下：

非正式问题1. 给定一个高阶张量流$\mathcal{X}$，其中包含具有多个属性和时间戳的事件，

- 找出$\mathcal{X}$ 的紧凑描述，以概括所有事件、– 不同的动态模式（即状态）、– 多方面潜在趋势（即组件），
- 报告异常及其类型

并在任意时间点进行增量式快速处理。

结果预览。 图1（a）-（c）展示了我们在本地出行数据中发现的一些现象。该数据集包含纽约市的出租车行程事件（上车地点ID、下车地点ID、时间），时间戳精确到小时，涵盖2019年7月1日至2020年6月30日。图1（d）展示了原始数据。数据表示为三阶张量流，其中每个方面对应一个属性。需要注意的是，这个张量既稀疏又高维，也就是说，每个方面／属性中都有大量维度。它没有呈现任何明显的模式，无论是状态还是组件。

- 状态识别： 如图1（a）所示，CubeScope 增量式地发现了九种状态（即不同的随时间演变的模式）。具体而言，它识别出状态#1（蓝色）和状态#2（橙色），分别对应工作日和周末。临近年末时，它识别出新的状态，状态#3和状态#4（绿色、红色），它们与包括感恩节、圣诞节和年末在内的某些节日相吻合。新的状态#5（紫色）和状态#6（棕色）表明人类移动发生了突变。事实上，由于一种新型病毒性大流行病COVID-19的出现，该市于3月16日下令关闭餐馆/酒吧 [11]，随后于3月22日下令关闭办公室 [12]。最后，在6月8日的重新开放令 [13], 之后，我们的方法生成了新的工作日和周末人类移动模式，即状态#8（灰色）和状态#9（深黄色），分别对应办公室工作人员和实体店零售购物恢复活动。
- 多方面成分分析：CubeScope 为每个属性提供可解释的摘要性组件。图1(b)展示了状态#1中上车地点/下车地点的三个主要成分，我们将其分别命名为“娱乐”“商业”和“夜生活”。这些区域与我们的直觉一致：娱乐组件分布在中央公园及附近博物馆周边，商业组件集中于中央车站等主要火车站，而夜生活组件则对应曼哈顿下城周边区域，该地区拥有大量餐厅和酒吧。图1(c)展示了12月中旬前后时间属性的三个主要成分。它们显示，娱乐组件在圣诞节期间出现峰值。商业组件在工作日持续呈现较高峰值，而在周末和圣诞节时数值较低。最后，夜生活组件在午夜时呈现峰值，尤其是在周末（即周五/周六午夜）。

贡献。 本文的主要贡献如下：

表1：方法的能力。

	TICC/++	CubeMarker	T-LSTM	LDA/NTM/++	TriMine	DBSTREAM/++	LOF/++	MemStream	CubeScope
高维张量	-	some	-	-	✓	-	-	✓	✓
稀疏性	-	-	✓	✓	✓	-	-	✓	✓
半无限数据	-	-	-	-	-	✓	-	✓	✓
分割	✓	✓	✓	-	-	-	-	-	✓
数据压缩	✓	✓	-	✓	✓	✓	-	-	✓
异常检测	-	-	-	-	-	-	✓	✓	✓
动态多方面模式	-	-	-	-	-	-	-	-	✓

- **高效**：我们提出了动态多方面模式（即状态和组件），用于概括高阶张量流并提供可解释表示。此外，我们将摘要问题形式化，以便在数据压缩范式下捕获这些模式。
- **通用性**：为了解决摘要问题，我们设计了 CubeScope，该工具还可执行数据压缩、模式发现和异常检测。实验结果表明，它在多个领域中具有实用性，例如在线营销分析和网络安全。
- **可扩展性**：我们提出的算法速度快，并且无论对于整个数据流长度还是每个属性的维度，都只需要恒定的计算时间。

可复现性。我们的源代码和数据集可在 [6] 获取。**文章结构**。本文其余部分安排如下：首先介绍相关研究，然后介绍我们提出的模型与算法、实验和结论。

2 相关工作

带时间戳的事件数据挖掘已在许多领域引起广泛关注 [30, 42, 46, 58, 66, 67, 69, 78, 82, 86–88]。表 1 展示了我们的方法相较于其他方法的优势，只有 CubeScope 满足所有要求。

动态建模与分割。如 [40, 48, 54, 83] 所述，线性动力系统（LDS）和隐马尔可夫模型（HMM）等经典方法经过扩展，可用于捕捉序列的不同模式。TICC [37] 基于马尔可夫随机场刻画多变量观测之间的相互依赖关系。这些不同的时间序列模式还使我们能够进行异常检测和预测 [26, 63–65]。已有研究提出了基于张量的时间序列分割方法 [39, 47]，将序列之间的潜在关系纳入考量。以往研究针对连续时间序列设计，因此无法对稀疏张量进行建模。近年来，已有许多深度神经网络模型被提出 [53, 59]。T-LSTM[17] 能够识别具有不规则时间间隔的疾病进展模式。由于这些模型大多是“黑箱”模型，且计算成本高，因此无法处理数据流摘要问题。此外，上述研究均未关注阶数高于 3 的大型稀疏张量。

摘要与聚类。概率生成模型 [18, 71],, 例如潜在狄利克雷分配（LDA） [22] 及其变体 [43, 70, 89], , 广泛用于分析大量类别型数据。近年来，研究者使用变分自编码器 [55, 85] 将主题模型扩展为基于神经网络的模型。变分自编码器 [49]。一组事件可以转换为张量 [41, 44, 76]。TriMine [68]对事件张量进行摘要，并发现维度组。与 [16, 51, 67, 78], 最小描述长度（MDL）原理 [33] 用于对时间序列和动态图进行摘要类似。不同于这些方法，我们的工作聚焦于张量流。数据流的处理与聚类 [14, 32, 36, 60], （例如 DBSTREAM [36], ）也受到广泛关注。然而，这些算法逐个处理数据点，无法捕捉多方面特征。

异常检测。 典型的异常检测方法 [15, 25,31, 61, 81], , 例如局部离群因子 (LOF) [24] 和基于树的方法 [34, 56], , 可通过将多个属性转换为数值属性来应用于事件张量。 [20, 45, 62, 79, 80] 以多方面记录流作为输入。MemStream [21] 能够学习动态变化的趋势，以应对称为概念漂移 [27, 35, 57]的时变数据分布。尽管这些方法能够检测多个异常，但无法识别异常类型，也无法捕获动态多方面模式。

总之，现有方法都未专门关注高阶张量流中的动态多方面模式建模、摘要和异常检测。

3 所提出的模型

本节将介绍我们提出的模型。

3.1 CubeScope 的设计理念

本文使用的符号见附录A。本文研究的场景是复杂时间戳事件流。我们持续监测一条事件条目，其中包含 M个分类属性和一个时间戳。在最近的时间 T，我们拥有一组事件，每个属性有 $U_1 \dots U_M$个唯一单元和 T个时间戳。

定义1（事件张量流）。设 $\mathbf{X} \in \mathbb{N}^{U_1 \times \dots \times U_M \times T}$是截至当前时间点$T$的一个 M+1 阶张量流。每到一个时间点T，且与前一时间点之间的间隔不重叠 $\tau \ll T$，我们就可以得到当前张量 $\mathbf{X}^C \in \mathbb{N}^{U_1 \times \dots \times U_M \times \tau}$，它是 $\mathbf{X}$ 的部分张量。$\mathbf{X}$ 的元素 $x_{u_1 \dots u_M, t}$表示在时间刻度 t，每个属性中第 u_1 至第 u_M 个单位对应的事件条目总数。

图1(d)展示了纽约市出租车行程的事件张量流，其中每个事件的形式为 *(上车地点ID, 下车地点ID, 时间)*, $M = 2$。这里，我们向读者介绍三个重要观察。

观察 1（高维）。该张量的每个属性都包含大量单位，例如，接送地点属性中的 $U_1 = 262$ 和 $U_2 = 264$ 。

观察 2（稀疏）。图中，大多数属性对的序列都非常稀疏，这使得所有典型的时间序列分析工具都无法发挥作用，因为这些序列看起来像噪声（例如，$\{0, 0, 0, 1, 0, 0, 1, 2, 0, \cdots\}$）。

观察 3（半无限）。事件张量流随时间演变，并以无界数据流的形式到达，因此无法存储所有历史数据。

因此，我们旨在对事件张量流进行摘要，并获得简洁描述。具体而言，我们关注两类模式：（**P1**）组件（即潜在群组及其

关系）和（**P2**）状态（即随时间演变的不同模式）。那么，能够同时刻画（**P1**）和（**P2**）的最简单数学模型是什么？我们该如何表述摘要问题？下面给出答案。

3.2 提出的解决方案：CubeScope

下面我们将详细介绍我们的模型。首先，我们通过引入**多方面成分分解**来描述每个当前张量 $\mathbf{X}^C$ 中的（*P1*）组件，随后提出用于表示（*P2*）状态和整个张量流的**紧凑描述**。最后，在数据压缩范式下，我们将问题形式化为最小化编码成本。

3.2.1 多方面成分分解（*P1*）。我们从最简单的情况开始，即只有一个当前张量 $\mathbf{X}^C$。第一步是将高维稀疏张量 $\mathbf{X}^C$ 描述为一个紧凑且易于解释的模型。因此，我们提出一种新的分解方法，用于建模事件的生成过程。在我们的模型中，我们假设事件集合背后存在 K 个主要趋势/组件。具体而言，第 k 个组件由关于 M 属性和时间的概率分布刻画，定义如下：

- $\mathbf{A}^{(m)}_k \in \mathbb{R}^{U_m}$：组件 k 的属性 m 上 U_m 个单位的概率分布。
- $\mathbf{B}_t \in \mathbb{R}^K$：时间 t 上各组件 K 的概率分布。

这里，我们将 $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(M)}$ 和 $\mathbf{B}$ 称为分量矩阵。由于我们将每个属性视为类别型，因此可以使用狄利克雷先验 [22] 来描述这些分量矩阵：

$$\mathbf{A}^{(m)}_k \sim \mathrm{Dirichlet}(\alpha^{(m)}),\ \mathbf{B}_t \sim \mathrm{Dirichlet}(\beta),$$

其中，$\alpha^{(m)}$ 和 β 是超参数[1]。

我们还将时间依赖关系纳入此模型，使每个分量矩阵都能捕捉数据流中其前序项的上下文。我们假设各组件的均值与前一时刻 $T-\tau$ 的均值相同，除非新到达的事件 $\mathbf{X}^C$ 已得到确认。基于这一假设，我们可以使用以下狄利克雷先验：$\mathrm{Dirichlet}(\alpha^{(m)}_1 \hat{\mathbf{A}}^{(m)}_k)$ 和 $\mathrm{Dirichlet}(\beta_1 \hat{\mathbf{B}}_t)$，其中 ${}_l\hat{\mathbf{A}}^{(m)}$ 和 ${}_l\hat{\mathbf{B}}$ 是时间 $T-l\tau$ 上的前一组分量矩阵。为了捕捉长期依赖关系，我们可以扩展此方法，使其依赖于过去的 L 个矩阵。

$$\mathbf{A}^{(m)}_k \sim \mathrm{Dirichlet}(\Sigma^L_{l=1}\alpha^{(m)}_l \hat{\mathbf{A}}^{(m)}_k),\ \mathbf{B}_t \sim \mathrm{Dirichlet}(\Sigma^L_{l=1}\beta_l \hat{\mathbf{B}}_t).$$

因此，生成过程可描述如下：

- 对于每个组件 $k = 1, \ldots, K$：
  - 对于每个属性 $m = 1, \ldots, M$：
    - $\mathbf{A}^{(m)}_k \sim$ 狄利克雷分布$(\Sigma^L_{l=1}\alpha^{(m)}_l \hat{\mathbf{A}}^{(m)}_k)$
- 对于每个时间 $t = 1, \ldots, \tau$：
  - $\mathbf{B}_t \sim$ 狄利克雷分布$(\Sigma^L_{l=1}\beta_l \hat{\mathbf{B}}_t)$
  - 对于每个条目 $j = 1, \ldots, N_t$：
    - $z_{t,j} \sim$ 多项分布$(\mathbf{B}_t)$ // 抽取一个潜在分量 $z_{t,j}$
    - 对于每个属性 $m = 1, \ldots, M$：
      - $e_{(m)t,j} \sim$ 多项分布$(\mathbf{A}^{(m)}_{z_{t,j}})$，// 在每个属性中抽取一个单位

其中，N_t 表示时间 t 的事件总数，$z_{t,j}$ 表示潜在分量。每个事件 $e_{t,j}$ 均从特定于分量的多项式分布中采样。我们指出，该模型有三方面的优势。首先，尽管事件张量中的活动较为稀疏，我们的模型仍能消除许多冗余（例如噪声），并将一组事件概括为 K 个组件。其次，事件条目由 $M+1$ 个分量矩阵生成，因此能够处理任意阶张量。第三，为捕捉时间依赖关系，该模型使用过去的 L 个分量矩阵，而非存储张量。

3.2.2 紧凑描述（*P2*）。尽管分量矩阵能够简洁地描述部分张量 $\mathbf{X}^C$，但对于包含各种不同类型动态模式的整个张量流 $\mathbf{X}$ 而言，这还不够。因此，我们引入了另一个高层架构。

定义 2（状态：θ）。令 θ 为一个由以下分量矩阵组成的状态：$\theta = \{\{\mathbf{A}^{(m)}\}^M_{m=1}, \mathbf{B}\}$，用于描述某种不同的动态模式；借此，我们可以将整个张量流划分并概括为多个片段。当存在 R 个状态时，状态集合定义为 $\Theta = \{\theta_r\}^R_{r=1}$。

此外，当存在 G 个切换位置时，状态分配定义为 $\mathbf{S} = \{s_g\}^G_{g=1}$，其中 $s_g = (t_s, r)$ 表示从每个切换位置 t_s 到第 r 个状态的历史。最后，我们采用上述所有部分来对 $\mathbf{X}$ 进行紧凑描述。

定义 3（紧凑描述）。令 $\mathbf{C} = \{R, \Theta, G, \mathbf{S}\}$ 表示整个张量流的紧凑表示，即：

- 状态的数量 R 和状态集合，$\Theta = \{\theta_r\}^R_{r=1}$，
- = 片段的数量 G 以及分配方案，$\mathbf{S}\ \{s_g\}^G_{g=1}$。

3.2.3 问题形式化。我们的最终目标是对该问题进行形式化，将所有数据流 $\mathbf{X}$ 摘要为一个紧凑表示 $\mathbf{C}$。我们的目标函数运用了最小描述长度（MDL）原则 [33]。简而言之，它遵循这样的假设：我们对数据压缩得越多，就越能了解其底层模式。具体而言，我们评估总编码成本，该成本可用于对原始张量流 $\mathbf{X}$ 进行无损压缩。摘要问题表述如下：

问题1。给定一个完整的事件流$\mathcal{X}$，求得紧凑描述C，使总编码成本最小

$$< \mathcal{X}; C > = < C > + < \mathcal{X}|C >, \tag{1}$$

其中 $< C >$ 是模型C的模型编码成本，而 $< \mathcal{X}|C >$ 是给定模型C时的数据编码成本。

模型编码成本。模型编码成本是描述模型所需的比特数。在我们的模型中，潜在分量的维度需要 $< d > = \Sigma^M_{m=1} \log^*(U_m) + \log^*(\tau) + \log^*(K)$ [2]。状态的数量需要 $< R > = \log^*(R)$。每个状态的模型编码成本 θ 由以下几项组成，

$$< \theta > = \sum_{m=1}^{M} < \mathbf{A}^{(m)} > + < \mathbf{B} >, \tag{2}$$

$$< \mathbf{A}^{(m)} > = |\mathbf{A}^{(m)}| \cdot (\log(K) + \log(U_m - 1) + c^F) + \log^*(|\mathbf{A}^{(m)}|), \tag{3}$$

$$< \mathbf{B} > = |\mathbf{B}| \cdot (\log(\tau) + \log(K-1) + c^F) + \log^*(|\mathbf{B}|), \tag{4}$$

其中 $|\cdot|$ 表示每个矩阵中的非零元素数量，而 c^F 是浮点运算成本[3]。数量

[1] 我们将 $\alpha^{(m)} = \beta = 1/K$ 设为默认值。

[2] 这里，$\log^*$ 是整数的通用编码长度。[3] 按照 [66, 82]，我们将8比特设为默认值。

片段数需要 $< G >= \log^*(G)$。每个移位点需要 $< s_g >= \log^*(t_s) + \log(R)$。

数据编码成本。 给定一个完整的状态集合 Θ，我们可以基于霍夫曼编码 [23]，对数据 X 进行编码，也就是说，为 X 中的每个值分配一定数量的比特。给定 θ 时，X 的数据编码成本计算如下：$< \mathrm{X}|\theta >= -\log P(\mathrm{X}|\{\mathbf{A}^{(m)}\}_{m=1}^{M}, \mathbf{B})$。因此，给定 C 时，X 的数据编码成本计算如下：

$$< \mathcal{X}|C >= \sum_{r=1}^{R} -\log P(\mathcal{X}[r]|\theta_r), \tag{5}$$

其中，$\mathrm{X}[r]$ 是由第 r 个状态分配的一组部分张量。最后，总编码成本 $< \mathrm{X}; \mathrm{C} >$ 表示如下：

$$\begin{aligned} < \mathcal{X}; C > &=< C > + < \mathcal{X}|C > \\ &=< d > + < R > + < G > \\ &+ \sum_{r=1}^{R} < \theta > + \sum_{g=1}^{G} < s_g > + < \mathcal{X}|C > . \end{aligned} \tag{6}$$

4 流式算法

到目前为止，我们已经介绍了如何表示两个概念（即组件和状态），并在无损压缩的背景下表述摘要问题（即 Problem1）。接下来，我们的目标是在流式场景中解决该问题。在本节中，我们旨在明确如何将整个事件流增量式地概括为紧凑描述 C，以及如何利用该紧凑描述进行流式异常检测。

为解决这些问题，我们现在提出一种流式算法 CubeScope，它由两个子算法 C-Decomposer 和 C-Compressor 组成。算法 1（见附录 B）展示了整体流程，图 2 说明了所提出算法的工作方式。直观而言，我们的算法会持续根据非重叠到达张量 X^C 生成状态 θ_c。随后，它使用 θ_c 更新紧凑描述 C，并衡量 X^C 的异常度。接下来，我们将详细介绍每个子算法。

4.1 C-Decomposer

我们首先旨在对 X^C 进行增量监控，并估计一个候选状态 θ_c（即 $\{\mathbf{A}^{(m)}\}_{m=1}^{M}, \mathbf{B}$），以最佳描述 X^C。根据第 3.2.1 节中的生成过程，我们提出一种基于折叠吉布斯采样的高效估计方法 [73]。具体而言，对于 X^C 中的每个非零条目 $x_{u_1,\ldots,u_M,t}$，我们以概率 p 抽取潜在分量 $z_{u_1,\ldots,u_M,t}$：

$$\begin{aligned} &p(z_{u_1,\ldots,u_M,t} = k \mid \mathcal{X}^C, \mathbf{B}', \hat{\mathbf{B}}, \beta, \{\mathbf{A}^{(m)'}, \hat{\mathbf{A}}^{(m)}, \alpha^{(m)}\}_{m=1}^{M}) \\ &\propto \frac{b'_{t,k} + \sum_{l=1}^{L} \beta_l \hat{b}_{t,k}}{\sum_{k=1}^{K} b'_{t,k} + L\beta} \cdot \prod_{m=1}^{M} \frac{a^{(m)'}_{k,u_m} + \sum_{l=1}^{L} \alpha^{(m)}{}_l \hat{a}^{(m)}_{k,u_m}}{\sum_{u=1}^{U_m} a^{(m)'}_{k,u_m} + L\alpha^{(m)}}, \end{aligned} \tag{7}$$

其中，$a^{(m)}{}_{k,\,u_m}$ 和 $b_{t,\,k}$ 分别是分配给第 u_m 个单位和时间刻度 t 的该组件 k 的总计数。请注意，撇号（例如 $b_{t,\,k}$）表示排除条目 $x_{u_1,\ldots,u_M,\,t}$ 后得到的计数。采样器预热后，我们可以得到 X^C 的分量矩阵，即 $\{\tilde{\mathbf{A}}^{(m)}\}_{m=1}^{M}$ 和 $\tilde{\mathbf{B}}$</style}，如下所示：

$$\tilde{\mathbf{A}}^{(m)}_{k,u_m} \propto \frac{a^{(m)}_{k,u_m} + \sum_{l=1}^{L} \alpha^{(m)}{}_l \hat{a}^{(m)}_{k,u_m}}{\sum_{u=1}^{U_m} a^{(m)}_{k,u_m} + L\alpha^{(m)}}, \tilde{\mathbf{B}}_{t,k} \propto \frac{b_{t,k} + \sum_{l=1}^{L} \beta_l \hat{b}_{t,k}}{\sum_{k=1}^{K} b_{t,k} + L\beta}. \tag{8}$$

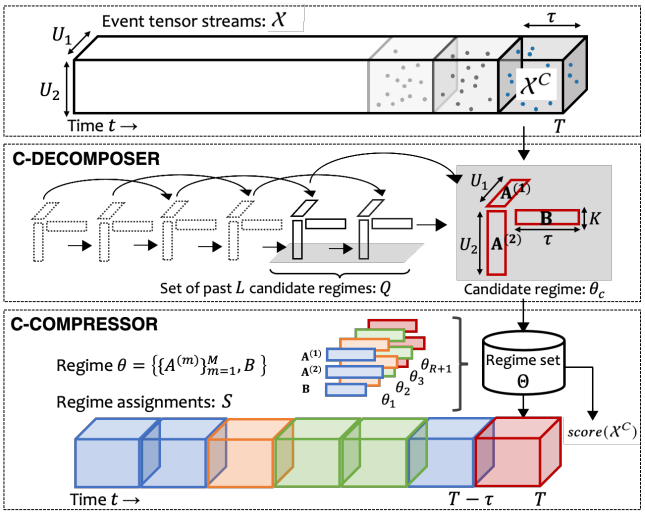


图 2：CubeScope 算法示意图。C-Decomposer：给定当前张量 $\mathcal{X}^C \in \mathbb{N}^{U_1 \times U_2 \times \tau}$，它首先根据过去的状态（本图中为 $L = 2$）将 $\mathcal{X}^C$ 分解为一个候选状态 θ_c。C-Compressor：它从候选状态 θ_c 和状态 $\{\theta_r\}_{r=1}^{R} (= \Theta)$ 中，为 $\mathcal{X}^C$ 分配最优状态。如果分配的是候选状态，CubeScope 就会将其作为 θ_{R+1} 插入状态集合 Θ 中。它还会计算 $\mathcal{X}^C$ 的异常度分数。

算法 2（见附录 B）详细展示了 C-Decomposer。它首先利用公式 (7)，为 X^C 中的每个条目 $x_{u_1,\ldots,u_M,\,t}$ 分配潜在分量 $z_{u_1,\ldots,u_M,\,t}$。确定潜在分量后，我们可以直接利用公式 (8) 计算目标矩阵。这里，我们将历史 L 个分量矩阵保存在历史参数集中，即大小为 L 的先进先出队列 Q。计算出当前分量矩阵后，移除最旧的矩阵，并将估计矩阵插入其中 Q。

尽管我们的张量数据是高维的（观察 1），C-Decomposer 并不依赖维度；也就是说，它的运行时间为 $O(N)$，其中 N 是事件数量（详情请参见引理 4.1）。相比之下，传统的张量算法（如交替最小二乘法（ALS）[50]）的运行时间会随所有属性而变化，即为 $O(\prod_{m=1}^{M} U_m)$；对于高阶张量，这可能会导致计算开销非常大（$M \geq 4$）。

4.2 C-Compressor

接下来，我们分步介绍 C-Compressor。算法 3（见附录 B）展示了整体流程。得到候选状态 θ_c 后，我们的下一个目标是为整个张量流 X 找到紧凑描述 C。然而，X 是半无限的（观察 3），因此我们无法处理所有历史数据。为了高效地找到紧凑描述 C，我们采用一种基于插入的算法，该算法会为 X 维护合理的描述，并在必要时生成一个状态。具体而言，该算法仅跟踪两个状态：先前状态 θ_p 和候选状态 θ_c。给定包含这两个状态的当前张量 X^C，算法会根据方程（6）比较每个状态对应的额外成本，然后选择使附加成本最小的下一步流程：

- If C-Compressor uses θ_p, it stays in the previous regime.

• 如果选择 θ_c，C-Compressor 会先在 Θ 中寻找一个更合适的状态，以避免重复。然后，采用成本最低的状态。

此处，附加成本 $<\mathrm{X}^C;\theta_*>$写作如下：

$$<\mathrm{X}^C;\theta_*> = \Delta <\mathrm{C}> + <\mathrm{X}^C|\theta_*>, \tag{9}$$

$$\Delta <\mathrm{C}> = \text{对数}^*(R+1) - \text{对数}^*(R) + <\theta_*> + \log^*(G+1) - \log^*(G) + <s>, \tag{10}$$

其中 θ_* 表示任意状态。如果我们需要移动另一个现有状态来表示 X^C，那么 $\Delta <\mathrm{C}> = \log^*(G+1) - \log^*(G) + <s>$；如果描述 X^C 需要新状态，则成本为公式 (10) 中所有项的总和；否则，$\Delta <\mathrm{C}> = 0$。

在线机制更新。 每当选中一个现有状态时，就将 θ_c 加到该状态中的每个计数上：

$$\tilde{\mathrm{A}}^{(m)}_{k,u_m} \leftarrow \frac{a^{(m)}_{k,u_m} + \sum_{l=1}^{L} \alpha^{(m)}{}_l \hat{a}^{(m)}_{k,u_m} + {}_c a^{(m)}_{k,u_m}}{\sum_{u=1}^{U_m} a^{(m)}_{k,u_m} + L\alpha^{(m)} + \sum_{u=1}^{U_m} {}_c a^{(m)}_{k,u_m}}, \tag{11}$$

其中 c（例如，${}_c a^{(m)}{}_{k,u_m}$）是候选状态中的一个计数。$\tilde{\mathbf{B}}_{t,k}$ 与之类似，为简洁起见，此处省略。随着现有状态更新，候选状态的影响会逐渐衰减；换言之，每个状态都会随着更新而收敛。

异常检测。 最后，我们利用紧凑描述 C 进行异常检测。基于压缩的技术天然适用于异常检测和罕见实例检测。在给定的紧凑描述 C 中，高频状态能够以较短的码长压缩绝大多数过去的数据。换言之，它表示数据流中的常态，因此对罕见实例需要较长的码长。因此，我们将 X^C 的编码成本视为其异常度分数；压缩成本越高，它"由不同机制生成、因而值得怀疑"的可能性就越大 [38]。

$$norm = \arg\max_{r\in R} |\mathcal{S}_r^{-1}|,$$

$$score(\mathcal{X}^C) = <\mathcal{X}^C|\theta_{norm}>, \tag{12}$$

其中，$|\mathcal{S}_r^{-1}|$ 表示状态 θ_r的总分段长度。请注意，在数据流中，正常状态的概念会随时间变化，这称为概念漂移 [21]。随着概念漂移，这种方法可以自适应地调整标准，以判断输入的张量。

引理 4.1（CubeScope 的时间复杂度）。*CubeScope* 的时间复杂度下界为 $O(N)$，上界为 $O(N+R)$，按进程计，其中 R为状态数量，N为$\mathcal{X}^C$ 中的事件条目总数（即 $\sum_{u_1}\cdots\sum_{u_M}\sum_t x_{u_1,\ldots,u_M,t}$）。

证明。请参见附录 B。 □

5 实验

本节评估 CubeScope 的性能。我们通过实验回答以下问题。

（问题一）有效性：CubeScope 能否成功地从给定的张量流中发现紧凑描述 C？（问题二）准确性：它在建模、聚类和流式异常检测方面的表现有多准确？

（Q3）可扩展性：在计算时间方面，它的扩展能力如何？**数据集与实验设置**。我们使用了八个真实数据集和四组合成数据。真实数据集包含本地出行、电子商务和网络流量/入侵的事件张量流

表 2：数据集描述

数据集	条目的形式	顺序
本地出行：乘车信息属性与时间戳 → #rides		
#1*NYC-Taxi* [8]	(接送地点*ID*，时间)	3
#2共享单车 [2]	(用户年龄，起始/终点站*ID*，时间)	4
电子商务：购买信息属性和时间戳 → #purchases		
#3珠宝 [4]	(价格，品牌，宝石，配饰类型，时间)	4
#4电子产品 [3]	(品牌、商品类别、时间)	3
网络流量/入侵：访问详情属性和时间戳 →#访问记录		
#5空军 [5]	(协议类型、服务、标志、土地、持续时间 源/目的字节数、错误分片、紧急标志、时间)	10
#6外部 [1]	（协议、源/目的 *IP* 地址、源/目的端口、标志、持续时间、数据包、字节、时间）	10
#7*OpenStack* [1]	"	10
#8京都 [9]	（源/目的字节数、计数、相同服务/*Serror*/服务 *Serror* 率、目的主机 *Serror* 率/相同源端口率/服务 *Serrors* 率，目的主机计数/服务计数、持续时间、服务、标志、时间）	15

，并在表 2 中进行了汇总。合成数据和实验设置详见附录 C.1。

基线方法。我们的实验采用十二种基线方法进行评估。概率生成模型：

• 潜在狄利克雷分配（LDA） [22] - 一种经典主题模型，其中主题分布为多项分布。
• 神经主题模型（NTM） [85] - 一种基于神经变分推断的主题模型。

• TriMine [68] - 一种用于高阶张量的分解方法，其条目由多个属性和一个时间戳构成。

时间序列、张量和数据流的聚类方法：

• K 均值 - 使用欧几里得距离的标准 K 均值聚类算法。

• TICC [37] - 一种用于多变量时间序列的聚类方法，每个簇都由一个相关网络来表征。
• CubeMarker [39] - 一种用于发现张量时间序列中不同模式的离线方法。
• 时间感知 LSTM (T-LSTM) [17] - 一种用于具有不规则时间间隔的序列的时间序列聚类方法。

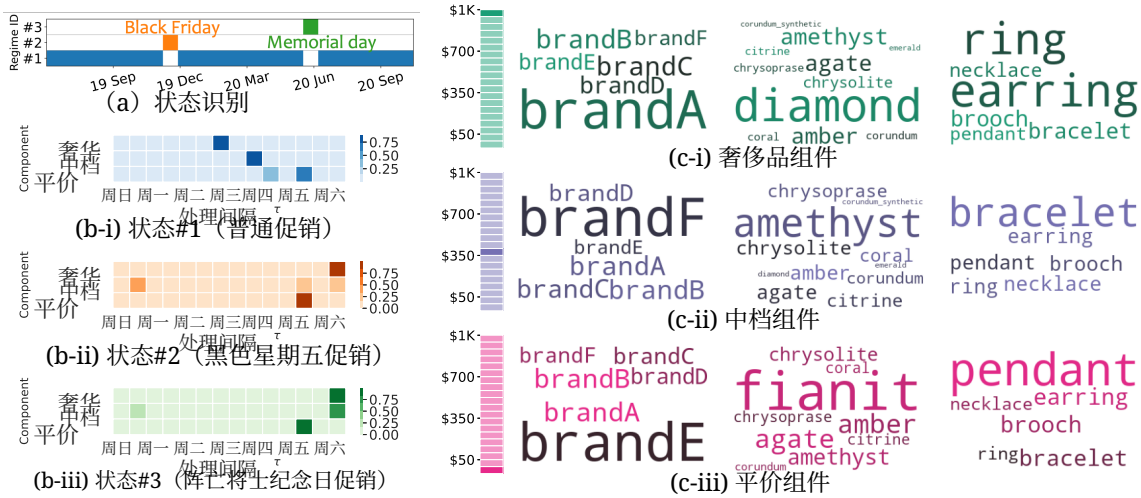
• DBSTREAM [36] - 一种用于演化数据流的聚类算法，可增量更新簇的密度。无监督异常检测方法：

• 局部离群因子 (LOF) [24] - 一种基于密度的方法，用于处理一组数据点。
• 孤立森林 (iForest) [56] - 一种离线方法，通过由数据点的随机切分构成的森林来隔离离群点。
• 鲁棒随机切分森林 (RRCF) [34] - 一种基于树的方法，专为流式数据设计。
• MemStream [21] - 一种流式方法，使用去噪自编码器和记忆模块。

5.1 问题一：有效性

我们首先展示 CubeScope 在真实数据集上的效果。另请参见附录 C.2 中电子产品 的数据结果。**本地出行**。 关于*NYC-Taxi* 的结果已在图1中展示。如前所见，CubeScope 能识别多个状态及其变换点（图1 (a)），并捕捉潜在分量（图1 (b)(c)）。这些模式反映了复杂的社会状况，有助于我们理解人类活动。

在线营销分析。 图3展示了我们对珠宝 数据集的挖掘结果。该数据集是从一家匿名珠宝店收集的电子商务日志



（a）状态识别

(b-i) 状态#1（普通促销）

(b-ii) 状态#2（黑色星期五促销）

(b-iii) 状态#3（阵亡将士纪念日促销）

(c-i) 奢侈品组件

(c-ii) 中档组件

(c-iii) 平价组件

图3：CubeScope 对珠宝数据集的市场分析：(a) CubeScope 自适应地捕捉了促销活动引起的购买行为变化。(b) 时间维度上的三个分量（奢华、中档、平价）。颜色越深，表示各分量与时间之间的关系越强。这展示了各分量何时吸引消费者兴趣。(c) 状态#1中各属性（价格/品牌/宝石/配饰类型）对应的三个分量。每列展示四个属性。价格排名中颜色越深、词云中文字越大，表示该属性与分量之间的关系越强。

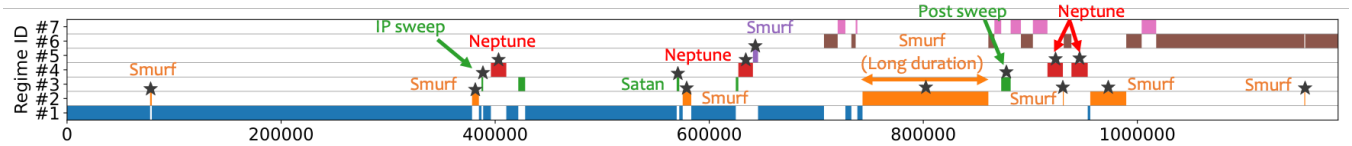


图4：CubeScope 对空军数据集进行实时入侵检测：星号表示入侵。它成功识别了持续时间各不相同的多种入侵类型（即状态#2：Smurf攻击，状态#3：探测攻击，状态#4：Neptune攻击）。

。每条日志由四种属性组成，分别是20种价格、6个匿名品牌、32种宝石和8种配饰类型，时间戳间隔为12小时。价格属性按每50美元划分，最高至1000美元，即共20个阶段。

- 状态识别：如图3(a)所示，CubeScope 生成状态#1 并开始监控张量流。十一月下旬，它检测到状态转变并生成状态#2。同样，五月下旬，它生成了新的状态#3。这些时段与黑色星期五 [4] 和阵亡将士纪念日 [5]相吻合。这表明我们的方法能够捕捉促销活动引起的购买行为变化。

- 多方面成分分析： 图3 (b)(c)展示了三个组件，我们将其分别手动命名为“平价”“中档”和“奢华”。首先，图3 (b)展示了每个状态中时间属性（即B）对应的三个组件。它说明了每个组件何时吸引消费者的兴趣。图3 (c)展示了状态#1中每个属性（即 $\{\mathbf{A}^{(m)}\}_{m=1}^{4}$）对应的三个组件。这些组件揭示了四个属性（即价格、品牌、宝石和配饰类型）中的潜在群组。结果表明，每个组件（行）都与不同的品牌或配饰密切相关。

在此，我们向读者介绍一些应用场景。对于定向广告和促销策略，分析人员会研究包含数百万、数十亿甚至数万亿个 [74] 事件的购买日志。

然而，这种方法需要专业知识和时间资源。由于 CubeScope 能够自动、高效地将海量数据概括为少数几个组件，因此可以为分析人员提供市场或用户偏好的摘要。此外，分析流行风潮和促销导致购买行为发生了怎样的变化，也是一个更为关键的问题。我们的方法将动态变化中的变化识别为状态转变，并自适应地为每种状态生成摘要。

网络安全。我们展示了CubeScope的实时入侵检测结果。图4展示了空军 数据集的结果，该数据集包含在军事网络环境中模拟的多种入侵，共计120万条记录。我们调查了检测到的状态，发现其中大多数都对应于实际入侵。例如，状态#2（橙色）和状态#4（红色）分别对应Smurf攻击和Neptune攻击。状态#3（绿色）捕获了IP扫描/Stan/端口扫描。这些入侵被归类为探测攻击 [77]。最重要的是，这些异常会随时间出现，因此其数量、持续时间和特征事先都是未知的，而CubeScope则完全自动化。它能够在流式场景中自动识别异常及其类型，同时更新每种异常类型的信息。我们还在第5.2节中从聚类和异常检测的角度对这一结果进行了定量分析。

5.2 问题二：准确性

接下来，我们从建模、聚类和异常检测方面评估 CubeScope 的准确性。

[4]黑色星期五是十一月第四个星期五的大型促销活动。[5]阵亡将士纪念日当天，大多数珠宝店都会举行特别促销活动。

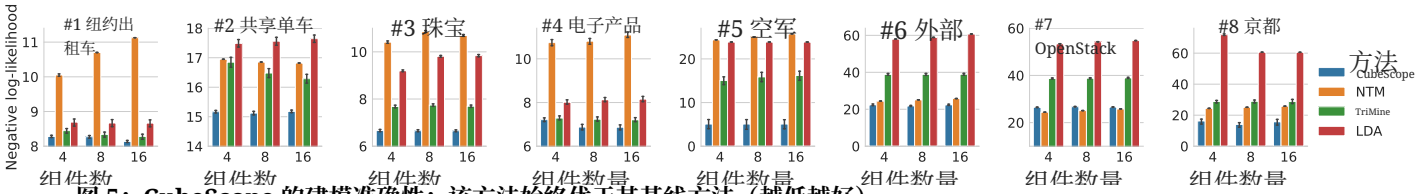


图 5：CubeScope 的建模准确性：该方法始终优于其基线方法（越低越好）。

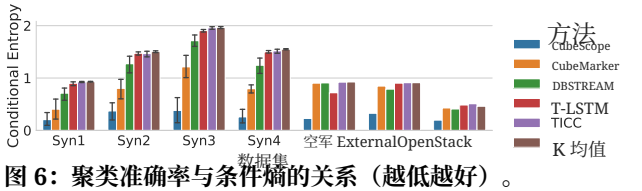


图 6：聚类准确率与条件熵的关系（越低越好）。

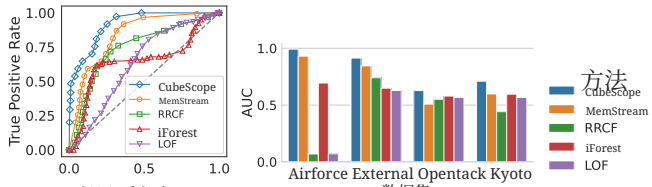


图 7：检测准确率（越高越好）：CubeScope 始终表现最佳。（左）外部 数据集的 ROC曲线。（右）所有数据集的 ROC-AUC。

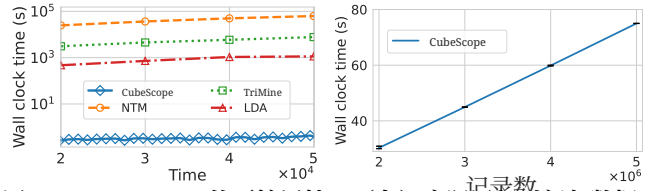


图 8：CubeScope 的可扩展性：（左）实际运行时间与数据流长度的关系。无论何时，CubeScope 都胜过竞争方法，其速度最高可达基线方法的 312,000 倍。（右）平均实际运行时间与进程中记录数的关系。该算法呈线性扩展。

建模。 我们比较了 CubeScope 与概率生成模型的建模准确性。图 5 展示了对于每种模型，长度为 168 的每个当前张量 $\mathcal{X}^C$ 的平均负对数似然。为进行公平比较，我们为所有模型使用 4、8 和 16 个成分/主题。数值越低，表示模型构建效果越好。不出所料，CubeScope 在所有数据集上都实现了较高的建模准确性，因为它能够捕获高阶张量流。由于 LDA 和 NTM 将张量视为大型矩阵，因此无法捕获多方面特征。TriMine 专为高维稀疏张量设计，但无法捕获包含各种随时间演变的模式（即状态）的张量流。

聚类。 接下来，我们展示 CubeScope 识别状态的准确度。我们同时使用带标签的数据集和合成数据，因为仅评估包含某些不会重复出现的时变模式（即某些聚类只出现一次）的真实数据集是不够的。我们生成了四种类型的合成数据，如下所示 [37]（详情见附录 C.1），并对它们进行了十次评估，报告均值和标准差。最后，我们将基于预测状态标签混淆矩阵（CM）计算的标准条件熵（CE）度量，与真实聚类标签进行比较。CE 得分根据以下公式衡量两个聚类之间的差异：$CE = -\Sigma_{i,\ j} \frac{CM_{i,\ j}}{\Sigma_{i\ j} CM_{i,\ j}} \log \frac{CM_{i,\ j}}{\Sigma_j CM_{i,\ j}}$。请注意，理想的混淆矩阵应为对角矩阵，此时 $CE = 0$。图 6 将 CubeScope 与聚类方法进行了比较。我们的方法始终优于其他方法，因为它能够处理高维稀疏张量。TICC 未能捕捉稀疏序列。T-LSTM 专为稀疏序列设计，但无法处理高维张量。CubeMarker 能够捕捉张量时间序列，但无法处理稀疏张量。DBSTREAM 能够识别数据流中的聚类，但无法捕捉张量流中的多方面特征。

异常检测。 我们评估了四个包含真实异常的真实数据集上的异常检测性能。首先，我们计算 CubeScope 和无监督基线方法的异常分数，然后选取异常程度最高的 top-k k = 个时段（„）。接下来，我们计算各方法输出的真正率和假正率。图 7 展示了外部 数据集的 ROC曲线以及所有数据集的 ROC-AUC。数值越高，表示检测准确率越高。CubeScope 在每个数据集上都取得了较高的检测准确率，而其他方法无法很好地检测异常，因为只有我们的方法能够捕获动态多方面模式，并将其用于后续的异常检测。

5.3 Q3. 可扩展性

最后，我们评估了 CubeScope 处理大型张量流所需的计算时间。图 8 左侧展示了在大型*NYC-Taxi*数据集上进行实验的实际运行时间。得益于增量更新，我们的方法不受数据流长度的影响。事实上，我们的方法实现了恒定的计算时间，速度比基线方法快达五个数量级。图 8 右侧展示了输入张量大小变化时 C-Decomposer 的计算时间。由于 CubeScope 能够以 $O(N)$ 时间快速、高效地估计模型（如引理 4.1 所述），其复杂度随事件数量线性增长。

6 结论

本文聚焦于高阶事件张量流的动态摘要，并介绍了 CubeScope。它具备我们在引言中列出的所有理想特性；

- 高效：它能够增量捕获动态多方面模式，并将半无限集合的事件张量流摘要为可解释表示。
- 总体而言：我们在各种数据集上进行的实验表明，CubeScope 能够成功发现有意义的模式和异常，并且优于最先进的建模、聚类和异常检测方法。
- 可扩展：其计算时间是恒定的，不受输入数据长度以及每个属性的维度影响。

致谢

我们衷心感谢匿名审稿人在审稿过程中付出的时间和努力。本研究得到 JSPS KAKENHI 科学研究资助金 JP20H00585、JP21H03446、JP22K17896，NICT 03501，MIC/SCOPE JP192107004，JST-AIP JPMJCR21U4，以及 ERCA 环境研究与技术开发基金 JPMEERF20201R02 的支持。

参考文献

[1] [n.d.]。CIDDS（科堡入侵检测数据集）。https://www.hs-coburg.de/forschung/forschungsprojekte-oeffentlich/informationstechnologie/cidds-coburg-intrusion-detection-data-sets.html。[2] [n.d.]。Citi Bike行程历史数据。https://ride.citibikenyc.com/system-data。[3] [n.d.]。电子产品商店电子商务购买历史数据。https://www.kaggle.com/mkechinov/ecommerce-purchase-history-from-electronics-store。[4] [n.d.]。珠宝商店电子商务购买历史数据。https://www.kaggle.com/mkechinov/ecommerce-purchase-history-from-jewelry-store。[5] [n.d.]。KDD Cup 1999数据集。http://kdd.ics.uci.edu/databases/kddcup99/kddcup99. html。[6] [n.d.]。CubeScope。https://github.com/kotaNakm/CubeScope。[7] [n.d.]。River：Python中的在线机器学习。https://riverml.xyz/dev/api/cluster/DBSTREAM/。[8] [n.d.]。TLC行程记录数据。https://www1.nyc.gov/site/tlc/about/tlc-trip-record- data.page。[9] [n.d.]。来自京都大学蜜罐的流量数据。https://www.takakura. com/Kyoto_data/。[10] 2019。利用女性健康追踪移动应用的大规模数据预测妊娠。在WWW中。2999–3005。[11] 2020。https://www1.nyc.gov/assets/home/downloads/pdf/executive-orders/2020/eeo-100.pdf。[12] 2020。https://www.state.gov/wp-content/uploads/2020/03/2020-03-20-Notice-New-York-on-Pause-Order.pdf。[13] 2020。https://www.governor.ny.gov/news/governor-cuomo-announces-new-york-city-enter-phase-1-reopening-june-8-and-five-regions-enter。[14] Charu C. Aggarwal、Jiawei Han、Jianyong Wang 和 Philip S. Yu。2003年。演化数据流聚类框架。在VLDB中。81–92。[15] Leman Akoglu、Hanghang Tong 和 Danai Koutra。2015年。基于图的异常检测与描述：综述。《数据挖掘与知识发现》29，3（2015），626–688。[16] Miguel Araujo、Spiros Papadimitriou、Stephan Günnemann、Christos Faloutsos、Prithwish Basu、Ananthram Swami、Evangelos E Papalexakis 和 Danai Koutra。2014年。Com2：快速自动发现时间（“彗星”）社区。在亚太知识发现与数据挖掘会议中。Springer，271–283。[17] Inci M. Baytas、Cao Xiao、Xi Zhang、Fei Wang、Anil K. Jain 和 Jiayu Zhou。2017年。通过时间感知LSTM网络进行患者亚型划分。在KDD中。65–74。[18] Alex Beutel、Kenton Murray、Christos Faloutsos 和 Alexander J Smola。2014年。Cobafi：协同贝叶斯过滤。在WWW中。97–108。[19] Siddharth Bhatia、Arjit Jain、Pan Li、Ritesh Kumar 和 Bryan Hooi。2021年。MStream：多方面数据流中的快速异常检测。在WWW中。ACM / IW3C2，3371–3382。[20] Siddharth Bhatia、Arjit Jain、Pan Li、Ritesh Kumar 和 Bryan Hooi。2021年。MStream：多方面数据流中的快速异常检测。在WWW中。3371–3382。[21] Siddharth Bhatia、Arjit Jain、Shivin Srivastava、Kenji Kawaguchi 和 Bryan Hooi。2022年。MemStream：基于记忆的流式异常检测。在WWW '22：2022年ACM Web大会，线上活动，法国里昂，2022年4月25日至29日，Frédérique Laforest、Raphaël Troncy、Elena Simperl、Deepak Agarwal、Aristides Gionis、Ivan Herman 和 Lionel Médini（编）。ACM，610–621。[22] David M Blei、Andrew Y Ng 和 Michael I Jordan。2003年。潜在狄利克雷分配。《机器学习研究杂志》3（2003），993–1022。[23] Christian Böhm、Christos Faloutsos、Jia-Yu Pan 和 Claudia Plant。2007年。Ric：无参数抗噪聚类。TKDD 1，3（2007），10–es。[24] Markus M. Breunig、Hans-Peter Kriegel、Raymond T. Ng 和 Jörg Sander。2000年。LOF：识别基于密度的局部离群点。在2000年ACM SIGMOD数据管理国际会议（SIGMOD '00）论文集。计算机协会，纽约州纽约市，美国，93–104。[25] Varun Chandola、Arindam Banerjee 和 Vipin Kumar。2009年。异常检测：综述。《ACM计算调查》41，3，第15篇（2009年7月），58页。[26] Pudi Chen、Shenghua Liu、Chuan Shi、Bryan Hooi、Bai Wang 和 Xueqi Cheng。2018年。NeuCast：电网时间序列的季节性神经预测。在IJCAI中。3315–3321。[27] Lianhua Chi、Bin Li、Xingquan Zhu、Shirui Pan 和 Ling Chen。2017年。利用哈希实现带概念漂移的自适应实时图流分类。《IEEE控制论汇刊》48，5（2017），1591–1604。[28] Eunjoon Cho、Seth A. Myers 和 Jure Leskovec。2011。友谊与移动性：基于位置的社交网络中的用户移动。在 KDD。1082–1090。[29] Gianmarco De Francisci Morales、Albert Bifet、Latifur Khan、Joao Gama 和 Wei Fan。2016。物联网大数据流挖掘。在 KDD。2119–2120。[30] Shohreh Deldari、Daniel V. Smith、Hao Xue 和 Flora D. Salim。2021。利用自监督对比预测编码进行时间序列变点检测。在 WWW。ACM / IW3C2，3124–3135。[31] Hadi Fanaee-T 和 João Gama。2016。基于张量的异常检测：跨学科综述。《Knowledge-Based Systems》98（2016），130–147。[32] Shufeng Gong、Yanfeng Zhang 和 Ge Yu。2017。通过探索密度山的演化对数据流进行聚类。《VLDB Endowment论文集》11，4（2017），393–405。[33] Peter D Grünwald、In Jae Myung 和 Mark A Pitt。2005。最小描述长度进展：理论与应用。MIT出版社。[34] Sudipto Guha、Nina Mishra、Gourav Roy 和 Okke Schrijvers。2016。基于鲁棒随机切分森林的流异常检测。载于《第33届国际机器学习会议论文集》第48卷（ICML'16）。JMLR.org，2712–2721。[35] Manish Gupta、Jing Gao、Charu C Aggarwal 和 Jiawei Han。2013。时间数据的离群点检测：综述。《IEEE知识与数据工程汇刊》26，9（2013），2250–2267。[36] Michael Hahsler 和 Matthew Bolaños。2016。基于微簇间共享密度的数据流聚类。《IEEE知识与数据工程汇刊》28，6（2016），1449–1461。[37] David Hallac、Sagar Vare、Stephen Boyd 和 Jure Leskovec。2017。基于托普利兹逆协方差的多变量时间序列数据聚类。在 KDD。[38] Douglas M Hawkins。1980。离群点识别。第11卷。Springer。[39] Takato Honda、Yasuko Matsubara、Ryo Neyama、Mutsumi Abe 和 Yasushi Sakurai。2019。复杂传感器序列的多方面挖掘。在 ICDM。[40] Bryan Hooi、Shenghua Liu、Asim Smailagic 和 Christos Faloutsos。2017。BeatLex：利用模式对时间序列进行摘要和预测。在 PKDD，第10535卷。3–19。[41] Bryan Hooi、Kijung Shin、Shenghua Liu 和 Christos Faloutsos。2019。SMF：具有季节性模式的漂移感知矩阵分解。在 SIAM。621–629。[42] Wenjie Hu、Yang Yang、Ziqiang Cheng、Carl Yang 和 Xiang Ren。2021。利用演化状态图进行时间序列事件预测。载于第14届ACM国际网络搜索与数据挖掘会议论文集。580–588。[43] Tomoharu Iwata、Shinji Watanabe、Takeshi Yamada 和 Naonori Ueda。2009。用于分析消费者购买行为的主题跟踪模型。在 IJCAI。[44] Jun-Gi Jang 和 U Kang。2021。快速且节省内存的 Tucker 分解，用于回答多样化的时间范围查询。载于第27届ACM SIGKDD知识发现与数据挖掘会议论文集。725–735。[45] Meng Jiang、Alex Beutel、Peng Cui、Bryan Hooi、Shiqiang Yang 和 Christos Faloutsos。2015。用于多模态数据稠密块的通用可疑度度量。载于2015年IEEE国际数据挖掘会议。IEEE，781–786。[46] Harshavardhan Kamarthi、Lingkai Kong、Alexander Rodríguez、Chao Zhang 和 B Aditya Prakash。2022。CAMul：校准准确的多视图时间序列预测。载于2022年ACM Web大会论文集。3174–3185。[47] Koki Kawabata、Yasuko Matsubara、Takato Honda 和 Yasushi Sakurai。2020。张量流中社交活动的非线性挖掘。在 KDD。2093–2102。[48] Koki Kawabata、Yasuko Matsubara 和 Yasushi Sakurai。2019。数据流中的自动序列模式挖掘。在 CIKM。1733–1742。[49] Diederik P Kingma 和 Max Welling。2013。自编码变分贝叶斯。arXiv预印本 arXiv:1312.6114（2013）。[50] Tamara G Kolda 和 Brett W Bader。2009。张量分解及其应用。《SIAM评论》51，3（2009），455–500。[51] Danai Koutra、U Kang、Jilles Vreeken 和 Christos Faloutsos。2014。VOG：大型图的摘要与理解。载于2014年SIAM国际数据挖掘会议论文集。SIAM，91–99。[52] Mathias Kraus 和 Stefan Feuerriegel。2019。利用基于Wasserstein距离的序列匹配，对购物篮进行个性化购买预测。在 KDD。2643–2652。[53] Changhee Lee 和 Mihaela Van Der Schaar。2020。利用深度预测聚类进行疾病进展的时间表型分析。在 ICML。5767–5777。[54] Lei Li、B Aditya Prakash 和 Christos Faloutsos。2010。用于时间序列的简约线性指纹提取。技术报告。卡内基梅隆大学匹兹堡分校计算机科学学院。[55] Xiangsheng Li、Jiaxin Mao、Weizhi Ma、Yiqun Liu、Min Zhang、Shaoping Ma、Zhaowei Wang 和 Xiuqiang He。2021。主题增强知识感知检索模型，用于多样化相关性估计。在 WWW。756–767。[56] Fei Tony Liu 、 Kai Ming Ting 和 Zhi-Hua Zhou。2012。基于隔离的异常检测。《ACM数据知识发现汇刊》（TKDD）6，1（2012），1–39。[57] Jie Lu、Anjin Liu、Fan Dong、Feng Gu、Joao Gama 和 Guangquan Zhang。2018。概念漂移下的学习：综述。《IEEE知识与数据工程汇刊》31，12（2018），2346–2363。[58] Yue Lu、Renjie Wu、Abdullah Mueen、Maria A Zuluaga 和 Eamonn Keogh。2022。矩阵剖面XXIV：将时间序列异常检测扩展至数万亿数据点和超高速到达的数据流。载于第28届

ACM SIGKDD知识发现与数据挖掘会议。1173–1182。[59] Qianli Ma、Jiawei Zheng、Sen Li 和 Gary W Cottrell。2019。用于时间序列聚类的表示学习。《神经信息处理系统进展》32（2019），3781–3791。[60] Stratos Mansalis、Eirini Ntoutsi、Nikos Pelekis 和 Yannis Theodoridis。2018。数据流聚类算法评估。《统计分析与数据挖掘：ASA数据科学期刊》11，4（2018），167–187。[61] Emaad Manzoor、Hemank Lamba 和 Leman Akoglu。2018。xstream：特征演化数据流中的离群点检测。载于第24届ACM SIGKDD知识发现与数据挖掘国际会议论文集。1963–1972。[62] Hing-Hao Mao、Chung-Jung Wu、Evangelos E Papalexakis、Christos Faloutsos、Kuo-Chen Lee 和 Tien-Cheu Kao。2014。Malspot：多种恶意网络行为模式分析。载于亚太知识发现与数据挖掘会议。Springer，1–14。[63] Yasuko Matsubara 和 Yasushi Sakurai。2016。数据流中的状态转变：协同演化时间序列的实时预测。在 KDD。1045–1054。[64] Yasuko Matsubara 和 Yasushi Sakurai。2019。时变数据流的动态建模与预测。在 KDD。458–468。[65] Yasuko Matsubara、Yasushi Sakurai 和 Christos Faloutsos。2014。AutoPlait：协同演化时间序列的自动挖掘。在 SIGMOD。[66] Yasuko Matsubara、Yasushi Sakurai 和 Christos Faloutsos。2015。网络如丛林：用于协同演化在线活动的非线性动力系统。在 WWW。[67] Yasuko Matsubara、Yasushi Sakurai 和 Christos Faloutsos。2016。竞争性局部活动的非线性挖掘。在 WWW。[68] Yasuko Matsubara、Yasushi Sakurai、Christos Faloutsos、Tomoharu Iwata 和 Masatoshi Yoshikawa。2012。复杂带时间戳事件的快速挖掘与预测。在 KDD。271–279。[69] Charalampos Mavroforakis、Isabel Valera 和 Manuel Gomez-Rodriguez。2017。网络学习活动动态建模。在 WWW。ACM，1421–1430。[70] Yu Meng、Yunyi Zhang、Jiaxin Huang、Yu Zhang、Chao Zhang 和 Jiawei Han。2020。通过联合球形树与文本嵌入进行层次主题挖掘。在 KDD。1908–1917。[71] Maya Okawa、Tomoharu Iwata、Takeshi Kurashima、Yusuke Tanaka、Hiroyuki Toda 和 Naonori Ueda。2019。深度混合点过程：利用丰富上下文信息进行时空事件预测。在 KDD。373–383。[72] Fabian Pedregosa、Gaël Varoquaux、Alexandre Gramfort、Vincent Michel、Bertrand Thirion、Olivier Grisel、Mathieu Blondel、Peter Prettenhofer、Ron Weiss、Vincent Dubourg 等。2011。Scikit-learn：Python中的机器学习。《机器学习研究杂志》12（2011），2825–2830。[73] Ian Porteous、David Newman、Alexander Ihler、Arthur Asuncion、Padhraic Smyth 和 Max Welling。2008。用于潜在狄利克雷分配的快速折叠吉布斯采样。在 KDD。569–577。[74] Thanawin Rakthanmanon、Bilson Campana、Abdullah Mueen、Gustavo Batista、Brandon Westover、Qiang Zhu、Jesin Zakaria 和 Eamonn Keogh。2012。利用动态时间规整搜索和挖掘数万亿条时间序列子序列。载于第18届ACM SIGKDD知识发现与数据挖掘国际会议论文集。262–270。[75] Yasushi Sakurai、Yasuko Matsubara 和 Christos Faloutsos。2016。网络大规模时间序列数据挖掘。在 WWW。1029–1032。[76] Aaron Schein、John Paisley、David M Blei 和 Hanna Wallach。2015。贝叶斯泊松张量分解：从稀疏二元事件计数中推断多边关系。在 KDD。1045–1054。[77] Benedetto Marco Serinelli、Anastasija Collen 和 Niels Alexander Nijdam。2020。利用KDD Cup 1999和NSL-KDD数据集对Anidinr：基于异常的网络入侵检测系统进行训练指导。《计算机科学程序汇刊》175（2020），560–565。[78] Neil Shah、Danai Koutra、Tianmin Zou、Brian Gallagher 和 Christos Faloutsos。2015。TimeCrunch：可解释的动态图摘要。在 KDD。[79] Lei Shi、Aryya Gangopadhyay 和 Vandana P Janeja。2015。STenSr：用于异常检测和模式发现的时空张量流。《知识与信息系统》43，2（2015），333–353。[80] Kijung Shin、Bryan Hooi、Jisu Kim 和 Christos Faloutsos。2017。Densealert：张量流中的增量式稠密子张量检测。载于第23届ACM SIGKDD知识发现与数据挖掘国际会议论文集。1057–1066。[81] Koen Smets 和 Jilles Vreeken。2011。与众不同者：异常的识别与刻画。载于2011年SIAM国际数据挖掘会议论文集。SIAM，804–815。[82] Tsubasa Takahashi、Bryan Hooi 和 Christos Faloutsos。2017。Autocyclone：利用鲁棒张量分解自动挖掘周期性在线活动。在 WWW。213–221。[83] Veronica Tozzo、Federico Ciech、Davide Garbarino 和 Alessandro Verri。2021。统计模型耦合支持复杂局部多变量时间序列分析。在 KDD。1593–1603。[84] Yichen Wang、Robert Chen、Joydeep Ghosh、Joshua C Denny、Abel Kho、You Chen、Bradley A Malin 和 Jimeng Sun。2015。Rubik：知识引导的张量

用于健康数据分析的分解与补全。载于 *KDD*。1265–1274。[85] Yue Wang、Jing Li、Hou Pong Chan、Irwin King、Michael R. Lyu 和 Shuming Shi。2019年。面向社交媒体语言的主题感知神经关键词生成。载于 计算语言学协会。2516–2526。[86] Yuan Xue、Denny Zhou、Nan Du、Andrew M. Dai、Zhen Xu、Kun Zhang 和 Claire Cui。2020年。用于相关事件发生时间预测的深度状态空间生成模型。载于 *KDD*。ACM，1552–1562。[87] Jaewon Yang、Julian McAuley、Jure Leskovec、Paea LePendu 和 Nigam Shah。2014年。在随时间演化的事件序列中发现进展阶段。载于 *WWW*。783–794。[88] Shuochao Yao、Shaohan Hu、Yiran Zhao、Aston Zhang 和 Tarek F. Abdelzaher。2017年。DeepSense：用于时间序列移动感知数据处理的统一深度学习框架。载于 *WWW*。ACM，351–360。[89] Jianhua Yin、Daren Chao、Zhongkun Liu、Wei Zhang、Xiaohui Yu 和 Jianyong Wang。2018年。短文本流的基于模型的聚类。载于 第*24*届*ACM*知识发现与数据挖掘特别兴趣小组国际会议论文集。2634–2642。

表3：符号和定义。

符号	定义
M	复杂事件张量的属性数量
$U_1 \dots U_M$	每个属性中由若干唯一单元组成的集合
T	整个张量流的长度
τ	当前张量的长度
$\mathcal{X}$	整个事件张量流，$\mathbf{X} \in \mathbb{N}^{U_1 \times \cdots \times U_M \times T}$
$\mathcal{X}^C$	当前事件张量，$\mathcal{X}^C \in \mathbb{N}^{U_1 \times \cdots \times U_M \times \tau}$
K	潜在分量的数量
$\mathbf{A}^{(m)}$	第m个属性分量矩阵，$K \times U_m$
$\mathbf{B}$	时间分量矩阵，$\tau \times K$
Q	用于保留过去分量矩阵的先进先出队列
L	队列大小 Q
R	状态数量
θ_r	第r个状态参数，即 $\theta_r = \{\mathbf{A}^{(1)}, \dots\dots, \mathbf{A}^{(M)}, \mathbf{B}\}$
Θ	状态集合，即 $\Theta = \{\theta_1, \dots\dots, \theta_R\}$
G	状态分配的数量（即片段数）
s_g	时间 t_s 转变到第 r 个状态的轨迹，即 $s_g = (t_s, r)$
$\mathcal{S}$	状态分配，即 $\mathbf{S} = \{s_1, \dots\dots, s_G\}$
$\lvert\mathbf{S}_r^{-1}\rvert$	状态 θ_r的总分段长度
C	紧凑描述，即 $\mathbf{C} = \{R, \Theta, G, \mathbf{S}\}$
$score(\mathbf{X}^C)$	$\mathcal{X}^C$的异常度分数

算法1 CubeScope ($\mathcal{X}^C$, C Q)

输入： 1. 当前张量 $\mathcal{X}^C \in \mathbb{N}^{U_1 \times \dots \times U_M \times \tau}$
2. 先前候选解 $\mathbf{C} = \{R, \Theta, G, \mathbf{S}\}$
3. 先前的历史参数集 Q
输出： 1. 更新后的候选解 C'
2. 更新后的历史参数集 Q'
3. 异常度分数 $score(\mathbf{X}^C)$

θ_c, Q' = C-Decomposer（$\mathcal{X}^C, Q$）;
C', $score(\mathcal{X}^C)$ = C-Compressor (θ_c, $\mathcal{X}^C$, C);
返回 $C', Q', score(\mathcal{X}^C)$;

算法 2 C-Decomposer ($\mathbf{X}^C, Q$)

输入： 1. 当前张量 $\mathcal{X}^C \in \mathbb{N}^{U_1 \times \dots \times U_M \times \tau}$
2. 先前的历史参数集 Q
输出： 1. 当前模型参数集 $\theta_c = \{\mathbf{A}^{(1)}, \dots \mathbf{A}^{(M)}, \mathbf{B}\}$
2. 更新后的历史参数集 Q'

对每次迭代**执行**
对于非零元素 x，位于$\mathcal{X}^C$ **中执行**
对于条目 x **执行**
绘制隐变量 z；// 根据公式 (7)
结束循环
结束循环
</style>**结束循环**
计算 $\mathbf{A}^{(1)}$, ..., $\mathbf{A}^{(M)}$, $\mathbf{B}$; //根据公式 (8)
$\theta_c \leftarrow \mathbf{A}^{(1)}$, ..., $\mathbf{A}^{(M)}$, $\mathbf{B}$;
Q.deque; // 移除最旧的一组分量矩阵
$Q' \leftarrow Q$.enque(θ_c); // 插入一组当前估计矩阵 θ_c
返回 θ_c, Q';

算法 3 C-Compressor (θ_c, $\mathbf{X}^C$, $\mathbf{C}$)

输入： 1. 候选模型参数集 $\theta_c = \{\mathbf{A}^{(1)}, \dots \mathbf{A}^{(M)}, \mathbf{B}\}$
2. 新观测张量 $\mathcal{X}^C \in \mathbb{N}^{U_1 \times \dots \times U_M \times \tau}$
3. 先前紧凑描述 $\mathbf{C} = \{R, \Theta, G, \mathbf{S}\}$
输出： 1. 更新后的紧凑描述 $\mathbf{C}' = \{R', \Theta', G', \mathbf{S}'\}$
2. 异常度分数 $score(\mathbf{X}^C)$

/* 更新紧凑描述 C */
if $< \mathcal{X}^C; \theta_p >$ 小于 $< \mathcal{X}^C; \theta_c >$ **then**
/* 保持在先前状态 θ_p */
$\theta'_p \leftarrow$ 状态更新（θ_p, θ_c）; // 根据公式 (11)
否则
$\theta_e = \arg\min_{\theta \in \Theta} < \mathbf{X}^C; \theta >$;
如果 $< \mathcal{X}^C;\ \theta_c >$ 小于 $< \mathcal{X}^C;\ \theta_e >$ **则**
/* 转换到候选状态 θ_c */
$R' \leftarrow R + 1$; $\Theta' \leftarrow \Theta \cup \theta_c$;
$G' \leftarrow G + 1$; $\mathbf{S}' \leftarrow \mathbf{S} \cup (t,\ R + 1)$;
else
/* 转到现有状态 θ_e */
$\theta'_e \leftarrow$ 状态更新(θ_e, θ_c); // 根据公式 (11)
$G' \leftarrow G + 1$; $\mathbf{S}' \leftarrow \mathbf{S} \cup (t,\ e)$;
结束 if
结束 if
/* 计算异常度分数*/
$norm \leftarrow$ 参数最大值 $\max_{r \in R} \lvert\mathbf{S}_r^{-1}\rvert$;
$score(\mathcal{X}^C) \leftarrow < \mathcal{X}^C | \theta_{norm} >$;
返回 $C' = \{R',\ \Theta'\ G'\ \mathcal{S}'\},\ score(\mathcal{X}^C)$;

附录

A 所提出的模型

表3列出了本文使用的符号及其定义。所有对数均以2为底，按照惯例，我们使用 $0 \log 0 = 0$。

B 流式算法

算法1展示了 CubeScope 的整体流程，该流程由 C-Decomposer（算法 2）和 C-Compressor（算法. 3）组成。C-Decomposer 持续监控当前张量 $\mathbf{X}^C$并生成一个状态 θ_c。随后，C-Compressor 使用 θ_c更新紧凑描述 C，并测量 $\mathbf{X}^C$的异常度。

引理 4.1 的证明。

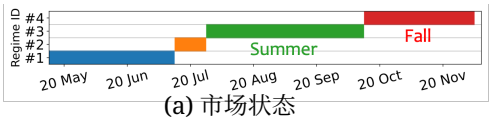
证明。对于每个时间点，CubeScope 首先运行 C-Decomposer，针对 $\mathbf{X}^C$中的每个非零元素条目 $x_{u,t}$抽取隐变量 $z_{u,t}$。此进程需要 $O(\#iter \cdot KN)$，其中 $\#iter$ 是抽取的迭代次数，K 是组件数量，N是 $\mathbf{X}^C$中的事件条目总数（即 $\sum_{u_1} \cdots \sum_{u_M} \sum_t x_{u_1,\dots,u_M,t}$ ）。由于 $\#iter$ 和 K 都是较小的常数值，因此可以忽略。由此，C-Decomposer 的复杂度为 $O(N)$。在 C-Compressor 中，它会跟踪 θ_c和 θ_p。如果它将先前状态 θ_p 用于当前张量 $\mathbf{X}^C$，就能快速更新状态，所需时间为 $O(1)$。否则，它会尝试在 Θ中寻找最优状态，所需时间为 $O(R)$。总的来说，CubeScope 需要运行这两个算法。因此，每个进程的复杂度下限为 $O(N)$，上限为 $O(N + R)$。 □

模型初始化。 当我们以状态集合 Θ and Q 作为历史参数集开始本轮迭代时，会从初始张量 $\mathbf{X}_s$ 中以间隔 τ均匀选取若干样本片段，然后分别估计每个片段的模型参数 θ_s。通过逐步增加模型参数 θ_s的数量并监控总编码成本，确定最合适的状态集合：

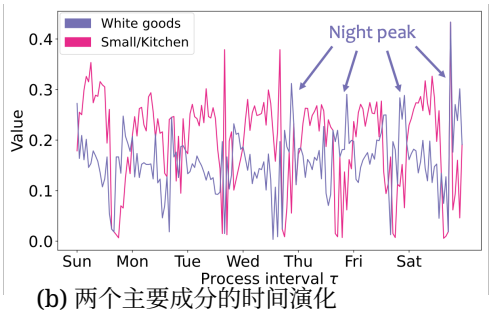
$$\Theta = \arg\min_{\Theta \in \Theta_s} < \mathcal{X}_s; \Theta >, \tag{13}$$

其中，$\Theta_s = \{\theta_{s1}, \theta_{s2}, \dots\dots\}$ 是根据每个样本估计出的状态集合。我们还将 L 设为变换点的平均间隔（即片段的平均长度）。

处理间隔 τ。参数 τ决定当前张量的大小以及状态的最小粒度。用户需要了解不同粒度下的状态（例如每日和每周的模式），因此通常根据应用选择 τ。C-Decomposer 的运行时间随当前张量中的记录数线性增长，而非随 τ的大小增长。尽管较大的 τ会使算法处理更大的当前张量，但由于我们假设张量流是稀疏的，因此对运行时间的影响应该很小。



(a) 市场状态



(b) 两个主要成分的时间演化



(c-i) 热门品牌（属于）白色家电（上方）和小型/厨房用品（底部）组件。



（c-ii）热门商品位于白色家电（顶部）和小型/厨房用品（底部）组件。



（d）热门商品的变化随状态转变

图9：电子产品领域的客户行为建模：（a）CubeScope 共发现四种状态，反映了客户行为的季节性。（b）两个组件呈现明显的日周期性；白色家电组件在夜间出现峰值。小型/厨房用品组件在白天出现较高峰值。（c）各组件中的热门品牌/商品类别。尺寸越大表示关系越密切。（d）热门商品类别随状态转变而变化。上：空调在状态#2（初夏）成为最常见的商品。下：暖风机是状态#4（秋季）的热门商品。

C 实验

C.1 实验设置

我们在一台配备 192GB 内存、运行 Linux 的 Intel Xeon E5-2637 3.5GHz 四核 CPU 上进行了实验。

生成数据集。我们首先生成四种稀疏事件张量（1、2、3 和 4），其观测数为 $100K$，表示为 $\mathcal{X} \in \mathbb{N}^{100\times100\times100\times100}$。事件条目的每个属性均从多项分布中抽取，其参数由随机值 $[0.1, 0.5]$ 和狄利克雷先验定义。最后，通过组合不同的事件张量，构建四个不同的合成数据集，如下所示 [37]: “1,2,1”、“1,2,3,2,1”、“1,2,3,4,1,2,3,4”、“1,2,2,1,3,3,3,1”。

实现与参数。我们使用了 [72] 中 LDA、K 均值、LOF 和 iForest 的开源实现。对于 NTM，我们基于 PyTorch 框架实现，并采用学习率为 $1e-3$ 的 Adam优化，遵循 [85] 中的设计和参数设置。我们还使用了作者提供的 TICC [37], 、T-LSTM [17]、RRCF [34]、CubeMarker [39], 和 MemStream [21], 的开源实现，并遵循原论文建议的参数设置。为了公平比较计算时间，我们使用 Python 实现了 TriMine，遵循作者提供的 C 语言实现。LDA/NTM 的输入是所有类别的词袋表示，即 $\mathbf{W} \in \mathbb{N}^{\tau\times(U_1+\cdots+U_M)}$。评估聚类准确率时，当前张量的宽度设为 10。由于 TICC 和 T-LSTM 需要指定聚类数量，因此我们将其设为真实的聚类数量。DBSTREAM（由 [7], 实现）和 CubeScope 会自动确定聚类数量。我们将 DBSTREAM 中每个微簇的半径设为 8。5，并为 CubeScope 设置组件数量 $K = 8$。为验证检测准确率，我们为所有方法设置 $\tau = 1$。我们使用长度为 5τ 的数据流来对 CubeScope 进行模型初始化。

C.2 有效性

我们还展示了 CubeScope 在电子产品数据集上的效果。

在线营销分析。图9展示了 CubeScope 在电子产品数据集上的流挖掘结果。该数据是从一家大型家用电器和电子产品网上商店获取的一年期购买数据。数据包含两个属性列表：867 个品牌和 124 个商品类别，并带有小时级时间戳。

- 状态识别：CubeScope 在图9（a）中发现了四种状态。具体而言，我们的方法发现，7月前后的一小段时间内出现了状态#2，随后夏季出现了状态#3，秋季则出现了状态#4。这一结果表明，购买行为会随季节更替而变化。

- 多方面成分分析：图9（b）展示了状态#1中两个主要成分随时间的变化，这两个成分呈现在时间分量矩阵$\mathbf{B}$中。我们将这两个成分分别命名为“白色家电”和“小型/厨房用品”。这两个成分呈现出截然不同的行为。“白色家电”序列在夜间达到峰值，而“小型/厨房用品”序列则在白天达到峰值。图9（c）展示了状态#1中的属性成分矩阵$\{\mathbf{A}^{(m)}\}_{m=1}^{2}$，即两个成分（行）与两个属性（列）之间的潜在关系。图9（d）展示了热门商品类别随状态转变而发生的变化。这些变化合乎情理。如上图所示，状态#2（初夏）中的“白色家电”成分与空调的关系最为密切。同样，下图展示了状态#4（秋季）中的“小型/厨房用品”成分，其中暖风机成为热门商品。