An Algorithmic Approach to Event Summarization

一个事件摘要算法的方法

摘要：

最近,很多研究已经指向总结事件数据,希望总结将会引领我们更好的理解产生事件的系统。然而，并不是提供一个系统的总体照片，通过很多最新的方法获得的总结是分段的，每个片段描述一个孤立的快照系统。我们认为最好的总结,无论是在它的最小描述长度还是可解释性方面都获得了内部动力学系统的理解。这样的理解包括,例如,什么是系统的内部状态,以及这些状态之间是如何进行系统交替的。在本文中,我们采用事件数据汇总的一个算法。更具体地说,我们使用一个隐藏的马尔可夫模型描述事件生成过程。我们表明,基于隐马尔可夫模型学习的总结事件，达到简短描述长度和高可解释性。实验表明,我们的方法是有效的和实在的。

一般术语：

算法

关键词：

事件总结,隐马尔可夫模型,最小的描述长度

1，介绍

很多复杂的系统采用复杂的记录保持机制来记录发生在系统里的各种各样的事件。　日志通常包括一系列的事件,每个事件与时间戳相关联。事件日志包含系统的丰富信息。例如，在大规模集群系统中，事件日志包含关于软件和硬件故障和失败的信息。越来越多的努力正在致力于事件日志的分析,在今天，我们在系统事件日志模式的历史活动中,进行了现状的评估和对未来风险的预测。

事件日志分析提出了重大挑战。日志通常是又大又嘈杂的；从日志文件找到一个可行见解的模式的难度不小于海底捞针。一些现有的工作人员对事件日志分析使用数据挖掘方法，但是大多数的方法仍然是在发现数据频繁发生的阶段。这些方法报告大量的频繁集或模式,但它们的含义或意义是数不清的。以我们最好的知识来看，现有的工作都没有探索发现的模式之间的关系,他们也没有揭示这种模式之间是如何相互作用来组成底层系统的动态。

在本文中,我们提出了一个解决事件摘要的算法。不像以前的工作输出大量的不相关的，数据内的本地模式，我们的方法努力将模式转换为有机系统的组成部分，这样我们就可以有一个全局视图和对系统有一个更好地理解。特别地，我们希望找到生成事件的机制,并且使用机制作为总结。它根据一些相似之处找到一个字符串的算法复杂度,也就是说,最短和最有效的程序,可以生产字符串。当然,没有机械的方式这样做。我们要做的是让一个受过教育的猜测,事实证明,不仅导致一个更简洁的总结,但是也更可翻译的总结。

1.1 事件序列总结

现有的大部分工作集中在挖掘频繁集或事件日志模式。然而,单单频繁的发作不能创建一个系统的全局模型。一些近期的作品关注于事件序列模式，而不是寻找频繁的模式。例如,在[10],一个事件序列分为不相交的时间间隔，并且每个时间间隔都会生成一个描述事件的模式。结果是一个序列模式,构成了原始事件数据的总结。总结的问题是最优化的，因为它的目标就是找到一个有着最小描述长度的模式。然而,一个最小长度的描述没有必要揭示系统的复杂性。特别是,它不描述模式之间的关系。这样的关系-例如，模式X每两个小时左右定期发生，或模式Y出现在模式Z之后-这是很重要的，因为他们改善了对系统的理解。我们用下面的例子来说明其局限性。

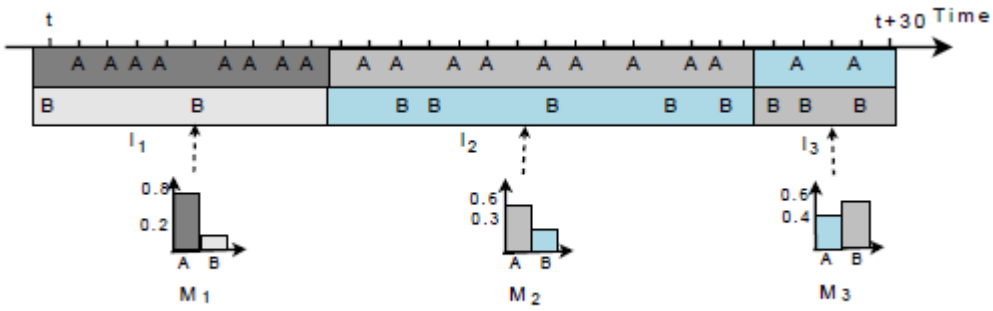


图1 动机的例子

例1：图1显示了一个事件子序列跨越时间(t,t + 30)和包含类型A和B的事件。在[10]段的总结方法序列分为多个时间段和学习事件模型分布在各区间。序列分为3个间隔I1、I2 I3,导致3模型M1,M2,M3(深点的颜色代表了高频事件)。[10]的目标是以这样一种方式分裂事件序列导致模型整体描述长度最小。

由此产生的模型——事件日志的摘要——揭示了隐藏在事件序列的系统动力学的能力有限。例如，它不提供信息，如M1不同于M3但是相似于M2。也不告诉我们,M1通常是跟随M2。换句话说，它没有发现不同模型之间的联系。所以,从这个意义上说,这种方法没有提供一个全球事件序列的结构。

总之,segment-by-segment方法优化最小描述长度有以下限制:

- 它们不确定在不同模型中的相互作用，而且也不能解释隐藏在数据后面的动态，或预测它们未来的模式。

- 它们专注于最好的压缩数据。然而,对于很长一段事件序列(我们有很多重复的模型将显示这是一个非常可能出现的情况),压缩率很低,因为相同模型的副本存储。

1.2 我们的方法

我们想要找到生成事件的机制，然后我们编码机制作为总结。当然，没有寻找这样的算法机制。在本文中，我们提出以下的“教育猜想”：系统运行在不同的状态；在每个状态下，系统展示稳定行为;状态转换有一定的规律性。这种猜测往往使 我们总结的事件数据更简洁,和得到更可判断的方法。 事实上,受过教育的猜测是基于以下的观察:

1. 一个系统运行在不同的状态是显而易见的。例如，在某个确定的时间内，一些时间频繁的发生,而且在另一段时间内，另外一系列

时间频繁的发生。换句话说，在每个状态下，系统的行为显示了特定的特征。

2. 一旦一个系统在一个状态下，他将会一直保持在这个状态，知道另外某个事件发生引导它去向另外一个状态。当内存使用量低于物理内存容量时，系统的行为在一定程度上表现出一定的模式。情况一直持续到内存占用超过容量，在这个时候系统启动分页并演示另一种模式。

3. 系统在一组状态中反复交替。

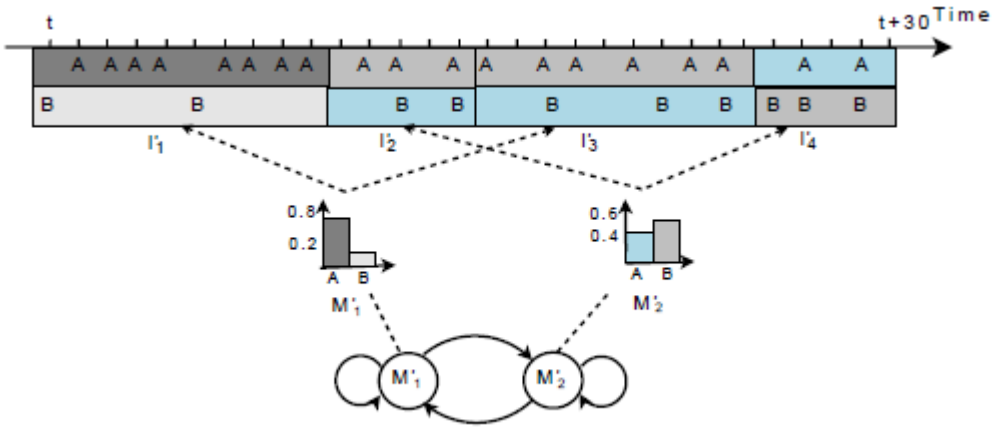


图2：我们的方法

这些观察激励我们总结数据首先通过理解系统。考虑一下在图1中的这个例子。从整个序列假设,我们可以学习两种状态:

- 状态1，事件A发生频繁，事件B发生频率较低。I1和I2是这个状态下的例子。
- 状态2，事件B发生频繁，事件A发生频率较低。I1和I2是这个状态下的例子。

通过给定的这两个状态，我们对图2所示的(t,t + 30)做了总结。具体来说,I2在图1中被分成2个间隔:I2’和I3’,分别对应于M2’和M3’。此外，两个状态不是孤立的，而是交替发生的。换句话说，我们希望发现我的一组模型和它们之间的时间关系。 很明显，这些信息给我们一个对这个系统更好的理解，促进对其未来业务的预测。

注意，上述结果不同于我们在图1中所知道的。此外，在我们的方法中所学到的模型不能简单的聚类和相关联的分部由分部概要在图1中，因为它的结果可能包括许多没有意义的模型，MDL和HMM。为了分析我们方法的优势，我们采用最小描述长度（MDL）原理。我们的直觉是，如果我们可以更好地了解系统，我们应该能够提高数据的压缩比。

这确实是如此。基于MDL准则，为了代表的事件序列，我们产生两种成本：编码模型的成本，以及给定模型的数据编码的成本。前者取决于我们发现了多少模型，和编码的每个模型的成本，后者取决于每一个模型拟合数据的程度。

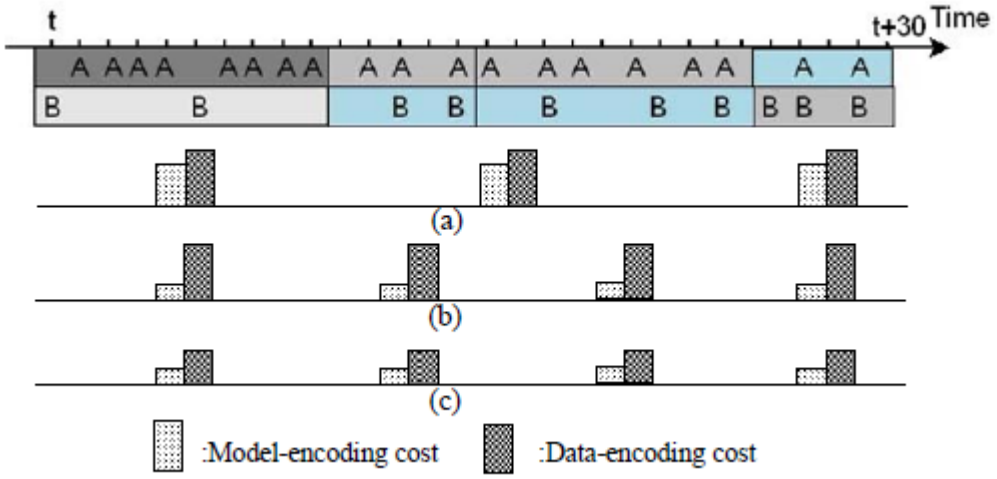


图3：成本数据的总结：（a）segment-by-segment方法;（b）我们的做法，假设事件是彼此独立的；（c）我们的方法，利用事件之间的相关性

不难看出为什么我们的方法更有优势与压缩比segment-by-segment方法相比。继续前一个例子，图3说明了三种不同的方法所产生的编码成本。对于每一个间隔，我们使用双棒来表示编码成本。左栏表示编码模型的成本，而右边的栏表示给定模型的编码数据的成本。图3(a)显示segmentby-segment方法的成本,图3(b)显示我们的的方法。不同来自于一个事实，即我们只编码模型一次（每一个模型是不同的）。对于每一个区间，用哪个模型来描述它，我们只编码模型的序数。它是更有效的。

这两种方法都没有考虑事件的相关性：某些事件可能发生在一起,而其他可能发生完全的相互关系。利用这样的特性可以降低编码成

本，因为一个事件类型的不确定性可能会减少另一个的信息。在本文中,我们利用相关性提出另一种方法。图3(c)显示了其成本。我们可以看到，这两种编码模型和编码数据的成本减少了段的分段方法。

内部状态确定我们观察到的事件发生，是隐藏的，随着时间的推移不断发展。因此，它是很难建立一个基于MDL准则的目标函数，然后通过最小化直接找到最好的总结。事实上,内部状态是类似于一个隐藏的隐状态马尔可夫模型(HMM)。后来，我们将显示的最小描述长度和一个隐马尔可夫模型的最大产生概率之间的等价性。这使我们能够应用基于HMM方法学习总结。

2.相关工作

频繁集挖掘。频繁集挖掘是一种流行的方法来获得时间序列的事件序列[1，15，13，2，16]。Agrawal等人提出了几种基于Apriori算法的方法来挖掘序列模式。Mannila 等人提出的 [15]的框架集的飞行常客英里的探索为从事件序列的时态模式挖掘。它发现在时间序列中频繁出现的。一个情节的频率被定义为在该情节发生至少一次的时间轴的窗口的数目。建议的计数算法使用一个有限状态自动机，以获得一组候选集的频率。一些扩展基于windows的频率的扩展模型也在(2,16)被提出。总体而言，频繁情节可以被视为局部模型，它们是不足以揭示系统的内部动态，因为它们不模型之间的关系模式。换句话说，模式是孤立的。

拉克等人[13，12]提出了一种基于情节非重叠出现的概念，频率测量。一种有效的方法基于频繁测量被提出对于频繁发作的。此外，作者建立了频繁发作和HMM模型之间的联系。他们引入了一类特殊的离散隐马尔可夫模型，称为集生成HMM（EGH），和每集之间用一个独特的EGH联系。它有更频繁的发作的特性，越有可能是由相应的EGH生成。然而，每个HMM仅代表一集，而且**没有试图总结整个组模式提供了一个系统的全局视图生成事件序列。换句话说，**它仍然集中在当地的事件序列中的模式。

事件摘要　近期的一些作品进行总结，通过事件序列分割[14、11，10]。Mannila等[14]关注单一事件序列类型。在每个时间间隔内，事件的发生是由一个恒定的强度模型为蓝本。Koivisto等人[11]根据最小描述长度的方法识别序列中的块结构。Kiernan等人将在[14]提出了处理多个事件类型的模型。MDL原则是用来测量质量摘要。但在这两种作品中，他们模拟了不同段之间的关系，而且段都是孤立的。在某种意义上,我们结合两者优点的频繁模式和分割模型来构造一个更好的全局模型的事件序列。我们的模式能够描述事件序列中的数据,以及模式之间的关系。

时间序列分割。　许多工作已经完成了时间序列分割[9，8，6，7]。在每个分部的子时间序列是由一个线段[9]或平均值的时间序列中的段[8]表示。我们的目标是段时间序列,以便总误差最小化。因此,他们的目标是学习每一段当地的模式,不提供模式之间的联

系,这使得它们类似于我们在（14,10）之前讨论过的方法。DFT(离散傅里叶变换)基于方法[5]和DWT(离散小波变换)基于方法[17]可以学习时间序列的某种全球模式,但是一般来说,模式派生的这种方法不容易解释。

流分类学习中隐藏的概念　　使用有限数量的概念来总结数据在一个无休止的数据流[19，20]的想法已经出现在其他采矿任务中。流数据进行分类,而不是学习新的流分类,杨等[21]从类似的数据中发现了历史分类。在（4,3）中隐藏概念的设定控制数据生成是固定的。这使得他们能够首先找到一组发生在流中隐藏的概念，然后检测到当前的概念，并使用相应分类对未标记的数据进行分类。我们的工作是基于相同的假设。然而,解决问题的困难和技术是不同的。此外，我们可以平衡模型的数量和总结的质量，而不是固定数量的概念。

3.问题陈述

一个时间序列包含发生在离散时间点的事件。特别的，时间有m种不同的类型E = {E1，E2，　•　•　•　，Em},并且一个事件发生是一对（E，t），其中E∈ E且t是它的发生时间，表示为一个整数在区间[1，T]。表1总结了本文以及其他地方使用过的符号。

Symbol	Description
T	length of time line
m	number of event types
$\mathcal{M}$	set of models
$\mathcal{M}_{ind}$	models for independent events
$\mathcal{M}_{dep}$	models for correlated events
$\mathcal{I} = (I_1, \cdots, I_k)$	a segmentation of the event sequence
$\sigma(i)$	model corresponding to segment I_i
$n(E, I_i)$	number of time points at which event E occurs

Table 1: Notations

3.1 模型

我们想要找到生成事件的机制序列,并使用它作为总结。因为找到确切的机制是不可能的，我们假设隐藏的机制可以简化为一个状态机，而且在每个状态下，系统表现出稳定的行为,或换句话说,它产生的事件在每个状态表现出稳定的统计特性。

我们的第一个任务是模拟每个状态，或模拟运行在一个状态下系统产生的事件的特征。在本文中,我们采用一个简单的模型,它只描述了每个事件的强度,但是忽略事件的相对顺序。具体来说，每一个模型Mi是由一组概率组成的：

Mi = (pi(E1), pi(E2), • • • , pi(Em))

pi(Ej)是事件类型Ej的发生概率。

当然，我们可以使用更复杂的模型。例如，在很多应用程序中，事件发生的不同类型是相关的。换句话说，某个特定类型事件的发生/缺失可以关于其他类型事件发生/缺失的的信息。一个模型捕捉

这种相关性可能透露系统更多的细节，并产生一个更简洁的总结序列。我们用Mind表示模型,假设事件是独立的, 和Mdep模型假定事件相关的。对于Mdep,我们用条件概率来描述不同类型的事件的发生。我们描述如何学习思想,讨论Mdep第六节的细节。

3.2 问题陈述

事件日志总结的问题可以表示如下:

问题陈述：给定一个事件日志，把它分成一个序列片段，在给定的类里用模型来描述每个片段，这样的整体描述有一个最小长度。

在本文中，我们考虑两种模型，Mind and Mdep。问题陈述隐式地要求我们考虑模型之间的关系，这可能导致更短描述长度的总结和剥离生成序列的内部机制。

4.目标生成函数和HMM

在本节中,我们介绍一个基于MDL原则的目标函数。为了找到有最小描述长度的总结，我们建立总结之间的等价关系和最好的拟合HMM。

4.1 目标函数

我们一个事件序列分割成不相交的部分, 并用一个特定的模型描述每个段。为了测量总结的质量，我们提出一个与MDL原则一致的目标函数Q。根据MDL原则,我们需要编码模型和模型中给定的数据。在这节中，我们为模型类Mind定义了目标函数。在第六节中，我们把它扩展为模型类Mdep。为每个模型Mi∈M,我们需要编码概率π(E1)

，・・・，π(Em)。每个需求概率logm位编码[18]。所以编码模型Mi的成本mlogm和成本的编码M,用厘米表示为Cm = |M| ・ mlogm，其中|M|表示M中模型的数量。显然,Cm只取决于m,事件类型的数量和|M| ，模型的数量。它不依赖于事件日志的大小。因为它是通常情况下,T≫ M和T≫|M|事件序列的大小,在总结大事件日志时，该费用是微不足道的。

编码事件序列的成本包括以下两个部分：

- Cs:编码分割的成本，其中包括编码段的长度和从段到模型的映射。
- Cd:编码事件发生的成本。

我们首先讨论在一个单独的段里边如何编码一个事件的发生。假设模型Mi =(π(E1)，・・・，π(Em)) 用来描述I。观察I中所有事件的概率，在模型Mi中给定的是

$$p(I|M_i) = \prod_{j=1}^{m} p_i(E_j)^{n(E_j,I)}(1-p_i(E_j))^{n(\bar{E}_j,I)|} \qquad (1)$$

其中n（E，I）是当E发生时I中时间点的数量，n(¯E ; I)是E没有发生时I中时间点的数量。

因此，用多少位来描述I，在模型Mi中给定了，

-log p(I|Mi)

显然，更好的Mi适合I，更大的p（I/Mi），和更小的-logp（I/Mi）。我们使用M б （i） 来表示这个描述间隔Ii的模型。给定整个细分和模型组M，所需的总比特数描述在所有领域发生的事件是

$$C_d = -\sum_{i=1}^{|\mathcal{I}|} \log p(I_i|M_{\sigma(i)})$$

第二部分的成本,Cs，是编码分段所需的位数。对每一部分，我们对它相应的模型和长度进行编码。为了编码相应的模型，一个简单的方法就是编码模型的id。我们需要log(|M|)位去编码每一段。然而，如果我们知道模型之间的关系，我们可以用更少的字节来编码同样的信息。直观来说，模型有更高的发生概率，我们需要用更少的字节来编码。用pi(1 ⩽ i ⩽ |M|)来表示模型的先验概率，pij (1 ⩽ i; j ⩽ |M|, i = j)表示跟随模型Mj的模型Mi发生的概率。对于第一部分I1，我们用先验概率p б （1）代表它的模型。至于Ik’s (k ⩾ 2),没有直接编码模型，而是基于先前段的模型Ik-1编码。因此，编码映射从领域模型所需的成本为：

$$C'_s = -\log p_{\sigma(1)} - \sum_{k=2}^{|\mathcal{I}|} \log p_{\sigma(k-1)\sigma(k)}$$

我们考虑用长度分布的概率来对每一段的长度进行编码。我们可以推出它的分布通过发现系统会保持一个状态多长时间。系统保

持在状态i的概率是pii。因此，对于模型Mi描述的部分I，它长度的概率与pii∧（|I|-1）成比例。我们执行了一次转换，使用更多的位来编码一个罕见的长度，和更少的比特编码一个频繁的长度。编码所有段的长度所需的成本是：

$$C_s'' = \sum_{k=1}^{|\mathcal{I}|} (|I_k| - 1) \cdot (-\log p_{\sigma(k)|\sigma(k)})$$

编码细分的总成本是Cs’和Cs’’的总和：

Cs=Cs’+Cs’’

我们使用一组概率,pi和pij，1<=i,j<=|M|,来描述系统的动力学。这些概率也需要进行编码。然而，因为i和j是以|M为界限的，事件日志的大小增加时它也不会增加，我们不把它包括在目标函数Q中，只是作为Cm。

最后，目标函数为：

$$Q(M, I) = Cd + Cs \quad (2)$$

我们的目标是事件日志分割成一序列段，每个段映射到一个特定的模型，公式2是最小的。

4.2 隐马尔可夫模型

最小化公式2分割是一个具有挑战性的任务。如果我们知道了这组隐藏的模型M = {M1, ・ ・ ・ , M|M|}，那么分割时间序列在上下文中不重叠出现不会很困难。如果我们知道分区 I = {I1, ・ ・ ・ , I|I|}，然后我们可以聚类相似段获得M。当然，M和I的挑战都不知道。

解决这个问题未能规模或找不到最佳的总结。对于事件日志生成时间[1，T]，有$T^{|I|}$种可能的分割。此外，对于每个分割I，有$|I|^{|M|}$中段和模型之间可能的映射。因此，搜索整个空间以解决优化问题是不可行的。一个更可行的方法是在[10]，动态程序-明是一个贪婪的方法相结合，总结事件日志。然而，该方法没有系统的内部动力学的知识，和段总结是次优为重要信息系统的信息是不完全的利用。

在本文中，我们提出了一种基于隐马尔可夫模型的方法来解决这个问题。为了方便讨论,我们使用一个m 维列向量代表观察到的事件在时间t:如果Ei发生在时间t,$o^t = 1$，否则$o^t = 0$。因此，m*T排列$O = (o1, \cdots, oT)$表示事件日志超过时间[1, T]。给定一个观察到的事件序列O，我们学习一个隐马尔可夫模型λ，而且相对应的最优状态序列。在4.3节，我们证明最小化目标函数$Q(M, I)$相当于生产概率最大化。换句话说，学习λ和使我们模式集合M和分割I，从而最大限度地提高目标函数。

一般来说，HMM，表示为 $\lambda = \{S, A, B, \pi\}$，用以下因素来描述：

- 一系列状态S = {1, 2, ・・・, K}
- 状态装换概率 A = {a**i j**}，状态i转换到j的概率为aij
- 输出概率B = {b**i** (o)}，其中i是一个状态，m维矢量o = (o^1, ・・・, o^m)是一个观察， b**i** (o)是系统在状态i的观察o的概率。

- 初始概率π = {πi}，而 π i系统在状态i开始的概率

生产概率　给定HMM的λ，一个观察序列O，生产概率的问题λHMM产生O沿状态序列，

$\mathbf{s} = (s_1, \cdots, s_T)$:

$$P(O, \mathbf{s}|\lambda) = \pi_{s_1} b_{s_1}(o_1) \prod_{j=2}^{T} a_{s_{j-1}, s_j} b_{s_j}(o_j) \qquad (3)$$

其中st是在时间点t的状态。生产更大概率意味着λ和状态序列s能更好地描述O。最好的状态序列是最大的生产概率。

4.3 等价关系P（O，S｜λ）和Q

首先,我们建立术语之间的连接。

- 每个HMM状态i对应于模型Mi ∈ M.
- 在状态转移概率，ajj等价于上下文切换概率pij
- 输出概率bi(o)，1 ⩽ i ⩽ K,是基于Mi的发生概率定义的：

$$b_i(o) = \prod_{j=1}^{m} q_{ij}(o)$$

$$q_{ij}(o) = \begin{cases} p_i(E_j) & \text{if } o^j = 1; \\ 1 - p_i(E_j) & \text{if } o^j = 0. \end{cases}$$

- 初始的概率与模型的概率是相等的，也就是， πi = pi, 1 ⩽ i ⩽ K.

其次，建立连接的状态序列和s之间的分割I。给定一个状态序列s = （s1, s2, • • • , sT ），我们可以通过合并获得分割连续时间点对应于相同的状态。最后每一部分对应一个状态,该状态表示的模型。

现在,我们讨论生产概率之间的等价性和目标函数。我们首先分解生产概率 P （0, s| λ ）成两个部分：

$$p(O,\mathbf{s}|\lambda) = \prod_{j=1}^{T} b_{s_j}(o_j) \cdot \pi_{s_1} \prod_{j=2}^{T} (a_{s_{j-1}s_j}) \qquad (4)$$

$$P_1 = \prod_{j=1}^{T} b_{s_j}(o_j) \text{ and } P_2 = \pi_{s_1} \prod_{j=2}^{T} (a_{s_{j-1}s_j}).$$

编码事件发生的成本。 通过一些代数转换，我们有：

$$\begin{aligned} -\log P_1 &= -\log \prod_{j=1}^{T} b_{s_j}(o_j) \\ &= -\log(\prod_{i=1}^{|I|} \prod_{j \in I_i} b_{s_j}(o_j)) \\ &= -\log(\prod_{i=1}^{|I|} P(I_i|M_{\sigma(i)})) \\ &= C_d \qquad (5) \end{aligned}$$

编码细分成本。 同样，- log *P2*等价于编码细分成本 Cs：

$$
\begin{aligned}
-\log P_2 &= -\log(\pi_{s1}\prod_{j=2}^{T} a_{s_{j-1}s_j}) \\
&= -\log(\pi_{s1}\prod_{j\geq 2, s_{j-1}\neq s_j} a_{s_{j-1}s_j}) \\
&\quad -\log(\prod_{j\geq 2, s_{j-1}= s_j} a_{s_{j-1}s_j}) \\
&= C'_s + C''_s = C_s \qquad (6)
\end{aligned}
$$

总之，我们有：

$$-\log p(O, \mathrm{s}|\lambda) = Q(\mathcal{M}, \mathcal{I})$$

因此我们总结事件序列的控制点和最优状态序列的概率最大化生产问题。此外，一旦我们得到隐马尔可夫模型和最佳状态序列，我们可以得到的集合的模型和分割。

5. 独立情况：Mind

在本节中,我们讨论如何有效地学习HMM,最优状态序列的事件日志。我们的方法包含两个阶段。在第一阶段,我们初始化HMM,在第二阶段中,我们使用一个迭代方法改进模型。

5.1 初始化

我们提出两种方法:一个随机方法和基于集群的方法,启动HMM的参数 λ 。

随机的方法。 在随机方法中，我们假设在Mind中有K个模型，K是估计值，而且将会在第二阶段改变。对于每一个模型Mi，1 ≤ i ≤ K,我们生成事件发生概率pi（Ej ），j = 1，・・・，m 。所有转移概率和初始概率也随机发起。

基于聚类的方法。 基于聚类的方法需要以下步骤。我们第一次分区序列为多个段。对于每个部分,我们学习一个模型。然后,我们集群获得模型K组。最后,基于模型的团体,我们发起的HMM的参数。

我们在一个自底向上的方式段的事件序列。该算法首先考虑每个时间点为一段，I = {I1，・・・，IT }。对于每一个Ii，我们学习此段中基于事件模型的事件发生。然后，我们尝试合并到一个更大的段的相邻段。选择段对合并的标准是合并应产生最小的增加。假设对于段 Ii和Ii+1，合并他们生成新的段I’而且从I’中学到的模型是M’。增加的成本可以计算为：

$$-\log(p(I'|M')) + \log(p(I_i|M_{\sigma(i)})) + \log(p(I_{i+1}|M_{\sigma(i+1)}))$$

一旦选择了一对,我们学习一个基于新模型的段。合并一直持续到段的数量达到指定的阈值。

在第二步中,我们使用集群获得的模型k - means聚类算法。因为每个模型表示概率的事件类型,我们使用K-L 散度两个模型之间的距离测量。Mi = {pi(E1)，・・・， pi(Em)}, andMj = {pj(E1)，・・・， pj(Em)},然后

$$Dis(M_i, M_j) = \sum_{l=1}^{m} p_i(E_l) \log \frac{p_i(E_l)}{p_j(E_l)} + \sum_{l=1}^{m} p_j(E_l) \log \frac{p_j(E_l)}{p_i(E_l)}$$

在第三个步骤中,对于每个集群,我们学习发生概率。基于他们,我们启动输出概率每个状态。概率和初始的过渡概率是根据初始化段序列。

5.2 迭代隐马尔可夫模型

我们用 $\lambda 0$表示原始的HMM。然后,我们使用一个迭代过程改进的初始模型。让 $\lambda k-1$表示轮K-1得到HMM。在k轮,迭代需要两个步骤。第一步，基于 $\lambda k-1$,，我们学习最优状态序列sk。第二步，基于s^k，我们学习了一个新的模型， λk。这个过程一直持续直到HMM没有任何改变。我们将显示在每一个迭代轮,生产概率不低于在上一轮。

学习最佳状态序列　在k轮的第一步,我们学习最可能使用维特比算法的隐状态序列。维特比算法计算转发概率， $\delta t(i)$，是最有可能的隐状态序列到t的概率且st = i。它认为：

$$\begin{aligned}\delta_t(i) &= \max_{s_1,\cdots,s_{t-1}} P(o_1,\cdots,o_t,s_1,\cdots,s_{t-1},s_t=i) \\ &= \max_{s_1,\cdots,s_{t-1}} \pi_{s_1} b_{s_1}(o_1) \prod_{j=2}^{t} (a_{s_{j-1}s_j} b_{s_j}(o_j))\end{aligned}$$

假设在时间t-1，，我们已经有了所有的可能性δT-1（J），J = 1，・・・，K。然后，我们计算δT（i），i = 1，・・・，K，基于δT-（J），为

$$\delta_t(i) = \arg\max_j(\delta_{t-1}(j)a_{ji})b_i(o_t)$$

最后，在时间t，δT（i）是隐藏状态和ST =i用回溯的最可能的序列的概率方法的正确性。回顾 $\delta T(i)$中最大的一个， $i = 1, 2, ・・・, K$，我们获得最优状态序列,并将其表示为s^k。更新HMM 第二步，基于 $_sk$，我们学习一种新的HMM， $_\lambda K$，使P（0，$_sk$ | $_\lambda K$）最大化。

记得P（0，SK |λK-1）可以分解为两个部分，P1和P2，P1的输出概率确定。

$$P_1 = \prod_{t=1}^{T} b_{s_t}(o_t)$$

这让我们更新输出概率最大化P1。它可以转换为

$$P_1 = \prod_{i=1}^{K} p(D_i|M_i)$$

其中 Di = {t|st = i}和p(Di|Mi)相似于Eq. 1中的p(I|Mi)。因此，我们把p分解为k个独立的部分，p(Di|Mi)， 1 ≤ i ≤ K，第i部分只涉及到模型Mi，这意味着我们可以更新不同概率独立出现的模型。

此外，我们对P（DI ｜ MI）分解为M个部分，每个部分对应一个事件类型：

$$p(D_i|M_i) = \prod_{j=1}^{m} p(E_j|M_i)$$

$$p(E_j|M_i) = p_i(E_j)^{n(E_j,D_i)}(1 - p_i(E_j))^{n(\bar{E_j},D_i)}$$

这意味着我们可以更新独立事件类型的发生概率。因此,事件Ej,它的发生在模型更新Mi概率:

$$p_i(E_j) = \frac{n(E_j, D_i)}{|D_i|}$$

P2是由初始概率和转移确定，为了达到最大限度的P2，我们更新初始概率 π i为

$$\pi_i = \frac{n(s_t = i)}{T}$$

其中N（St=i）是我出现在 $\mathbf{s}^k$ 的时候状态数，并更新概率aij为

$$a_{ij} = \frac{n(s_t = i, s_{t+1} = j)}{n(s_t = i)}$$

在某些情况下,某些状态将不会发生序列。那时,我们从HMM删除它。所以,许多状态,K,可能在细分变量的过程是不同的。

因此,我们λk获得新模型。它可以轻松地证明任何HMM,它持有

$$p(O, \mathbf{s}^k|\lambda^k) \geq p(O, \mathbf{s}^k|\lambda)$$

方法的正确性

在k轮第一步，学习状态序列，SK，是最优的基础上产生的事件序列λK-1，持有

$$p(O,\mathbf{s}^k|\lambda^{k-1}) \geq p(O,\mathbf{s}^{k-1}|\lambda^{k-1})$$

在第二步，K是最大化状态序列$\mathbf{s}^k$ 生产概率模型，认为

$$p(O,\mathbf{s}^k|\lambda^k) \geq p(O,\mathbf{s}^k|\lambda^{k-1})$$

结合上述两个不等式,我们获得

$$p(O,\mathbf{s}^k|\lambda^k) \geq p(O,\mathbf{s}^{k-1}|\lambda^{k-1})$$

这意味着每一轮的最优状态序列的概率是不小于在上一轮。

6.相关案例：Mdep

模型类思维假设出现的不同- ent事件类型是独立于彼此。事实上,不同的事件类型之间的相关性存在于大多数应用程序中。对于想要了解系统行为的用户是很有用的，而且是很重要的因素对于想要减少编码成本的用户。在本节中,我们介绍一个新的模型类,Mdep,利用事件发生之间的相关性。

我们说明了利用一个相关性的例子的优势。假设我们知道E1发生,并假设E2的发生取决于E1的意义在E1发生的情况下，很有可能E2也会发生。在这种情况下， 条件概率p(E2 | E1)接近1。来描述E1和E2的出现,我们首先描述E1与它发生概率p(E1)。然后我们使用p(E2 | E1)来描述E2的发生。所以,E2的编码发生的成本为 -log(p(E

2 | E1))。在大多数情况下，这比用发生概率- log(E2)的编码成本更低。

6.1 Mdep的定义

我们使用根树木来表达不同事件类型之间的相关性。一般来说,一个事件的发生类型可以依赖于任何数量的事件类型。但在这个工作中,我们假设每个事件类型只取决于一个事件类型。原因是:1)在许多应用中，主要的质量相关的事件的类型取决于另一个事件类型的形式，而不是在其他多个事件类型。针对这种关系了对于系统的行为和编码成本的额外节省，我们有更多的了解。2) 描述多个事件类型之间的相关性，将导致很难对用户的理解非常复杂的模型。

在工程各模型MI，我们组织相关事件类型等之间是一种相关的树，记为ctreei，其中每个节点代表一个事件类型。根节点所代表的事件类型叫做根事件类型,并表示为Eri。事件类型代表由其他节点是根事件类型。 一个有向边从El到 Ej表明出现Ej依赖于El。

我们假设根不依赖于任何事件类型,我们代表它的发生概率,p（Eri）。无根的事件类型的Ej，它依赖于模型Mi中Edij表示的事件类型。它的发生所描述的两个条件概率:pi(Ej | Edij)和pi(Ej |¯Edij),前者是条件概率Ej的出现给Edij发生,后者是鉴于Edij不发生。如果一个特定的事件类型Ej并不依赖于任何事件类型,它拥有p(Ej | El)= p(Ej),El是任何事件类型。 因此,使用条件概

率不会导致额外的编码成本。事实上，事件发生的概率Mind和Mdep分别用Eq.7和Eq.8表示,显然这两个模型类之间的区别。

$$p_i(E_1, E_2, \cdots, E_m) = p_i(E_1)p_i(E_2)\cdots p_i(E_m) \quad (7)$$

$$p_i(E_1, E_2, \cdots, E_m) = p_i(E_{r_i}) \prod_{j=1, j\neq r_i}^{m} p_i(E_j|E_{d_{ij}}) \quad (8)$$

图4(b)显示了一个示例CTreei,ri = 3。至于非根事件类型,我们有di4 = 1,di5 = 1,di2= 3 ，di1 = 3 。因为事件类型是互相独立的,它可以被视为一个森林有五根节点,如图4(a)所示。树的相关的一个重要特点是它是无环的，这可以从公式8推断。此外，在学习HMM的过程中，每个模型的相关树结构将被更新，以便它可以更好地描述依赖。具体来说，对于每个模型，我们需要决定哪种事件类型将是最好的根事件，非根事件类型取决于哪种事件类型。

6.2 目标函数Q

正如在Mind，用Mdep中的模型总结一个事件日志，需要三个成本，Cm, Cd and Cs。Mdep中模型的编码成本比在MInd中的高，我们需要编码(条件)概率和相关的结构树。然而,这部分成本并不随着事件日志的大小的增加而增加。所以我们忽略它的目标函数。

Cs的定义与第四节是一样的。现在我们讨论的Cd是编码事件发生的成本。假设我们使用模型Mi编码段I的事件发生。在模型MI中，根事件是Eri，和事件类型非根事件靠的是Edij。I中事件发生的概率，给定的是Mi，是

$$p(I|M_i) = \prod_{j=1}^{m} p(E_j|M_i)$$

如果Ej是根事件，我们有

$$p(E_j|M_i) = p_i(E_j)^{n(E_j,I)}(1-p_i(E_j))^{n(\bar{E}_j,I)} \tag{9}$$

否则，如果它是一个非根事件，有

$$\begin{aligned} & p(E_j|M_i) \\ = \; & p_i(E_j|E_{d_{ij}})^{n(E_j,E_{d_{ij}},I)}(1-p_i(E_j|E_{d_{ij}}))^{n(\bar{E}_j,E_{d_{ij}},I)} \\ & p_i(E_j|\bar{E}_{d_{ij}})^{n(E_j,\bar{E}_{d_{ij}},I)}(1-p_i(E_j|\bar{E}_{d_{ij}}))^{n(\bar{E}_j,\bar{E}_{d_{ij}},I)} \end{aligned} \tag{10}$$

其中n(Ej, Edij , I)表示Ej 和Edij都发生的时间点的数量，$n(\bar{E}j ; Edij ; I)$表示Edij发生但是Ej没发生时时间点的数量，$n(Ej ; \bar{E}dij ; I)$ and $n(\bar{E}j ; \bar{E}dij ; I)$有相同的意思。编码Ii中数据的字节数量达到-log $p(Ii/M\ (i))$,而且成本Cd是

$$C_d = \sum_{i=1}^{|\mathcal{I}|} -\log p(I_i|M_{\sigma(i)})$$

我们定义目标函数是

$$Q(\mathcal{M}_{dep}, \mathcal{I}) = C_s + C_d$$

6.3 相应的工程模型

对应于Mdep状态的HMM，i还是代表模型Mi。初始概率和过渡是相同的。让o=(o^1, ・・・, o^m) 表示一个观察，其中 $o\mathcal{J} =$

0表示Ej发生，$oj = 1$表示没有发生。状态i生成o的输出概率变化为：

$$b_i(o) = \prod_{j=1}^{m} q_{ij}(o)$$

有qij的两个例子。在第一种情况下，EJ是Mi根事件类型，所以qij仅取决于Oj，我们有：

$$q_{ij}(o) = \begin{cases} p_i(E_j), & o^j = 1; \\ 1 - p_i(E_j), & o^j = 0. \end{cases}$$

在第二种情况下,Ej是一个非根的事件类型,因此qij依赖于oj和odij,Edij 是Ej依赖的事件类型。我们有

$$q_{ij}(o) = \begin{cases} p_i(E_j|E_{d_{ij}}), & o^j = 1 \text{ and } o^{d_{ij}} = 1; \\ 1 - p_i(E_j|E_{d_{ij}}), & o^j = 0 \text{ and } o^{d_{ij}} = 1; \\ p_i(E_j|\bar{E}_{d_{ij}}), & o^j = 1 \text{ and } o^{d_{ij}} = 0; \\ 1 - p_i(E_j|\bar{E}_{d_{ij}}), & o^j = 0 \text{ and } o^{d_{ij}} = 0; \end{cases}$$

与第四节使用相同的技术,我们可以建立生产之间的等价概率和目标函数:所以,我们的目标仍然是找到一个HMM λ和最优状态序列s,P(O s ｜λ)是最优化的。

6.4 学习隐马尔可夫模型算法

我们还使用一个迭代的过程来学习Mdep。在初始化阶段，一个随机的方法和基于聚类的方法用于初始化HMM。在随机的方法，我们首先随机产生关联树的每个模型的。然后为每个

边缘，我们随机生成条件概率。在集群的基础上，我们发起了一个统一的林前关系树的所有模型。此外，KL散度的计算是根据条件概率。

在每一个迭代轮，学习的最佳状态序列的步骤是类似的情况下，除了那个输出概率的定义是不同的。现在我们讨论的第二个步骤，我们在其中更新HMM。

在轮 k,认为，我们已经获得最优状态序列的$\mathbf{s}^k$，并尝试学习 λ^k。首先，我们更新变换位置概率和5节初始概率的方法.输出更新概率的方法更加复杂，原因是，除了更新率以外，我们也更新相关树结构为每个模型。具体来说,对于每个模型,我们学习新根的事件,和每一个非根事件的相关事件类型。更新过程包含两个步骤。首先,我们更新所有可能的概率；然后我们学习相关性树,基于概率更新达到比之前更高的概率比。

回想一下，在5节中，P1分解为K个部分，P（DI ｜ MI），（1≤i≤K），和每一个都可以进一步分解为M，P（EJ ｜ MI），（1≤J≤m）。在Mdep的情况下，这种分解还认为，除了P的公式（EJ ｜ MI）式9和式10的变化。我们更新了所有的可能性（1≤J，L≤m）模型MI如下：

$$p_i(E_j) = \frac{n(E_j, D_i)}{|D_i|} \tag{11}$$

$$p_i(E_j|E_l) = \frac{n(E_j, E_l, D_i)}{n(E_l, D_i)} \tag{12}$$

$$p_i(E_j|\bar{E}_l) = \frac{n(E_j, \bar{E}_l, D_i)}{n(\bar{E}_l, D_i)} \tag{13}$$

所以，P（DI ｜ MI）最大化。

更新相关树，我们之间建立的连接（DI ｜ P MI）和相关树CTree

- 为根节点Eri ，我们将重写为p(Eri |Mi)（计算公式9）。
- 从Ej到Edij的边缘,我们设置它的重量wdij;j是p(Ej ｜ Mi)(计算公式10)。

因此,产品重量的边缘和根节点相当于p(Di ｜ Mi)。所以我们更新相关的目标树对于Mi变成了寻找相关树,这样它的重量是最大化的产物。现在我们介绍我们的方法。很容易看到学习的最佳关联树,最大化p(Di ｜ Mi)，是一个NP问题。在本文中,我们提出一个贪婪算法学习相关树CTree 产品的重量是比当前树,CTreei。一个节点列表中问维护事件类型已经处理过的，一开始它是空的。该算法包含两个步骤:

步骤一：学习新根事件类型。对于任何事件类型Ej,我们检查是否设置它作为新根可以增加产品的重量。具体来说，我们发现事件类型Ej，j=ri， 满足：

$$w_{r_i} \cdot w_{d_{ij},j} \leq w_j \cdot w_{j,r_i}$$

左边的部分不平等权重的乘积的Eri 和EjCTreei在前面的树。正确的新树的一部分,我们切换Ej,Eri,即我们设置Ej作为新根,Eri作为它的依赖。

如果存在满足上述不等式的事件类型，我们将选择一个具有最大重量的产品作为新的根事件类型。我们也改变了以前的根事件类型Eri为非根事件类型和设置新的根的事件类型。如果没有，我们保持以前的根不变。在这两种情况下，新根加入NL。

步骤二：学习新的相关的事件类型。我们遍历以前的树CTreei后订单。假设当前的事件类型Ej,我们检查是否wdij更长对于任何l。有两种情况:

- 如果这是真的，Ej仍然是事件类型Edij取决的事件类型，所以我们添加Ej到NL。
- 如果没有,我们试图找到另一个新的Ej事件类型。为了避免产生关联树中的一个圆,我们只检查事件类型El满足

$$wl,j > wdij,j$$

如果这是真的，我们选择其中的一个是EJ取决于新的事件类型。最后，我们把Ej添加到NL。

很容易看到,依赖树不包含周期。此外,由于我们只有当调整依赖重量会增加,新产品树CTree的体重是不少于CTreei。然而,由于方法是贪婪算法,这个过程可能不会导致最优解决方案。

7.实验评价

我们进行综合的和现实生活中的数据集实验。我们对性能进行比较，我们的方法在[10]根据各种设置的方法。所有的算法研究采用垫实验室7。实验进行一个PC与英特尔奔腾4 2.4GHz CPU和2GB RAM。

7.1 算法的比较

我们测试我们的方法的有效性在4例:两个模型类(思想和Mdep)结合两个HMM初始化(随机和基于聚类的)：

	Model Class	HMM Initializer
RAN-IND	$\mathcal{M}_{ind}$	Random
CLU-IND	$\mathcal{M}_{ind}$	Clustering
RAN-DEP	$\mathcal{M}_{dep}$	Random
CLU-DEP	$\mathcal{M}_{dep}$	Clustering

表2:4例中的方法

我们比较了上述4种算法在[10]的方法，称为贪婪算法，因为它使用了一个贪心算法找到每一段都学习的最优分割和局部模型。

7.2 合成实验数据

首先,我们对合成数据进行实验。

数据集

我们使用以下参数来描述一种合成事件序列：K，我们人为的“植物”生成的事件序列模型的数量（或隐藏的数量）；Kini，在初始阶段的数量模型;m，事件类型的数量（E的大小）；n，时间序列

中包含的段的数量；d，事件类型之间的依赖；li,1<=i<=k,模型Mi生成片段的平均长度。

我们生成如下的K模型。首先，我们为每个模型随机生成d对相互关联的事件类型。然后在每个模型中输出所有的概率，包括独立事件类型p（E|E’）p（E|¯E’），非独立事件类型p（E），是随机生成的。所有的初始概率和转移概率也是随机生成的。

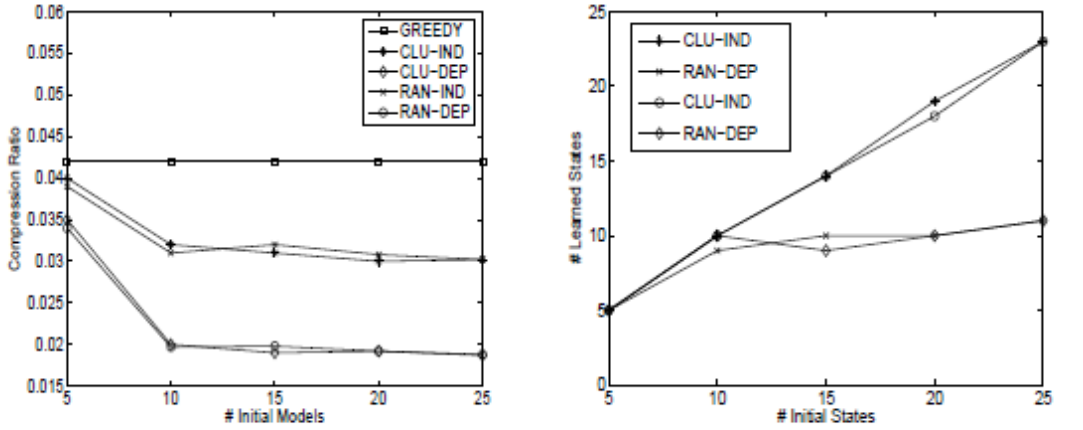
模型生成之后,我们生成n段基于模型的事件发生。首先，我们根据-log pii设置li，1≤i≤k。然后基于初始概率和过渡概率,我们生成一个状态序列长度为n。基于这个状态序列，我们生成一个片段的序列。假设第t状态是i,我们根据模型Mi生成事件。It的长度是根据高斯分布以及平均值li和li*5%。非依赖事件类型，事件发生的产生的高斯分布与输出的概率，作为平均10%*P（E）的方差。事件类型在某些事件类型的事件悬而未决，从条件概率的高斯分布生成性，P（E ｜ E′）或P（E ｜ E¯′），为平均10%*P（E ｜ E′）或10%*P（E ｜ E¯′）的方差；

压缩比　压缩比测量算法如何能对事件序列进行压缩。我们采用[10]的公式：

$$CR(A) = \frac{Q(A)}{Q_{unit}}$$

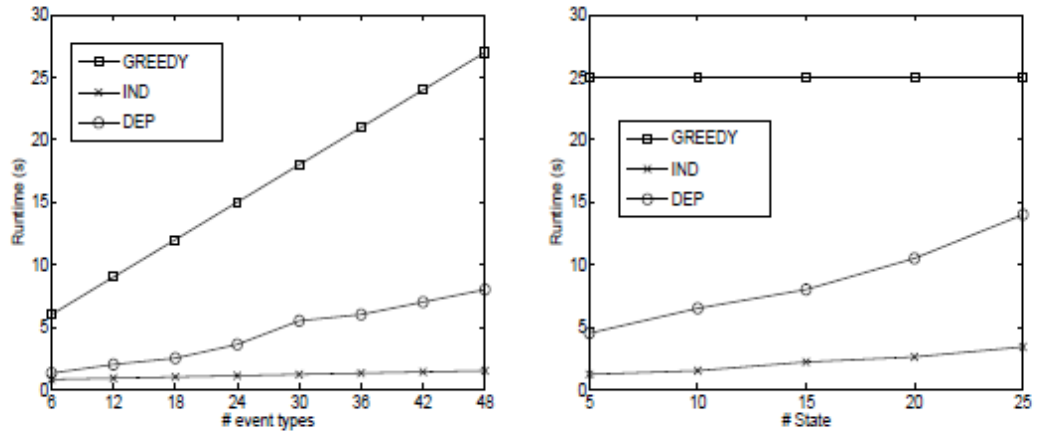
A可以是任意的5种方法，CLU-IND, RAN- IND, RAN-DEP, CLU-DEP, or GREEDY; Q(A)是用方法A生成事件序列的编码成本。在目

标方法中，Q = Cm + Cs + Cd;在贪婪算法中，成本计算在[1 0]中。 Qunit 是测量作为基线成本在各时间点是一个段，在段里，每个事件类型是由不同的概率描述。较低的压缩比CR(A)，方法A更好的总结事件序列。



(a) Compression Ratio (b) Detection of Hidden States

表8 K = 10, n = 100, m = 20,d = 10



(a) VS. m (b) VS. K_{ini}

表9: (a) n = 100, d = m=2, K = 10,Kini = 10; (b) n = 100, m = 20, d = 10, Kini = K

压缩率的比较。在第一个实验中，我们比较5种不同的压缩比（事件类型数目）。结果其他参数的设置，如图5所示。当m增加，所有5种方法的压缩比降低。对于贪心算法，压缩比减少，因为它的集群事件类型和使用一个概率代表在每个聚类后的事件的发生，从而降低了编码模式的成本。CLU-IND和RAN-IND 比贪婪的低压缩比。原因是，我们使用一个特定的fi概率来描述一个事件类型，所以我们的模型能反映发生的事件更明显准确地。至于费用的编码模型，我们的方法有更低的成本，因为我们只有一次编码每个模型。CLU-DEP 和RAN-DEP压缩率较低,因为他们利用事件类型之间相关性。

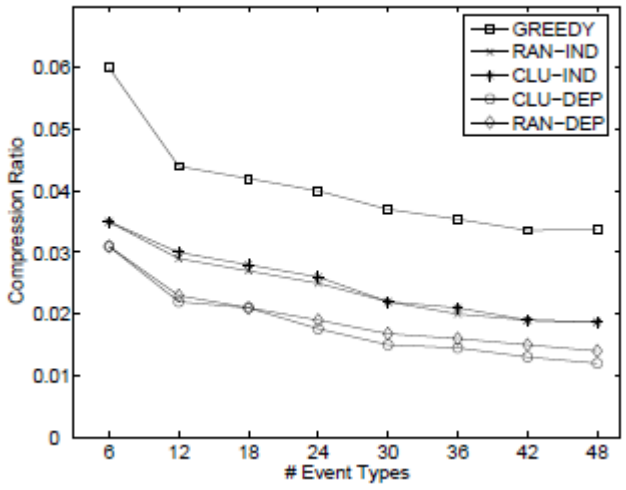


表5：n = 100, d = m=2, K = 10, Kini = 10

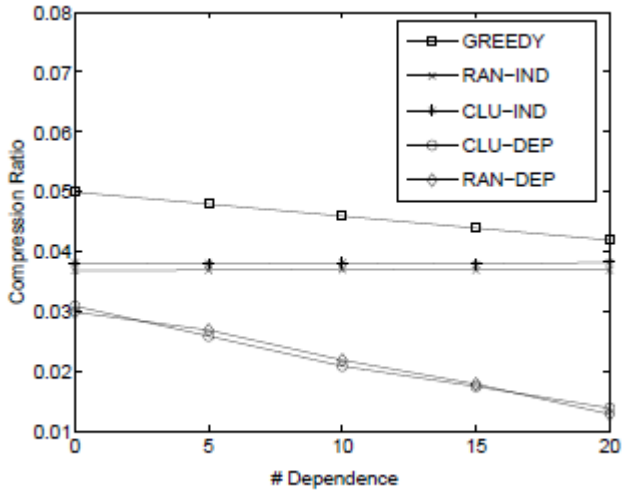


表6：n = 100, m = 30, K = 10, Kini = 10

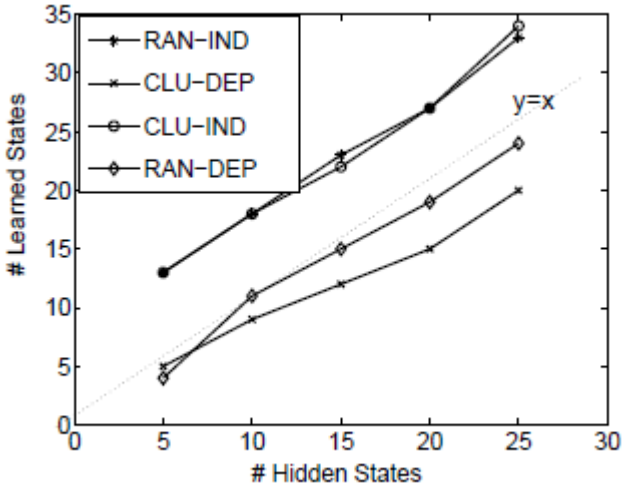


表7：n = 100, m = 20, d = 10, Kini = K + 10

在第二个实验中,我们测试的有效性,事件类型之间的相关性。当d不同的时候压缩比是可比较的。结果在图6中显示出来。可以看出，CLU-IND 和 RAN-IND 相对固定的压缩比，而在其他3种情况比减小。CLU-IND和RAN-IND的压缩比没有改变，因为他们不使用事件相关描述事件发生。在贪婪算法，当d增加，更多的相关性之间发生的

事件类型，以及其功能类型具有类似事件发生的概率增加。那么多的集群事件类型减少，这将降低编码模型的成本。在CLU-DEP和RAN-DEP压缩比下降之最。具体来说，在贪婪算法中，压缩比从0.05降低至0.042，然而在CLU-DEP和RAN-DEP中，压缩比从0.03降低至0.013.结果清楚地验证利用之间的事件类型相关的有效性。

初始状态的影响　在一般情况下，用户不知道在系统中的隐藏状态。因此，这是可取的，该方法可以准确地检测到真正的隐藏状态。症状之一是初始状态，数量丰富，能够收敛到真实的隐状态。所以，在这个实验中，我们测试的压缩比和数量的Lea当受到Kini丰富的变化。结果在图8（a）和图8（b）分别地显示。当Kini< k的时候，可以看出，所有提出的应用方法有类似的行为。他们不能学习足够的状态,而且压缩比可以。但是即使在这种情况下，我们的方法仍然比贪婪算法要好，它验证了一次编码的优势。当Kini≥ K，所有提出方法的压缩比在Kini增加时保持稳定。但是隐状态侦破的能力是相当不同地基于 ap- proaches的不同方式及随机方法聚类。在RAN-IND和RAN-DEP，状态收敛到真实的正确数量。然而在RAN-IND和RAN-DEP，最后的学习状态数量仍类似于Kini。其原因是，在集群为基础的方法，初始状态可以描述某些段，所以在迭代过程中，他们是不太可能被删除。事实上，由这2种方法学习的状态也有意义，可以从他们的压缩比看出来。相反的，在RAN-IND和RAN-DEP，因为状态是随机启动的，一些没有意义的状态将会被删除。他们更有可能收敛于真实值k .因此,随机方法是更好的选择学习真正的状态数。

检测隐藏的状态　在这个实验中，当K变化时，我们测试的准确性，检测隐藏状态时。我们总是设置为Kini= k + 10。结果如图7所示。它与上次实验的结果是一致的。 在CLU-IND和CLU-DEP，它的学习状态是接近Kini。换句话说，他们往往保持初始状态不变。相比之下，RAN-IND和 RAN-DEP可以正确检测到K。

效率　在这个实验中，我们测试所提出的方法的效率。被认为是2个因素，一个是事件类型，m，和其他一些是隐藏的状态，K。由于迭代不同，很难通过总运行时间比较方法。因此，我们比较了在每个迭代轮的建议的方法的运行时。结果图（1）和（2）分别示于图9。

可以看出，当m增加，在贪婪算法的运行时，比我们的方法更迅速。原因是，在每一个时间点，贪婪需要集群事件类型，而我们的方法只需要计算问题事件发生的能力，这是更有效的。在CLU-DEP和RAN-DEP，我们需要更新相关树在每一次迭代，使之低于CLU-IND和CLU-DEP。隐藏状态的数量K，在效率上比m有更大的影响，因为在我们的方法，每一轮的时间复杂度为O（k2）。作为迭代轮，它稍微地不同于集群•基于方法和随机方法。一般来说,前者迭代3 - 4轮,后者迭代5 - 6轮。因此,塔尔运行时的方法相比较于贪婪算法。

7.3 真实数据实验

在本节中,我们进一步测试我们的方法在真实场景中的性能。通过使用Windows XP管理的事件日志,我们表明,我们的方法不仅可以改善压缩比,还发现一些隐藏在事件日志中有意义的事情。

真实数据集包括Windows XP事件查看器产生的系统日志。它包含56种不同的事件类型（m=56）。日志数据从2009年3月到2009年7月跨越了一段时间。它包括11365事件发生。我们从日志文件中提取了4个数据集。第一个，用“Original”表示，一个事件发生中包括至少6200个时间戳。每个时间戳对应于1秒。为了测试事件类型之间的相关性的方法,我们构建3个其他的基于原始数据的数据集。我们把整个跨度分成间隔序列与固定的时间。在第一个数据集中，用MIN表示，时间是一分钟。它包含120829分钟(N = 120，829)。在第二个数据集中，用10MIN表示，时间是10分钟(N = 12，083).在第三个数据集中，用HOUR表示，时间是一小时（N=2,000）。表3显示了四个数据集的压缩比，被比作GREEDY。我们的方法显然已经降低压缩比与贪婪算法相比,尤其是在10分钟数据集和小时数据集。

Data	Original	MIN	10MIN	HOUR
GREEDY	1	1	1	1
CLU-IND	0.912	0.738	0.594	0.732
RAN-IND	0.943	0.762	0.621	0.712
RAN-DEP	0.826	0.619	0.512	0.528
CLU-DEP	0.791	0.595	0.486	0.471

表三：对日志数据的压缩比

此外,我们发现了一些有意义的状态。例如，在MIN数据集中，状态4对应电脑开始的时期，此时事件6005和6009经常同时发生。它的持续时间大约是5分钟。状态22对应上网的时间，此时事件2504和事件4021经常频繁发生。它的时间总是超过100分钟。其他状态包括关闭, windows更新・・・。结果验证该模型可以找到系统的隐状态。

相反的，贪婪算法只是把事件序列分成了三个间隔。从它的结果中很难得到有意义的内容。

8.结论

在本文中，我们从它的事件日志学习动态系统。我们学习了一组模型，而不是分割事件序列，并描述每个分部与当地的模型，我们学习了一套具体系统的状态。此外，我们学习不同状态之间的关系。这有效的产生了一个描述系统的HMM。我们展示了基于最小描述长度原则的事件数据总结的任务可以通过从事件数据中学习HMM来实现。在本文中，我们提出了重复的方法来学习HMM。我们还利用不同事件类型之间的相关性,进一步改善总结。我们测试的合成数据集和真实数据集所提出的技术。实验结果表明，我们的方法不仅总结了事件序列，也揭示了隐藏的系统中生成事件的状态，同时我们更好地理解系统。