电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

学 士 学 位 论 文

BACHELOR DISSERTATION

论文题目 **基于同步原理的多边算法聚类研究**

学生姓名 **高崇铭**

学　　号 **2012001010016**

专　　业 **计算机科学与技术**

学　　院 **英才实验学院**

指导教师 **邵俊明**

指导单位 **计算机科学与工程学院**

2016 年 **06** 月 **02** 日

摘 要

近年来，双边聚类成为了生物信息学、文本挖掘和推荐系统里广受关注的问题，越来越多的研究工作围绕其展开。目前存在的相关算法大多以启发式搜索的方式来找寻数据矩阵中的联合簇，搜索指标的选取决定了这类算法的性能。在本文中，我们提出了一种全新的基于同步思想的双边聚类算法CoSync，其工作方式为用动态交互的方式来自动第发掘嵌入数据矩阵中的联合簇。CoSync算法的核心思想是将数据矩阵视为一个动态系统，矩阵中每一个元素的值都将受到其邻居元素值的加权影响，随着迭代属于同一联合簇子矩阵的元素值终将同步到一起，构成一个常数值子块。我们将展示CoSync吸引人的优势：(a) CoSync可以根据数据内在的结果，自动识别出高质量的联合簇。(b) CoSync对联合簇在数据矩阵中的位置分布没有限制，其可以发掘现实数据中分布复杂的联合簇。(c) 加入非负矩阵分解的模块后，CoSync可以对任意高维数据矩阵进行分析。最后，实验证明我们的算法能够成功发掘出数据集中高质量的联合簇，且性能超过国际上有代表性的其他算法。

关键词：双边聚类，同步，基因表达数据

ABSTRACT

Co-clustering has gained growing attentions recently due to its wide practical applications in biological data analysis, text mining and recommender systems. Most existing co-clustering algorithms usually search co-clusters by heuristic searching algorithm, the performance depends on the choosing of criteria. In this paper, we propose a new synchronization-inspired co-clustering algorithm by dynamic simulation, called CoSync, which aims at discovering all co-clusters embedding in a given gene expression data matrix. The basic idea is to view the data matrix as a dynamical system, and the weighted two-sided interactions are imposed on each entry of the matrix from both aspects of rows and columns, resulting in the values of all entries in a co-cluster synchronizing together. We demonstrate that our new co-clustering approach has several attractive benefits: (a) CoSync is capable of identifying co-clusters with high-quality, driven by the intrinsic data structure. (b) Without any co-cluster structure assumption, CoSync supports finding co-clusters of arbitrary size, not limited to disjoint co-clusters. (c) In conjunction with non-negative matrix factorization, CoSync allows analyzing large-scale data. Experiments show that our algorithm faithfully uncovers co-clusters embedded in data sets and has good performance compared to state-of-the-art algorithms.

Keywords: Co-clustering, Synchronization, Gene Expression Data

目 录

第1章 绪论

1.1 引言-从数据挖掘谈起

现如今，我们处于一个充满数据的时代。在每一天我们使用计算机、手机时候，都有大量数据产生，接着被以各种形式记录、保留下来。许多数据都给我们的生活提供着极大的便利，比如根据用户的音乐的历史记录，音乐软件能推荐更多的适合该客户口味的音乐；再比如当我用键盘输入这段文字的时候，中文输入软件根据词库里的数据，将键入的字母序列转换为一段段可能性最大的中文词汇或句子。在经济学、医学、生物学、社会科学等学科中，海量的数据无时无刻不在产生，然而如何合理地利用这些数据，使其提供我们需要的信息，成为了现在各个领域面临的问题。

在这样多领域的需求下，数据挖掘（Data Mining）这门交叉学科应运而生。通常来说，数据挖掘是数据库知识发现（Knowledge-Discovery in Databases）中的一个步骤，其目的是在大量的数据中自动搜索隐藏于其中的特殊信息，从而为之后的分析决策提供理论依据。下面将简要介绍下数据挖掘的主要步骤：

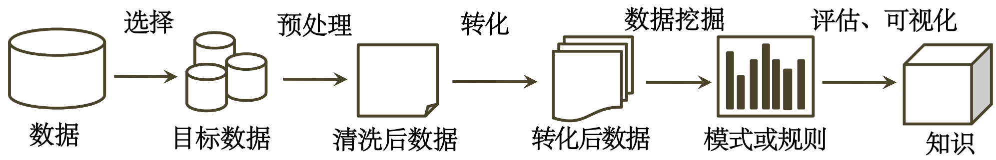


图 1-1 数据挖掘主要步骤图（来源：Synchronization Inspired Data Mining[1]）

- **数据采集** 所有工作开始之前，首先需要采集数据，包括确定数据种类、范围等，然后对数据进行初步选择，挑选出合适的数据。
- **数据预处理** 该过程包括对原始数据的处理，包括数据整合、去除噪声等。
- **数据转化** 对数据进行完预处理后，需要决定数据合适表示，例如特征选筛等。
- **数据挖掘** 这个过程中，人们采用各种方法，例如聚类、分类、关联规则分析等方法来发掘数据中的有用的信息。

- **结果评估与可视化** 最后，需要对得到的结果进行解释与评估，并可视化为易于人理解的形式，在这之后有可能需要重新进行挖掘。

这其中，**数据挖掘**是从数据中学习知识的最关键的步骤，因此很多时候，数据挖掘泛指从数据中学习知识的过程。数据挖掘的大量算法可以按照目的分为以下四类：

- **分类（Classification）**分类算法的目的是为特定变量确定类别或者标签，比如根据近年来我国的经济发展情况来确定房价是涨还是跌。一般来说，分类首先用历史数据作为训练集，学习出目标函数，然后用学到的目标函数来预测新来的未知数据点的类别。常见的分类算法有kNN[2],决策树[3],支持向量机[4]等。
- **聚类（Clustering）**聚类算法的目的是将数据分为许多类，使得相似的数据分在同一类中，不相似的数据分布在不同的类中，比如菜农可以根据一批辣椒的形状、辛辣程度将其聚拢成不同类别销售。常见的聚类方法有k-means[5], spectral clustering[6]和DBSCAN[7]等方法。
- **关联规则分析（Analysis of Association Rule）**关联规则分析的目的是从数据中发现经常出现的模式，一个经典的例子是人们从超市的大量销售记录中发现买尿布的人也常常买啤酒。经典的关联规则分析方法有：Apriori[8]和FP-growth[9]等。
- **奇异点检测（Animation Detection）**奇异点检测的目的是发现数据集中存在的奇异点，即与大多数点不相似的少数数据点，比如邮件代理公司会根据正常邮件与垃圾邮件的特征对比，来为用户标记垃圾邮件。通常来说大多数聚类算法都可以作为奇异点检测算法。

相对于数据挖掘的其他算法，聚类的知识目前还不够系统化。一个重要原因是聚类不存在客观标准：给定数据集，总能从某个角度找到以往算法未覆盖的某种标准从而设计出新算法[10]。但聚类技术本身在现实任务中非常重要，近些年关于聚类的新算法在数据挖掘、机器学习、人工智能的顶级会议乃至《自然》和《科学》上都频出不穷。本文也将提出一种全新的基础聚类算法，在此之前，先引入由特殊需求引入的新型聚类技术：双边聚类技术。

1.2 双边聚类技术

聚类技术有很多变种，其中双边聚类（Co-Clustering，或Bi-Clustering，Twomode clustering）就是一种，其致力于突破传统聚类的限制，在两个空间中同时进行聚类，从而创造更好的应用价值。现在首先来谈谈双边聚类是什么，随之引入一个正式的双边聚类的问题描述。

1.2.1 “同时聚类”需求的产生

在传统的聚类中，对于一个数据集，总是给定一个特征空间，对数据集进行聚类。比如在传统的“用户-商品”推荐系统（Recommondaton System）中，想对用户进行聚类，那就要将各种商品作为特征集，通过不同用户喜欢的商品集合的异同来判断用户之间的相似性，从而最终达到对用户聚类的目的。同样的，如果想对商品集合进行聚类，那反之得将商品集合作为数据集，用户作为特征集，对商品进行聚类。至于聚类的目的，对同一个用户簇可以推荐相同的商品，而同一个商品簇可以归类到一起进行管理，这是后话了。

同样的，对于文本挖掘（Text Mining），大量的词汇和文档也可以组成两个空间，将其中一个空间作为特征集，对另外一个进行聚类，此类例子还有很多。那么，有的时候，我们不禁发问：能否同时对两个空间进行聚类？比如在推荐系统中同时对用户集合和商品集合进行聚类，于是在得到相似的用户的群组的同时，得到该群组喜欢的商品集合！同样地，在文本挖掘中我们是否可以在得到相似的文本集的同时，得到该文本集包含的词汇集？

要是这种两个空间“同时聚类”的问题能够解决，那么在大量应用中将得到极好的结果和可解释性，甚至颠覆传统聚类方法的价值，从而为科研和生产提供全新的方向。事实上，现在已经有大量的双边聚类算法诞生并且投入应用，先来看看最初的双边聚类算法是怎么出现的。

1.2.2 双边聚类问题描述

双边聚类问题[①]诞生于生物信息学中的基因表达问题（Gene Expression Profiling）上，简单来说，近年来生物信息检测技术的进步为科学家们提供了大量的基因表达数据，即大量的基因在不同样本(不同个体、不同组织或者不同环境)的表达的程度，可以用一个“基因-样本”的矩阵A来表示，其中的元素a_{ij}表示编号为i的基因在样本j下的表达程度，数值越大表达的程度越大。

① https://en.wikipedia.org/wiki/Biclustering

依据传统的聚类技术，我们可以将基因进行聚类，体现在矩阵中形成横向的簇，如图1-2(a)所示；也可以将样本进行聚类，体现在矩阵中形成纵向的簇，如图1-2(b)所示。

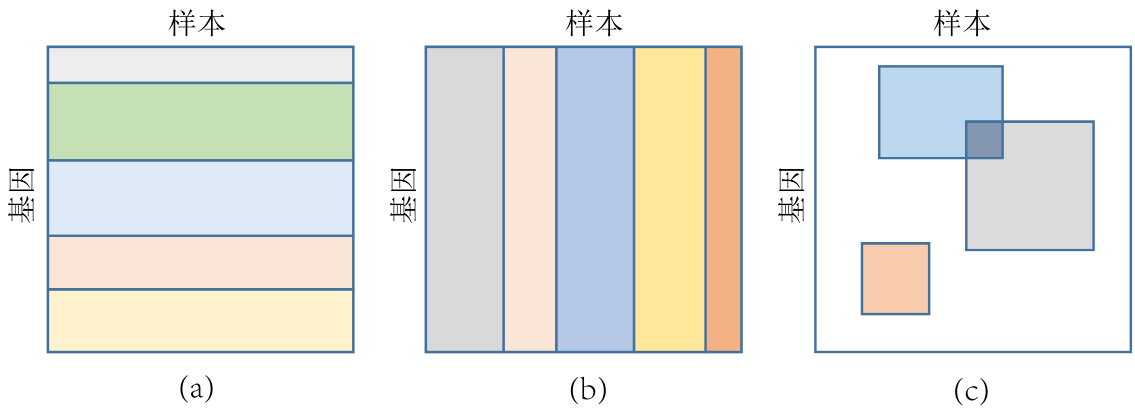


图 1-2 数据挖掘主要步骤图

然而在生物信息学中，我们最需要的信息是哪些特定的基因在哪些条件下会一起比较大程度的表达，从而在后面的转录、翻译程序中形成我们关心的RNA、蛋白质。此时我们需要的聚类结果如图1-2(c)。

根据以上的基因表达问题，我们可以正式定义双边聚类这一问题，首先我们定义联合簇：

定义 1.2.1 (联合簇(Bicluster))　在任一$M \times N$的关联矩阵A中，行的集合为V，列的集合为U。任取V的一个子集I，U的一个子集J，则I和U可以组合成矩阵B，使之成为A的一个子矩阵。这个矩阵B即为一个联合簇。

联合簇具体的形式和产生机制因算法不同而不同，在后一章节的第2.1节将会详述。而双边聚类即找到矩阵A中所有的联合簇（子矩阵）的过程。

这一类问题，早在2002年，就被Tanay及其同事[11]证明为NP难问题，故大部分的算法都采取启发式的搜索的手段来解决此问题。关于联合簇的具体形式和条件，在不同的方法中，有不同的定义。更多的信息可以参看这篇2005年的综述[12]。关于相关算法细节介绍将在后一章节的第2.2节中介绍。

1.3 本文主要贡献与创新点

本文的核心工作是提出了一种算法CoSync,以全新的双边聚类思维方式：动态的方式，使矩阵中元素进行自发地变化达到聚合，从而得到联合簇。这种方法区

别于目前所有的双边聚类算法，细节将在第3章中进行介绍。其贡献与创新点如下：

1. **全新的视角**：CoSync找寻联合簇的方式非常规的启发式搜索，而是利用全新的视角：动态模拟（dynamic simulation）。该方法建立在两个聚类空间的加权交互上，使得隐藏在矩阵中的、能体现相关生物学模式的联合簇能够直观地自动浮现出来。
2. **抓住内在自然结构**：CoSync方法对数据集的结构和联合簇的形状没有假设限制，其工作原理使得找寻到的联合簇是严格的数据本身内在的结构体现。
3. **克服了高维诅咒**：对于高维矩阵，CoSync结合了非负矩阵分解（NMF）的相关技术，将其分解为两个低维矩阵，同时保留了主要信息。此举使得高维矩阵的计算原本极高的时间复杂度降低了不止一个量级，从而让CoSync可以对大规模真实数据集问题求解。

1.4 本文的结构组织与章节安排

本章从数据挖掘学科讲起，聚焦到具体的子分支：聚类问题上，由特殊的“同时聚类”的需求引入了双边聚类问题，并做了正式的定义。接下来章节将安排如下：

- 第2章为相关工作，其将对目前所有的双边聚类算法做一个简要的综述，根据其定义的联合簇结构和算法工作原理将其分成几个分支分别介绍，并指出它们算法的内在缺陷。同时还将介绍本文动态思想的来源：Sync算法的思想和工作原理。
- 第3章为本文的主体方法，分布提出了CoSync算法中包含的模型：加权双边交互模型、最大同值子矩阵搜索算法和非负矩阵分解算法。
- 第4章为本文的实验验证部分，首先指定算法工作的人工数据集和真实数据集，之后定义衡量算法好坏的评价指标，接着用一系列的实验来说明CoSync算法的效果，并对每个实验结果进行说明。
- 第5章为总结和展望部分，总结了这篇文章的主要工作，给出客观的评价。最后给出了本工作没有涉及的部分和之后可以继续深入做下去的一些工作。

第2章 相关工作简介

2.1 联合簇的不同形式分类

由于没有一个统一的标准，联合簇在不同双边聚类算法中有着不同的定义，从而有了不同的应用场景。现在来归纳性的总结联合簇的不同模式，详细的归类可以参加这篇综述[13]。

在定义1.2.1中我们已经给出了联合簇的表示方法，即矩阵B，I和J分别为其行、列集合，其可以表示为以下形式：

$$B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1|J|} \\ b_{21} & b_{22} & \dots & b_{2|J|} \\ \vdots & \vdots & \ddots & \vdots \\ b_{|I|1} & b_{|I|2} & \dots & b_{|I||J|} \end{pmatrix}$$

其中b_{ij}代表联合簇中第i行第j列的元素。为方便后面的表达，定义b_{iJ}表示矩阵B第i行的均值，b_{Ij}表示第j列的均值，b_{IJ}表示整个矩阵B的均值。

2.1.1 根据联合簇的值进行分类

根据联合簇中的值，Kriegel[14]将它们初步分为四种类别。现在分别展开介绍。

- **常数矩阵模式**：这种情况中联合簇B矩阵中所有值均为同一常数：$b_{ij} = \pi$。
- **常数行（列）模式**：联合簇B矩阵不再是同一常数，而是其中每一行（每一列）为同一常数。可以认为产生这种情况的原因是在上一种情况：常数矩阵模式中的常数π上进行以行(列)为单位加上一个常数β或者乘上一个常数α，可以表示成如下形式：

  -加常数模式：$b_{ij} = \pi + \beta_i$，或者$b_{ij} = \pi + \beta_j$。

  -乘常数模式：$b_{ij} = \pi \times \alpha_i$，或者$b_{ij} = \pi \times \alpha_j$。
- **行列耦合模式**：这种情况下的联合簇B矩阵同时被行与列影响，同样也分为行影响与列影响。

  -加常数模式：$b_{ij} = \pi + \beta_i + \beta_j$。

-乘常数模式：$b_{ij} = \pi \times \alpha_i \times \alpha_j$。

- **耦合演化模式（Coherent evolutions）**：这是最复杂的一种情况，在这种情况下，联合簇B矩阵里的模式不能用常数或者行或列加上、乘以一个常数来刻画，而是用更复杂的关系式来定义。比如不同行或不同列之间线性相关或负线性相关。

需要说明的是，以上几种定义不是互斥的，一个联合簇可能同时满足以上几种关系。这种定义方式是方便的，Aguilar[15]也从另一角度中使用了与此相同定义方法。

2.1.2 根据联合簇排列方式进行分类

相比起上一种分类方法，现在这种分类方式是从联合簇的排列方式这个角度出发的。这种分类方式直接决定了一个双边聚类算法能够解决问题的种类。几种联合簇的排列方式如下：

- **行列全覆盖模式**：在整个矩阵A中，双边聚类得到的所有联合簇的行（列）集合必须覆盖原来矩阵A的所有行（列）。
- **非全覆盖模式**：现在的得到的若干联合簇不必覆盖原矩阵A的所有行（列）集合。这种放宽限制的方式使得联合簇的定位更准确。
- **行列互斥模式**：双边聚类得到的联合簇的行（列）集合中，不能出现交集，体现在各个联合簇的子矩阵不能有重叠（Overlapping）部分。
- **非互斥模式**：不要求互斥模式的条件，得到的若干聚类簇的矩阵可以有重叠的部分，在一些特定问题上的解释性更强。

以上就是关于联合簇的概念和分类介绍，接下来我们将简单回顾国内外所有双聚类算法的原理。由于篇幅有限，本文不会设计它们的原理细节，只是将他们进行大概的分类。

2.2 双边聚类算法简介

双边聚类这个概念最早是在1972年被J.A.Hartigan[16]提出，直到2000年，Cheng和Church[17]才正式提出第一个双边聚类算法，其应用就是基因表达数据，直到今天，他们的方法对新理论的提出都有着重要的参考价值。继那之后，双边聚类问题得到了大量的关注，更多优秀的算法随之被发明了出来。

Tanay等人[11]证明了双边聚类问题为NP难问题，其复杂度远高于一般的聚类问题，故双边聚类几乎所有方法都是基于启发式搜索的最优化问题。在这些启发式搜索的方法中，合适的代价函数和搜索策略决定了算法的有效性。当然，也有少数方法的思想不是基于这样的启发式搜索，在这些方法里存在着各式各样地的策略和算法概念。接下来就将分别介绍两种思想下的方法体系：

2.2.1 基于启发式搜索的双边聚类算法

对于任意NP问题的求解，暴力搜索的的方法是不可取的，否则时间复杂度会随着数据规模的增加而指数或者更快地上升。通常这类问题都会用启发式的搜索来求解，必须有一定的评价指标（evaluation measure）来引导搜索的方向。在双边聚类问题里，不同的启发式算法或基于不同的评价指标作为目标函数，或采取不同的优化方案作为搜索策略。本文列出了几种主要的算法类别，并不代表所有的双边聚类算法：

(1) 贪心迭代式搜索算法

贪心算法的策略即用迭代的方式向最优解逼近，其中每次迭代都取本次的最优解，算法最终在有限时间内结束。最能体现这种思想的就是“同时聚类”思想创始人Hartigan[16]提出的直接搜索法（Direct Clustering），其工作原理就是用分治法，将原始矩阵不断分为子矩阵，最终收敛得到联合簇。之后Cheng和Church[17]提出了矩阵的最小平方残差（Mean Squared Residue）作为指标搜索，之后Mukhopadhyay等人[18]改进MSR 为SMSR(Scaling MSR)后又进行更为深入的研究。Yip等人[19]提出了HARP算法，利用RI(relevance idnex)因子对矩阵进行自动的，层次性的搜索。除了普通的贪心迭代式搜索，还有在目标函数中加了随机扰动项的随机迭代式搜索，如Yang等人[20]提出的FLOC算法等。相关算法还有很多，不再列举。

(2) 自然仿生式搜索算法

自然仿生式的方法的思想都是受一些自然规律启发而发明的，在双边聚类问题中，Bryan等人[21]基于模拟退火[22]的优化方式进行搜索，Liu等人[23]基于2011年最火的粒子群优化算法[24]，用模拟鸟群和鱼群集群觅食的方式来指导搜索策略。Coelho等人[25]提出了基于人工免疫系统[26]的算法，利用记忆性来进行搜索。除此之外还有一些基于进化计算的算法，不再列举。

(3) 基于传统聚类扩展的算法

这一类算法没有从新的角度入手，仍然是从传统聚类的方法，对矩阵的行空间空间进行聚类，但用比较巧妙的方式将列空间也考虑进去，于是形成了联合簇。Cano等人[27]就提出了一种基于奇异值分解（SVD）的方法，给矩阵降维聚类的同时，记录上降维后的特征空间，于是每个降维后的聚类簇都能与其特征空间关联起来形成联合簇。Yan等人[28]对矩阵的两个维度分别进行层次聚类，然后用聚类簇的“稳定性”将它们关联起来，找到聚类簇。

2.2.2 非度量式的双边聚类算法

这里将介绍一些并非基于某评价指标搜索的算法，我们根据算法最核心部分与各个领域的关联将它们分为三类，但这种划分并不互斥，各个算法只是归于我们认为的最相关的部分。

(1) 基于图的算法

将图论和聚类联系起来的工作，最早起源于2000年Shi等人[29]的研究，也就是谱聚类的起源文章。基于图的聚类方式将测度空间转换到度量空间，从而巧妙地与图联系起来，进一步将聚类问题转化为图论中的优化问题，使得聚类问题的效率和准确率都大大提高。在双边聚类问题中，Tanay等人[11]提出了SAMBA算法，将矩阵的两个维度转换为图的节点，进而将双边聚类转化为二分图划分问题，巧妙地得到了联合簇。而另一算法QUBIC[30]则预先将矩阵化为离散值，得出不同行、列之间的相似度后，用谱聚类思想得到聚类簇。

(2) 基于概率论的算法

概率论始终是各个领域中不可缺少的一个数学分支，生活中也充满了各种概率问题，事实上，所有事件的发生都是有概率的，围绕着概率的计算可以讨论出很多有趣的问题。在双边聚类中，Lazzeroni和Owen[31]提出了plaid算法，其认为一个矩阵为很多联合簇，也就是子矩阵的加权叠加结果，求解联合簇的过程也就是利用约束求解一个“加权拼图”的过程。Sheng等人[32]提出了基于吉布斯采样的求解算法，其利用了一个简单的频率模型来刻画双边聚类，从而找到联合簇。

(3) 基于线性代数的算法

双边聚类问题的数据是矩阵，其聚类簇为该矩阵的子矩阵，利用线性代数里的方法，可以将矩阵代表的向量空间转化映射到另一个线性空间，使得在映射后的线性空间中找寻聚类簇变得容易。Kluger等人[33]提出的谱双边聚类就是典型，虽然数学指导思想不一样，其方法和实质和基于图的算法如出一辙。Carmona-Saez等人[34]提出的非平滑非负矩阵分解(nsNMF)则利用矩阵分解理论，讲原矩阵分解为两个子矩阵，之后分别聚类再关联起来得到聚类簇。

2.3 同步聚类Sync算法

自然界的同步（Synchronization）是一个非常神奇的自然规律，很多现象很早就被人们发现，比如蜂群，鱼群的集体移动，鸟类的迁徙等等[35]。早在1665年，伟大的数学家和物理学家，摆钟的发明者Christiaan Huygens[36]注意到到了同一个支架上的摆钟的摆动总是完美同步的，对此他解释到这一现象现象可能是空气扰动和支架微小的振动引起的。之后1975年Kuramoto[37]提出了经典的Kuramoto模型，准确地刻画了同步的物理机制。从此之后，同步便广泛地受到了人们关注，并渐渐成为了物理学、生物学、化学和社会科学的研究热点。

本文提出的双边聚类算法就是基于同步现象的，其理论采用了Shao等人[1]提出的的基础聚类算法Sync，在此简单介绍Sync的工作原理。

聚类的目的是让相似的点聚到同一个簇中，体现在测度空间中，则是分布在空间中靠的近的点集聚为一起。利用同步思想，Sync给空间中的点之间引入交互左右，使得每个点对以自己为中心、半径为ϵ内的点有一个引力的作用，引力与距离的关系用$sin(\cdot)$函数来刻画。这样随着时间流逝，引力将使点集产生位移，让靠的近的点集自发聚集为簇。现在给出Sync算法的几个核心定义：

定义 2.3.1 (点x的ϵ邻域邻居) 数据集$\mathcal{D}$中，在测度空间中任意点x的ϵ邻域邻居定义如下：

$$N_\epsilon(x) = \{y \in \mathcal{D} | dist(y, x) \leq \epsilon\} \tag{2-1}$$

其中$dist(y, x)$代表点y与x间的欧式距离。

定义 2.3.2 (Sync动态交互模型) 令$x \in \mathcal{R}^d$为数据集$\mathcal{D}$中的一个点，x_i是点x的第i维的值。将点x视为一个相位振子（phase oscillator），则值x_i在x的ϵ邻域邻居中的交互模型为：

$$x_i(t+1) = x_i(t) + \frac{1}{|N_\epsilon(x(t))|} \cdot \sum_{y \in N_\epsilon(x(t))} \sin(y_i(t) - x_i(t)), \tag{2-2}$$

其中$sin(\cdot)$是耦合函数。$x_i(t+1)$为$t+1$时刻点x第i维的值。

为了刻画整个数据集同步的程度，需要定义一个同步因子R_c：

定义 2.3.3 (同步因子) 同步因子R_c的作用是刻画Sync算法在数据集$\mathcal{D}$上的同步程度，从而结束算法迭代程序，其表示为：

$$R_c = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|N_\epsilon(x)|} \sum_{y \in N_\epsilon(x)} \mathbf{e}^{-||y-x||}. \tag{2-3}$$

随着Sync的迭代过程，同步因子$R_c(t)$将渐渐收敛至1，算法也就结束，此时点集已经自动聚类为簇，如图2-1(c)所示。

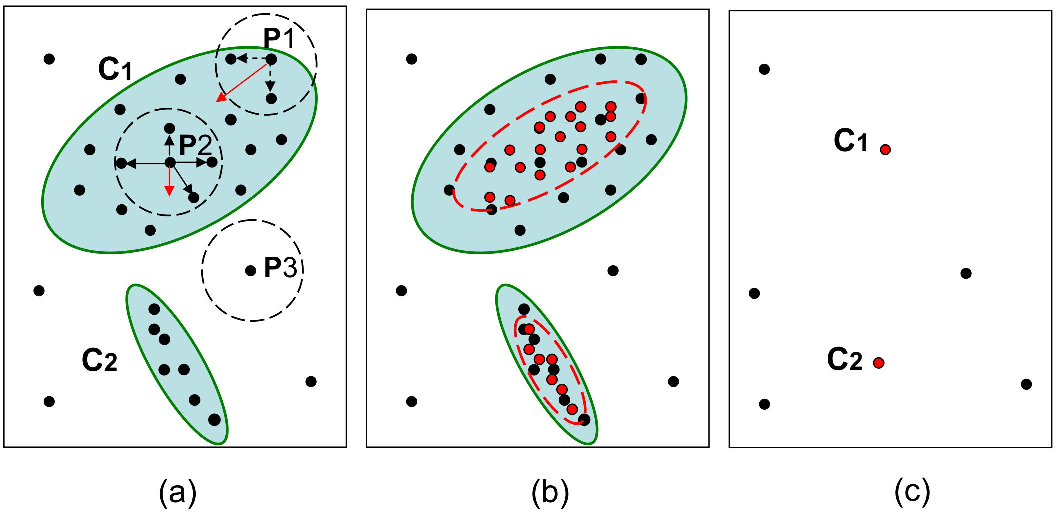


图 2-1 Sync算法聚类示意图 (a) 数据集的初始状态，黑色箭头代表点与点之间的相互交互作用，红色箭头代表点进行位移的方向。(b) Sync算法进行一段后与初态的对比图，红色部分为新的状态图。(c) 算法收敛后的最终状态，图中包含聚类簇C_1和C_2和若干离群点。（来源：Synchronization Inspired Data Mining[1]）

Sync算法作为一种动态的基础聚类算法，能巧妙地抓住了数据内在结构从而自动得到良好的聚类簇结果，说明了同步思想应用在聚类问题中的有效性。

2.4 本章小结

本章介绍了双边聚类中联合簇的各种类别，可以按照联合簇矩阵中的值或者排列方式划分。之后按是否用评价指标进行启发式搜索，将国际上比较有影响力的数十种双边聚类算法化为两类，每类中的算法按照核心思想进一步归类，对每一个子类的若干算法都进行了介绍。最后，介绍了本文引用聚类模型Sync算法的核心概念。接下来我们将介绍本文中心工作：基于Sync模型的双边算法CoSync。

第3章 动态双边聚类算法*CoSync*

3.1 双聚类问题的全新切入点

就如同上一章中回顾的那样，对于双边聚类问题已有各式各样的方法来解决。在我们这次工作中，我们将要使用一种全新的思想来看待这个问题，即*动态交互*。在这种角度下，我们把整个数据的关联矩阵视为一个动态系统，用同步交互的方式来自动地引导矩阵中值的变化。

我们用图3-1说明如何从矩阵中用这种自动的交互得到联合簇。对于一个数据集的交互矩阵A中每一个元素e_{ij}所在行和列，我们首先找寻相关的行与列，即与选定行或列距离小于一定阈值的行或列。随后找到的相关的行或列中下标对应为i或j的元素，即图3-1(b)中与红色小框同行或同列的蓝色和橘色小框所代表的元素，让它们对e_{ij}产生动态的加权交互，体现为一个“引力”，使得交互的结果趋近于缩小差值。这个过程将一直迭代下去直至收敛，收敛后对矩阵行列重新排序，得到结果矩阵如3-1(c)所示。

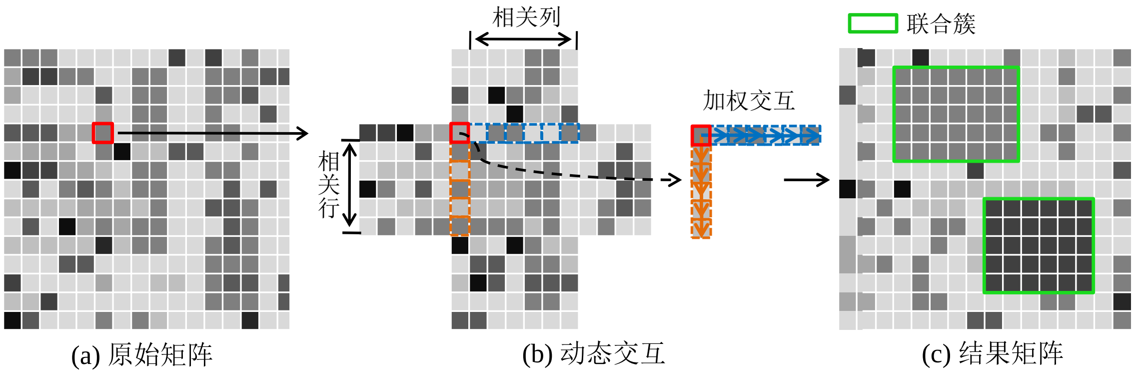


图 3-1 CoSync算法中矩阵元素在行列上的动态交互示意图 (a)关联矩阵的原始状态图，颜色深浅不一的小矩形代表原矩阵中元素值的大小，颜色越深值越小。(b) 矩阵总任意一元素的交互示意，红色小框代表的元素将被相关行列中对应的元素交互影响。(c) 算法收敛后矩阵行列排好序的最终状态，两个绿色框代表的联合簇都被自动发掘出来了。

关于交互的强弱，即“引力”的大小，应该遵循这样的原则：当某些行（或列）之间的距离越小，则说明它们之间的相似性越大，它们越倾向于包含我们想要找寻的联合簇，故越应该受到我们的关注。在这样原则的引导下，我们将对应

交互的权重调高，反之若行（或列）间的距离越大，则将对应的交互的权重调小。

这个模型的细节将在第3.2节进行详述，通过这个迭代的加权交互过程，聚类簇已经自动浮现出来了，体现为图中值全部一致的绿色方框，剩下的工作就是用算法将其识别出来。需要注意的是，搜索结果矩阵中值相等的子矩阵也不是一个平凡的问题，若暴力搜索则时间复杂度为指数函数，故我们提出一种巧妙地搜索方法，见第3.3节。

3.2 双边加权交互聚类模型

现在将正式引入模型定义，首先来对一些符号表示进行说明。

符号说明：对于矩阵$A=(A_I,A_J)$，A_I和A_J分别为矩阵A的行列集合。$a_{i\cdot}\in A_I$代表矩阵的第i行，$a_{\cdot j}\in A_J$代表矩阵的第j列。一个联合簇可以表示为矩阵A的子矩阵，用B表示，$B=A(I_S,J_S)$，其中I_S为A_I的子集的索引集合，J_S为J_S的A_J的子集的索引集合。a_{ij}对应着矩阵A第i行第j列的值。

定义 3.2.1 (行或列的ϵ邻域邻居)　给一个矩阵A和一个常数$\epsilon\in\mathcal{R}$，任意一行$a_{i\cdot}\in A_I$的ϵ邻域邻居$N_\epsilon^r(a_{i\cdot})$可以表示为：

$$N_\epsilon^r(a_{i\cdot})=\{a_{p\cdot}\big|dist(a_{p\cdot},a_{i\cdot})\le\epsilon,p\in I\} \tag{3-1}$$

其中$dist(\cdot,\cdot)$是一个距离函数，在这里用的是欧几里得距离。列$a_{\cdot j}\in A_J$的ϵ邻域邻居$N_\epsilon^c(a_{\cdot j})$按照同样方式定义如下：

$$N_\epsilon^c(a_{\cdot j})=\{a_{\cdot q}\big|dist(a_{\cdot q},a_{\cdot j})\le\epsilon,q\in J\} \tag{3-2}$$

如同Kurumoto model[37]和Sync算法[1]中的同步交互模型类似的，我们定义双边模型如下：

定义 3.2.2 (双边交互聚类模型)　对于矩阵A中的元素a_{ij}，分别找出其行$a_{i\cdot}$或者列$a_{\cdot j}$的ϵ邻域邻居，则邻居行、列中和元素a_{ij}交互的值分别为a_{pj}和a_{iq}，其中$p\in N_\epsilon^c(a_{i\cdot}(t)),q\in N_\epsilon^r(a_{\cdot j}(t))$。在$t$时刻的交互模型为：

$$\begin{aligned}a_{ij}(t+1)=a_{ij}(t)&+\frac{1}{2|N_\epsilon^r(a_{i\cdot}(t))|}\cdot\sum_{a_{p\cdot}\in N_\epsilon^c(a_{i\cdot}(t))}\sin(a_{pj}(t)-a_{ij}(t))\\&+\frac{1}{2|N_\epsilon^c(a_{\cdot j}(t))|}\cdot\sum_{a_{\cdot q}\in N_\epsilon^r(a_{\cdot j}(t))}\sin(a_{iq}(t)-a_{ij}(t))\end{aligned} \tag{3-3}$$

其中$sin(\cdot)$是耦合函数，$a_{ij}(t+1)$是矩阵元素A_{ij}在$t+1(t=(0,\ldots,T))$时刻下的取值，这个交互模型中右边两项分别刻画了矩阵的行、列交互。如同我们在前面所描述的，距离越小的行（或列）的交互影响力应该越大，我们定义一个交互权重因子如下：

定义 3.2.3 (交互权重因子) 对于矩阵元素A_{ij}的ϵ邻域行邻居$N_\epsilon^r(a_{\cdot j}(t))$，其对$A_{ij}$的交互权重因子定义为：

$$w^r(j)=e^{-\lambda\cdot\sigma_j} \tag{3-4}$$

其中σ_j为向量$\nu_{pj}=\{abs(a_{pj}-a_{ij})\big|a_{p\cdot}\in N_\epsilon^r(a_{i\cdot}(t))\}$的标准差。同样的，对于$\epsilon$邻域列邻居$N_\epsilon^c(a_{i\cdot}(t))$的交互权重因子定义为：

$$w^c(i)=e^{-\lambda\cdot\sigma_i} \tag{3-5}$$

其中σ_i为向量$\nu_{iq}=\{abs(a_{iq}-a_{ij})\big|a_{\cdot q}\in N_\epsilon^c(a_{\cdot j}(t))\}$的标准差。

双边交互聚类模型：现在基于新定义的交互权重因子，之前的模型3.2.2可以进行改写。这次我们将行、列的交互分开来写为以下形式：

$$I_{row}(t)=\frac{w^r(j)}{2|N_\epsilon^r(a_{i\cdot}(t))|}\cdot\sum_{a_{p\cdot}\in N_\epsilon^r(a_{i\cdot}(t))}\sin(a_{pj}(t)-a_{ij}(t)) \tag{3-6}$$

$$I_{col}(t)=\frac{w^c(i)}{2|N_\epsilon^c(a_{\cdot j}(t))|}\cdot\sum_{a_{\cdot q}\in N_\epsilon^c(a_{\cdot j}(t))}\sin(a_{iq}(t)-a_{ij}(t)) \tag{3-7}$$

最后，加权双边交互聚类模型可以写为如下形式：

$$a_{ij}(t+1)=a_{ij}(t)+I_{row}(t)+I_{col}(t) \tag{3-8}$$

为了衡量矩阵中的同步的程度，从而决定算法的终结，我们引入同步因子r如下：

定义 3.2.4 (同步因子) 同步因子r在随时间迭代的每一步都会被计算，以决定是否终结加权双边交互算法，其表达式为：

$$\begin{aligned} r=&\frac{1}{2|I|}\sum_{i=1}^{|I|}\frac{1}{|N_\epsilon^r(a_{i\cdot})|}\sum_{a_{p\cdot}\in N_\epsilon^r(a_{i\cdot})}\mathbf{e}^{-||a_{p\cdot}-a_{i\cdot}||}\\ &+\frac{1}{2|J|}\sum_{i=1}^{|J|}\frac{1}{|N_\epsilon^c(a_{\cdot j})|}\sum_{a_{\cdot q}\in N_\epsilon^c(a_{\cdot j})}\mathbf{e}^{-||a_{\cdot q}-a_{\cdot j}||} \end{aligned} \tag{3-9}$$

双边交互聚类模型的运行过程如图3-1所示。当$r(t)$随着时间t收敛，代表着双边加权交互模型的同步进行到了一定程度，于是可以终结迭代。

3.3 同步矩阵的同值子矩阵搜索

在上一节的双边加权动态交互迭代结束后，矩阵中属于同一联合簇的元素现已经同步为同一数值了，但由于联合簇分在矩阵中，需要设计算法将其位置找出，如式3.3所示。

$$A = \begin{pmatrix} 0.3 & 0.3 & 1.1 & 1.1 \\ 0.3 & 0.3 & 1.1 & 1.1 \\ 0.3 & 0.3 & 0.3 & 0.3 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.3 & 0.5 & 0.3 & 0.3 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

式中包含三个值：$0.3, 0.5$和1.1，现在需要一种策略，来自动对应不同值所对应的不连续的子矩阵，找出其行、列的索引集合。如前文所述，这也不是一个平凡的问题，如果直接搜索，时间复杂度随矩阵规模指数增长，使该问题成为一个难以直接求解的NP问题。

注意到在结果矩阵中能找到的同值的子矩阵非常多，不加约束的话，很多元素都会被重复出现在小尺寸的子矩阵中。这是不符合易于理解的联合簇的定义方式的，我们不要求找到所有的同值子矩阵，我们只关心整个矩阵中尺寸较大的联合簇，也就尺寸较大的子矩阵。

本着这样的思想，我们重新定义搜索问题为结果矩阵中的尺寸最大同值子矩阵的搜索问题。可以看出比起之前的搜索所有子矩阵，目前的问题只需搜索尺寸最大子矩阵是一个很大的简化。正式问题描述如下：

定义 3.3.1 (最大尺寸同值子矩阵搜索问题) 对于经过双边加权聚类收敛的结果矩阵$A = (A_I, A_J)$，要求是在子矩阵中找到最大的子矩阵$B = A(I_S, J_S) = \pi$为一常数矩阵，其中I_S, J_S分别为A_I, A_J的子集的索引集合，其尺寸$|I_S| \cdot |J_S|$的值最大。

目标函数写做如下形式：

$$\max_{I_S,J_S} \quad Size(B) = |I_S| \cdot |J_S| \tag{3-10}$$
$$s.t. \quad \begin{cases} A(I_S, J_S) = \pi \\ I_S \in I, J_S \in J \end{cases}$$

注意到由于每次搜索一个子矩阵的时候，其值π是固定的，而其他值的具体取值并不影响本次搜索。故为了方便表示，对矩阵中的每一个取值π_k都引进一个指示矩阵$A^{(\pi_k)}$，对$A^{(\pi_k)}$中的每个元素$A_{ij}^{(\pi_k)}$，其取值如下：

$$A_{ij}^{(\pi_k)} = \begin{cases} 1 & \text{if } A_{ij} = \pi_k, \\ 0 & \text{otherwise.} \end{cases} \tag{3-11}$$

则搜索原矩阵A中每一个取值π_k所对应的子块对应于搜索指示矩阵$A^{(\pi_k)}$中值为1的最大子块。如式3.3中三个值的指示矩阵如图3-2所示：

最大联合簇

	a	b	c	d
1	1	1	0	0
2	1	1	0	0
3	1	1	1	1
4	0	0	0	0
5	1	0	1	1
6	0	0	0	0

(a) 指示矩阵$A^{(0.3)}$

	a	b	c	d
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	1	1	1	1
5	0	1	0	0
6	1	1	1	1

(b) 指示矩阵$A^{(0.5)}$

	a	b	c	d
1	0	0	1	1
2	0	0	1	1
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0

(c) 指示矩阵$A^{(1.1)}$

图 3-2 指示矩阵与特定值对应的最大联合簇

从图3-2中可以明显地看出，对应于0.3的指示矩阵中最大联合簇有两个，尺寸均为6，取值为0.5的指示矩阵中最大联合簇尺寸为8，而取值1.1的指示矩阵中最大联合簇尺寸为4。

写成指示矩阵的形式后，可以看出这个问题可以类比于为频繁项挖掘问题[38]：将一行视为一个购物篮数据，一列视为一种商品。通常来说搜索频繁项问题的基础问题有两个：*Apriori*[8]和*FP-growth*[9]，在这里我们借用*FP-growth*的数据结构FP-tree。问题3.3.1的具体解决方案如下。

最大同值子矩阵搜索算法：

(1) 离散化：经过第3.2节中双边交互算法得到的结果矩阵，由于其取值为浮点数，首先要控制其精度，比如取到小数点后两位。此时矩阵A中取值范围为有限多个。

(2) 转化为指示矩阵：对矩阵A中的每一个取值π_k都引进一个指示矩阵$A^{(\pi_k)}$，例如下图为一个特定的指示矩阵，最大子矩阵为粗体标记的数字1所在的子块中。

	a	**b**	**c**	**d**	**e**
1	1	**1**	**1**	0	0
2	0	**1**	**1**	1	0
3	1	0	1	1	1
4	1	0	0	1	1
5	1	**1**	**1**	0	0
6	1	**1**	**1**	1	0
7	1	0	0	0	0
8	1	**1**	**1**	0	0
9	1	1	0	1	0
10	0	**1**	**1**	0	1
11	1	1	0	0	0
12	1	0	0	1	1
13	0	**1**	**1**	0	0
14	1	0	0	0	0

图 3-3 指示矩阵示意图

(3) 建立FP-tree：接下来的搜索将对每一个指示矩阵进行。对任一指示矩阵$A^{(\pi_k)}$，首先统计每一列包含数字1的频数，之后将矩阵的列按照频数递减的顺序重新排序，图3-3中的指示矩阵频数表统计如图(a)所示.

列号	**频数**
a	11
b	9
c	8
d	6
e	4

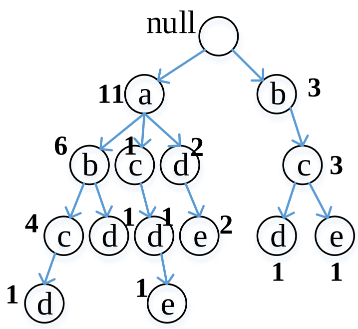
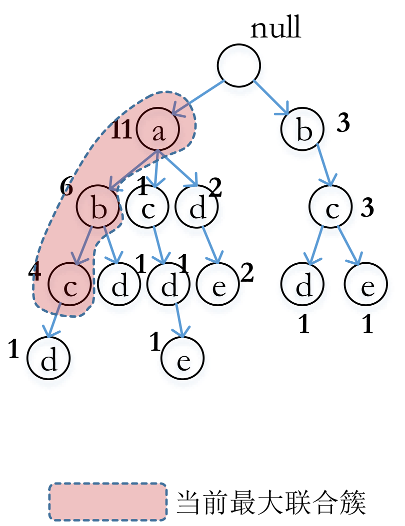


图 3-4 频数统计表和FP-tree

建立FP-tree并逐行排序后的指示矩阵，遇上数字1的时候，若树中无节点，则建立新的节点，将该节点计数$count$值设为1；否则将对应的节点$count$值加1。将该行的行号记录到节点对应的行号集合I_S中。

(4) 搜索与嫁接：这一步将反复搜索与嫁接两个步骤，直到null节点下没有任何节点停止。具体步骤如下：

- **搜索**：将null节点下按频数表的排序从最靠前的子结点开始，采用深度优先遍历，计算遍历到的节点所对应矩阵子块的尺寸$Size$，计算方式如式3-10所示。记录目前为止最大的$Size$值对应的节点。图3-5演示了搜索步骤进行到节点c，即指示矩阵的第三列时的示意图。图中c节点对应的矩阵子块行集合为$I_S = 1,5,6,8$，列集合为$J_S = a,b,c$，尺寸为$Size = 4\times 3 = 12$。



	a	b	c	d	e
1	1	**1**	**1**	0	0
2	0	**1**	**1**	1	0
3	1	0	1	1	1
4	1	0	0	1	1
5	1	**1**	**1**	0	0
6	1	**1**	**1**	1	0
7	1	0	0	0	0
8	1	**1**	**1**	0	0
9	1	1	0	1	0
10	0	**1**	**1**	0	1
11	1	1	0	0	0
12	1	0	0	1	1
13	0	**1**	**1**	0	0
14	1	0	0	0	0

图 3-5 FP-tree上的search

- **嫁接**：当搜索步骤遍历完null下一个节点分支（如图3-5中a节点分支）时候，需要去除该节点，并将该节点下所有子分支嫁接到null节点下其他分支下。如图3-6所示，图中a节点被去除去，a节点下所有分支被嫁接到null节点下的其他分支下。

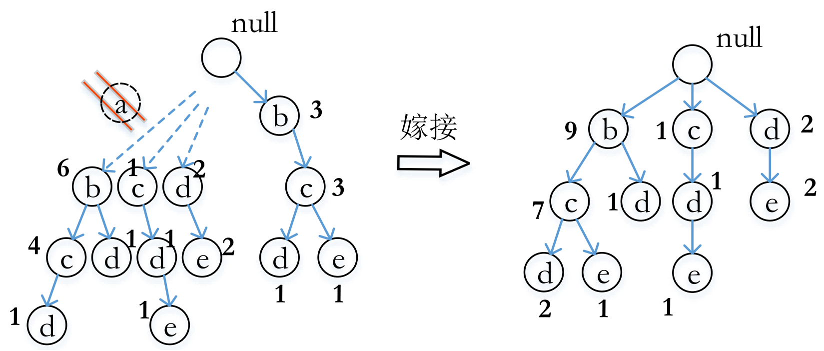
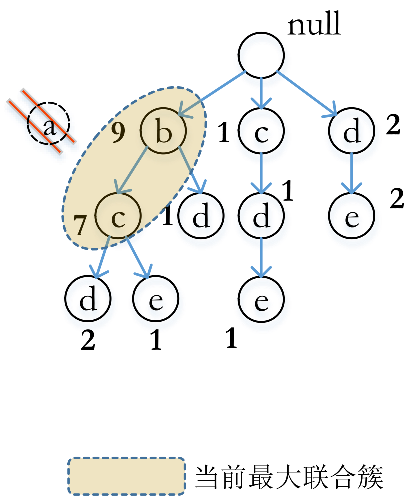


图 3-6 FP-tree的嫁接图

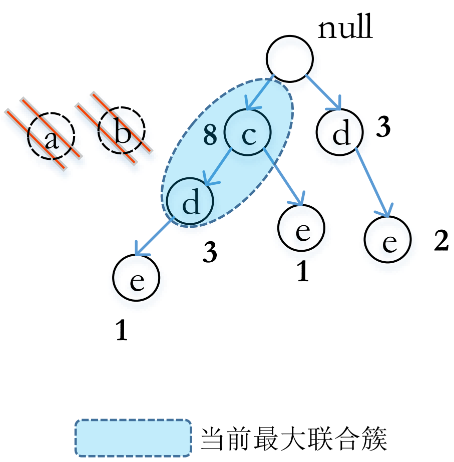
重复这这个搜索与嫁接的步骤，直到null节点下没有任何节点。图3-7,3-8演示了上图所示的标示矩阵及其对应的FP-tree在搜索与嫁接过程中的两个快照。



剪除

	a	b	c	d	e
1	1	1	1	0	0
2	0	1	1	1	0
3	1	0	1	1	1
4	1	0	0	1	1
5	1	1	1	0	0
6	1	1	1	1	0
7	1	0	0	0	0
8	1	1	1	0	0
9	1	1	0	1	0
10	0	1	1	0	1
11	1	1	0	0	0
12	1	0	0	1	1
13	0	1	1	0	0
14	1	0	0	0	0

图 3-7 FP-tree上的search。图中原本的a分支已被去除，a节点下的子分支嫁接到null节点下。这一举措对后续搜索步骤的影响为对去除第一列的指示矩阵进行搜索。



剪除

	a	b	c	d	e
1	1	1	1	0	0
2	0	1	1	1	0
3	1	0	1	1	1
4	1	0	0	1	1
5	1	1	1	0	0
6	1	1	1	1	0
7	1	0	0	0	0
8	1	1	1	0	0
9	1	1	0	1	0
10	0	1	1	0	1
11	1	1	0	0	0
12	1	0	0	1	1
13	0	1	1	0	0
14	1	0	0	0	0

图 3-8 FP-tree上的search。图中原本的a分支、b分支都已被去除，a、b节点下的子分支嫁接到null节点下。这一举措对后续搜索的影响为对去除第一、二列的指示矩阵进行搜索。

在搜索与嫁接步骤结束后，该指示矩阵下所有子块均遍历一遍，算法输出最大联合簇对应的行、列集合。

3.4 高维数据下的非负矩阵分解

对于第3.2节中描述的双边加权聚类模型，我们需要对每一行和每一列搜索相似的邻居，即ϵ邻域邻居。但是随着矩阵维度的增加，在度量相似性的时候会碰上高维诅咒（Curse of dimensionality）的困扰[①]。此时测度空间中任意两点的距离趋向同一个定值，给我们的算法带来困扰。

因此，我们在本章中介绍非负矩阵分解（NMF）的算法。它将把原矩阵转换为两个不包含负元素的低阶矩阵，且这种转换保留了大部分的信息，使得在原矩阵上的行、列的相似性度量可以转化为在两个低阶矩阵上分别做相似性度量。

非负矩阵分解：给定一个矩阵$A \in \mathbb{R}^{m\times n}$以及一个正整数$k \leq \min(m,n)$，非负矩阵分解的目的是找到两个因子矩阵：$W \in \mathbb{R}^{m\times k}$和$H \in \mathbb{R}^{n\times k}$，使得：

$$\mathbf{A} \approx \mathbf{W}\mathbf{H}^T \tag{3-12}$$

在这种情况下在原矩阵A的距离度量可以用两个新矩阵W和H上的距离度量估计。在本文中我们将采取广泛使用的弗罗贝尼乌斯范数（Frobenius norm），将式3-12的估计工作转为以下最小化求解问题：

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & f(\mathbf{W},\mathbf{H}) = ||\mathbf{A} - \mathbf{W}\mathbf{H}^T||_F^2 \\ s.t. \quad & \mathbf{W} \geq 0, \mathbf{H} \geq 0 \end{aligned} \tag{3-13}$$

用式3-13求出A的低阶估计后，原矩阵的第i行$A(i,:)$可表示为$A(i,:) = W(i,:) \cdot H^T$，原矩阵第$j$列$A(:,j)$可表示为$A(:,j) = W \cdot H(:,i)^T$，见下图：

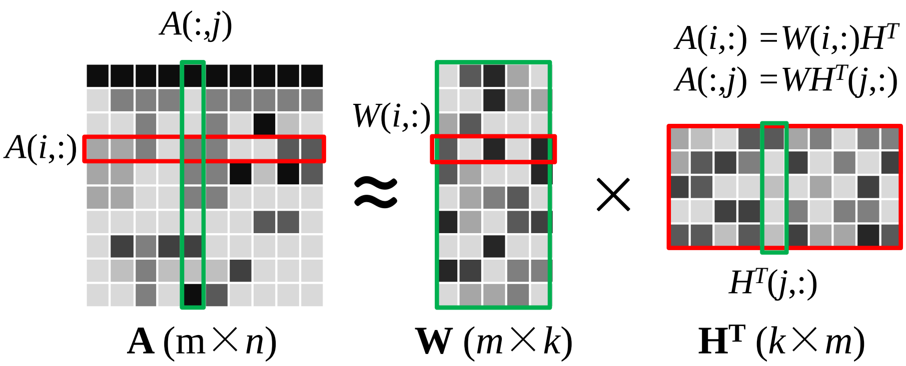


图 3-9 非负矩阵分解图

在W和H上进行相似性度量有两个显著的好处：(1)减小了高维诅咒带来的困扰；(2)大大减小函数的计算时间和空间。

① https://en.wikipedia.org/wiki/Curse_of_dimensionality

3.5 CoSync算法

以式3-8的交互模型为核心，最大子矩阵搜索算法和NMF算法为辅助，CoSync算法全过程可以总结如下几个步骤。完整的CoSync算法伪代码如算法1表示。

1.**初始化**：给定数据矩阵A，首先对其进行L2归一化。若A的维度比较高，则用第3.4节所述的NMF的算法将A分解为W和H矩阵再进行后续求解。

2.**双边加权动态交互**：对于矩阵A的每一个元素A_{ij}，我们先在矩阵A上（或者分别在W和H上）用式3-1和3-2找出其ϵ邻域邻居$N_\epsilon^r(a_{i\cdot})$和$N_\epsilon^c(a_{\cdot j})$，那对于该时刻$t$的下一时刻元素$A_{ij}$取值$a_{ij}(t+1)$将会根据式(3-8)进行求解。在同步思想的作用下，同一联合簇中的矩阵元素将趋向于取同样的值。

3.**同值子矩阵搜索**：在加权交互步骤结束后的同步矩阵将被转换为第3.3节中的FP-tree，用我们提出的最大子矩阵搜索方法求解。

3.6 时间复杂度分析

假设原矩阵A的维度为$(M \times N)$，对于其中元素A_{ij}，其行、列邻居个数分别为p, q，则CoSync算法在找寻行列邻居的计算次数为$\binom{N}{2}M(M-1)+\binom{M}{2}N(N-1)$，即复杂度为$O(N^2M^2)$。若引入NMF，将$A$转化为$W(M \times K)$和$N(K \times N)$，其中$K \ll (M, N)$，则邻居搜索的复杂度下降到$\min\left(O\left(K^2M^2\right), O\left(K^2N^2\right)\right)$。假如在邻居找寻步骤中任意一点的行、列邻居数分别为P, Q，则在双边加权交互模型中的计算次数为$MN(P(N+1)+Q(M+1))$，由于P, Q为常数，其计算复杂度为$O(MN(M+N)\times T)$，其中T为迭代次数。在最后的最大同值子矩阵寻找中，最坏的时间复杂度为N^2M。

3.7 本章小结

本章从CoSync解决问题的思想方式方式开始，详述了同步的和动态交互的工作原理（第3.1节）；接着在第3.2节详细说明了CoSync算法核心：双边加权交互聚类模型，给出了模型公式并引入控制同步迭代结束的条件：同步因子；在交互模型收敛后，得到的同步矩阵已有大量的同值子块，需要用第3.3节给出的最大同值子矩阵搜索算法来发掘这些聚类簇。面对现实情况中的高维数据集，传统的距离度量方式都会失效，第3.4节提出的NMF算法来解决这一困扰。最后，在介绍完上述所有模型和算法后，第3.5节总结了CoSync算法的完整工作流程和伪代码，第3.5节给出了算法的时间复杂度分析。

Algorithm 1 CoSync

输入: 数据集矩阵A
$A = norm(A)$; //行或列归一化
if 矩阵规模超过一阈值 **then**
  [W, H] = NMF(A); //非负矩阵分解
end if
while 循环变量为1 **do**
  // 行交互
  for 每一个行向量 $a_{i\cdot} \in A$ **do**
    if 矩阵规模超过一阈值 **then**
      在矩阵 W 上寻找ϵ邻域邻居 $N_\epsilon^r(a_{i\cdot})$;
    else
      在矩阵 A 上寻找ϵ邻域邻居 $N_\epsilon^r(a_{i\cdot})$;
    end if
    for $N_\epsilon^r(a_{i\cdot})$中每一列 $j \in J$ **do**
      用(3-4)式来计算列权重因子$w(j)$;
    end for
    用(3-6)式来计算每一个ϵ邻域邻居$a_{p\cdot} \in N_\epsilon^r(a_{i\cdot})$对$a_{i\cdot}$的交互值;
  end for
  // 列交互
  for 对每一个列向量 $a_{\cdot j} \in A$ **do**
    if 矩阵规模超过一阈值 **then**
      在矩阵 H 上寻找ϵ邻域邻居 $N_\epsilon^c(a_{\cdot j})$;
    else
      在矩阵 A 上寻找ϵ邻域邻居 $N_\epsilon^c(a_{\cdot j})$;
    end if
    for $N_\epsilon^c(a_{\cdot j})$中每一行 $i \in I$ **do**
      用(3-5)式来计算行权重因子$w(i)$;
    end for
    用(3-7)式来计算每一个ϵ邻域邻居$a_{\cdot q} \in N_\epsilon^c(a_{\cdot j})$对$a_{\cdot j}$的交互值;
  end for
  用式(3-8)来更新矩阵 A;
  用式(3-9)来计算同步因子 r;
  if 同步因子 r 收敛 **then**
    循环变量设为0;
  end if
end while
//寻找联合簇
for 对于每一个离散值 π_k **do**
  用(3.3)节提出的算法搜寻其对应的最大子矩阵$B^{(\pi_k)}$;
end for
找到所有的联合簇 B;
输出: 联合簇矩阵B

第4章 实验设计,算法实现与评估

4.1 实验设计介绍

本章中将设计实验对CoSync算法进行检验。首先我们将设计合适的人工数据集，并预先设置评价指标来衡量算法结果的好坏。之后我们将针对现实中的数据集，找寻合适可行的、流行的的数据集进行我们的实验。为了说明CoSync算法的高效和优秀，我们将采用国际上一些有影响力的双边聚类算法进行对比实验。我们选取了可获取程序的几个算法，包括：ITCC[39]，MSSRCC[40]，Spectral Clustering[33]，Plaid[31]。

在4.2节中，我们将详述怎么用高斯分布与随机分布构造人工数据集，并给出实验的结果与评估，之后与上述的几个算法进行性能对比；在4.3节中，我们将用基因表达数据来检测CoSync算法的性能，以及评价在生物信息学中联合簇是否显著的统计结果。由于联合簇在基因数据集上关于基因集没有先验标签，其生物学解释需要用有效性检验的专家知识来评估，不同算法找出的不同簇之间无法直接对比，故在真实数据上我们仅给出CoSync算法的运行结果。

此处给出本文实验平台环境：CoSync算法的实现语言为Java和Matlab，所有实验均在同一台PC机上完成，运行机器CPU主频2.3GHz，内存8.0GB。

4.1.1 数据集的选取与处理

针对CoSync算法的特性和适用条件，本文实验中人工数据集的设计以及真实数据集的选取需要尽量遵循以下几个原则：

1.**数据稠密无空值**：对于大多数双边聚类问题，空值的存在将会使得相似性度量失去意义，故本次实验的数据集将严格遵循无空值这一条件。

2.**规模适中**：用于实验的数据集矩阵维度应控制在$(50 \times 50) \sim (1000 \times 1000)$的范围内。若矩阵选取过小，则体现不出算法的功能和效率；若过大则不能在短时间内得出结果，并且给结果评价带来困难。

3.**维度比例适中**：实验矩阵的行、列数目比例应倾向于$1:1$（方阵）。若比例过于失衡，则维度的差异将给双边交互模型带来较大误差。

4.**取值归一化**：由于CoSync的交互模型中存在$sin(\cdot)$函数，故要求数据矩阵中的值的取值范围为$(-\frac{\pi}{2},\frac{\pi}{2})$。为了说明方便及后续结果的可扩展性，我们按照$L2$归一化的方式将矩阵取值控制在$(0,\frac{\pi}{2})$。

5.**易于可视化**：人工数据集的联合簇空间分布应易于可视化，即应该使同一联合簇的元素在原数据矩阵中连续分布，为了方便说明，我们将尽量使联合簇连续分布在原矩阵左上角的子区域内。真实数据集没有这一要求。

以上几条原则都是我们建议的方案，适用与CoSync算法工作的方案，对于不满足以上原则的数据集，算法结果可能将受到影响。本文人工数据集的设计按照上述方案进行设计，而真实数据集只能尽量满足以上原则。比如在文本数据、购物篮数据和基因表达数据中只有基因表达数据符合稠密性，但基因表达数据难以满足规模适中、比例适中的条件。

4.1.2 评价指标建立

在信息检索和统计学分类问题中，用来衡量机器学习相关算法结果质量好坏的统计量很多，常见的有互信息（MI）①、混淆矩阵（Confusion Matrix）②还有精确率与召回率（Precision and Recall）③等。在本文中我们将用精确率与召回率来刻画结果的好坏。

令在本次实验中对于任意一个联合簇矩阵B，用CoSync算法找出的子矩阵为$\hat{B}$，关于矩阵B精确率和召回率的概念定义如下：

精确率：对于矩阵$\hat{B}$中元素，存在于真实联合簇B中的比率，即：

$$precision = \frac{\{b \big| b \in B, b \in \hat{B}\}}{\{b \big| b \in B\}} \tag{4-1}$$

召回率：对于真实联合簇B中的元素，存在于矩阵$\hat{B}$中的比例，即：

$$recall = \frac{\{\hat{b} \big| \hat{b} \in B, \hat{b} \in \hat{B}\}}{\{\hat{b} \big| \hat{b} \in \hat{B}\}} \tag{4-2}$$

本文中精确率和召回率都将被用于人工数据和基因数据集样本集的算法评测中。但由于真实世界中基因没有明确的簇标签，故不能用此方法来进行评价。在第4.3节中我们将会介绍如何用生物学中的显著性来对联合簇中基因集进行评测。

① https://en.wikipedia.org/wiki/Mutual_information

② https://en.wikipedia.org/wiki/Confusion_matrix

③ https://en.wikipedia.org/wiki/Precision_and_recall

4.2 人工数据集上的实验设计与实现

在本节中我们将首先说明数据集的构造方法，以简单的数据集来证明CoSync算法的工作原理，接着我们将CoSync和其他算法进行相比，得出结论。

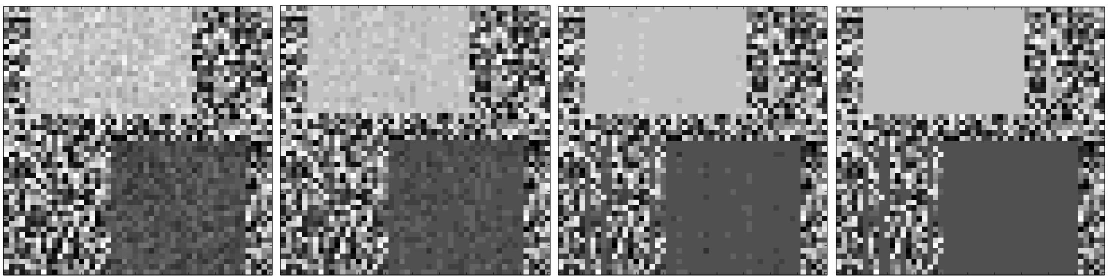
4.2.1 人工数据集构造

为了测试及证明CoSync算法的有效性与正确性，我们现在构造带有噪声的人工数据集来进行测试。遵循上一小节所述的数据集需满足的条件，这里我们将说明我们的构造方法。

数据矩阵中除去联合簇的部分取为均匀分布$U(0,\frac{\pi}{2})$，代表无意义的噪声。联合簇部分用高斯分布$N(\mu,\sigma)$来模拟。对于取值为$c(0\leq c\leq\frac{\pi}{2})$的联合簇，则高斯分布的均值取值$\mu$取为$\mu=c$。根据$3\sigma$原则 ①，一个联合簇中68%的数据将会落在$(\mu-\sigma,\mu+\sigma)$的范围内，95%的数据将会落在$(\mu-2\sigma,\mu+2\sigma)$的范围内，考虑到整个数据集的取值范围是$(0,\frac{\pi}{2})$，我们将$\sigma$的取值固定在$\sigma=0.1$。关于数据集的具体构造参数将在后面下一小节给出。

4.2.2 算法概念证明

首先我们将设计实验揭示CoSync的工作原理，观察其工作流程。我们用上一节的构造方法，取矩阵A尺寸为(100×100)，其中包含两个联合簇，其数据分别从高斯分布$N(1.2,0.1)$和$N(0.5,0.1)$中抽样，其余部分将从均匀分布$U(0,\frac{\pi}{2})$中抽样。矩阵A的可视化结果如图4-1(a)所示。



(a) 原始矩阵　(b) T = 1　(c) T = 5　(d) T = 30

图 4-1 CoSync算法动态双边交互过程图。 (a)原始矩阵及包含在其中的两个联合簇。(b)-(c)动态交互在的过程，可以看出联合簇中的值随着时间渐渐同步。(d)算法收敛的同步矩阵，可以看到两个嵌入的聚类簇被完美地发掘出来。

① https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule

可以看出图中两个联合簇的轮廓，需要注意的是，产生这种可见“轮廓”的原因是为了可视化方便而设置的，当矩阵的行、列顺序被打乱后，联合簇将无法被肉眼识别出来，但行、列顺序并不影响CoSync算法的工作。

图4-1(b)-(d)展示了CoSync算法的工作流程。T代表双边交互模型的迭代次数，可见在$T=5$次迭代处，联合簇的轮廓就已经逐渐清晰。往后的迭代中，收敛速度逐渐变慢，直到进行到30次，同步因子的改变量小于一定阈值，停止迭代得到收敛的同步矩阵。原始的两个联合簇都被完整地挖掘出来。图4-2展示了这一过程中同步因子r的收敛过程。

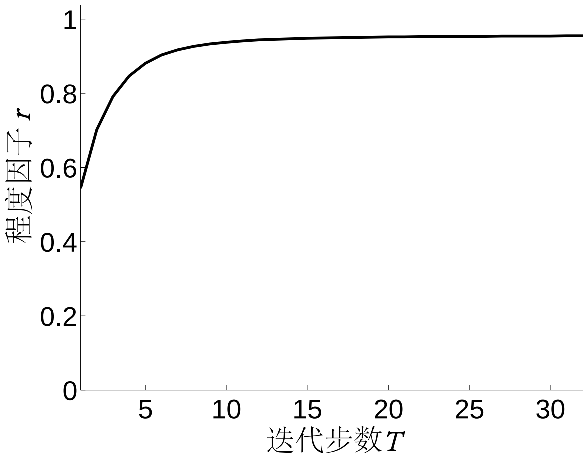


图 4-2 同步因子随时间收敛图

从图中可以看到同步因子将收敛于1，且其收敛的速度初始很快，随着迭代的进行越来越缓慢，形成一个平稳的曲线。

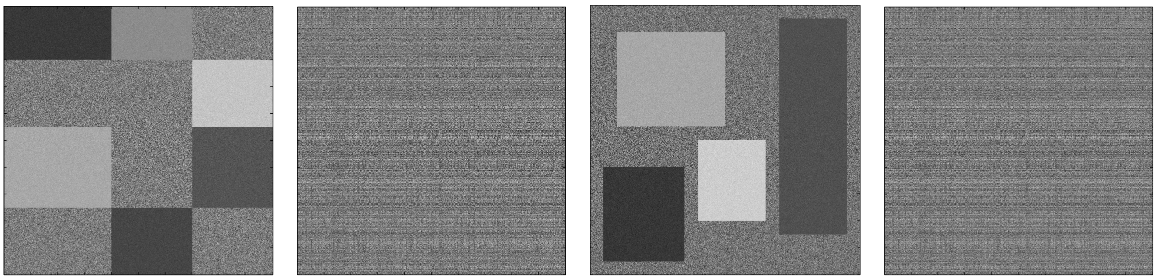
受限于篇幅的原因，关于更多CoSync在人工数据集上的结果被置于附录中。从这些实验中我们可以得出结论，CoSync算法对于原始矩阵中嵌入的少量大尺寸联合簇的挖掘结果表现良好，受动态同步的思想驱动，嵌入在数据集中的联合簇都被自然地挖掘了出来，且对参数的上下浮动的敏感性较低，具有不错的鲁棒性。但是若在数据集中嵌入的联合簇的分布复杂，且尺寸小数量多的情况下，则受限于相似性度量难以有效发挥作用，不能很好地同步在一起。

4.2.3 CoSync与其他算法的性能比较

为了证明我们的算法相对于与其他算法的优势，说明CoSync算法能挖掘出任意排列方式的联合簇，我们设计了两个代表性的(1000×1000)的中大型数据矩阵。

其中一个矩阵中的联合簇将以棋盘（checkboard）的方式分布，如图4-3(a)所示；而另一个矩阵中的联合簇位置分布任意，如图4-3(c)所示。图中矩阵里嵌入的颜色深浅不一的子矩阵从均值不一的高斯分布$N(\mu,\sigma)$中采样得到。为了消除行、列的排序对算法性能的影响，图4-3(b)和(d)是棋盘分布矩阵和随机分布矩阵行列顺序打散后的结果。所有的实验都将在乱序上的矩阵上进行。

接着我们将分别在这两个数据矩阵上，用CoSync，ITCC，MSSRCC，Spectral Clustering，Plaid进行测试。国际上很多主流算法由于自身局限性，对数据矩阵有较强的假设：联合簇的分布为棋盘式的，比如本次进行试验的ITCC，MSSRCC和Spectral CLustering算法。在这样的假设下，这些算法将分别对行、列空间进行聚类，最后把同一个簇的行列集合放到一起形成子矩阵。此外除CoSync外的四种算法都需要指定双边聚类的行、列各自划分的数目，相当于一个先验参数。下面是两个数据集分别的实验结果：



(a)棋盘分布矩阵 (b)棋盘分布行列打散 (c)随机分布 (d)随机分布行列打散

图 4-3 棋盘分布矩阵及行列顺序打散矩阵图

棋盘分布下的算法比较：在本实验中，首先针对ITCC，MSSRCC和Spectral CLustering算法的特性，对棋盘式分布的矩阵进行测试。实验矩阵尺寸为(1000×1000)，嵌入了6个从均值不同的高斯分布$N(\mu,\sigma)$中抽样的联合簇，其可视化图行列打散的的矩阵如图4-3(a)-(b)所示。由于数据集的分布为棋盘分布，故行、列的划分数目很好确定，分别为4与3。

在5种算法完成测试后，各自得到输出的代表联合簇的行、列集合。我们对它们进行排序整理，重新规划为易于观察的矩阵，其可视化结果如图4-4(b)-(f)所示。

图中红色方框为结果中识别出的聚类簇。可见只有CoSync和Plaid算法能够识别分散在矩阵中的聚类簇，其他三种对聚类簇的识别都是基于全局划分的。在图4-4(b)中，可见CoSync算法分毫不差地挖掘出嵌入原始矩阵的6个联合簇，做到

了识别度100%；而在图4-4(f)中，Plaid模型识别出了6个聚类簇中的4个，识别度为66.6%。ITCC、MSSRCC与Spectral Clustering算法用划分的方式将原矩阵划分为$4 \times 3 = 12$块，图4-4(c),(e)中ITCC和Spectral Clustering的12块子矩阵中成功地包括了嵌入的6块联合簇，而图4-4(d)中的MSSRCC算法的划分没有成功地圈出联合簇。

总体来看，在这个实验集上，CoSync表现出了良好的特性，甚至胜出了专为棋盘式数据设计的ITCC，MSSRCC和Spectral Clustering算法。

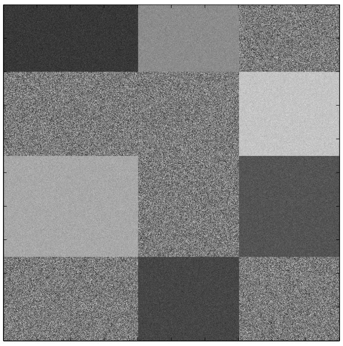
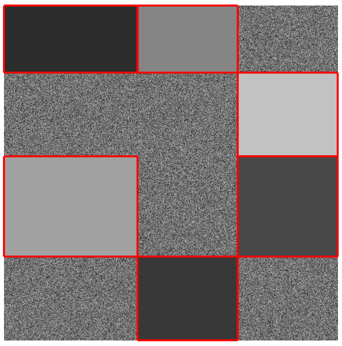
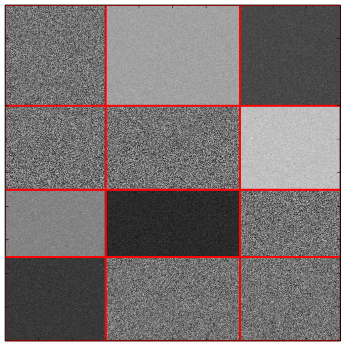
随机分布下的算法比较：在现实数据中，联合簇的嵌入不可能按照棋盘式方式规律分布，其分布是无规律而杂乱的。我们在本次实验中模拟了这种情况，将4个大小、尺寸不一致的联合簇随机嵌入(1000×1000)的矩阵中，注意联合簇间不能互相遮挡重叠。其可视化图行列打散的的矩阵如图4-3(c)-(d)所示。

同样地，5种算法在该数据集上运行完毕后，我们对各自输出的结果矩阵行列顺序整理重排，其可视化结果如图4-4(h)-(l)所示。从结果中可以看出，CoSync算法依然准确地识别出了4个嵌入的联合簇，而Plaid算法仅识别出了1个联合簇。值得注意的是，在图4-4(i)-(k)中的ITCC，MSSRCC和Spectral Clustering三种基于划分的方法此时已经失去效果。ITCC和Spectral Clustering仅划分出了一个联合簇。但由于受于方法本身的限制，它的划分不能将其余联合簇划分出来。MSSRCC则没有划分到正确的聚类簇。

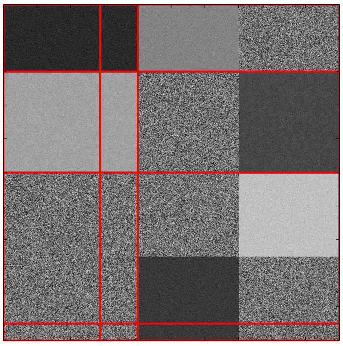
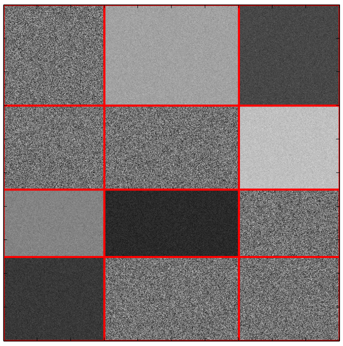
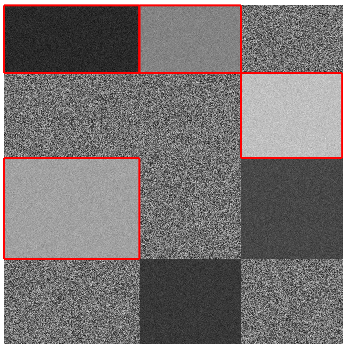
在这次的比较中，CoSync发挥了其优势：能处理以任何分布方式的聚类簇，且发掘效果显著，胜出了其余几种代表性方法。

4.2.4 人工数据集测试小结

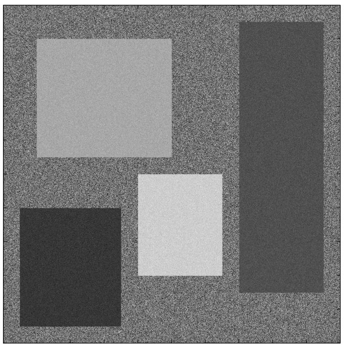
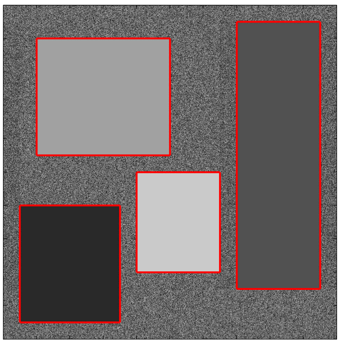
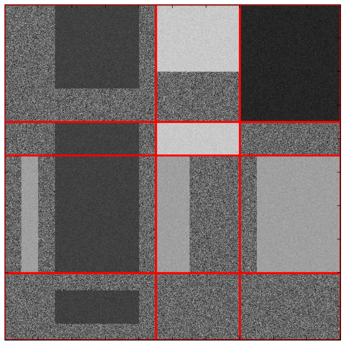
我们在本节的人工数据集测试中，首先用比较简单的(100×100)数据集证明了CoSync算法的可行性，并作出数据集在同步过程中的变化图。接下来，在联合簇以棋盘方式和任意方式分布中大规模的(1000×1000)数据集上，我们将CoSync算法和国际上比较有代表性的双边聚类算法ITCC，MSSRCC，Spectral Clustering和Plaid算法进行比较。两个的比较的结果中，CoSync都成功地识别出了矩阵中嵌入的所有联合簇，而其他算法都没有做到这一程度，这充分显现了CoSync算法在这一类数据集上的优势。总体来看，本次实验展示了CoSync算法的高效性和能抓住数据集本质结构的特性。

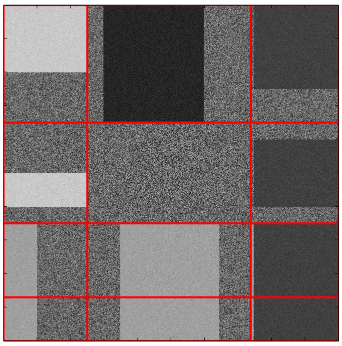
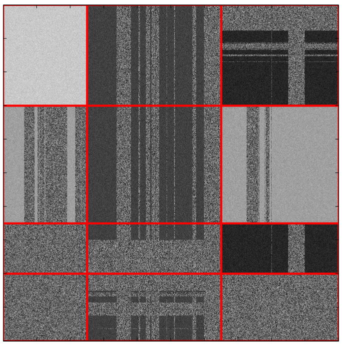
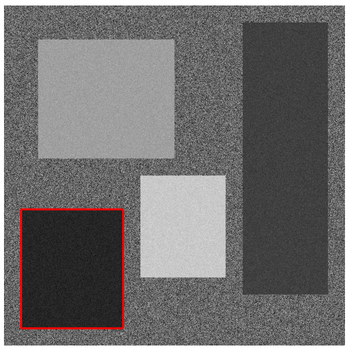
(a) 棋盘分布矩阵 (b) CoSync (c) ITCC

(d) MSSRCC (e) Spectral (f) Plaid

(g) 随机分布矩阵 (h) CoSync (i) ITCC

(j) MSSRCC (k) Spectral (l) Plaid

图 4-4 五种算法在棋盘分布矩阵和随机分布矩阵上运行结果图。(a)为棋盘分布原矩阵。(b)-(f)为五种算法在棋盘分布矩阵上的聚类结果可视化。(g)为随机分布原矩阵。(h)-(l)是五种算法在随机分布矩阵上的聚类结果可视化。图中红色方框为算法找出的聚类簇。

4.3 基因表达数据上的算法实现及评估

在本小节中，我们将在基因数据集上对CoSync算法进行测试。首先我们将介绍选本次实验选用的基因数据集，之后给出CoSync运行的聚类簇可视化结果，最后根据生物信息学中对基因的评价方法来说明CoSync算法的运行结果。

4.3.1 基因数据集选取

DNA微阵列数据是一种观测基因在特定样本上表达程度的数据①，可以用一个矩阵来表示，其中行代表基因，列代表特定样本，如不同个体，不同组织，或者不同环境下观测到的数据。

本文第一章提过，双边聚类这一领域就是在基因数据集上发展起来的，后面只有少部分方法能扩展到购物篮数据集以及文本-词汇数据集上。导致这一现象的原因是现实情况中，只有基因数据集密集无缺失值，这是距离度量的必要条件。

在本章第4.1.1节中的数据集选取原则中提到，CoSync可以处理的数据集必须是数值型无缺失值的，基因数据集满足这一条件。然而基因数据集的尺寸规模几乎都为1000×10量级，即数千规模的基因在数十个样本上的表达程度，使得矩阵的行列比例严重失调。前文说过，数据集比例失调可能会带来算法正确率降低。面对这一困扰，我们只能在实验中尽量以调参数的方式来进行应对。

本次实验中，我们选取了四个通用的基因表达数据集②，皆被国际众多知名双边聚类算法所使用，其信息总结如表4-1。

表 4-1 四个基因数据集的统计信息表

数据集	#基因数目	#样本数目	样本类别名称	样本分布
Colon	1096	62	Normal/Tumor	20/42
Leukemia	3571	72	ALL/AML	47/25
Lung	2401	181	ADCA/MPM	150/31
MLL	2474	72	ALL/AML/MLL	24/28/20

可见数据集的行列比例很不均衡。其样本类别只有两个或者三个，其中ALL、AML、ADCA、MPM和MLL均为疾病名称的缩写，而基因没有直接的类别标签。

① https://en.wikipedia.org/wiki/DNA_microarray

② 基因数据集矩阵及对应的基因名称来源:http://datam.i2r.a-star.edu.sg/datasets/krbd/index.html

4.3.2 CoSync在基因数据集上的联合簇挖掘

真实数据集与人工数据集的差别在于真实数据集的联合簇数目更多、大小更不规律、分布更不均衡，取不同的ϵ邻域邻居将得到不同层次下的联合簇。我们结合专家知识来制定合适的ϵ取值范围，在取值范围内用CoSync对上述数据集进行多次测试后，得到一些特定值的大尺寸联合簇。

我们将CoSync，ITCC，MSSRCC，Spectral Clustering和Plaid算法在四个数据集上进行了实验。如前文所述，基因空间的评价无法直接对比，其将用4.3.3中所述的基因集丰富度分析来来进行。故我们仅将对他们找出的联合簇中的样本空间进行评价对比，用4.1.2节中叙述的准确率和召回率作为评价指标。表4-2展示了四种算法在四个数据集上找到的所有联合簇在样本空间中的评价。

表 4-2 五种算法在四个数据集上找出聚类簇的样本空间评价

	CoSync			**Plaid**			**ITCC**		**MSSRCC**		**Spectral**	
	#C	Pre.	Rec.	#C	Pre.	Rec.	#C	Avg.	#C	Avg.	#C	Avg.
Colon	11	0.95	0.66	4	0.71	0.66	2	0.82	2	0.86	2	0.73
Leukemia	28	0.96	0.71	2	0.81	0.43	2	0.95	2	0.93	2	0.74
Lung	23	1.00	0.96	3	1.00	0.50	2	0.85	2	0.99	2	1.00
MLL	23	0.99	0.67	4	0.88	0.88	3	0.83	3	0.93	3	0.64

可以看到ITCC，MSSRCC和Spectral Clustering算法的由于基于划分，故其样本空间划分数目已经提前被指定好，均为数据中样本类别的真实数目。在自动找寻联合簇的算法CoSync和Plaid当中，Plaid找出的簇仅为$2 \sim 4$个，而CoSync算法在Colon数据集中找出11个联合簇，而其他三个数据集上，都发掘出了20个以上的联合簇。

关于样本的评价标准，CoSync在四个数据集上准确率都达到了令人满意的95%以上，而召回率达到了66%以上。这说明了CoSync算法找出的联合簇都能代表真实数据中的数据结构。而召回率在这重要性不是很高，因为这仅代表着算法找出的样本没有覆盖整个样本维空间。从准确率上来看，其他算法的性能都没有超过CoSync，而从召回率的角度看，基于划分的方法明显超出CoSync算法，这是合理的结果。

其次，我们还能从图中看出，数据集Lung在五个算法下都表现出了优良的性能，而Colon数据集在5个算法下的性能都不如其他数据集。这说明数据集本身的

结构会影响所有的算法效能。

为了观察CoSync找出的聚类簇的具体形态，我们为上述结果中CoSync的前十大尺寸的聚类簇进行更细的观察。表4-3列举了四个数据集中尺寸最大的10个联合簇的信息，包括该联合簇的行、列数目，以及关于样本维度上的精确率和召回率。

从表中的结果中我们可以看出来，CoSync的找寻质量还是较高的。大多数聚类簇的准确率都是1.00，在Lung数据集上甚至所有的聚类簇在样本维空间上的聚类准确率都为1.00。这一结果更肯定了我们上述的判断：CoSync算法找出的聚类簇的准确率很高。

ITCC，MSSRCC和Spectral Clustering算法这样基于划分的方法在没有标签的情况下，决定划分簇数目是非常困难的，错误的参数指定将会带来完全偏离真实情况的结果。接下来，我们将对五种算法找出的联合簇对基因维度进行评价。

4.3.3 基因集丰富度分析

我们实验数据的基因表达数据来源于对数以千的基因在特定样本下进行表达的探测。这其中样本的分类是明确的，但是基因却没有确定的类标签。为了衡量数据挖掘、人工智能算法对基因分类、聚类的结果，一种专门的衡量指标被提了出来，即基因集丰富度分析。

基因丰富度分析①，也称为功能性丰富度分析，是一种为基因和蛋白质的归类显著性进行评判的一种分析方法。基因没有类标签，但大量的实验表明在一些特定的试验下，观测基因经历一些生物过程最终表达为蛋白质的情况，一部分基因会表达得更明显，而另外的基因则被抑制或不表达。搜集这些历史记录，大量的特定的基因集合被记录下来，它们被记录在基因本体库②中。基因本体库中的特定基因集合可以分为以下三个邻域类别：

- **细胞组件（cellular component）**：细胞的每个部分和细胞外环境。
- **分子功能（molecular function）**：基因产物在分子级别的主要活动，比如结合以及催化。
- **生物过程（biological process）**：细胞内发生的，可以定义开始和结束的事件或行动。

① https://en.wikipedia.org/wiki/Gene_set_enrichment

② https://en.wikipedia.org/wiki/Gene_ontology

有了基因本体库，我们用算法对基因进行聚类的结果便有了比对的参照物。针对我们算法找出的基因数据集与基因本体中不同情况下的基因集合，用假设假设检验①的方式，便能得知我们找出的基因集合是否合理：是否完全属于某一个基因本体中的基因集合？是否太过随机，没有任何代表性？用这种方式，我们便能评价CoSync找出的聚类簇中基因空间中的结果好坏。

统计假设检验用p-value②来作为显著与否的评价指标。其概念定义如下：

显著性因子p-value：一种在原假设为真的前提下出现观察样本以及更极端情况的概率。设某假设为H_0，则p-value在H_0为真的条件下，因样本偏差太大而拒绝H_0的概率，即：

$$p-value = P(\text{拒绝接受}H_0 \Big| H_0\text{为真})$$

p-value值越小，则说明拒绝H_0的概率越小，也就是假设H_0越可信。当p-value取0时候，假设H_0与观测相比毫无差别。而当p-value值打过一定程度时候，结果就不可信了。对此人们通常设置一个阈值α，当$p-value \geq \alpha$时，便拒绝假设H_0。

在基因库丰富度分析中，令真实的一个基因集合为$\mathcal{G}$，而预测出来的基因集合为$\hat{\mathcal{G}}$，对应假设H_0的命题为：

$$H_0 = \{\hat{\mathcal{G}} \subseteq \mathcal{G}\}$$

即p-value在基因库丰富度分析中，可以作为任意给定的基因集合的显著性的指标。由此我们可以预先设置一个拒绝阈值：α，之后对每一个聚类簇的基因与基因本体中每一个基因集合对比，算出一个对应的p-value值，若其大于α，则不考虑基因本体中这个基因库。最后我们对基因本体库中的满足$p-value \leq \alpha$的基因集合列举出来。我们的算法找出的基因簇很可能就是这些基因集的子集。

值得说明的是，给定我们的基因集合，对基因本体库里任意的基因集合都能算出一个p-value，虽然这些p-value本身值越小越好，但它们互相之间是不可比较的。例如我们不能因为我们的基因集对与艾滋病蛋白相关的基因本体集的p-value比其于白血病蛋白相关的基因本体集的p-value值小，就说明我们的基因集更倾向于与艾滋病的蛋白表达相关。也正是这个原因，有三个浅显的结论：

- 不同算法间找出聚类簇中基因集合的p-value值之间不能相互比较。
- 同一算法中，不同聚类簇基因集的p-value值不能相互比较。

① https://en.wikipedia.org/wiki/Statistical_hypothesis_testing

② https://en.wikipedia.org/wiki/P-value

表 4-3 各个数据集上前十大联合簇信息。其中 P 和 R 分别代表每个联合簇在样本空间中的精确率和召回率。 $No.G.$ 和 $No.S.$分别是联合簇包含的基因和样本的数目。 N 和 T 分别代表Colon数据集中的Normal和Tumor。 A 和 M 分别代表Lung数据集中的ADCA和MPM。 AL, AM 和 ML 分别代表Leukemia和MLL数据集中的ALL，AML和MLL。

	Colon					**Leukemia**				
cID	**Size**	**No.G.**	**No.S.**	**P**	**R**	**Size**	**No.G.**	**No.S.**	**P**	**R**
1	1480	296	5(N)	1.00	0.23	3216	268	12(7AL/5AM)	0.58	0.15
2	966	138	7(5N/2T)	0.71	0.23	2570	514	5(4AM/1AL)	0.80	0.16
3	666	111	6(T)	1.00	0.15	2320	464	5(AL)	1.00	0.11
4	510	85	6(5N/1T)	0.83	0.23	1480	296	5(AL)	1.00	0.11
5	420	84	5(N)	1.00	0.13	1215	243	5(AM)	1.00	0.20
6	290	58	5(T)	1.00	0.13	625	125	5(AL)	1.00	0.11
7	205	41	5(T)	1.00	0.13	320	64	5(AL)	1.00	0.11
8	125	25	5(T)	1.00	0.13	294	49	6(AM)	1.00	0.24
9	65	13	5(T)	1.00	0.13	264	22	12(AL)	1.00	0.26
10	49	7	7(T)	1.00	0.13	242	22	11(AL)	1.00	0.23
	Lung					**MLL**				
cID	**Size**	**No.G.**	**No.S.**	**P**	**R**	**Size**	**No.G.**	**No.S.**	**P**	**R**
1	5614	401	14(A)	1.00	0.09	4228	302	14(AM)	1.00	0.50
2	4394	338	13(M)	1.00	0.42	3765	251	15(AL)	1.00	0.63
3	3960	264	15(A)	1.00	0.10	2954	211	14(AL)	1.00	0.58
4	3806	346	11(M)	1.00	0.36	2071	109	19(AM)	1.00	0.68
5	2344	293	8(A)	1.00	0.05	1584	99	16(AM)	1.00	0.57
6	2210	221	10(M)	1.00	0.32	918	102	9(AL)	1.00	0.38
7	1770	177	10(M)	1.00	0.32	890	89	10(AL)	1.00	0.42
8	1035	115	9(M)	1.00	0.29	715	143	5(3ML/2AL)	0.60	0.15
9	950	190	5(A)	1.00	0.03	533	41	13(AM)	1.00	0.46
10	420	30	14(M)	1.00	0.45	510	34	15(AM)	1.00	0.54

- 同一算法中，同一聚类簇中的基因集对基因本体中不同基因集的p-value值之间不能相互比较。

由于CoSync与其他算法找出聚类簇的基因集显著结果无法比较，故本文中我们仅对我们的CoSync算法进行基因集丰富度分析。事实上，现在已有用于基因库丰富度分析的公开网站，并且已经很成熟和快速，本次我们将使用基因本体库最齐全的Gene Ontology Consortium网站(`http://geneontology.org/`)来进行丰富度分析。表4-4列出了CoSync联合簇找出的联合簇基因集进行丰富度分析的结果，阈值α取为0.05。受于篇幅的限制，此处仅列出每个数据集中前3大的联合簇的分析结果。

表中CC和BP以及后面跟的专业名词分别代表细胞组件与生物过程中特定的物质组成部分或者生物学过程。可以看出每个聚类簇在与基因本体库中的基因集进行匹配时，p-value≤ 0.05的基因集最少在Colon数据集中能匹配到65个，最多在Leukemia数据集中能匹配539个。找出其各自p-value最小的4个进行进一步剖析，可以看到它们的p-value都小于1×10^{-3}级别。有理由相信这部分基因就是对应细胞组件或者生物过程中某特定过程的组成基因的子集。图4-5中，我们对每个数据集中大尺寸联合簇进行了可视化，由于篇幅原因，对于样本的每个类别我们只取一个聚类簇。

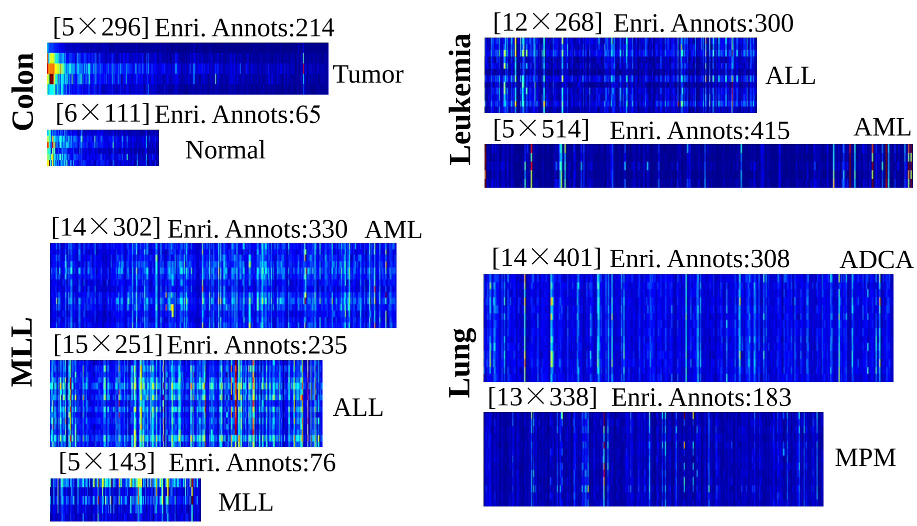


图 4-5 基因表达数据联合簇可视化图

从上面简单的评估中，我们可以看出CoSync能够找到一些高质量的聚类簇。关于对聚类基因集的更进一步权威的生物学验证，已经超出本实验的范围，本文不再展开。

表 4-4 CoSync算法找出的联合簇在基因空间上丰富度分析结果表。这里 BP 代表“Biological Process”，CC 代表“Cellular Component”。 #Annotations 代表 p-value <0.05的所有基因本体库中的基因集数目。

Data Sets	CluID	#Genes	#Annotations	Top Enriched Annotations	Count	Percentage (%)	P-Value
Colon	C1	296	214	CC:cytosol	34	23.61%	5.84E-07
				CC:plasma membrane part	42	29.17%	3.01E-05
				BP:negative regulation of molecular function	13	9.03%	7.69E-05
				BP:striated muscle tissue development	8	5.56%	1.34E-04
	C2	138	157	CC:plasma membrane part	22	37.29%	4.81E-05
				CC:plasma membrane	28	47.46%	3.87E-04
				CC:cytosol	15	25.42%	5.20E-04
				CC:actin cytoskeleton	7	11.86 %	6.69E-04
	C3	111	65	BP:regulation of system process	8	15.69%	7.55E-05
				BP:heart development	6	11.76%	7.81E-04
				BP:negative regulation of molecular function	7	13.73%	8.82E-04
				CC:striated muscle thin filament	3	5.88%	1.13E-03
Leukemia	C1	268	300	BP:response to endogenous stimulus	27	10.67%	2.47E-09
				CC:plasma membrane part	71	28.06%	4.54E-09
				CC:integral to plasma membrane	48	18.97%	5.34E-09
				BP:response to hormone stimulus	25	9.88%	7.40E-09
	C2	514	415	CC:cytosol	123	24.45%	2.08E-28
				CC:intracellular organelle lumen	142	28.23%	1.10E-26
				CC:organelle lumen	142	28.23%	1.13E-25
				CC:membrane-enclosed lumen	142	28.23%	8.10E-25
	C2	464	539	CC:plasma membrane part	114	25.79%	2.37E-11
				CC:integral to plasma membrane	73	16.52%	2.72E-10
				CC:intrinsic to plasma membrane	74	16.74%	2.95E-10
				CC:plasma membrane	160	36.20%	2.30E-09
Lung	C1	401	308	CC:cytosol	88	23.34%	4.71E-17
				CC:intracellular organelle lumen	87	23.08%	2.18E-09
				CC:organelle lumen	88	23.34%	2.96E-09
				CC:membrane-enclosed lumen	89	23.61%	3.47E-09
	C2	338	183	CC:cytosol	62	19.68%	1.74E-09
				BP:response to organic substance	37	11.75%	6.39E-07
				BP:negative regulation of apoptosis	24	7.62%	1.07E-06
				BP:negative regulation of programmed cell death	24	7.62%	1.35E-06
	C3	264	105	BP:translational elongation	29	11.69%	2.29E-27
				CC:cytosol	75	30.24%	1.13E-23
				BP:translation	39	15.73%	4.81E-22
				CC:ribosome	30	12.10%	3.84E-19
MLL	C1	302	330	BP:RNA splicing, via transesterification reactions with bulged adenosine as nucleophile	22	7.46%	1.28E-12
				BP:RNA splicing, via transesterification reactions	22	7.46%	1.28E-12
				BP:nuclear mRNA splicing, via spliceosome	22	7.46%	1.28E-12
				CC:membrane-enclosed lumen	73	24.75%	1.58E-10
	C2	251	235	BP:RNA splicing	28	11.43%	3.37E-14
				BP:mRNA metabolic process	31	12.65%	8.00E-14
				BP:mRNA processing	29	11.84%	9.54E-14
				CC:cytosol	57	23.27%	1.05E-12
	C3	211	127	CC:organelle membrane	36	17.48%	3.86E-08
				CC:organelle inner membrane	19	9.22%	6.29E-08
				CC:organelle envelope	26	12.62%	6.34E-08
				CC:envelope	26	12.62%	6.75E-08

4.3.4 基因表达数据集上测试小结

在本节中，我们说明了选用基因数据集的原因，说明了评价聚类得到的基因集的方法与标准，即基因集丰富度分析和统计假设检验p-value。我们在给定的四个知名基因表达数据集进行测试。在测试得到的聚类簇中，我们可以将样本维度的集合与已知类标签对比，结果显示其准确率和召回率都非常高，优于ITCC，MSSRCC，Spectral CLustering和Plaid算法。在基因集的评测上，我们仅对CoSync算法的聚类簇进行评析，能看到每个聚类簇的基因集都能高质量地匹配大量基因本体库中的特定基因集。这说明CoSync在基因数据集上工作的可行性与高效性。

4.4 本章小结

在本章中，我们设计了一系列实验对CoSync算法进行验证。在第4.1节中，我们给出了实验集的选取和处理的五条原则，并给出了评价结果好坏的指标：准确率和召回率。在4.2节中我们首先用一个小规模实验集合对CoSync的工作流程作了说明，接着在棋盘分布与随机分布的两个中规模数据集上将五种算法进行了比较，说明了CoSync算法的优秀。最后在4.3节中我们对基因表达数据进行测试，在样本维度上，CoSync的运行结果仍然好于其他四种算法，而在基因维度上，CoSync也能高质量匹配大量基因本体库中的基因集，说明了CoSync在基因数据集上的高效性。

总结来看，无论是在人工数据集还是现实中的基因数据集上，CoSync的结果表现都非常出色，且优于一些国际上双边聚类代表性算法。实验的成功说明了CoSync算法中同步思想的正确与高效。

第5章 全文总结与展望

5.1 全文总结

本文我们介绍了目前比较火热的双边聚类问题，提出一种基于同步原理的全新的双边聚类算法CoSync，并在人工数据集以及基因数据集上进行测试，取得了优秀的成果。各章节的主要内容总结如下：

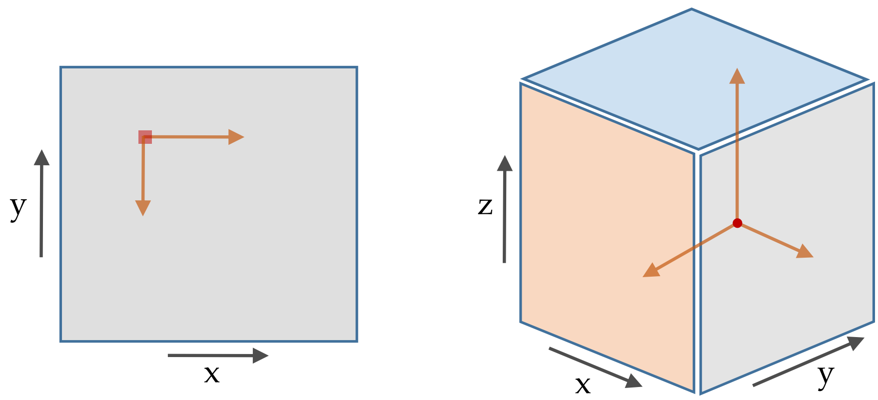
- 第1章为引言部分，从数据挖掘的聚类领域谈起，介绍了双边聚类需求的产生，并正式定义了双边问题以及联合簇。
- 第2章的第一部分整理和回顾了国际上相关的知名研究成果。介绍了从两个角度对联合簇具体形式的分类，之后对现有的大部分双边聚类算法进行了分类总结，将它们大致分为基于启发式搜索的方法和非度量式方法，针对每个方法都列举了一些算法的工作原理。接下来第二部分介绍了自然界中同步的概念以及用同步思想作为聚类原理的Sync算法。
- 第3章开始正式介绍CoSync算法的多个步骤。以同步聚类思想为切入点，我们提出了全新的双边加权交互模型，使得数据集矩阵中的联合簇能够随着时间自动收敛为常数值。在交互模型收敛后，我们提出了一种启发式的同值子矩阵搜索算法来挖掘结果矩阵中的常数值子块。最后，为了能够处理高维数据，避开高维诅咒的困扰，我们引入非负矩阵分解。至此，CoSync算法便能在大规模数据矩阵中进行联合簇挖掘。
- 第4章为实验部分，为了证明CoSync算法的可行性和高效性，我们将双边聚类中有代表性的ITCC，MSSRCC，Spectral Clustering和Plaid算法一起加入实验。实验在人工数据集和基因数据集上展开，最后结果显示CoSync具有极好的性能，超越了其他对比算法。

5.2 后续工作展望

本文中我们已经完成了关于双边聚类新算法CoSync的所有工作，之后我们将对全新的工作开展工作，即围绕**多边聚类**问题进行研究。

考虑这样一个问题，在推荐系统中，我们能得到不同用户在不同时间内，对不同商品的喜好数据，这种数据可以写为一个数据立方体A，其中任一元素A_{ijk}表示第i个用户在k个时间段内，对j个商品的评分。类比双边聚类，多边聚类即在类似的时间段内，找出相似的用户以及对应的相应的商品。

双边聚类的原理是在数据矩阵中，对每一个元素都用其行列邻居对它进行交互，最终达到同步的状态。那现在拓展这个思想，我们在一个数据立方体甚至更高维数据张量(tensor)①中，仍然用这种同步交互的思想，对数据张量中的元素进行动态交互。图5-1给出了在数据矩阵上进行双边聚类以及在数据立方体上进行多边聚类的交互示意图。



(a)双边聚类

(b)多边聚类

图 5-1 从双边聚类到多边聚类

如图5-1(b)所示，数据立方体中任一元素将被其x, y, z三个维度上的邻居影响交互，随着时间迭代最后达到同步的状态。此时的聚类簇结构将表现为数据立方体中包含的常数子块。

关于多边聚类问题，目前国际上研究的成果很少。一方面，真实世界中，不稀疏的数据立方体或者更高维的数据很难获取，另一方面，处理这样的数据困难而效率低下。我们将用同步的思想，对多边聚类问题展开研究，争取在这一领域作出成果，向国际研究前沿进军。路漫漫其修远兮，吾将上下而求索！

① https://en.wikipedia.org/wiki/Tensor

参考文献

[1] J. Shao. Synchronization Inspired Data Mining[M]. Lambert Academic Publishing, 2011

[2] L. E. Peterson. K-nearest neighbor[J]. Scholarpedia, 2009, 4(2):1883

[3] J. R. Quinlan. Induction of decision trees[J]. Machine learning, 1986, 1(1):81–106

[4] C. Cortes, V. Vapnik. Support-vector networks[J]. Machine learning, 1995, 20(3):273–297

[5] J. A. Hartigan, M. A. Wong. Algorithm AS 136: A k-means clustering algorithm[J]. Journal of the Royal Statistical Society. Series C (Applied Statistics), 1979, 28(1):100–108

[6] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm[J]. Advances in neural information processing systems, 2002, 2:849–856

[7] M. Ester, H.-P. Kriegel, J. Sander, et al. A density-based algorithm for discovering clusters in large spatial databases with noise.[M]. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996, 226–231

[8] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules[M]. Proc. 20th int. conf. very large data bases, VLDB, 1994, 487–499

[9] J. Han, J. Pei, Y. Yin. Mining frequent patterns without candidate generation[M]. ACM Sigmod Record, 2000, 1–12

[10] V. Estivill-Castro. Why so many clustering algorithms: a position paper[J]. ACM SIGKDD explorations newsletter, 2002, 4(1):65–75

[11] A. Tanay, R. Sharan, R. Shamir. Discovering statistically significant biclusters in gene expression data[J]. Bioinformatics, 2002, 18(suppl 1):S136–S144

[12] A. Tanay, R. Sharan, R. Shamir. Biclustering algorithms: A survey[J]. Handbook of computational molecular biology, 2005, 9(1-20):122–124

[13] B. Pontes, R. Giráldez, J. S. Aguilar-Ruiz. Biclustering on expression data: A review[J]. Journal of biomedical informatics, 2015, 57:163–180

[14] H.-P. Kriegel, P. Kröger, A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2009, 3(1):1

[15] J. S. Aguilar-Ruiz. Shifting and scaling patterns from gene expression data[J]. Bioinformatics, 2005, 21(20):3840–3845

[16] J. A. Hartigan. Direct clustering of a data matrix[J]. Journal of the american statistical association, 1972, 67(337):123–129

[17] Y. Cheng, G. M. Church. Biclustering of expression data.[M]. Ismb, 2000, 93–103

[18] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay. A novel coherence measure for discovering scaling biclusters from gene expression data[J]. Journal of Bioinformatics and Computational Biology, 2009, 7(05):853–868

[19] K. Y. Yip, D. W. Cheung, M. K. Ng. Harp: A practical projected clustering algorithm[J]. Knowledge and Data Engineering, IEEE Transactions on, 2004, 16(11):1387–1397

[20] J. Yang, H. Wang, W. Wang, et al. An improved biclustering method for analyzing gene expression profiles[J]. International Journal on Artificial Intelligence Tools, 2005, 14(05):771–789

[21] K. Bryan, P. Cunningham, N. Bolshakova. Application of simulated annealing to the biclustering of gene expression data[J]. Information Technology in Biomedicine, IEEE Transactions on, 2006, 10(3):519–525

[22] A. Das, B. K. Chakrabarti. Quantum annealing and related optimization methods[M]. Springer Science & Business Media, 2005

[23] J. Liu, Z. Li, X. Hu, et al. Biclustering of microarray data with MOSPO based on crowding distance[J]. BMC bioinformatics, 2009, 10(4):1

[24] J. Kennedy. Particle swarm optimization[M]. Springer, 2011, 760–766

[25] G. P. Coelho, F. O. de França, F. J. Von Zuben. Multi-objective biclustering: When non-dominated solutions are not enough[J]. Journal of Mathematical Modelling and Algorithms, 2009, 8(2):175–202

[26] L. N. De Castro, J. Timmis. Artificial immune systems: a new computational intelligence approach[M]. Springer Science & Business Media, 2002

[27] C. Cano, L. Adarve, J. López, et al. Possibilistic approach for biclustering microarray data[J]. Computers in biology and medicine, 2007, 37(10):1426–1436

[28] W.-H. Yang, D.-Q. Dai, H. Yan. Finding correlated biclusters from gene expression data[J]. Knowledge and Data Engineering, IEEE Transactions on, 2011, 23(4):568–584

[29] J. Shi, J. Malik. Normalized cuts and image segmentation[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2000, 22(8):888–905

[30] G. Li, Q. Ma, H. Tang, et al. QUBIC: a qualitative biclustering algorithm for analyses of gene expression data[J]. Nucleic acids research, 2009:gkp491

[31] L. Lazzeroni, A. Owen. Plaid models for gene expression data[J]. Statistica sinica, 2002:61–86

[32] Q. Sheng, Y. Moreau, B. De Moor. Biclustering microarray data by Gibbs sampling[J]. Bioinformatics, 2003, 19(suppl 2):ii196–ii205

[33] Y. Kluger, R. Basri, J. T. Chang, et al. Spectral biclustering of microarray data: coclustering genes and conditions[J]. Genome research, 2003, 13(4):703–716

[34] P. Carmona-Saez, R. D. Pascual-Marqui, F. Tirado, et al. Biclustering of gene expression data by non-smooth non-negative matrix factorization[J]. BMC bioinformatics, 2006, 7(1):1

[35] B. Frisch, N. Koeniger. Social synchronization of the activity rhythms of honeybees within a colony[J]. Behavioral ecology and sociobiology, 1994, 35(2):91–98

[36] C. Huygens. Horologium oscillatorium: 1673[M]. Dawson, 1966

[37] Y. Kuramoto. Chemical oscillations, waves, and turbulence[M]. Springer Science & Business Media, 2012

[38] R. Agrawal, T. Imieliński, A. Swami. Mining association rules between sets of items in large databases[J]. ACM SIGMOD Record, 1993, 22(2):207–216

[39] I. S. Dhillon, S. Mallela, D. S. Modha. Information-theoretic co-clustering[M]. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, 89–98

[40] H. Cho, I. S. Dhillon. Coclustering of human cancer microarrays using minimum sum-squared residue coclustering[J]. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2008, 5(3):385–400

致 谢

大学四年来，我经历了很多，也成长了很多，这个过程少不了很多帮助我的人、改变我的人，借此机会，我要对你们表示我最真诚的感谢。

谢谢我的父亲和母亲，是你们给了我最无微不至的照顾和无条件的支持。当我得意时，是你们分享我的喜悦，并嘱咐我不要骄傲；当我落寞时，是你们鼓励我，让我重整旗鼓。我的每一个重大决定都能得到你们的支持，我取得的每一项成就都离不开你们。如今，我常年不在你们身边，当我一天一天强大，你们却一天一天老去，这是我心头的最痛。你们的恩情，我此生难报，只希望自己变得更强大，有能力保护你们、照顾你们，如同当初你们对我那样。

其次，我必须对我的科研导师邵俊明老师表达我由衷的敬意和谢意。自从大三我跨入教研室，邵老师就成了我最尊敬的人。邵老师以身作则，让我懂得了什么叫做科研，因此我奋斗，我每一天的努力都为缩小自己和邵老师之间的差距。邵老师的人格也让我肃然起敬，可以说作为邵老师的学生，他的高尚、包容与正直能让我们每一个人自惭形秽。我从邵老师身上学到的远远不止做学问，还有做人。我大学最庆幸的一件事情之一就是自己能找到邵老师作为自己的科研导师。

我也要感谢给我上课的每一个老师，你们传授我知识，你们让我看到世界。我忘不了徐全智老师对每一个学生的鞭策与鼓励，忘不了胡建浩老师每节课的“库式论坛”，忘不了每一个含辛茹苦的老师！你们是真正的大师，大学如果没有你们则不能称之为大学。

最后，我也要感谢四年的同学们，我庆幸我们能在一起生活，一起交流学习，同时互相竞争。四年来，我们共同仰望星空，脚踏实地。如今大家各奔东西，祝愿大家都能追到自己的梦想。大学中我最好的朋友们，互相关心互相为对方着想的、能称为兄弟姐妹的各位，我不担心毕业后会失去你们，我相信友谊天长地久。保重，各位，但愿人长久，千里共婵娟。

附录 A 人工数据集上的CoSync运行结果

以下是本次在人工数据集上进行的部分结果，每一幅图的左边为原始数据矩阵，每一个色块的轮廓代表一个联合簇；右边为CoSync完成联合后的结果。

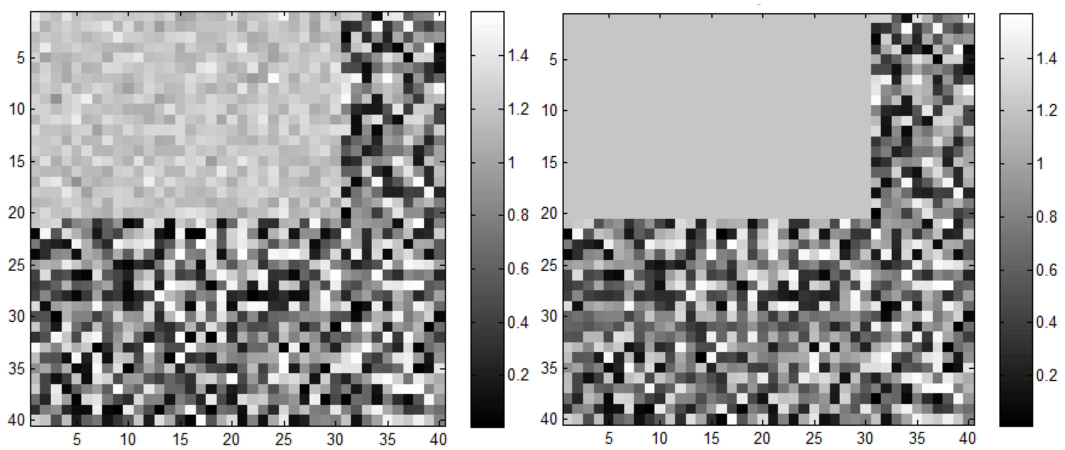


图 A-1 单个联合簇

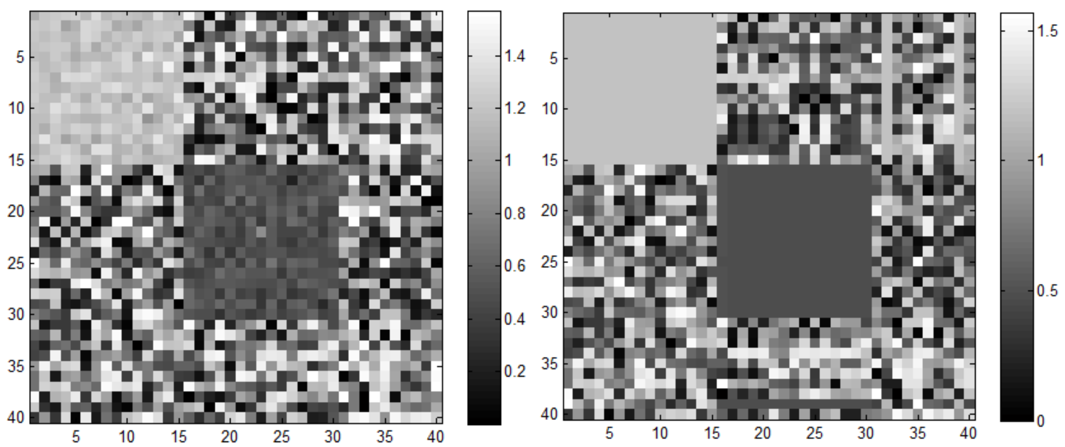


图 A-2 双联合簇，对角分布

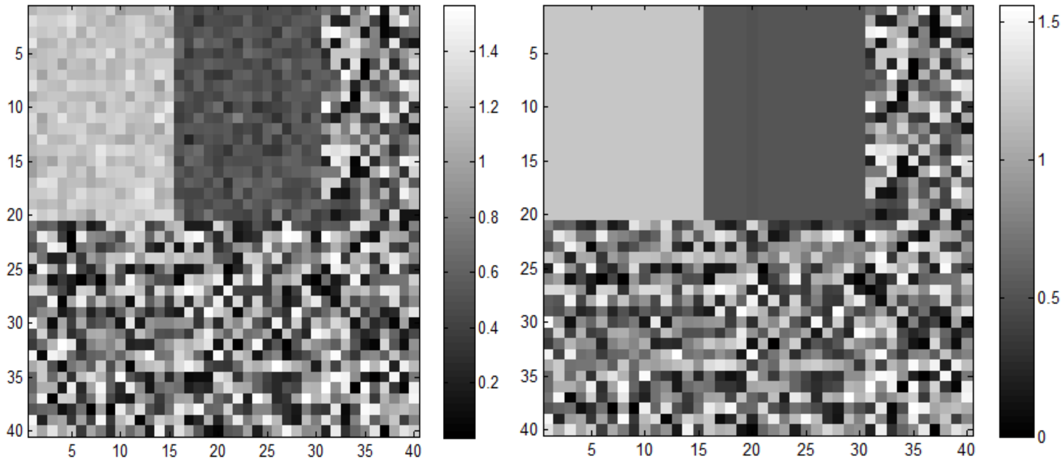


图 A-3 双联合簇，并排分布

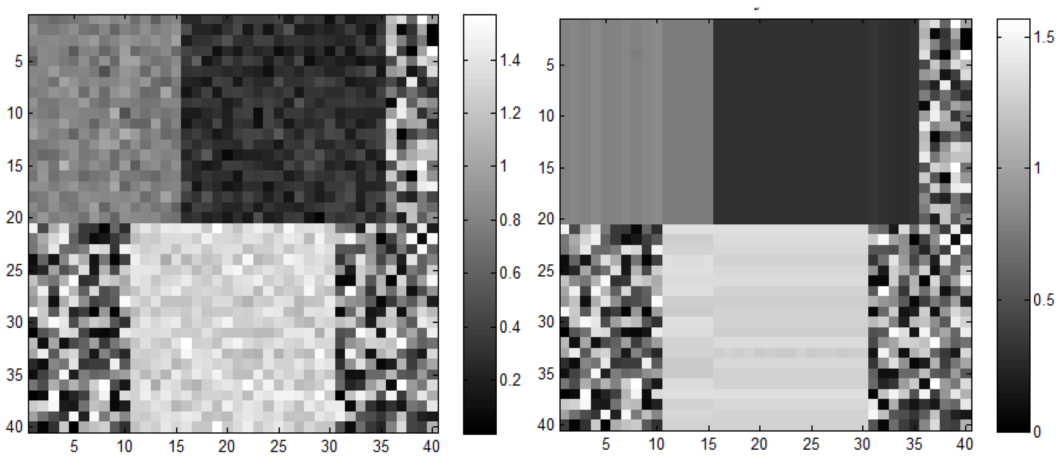


图 A-4 三联合簇，不规则分布

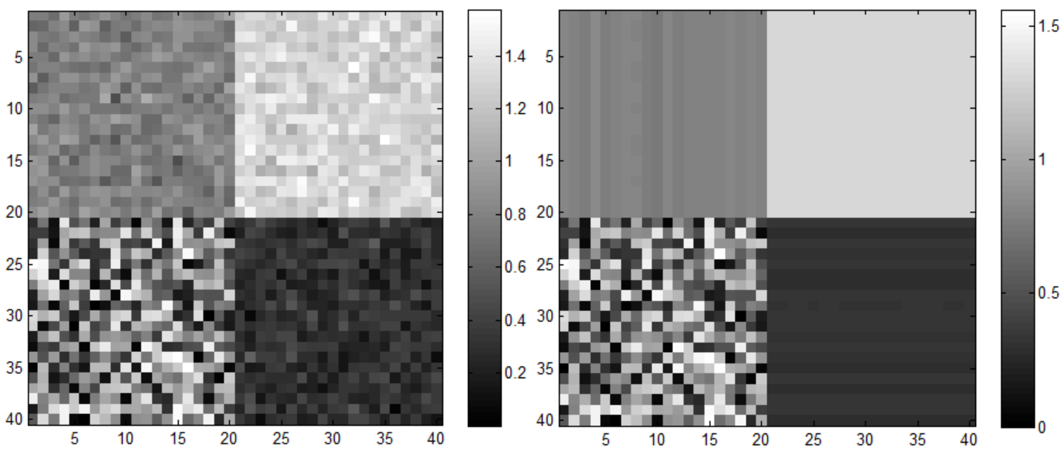


图 A-5 三联合簇，规则分布