摘 要

为了适应日益增长的宽带信号和非线性系统的工程应用，用于分析瞬态电磁散射问题的时域积分方程方法研究日趋活跃。本文以时域积分方程时间步进算法及其快速算法为研究课题，重点研究了时间步进算法的数值实现技术、后时稳定性问题以及两层平面波算法加速计算等，主要研究内容分为四部分。

……

关键词：时域电磁散射，时域积分方程，时间步进算法，后时不稳定性，时域平面波算法

ABSTRACT

With the widespread engineering applications ranging from broadband signals and nonlinear systems, time-domain integral equations (TDIE) methods for analyzing transient electromagnetic scattering problems are becoming widely used nowadays. TDIE-based marching-on-in-time (MOT) scheme and its fast algorithm are researched in this dissertation, including the numerical techniques of MOT scheme, late-time stability of MOT scheme, and two-level PWTD-enhanced MOT scheme. The contents are divided into four parts shown as follows.

……

Keywords: time-domain electromagnetic scattering, time-domain integral equation (TDIE), marching-on in-time (MOT) scheme, late-time instability, plane wave time-domain (PWTD) algorithm

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：

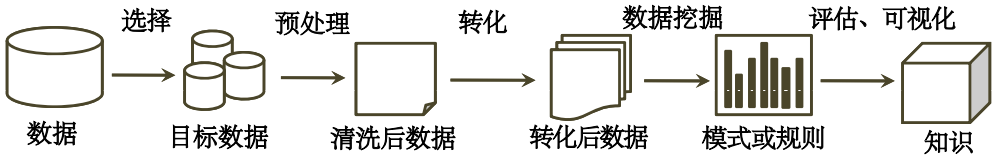

图 1-1 数据挖掘主要步骤图

- **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*[1],决策树[2],支持向量机[3]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*[4], *spectral clustering*[5] 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*[6], *DBSCAN*[7]和*FP-growth*[8]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[9]。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，Two-

mode 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。

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。

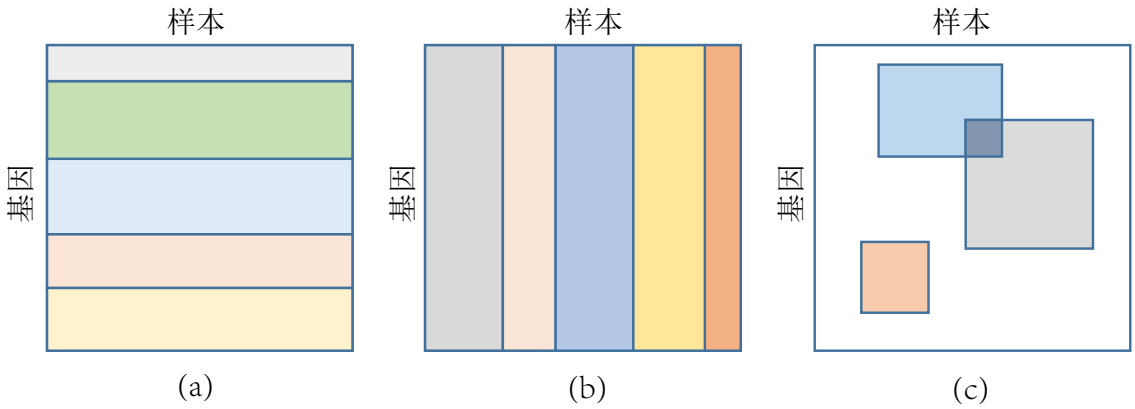

图 1-2 数据挖掘主要步骤图

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[10]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[11]。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*可以对大规模基因表达问题求解。

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，提出了*CoSync*算法，之后详细地推导出找寻联合簇，发现结果矩阵中的模式（Pattern）的启发式方法。最后将非负矩阵分解的方法结合到*CoSync*中，得到算法的全貌。
- 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[12]。

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[13]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[14]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[15]提出，直到2000年，Cheng和Church[16]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[10]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[15]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[16]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[17]改进*MSR* 为*SMSR(Scaling MSR)*后又进行更为深入的研究。Yip等人[18]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[19]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[20]基于模拟退火[21]的优化方式进行搜索，Liu等人[22]基于2011年最火的粒子群优化算法[23]，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[24]提出了基于人工免疫系统[25]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[26]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[27]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[28]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[10]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*[29]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[30]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[31]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[32]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[33]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[34]。早在1665年，伟大的数学家和物理学家，摆钟的发明者Christiaan Huygens[35]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[36]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[37]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，需要定义一个程度因子R_c：

定义 2.3.3 (同步程度因子) 同步程度因子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：

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

随着*Sync*的迭代过程，同步程度因子$R_c(t)$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。

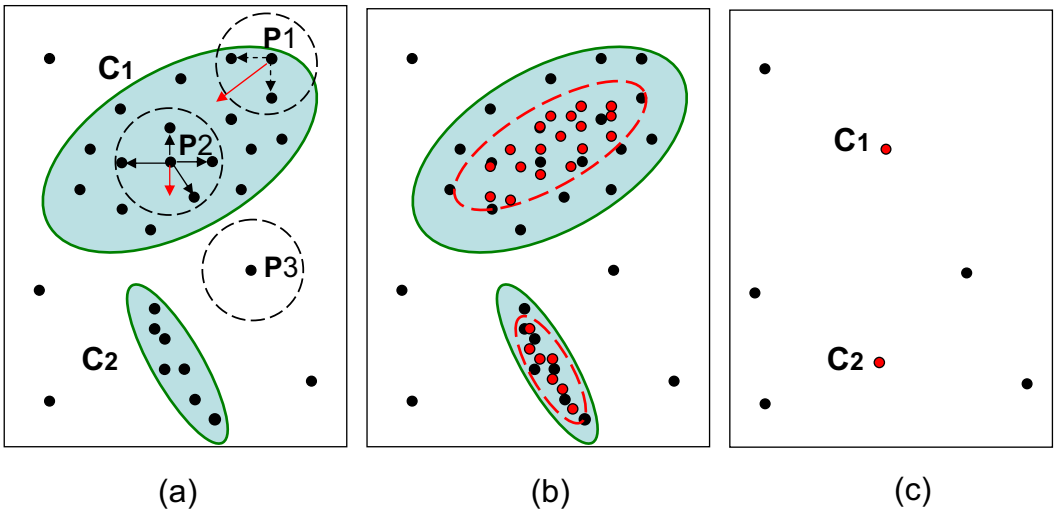

图 2-1 *Sync*算法聚类示意图 (a) 数据集的初始状态，黑色箭头代表点与点之间的相互交互作用，红色箭头代表点进行位移的方向。(b) *Sync*算法进行一段后与初态的对比图，红色部分为新的状态图。(c) 算法收敛后的最终状态，图中包含聚类簇C_1和C_2和若干离群点

*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。

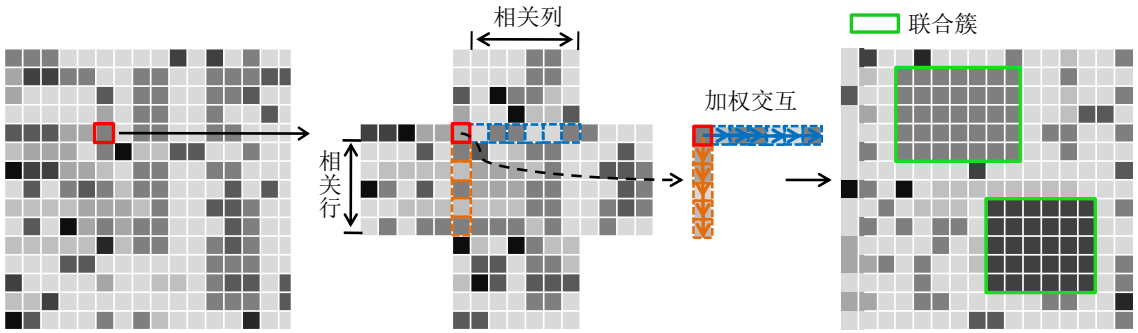

图 3-1 *CoSync*算法中矩阵元素在行列上的动态交互示意图 (a)关联矩阵的原始状态图，颜色深浅不一的小矩形代表原矩阵中元素值的大小，颜色越深值越小。(b) 矩阵总任意一元素的交互示意，红色小框代表的元素将被相关行列中对应的元素交互影响。(c) 算法收敛后矩阵行列排好序的最终状态，两个绿色框代表的联合簇都被自动发掘出来了。

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*CoSync*

现在将正式引入*CoSync*的模型定义，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邻域邻居可以表示为：

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

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$N_\epsilon^r(a_{i\cdot})$按照同样方式定义。

如同Kurumoto model[36]和*Sync*算法[37]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：

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

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$N_\epsilon^r(a_{.j}(t))$，其对A_{ij}的交互权重因子定义为：

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

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

$$w^c(i)=e^{-\lambda\cdot\sigma_i} \tag{3-4}$$

其中σ_i为向量$\nu_{iq}=\{abs(a_{iq}-a_{ij})\big|q\in N_\epsilon^c(a_{.j}(t))\}$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：

$$I_{row}(t)=\frac{w^r(j)}{2|N_\epsilon^r(a_{i.}(t))|}\cdot\sum_{p\in N_\epsilon^r(a_{i.}(t))}\sin(a_{pj}(t)-a_{ij}(t)) \tag{3-5}$$

$$I_{col}(t)=\frac{w^c(i)}{2|N_\epsilon^c(a_{.j}(t))|}\cdot\sum_{q\in N_\epsilon^c(a_{.j}(t))}\sin(a_{iq}(t)-a_{ij}(t)) \tag{3-6}$$

最后，加权双边交互聚类模型可以写为如下形式：

$$a_{ij}(t+1)=a_{ij}(t)+I_{row}(t)+I_{col}(t) \tag{3-7}$$

为了衡量矩阵中的同步的程度，从而决定算法的终结，我们引入程度因子r如下：

定义 3.2.4 (程度因子)

$$\begin{aligned}r=&\frac{1}{2|I|}\sum_{i=1}^{|I|}\frac{1}{|N_\epsilon^r(a_{i.})|}\sum_{p\in N_\epsilon^r(a_{i.})}\mathbf{e}^{-||a_{p.}-a_{i.}||}\\&+\frac{1}{2|J|}\sum_{i=1}^{|J|}\frac{1}{|N_\epsilon^c(a_{.j})|}\sum_{q\in N_\epsilon^c(a_{.j})}\mathbf{e}^{-||a_{.q}-a_{.j}||}\end{aligned} \tag{3-8}$$

3.3 结果矩阵上的模式搜索

第4章 实验设计,算法实现与评估

4.1 全文总结

本文以时域积分方程方法为研究背景，主要对求解时域积分方程的时间步进算法以及两层平面波快速算法进行了研究。

……

4.2 后续工作展望

时域积分方程方法的研究近几年发展迅速，在本文研究工作的基础上，仍有以下方向值得进一步研究：

……

第5章 全文总结与展望

5.1 全文总结

本文以时域积分方程方法为研究背景，主要对求解时域积分方程的时间步进算法以及两层平面波快速算法进行了研究。

……

5.2 后续工作展望

时域积分方程方法的研究近几年发展迅速，在本文研究工作的基础上，仍有以下方向值得进一步研究：

……

参考文献

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

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

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

[4] 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

[5] 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

[6] 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

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

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

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

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

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

[13] 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

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

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

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

[17] 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

[18] 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

[19] 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

[20] 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

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

[22] 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

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

[24] 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

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

[26] 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

[27] 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

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

[29] 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

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

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

[32] 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

[33] 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

[34] 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

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

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

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

[38] 王浩刚, 聂在平. 三维矢量散射积分方程中奇异性分析[J]. 电子学报, 1999, 27(12):68–71

[39] X. F. Liu, B. Z. Wang, W. Shao. A marching-on-in-order scheme for exact attenuation constant extraction of lossy transmission lines[C]. China-Japan Joint Microwave Conference Proceedings, Chengdu, 2006, 527–529

[40] 竺可桢. 物理学[M]. 北京: 科学出版社, 1973, 56–60

[41] 陈念永. 毫米波细胞生物效应及抗肿瘤研究[D]. 成都: 电子科技大学, 2001, 50–60

[42] 顾春. 牢牢把握稳中求进的总基调[N]. 人民日报, 2012年3月31日

[43] 冯西桥. 核反应堆压力容器的LBB分析[R]. 北京: 清华大学核能技术设计研究院, 1997年6月25日

[44] 肖珍新. 一种新型排渣阀调节降温装置[P]. 中国, 实用新型专利, ZL201120085830.0, 2012年4月25日

[45] 中华人民共和国国家技术监督局. GB3100-3102. 中华人民共和国国家标准–量与单位[S]. 北京: 中国标准出版社, 1994年11月1日

[46] M. Clerc. Discrete particle swarm optimization: a fuzzy combinatorial box[EB/OL]. `http://clere.maurice.free.fr/pso/Fuzzy_Discrere_PSO/Fuzzy_DPSO.htm`, July 16, 2010

致 谢

在攻读博士学位期间，首先衷心感谢我的导师 XXX 教授，……

……

The Name of the Game

1.1 xxx

1.1.1 xxx

1.1.1.1 xxxx

1.2 xxx

1.2.1 xxx

1.2.1.1 xxxx

English words like 'technology' stem from a Greek root beginning with the letters $\tau\epsilon\chi\ldots$; and this same Greek word means *art* as well as technology. Hence the name TEX, which is an uppercase form of $\tau\epsilon\chi$.TeX (actually TEX), meaning of $\tau\epsilon\chi$

Insiders pronounce the χ of TEX as a Greek chi, not as an 'x', so that TEX rhymes with the word blecchhh. It's the 'ch' sound in Scottish words like *loch* or German words like *ach*; it's a Spanish 'j' and a Russian 'kh'. When you say it correctly to your computer, the terminal may become slightly moist.

The purpose of this pronunciation exercise is to remind you that TEX is primarily concerned with high-quality technical manuscripts: Its emphasis is on art and technology, as in the underlying Greek word. If you merely want to produce a passably good document—something acceptable and basically readable but not really beautiful—a simpler system will usually suffice. With TEX the goal is to produce the *finest* quality; this requires more attention to detail, but you will not find it much harder to go the extra distance, and you'll be able to take special pride in the finished product.

On the other hand, it's important to notice another thing about TEX's name: The 'E' is out of kilter. This logo displaced 'E' is a reminder that TEX is about typesetting, and it distinguishes TEX from other system names. In fact, TEX (pronounced *tecks*) is the admirable *Text EXecutive* processor developed by Honeywell Information Systems.

Since these two system names are Bemer, Robert, see TEX, ASCII pronounced quite differently, they should also be spelled differently. The correct way to refer to TEX in a computer file, or when using some other medium that doesn't allow lowering of the 'E', is to type '—TeX—'. Then there will be no confusion with similar names, and people will be primed to pronounce everything properly.

此名有诗意

1.1 xxx

1.1.1 xxx

1.1.1.1 xxxx

1.2 xxx

1.2.1 xxx

1.2.1.1 xxxx

英语单词“technology”来源于以字母$\tau\epsilon\chi\ldots$开头的希腊词根；并且这个希腊单词除了 technology的意思外也有art的意思。因此，名称TEX是$\tau\epsilon\chi$的大写格式。

在发音时，TEX的χ的发音与希腊的chi一样，而不是“x”，所以TEX与blecchhh押韵。“ch” 听起来象苏格兰单词中的loch 或者德语单词中的ach；它在西班牙语中是“j”，在俄语中是“kh”。当你对着计算机正确读出时, 终端屏幕上可能有点雾。

这个发音练习是提醒你，TEX主要处理的是高质量的专业书稿：它的重点在艺术和专业方面, 就象希腊单词的含义一样。如果你仅仅想得到一个过得去——可读下去但不那么漂亮——的文书, 那么简单的系统一般就够用了。使用TEX的目的是得到最好的质量；这就要在细节上花功夫, 但是你不会认为它难到哪里去，并且你会为所完成的作品感到特别骄傲。

另一方面重要的是要注意到与TEX名称有关的另一件事: “E”是错位的。这个偏移“E”的标识提醒人们，TEX与排版有关，并且把TEX从其它系统的名称区别开来。实际上，TEX(读音为 tecks)是Honeywell Information Systems 的极好的Text EXecutive处理器。因为这两个系统的名称读音差别很大，所以它们的拼写也不同。在计算机中表明TEX文件的正确方法，或者当所用的方式无法降低“E”时，就要写作“TeX”。这样, 就与类似的名称不会产生混淆, 并且为人们可以正确发音提供了条件。