电子科技大学
University of Electronic Science and Technology of China

基于同步原理的多边聚类算法研究

中期答辩

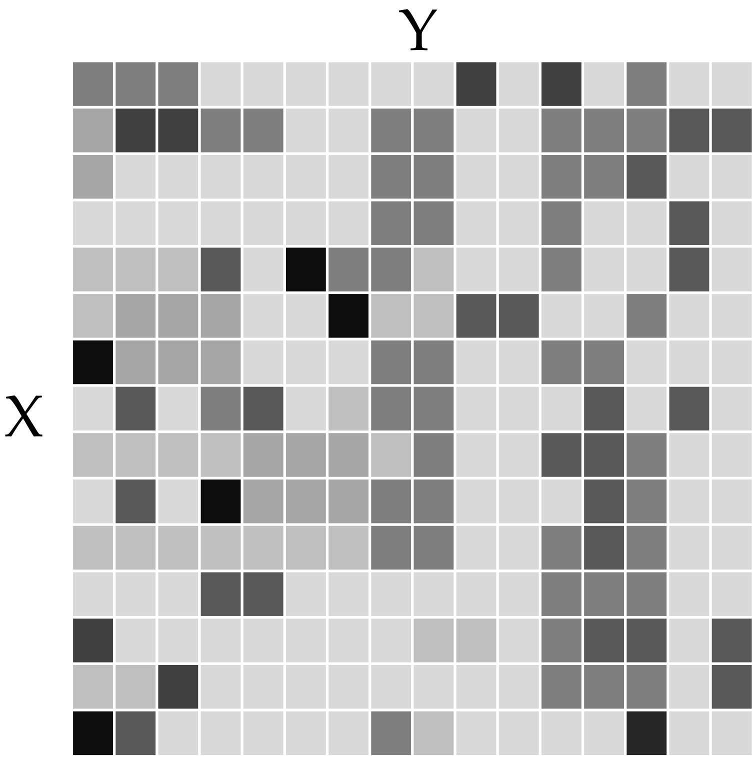
英才实验学院
高崇铭（2012001010016）
指导老师：邵俊明

目录

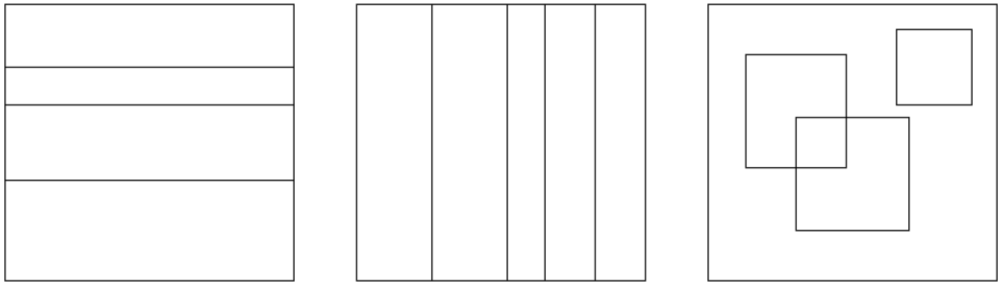
双边聚类算法*Co-Sync*
英才实验学院
高崇铭（*2012001010016*）
指导老师：邵俊明

1. 背景与动机

➤Co-clustering基本思想



Co-clustering思路：抓住X空间与Y空间的依赖性，同时对X空间与Y空间进行聚类。对应于关联矩阵，找寻行、列都相似的子矩阵



a. Row clustering b. Column clustering c. Co-clustering

X和Y的含义：

推荐系统：商品与顾客
文本分析：文档与词汇
基因表达：基因与样本

2. Co-clustering相关技术

➢主流算法

1. 基于Residue的方法： Cheng and Church's Algorithm(2000)

plaid(2002) , MSSRCC(2008) …

2. 基于Graph cut的方法：二分图划分、谱聚类 及其变种, (H. Zha, 2001) , (Y. Kluger, 2003) , (I. S. Dhillon, 2004), (B. Gao, 2005), Qubic(G. Li, 2009) …

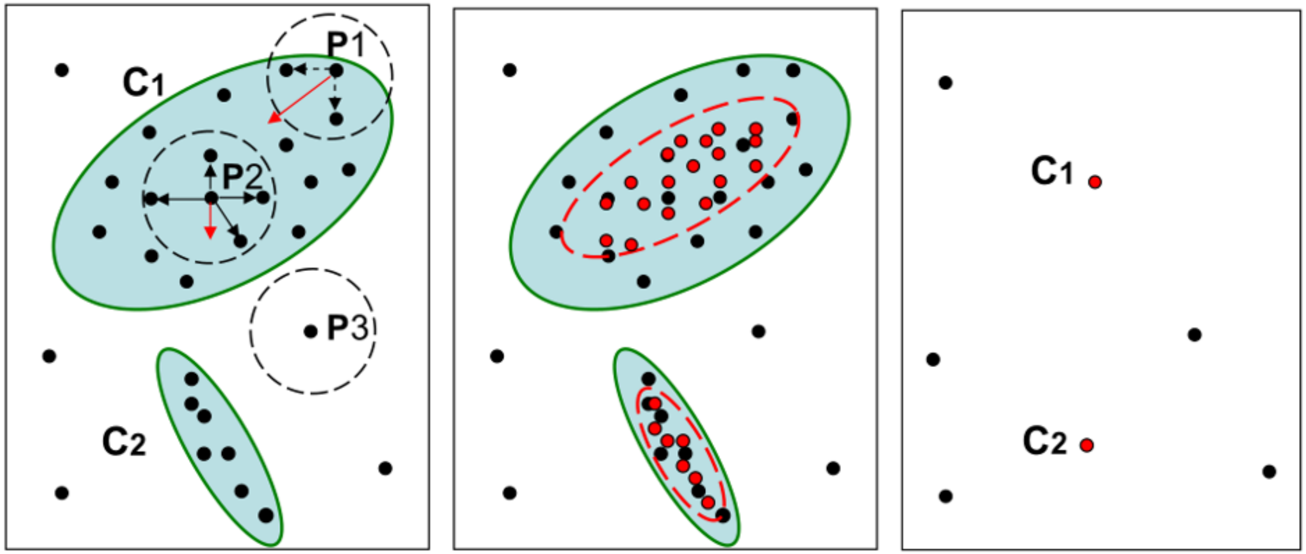
3.基于Information-theory的方法： (R. El-Yaniv, 2001), ITCC (I. S. Dhillon, 2003), (A. Banerjee,2004), (Y. Song,2013)…

➢缺陷与不足

1. 多数算法都有严格的“block”块分布的限制，不能解决重叠块的问题。
2. 只能处理二维矩阵的问题，对高维张量下的问题无能为力。
3. 参数的选取很敏感，参数对算法结果的影响很大。

3. 新方式：Co-Sync算法

➢Sync聚类算法：



最形象的理解：
数据点在**拉力**的作用下随着时间**动态变化**

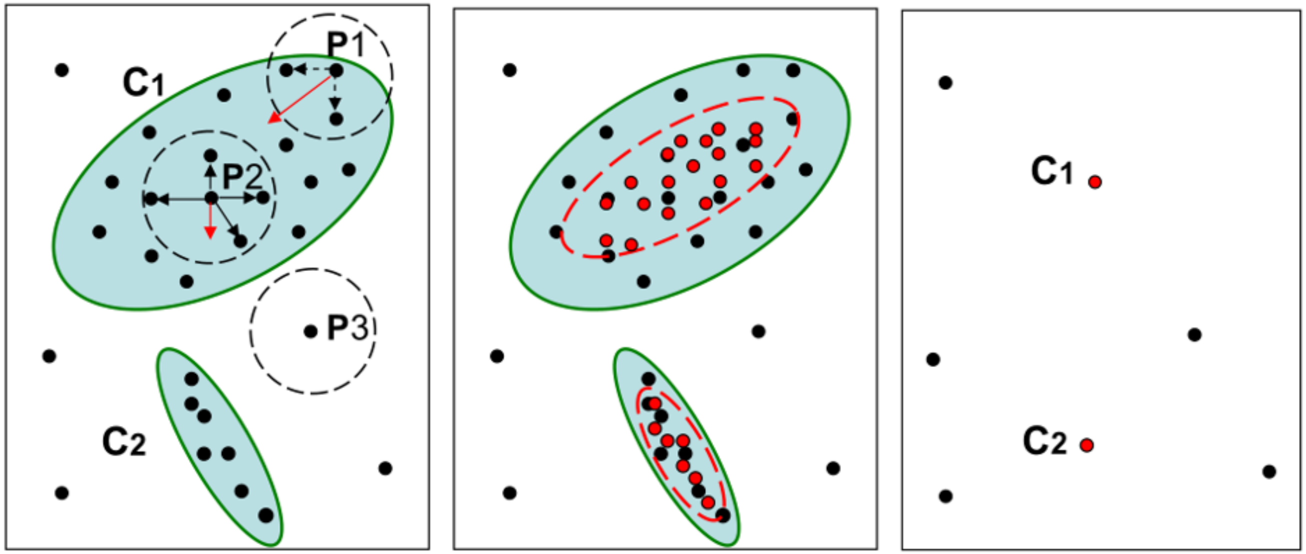
Basic interact function:

$$x_i(t+1) = x_i(t) + \frac{1}{|Nb_\epsilon(x(t))|} \cdot \sum_{y \in Nb_\epsilon(x(t))} \sin(y_i(t) - x_i(t)).$$

其中x是空间中任意一个点，集合$Nb\epsilon$是x的ϵ领域内的点集，y是集合$Nb\epsilon$中的任一点。 x_i 与y_i 代表点x与y在第i个维度上的值。

3. 新方式：Co-Sync算法

➢Sync聚类算法：



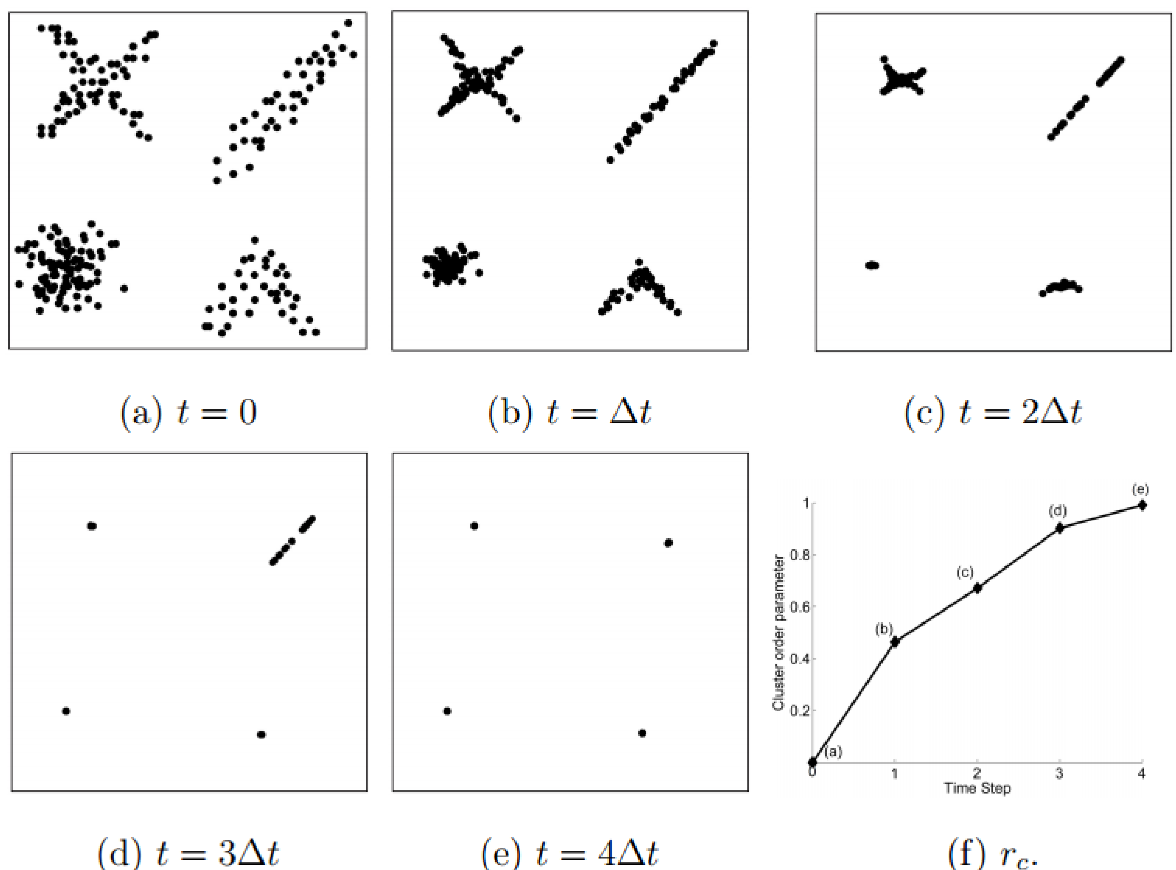
维度1	维度2
4.5	6.9
4.8	7.2
6.1	6.8
3.7	8.4
5.7	7.6
7.2	7.8
5.5	2.3
5.8	3
5.2	2.7
4.9	2.4

维度1	维度2
5.3	7.5
5.3	7.5
5.3	7.5
5.3	7.5
5.3	7.5
5.3	7.5
5.4	2.6
5.4	2.6
5.4	2.6
5.4	2.6

最形象的理解：
数据点在**拉力**的作用下随着时间**动态变化**

3. 新方式：Co-Sync算法

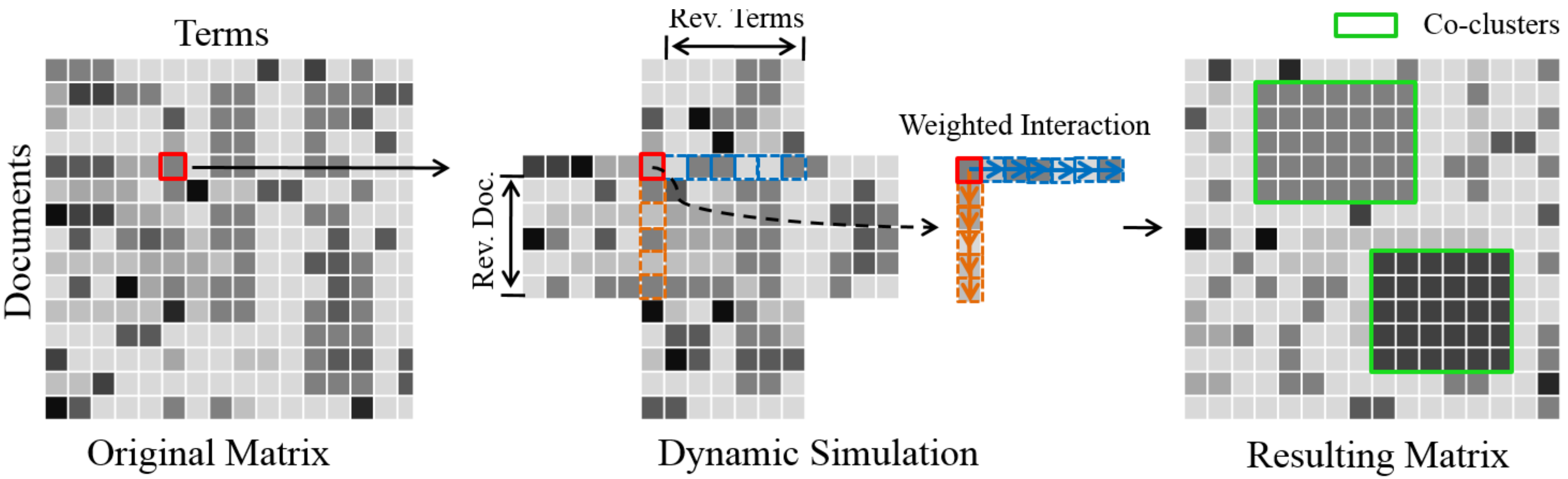
➢Sync算法聚类过程



二维Sync聚类过程截图

3. 新方式：Co-Sync算法

➤Co-Sync全过程：

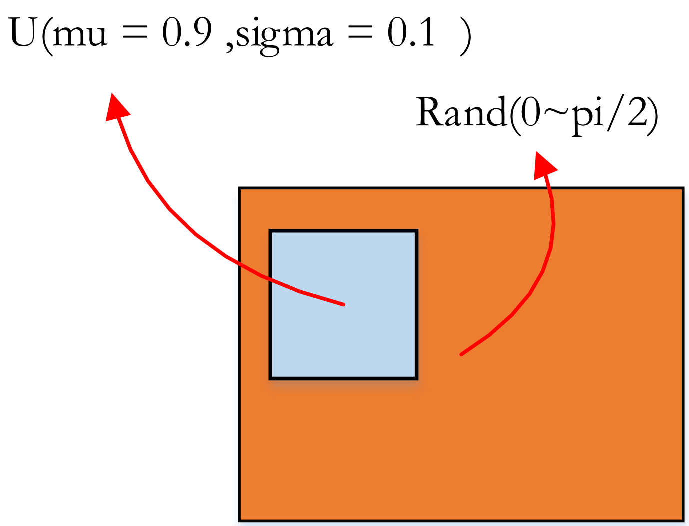


$$a_{ij}(t+1) = a_{ij}(t) + \frac{1}{2|N_\epsilon^r(a_{i\cdot}(t))|} \cdot \sum_{p \in N_\epsilon^c(a_{i\cdot}(t))} \sin(a_{pj}(t) - a_{ij}(t))$$

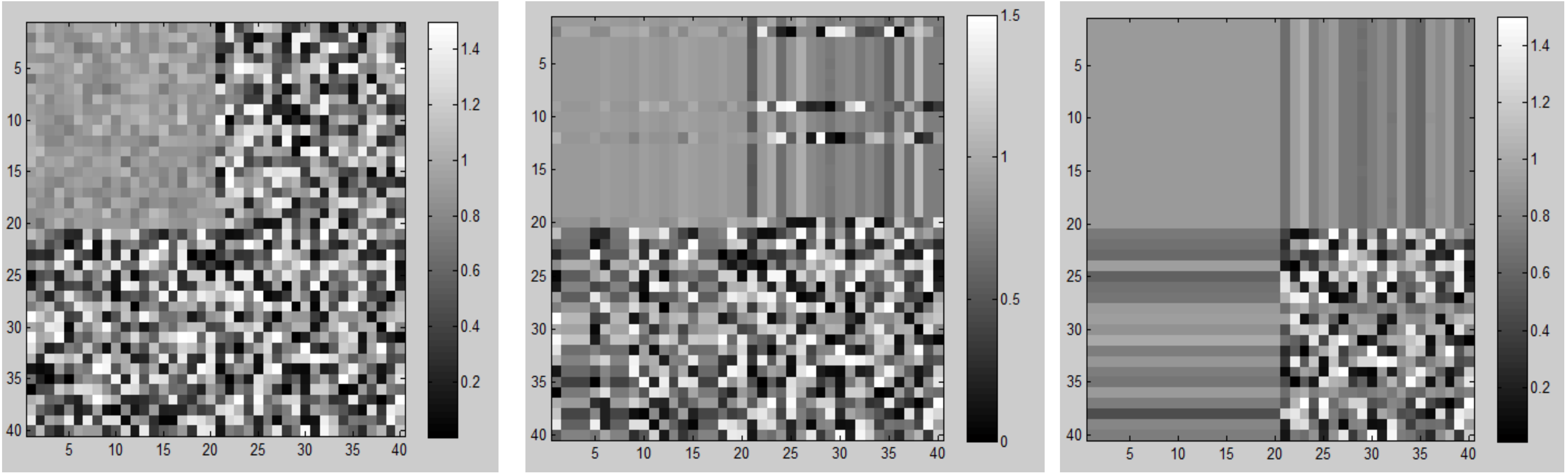
$$+ \frac{1}{2|N_\epsilon^c(a_{\cdot j}(t))|} \cdot \sum_{q \in N_\epsilon^r(a_{\cdot j}(t))} \sin(a_{iq}(t) - a_{ij}(t))$$

4. Co-Sync在人工数据集上的表现

➢构造混合高斯人工数据集

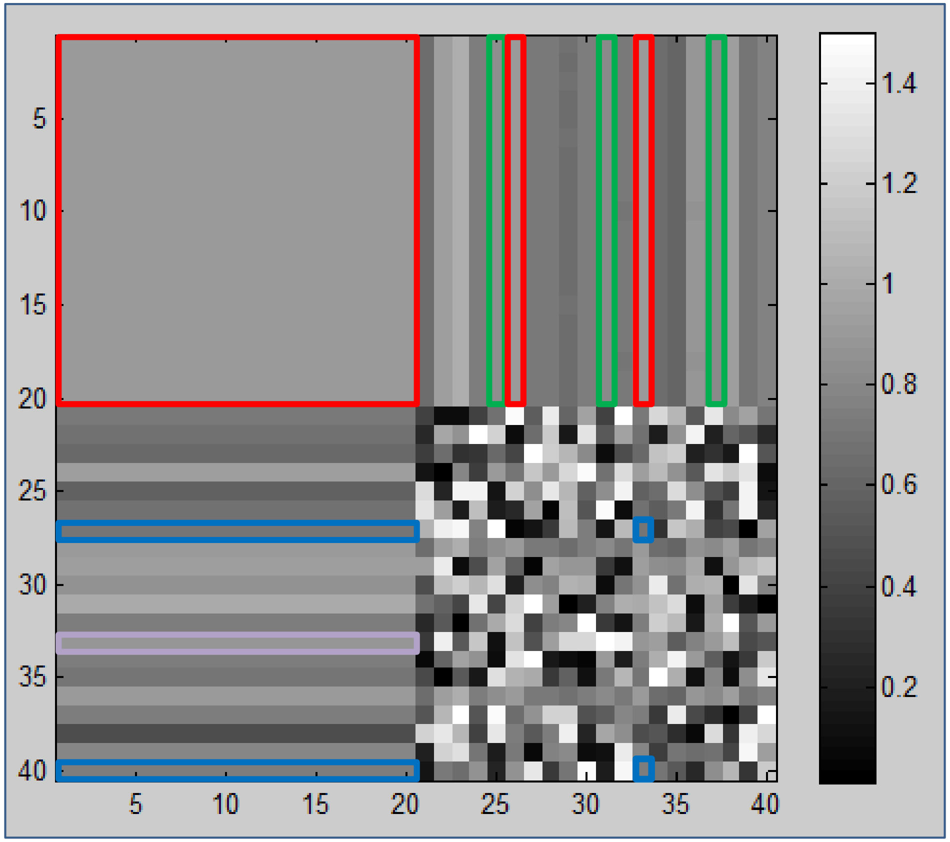


边际效应：
行与行，列与列之间拉扯形成



4. Co-Sync在人工数据集上的表现

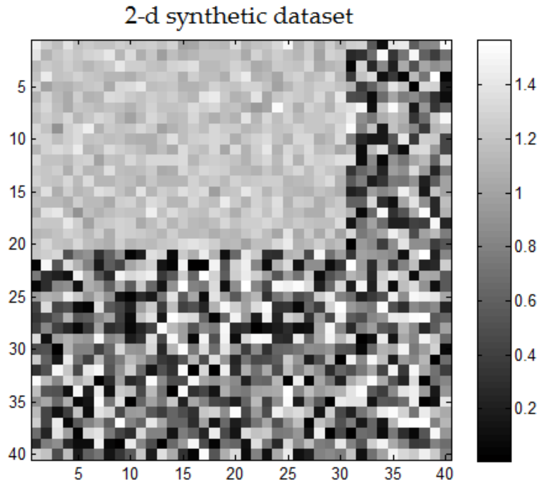
➢边际效应及其改进措施



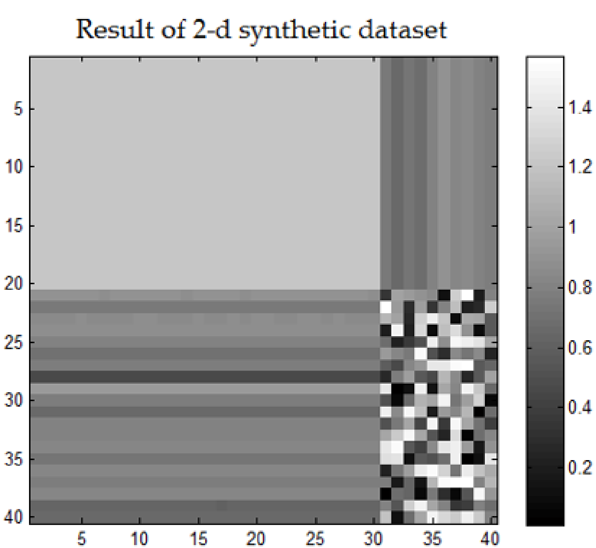
在行列交互中都引入一个权重weight，考虑取熵或者指数衰减函数

4. Co-Sync在人工数据集上的表现

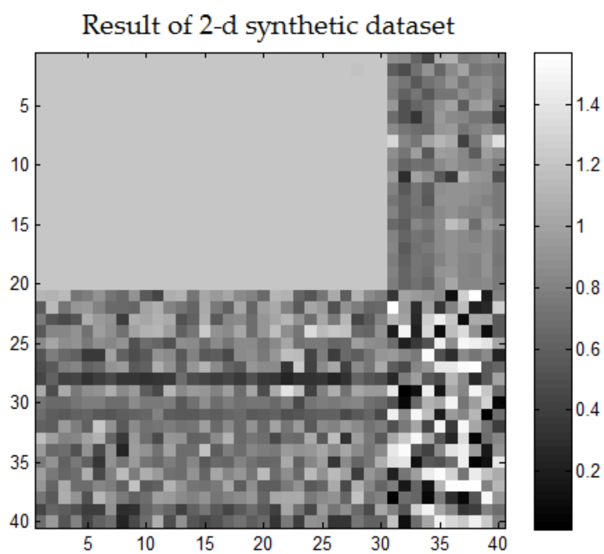
➢单个pattern实验



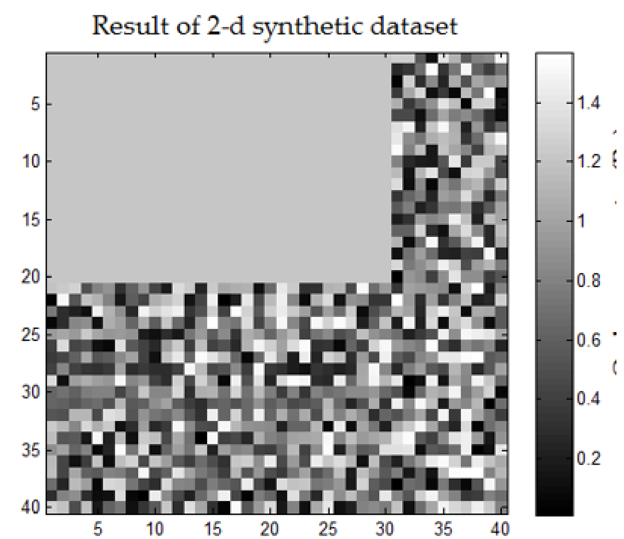

```
mu1 = 1.2;
mu2 = 0.4;
sigma = 0.1;
n = 40;
data = rand(n,n)*pi/2;
data(1:20,1:30)= mu1 + sigma* randn(20,30);
```



原始Co-Sync



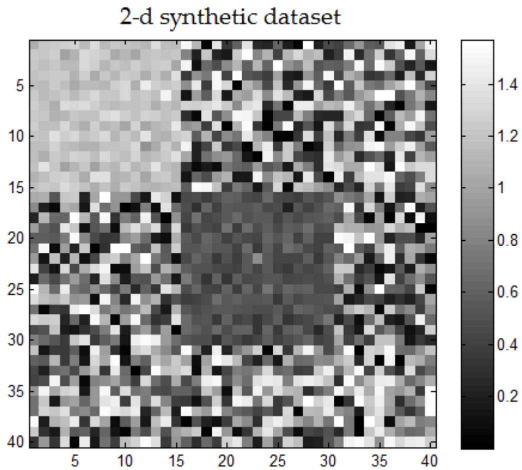
基于熵的Co-Sync



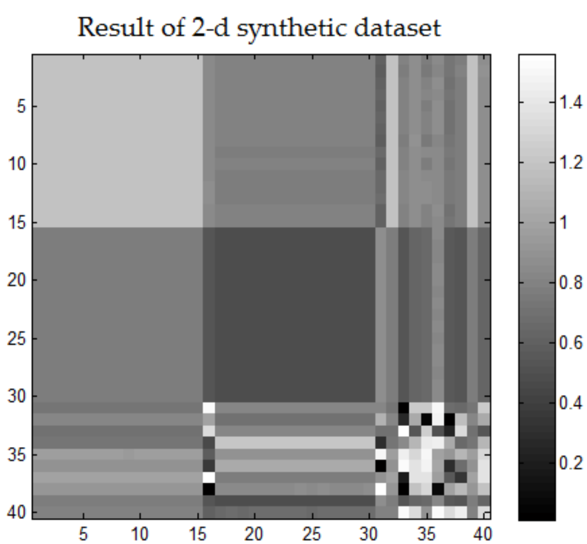
基于指数函数的Co-Sync

4. Co-Sync在人工数据集上的表现

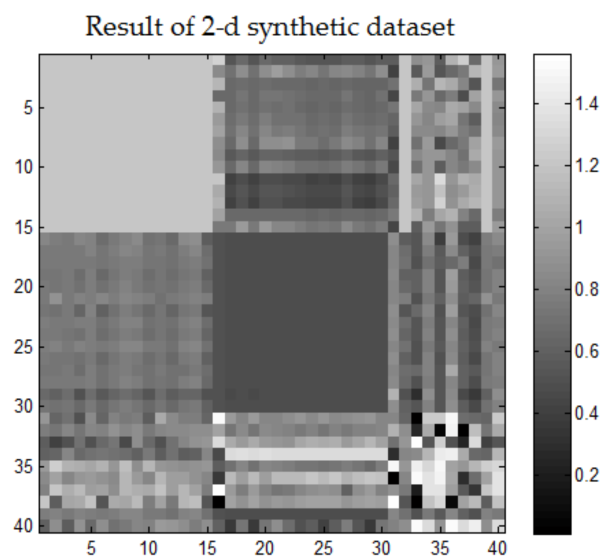
➢双pattern,无重叠实验



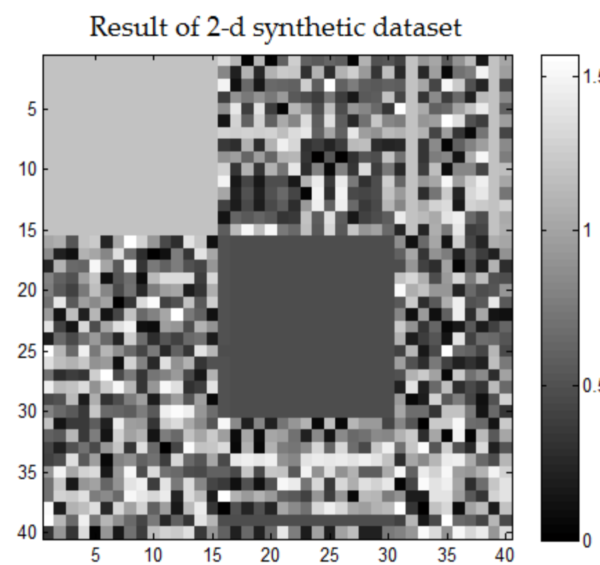

```
mu1 = 1.2;
mu2 = 0.5;
sigma = 0.1;
n = 40;
data = rand(n,n)*pi/2;
data(1:15,1:15) = mu1 + sigma * randn(15,15);
data(16:30,16:30) = mu2 + sigma * randn(15,15);
```



原始Co-Sync



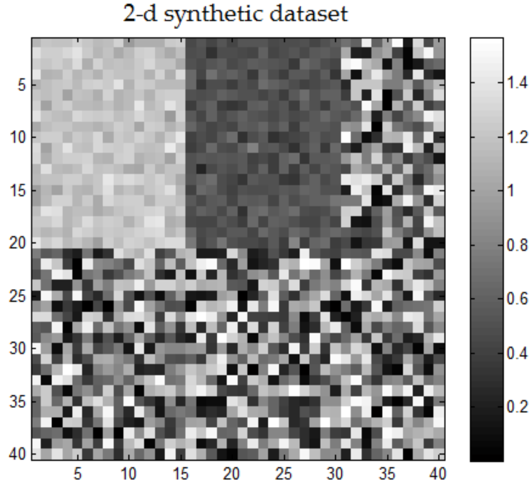
基于熵的Co-Sync



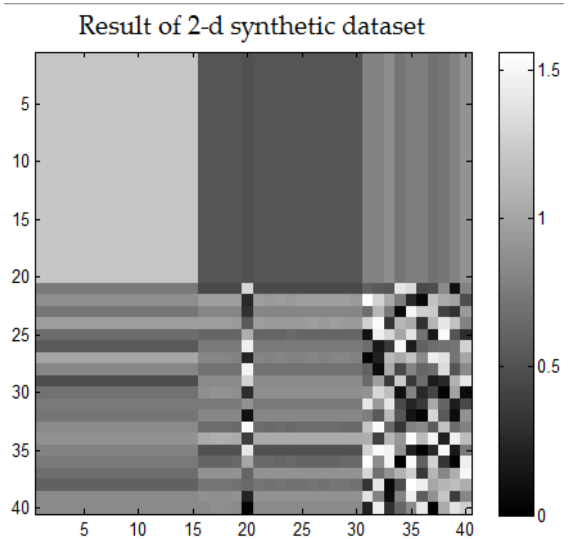
基于指数函数的Co-Sync

4. Co-Sync在人工数据集上的表现

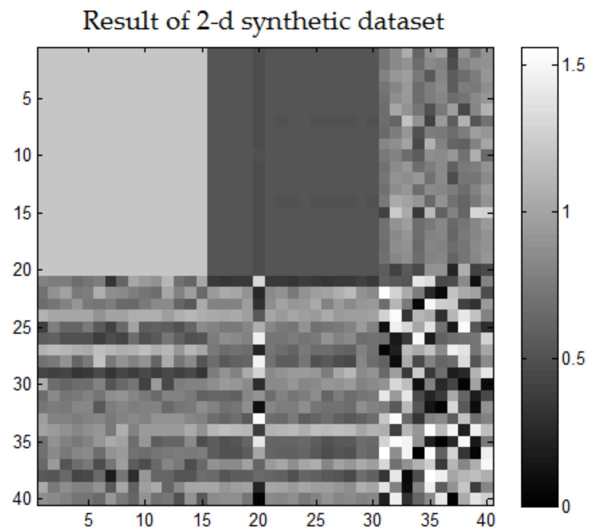
➤双pattern并列实验



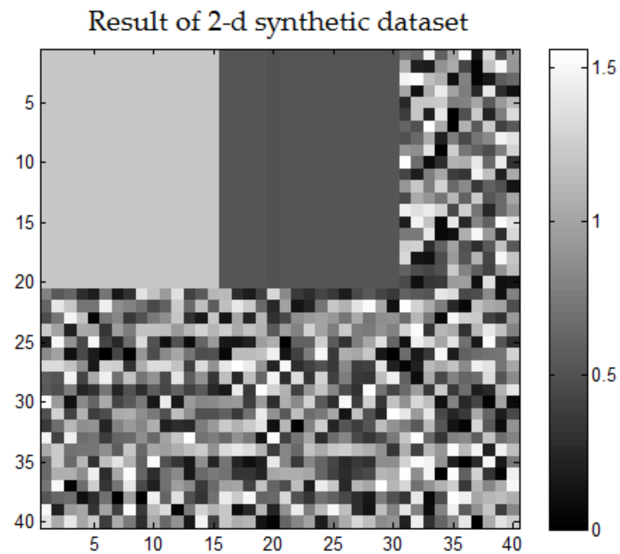

```
mu1 = 1.2;
mu2 = 0.5;
sigma = 0.1;
n = 40;
data = rand(n,n)*pi/2;
data(1:20,1:15) = mu1 + sigma * randn(20,15);
data(1:20,16:30) = mu2 + sigma * randn(20,15);
```



原始Co-Sync



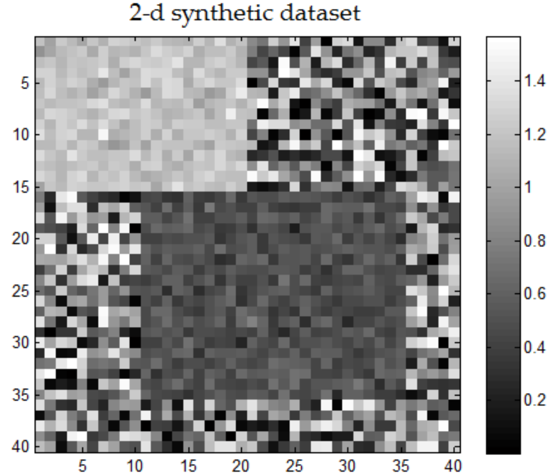
基于熵的Co-Sync



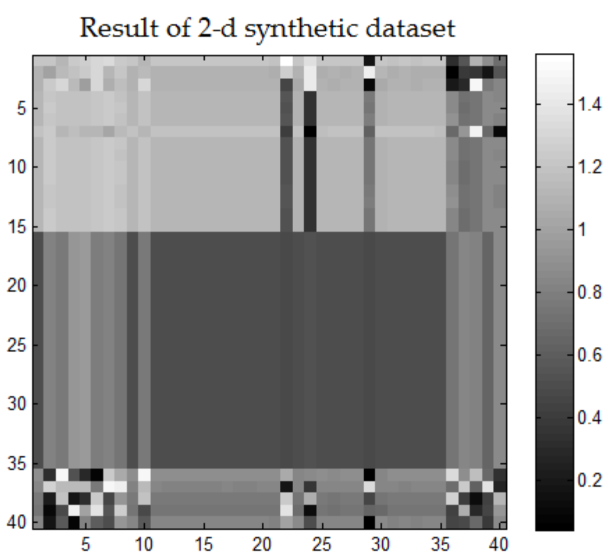
基于指数函数的Co-Sync

4. Co-Sync在人工数据集上的表现

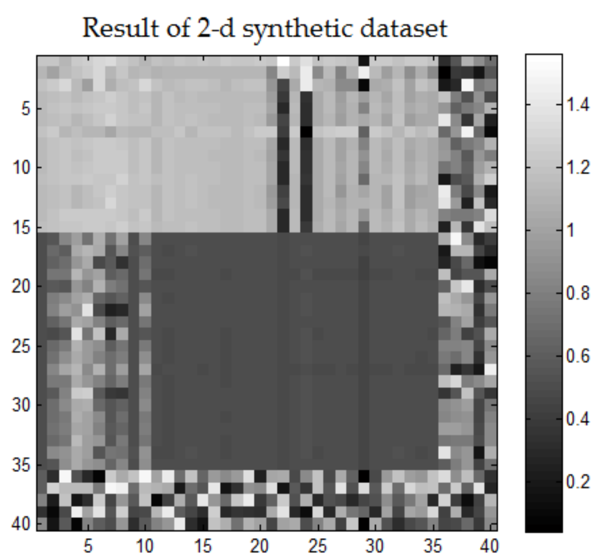
➢双pattern有不规则重叠实验



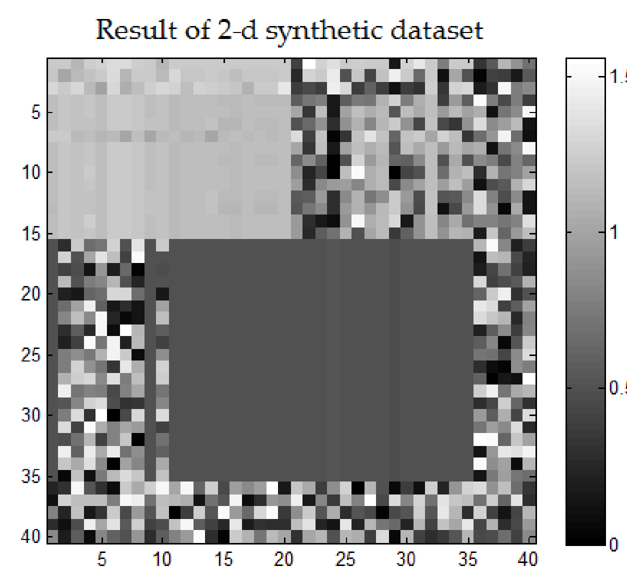

```
mu1 = 1.2;
mu2 = 0.5;
sigma = 0.1;
n = 40;
data = rand(n,n)*pi/2;
data(1:15,1:20) = mu1 + sigma * randn(15,20);
data(16:35,11:35) = mu2 + sigma * randn(20,25);
```



原始Co-Sync



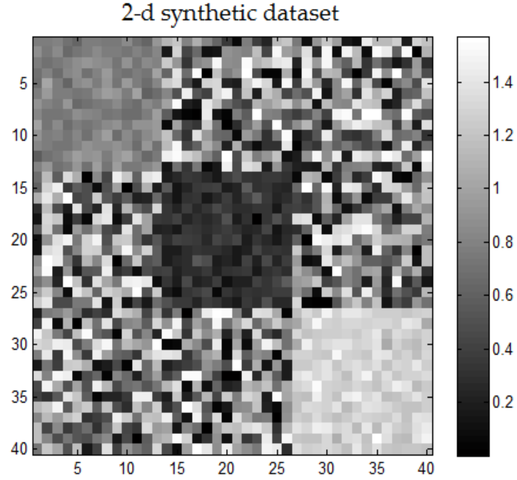
基于熵的Co-Sync



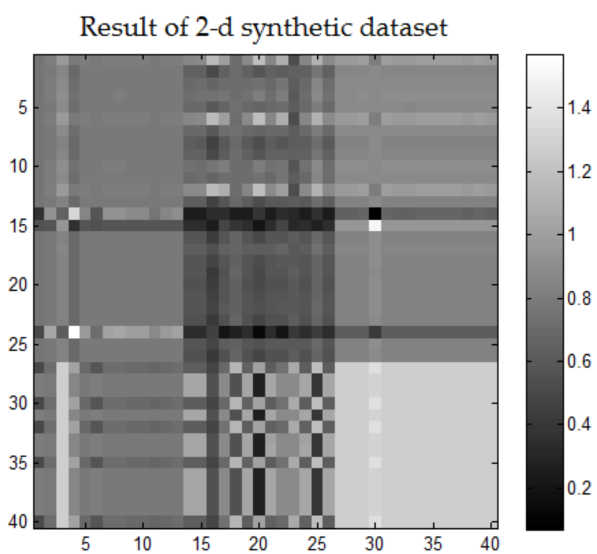
基于指数函数的Co-Sync

4. Co-Sync在人工数据集上的表现

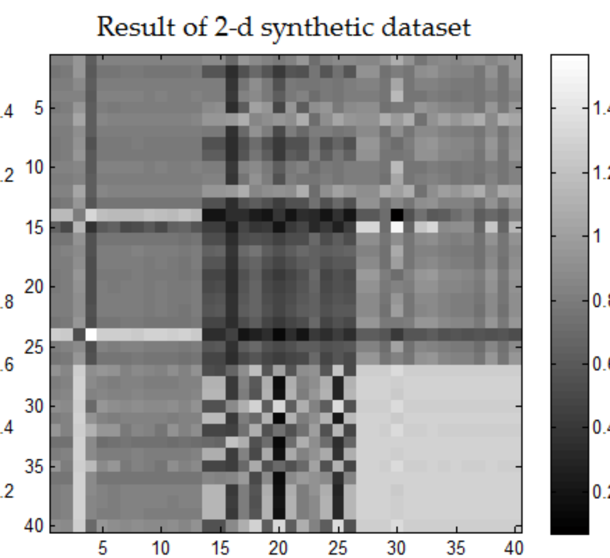
➢三pattern无重叠实验



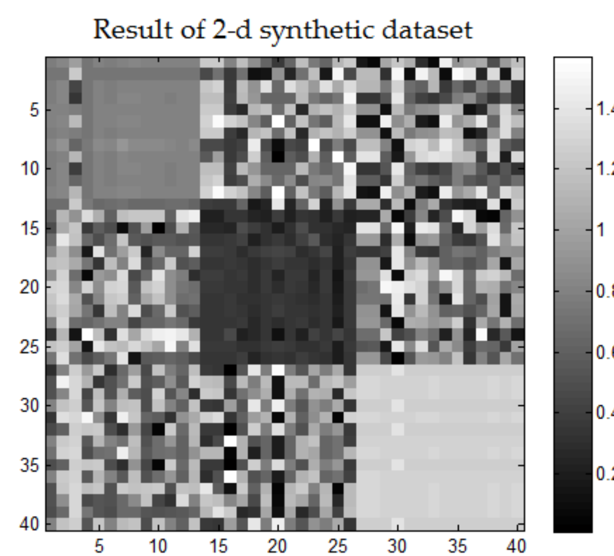

```
mu1 = 0.8;
mu2 = 0.3;
mu3 = 1.3;
% mu4 = 1.0;
sigma = 0.1;
n = 40;
data = rand(n,n)*pi/2;
data(1:13,1:13) = mu1 + sigma * randn(13,13);
data(14:26,14:26) = mu2 + sigma * randn(13,13);
data(27:40,27:40) = mu3 + sigma * randn(14,14);
```



原始Co-Sync



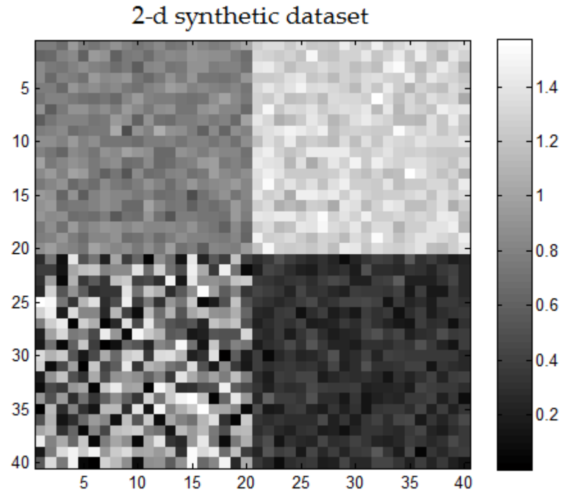
基于熵的Co-Sync



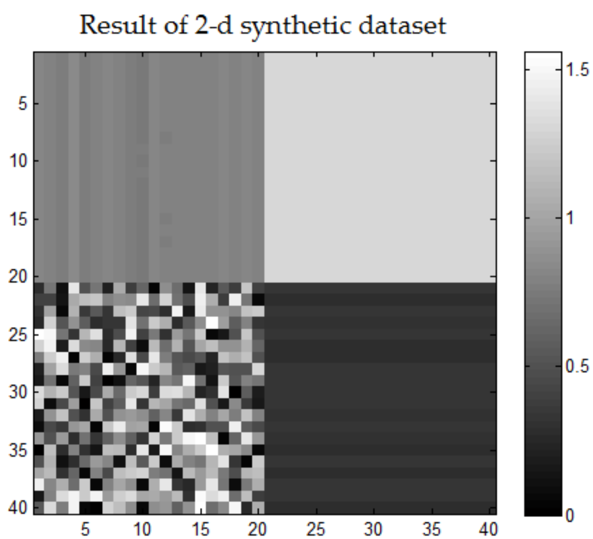
基于指数函数的Co-Sync

4. Co-Sync在人工数据集上的表现

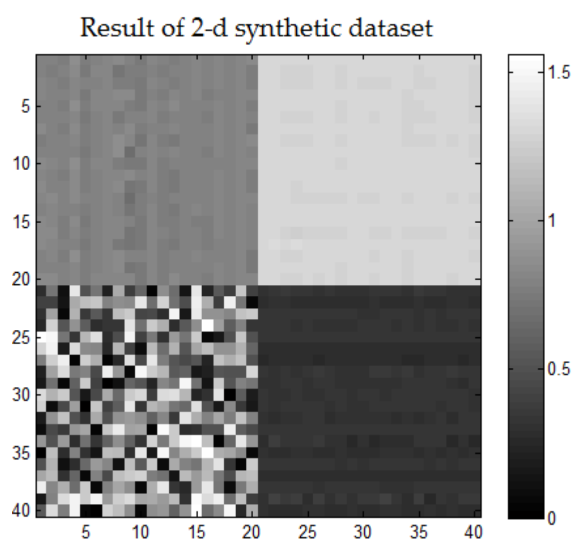
➢三pattern规则实验



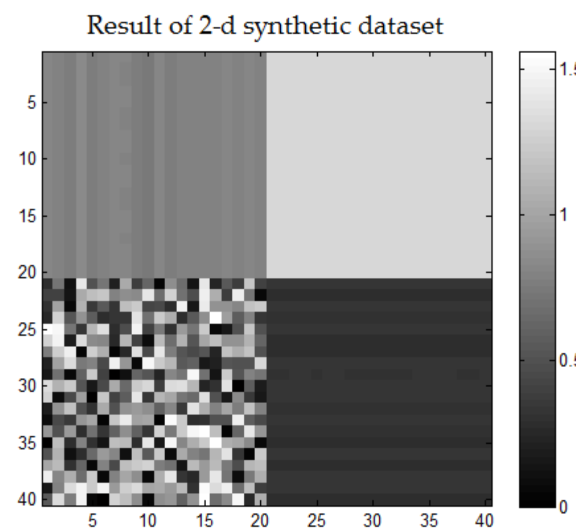

```
mu1 = 0.8;
mu2 = 0.3;
mu3 = 1.3
sigma = 0.1;
n = 40;
data = rand(n,n)*pi/2;
data(1:20,1:20) = mu1 + sigma * randn(20,20);
data(21:40,21:40) = mu2 + sigma * randn(20,20);
data(1:20,21:40) = mu3 + sigma * randn(20,20);
```



原始Co-Sync



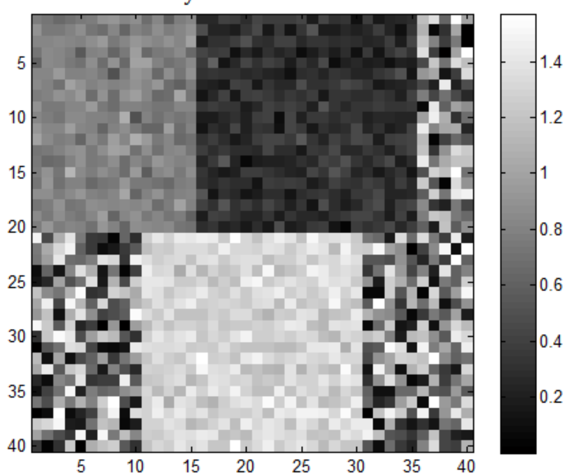
基于熵的Co-Sync



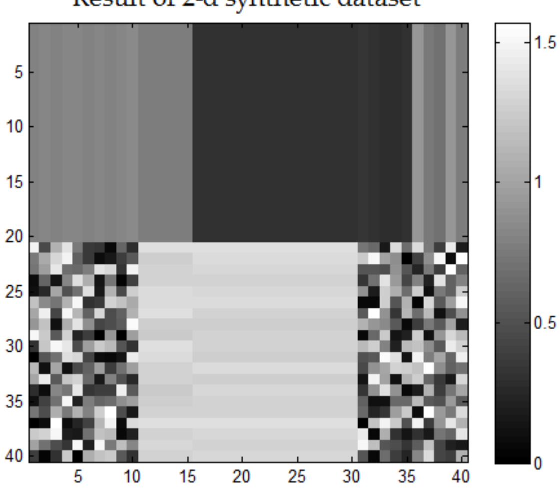
基于指数函数的Co-Sync

4. Co-Sync在人工数据集上的表现

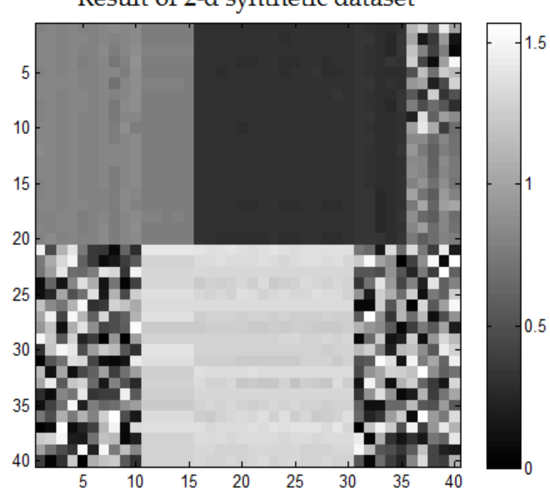
➢三pattern一般化实验



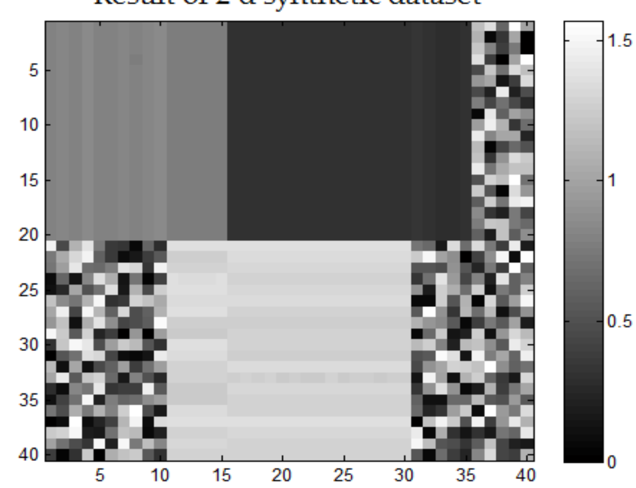

```
mu1 = 0.8;
mu2 = 0.3;
mu3 = 1.3
sigma = 0.1;
n = 40;
data = rand(n,n)*pi/2;
data(1:20,1:15) = mu1 + sigma * randn(20,15);
data(1:20,16:35) = mu2 + sigma * randn(20,20);
data(21:40,11:30) = mu3 + sigma * randn(20,20);
```



原始Co-Sync



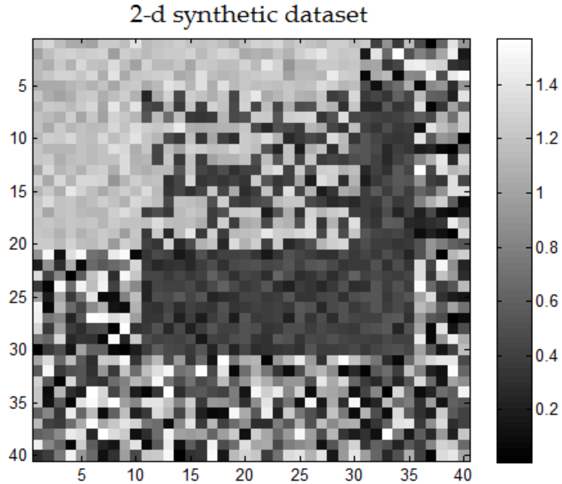
基于熵的Co-Sync



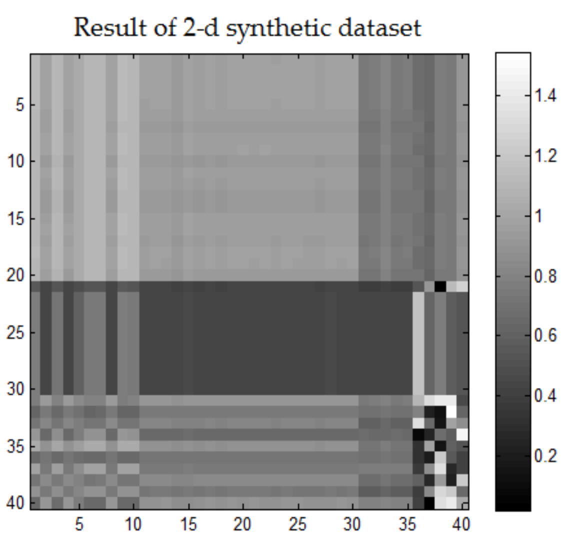
基于指数函数的Co-Sync

4. Co-Sync在人工数据集上的表现

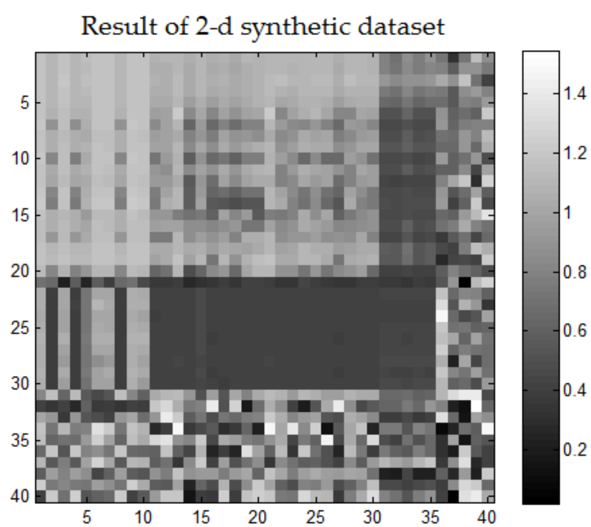
➢挑战：双pattern混合实验



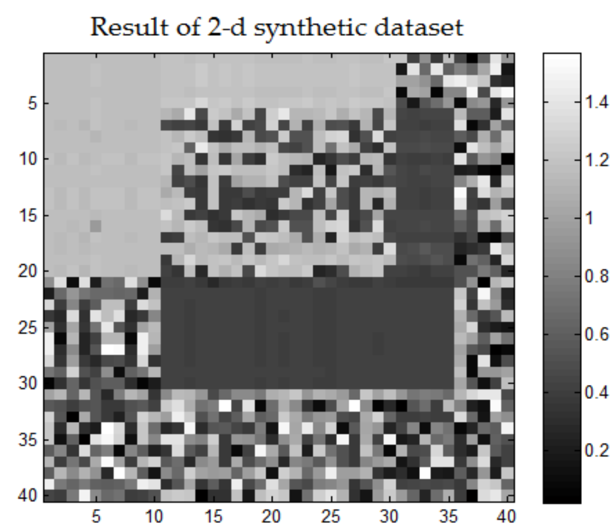

```
mu1 = 1.2;
mu2 = 0.4;
sigma = 0.1;
n = 40;
data = rand(n,n)*pi/2;
data(1:20,1:30) = mu1 + sigma * randn(20,30);
data(6:30,11:35) = mu2 + sigma * randn(25,25);
randnum = randi(2,15,20);
overlapping = data(6:20,11:30);
temp = mu1 + randn(size(overlapping)) * sigma;
overlapping(randnum == 2) = temp(randnum==2 );
data(6:20,11:30) = overlapping;
```



原始Co-Sync



基于熵的Co-Sync



基于指数函数的Co-Sync

5. 阶段总结

➢已经完成的工作

1. 阅读了大量相关论文，夯实基础。
2. 确定并实现了Co-Sync的基本框架。
3. 解释并找到了合适的方法来消除“边际效应”。
4. 在人工数据集上进行了大量实验，得到了目前比较理想的结果。

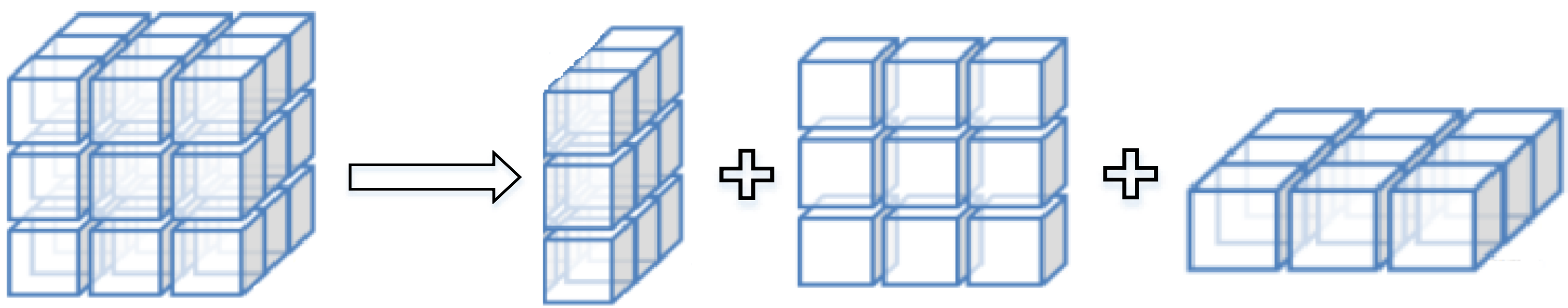
➢后续工作

1. 对人工数据集上一些特殊情况效果仍然不理想，并且方法对指数函数中的参数不够鲁班，需要找寻进一步的解决方案。
2. Co-Sync要求数据集不能稀疏，需确定比较好的实际数据集，确保Co-Sync可以很好的发挥作用。同时提出适合Co-Sync的归一化方法与缺失值处理方法。
3. 对找出的pattern进行自动识别。
4. 整理工作，写论文。之后对该算法的其他可能方案进行思考并扩展

6. 后续思考

➢Multi-Sync：

数据立方体：将$(M*N*T)$数据立方体拆解为$(M$个$M*T + N$个$M*T +$ T个$M*N)$的矩阵，分别进行Co-Sync算法后将改变量矩阵叠加进原矩阵，迭代这一过程。



高维tensor：思路一样， 将其拆解为很多二维空间的数据矩阵。其中d维空间的tensor可拆解在 C_d^2 个方向上的子矩阵。

Thanks

英才实验学院

高宗铭

2012001010016

指导教师:邵俊明