Appendix

下图生成代码

h = ezplot('exp(-4\*x)',[0,2]);

axis([0,1.5,0,1]);

set(h,'LineWidth',1.5);

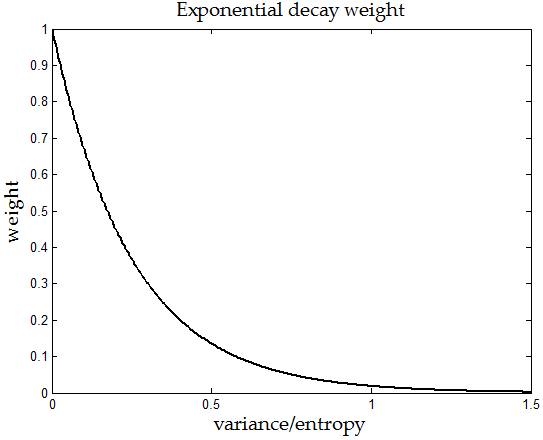
set(h,'Color',[0,0,0]);

title('Exponential decay weight','Color', 'k', 'fontsize', 15,'fontname','Palatino Linotype');

xlabel('variance/entropy','Color','k','fontsize',15,'fontname','Palatino Linotype');

ylabel('weight','Color','k','fontsize',15, 'fontname', 'Palatino Linotype');

set(gcf,'Color',[1,1,1]);



Based on Variance: ,

由于variance在(,0.3),故 通常取

Based on Entropy :

由于entropy在(0.1,2),故 通常取

Alpha 和 beita 取大了不能converge，取小了有边际效应。

* **测试1：单个pattern**

mu1 = 1.2;

mu2 = 0.4;

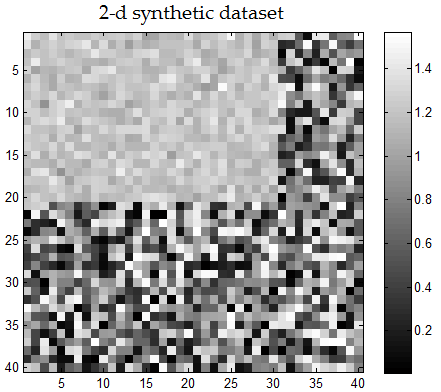
sigma = 0.1;

n = 40;

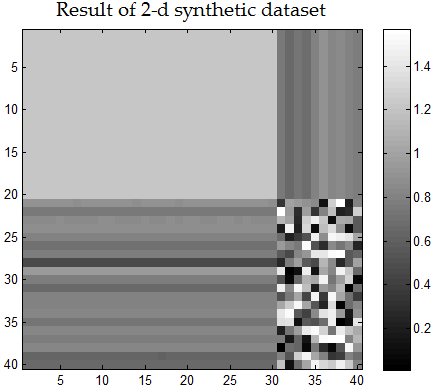
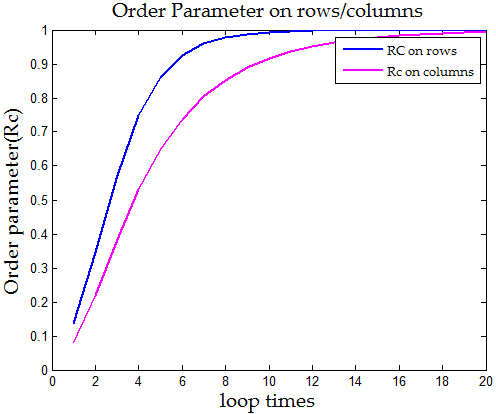
data = rand(n,n)\*pi/2;

data(1:20,1:30) = mu1 + sigma \* randn(20,30);

**1．原始matrix**



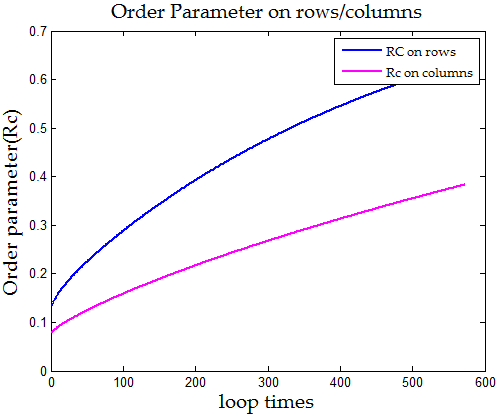
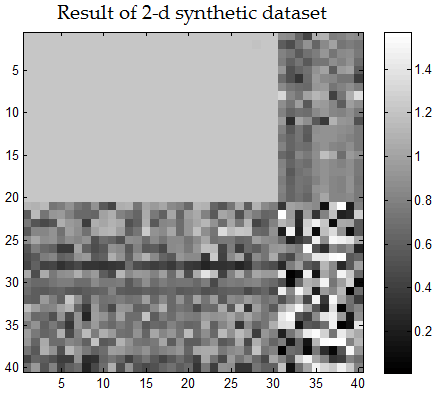
**2．Co-Sync结果**

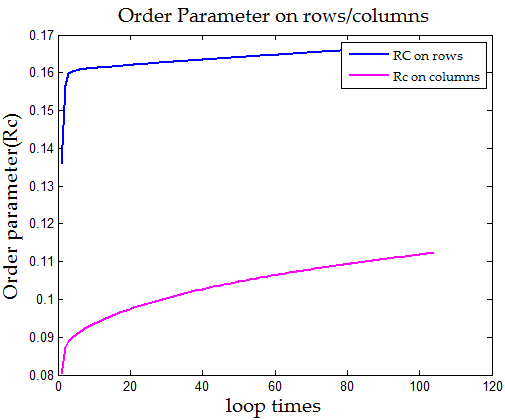
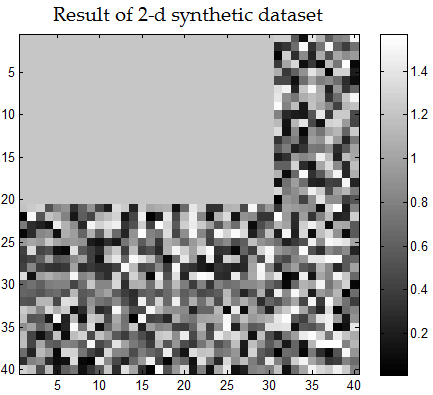
static double lambda = -50; // exp(lambda \* variance(variant))

static double beita = -2;//-Math.E; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**



**4．Co-Sync based on Variance 结果**



* **测试2：单个pattern,加大其Divergence，增加识别难度**

mu1 = 0.8;

mu2 = 0.4;

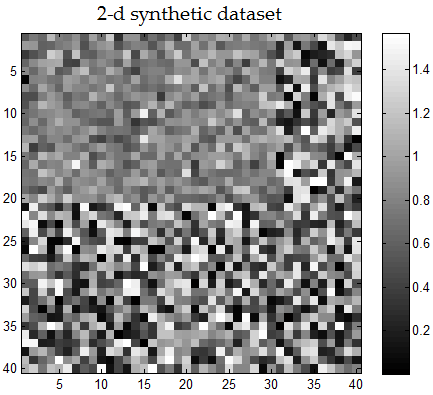
sigma = 0.2;

n = 40;

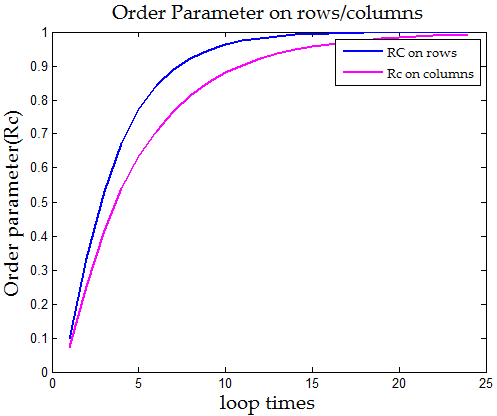
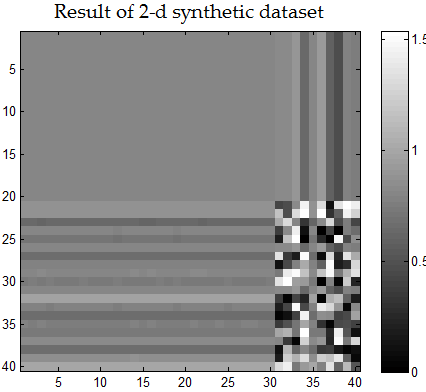
data = rand(n,n)\*pi/2;

data(1:20,1:30) = mu1 + sigma \* randn(20,30);

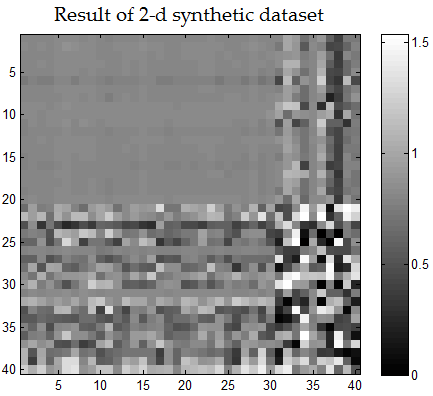
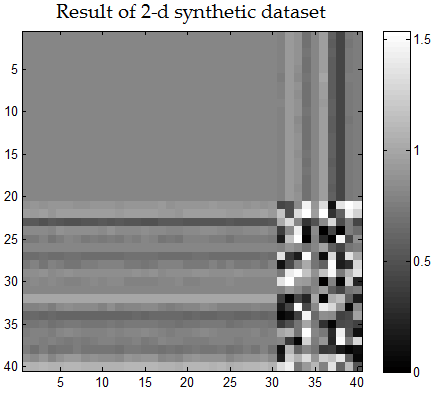
**1．原始matrix**



**2．Co-Sync结果**



**3．Co-Sync based on Entropy 结果**

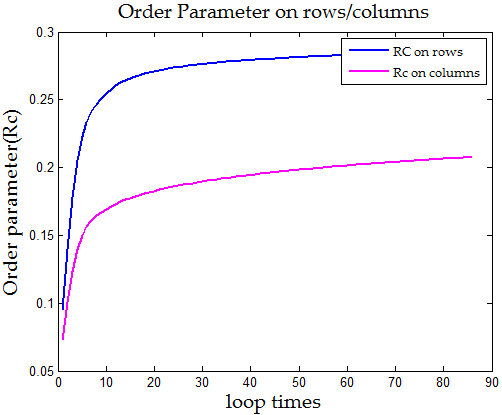
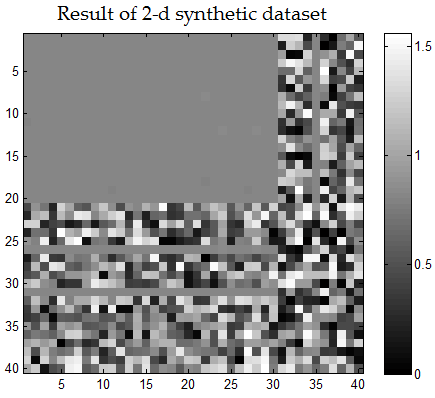


Beita = -1 beita = -2

**注意**上图中左右图衰减因子beita的值不一样，左边的pattern找到了，但出现solidarity effect，右边solidarity effect不明显，但pattern没有converge.这说明基于Entropy方法本身的局限性。

**4．Co-Sync based on Variance 结果**

static double lambda = -50;



* **测试3：双个pattern,without overlapping**

mu1 = 1.2;

mu2 = 0.5;

sigma = 0.1;

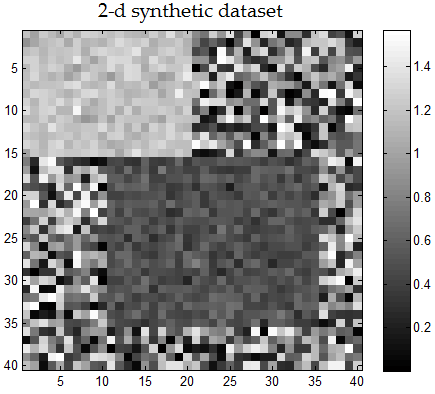
n = 40;

data = rand(n,n)\*pi/2;

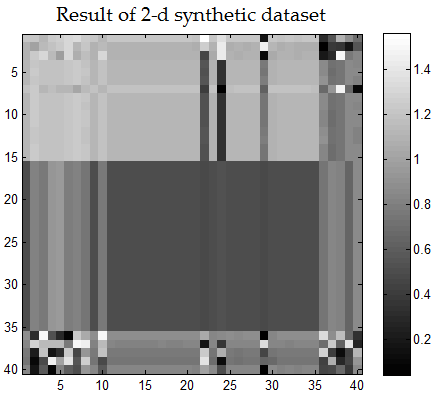
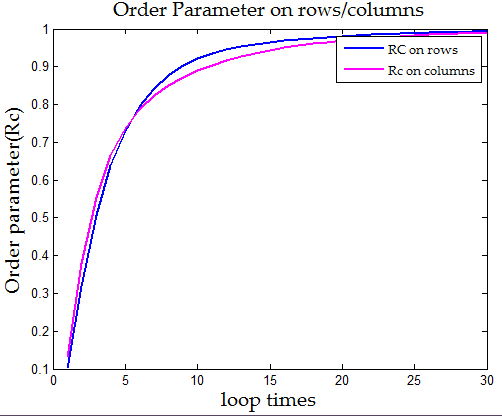
data(1:15,1:20) = mu1 + sigma \* randn(15,20);

data(16:35,11:35) = mu2 + sigma \* randn(20,25);

**1．原始matrix**



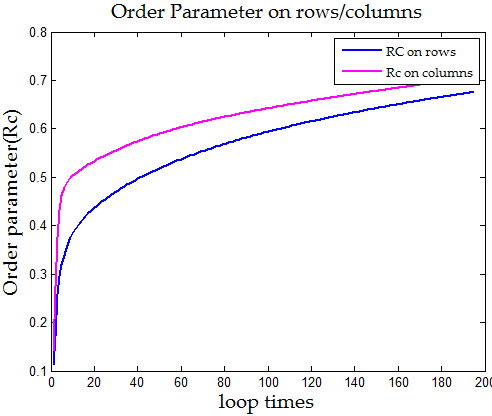
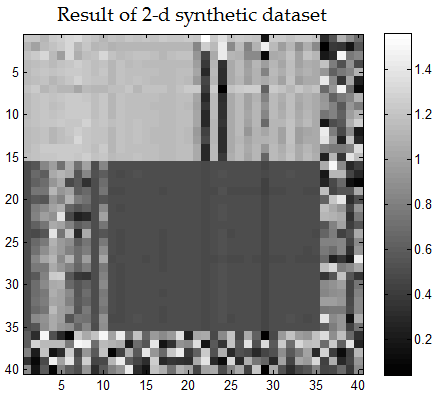
**2．Co-Sync结果**

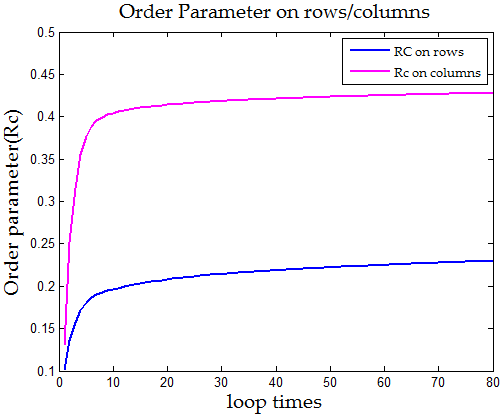
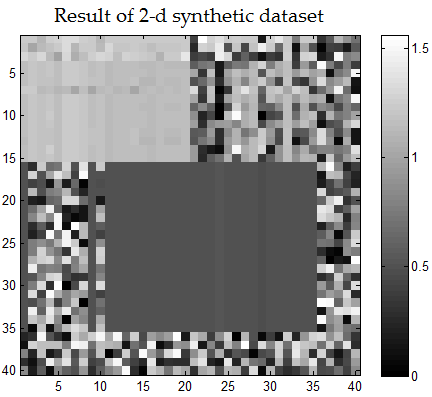
static double lambda = -100; // exp(lambda \* variance(variant))

static double beita = -1.8; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**



**4．Co-Sync based on Variance 结果**



* **测试4：双个pattern,without overlapping**

mu1 = 1.2;

mu2 = 0.5;

sigma = 0.1;

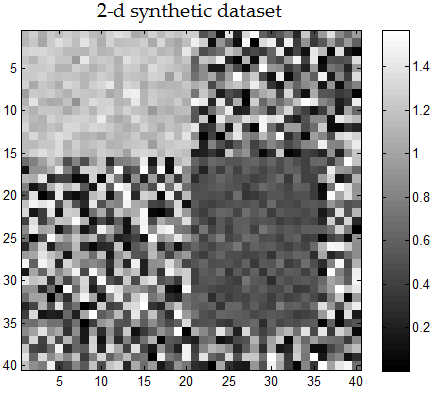
n = 40;

data = rand(n,n)\*pi/2;

data(1:15,1:20) = mu1 + sigma \* randn(15,20);

data(16:35,21:35) = mu2 + sigma \* randn(20,15);

**1．原始matrix**

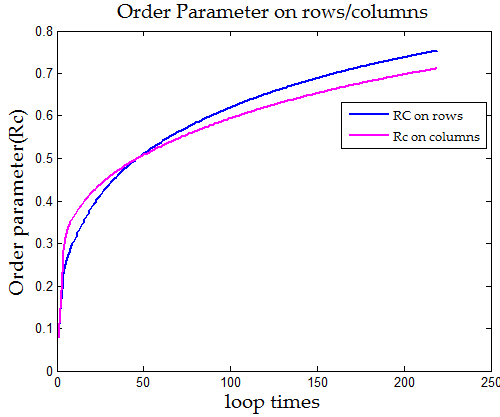
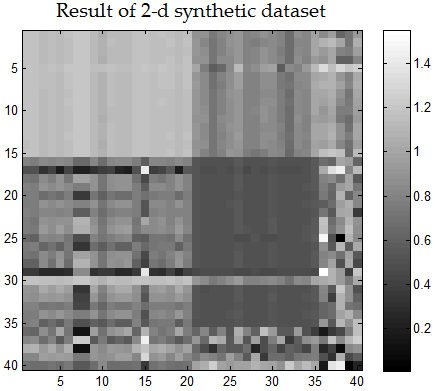


**2．Co-Sync结果（略）**

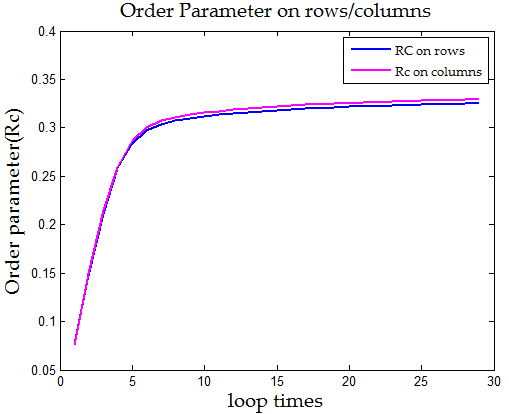
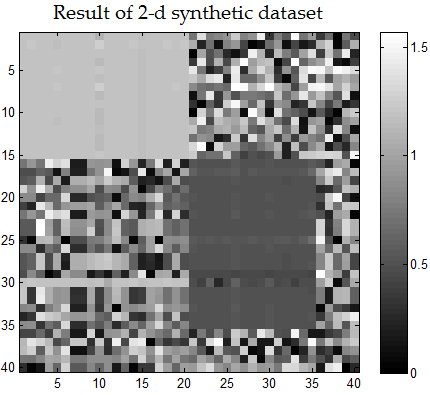
static double lambda = -100; // exp(lambda \* variance(variant))

static double beita = -2; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**



**4．Co-Sync based on Variance 结果**



* **测试5：双个pattern,without overlapping**

mu1 = 1.2;

mu2 = 0.5;

sigma = 0.1;

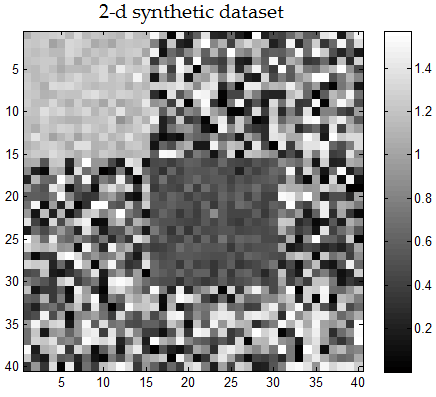
n = 40;

data = rand(n,n)\*pi/2;

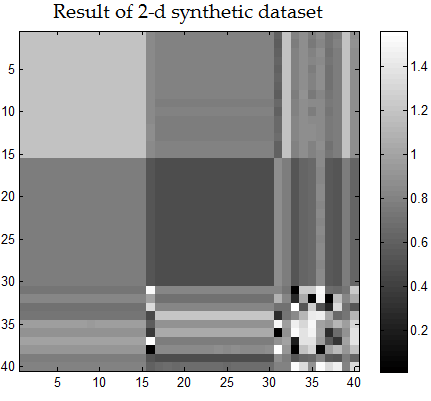
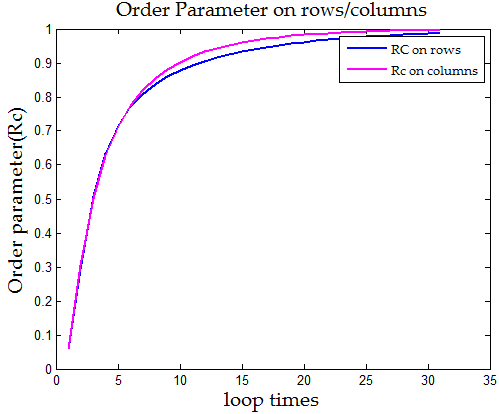
data(1:15,1:15) = mu1 + sigma \* randn(15,15);

data(16:30,16:30) = mu2 + sigma \* randn(15,15);

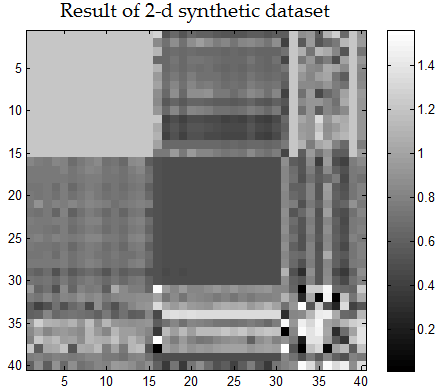
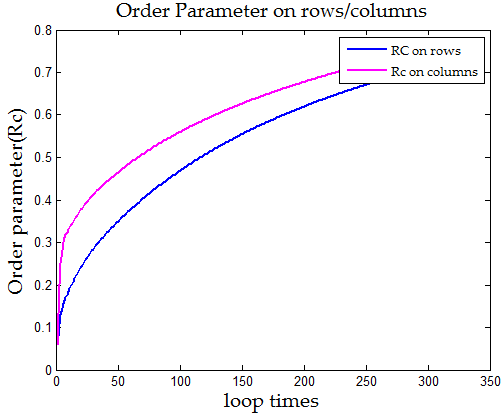
**1．原始matrix**



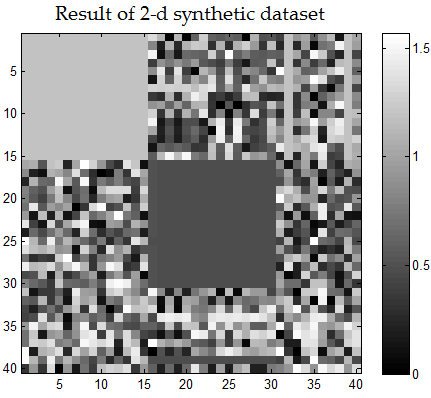
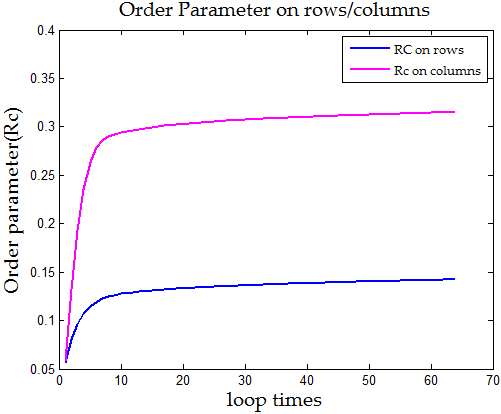
**2．Co-Sync结果**

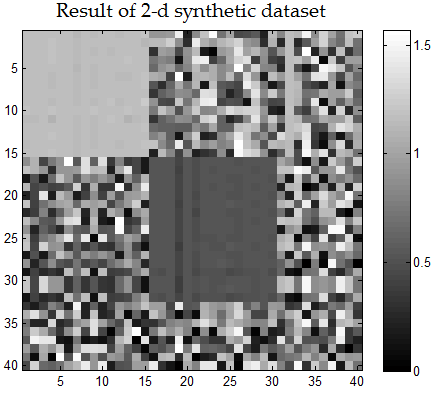
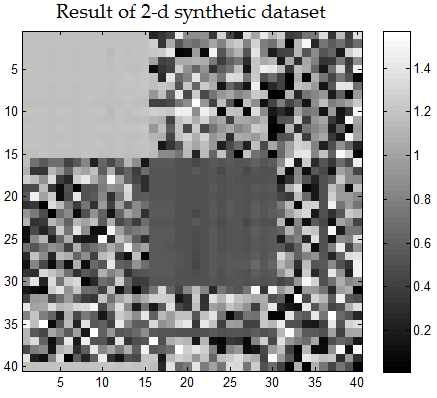
**3．Co-Sync based on Entropy 结果**

**4．Co-Sync based on Variance 结果**

值得玩味的上面这个图，看起来很理想，非常棒，但是其产生确实有一定概率的。如果用MATLAB在相同参数下重新初始化数据集一遍，得出的结果就没这么完美。如下图：

* **测试6：双个pattern,without overlapping**

mu1 = 1.2;

mu2 = 0.5;

sigma = 0.1;

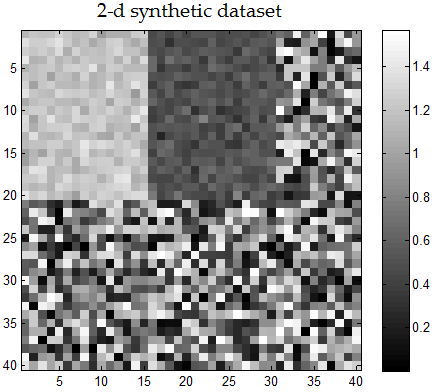
n = 40;

data = rand(n,n)\*pi/2;

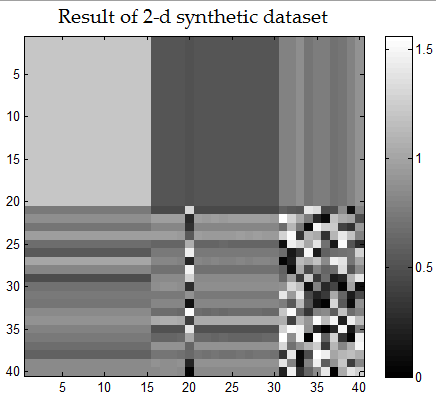
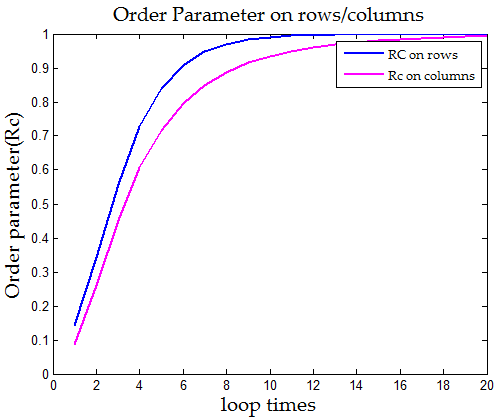
data(1:20,1:15) = mu1 + sigma \* randn(20,15);

data(1:20,16:30) = mu2 + sigma \* randn(20,15);

**1．原始matrix**



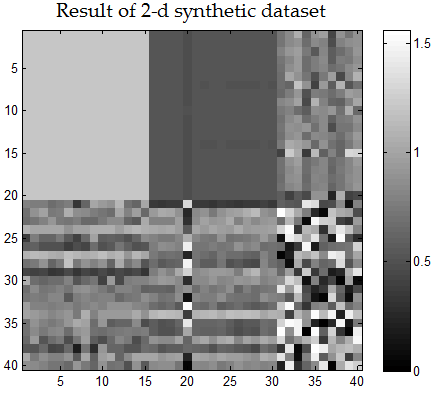
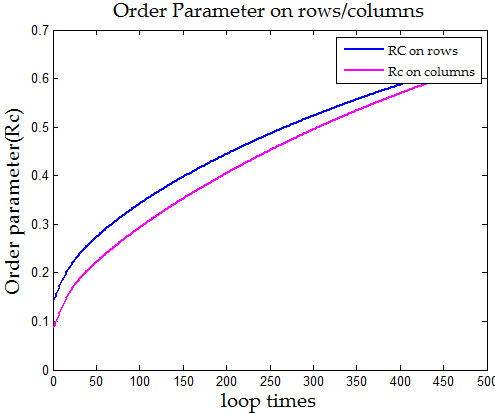
**2．Co-Sync结果**

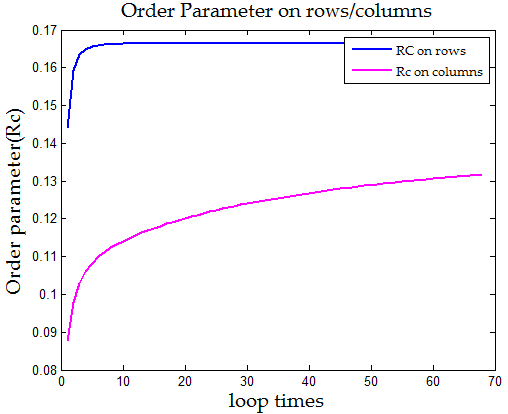
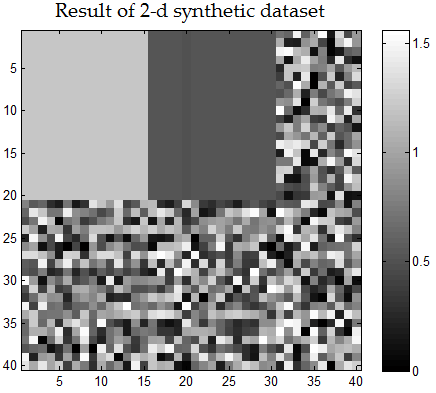
static double lambda = -100; // exp(lambda \* variance(variant))

static double beita = -2; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**

**4．Co-Sync based on Variance 结果**



* **测试7：双个pattern,without overlapping.减小两个pattern差距，其在图中的明暗差距减小！**

mu1 = 0.8;

mu2 = 0.5;

sigma = 0.1;

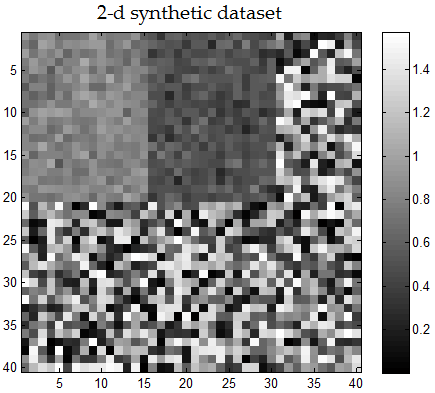
n = 40;

data = rand(n,n)\*pi/2;

data(1:20,1:15) = mu1 + sigma \* randn(20,15);

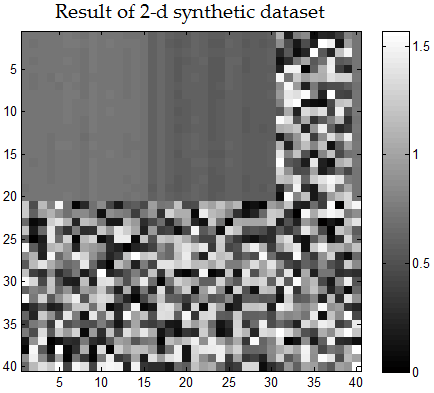
data(1:20,16:30) = mu2 + sigma \* randn(20,15);

**1．原始matrix**



**2．Co-Sync based on Variance 结果**

Based on variance 的结果，观察数据可以看出没有识别出原始的0.8，0.5的均值。靠的近，识别差。



0.6

0.73

**测试8：三个pattern,without overlapping.**

mu1 = 0.8;

mu2 = 0.3;

mu3 = 1.3

sigma = 0.1;

n = 40;

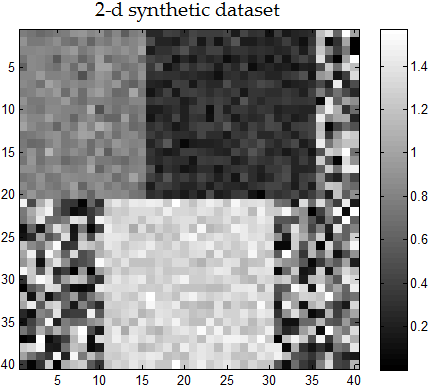
data = rand(n,n)\*pi/2;

data(1:20,1:15) = mu1 + sigma \* randn(20,15);

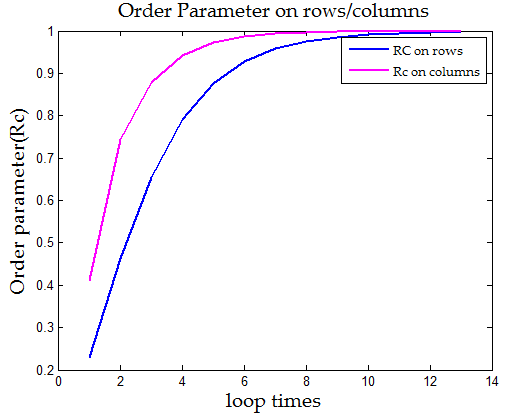
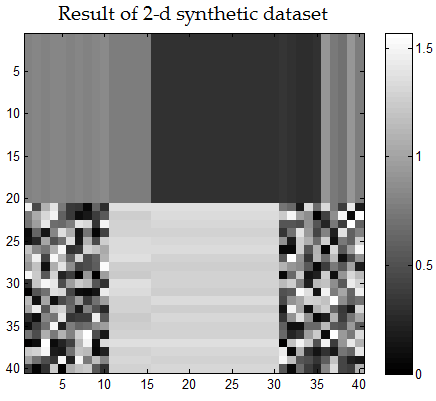
data(1:20,16:35) = mu2 + sigma \* randn(20,20);

data(21:40,11:30) = mu3 + sigma \* randn(20,20);

**1．原始matrix**



**2．Co-Sync结果**

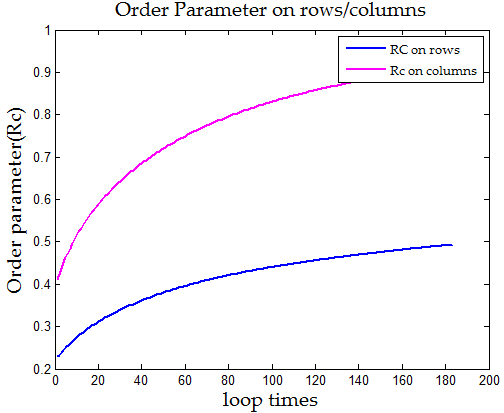
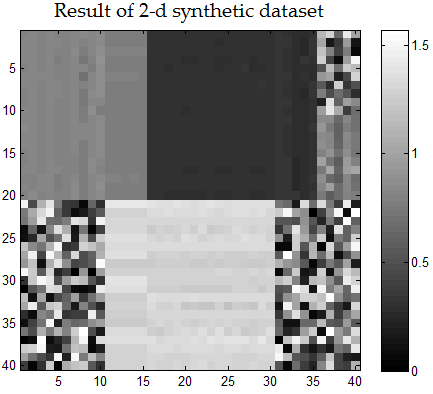


在这个图中，原始Co-Sync左下角和右下角竟然没有solidarity effect，再观察其下面那个白色sub-matrix，主要原因是因为其左右和上下的拉扯抵消的原因！

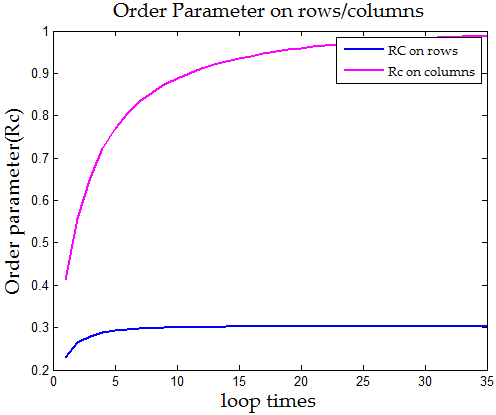
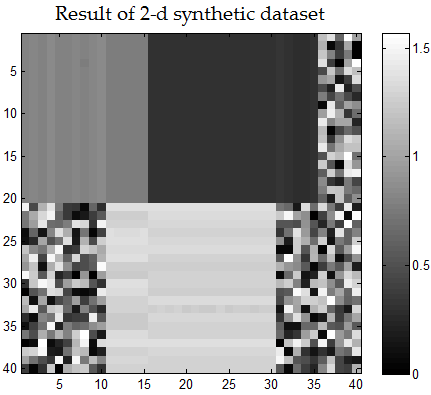
static double lambda = -100; // exp(lambda \* variance(variant))

static double beita = -2; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**



**4．Co-Sync based on Variance 结果**



* **测试9：三个Pattern，without overlapping**

mu1 = 0.8;

mu2 = 0.3;

mu3 = 1.3;

% mu4 = 1.0;

sigma = 0.1;

n = 40;

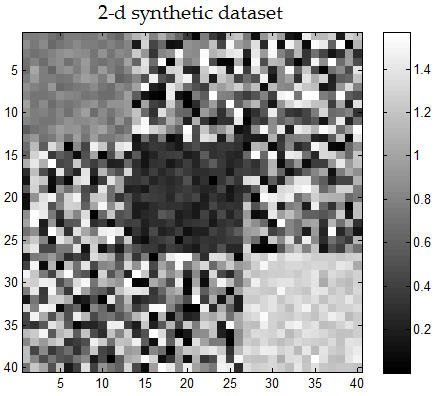
data = rand(n,n)\*pi/2;

data(1:13,1:13) = mu1 + sigma \* randn(13,13);

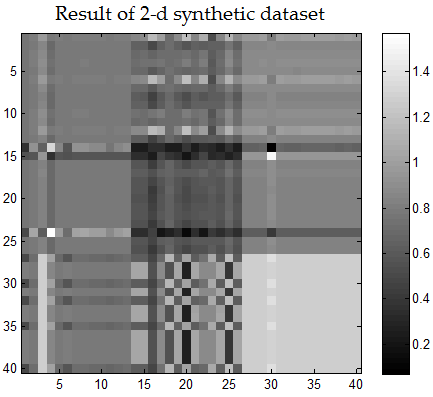
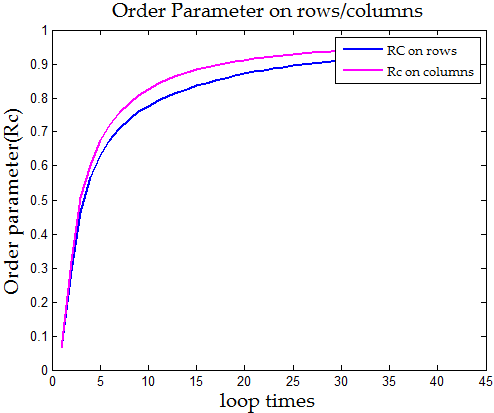
data(14:26,14:26) = mu2 + sigma \* randn(13,13);

data(27:40,27:40) = mu3 + sigma \* randn(14,14);

**1．原始matrix**



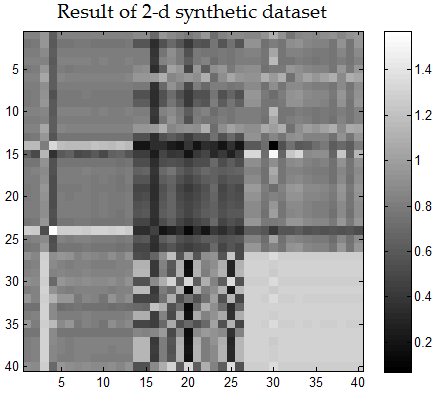
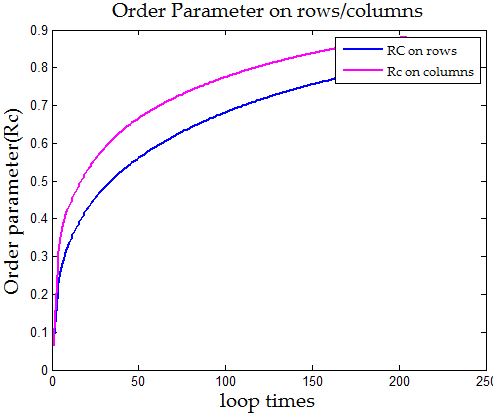
**2．Co-Sync结果**

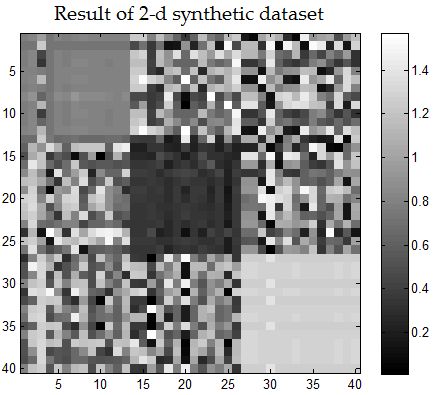
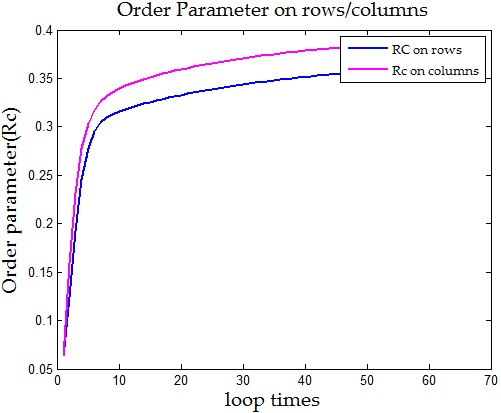
static double lambda = -100; // exp(lambda \* variance(variant))

static double beita = -2; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**

**4．Co-Sync based on Variance 结果**

* **测试10：三个Pattern，without overlapping**

mu1 = 0.8;

mu2 = 0.3;

mu3 = 1.3

sigma = 0.1;

n = 40;

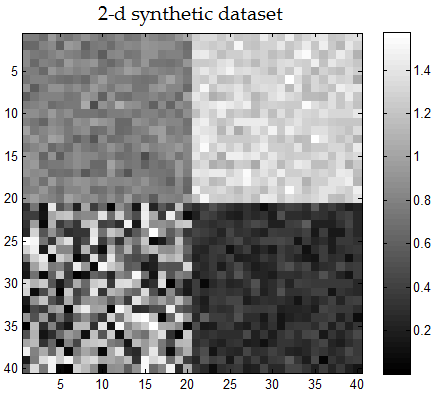
data = rand(n,n)\*pi/2;

data(1:20,1:20) = mu1 + sigma \* randn(20,20);

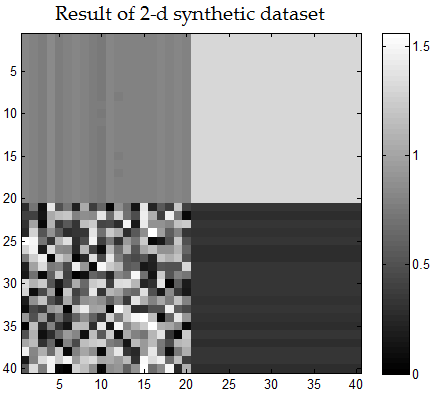
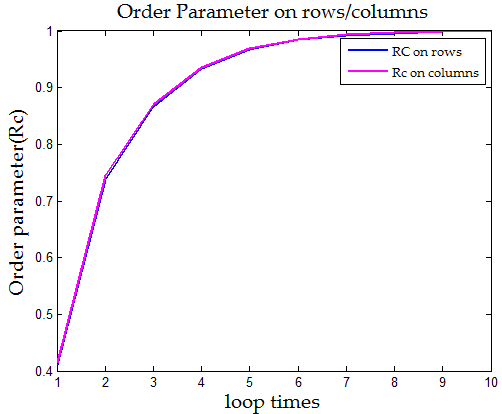
data(21:40,21:40) = mu2 + sigma \* randn(20,20);

data(1:20,21:40) = mu3 + sigma \* randn(20,20);

**1．原始matrix**



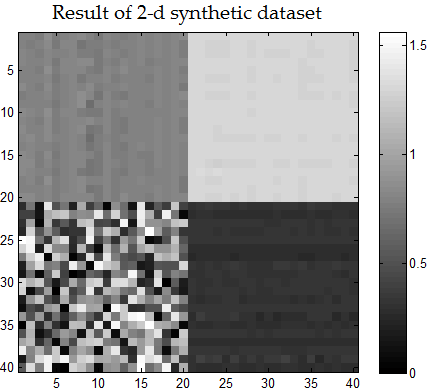
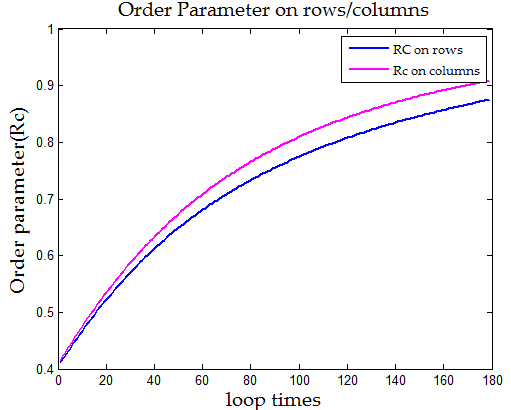
**2．Co-Sync结果**

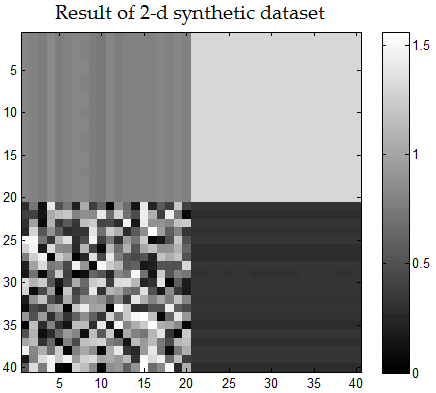
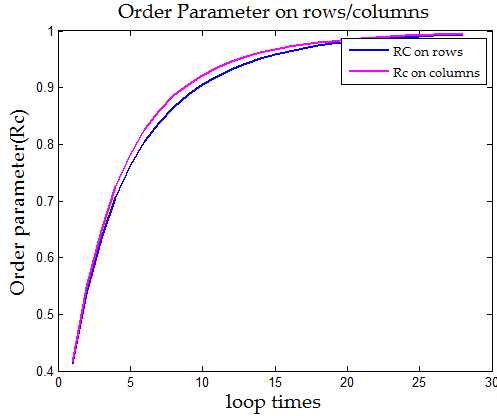
static double lambda = -100; // exp(lambda \* variance(variant))

static double beita = -2; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**

**4．Co-Sync based on Variance 结果**

* **测试11：三个Pattern，without overlapping**

mu1 = 0.6;

mu2 = 0.2;

mu3 = 1.4;

mu4 = 1.0;

sigma = 0.1;

n = 40;

data = rand(n,n)\*pi/2;

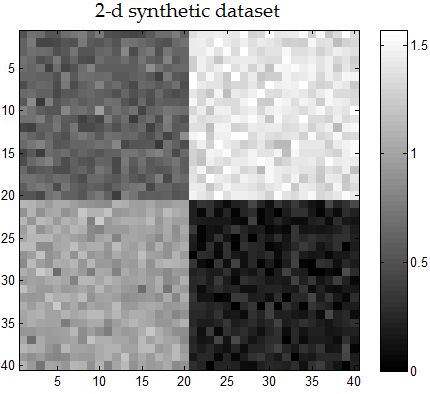
data(1:20,1:20) = mu1 + sigma \* randn(20,20);

data(21:40,21:40) = mu2 + sigma \* randn(20,20);

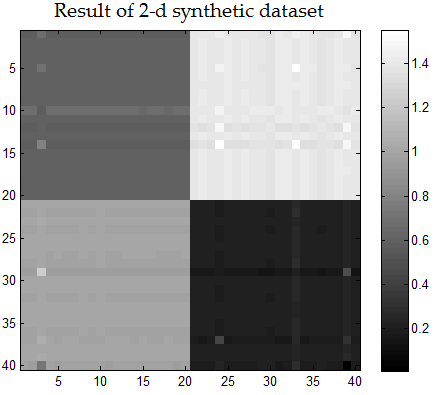
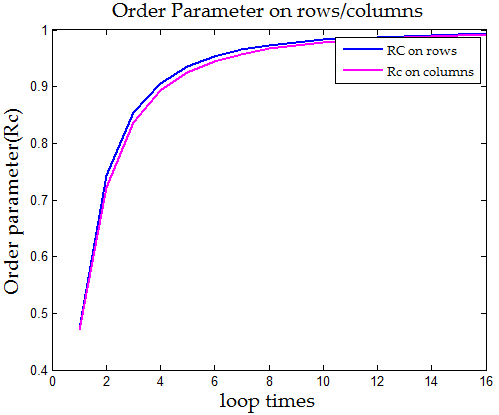
data(1:20,21:40) = mu3 + sigma \* randn(20,20);

data(21:40,1:20) = mu4 + sigma \* randn(20,20);

**1．原始matrix**



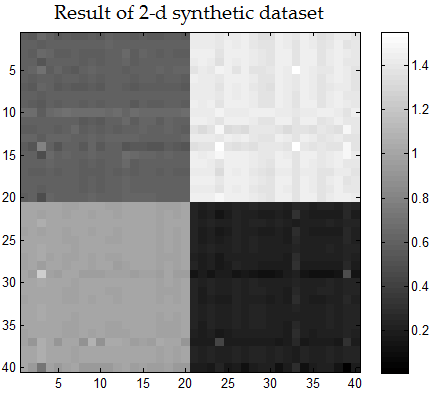
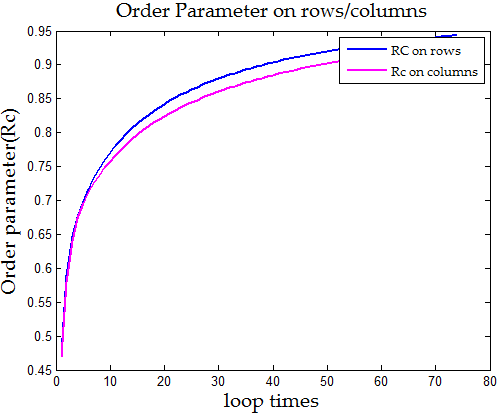
**2．Co-Sync结果**

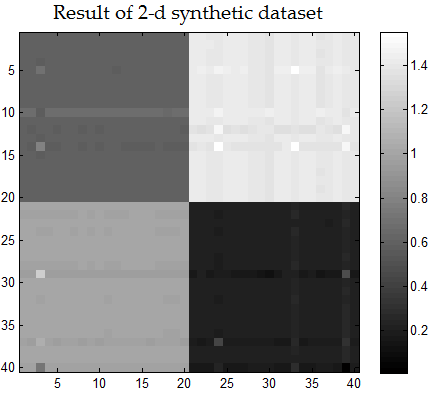
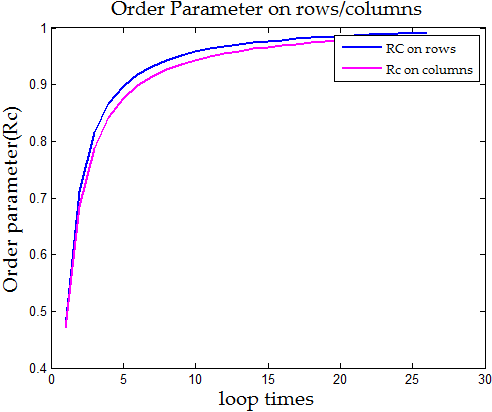
static double lambda = -100; // exp(lambda \* variance(variant))

static double beita = -2; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**

**4．Co-Sync based on Variance 结果**

* **测试12：2个Pattern，with overlapping**

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);

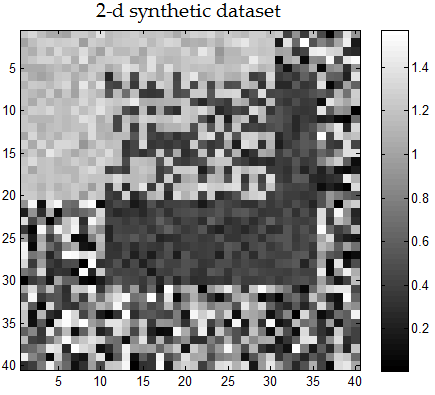
overlapping = data(6:20,11:30);

temp = mu1 + randn(size(overlapping)) \* sigma;

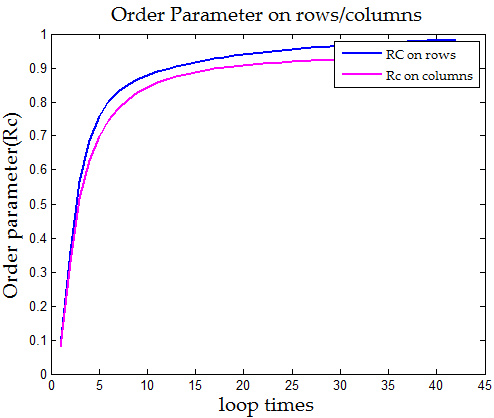
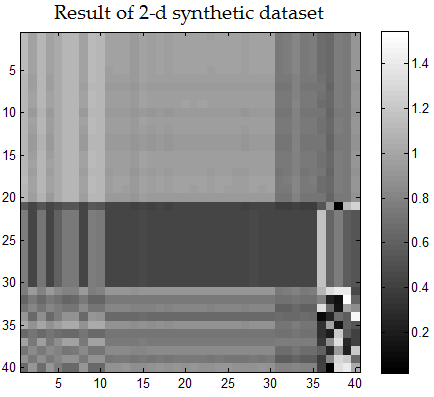
overlapping(randnum == 2) = temp(randnum==2 );

data(6:20,11:30) = overlapping;

**1．原始matrix**



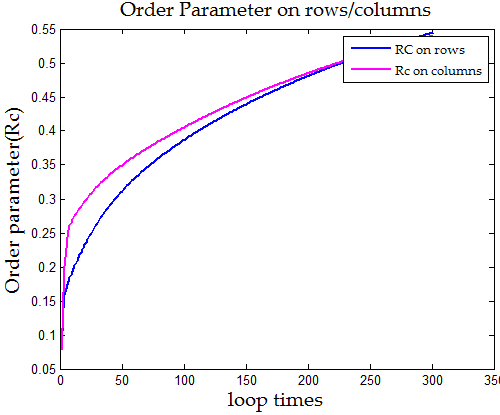
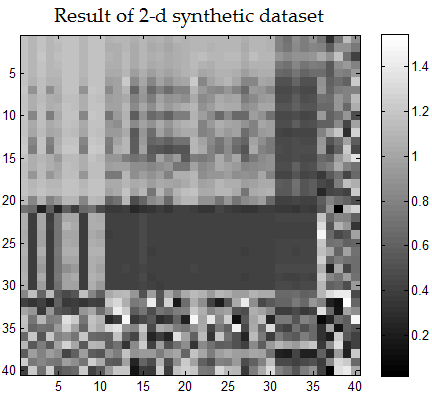
**2．Co-Sync结果**



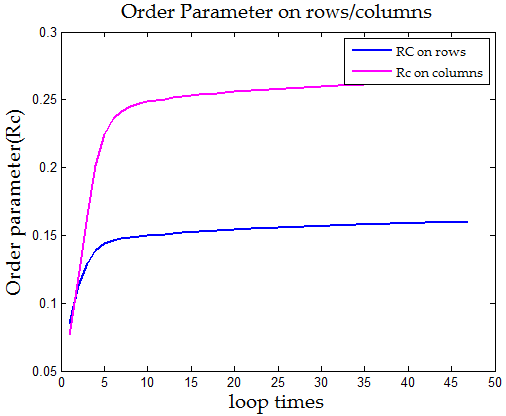
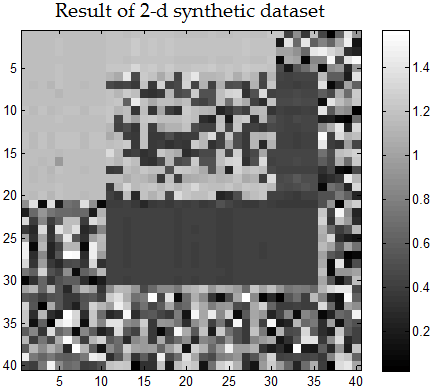
static double lambda = -100; // exp(lambda \* variance(variant))

static double beita = -2.8; // exp(beita \* entropy(variant));

**3．Co-Sync based on Entropy 结果**



**4．Co-Sync based on Variance 结果**



* **Conclusion:**

1、关于算法性能：Co-Sync based on Variance 任何时候都是最好的！而Co-Sync based on Entropy在多数时候比原始Co-Sync好，位居第二。Co-Sync大多数位居。而用PCA思想的，时间复杂度成百上千地增长、且结果还没有原始Co-Sync好。

2、几种算法的成功在synthetic dataset上都有一定概率，这个不确定性来自初始数据集。若小概率的偏移出现，则最后几种算法找出的pattern都不是很理想。

3、即使是效果最好的Co-Sync based on Variance，要找到最好的结果也存在一定的trade off between Convergence of pattern and clarity of solidarity effect. 但是这种取舍却要人来主观判断。

4、Co-Sync based on Variance/Entropy两者的Order parameter不能收敛到1，每次退出的条件都是：.

5、最关键的一点，指数衰减有一个衰减因子，这个衰减因子也需要人为定，而且会根据数据集变化而变化。而人为定的根据，是MATLAB可视化出的颜色图。对于大型数据集以及立方体失效。

6、也许我不太明白老师意思，overlapping的方式不对，上面的测试12在我看来说明不了什么。

其次是我发现的一个Sync算法本身的缺陷：

每次Interact的objects都是初始化中找到的一些邻居点。Interact range而不能随迭代而改变。这导致了若初始时候把一些远距离的噪声点加入neighbor list 后，之后会被该噪声一直影响。

mu1 = 1.2;

mu2 = 0.5;

sigma = 0.1;

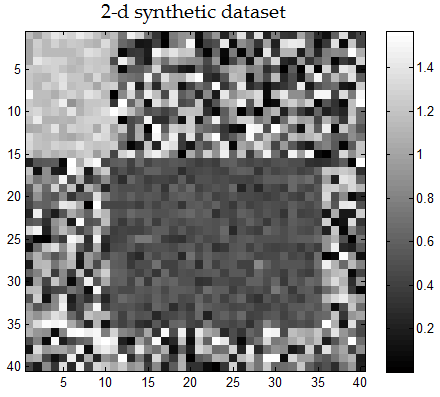
n = 40;

data = rand(n,n)\*pi/2;

data(1:15,1:10) = mu1 + sigma \* randn(15,10);

data(16:35,11:35) = mu2 + sigma \* randn(20,25);

**1．原始matrix**



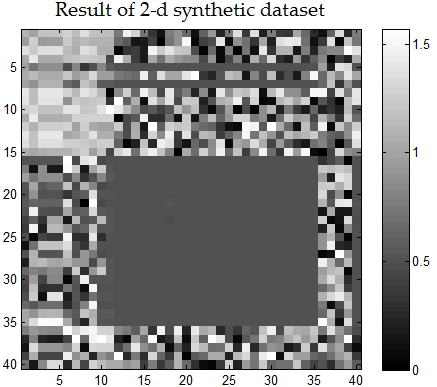
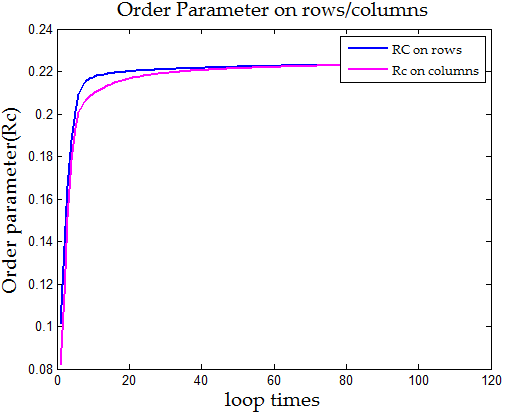
static int K = 4; //KNN 步骤中的K值

static int loopNum = 500;

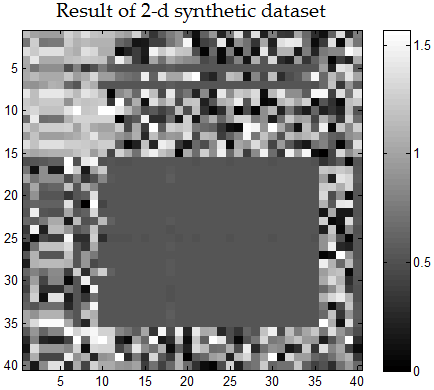
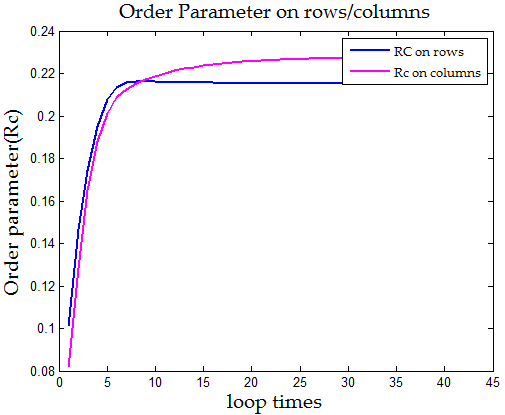
static double converge = 1e-4;

static double lambda = -100;

**2．Co-Sync based on Variance with exponential decay function结果**

**3．Co-Sync based on Variance with mean truncation point结果**

结果分析：

1. 该实验能看出左上角的小pattern由于size较小，由于高维诅咒的原因不能被基于距离的方法检测出来。所以需要提出新的方法改进。
2. 用方差的均值作为截断点的方案，收敛次数为45次，而用指数衰减的收敛次数为120次。但是把图放大了看，用截断点的方案结果中，噪声点更多！故其也不是最好的方案！