拉普拉斯正则化表示实验记录

代码用法和试验记录

代码用法和试验记录
- 试验概述
- sparse_graph.m
- fset.m
  - 使用kNN处理
    - Z
    - E
  - 使用距离处理
    - Z
    - E
  - Y*Z+E
- fpic.m
  - 使用knn处理
    - Z
    - E
  - 使用距离处理
    - Z
    - E
  - 分析
- fLRR.m
  - 原图使用kNN处理
    - Z
    - E
  - 原图使用距离处理
    - Z
    - E
  - 原图添加相同列向量kNN处理
    - Z
    - E
  - 原图添加相同列向量距离处理
    - Z
    - E
- 试验总结

试验概述

主要包含有sparse_graph.m、fset.m、fpic.m、fLRR.m和fkNN.m三个.m文件，每一个文件都可以测试普通kNN和kNN之后使用距离判断的差别，只要注释相应的区域即可

sparse_graph.m

这是论文的源代码，主要参数是

```
function [Z, E] =   sparse_graph_LRR(X, W, lambda, garma, beta, rho, DEBUG)
% inputs:
%        X -- D*N data matrix
%        W -- affinity graph matrix
```

其他参数是我照着论文设置的，修改了几次影响似乎不大

fset.m

这个是我自己设计的一个简单数据集，形如下图的100*100的矩阵，这个矩阵每一列元素之间的欧式距离相同，相似度也相同，从理论上来说，对角元素最大，其他元素相同的矩阵

$$\begin{Bmatrix} 4 & 5 & 5 \\ 5 & 4 & 5 \\ 5 & 5 & 4 \end{Bmatrix}$$

使用kNN处理

Z

	1	2	3	4	5	6	7	8	9	10
1	0.0359	-0.0059	-0.0059	-0.0059	-0.0060	-0.0060	-0.0060	-0.0060	-0.0060	-0.0060
2	-0.0059	0.0359	-0.0059	-0.0059	-0.0060	-0.0060	-0.0060	-0.0060	-0.0060	-0.0060
3	-0.0059	-0.0059	0.0359	-0.0059	-0.0060	-0.0060	-0.0060	-0.0060	-0.0060	-0.0060
4	-0.0059	-0.0059	-0.0059	0.0359	-0.0060	-0.0060	-0.0060	-0.0060	-0.0060	-0.0060
5	-0.0058	-0.0058	-0.0058	-0.0058	0.8600	-0.0144	-0.0144	-0.0144	-0.0144	-0.0144
6	-0.0058	-0.0058	-0.0058	-0.0058	-0.0144	0.8600	-0.0144	-0.0144	-0.0144	-0.0144
7	-0.0058	-0.0058	-0.0058	-0.0058	-0.0144	-0.0144	0.8600	-0.0144	-0.0144	-0.0144
8	-0.0058	-0.0058	-0.0058	-0.0058	-0.0144	-0.0144	-0.0144	0.8600	-0.0144	-0.0144
9	-0.0058	-0.0058	-0.0058	-0.0058	-0.0144	-0.0144	-0.0144	-0.0144	0.8600	-0.0144
10	-0.0058	-0.0058	-0.0058	-0.0058	-0.0144	-0.0144	-0.0144	-0.0144	-0.0144	0.8600

可以看出来，对角元素不是1，而且元素不全相同，同时每一行元素和每一列元素从第五个开始有很明显的下降，这是因为k=5只能找到前五个元素，只把前五个元素认为是与自己相似的元素

E

	1	2	3	4	5	6	7	8	9	10
1	0.1532	1.1113	1.1113	1.1113	1.0964	1.0964	1.0964	1.0964	1.0964	1.0964
2	1.1113	0.1532	1.1113	1.1113	1.0964	1.0964	1.0964	1.0964	1.0964	1.0964
3	1.1113	1.1113	0.1532	1.1113	1.0964	1.0964	1.0964	1.0964	1.0964	1.0964
4	1.1113	1.1113	1.1113	0.1532	1.0964	1.0964	1.0964	1.0964	1.0964	1.0964
5	1.1115	1.1115	1.1115	1.1115	0.9624	1.0880	1.0880	1.0880	1.0880	1.0880
6	1.1115	1.1115	1.1115	1.1115	1.0880	0.9624	1.0880	1.0880	1.0880	1.0880
7	1.1115	1.1115	1.1115	1.1115	1.0880	1.0880	0.9624	1.0880	1.0880	1.0880
8	1.1115	1.1115	1.1115	1.1115	1.0880	1.0880	1.0880	0.9624	1.0880	1.0880
9	1.1115	1.1115	1.1115	1.1115	1.0880	1.0880	1.0880	1.0880	0.9624	1.0880
10	1.1115	1.1115	1.1115	1.1115	1.0880	1.0880	1.0880	1.0880	1.0880	0.9624

噪声矩阵也受到k的影响，从第五列开始有较大的下滑

使用距离处理

Z

	1	2	3	4	5	6	7	8	9	10
1	0.0373	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053
2	-0.0053	0.0373	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053
3	-0.0053	-0.0053	0.0373	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053
4	-0.0053	-0.0053	-0.0053	0.0373	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053
5	-0.0053	-0.0053	-0.0053	-0.0053	0.0373	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053
6	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	0.0373	-0.0053	-0.0053	-0.0053	-0.0053
7	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	0.0373	-0.0053	-0.0053	-0.0053
8	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	0.0373	-0.0053	-0.0053
9	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	0.0373	-0.0053
10	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	-0.0053	0.0373

]

可以看到，使用距离之后明显的看到了各行之间的元素相似度相同

E

	1	2	3	4	5	6	7	8	9	10
1	0	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930
2	0.8930	0	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930
3	0.8930	0.8930	0	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930
4	0.8930	0.8930	0.8930	0	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930
5	0.8930	0.8930	0.8930	0.8930	0	0.8930	0.8930	0.8930	0.8930	0.8930
6	0.8930	0.8930	0.8930	0.8930	0.8930	0	0.8930	0.8930	0.8930	0.8930
7	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0	0.8930	0.8930	0.8930
8	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0	0.8930	0.8930
9	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0	0.8930
10	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0.8930	0

可以看到，噪声也恢复一样

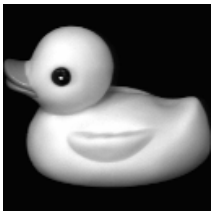
Y*Z+E

特别的是，这个简单数据集的恢复很差，和原始值相差极大，我自己的分析是，算法使用了奇异值分析，将数据里面的异常值4当成了噪声，但是这个噪声的分布又不符合一般噪声的分布规律，所以就会出现还原很差的结果

	1	2	3	4	5	6	7	8	9	10
1	-2.4892	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536
2	-1.5536	-2.4892	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536
3	-1.5536	-1.5536	-2.4892	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536
4	-1.5536	-1.5536	-1.5536	-2.4892	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536
5	-1.5536	-1.5536	-1.5536	-1.5536	-2.4892	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536
6	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-2.4892	-1.5536	-1.5536	-1.5536	-1.5536
7	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-2.4892	-1.5536	-1.5536	-1.5536
8	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-2.4892	-1.5536	-1.5536
9	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-2.4892	-1.5536
10	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-1.5536	-2.4892

fpic.m

这个是我加入了我自己使用的图片数据集，放入了两组目标，分别以下两组，并且是按照顺序读入的，便于观测




使用knn处理

Z

结果是一个对称矩阵，对角元素为1

	1	2	3	4	5	6	7	8	9	10
1	1.0000	5.4972e-10	5.4972e-10	5.4972e-10	-2.2033e-17	2.0099e-17	4.1418e-16	1.0354e-16	2.3880e-16	4.4381e-16
2	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	1.3019e-15	-1.3506e-15	2.1236e-16	4.4004e-16	6.2272e-16	6.0691e-16
3	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	5.4972e-10	-5.0118e-16	-3.2948e-16	-4.4401e-16	-3.1274e-16
4	5.4972e-10	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	5.4972e-10	7.7215e-16	4.0106e-16	3.7867e-16
5	6.1234e-17	-8.9429e-16	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	7.0018e-16	2.8678e-16	1.8871e-17
6	-2.1534e-17	-4.6588e-16	5.4972e-10	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	-4.2398e-16	7.2924e-17	-1.9048e-16
7	4.4194e-16	-2.0050e-16	-9.9904e-16	5.4972e-10	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	9.3329e-17	1.2114e-16
8	-2.6769e-16	9.3964e-16	6.8967e-16	1.7541e-16	1.7976e-16	1.8491e-16	5.4972e-10	1.0000	5.4972e-10	5.4972e-10
9	2.3706e-16	1.9121e-16	-1.7687e-16	4.1494e-16	1.0219e-15	-7.8273e-16	1.0894e-16	5.4972e-10	1.0000	5.4972e-10
10	1.2635e-16	7.1403e-16	-2.1415e-17	1.9999e-16	-2.3223e-16	-1.5275e-16	8.6967e-16	5.4972e-10	5.4972e-10	1.0000

E

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

使用距离处理

Z

与使用kNN处理的结构完全相同

	1	2	3	4	5	6	7	8	9	10
1	1.0000	5.4972e-10	5.4972e-10	5.4972e-10	-2.2033e-17	2.0099e-17	4.1418e-16	1.0354e-16	2.3880e-16	4.4381e-16
2	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	1.3019e-15	-1.3506e-15	2.1236e-16	4.4004e-16	6.2272e-16	6.0691e-16
3	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	5.4972e-10	-5.0118e-16	-3.2948e-16	-4.4401e-16	-3.1274e-16
4	5.4972e-10	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	5.4972e-10	7.7215e-16	4.0106e-16	3.7867e-16
5	6.1234e-17	-8.9429e-16	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	7.0018e-16	2.8678e-16	1.8871e-17
6	-2.1534e-17	-4.6588e-16	5.4972e-10	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	-4.2398e-16	7.2924e-17	-1.9048e-16
7	4.4194e-16	-2.0050e-16	-9.9904e-16	5.4972e-10	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	9.3329e-17	1.2114e-16
8	-2.6769e-16	9.3964e-16	6.8967e-16	1.7541e-16	1.7976e-16	1.8491e-16	5.4972e-10	1.0000	5.4972e-10	5.4972e-10
9	2.3706e-16	1.9121e-16	-1.7687e-16	4.1494e-16	1.0219e-15	-7.8273e-16	1.0894e-16	5.4972e-10	1.0000	5.4972e-10
10	1.2635e-16	7.1403e-16	-2.1415e-17	1.9999e-16	-2.3223e-16	-1.5275e-16	8.6967e-16	5.4972e-10	5.4972e-10	1.0000

E

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

分析

可以看到在这个数据集里面使用kNN和距离似乎没有任何差别，但是认真分析一下数据的相似度，可以看到，到第一个元素相似度相同的元素是5个，而我的k默认是5，恰好将所有的元素都包含在内，所以我尝试将k变为3，结果受到k的影响，可见这种影响是存在的

	1	2	3	4	5	6	7	8	9	10
1	1.0000	5.4972e-10	4.3043e-16	-8.4486e-16	-5.3538e-16	-6.0740e-17	4.9477e-16	-2.4540e-16	-2.2784e-16	2.1815e-16
2	5.4972e-10	1.0000	5.4972e-10	-4.1862e-16	1.1534e-16	9.9752e-17	-7.9137e-16	-1.3047e-16	1.2466e-16	-1.0423e-15
3	4.7207e-16	5.4972e-10	1.0000	5.4972e-10	5.4972e-10	4.4354e-16	2.4227e-16	-2.6044e-16	9.2971e-17	7.7712e-18
4	-4.3613e-16	-1.0678e-16	5.4972e-10	1.0000	5.4972e-10	4.1362e-16	3.1504e-16	3.2807e-16	-3.6129e-16	5.3982e-16
5	-5.7701e-16	-5.2303e-16	5.4972e-10	5.4972e-10	1.0000	5.4972e-10	6.8250e-16	3.5801e-16	1.0531e-15	-6.0378e-16
6	-1.2167e-17	3.6343e-16	2.4925e-16	-1.7486e-16	5.4972e-10	1.0000	5.4972e-10	7.0267e-16	4.3595e-16	-8.3026e-16
7	5.0517e-16	1.1862e-15	3.4104e-17	4.9826e-16	-8.0783e-17	5.4972e-10	1.0000	5.4972e-10	-8.5565e-16	1.9207e-15
8	2.5073e-16	5.9464e-16	3.4325e-16	5.9382e-16	-5.0935e-16	1.2682e-15	5.4972e-10	1.0000	5.4972e-10	-4.9636e-17
9	3.0906e-16	3.9355e-16	3.8440e-16	1.5636e-16	7.5822e-16	5.5911e-16	-1.2763e-15	5.4972e-10	1.0000	5.4972e-10
10	2.3550e-16	7.6879e-16	2.6798e-16	7.4417e-16	-2.2908e-16	-1.8841e-16	5.7326e-16	3.3352e-16	5.4972e-10	1.0000

fLRR.m

这个是我使用一张图片作为测试，测试算法的对单个图片变换与还原的效果

原图使用kNN处理

Z

	1	2	3	4	5	6	7	8	9	10
1	1.0000	1.6341e-08	1.6341e-08	1.6341e-08	1.6340e-08	1.8498e-10	1.8426e-10	1.8366e-10	1.8421e-10	1.8418e-10
2	1.6342e-08	1.0000	1.6341e-08	1.6341e-08	1.6340e-08	1.8479e-10	1.8404e-10	1.8341e-10	1.8392e-10	1.8390e-10
3	1.6341e-08	1.6341e-08	1.0000	1.6341e-08	1.6340e-08	1.8473e-10	1.8402e-10	1.8334e-10	1.8386e-10	1.8381e-10
4	1.6341e-08	1.6341e-08	1.6341e-08	1.0000	1.6340e-08	1.6340e-08	1.8413e-10	1.8344e-10	1.8392e-10	1.8384e-10
5	1.6341e-08	1.6340e-08	1.6340e-08	1.6341e-08	1.0000	1.6340e-08	1.6339e-08	1.8341e-10	1.8386e-10	1.8375e-10
6	1.8511e-10	1.8468e-10	1.8457e-10	1.6340e-08	1.6340e-08	1.0000	1.6339e-08	1.6338e-08	1.8372e-10	1.8359e-10
7	1.8461e-10	1.8416e-10	1.8403e-10	1.8431e-10	1.6339e-08	1.6339e-08	1.0000	1.6338e-08	1.6339e-08	1.8338e-10
8	1.8422e-10	1.8369e-10	1.8354e-10	1.8377e-10	1.8376e-10	1.6339e-08	1.6338e-08	1.0000	1.6339e-08	1.6338e-08
9	1.8438e-10	1.8381e-10	1.8366e-10	1.8382e-10	1.8377e-10	1.8367e-10	1.6338e-08	1.6338e-08	1.0000	1.6339e-08
10	1.8423e-10	1.8367e-10	1.8348e-10	1.8359e-10	1.8352e-10	1.8342e-10	1.8302e-10	1.6338e-08	1.6339e-08	1.0000

E

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

原图使用距离处理

Z

与使用kNN相同

	1	2	3	4	5	6	7	8	9	10
1	1.0000	1.6341e-08	1.6341e-08	1.6341e-08	1.6340e-08	1.8498e-10	1.8426e-10	1.8366e-10	1.8421e-10	1.8418e-10
2	1.6342e-08	1.0000	1.6341e-08	1.6341e-08	1.6340e-08	1.8479e-10	1.8404e-10	1.8341e-10	1.8392e-10	1.8390e-10
3	1.6341e-08	1.6341e-08	1.0000	1.6341e-08	1.6340e-08	1.8473e-10	1.8402e-10	1.8334e-10	1.8386e-10	1.8381e-10
4	1.6341e-08	1.6341e-08	1.6341e-08	1.0000	1.6340e-08	1.6340e-08	1.8413e-10	1.8344e-10	1.8392e-10	1.8384e-10
5	1.6341e-08	1.6340e-08	1.6340e-08	1.6341e-08	1.0000	1.6340e-08	1.6339e-08	1.8341e-10	1.8386e-10	1.8375e-10
6	1.8511e-10	1.8468e-10	1.8457e-10	1.6340e-08	1.6340e-08	1.0000	1.6339e-08	1.6338e-08	1.8372e-10	1.8359e-10
7	1.8461e-10	1.8416e-10	1.8403e-10	1.8431e-10	1.6339e-08	1.6339e-08	1.0000	1.6338e-08	1.6339e-08	1.8338e-10
8	1.8422e-10	1.8369e-10	1.8354e-10	1.8377e-10	1.8376e-10	1.6339e-08	1.6338e-08	1.0000	1.6339e-08	1.6338e-08
9	1.8438e-10	1.8381e-10	1.8366e-10	1.8382e-10	1.8377e-10	1.8367e-10	1.6338e-08	1.6338e-08	1.0000	1.6339e-08
10	1.8423e-10	1.8367e-10	1.8348e-10	1.8359e-10	1.8352e-10	1.8342e-10	1.8302e-10	1.6338e-08	1.6339e-08	1.0000

E

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

原图添加相同列向量kNN处理

Z

	1	2	3	4	5	6	7	8	9	10
1	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
2	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
3	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
4	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
5	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
6	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.8617e-10	1.8617e-10	1.8617e-10	1.8617e-10
7	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.8617e-10	1.0000	1.8617e-10	1.8617e-10	1.8617e-10
8	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.8617e-10	1.8617e-10	1.0000	1.8617e-10	1.8617e-10
9	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.8617e-10	1.8618e-10	1.8617e-10	1.0000	1.8617e-10
10	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.8617e-10	1.8617e-10	1.8617e-10	1.8617e-10	1.0000

E

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

原图添加相同列向量距离处理

Z

	1	2	3	4	5	6	7	8	9	10
1	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
2	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
3	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
4	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
5	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
6	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08
7	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08	1.6334e-08
8	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.6334e-08	1.6334e-08
9	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.0000	1.6334e-08
10	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.6334e-08	1.0000

E

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

试验总结

- 存在使用kNN存在Z矩阵病态的问题，这个问题可以通过增大k和使用距离两种办法解决，由于真实数据集相似元素较少，使用较大的k也可以解决问题
- 在数据没有噪声的情况下，Z基本上是一个单位矩阵，除了对角线之外，基本都是接近于零的数（我认为这是模型Y=Y*Z+E导致的必然结果）
- 在数据存在噪声的情况下，噪声E不为零，但是也受k的影响，而且只有噪声符合某种分布规律的时候才比较好