While trying to figure out the role of Laplacian matrix and its potential benefit of including it in the optimization goal, I realized that the reason I feel lost and disorganized at this stage is not because of the optimization goal but because of spectral clustering. So far I have been testing baseline spectral clustering, straight-up kmeans, and t-sne for each algorithm's output similarity matrices, and the frustrating part is that for each optimization goal, different clustering methods show the best performance. (For example, nuclear norm generally performs better with t-SNE, but when rank is controlled through hard-thresholding without convex relaxation, baseline spectral clustering works better.) Moreover, the performance is better in one method for some data sets but worse for other data sets. This is why performance comparison often feels inconclusive - even with the same similarity matrix, performance is different (not by much in terms of evaluation measure like NMI, but in terms of ordering - which performs better?).

I organized four methods below. However, there are more than just 4: within each method, I can use (a) binary adjacency matrix or continuous weighted edge matrix (if using binary adjacency matrix, knn or thresholding? how many k for knn? which threshold for thresholding?), (b) normalized Laplacian or unnormalized Laplacian, and (c) kmeans on entire matrix or kmeans on the matrix of only k eigenvectors. This exponentially increases the number of tests I need to conduct, and it has been difficult to make a consistent conclusion of which is better than which.

My current thought is to decrease the number of tests by trying only the conventionally better method. For example, we can only use normalized Laplacian and not unnormalized one, and use normalized spectral clustering instead of straight-up kmeans, even if this decision leads to poor performance for some optimization goals and for some data sets.

And of course on top of all this, we need to decide the kernel parameters. There are too many things we can play with, and I'm overwhelmed.

Kmeans

- start with similarity matirx S
- kmeans clustering on S, or on the first k eigenvalues of S

Baseline Spectral Clustering

- start with similarity matrix S
- construct adjacency matrix A with binary 0/1 elements using hard thresholding or knn
- Get Laplacian matrix L, either normalized $L = (D^{-1/2}(D-A)D^{-1/2})$ or unnormalized $L = (D-A)$.
- Get the matrix of eigenvectors U of L corresponding to the k smallest eigenvalues.
- kmeans clustering on U

Normalized spectral clustering by Ng et al (2002)

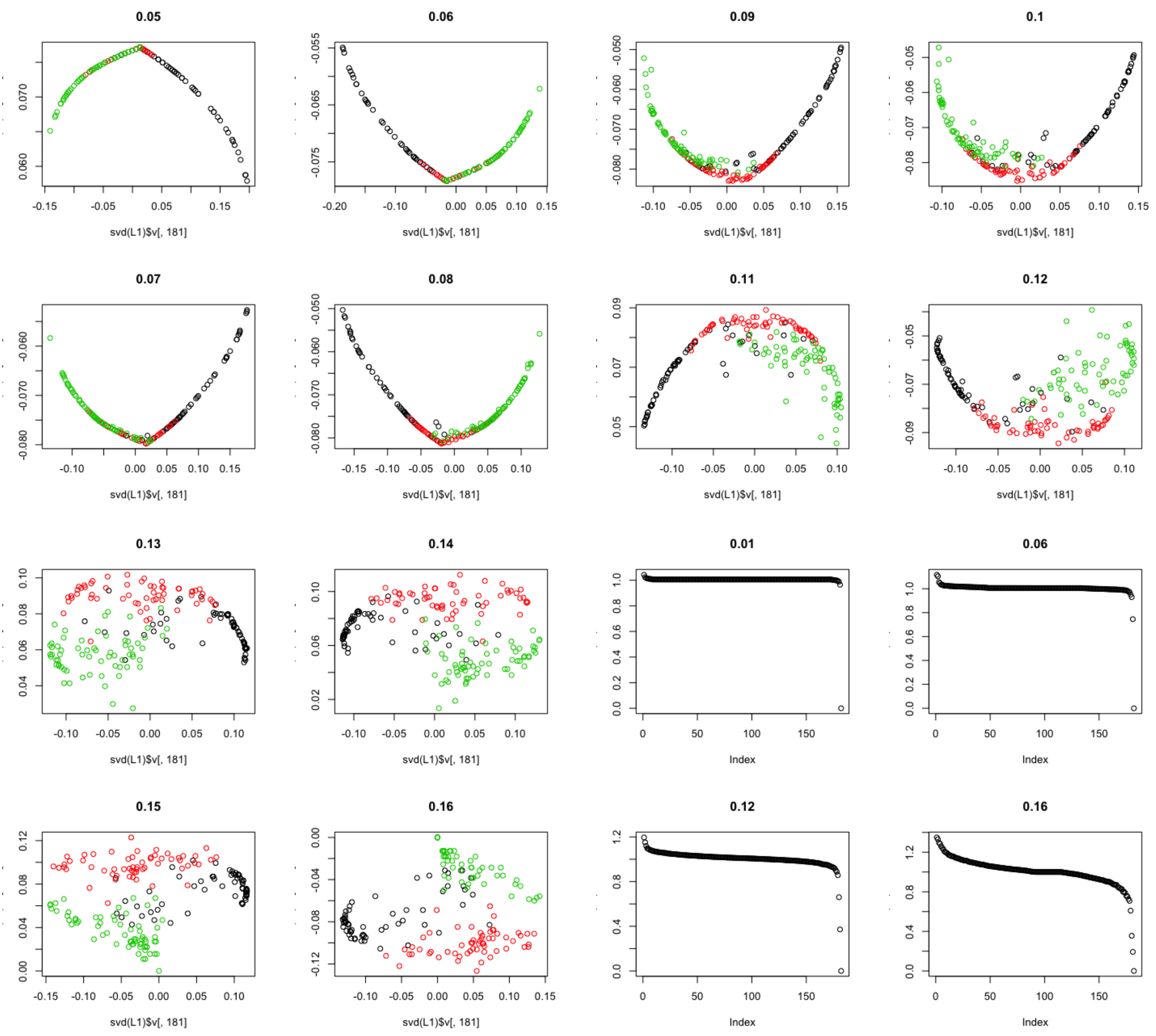
- start with similarity matrix S
- construct a similarity graph using weighted (or unweighted / binary) adjacency matrix
- construct normalized Laplacian matrix L

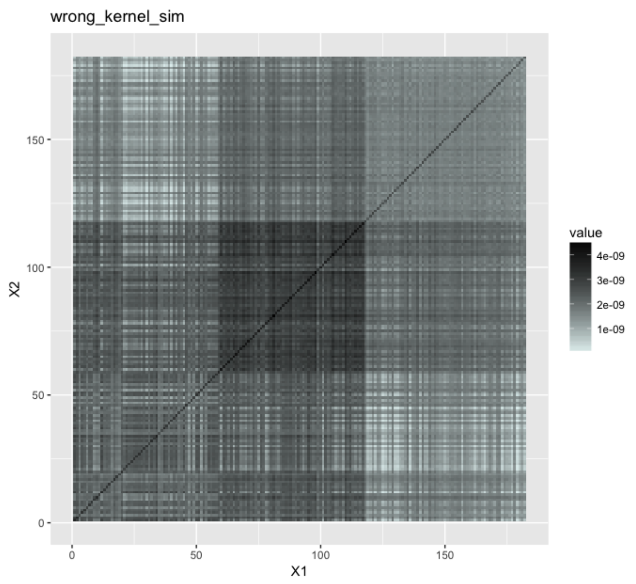
- compute the first k eigenvectors of L
- construct matrix T by normalizing the rows to norm 1
- kmeans clustering on T

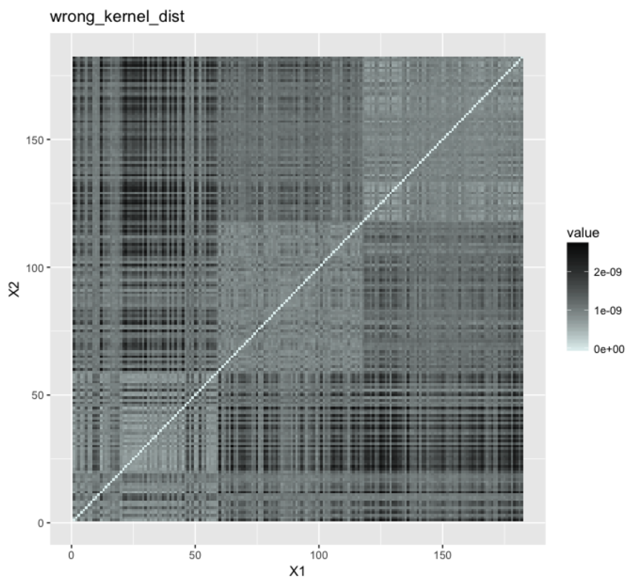
t-SNE (2008)

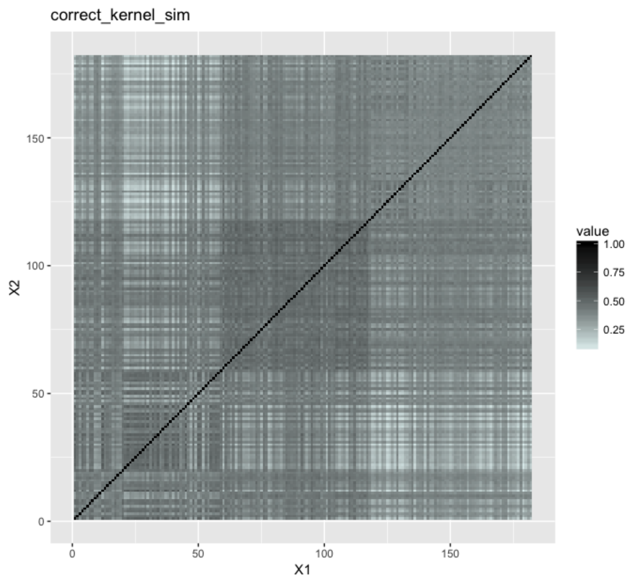
- start with similarity matrix S
- compute Laplacian L from similarity matrix S
- use initial solution $\mathcal{Y}^{(0)}$ as the last k eigenvectors of the Laplacian matrix L
- for $t = 1$ to T, compute low dimensional affinities q_{ij}, compute gradient, and update $\mathcal{Y}_t$

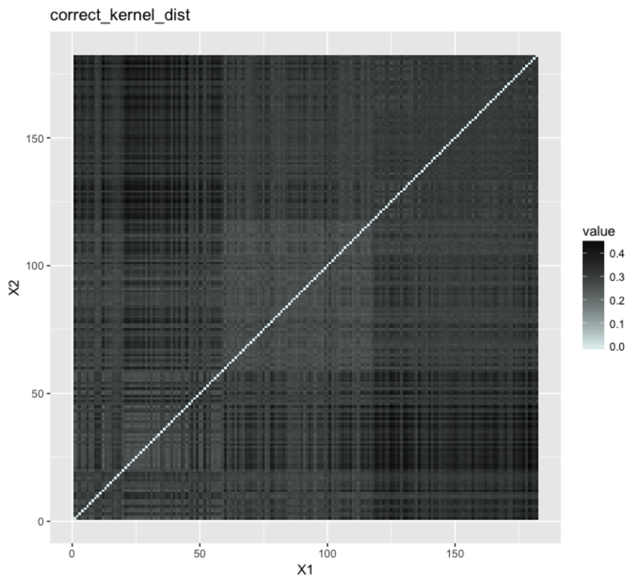
$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \mathcal{Y}} + \alpha(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$$


0.05
0.06
0.09
0.1
0.07
0.08
0.11
0.12
0.13
0.14
0.01
0.06
0.15
0.16
0.12
0.16
svd(L1)$v[, 181]
Index


wrong_kernel_sim
X2
X1
value
4e-09
3e-09
2e-09
1e-09
0
50
100
150


wrong_kernel_dist
X2
X1
value
2e-09
1e-09
0e+00
0
50
100
150


correct_kernel_sim
X2
X1
value
1.00
0.75
0.50
0.25
0
50
100
150


correct_kernel_dist
X2
X1
value
0.4
0.3
0.2
0.1
0.0
0
50
100
150