パターン認識
Pattern Recognition

佐藤真一
Shin'ichi Satoh

国立情報学研究所
National Institute of Informatics

July 4, 2023

Final Report

- Find any pattern recognition paper in top journals or top conferences
- e.g., IEEE TPAMI, IJCV, CVPR, ICCV, NeurIPS, ICML, ACMMM...
- Describe the following:
  - bibliographic info of the paper
  - brief of the paper
  - what is the problem, why it's important, how it's solved, validation?
  - why you selected the paper, what is exciting?
  - feedback to the lecture, any comments
- (about) 2-4 pages A4
- due: 07/31/2023
- send via ITC-LMS

Final Report

- Geman and Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, TPAMI, 1984.
- Moghaddam and Pentland, Probabilistic Visual Learning for Object Detection, TPAMI, 1997.
- Belhumeur, Hespanha, and Kriegman, Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, TPAMI, 1997.
- Shi and Malik, Normalized Cuts and Image Segmentation, TPAMI, 2000.
- Belkin and Niyogi, Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering, NIPS, 2001.
- Zhang and Sim, When Fisher meets Fukunaga-Koontz: A New Look at Linear Discriminants, CVPR, 2006.
- Felzenszwalb, Girshick, McAllester, and Ramanan, Object Detection with Discriminatively Trained Part Based Models, TPAMI, 2009.
- Antonio Torralba and Alexei A. Efros, Unbiased look at dataset bias, CVPR 2011.

Clustering Experiments

Four types of dataset:

- Two Gaussians (two flattened Gaussians)
- Four squares
- Four Gaussians
- Swiss role

Three clustering algorithms:

- k-means
- Single linkage
- Complete linkage.

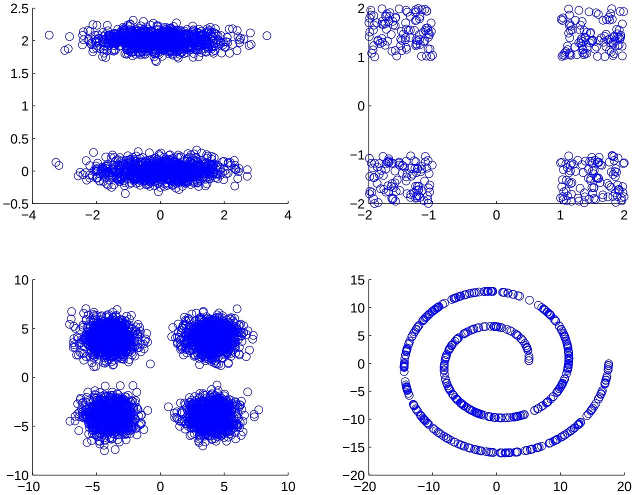
Clustering methods (recap)

k-means
: Select k representatives randomly, assign each data to its closest representative, and iteratively update representatives by their means of assigned data.

Single linkage
: Hierarchical aggromerative clustering method with cluster distance defined as minimum distance among items.

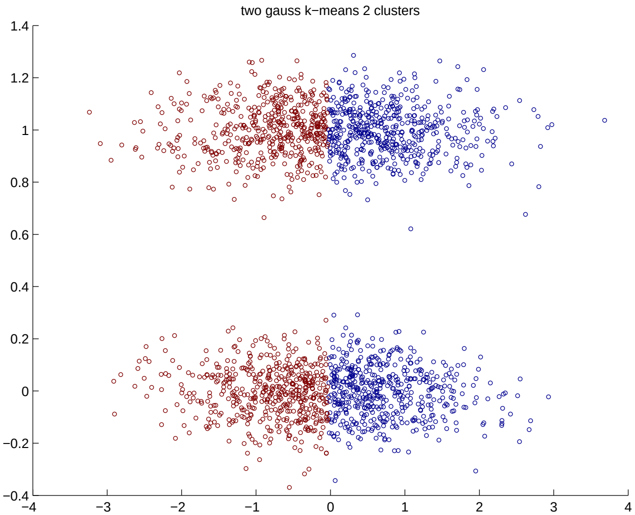
Complete linkage
: Hierarchical aggromerative clustering method with cluster distance defined as maximum distance among items.

Data

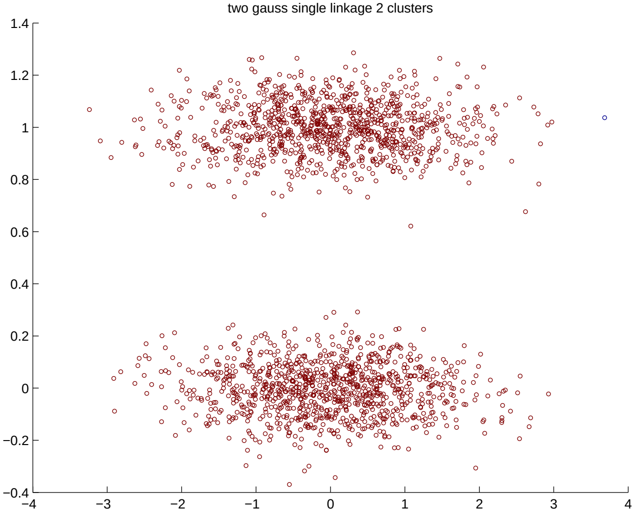


2.5
2
1.5
1
0.5
0
−0.5
−4
−2
0
2
4
2
1
0
−1
−2
−2
−1
0
1
2
10
5
0
−5
−10
−10
−5
0
5
10
15
10
5
0
−5
−10
−15
−20
−20
−10
0
10
20

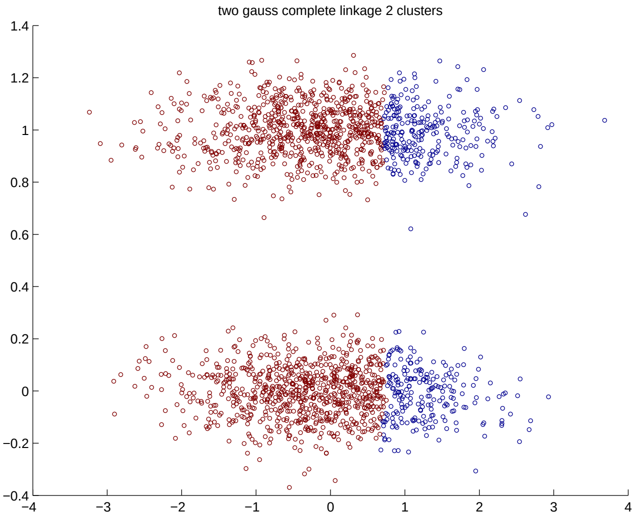
Two Gaussians



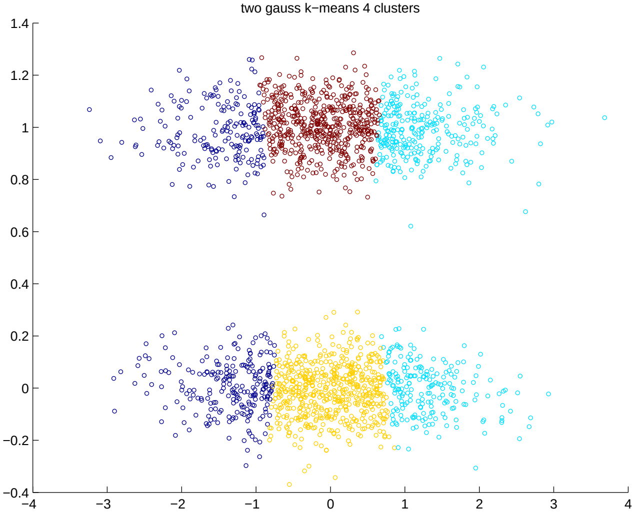
Two Gaussians



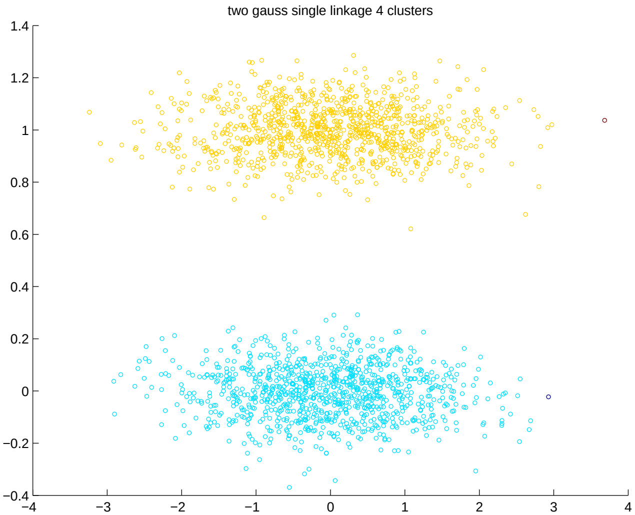
Two Gaussians



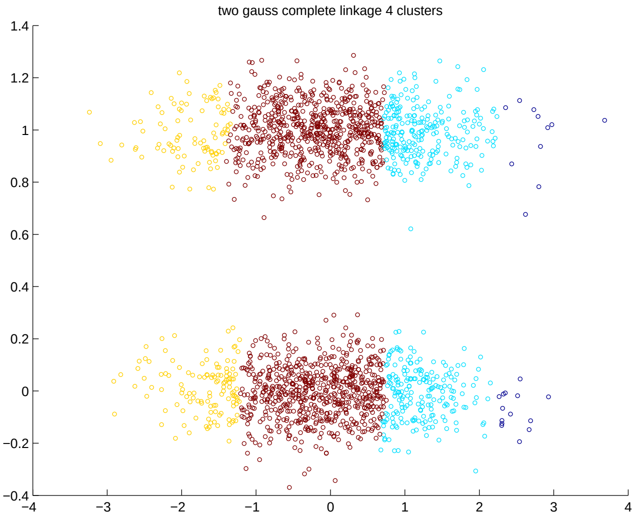
Two Gaussians



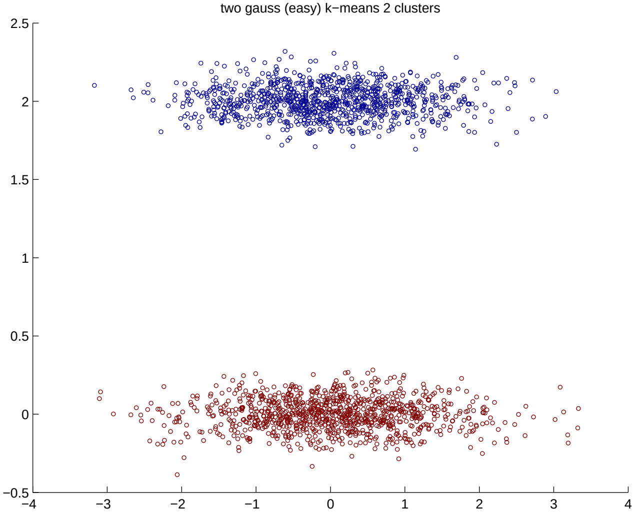
Two Gaussians



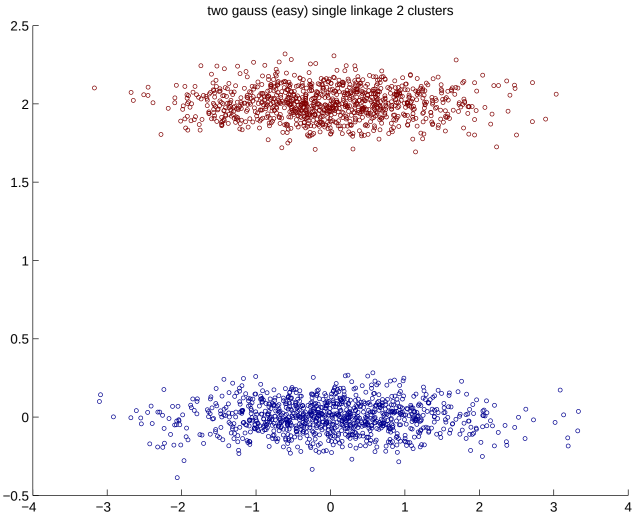
Two Gaussians



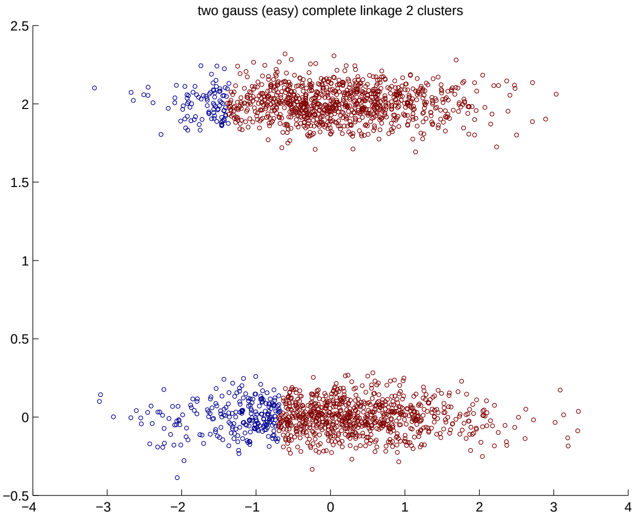
Two Gaussians



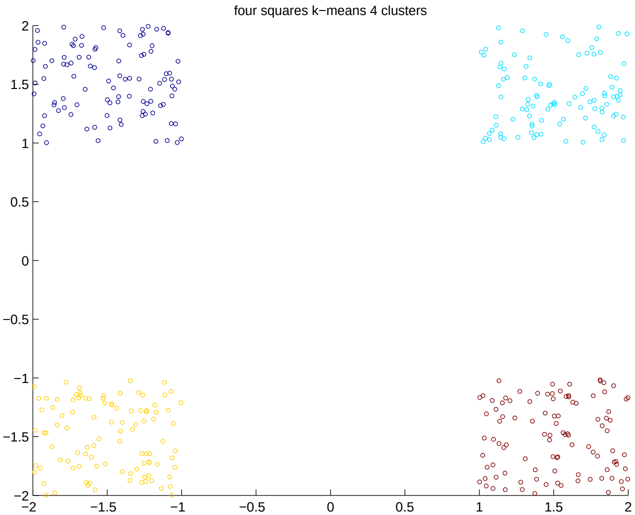
Two Gaussians



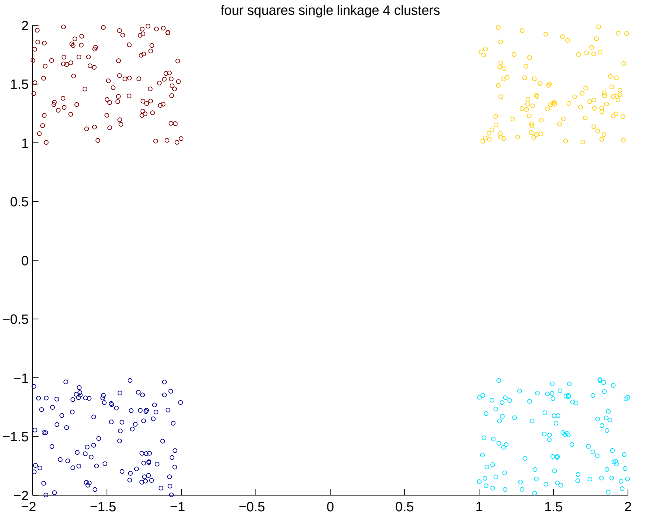
Two Gaussians



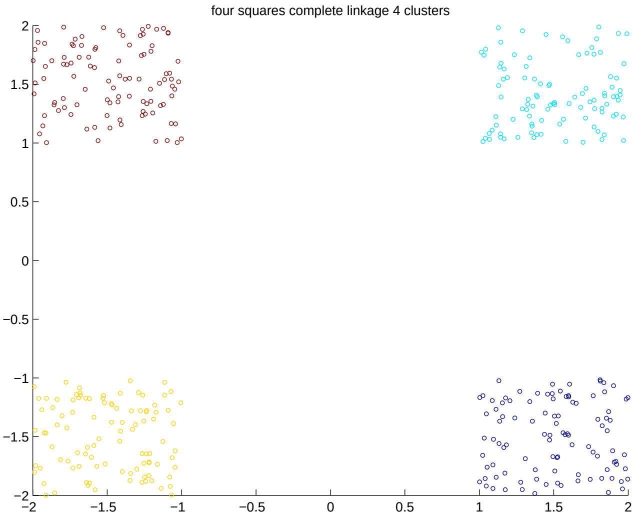
Four Squares



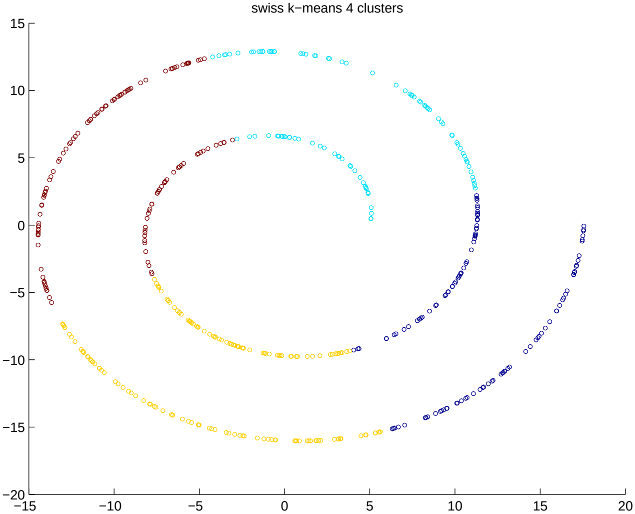
Four Squares



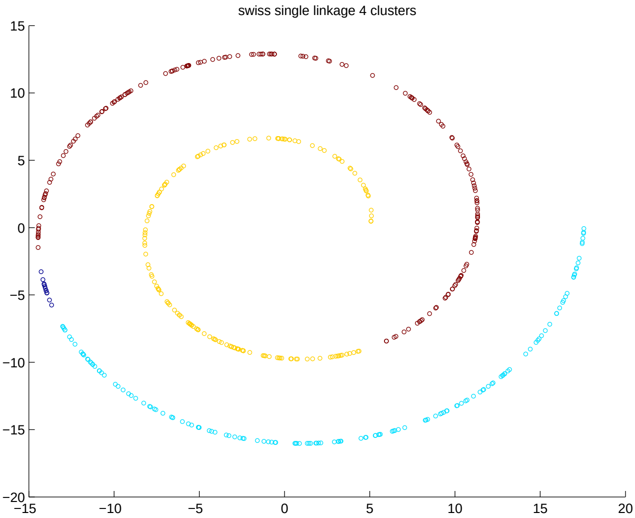
Four Squares



Swiss Role

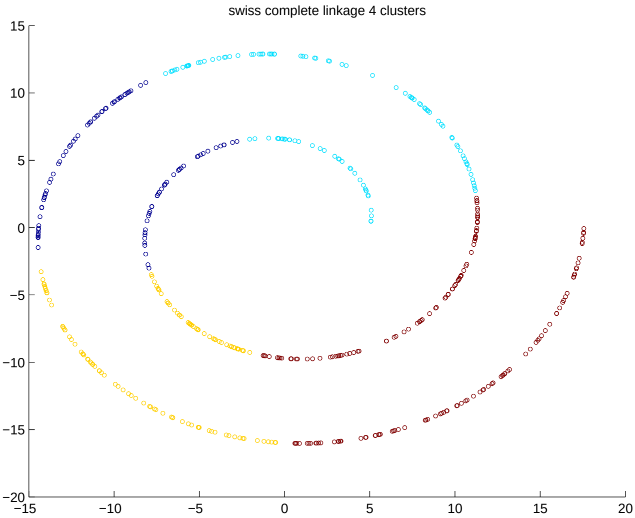


Swiss Role



swiss single linkage 4 clusters
15
10
5
0
−5
−10
−15
−20
−15
−10
−5
0
5
10
15
20

Swiss Role

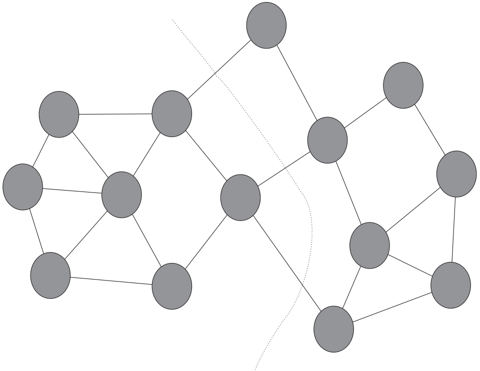

swiss complete linkage 4 clusters
15
10
5
0
−5
−10
−15
−20
−15
−10
−5
0
5
10
15
20

Graph Spectral Clustering

- So far we assume that data to be clustered have corresponding vectors, and distances can be evaluated as the Euclidean distances.
- But what if only distances (or similarities) are given?
- What if "distances" are effective in local region only?
- Graph-based clustering techniques may work in such situations.
- We represent data along with similarities as graph, and solve clustering as graph partitioning problem.

Graph Spectral Clustering

- Each data: node.
- Similarity between two nodes (if any): edge.
- Edge weight: similarity.
- Cut: sum of weights of edges which are cut.



Graph Spectral Clustering

Represent n-node graph as $n \times n$ matrix W where $w_{i,j}$ represents similarity between i-th and j-th nodes.
First we consider simple case to partition data into two sets.
We want to determine membership indicator q_i:

$$q_i = \begin{cases} 1 & \text{if } i \in A \\ -1 & \text{if } i \in B \end{cases}$$

$$\begin{aligned} J =& CutSize = \frac{1}{4} \sum_{i,j} w_{i,j} (q_i - q_j)^2 = \frac{1}{4} \sum_{i,j} w_{i,j} (q_i^2 + q_j^2 - 2 q_i q_j) \\ =& \frac{1}{2} \sum_{i,j} q_i (d_i \delta_{i,j} - w_{i,j}) q_j \\ =& \frac{1}{2} q^T (D - W) q \end{aligned}$$

$$d_i = \sum_j w_{i,j}$$

Minimization of $J = \frac{1}{2} q^T (D - W) q$ is hard (NP-complete).

So we relax q from discrete values to continuous values: then the solution of $\min J(q)$ can be obtained by the eigenvectors of

$$(D - W)q = \lambda q$$

Properties of Graph Laplacian

$L = D - W$ is called Laplacian matrix of the graph.
L is semi-positive definite: $x^T L x \geq 0$ for any x.
First eigenvector is $q_1 = [1\,1\,\cdots\,1]^T = e^T$ with $\lambda_1 = 0$.
Second eigenvector q_2 is the desired solution (called Fiedler vector).

$$\lambda_2 = \frac{cutsize}{|A|} + \frac{cutsize}{|B|}$$

Since J is insensitive to additive constant to q

$$J = \frac{1}{4} \sum_{i,j} w_{i,j}((q_i + c) - (q_j + c))^2$$

we sort q_2 and cut in the middle point.

Clustering Objective Functions

- Ratio Cut

$$J_{Rcut} = \frac{s(A,B)}{|A|} + \frac{s(A,B)}{|B|}$$

- Normalized Cut

$$\begin{aligned} J_{Ncut} &= \frac{s(A,B)}{d_A} + \frac{s(A,B)}{d_B} \\ &= \frac{s(A,B)}{s(A,A)+s(A,B)} + \frac{s(A,B)}{s(B,B)+s(A,B)} \end{aligned}$$

- Min-Max Cut

$$J_{MMcut} = \frac{s(A,B)}{s(A,A)} + \frac{s(A,B)}{s(B,B)}$$

$$s(A,B) = \sum_{i \in A} \sum_{j \in B} w_{i,j}, \;\; d(A) = \sum_{i \in A} d_i$$

Laplacian Eigenmaps

We embed low dimensional vector representation to each node.
To do so, we solve the following generalized eigenvalue problem:

$$Lf = \lambda Df.$$

(FYI the same solution with Normalized Cut.)
For $f_0, f_1, \ldots, f_n$ corresponding to $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_n$,
we correspond $[f_1(i)\, f_2(i)\, \ldots\, f_m(i)]^T$ to the i-th node.

Experiment

- The weights of edges are defined as:

$$w_{i,j} = \begin{cases} e^{-\frac{d_{i,j}^2}{\sigma^2}} & \text{if } i \in knn(j) \text{ or } j \in knn(i) \\ 0 & otherwise \end{cases}$$

- Convert i-th node to $[f_1(i)\, f_2(i)\, \ldots\, f_m(i)]^T$ using Laplacian eigenmaps.
- Apply k-means to the converted vectors.
- Constants: k of knn: 5, σ: 10, m: 2.

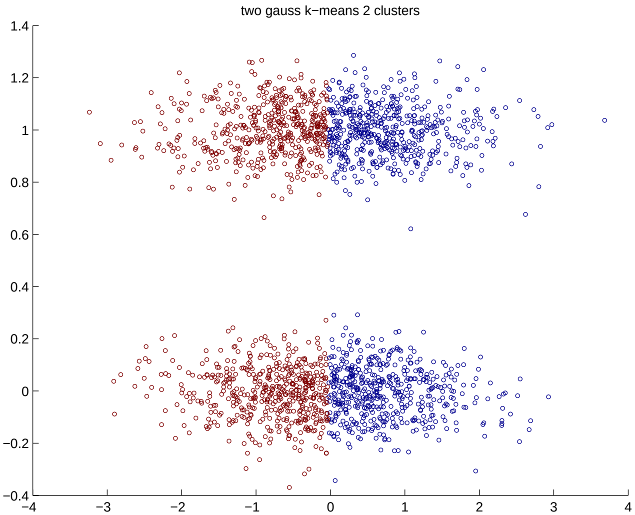
Laplacian Eigenmaps (Matlab)

```
k=10;
sigma=10;
n=size(x,1);
dt=squareform(pdist(x));
[sdt,idx]=sort(dt,'ascend');
dt=sdt(1:k+1,:);
nidx=idx(1:k+1,:);
tW=exp(-dt.^2/(sigma^2));
ii=repmat(1:n,k+1,1);
W=sparse(ii(:),double(nidx(:)),tW(:),n,n);
W=full(W);
W=max(W,W');
D=diag(sum(W,2));
[v,d]=eigs(D-W,D,10,'sa');
xx=v(:,2:3);
c=kmeans(xx,numc);
```

Laplacian Eigenmaps (Python)

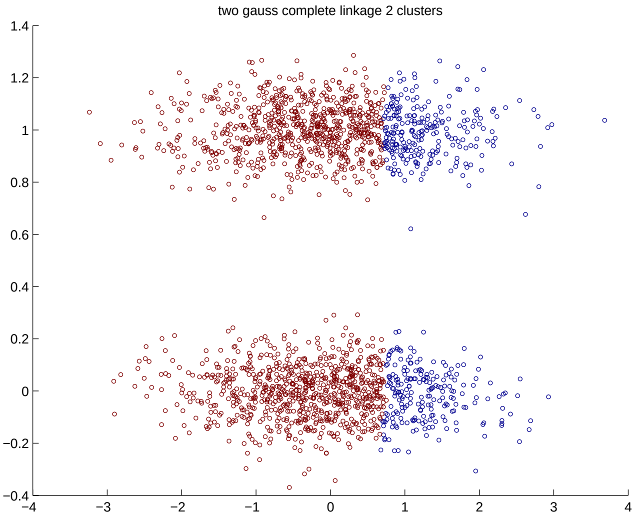
```
k = 10
sigma = 5.
n = x.shape[0]
dt = squareform(pdist(x))
idx = np.argsort(dt)
dt = np.array([dt[i, idx[i, range(k)]] for i in range(n)])
W = np.zeros([n, n])
for i in range(n):
    for j in range(k):
        W[i, idx[i, j]] = np.exp(-(dt[i, j]**2) / (sigma**2))
W = np.maximum(W, W.T)
D = np.diag(np.sum(W, axis=0))
dd, v = scipy.sparse.linalg.eigs(D - W, k=5, M=D, which='SR')
v = np.real(v)
idxdd = np.argsort(np.real(dd))
xx = v[:, idxdd][:, range(1, 4)]
c = sklearn.cluster.KMeans(n_clusters=numc).fit_predict(xx)
```

Two Gaussians

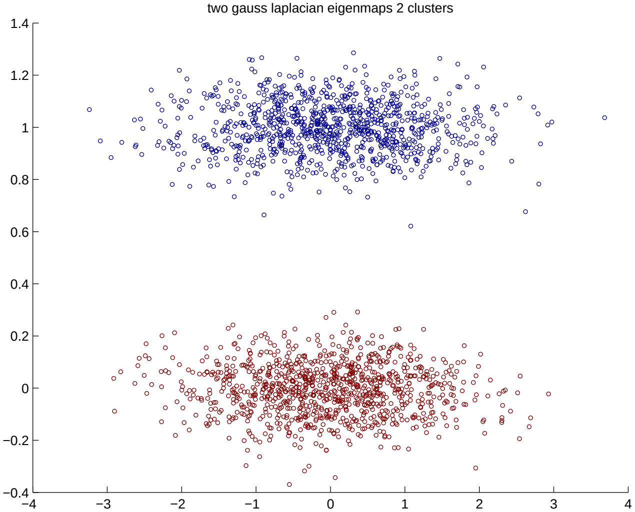


Two Gaussians

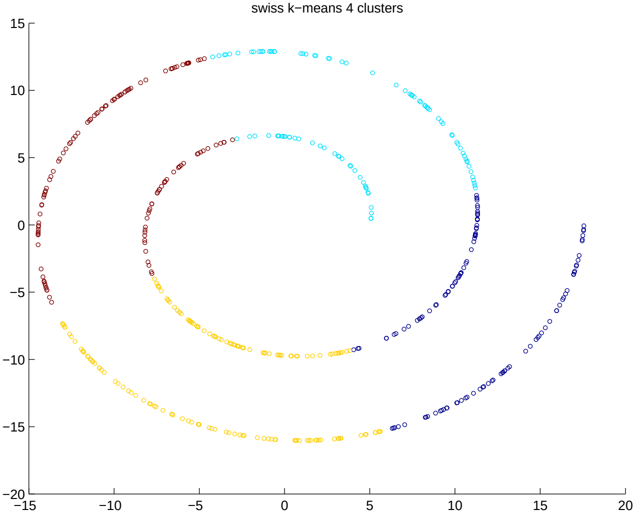
Two Gaussians



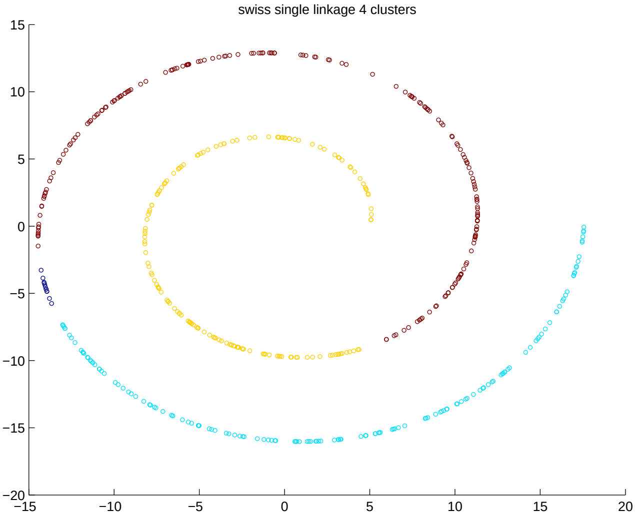
Two Gaussians



Swiss Role

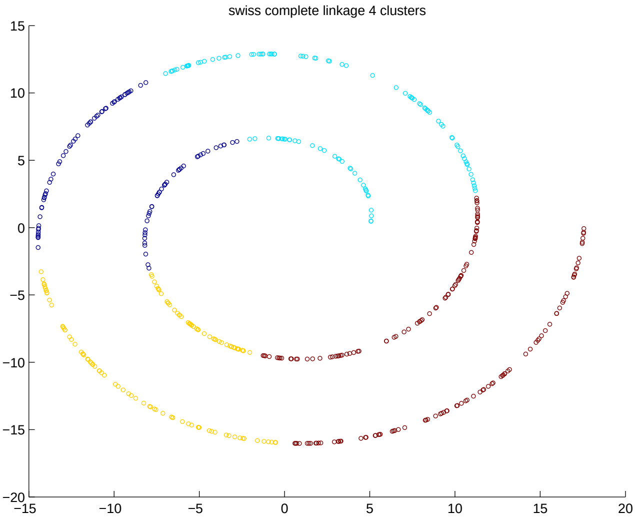


Swiss Role

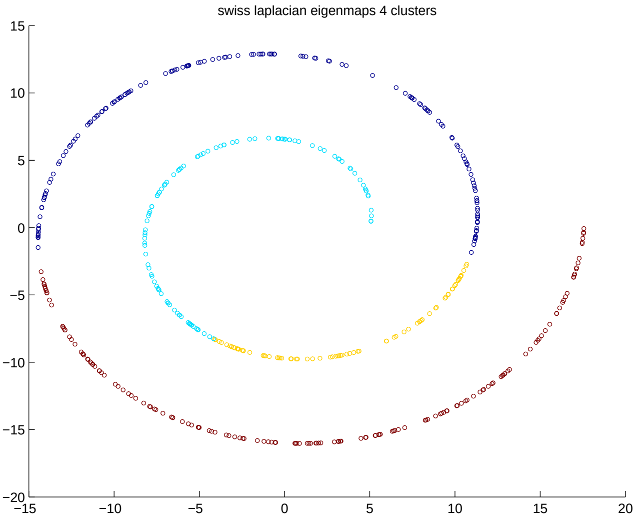


swiss single linkage 4 clusters
15
10
5
0
−5
−10
−15
−20
−15
−10
−5
0
5
10
15
20

Swiss Role

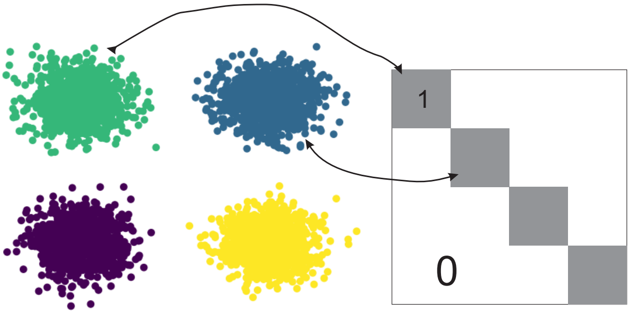

swiss complete linkage 4 clusters
15
10
5
0
−5
−10
−15
−20
−15
−10
−5
0
5
10
15
20

Swiss Role

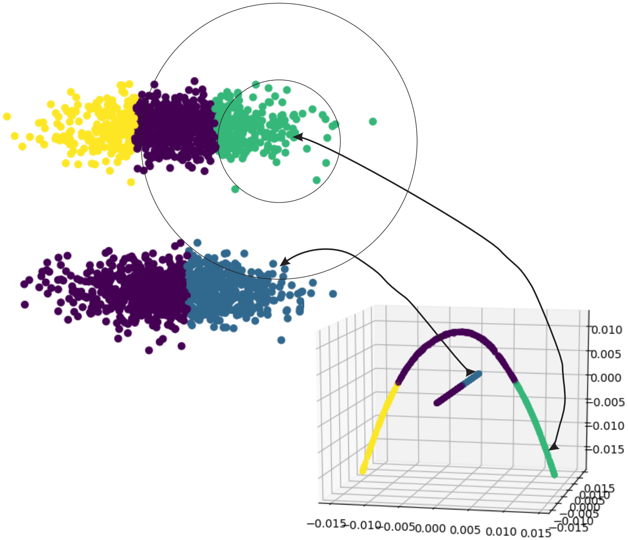


swiss laplacian eigenmaps 4 clusters
15
10
5
0
−5
−10
−15
−20
−15
−10
−5
0
5
10
15
20

Example



Example


0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.015 −0.010 −0.005 0.000 0.005 0.010 0.015