Clustering

Victor Kitov

Table of Contents

K-means algorithm

- Suppose we want to cluster our data into g clusters.
- Cluster i has a center μ_i, i=1,2,...g.
- Consider the task of minimizing

$$\sum_{n=1}^{N}\rho(x_n,\mu_{z_n})^2\to\min_{z_1,...z_N} \tag{1}$$

where $z_i\in\{1,2,...g\}$ is cluster assignment for x_i.
- Direct optimization requires full search and is impractical.
- K-means is a suboptimal algorithm for optimizing (1).

K-means algorithm

```
Initialize μ_j, j = 1,2,...g.

repeat while stop condition not satisfied:
    for i = 1,2,...N:
        find cluster number of x_i:
        z_i = arg min_{j∈{1,2,...g}} ||x_i − μ_j||
    for j = 1,2,...g:
        μ_j = (1 / Σ_{n=1}^N I[z_n=j]) Σ_{n=1}^N I[z_n = j] x_i
```

Possible stop conditions:

- cluster assignments $z_1, ...z_N$ stop to change (typical)
- maximum number of iterations reached
- cluster means $\{\mu_i,\ i = 1, 2, ...g\}$ stop changing significantly

Dynamic K-means algorithm

```
Initialize μ_j, j = 1,2,...g, z_i = 0, i = 1,2,...N

repeat while stop condition not satisfied:
   for i = 1,2,...N:
      find cluster number of x_i:
      z_i' = arg min_{j∈{1,2,...g}} ||x_i − μ_j||
      if z_i' != z_i:
         recalculate cluster means μ_{z_i} and μ_{z_i'}:
         μ_{z_i} = 1/(Σ_{n=1}^N I[z_n' = z_i]) Σ_{n=1}^N I[z_n' = z_i] x_i
         μ_{z_i'} = 1/(Σ_{n=1}^N I[z_n' = z_i']) Σ_{n=1}^N I[z_n' = z_i'] x_i
         z_i = z_i'
```

Initialize μ_j, $j = 1, 2, ...g$, $z_i = 0, i = 1, 2, ...N$

repeat while stop condition not satisfied:

for $i = 1, 2, ...N$:

find cluster number of x_i:

$z_i' = \arg\min_{j \in \{1,2,...g\}} ||x_i - \mu_j||$

if $z_i' != z_i$:

recalculate cluster means μ_{z_i} and $\mu_{z_i'}$:

$$\mu_{z_i} = \frac{1}{\sum_{n=1}^N \mathbb{I}[z_n' = z_i]} \sum_{n=1}^N \mathbb{I}[z_n' = z_i] x_i$$

$$\mu_{z_i'} = \frac{1}{\sum_{n=1}^N \mathbb{I}[z_n' = z_i']} \sum_{n=1}^N \mathbb{I}[z_n' = z_i'] x_i$$

$z_i = z_i'$

Converges in less iterations, situation when no objects correspond to some cluster is impossible.

Initialization of cluster centers

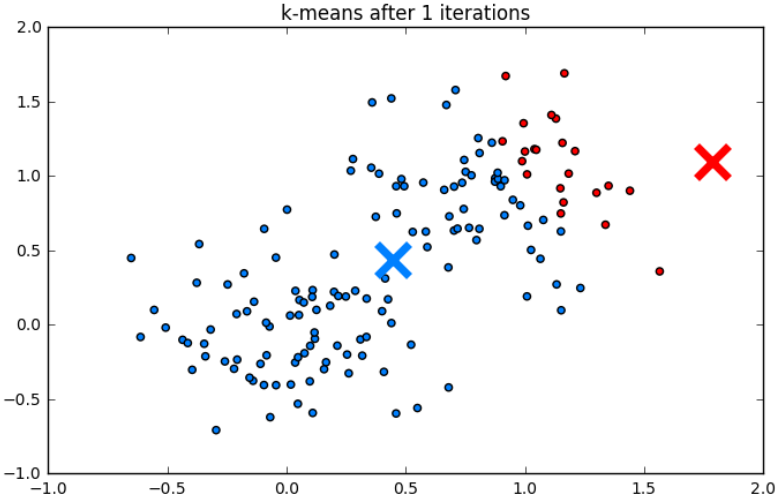
1. We can initialize $\{\mu_i,\ i = 1, 2, ...g\}$ with g randomly chosen measurements without replacement (typical)
2. Alternatively we can initialize $\{\mu_i,\ i = 1, 2, ...g\}$ with most distant set of points:

```
Estimate μ = (1/N) Σ_{i=1}^{N} x_i.
set μ_1 = argmax_{x∈x_1,...x_N} ρ(μ, x)
set μ_2 = argmax_{x∈{x_1,...x_N}} {ρ(μ_1, x)}
set μ_3 = argmax_{x∈{x_1,...x_N}} {ρ(μ_1, x) + ρ(μ_2, x)}
......................
set μ_g = argmax_{x∈{x_1,...x_N}} {Σ_{i=1}^{g-1} ρ(μ_i, x)}
```

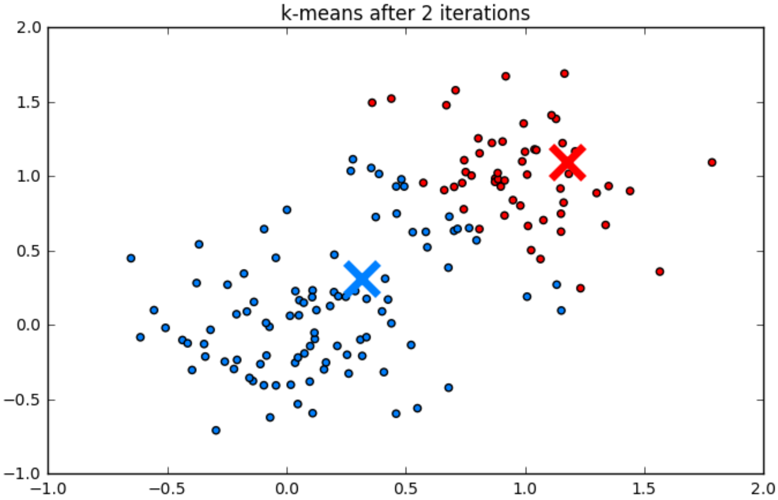
K-means properties

- Only local optimum is found
- Results depends on initialization
  - It is common to run algorithm multiple times with different initializations and then select the result minimizing criterion in (1).
- *Complexity:* $O(NDgI)$*, where* g *is the number of clusters and* I *is the number of iterations. Why?*
  - If clusters exist, algorithm converges with few iterations and complexity is $O(NDg)$

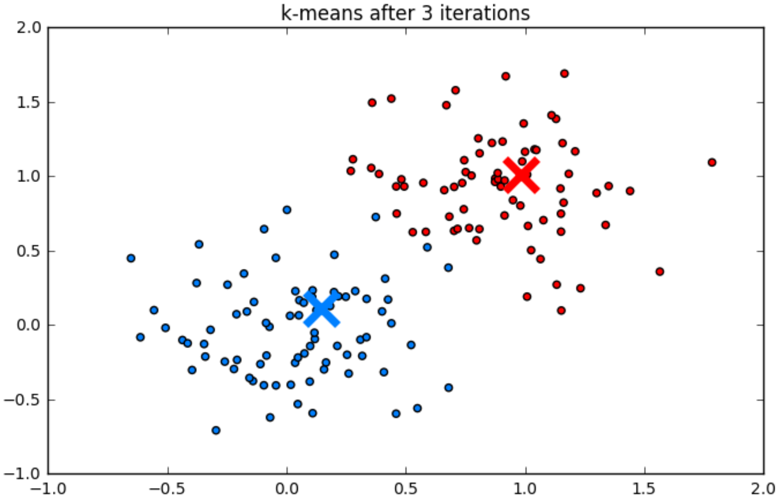
Example of K-means



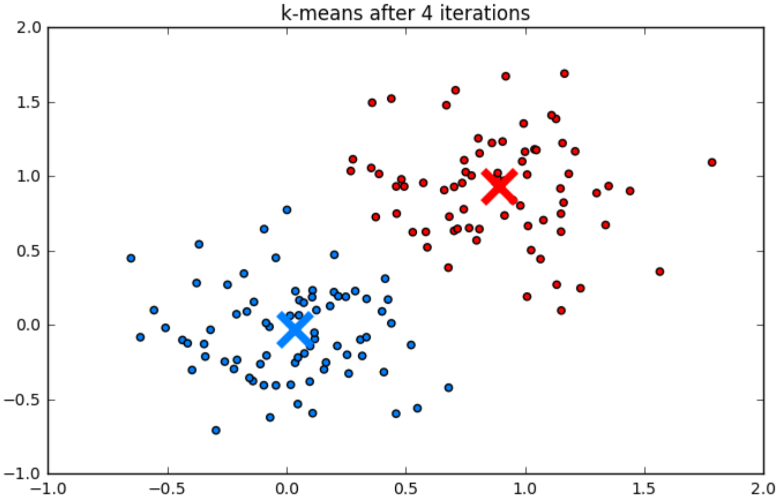
Example of K-means



Example of K-means

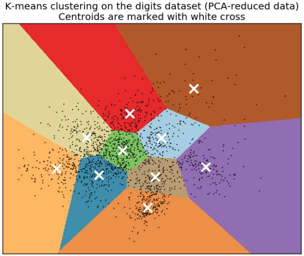


Example of K-means



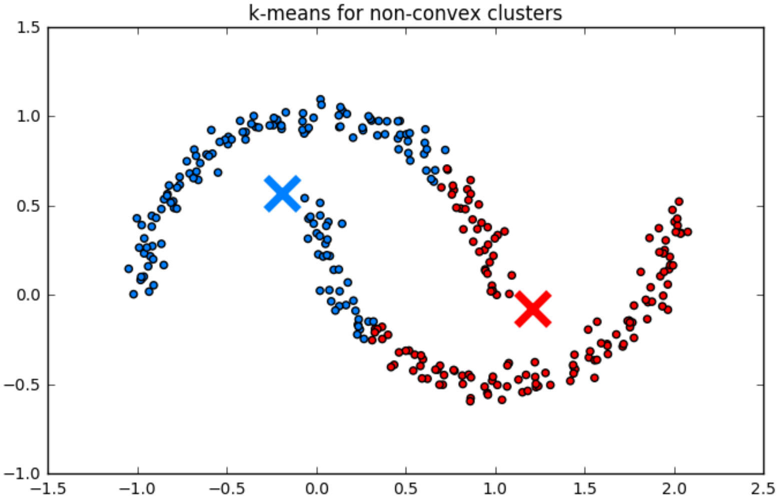
Gotchas

- K-means assumes that clusters are convex:

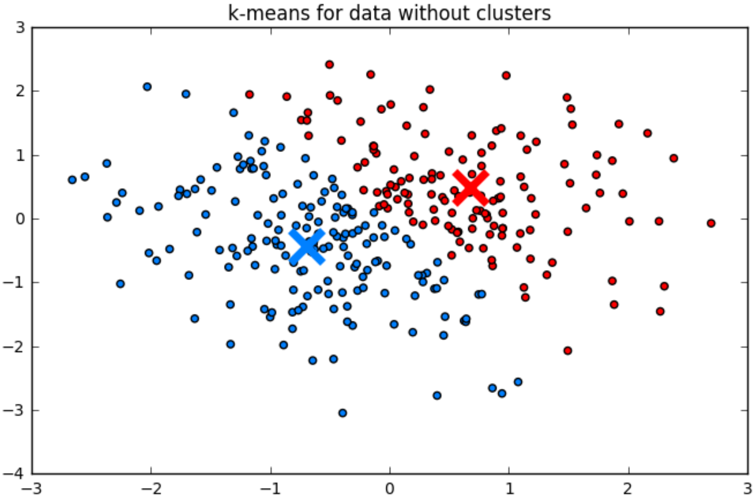


- It always finds clusters even if none actually exist
  - need to control cluster quality metrics

K-means for non-convex clusters



K-means for data without clusters



K-means and EM algorithm

```
Initialize μ_j, j = 1, 2, ...g.

repeat while stop condition not satisfied:
    for i = 1, 2, ...N:
        find cluster number of x_i:
        z_i = argmin_{j∈{1,2,...g}} ||x_i − μ_j||
    for j = 1, 2, ...g:
        μ_j = (1 / Σ_{n=1}^N 𝕀[z_n = j]) Σ_{n=1}^N 𝕀[z_n = j] x_i
```

- K-means is EM-algorithm when:

K-means and EM algorithm

```
Initialize μ_j, j = 1, 2, ...g.

repeat while stop condition not satisfied:
    for i = 1, 2, ...N:
        find cluster number of x_i:
        z_i = arg min_{j∈{1,2,...g}} ||x_i − μ_j||
    for j = 1, 2, ...g:
        μ_j = (1 / Σ_{n=1}^N I[z_n = j]) Σ_{n=1}^N I[z_n = j] x_i
```

- K-means is EM-algorithm when:
  - applied to Gaussians
  - with equal priors
  - with unity covariance matrices
  - with hard clustering

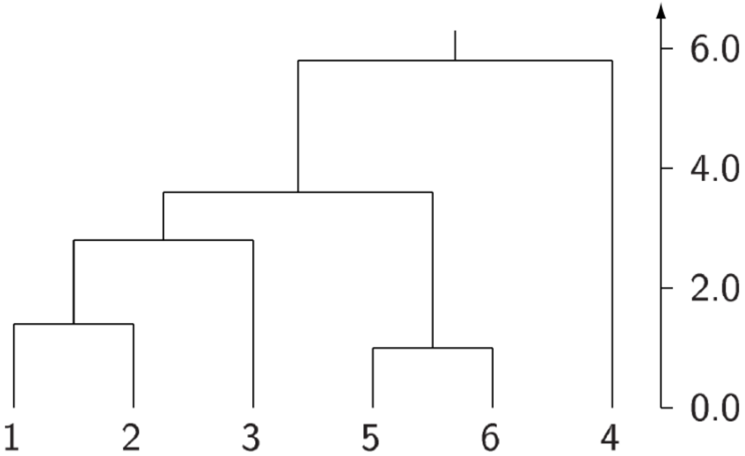
Table of Contents

Hierarchical clustering

Hierarchical clustering may be:

- top-down
  - hierarchical K-means
- bottom-up
  - agglomerative clustering

Agglomerative clustering



Agglomerative clustering - distances

- Consider clusters $A = \{x_{i_1}, x_{i_2}, ...\}$ and $B = \{x_{j_1}, x_{j_2}, ...\}$.
- We can define the following natural distances
  - nearest neighbour (or single link)
  $$\rho(A, B) = \min_{a \in A, b \in B} \rho(a, b)$$
  - furthest neighbour (or complete-link)
  $$\rho(A, B) = \max_{a \in A, b \in B} \rho(a, b)$$
  - group average link
  $$\rho(A, B) = \text{mean}_{a \in A, b \in B} \rho(a, b)$$
  - centroid distance ($\mu_U = \frac{1}{|U|} \sum_{x \in U} x$)
  $$\rho(A, B) = \rho(\mu_A, \mu_B)$$
  - median distance ($m_U = median_{x \in U}\{x\}$)
  $$\rho(A, B) = \rho(m_a, m_b)$$

Agglomerative clustering - distance properties

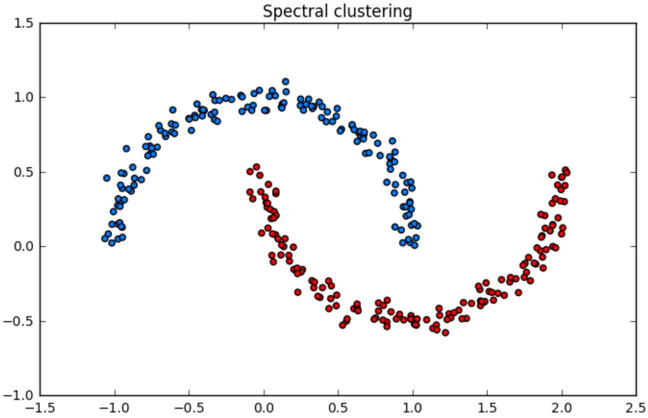
- *Suppose we modify distance* $\rho(x, x')$ *with monotone transformation* F*:* $\rho'(x, x') = F(\rho(x, x'))$*. Which of the cluster distances will not be affected by this change?*
- Lance-Williams recurrence formula:
    - $\rho(A \cup B, C)$ can be computed in $O(1)$ time using $\rho(A, C)$, $\rho(B, C)$ and $\rho(A, B)$

Agglomerative clustering - distance properties

- nearest neighbour may create stretched clusters
- furtherst neighbour creates very compact clusters.
- group average link, centroid and median distance give the compromise.
- however centroid and median distance may lead to non-monotonous joining distance sequences in agglomerative algorithm.
- in short - group average link is preferred.

Table of Contents

Spectral clustering - example



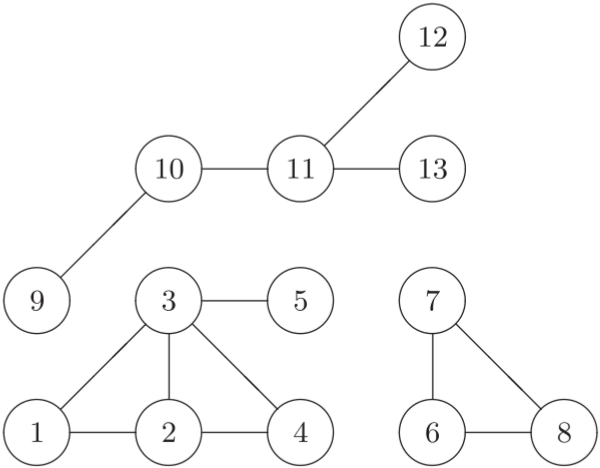
Description

- Spectral clustering reiles upon similarity matrix W between objects.
- Similarity matrix <-> weighted connection graph
- Examples:
  - nodes represent people, edge weights - how much they communicate
  - nodes represent web-pages, edge weights - scalar products of $TF-IDF$

Similarity matrix calculation

- $\|x_i - x_j\| < \textit{threshold}$
- RBF
- based on nearest neighbours

Graph with disjoint components



Graph Laplacian

- $W = W^T$, $w_{ij} \geq 0$ - the similarity between object i and object j.
- Define $D = \text{diag}\{d_1, ...d_N\}$, where $d_i = \sum_{j=1}^{N} w_{ij}$-weighted degree of node i.
- Define graph Laplacian

$$L = D - W$$

- Properties of graph Laplacian:
  - it is symmetric
  - *It has eigenvector* $\mathbf{1} \in \mathbb{R}^N$ *consisting of ones with eigenvalue 0. Why?*
  - it is positive semi-definite: $\forall f \in \mathbb{R}^N : f^T L f \geq 0$.
  - L has eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N = 0$

Positive semi-definiteness of Laplacian

Consider arbitrary $f \in \mathbb{R}^N$:

$$
\begin{aligned}
f^T L f = f^T D f - f^T W f = \sum_i d_i f_i^2 - \sum_{i,j,} f_i f_j w_{ij} = \\
\frac{1}{2}\left(\sum_i d_i f_i^2 - 2\sum_{i,j} w_{ij} f_i f_j + \sum_j d_j f_j^2\right) = \\
\frac{1}{2}\left(\sum_{i,j} w_{ij} f_i^2 - 2\sum_{i,j} w_{ij} f_i f_j + \sum_{j,i} w_{ji} f_j^2\right) = \\
\frac{1}{2}\left(\sum_{i,j} w_{ij} f_i^2 - 2\sum_{i,j} w_{ij} f_i f_j + \sum_{i,j} w_{ij} f_j^2\right) = \\
\frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2 \geq 0
\end{aligned}
\tag{2}
$$

Eigenvectors of Laplacian

- Consider eigenvector f corresponding to eigenvalue $\lambda = 0$.
  - $f^T L f = \lambda f^T f = 0$
- Using (2) we have that

$$0 = f^T L f = \frac{1}{2} \sum_{i,j} w_{i,j} (f_i - f_j)^2 \tag{3}$$

- If objects i and j are connected on the graph, there exists a path with $w_{uv} > 0$ along the path and from (3) it should be that $f_i = f_j$.
- So the set of eigenvectors of L is spanned by indicator vectors $I_{A_1}, I_{A_2}, ... I_{A_K}$ where A_i is i-th isolated region on the graph.
- Order of $\lambda = 0$ gives the number of isolated components.

Spectral clustering algorithm:

1. Find order K of $\lambda = 0$
2. Find set of eigenvectors $v_1, ...v_K$ corresponding to $\lambda = 0$
3. Cluster rows of $V = [v_1, ...v_K]$
4. Each row corresponds to object with the same index. Found clustering is the final clustering of initial objects.

Spectral clustering (unnormalized)

Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k eigenvectors $u_1, \dots, u_k$ of L.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns.
- For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \dots, C_k$.

Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

Practical application

- $L' = D^{-1}L$ is considered instead of L ("Normalized" Laplacian)
  - to account for different connectivity levels of different nodes

1. Most often singular values of L' are not exactly zero, but close to zero. So we select K smallest eigenvectors and corresponding K smallest eigenvectors.
2. Cluster rows of $[v_1, ...v_K]$
3. Found clustering is applied to objects with the same indexes.

Normalized spectral clustering

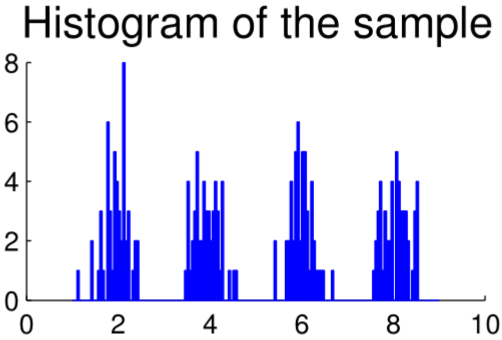
Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n\times n}$, number k of clusters to construct.

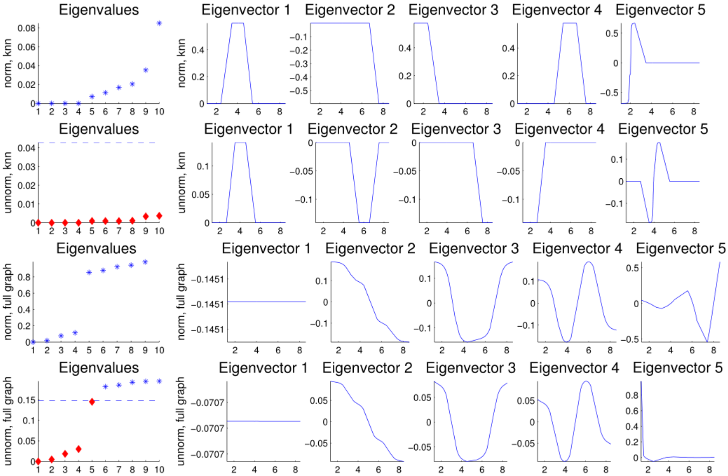
- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k eigenvectors $u_1, \dots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.**
- Let $U \in \mathbb{R}^{n\times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns.
- For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \dots, C_k$.

Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

Example



Example


Eigenvalues
Eigenvector 1
Eigenvector 2
Eigenvector 3
Eigenvector 4
Eigenvector 5
norm, knn
unnorm, knn
norm, full graph
unnorm, full graph