

3D Point Clouds

Lecture 4 – Clustering & Model Fitting

主讲人 黎嘉信

Aptiv 自动驾驶
新加坡国立大学 博士
清华大学 本科





1. Clustering - Spectral Clustering



2. Clustering – Mean-Shift and DBSCAN



3. Model Fitting - LSQ



4. Model Fitting – Hough Transform



5. Model Fitting - RANSAC



Lecture 3



K-Means

- Euclidean distance
- Hard assignment
- No modeling for a cluster
- Pre-defined cluster number k



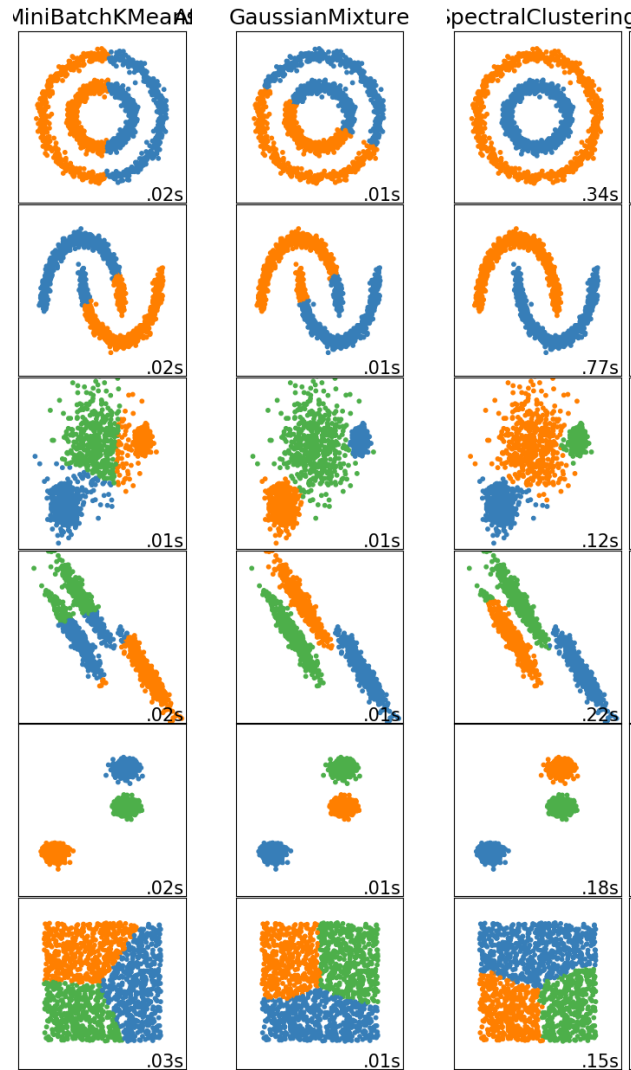
GMM

- Euclidean distance
- Probability formulation – soft clustering
- Mean and variance estimation for each cluster
- Pre-defined cluster number k



Spectral Clustering

- Works with connectivity
- Heuristic to determine cluster number k





Unnormalized Spectral Clustering

1. Build the graph to get adjacency matrix $W \in \mathbb{R}^{n \times n}$
2. Compute **unnormalized Laplacian L**
3. Compute the first (smallest) k eigenvectors **v_1, \dots, v_k of L**
4. Let $V \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors v_1, \dots, v_k as columns
5. For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i -th row of V
6. Cluster the points $\{y_i \in \mathbb{R}^k\}$ with k-means algorithm into clusters C_1, \dots, C_k
7. The final output clusters are A_1, \dots, A_k where $A_i = \{j | y_j \in C_i\}$



Spectral Clustering



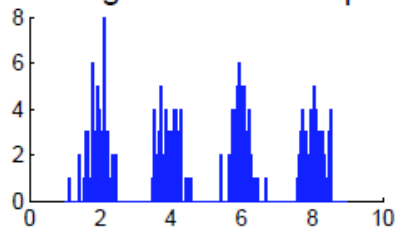
Selection of k can be done by eigenvalue analysis



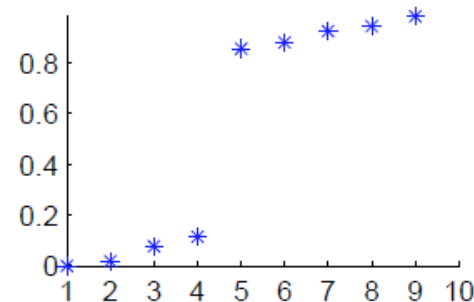
Most stable clustering is given by the value of k that maximizes the eigen-gap

- Eigengap is the difference between consecutive eigenvalues
- $\Delta_k = |\lambda_k - \lambda_{k-1}|$

Histogram of the sample



Eigenvalues





Normalized Spectral Clustering



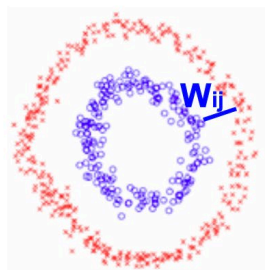
Unnormalized Spectral Clustering -> **approximated RatioCut**

$$\text{RatioCut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{|A_i|}$$

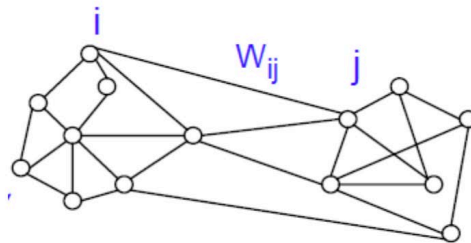
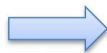


Normalized Spectral Clustering -> **approximated NormalizedCut**

$$\text{Ncut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)}$$



Data clustering



$G = \{V, E\}$

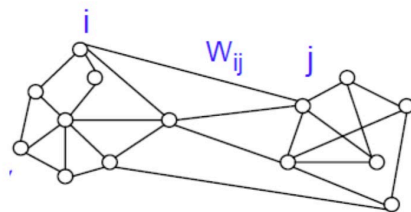


Graph Min-cut

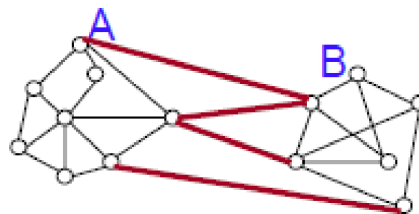


Min-cut: Partition graph $G = (V, E)$ into two sets A, B , such that the weights of edges connecting A, B is minimum

$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$$



$G = \{V, E\}$



Min-cut for k partitions. \bar{A}_i is the complement of subset $A_i \subset V$

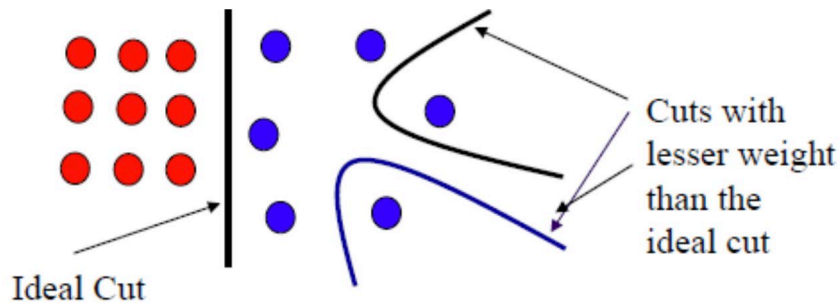
$$cut(A_1, \dots, A_k) = \sum_{i=1}^k cut(A_i, \bar{A}_i)$$



Graph Min-cut



Naïve min-cut suffers from degenerate results



Solution:

- Add constraints that the partition A_i can not be too small



How to evaluate the **size of a partition A_i** ?



Graph Min-cut – Size of A

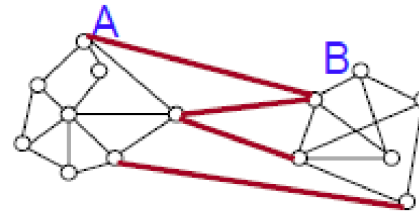
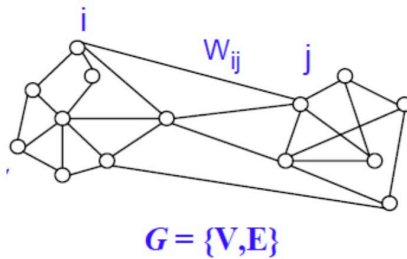
1. Number of vertices in A

$|A| :=$ the number of vertices in A

2. Sum of weights in A

$$\text{vol}(A) := \sum_{i \in A} d_i$$

$$d_i = \sum_{j=1}^n w_{ij} \quad \text{Weight sum for node } i$$





Spectral Clustering



Unnormalized Spectral Clustering -> **approximated RatioCut**

- Constrains the cluster have similar size
- $\text{Size}(A) = |A|$



Normalized Spectral Clustering -> **approximated NormalizedCut**

- Constrains the cluster have similar size
- $\text{Size}(A) = \text{vol}(A)$
- **$\text{Vol}(A)$ is large means nodes are closely connected inside $A \rightarrow A$'s element is similar.**



Spectral Clustering – Laplacian Matrix



Degree matrix D

- A diagonal matrix with degrees d_1, \dots, d_n on the diagonal
- $d_i = \sum_{j=1}^n w_{ij}$ is the row sum of adjacency matrix $W \rightarrow$ “how many edges are connected to node i ”



Unnormalized graph Laplacian matrix $L = D - W$



Normalized graph Laplacian matrix

- $L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$
- $L_{rw} = D^{-1} L = I - D^{-1} W$

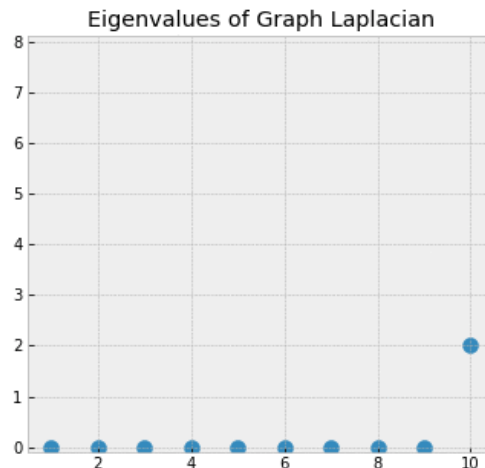
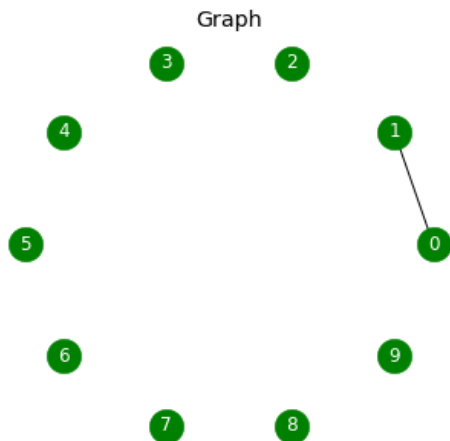


Spectral Clustering



Why do we need Laplacian matrix L ?

- Its eigenvalues / eigenvectors has some good properties.
 - One 0 eigenvalue \rightarrow one connected component
 - Corresponding eigenvectors \rightarrow which node belongs to that connected components





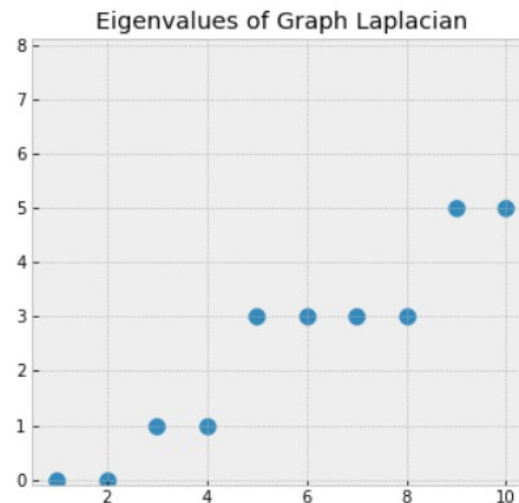
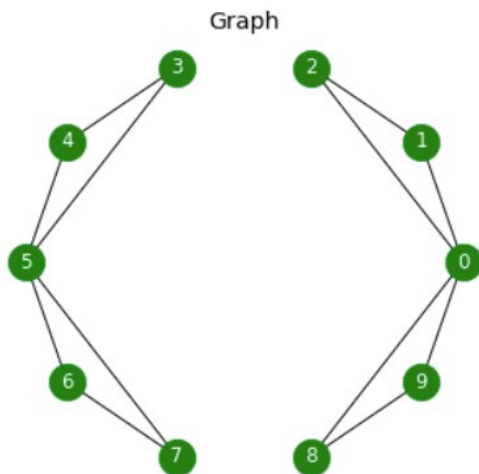
Spectral Clustering

$$W = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Why do we need Laplacian matrix L ?

- Its eigenvalues / eigenvectors has some good properties.
 - One 0 eigenvalue \rightarrow one connected component
 - Corresponding eigenvectors \rightarrow which node belongs to that connected components



1	0
1	0
1	0
0	1
0	1
0	1
0	1
0	1
1	0
1	0

=

1	0
1	0
1	0
1	1
1	1
1	1
1	1
1	1
1	0
1	0

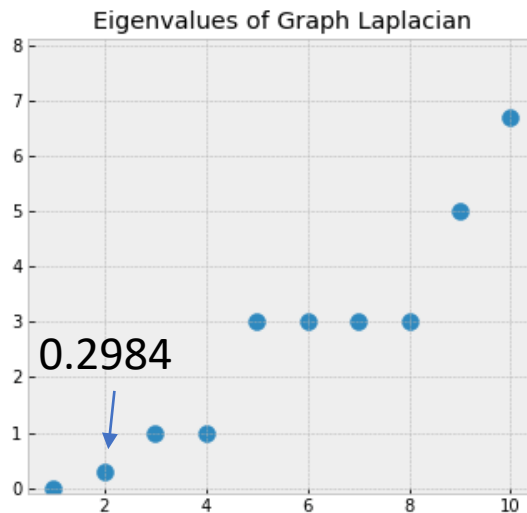
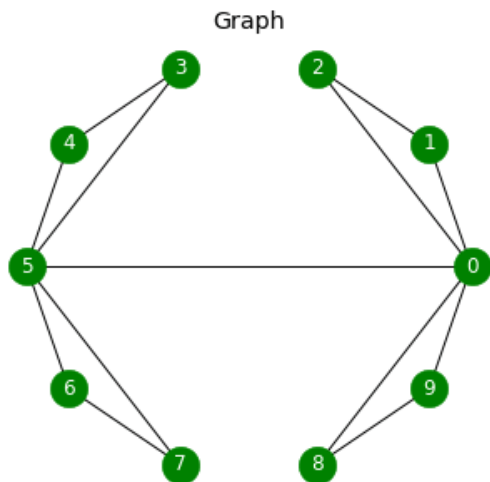


Spectral Clustering

$$W = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & \boxed{1} & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \boxed{1} & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$


Why do we need Laplacian matrix L ?

- Its eigenvalues / eigenvectors has some good properties.
 - One 0 eigenvalue \rightarrow one connected component
 - Corresponding eigenvectors \rightarrow which node belongs to that connected components



1	0	1	-0.23
1	0	1	-0.33
1	0	1	-0.33
0	1	1	0.33
0	1	1	0.33
0	1	1	0.23
0	1	1	0.33
0	1	1	0.33
1	0	1	-0.33
1	0	1	-0.33



Spectral Clustering – Graph Cut View



Unnormalized graph Laplacian matrix $L = D - W$

Proposition 1 (Properties of L) *The matrix L satisfies the following properties:*

1. *For every vector $f \in \mathbb{R}^n$ we have*

$$f^T L f = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} (f_i - f_j)^2$$

2. *L is symmetric and positive semi-definite.*

3. *The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbb{1}$.*

4. *L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.*



Proposition 1.1

1. For every vector $f \in \mathbb{R}^n$ we have $f^T L f = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} (f_i - f_j)^2$

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



Proposition 1.2 & 1.3 & 1.4

2. L is symmetric and positive semi-definite.

- $L = D - W$, D, W are symmetric
- $f^T L f \geq 0$, $\forall f \in \mathbb{R}^n$

3. The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbb{1}$.

$$L f = (D - W) f = [\cdots, d_i f_i - \sum_{j=1}^n w_{ij} f_j, \cdots]^T = 0 \cdot f, \quad f = \mathbb{1}$$

$$d_i = \sum_{j=1}^n w_{ij}$$

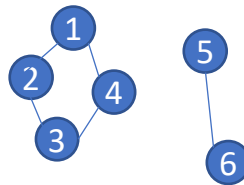
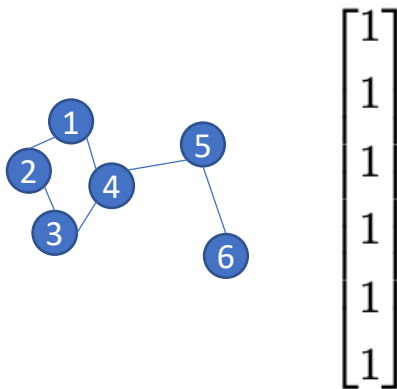
4. L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Direct result of 2 & 3



Spectral Clustering – Graph Cut View

Proposition 2 (Number of connected components) *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components A_1, \dots, A_k in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}$ of those components.*



$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

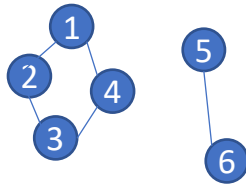


Spectral Clustering

- Wait, the eigenvector of the following graph is **NOT constant one**?

3. *The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbb{1}$.*

- $Lx = \lambda x$
- $Lx_1 + Lx_2 = \lambda_1 x_1 + \lambda_2 x_2$
- $\lambda_1 = \lambda_2 = 0$
- $L(x_1 + x_2) = \lambda(x_1 + x_2)$



x_1	x_2
$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

$$x_1 + x_2$$





Spectral Clustering – Graph Cut View

Proposition 2 (Number of connected components) *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components A_1, \dots, A_k in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}$ of those components.*

- Consider $k = 1$, i.e., G is a connected graph
- Assume f is eigenvector associated with eigenvalue 0

$$f^T L f = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} (f_i - f_j)^2 = f^T \cdot 0 = 0$$

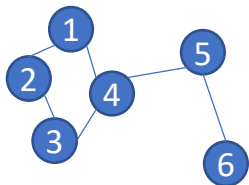
- $\forall i, j, f_i = f_j$ is the only choice to satisfy the above equation.



Spectral Clustering – Graph Cut View

$$f^T L f = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} (f_i - f_j)^2 = f^T \cdot 0 = 0$$

Graph



Connectivity / Similarity matrix

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$f^T L f = (f_1 - f_2)^2 + (f_2 - f_3)^2 + (f_3 - f_4)^2 + (f_4 - f_5)^2 + (f_5 - f_6)^2 = 0$$

\Rightarrow

$$f_1 = f_2 = f_3 = f_4 = f_5 = f_6$$

That is, the smallest eigenvector is constant vector



Spectral Clustering – Graph Cut View

Proposition 2 (Number of connected components) *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components A_1, \dots, A_k in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}$ of those components.*

- Consider $k \geq 2$, reorganize L into **block diagonal matrix** on the left
- The spectrum of L is the union of spectrum of L_i . So the 0-eigenvalue eigenvectors of L is shown on the right

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}$$

$$\begin{bmatrix} \mathbb{1}_1 & 0 & \cdots & 0 \\ 0 & \mathbb{1}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \mathbb{1}_k \end{bmatrix}$$



Spectral Clustering – Graph Cut Definition

For two disjoint subsets, $A, B \subset V$ For k disjoint subsets, A_1, \dots, A_k

$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}.$$

$$\text{cut}(A_1, \dots, A_k) = \sum_{i=1}^k \text{cut}(A_i, \bar{A}_i)$$

- Unnormalized Spectral Clustering -> **approximated RatioCut**

$$\text{RatioCut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{|A_i|}$$

- Normalized Spectral Clustering -> **approximated NormalizedCut**

$$\text{Ncut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)}$$



Approximated RatioCut for $k = 2$

- The problem is simplified into

$$\min_{A \subset V} \text{RatioCut}(A, \bar{A}) = \min_{A \subset V} \left(\frac{\text{cut}(A, \bar{A})}{|A|} + \frac{\text{cut}(\bar{A}, A)}{|\bar{A}|} \right)$$

- Given a subset $A \subset V$, construct a vector $f = [f_1, \dots, f_n]^T \in \mathbb{R}^n$

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A}. \end{cases}$$

- f determines the results of graph cut, but how to solve f ?

$$\begin{cases} v_i \in A & \text{if } f_i \geq 0 \\ v_i \in \bar{A} & \text{if } f_i < 0. \end{cases}$$



Approximated RatioCut for $k = 2$



Apply Proposition 1.1

$$\begin{aligned} f^T L f &= \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2 \\ &= \frac{1}{2} \sum_{i \in A, j \in \bar{A}} w_{ij} \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + \frac{1}{2} \sum_{i \in \bar{A}, j \in A} w_{ij} \left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 \\ &= \text{cut}(A, \bar{A}) \left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) \\ &= \text{cut}(A, \bar{A}) \left(\frac{|A| + |\bar{A}|}{|A|} + \frac{|A| + |\bar{A}|}{|\bar{A}|} \right) \\ &= |V| \cdot \text{RatioCut}(A, \bar{A}). \end{aligned}$$



Approximated RatioCut for $k = 2$



In addition, f is orthogonal to constant vector $\mathbb{1}$

$$\sum_{i=1}^n f_i = \sum_{i \in A} \sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i \in \bar{A}} \sqrt{\frac{|A|}{|\bar{A}|}} = |A| \sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}| \sqrt{\frac{|A|}{|\bar{A}|}} = 0.$$



$$\|f\| = \sqrt{n}$$

$$\|f\|^2 = \sum_{i=1}^n f_i^2 = |A| \frac{|\bar{A}|}{|A|} + |\bar{A}| \frac{|A|}{|\bar{A}|} = |\bar{A}| + |A| = n.$$



Approximated RatioCut for $k = 2$



Now the problem is converted to:

$$\min_{A \subset V} f^T L f, \text{ s.t., } f \perp \mathbb{1}, \|f\| = \sqrt{n},$$

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A}. \end{cases}$$



Approximation by dropping the last condition:

$$\min_{A \subset V} f^T L f, \text{ s.t., } f \perp \mathbb{1}, \|f\| = \sqrt{n}$$



Approximated RatioCut for $k = 2$



We are solving $\min_{A \subset V} f^T L f, \text{ s.t.}, f \perp \mathbb{1}, \|f\| = \sqrt{n}$



L is symmetric and Positive Semi-Definite



Recall the Rayleigh Quotients

Given a symmetric matrix $A \in S^n$,

$$\lambda_{\min}(A) \leq \frac{x^T A x}{x^T x} \leq \lambda_{\max}(A), \forall x \neq 0$$

$$\lambda_{\max}(A) = \max_{x: \|x\|_2=1} x^T A x$$

$$\lambda_{\min}(A) = \min_{x: \|x\|_2=1} x^T A x$$

The maximum and minimum are attained for $x = u_1$ and for $x = u_n$, respectively, where u_1 and u_n are the largest and smallest eigenvector of A , respectively.



Approximated RatioCut for $k = 2$



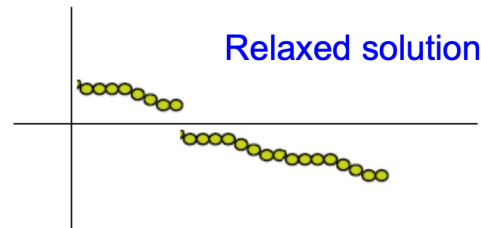
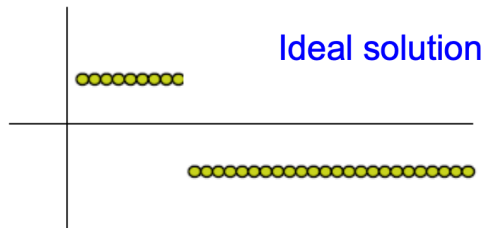
f is the **second** eigenvector of L



How can we get clusters?

$$\begin{cases} v_i \in A & \text{if } f_i \geq 0 \\ v_i \in \bar{A} & \text{if } f_i < 0. \end{cases}$$

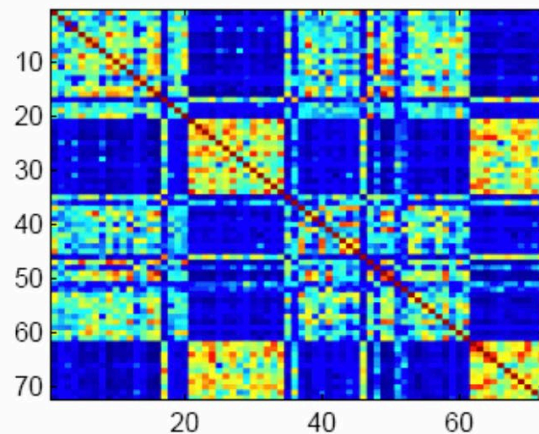
$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A}. \end{cases}$$



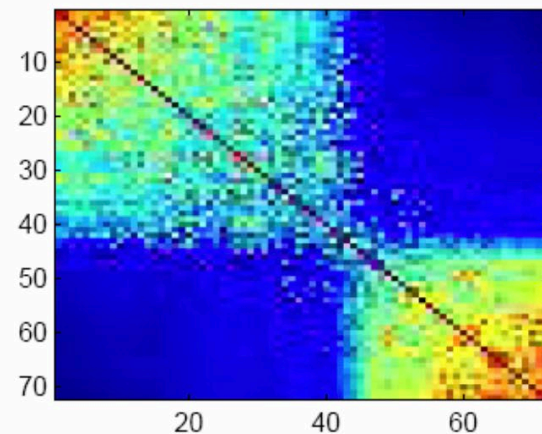
Run k-means on f .

- In practice, run k-means on the first two eigenvectors
- First eigenvector is graph connectivity, which helps as well

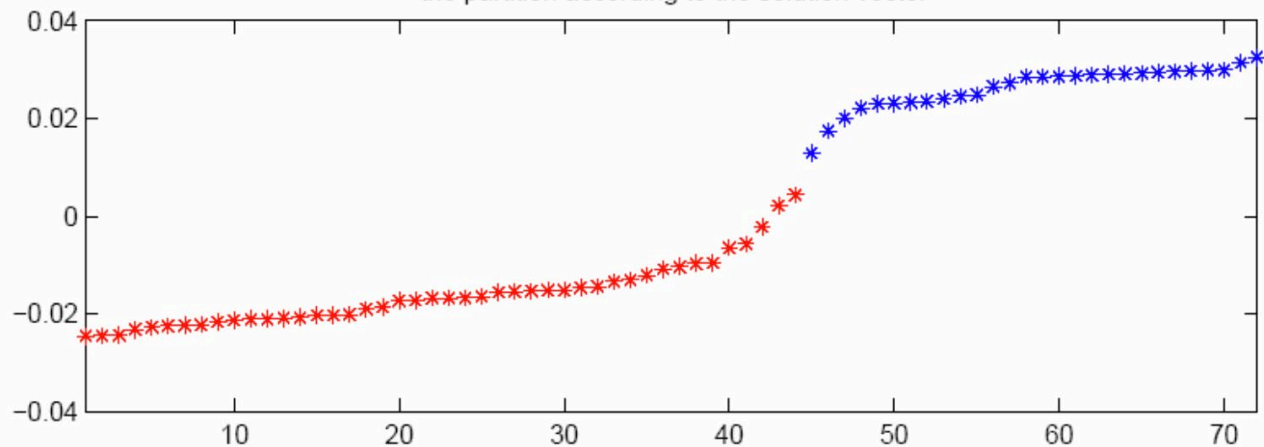
input affinity matrix



affinity matrix reordered according to solution vector



the partition according to the solution vector





Approximated RatioCut for $k \geq 2$

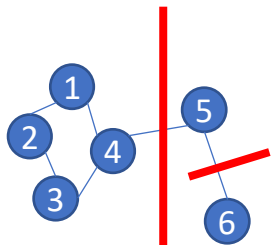


Construction the indication matrix $H \in \mathbb{R}^{n \times k}$, where,

$$h_{i,j} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } i \in A_j \\ 0 & \text{otherwise.} \end{cases}$$



A vertex belongs one cluster only $\rightarrow H$ has orthonormal columns $H^T H = I$



$$H = \begin{bmatrix} \frac{1}{\sqrt{4}} & 0 & 0 \\ \frac{1}{\sqrt{4}} & 0 & 0 \\ \frac{1}{\sqrt{4}} & 0 & 0 \\ \frac{1}{\sqrt{4}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{1}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{1}} \end{bmatrix}$$



Approximated RatioCut for $k \geq 2$



Construction the indication matrix $H \in \mathbb{R}^{n \times k}$, where,

$$h_{i,j} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } i \in A_j \\ 0 & \text{otherwise.} \end{cases}$$



A vertex belongs one cluster only $\rightarrow H$ has orthonormal columns $H^T H = I$



Denote the row vector as $h_i \in \mathbb{R}^k, i = 1, \dots, n$



Following similar calculations as $k = 2$, we have,

$$h_i^T L h_i = \frac{\text{cut}(|A_i|, |\bar{A}_i|)}{|A_i|}, \quad h_i^T L h_i = (H^T L H)_{ii}$$



Approximated RatioCut for $k \geq 2$



Recall the definition of RatioCut, we have

$$\begin{aligned}\text{RatioCut}(A_1, \dots, A_k) &= \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{|A_i|} \\ &= \sum_{i=1}^k h_i^T L h_i = \sum_{i=1}^k (H^T L H)_{ii} = \text{Tr}(H^T L H)\end{aligned}$$



Now the problem is transformed into

$$\min_{A_1, \dots, A_k} \text{Tr}(H^T L H) \text{ s.t., } H^T H = I, \quad h_{i,j} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } i \in A_j \\ 0 & \text{otherwise.} \end{cases}$$



Approximated RatioCut for $k \geq 2$



Approximation by dropping our H construction,

$$\min_{A_1, \dots, A_k} \text{Tr}(H^T L H) \text{ s.t., } H^T H = I$$



A more general form of Rayleigh Quotients, gives the solution:

- H contains the first k eigenvectors of L as columns



Apply k-means on the rows of H because of the approximation

~~$$h_{i,j} = \begin{cases} 1/\sqrt{|A_j|} & \text{if } i \in A_j \\ 0 & \text{otherwise.} \end{cases}$$~~



Normalized Spectral Clustering



The derivation is similar



The goal is Normalized Cut on the graph



Unnormalized Spectral Clustering -> **approximated RatioCut**

$$\text{RatioCut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{|A_i|}$$



Normalized Spectral Clustering -> **approximated NormalizedCut**

$$\text{Ncut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)}$$



Intuition of Spectral Cluster

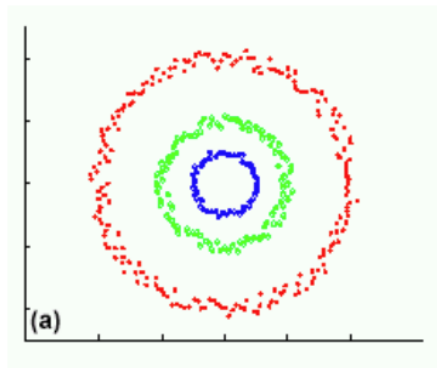
1. RatioCut or NormalizedCut

- Cut the graph into disjoint subsets with minimum cutting weights
- The size of the each subset should not be too small

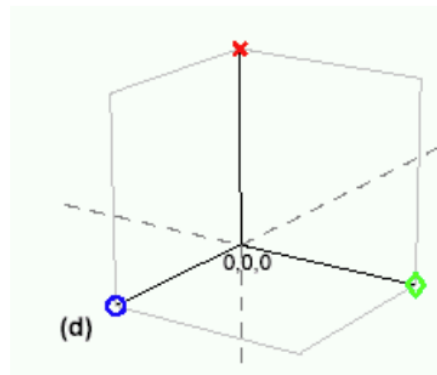
2. Dimension Reduction

- Project the n-dimension data into lower dimension space (spectral/eigenvector domain)

Original data



Projected data





Spectral Clustering - Summary



Complexity: $O(n^3)$

- This is the complexity of eigen decomposition
- K-means complexity is $O(n^2)$



Advantage

- No assumption on cluster shape
- Works with similarity, including Euclidean, connectivity
- Works with any dimensional data
- Able to estimate the number of clusters



Disadvantage

- Computational expensive
 - Can be alleviated using sparse similarity matrix and sparse eigen solver

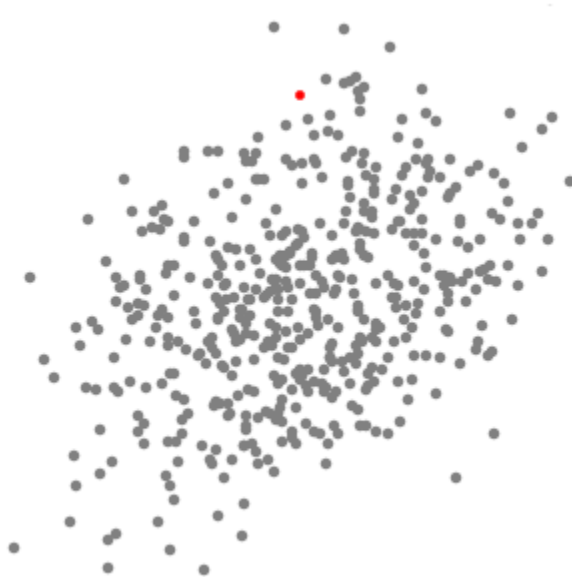


Mean Shift



Sliding windows hill climbing

- “Hill” is density

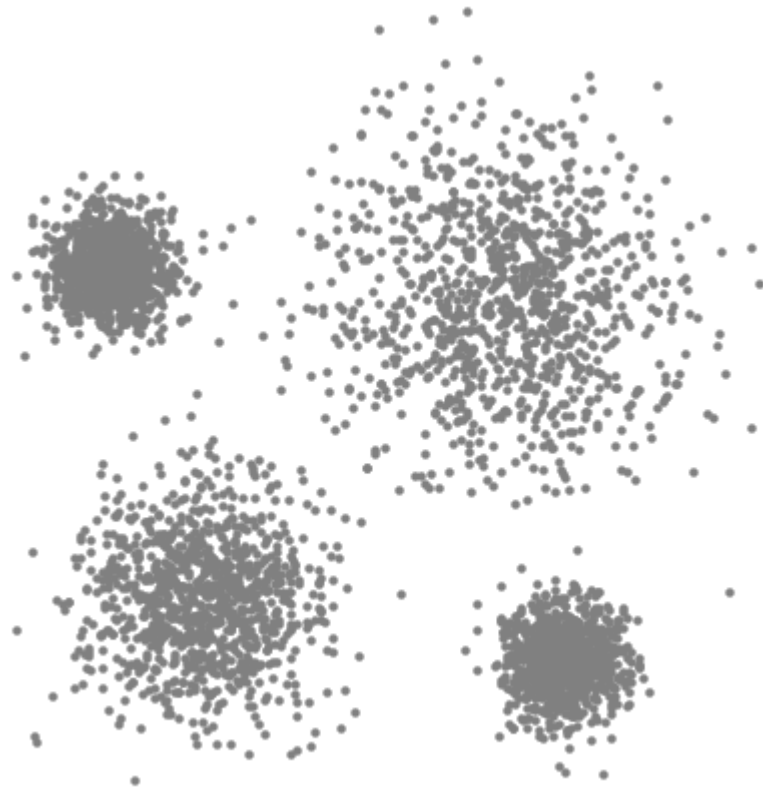




Mean Shift

1. Randomly select a circle with radius r
2. Move the circle to the center of the points inside
3. Repeat step-2 until it doesn't move
4. Repeat step-1,2,3. Remove overlapping circles
 - If circles overlap, select the one with most points
5. Determine clusters by finding the nearest circle center (similar to k-means)

Parameters: radius r





Mean Shift – Summary



Complexity $O(T \cdot n \cdot \log(n))$

- T is number of centers
- $n \cdot \log(n)$ is the complexity of radius based neighbor search, given 2D/3D data with kd-tree/octree



Advantage

- Automatically determines cluster numbers
- Single parameter
- Robust to outliers

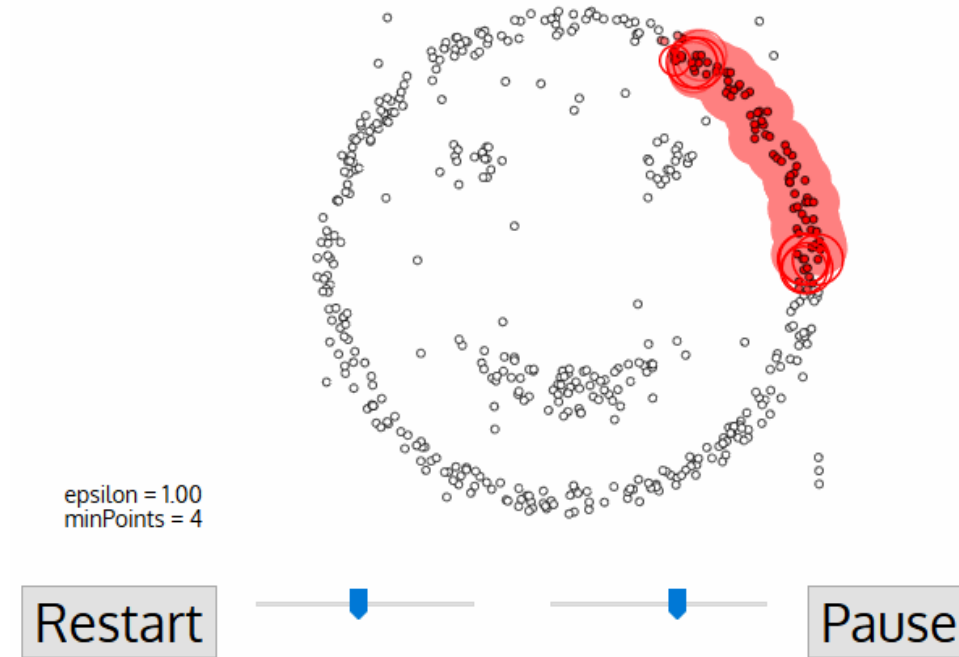


Disadvantage

- Hill climbing easily falls in local minima
- Depends on initialization
- Assumes clusters are in ellipse shape
- Mainly works in Euclidean space
- Doesn't scale with high dimensional data



Density-Based Spatial Clustering of Applications with Noise (DBSCAN)





DBSCAN

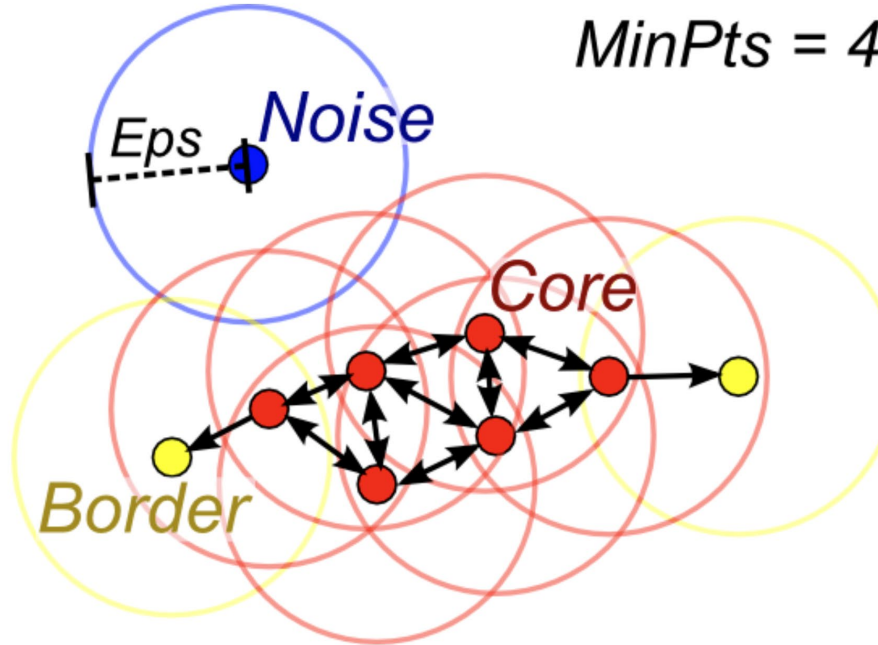
Preparation: all points labeled as unvisited

Parameters: distance r , min_samples

1. Randomly select a unvisited point p , find its neighborhood within r
2. Number of points within $r \geq \text{min_samples}$?
 - Yes. p is a **core point**, Create a cluster C , go to step 3, mark p as **visited**.
 - No. Mark p as **noise** and **visited**.
3. Go through points within its r -neighborhood, label it as C
 - If it is a **core point**, set it as the “new p ”, repeat step-3
4. Remove cluster C from the database, go to step-1
5. Terminate when all points are visited.



DBSCAN



Red: Core points. point number within circle ≥ 4

Yellow: Border points. Still part of the cluster because it is within r of a core point, but does not meet the `min_points` criteria

Blue: Noise point. Not assigned to a cluster.



DBSCAN – Summary



Complexity $O(n \cdot \log(n))$

- Radius NN search for each point



Advantage

- No assumption on cluster shape
- Automatically determines cluster numbers
- Robust to outliers



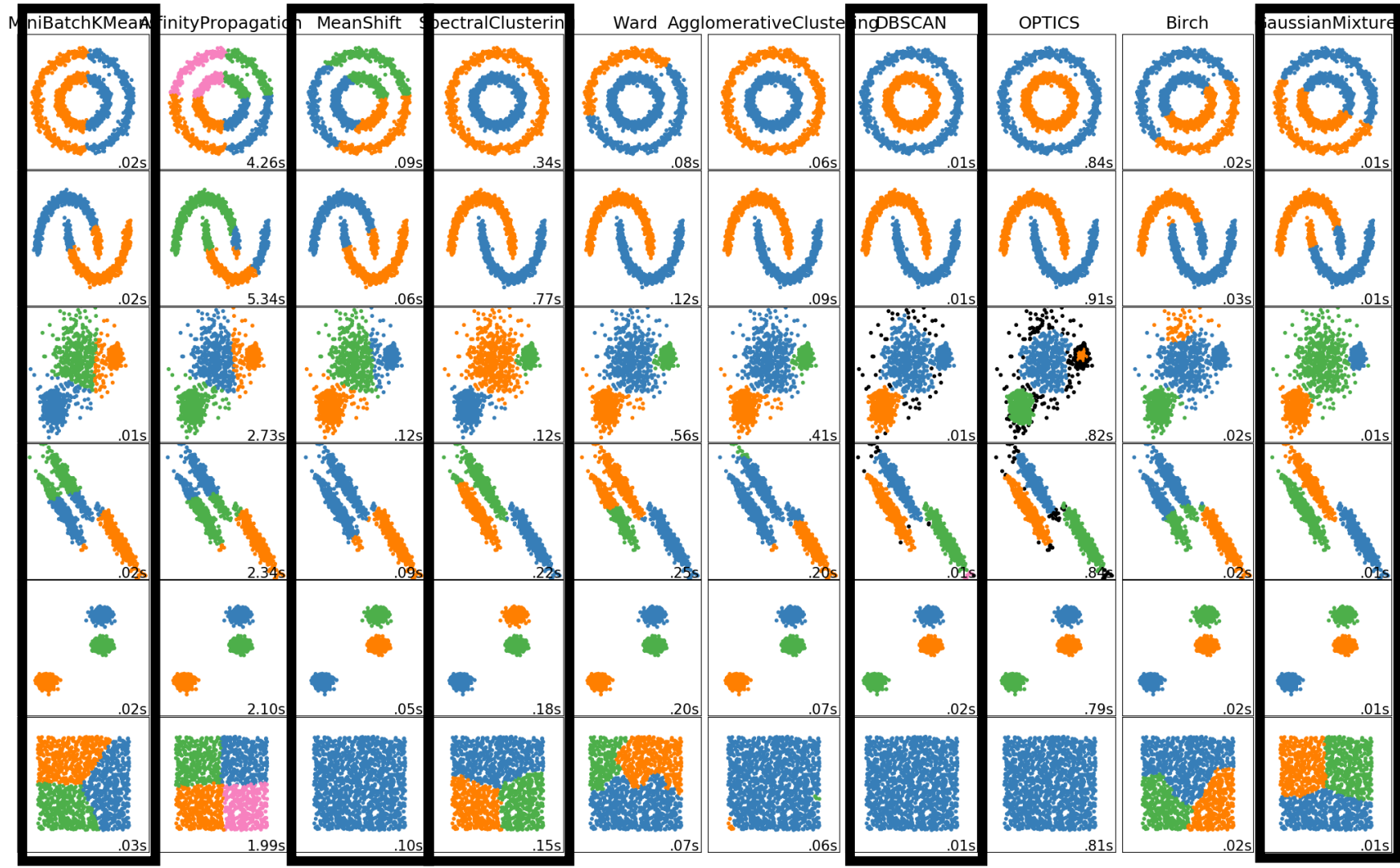
Disadvantage

- Doesn't work well with varying density
 - DBSCAN assume high density clusters are separated by some low density points
- Mainly works in Euclidean space
- Doesn't scale with high dimensional data



Clustering - Summary

	K-Means	GMM	Spectral	Mean Shift	DBSCAN
Metric	Euclidean	Euclidean	Similarity	Density /Euclidean	Density /Euclidean
# of clusters	Pre-defined	Pre-defined	Heuristic	Automatic	Automatic
Robustness to outlier	Bad	Medium	Good	Good	Good
High dimension data	Medium	Medium	Good	Bad	Bad
Complexity	$O(t \cdot k \cdot n \cdot d)$ t: iteration k: # of clusters n: # of data d: dimension	$O(t \cdot k \cdot n \cdot d)$ t: iteration k: # of clusters n: # of data d: dimension	$O(n^3)$ n: # of data	$O(Tn \log(n))$ n: # of data T: # of centers	$O(n \cdot \log(n))$ n: # of data





Model Fitting



Take line fitting as example



Approaches:

- Least Square
- Hough Transform
- Random Sample Consensus (RANSAC)



Model Fitting



If we know the inlier points

- Least Square



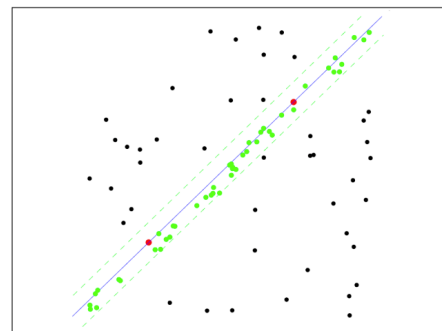
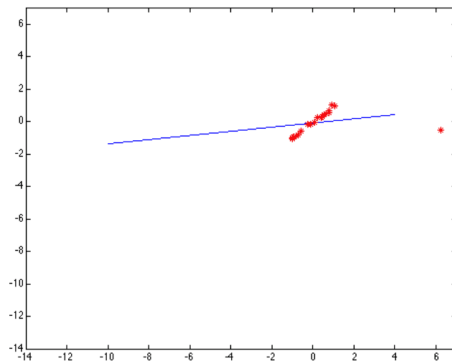
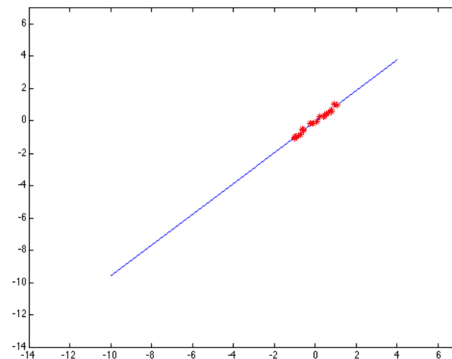
What if there is small amount of outliers?

- Robust Least Square, e.g., robust loss function
- Hough Transform
- RANSAC



What if there are lots of outliers / more than one models in data?

- Hough Transform
- RANSAC





Least Square Fitting



Given a set of points $\{p_1, \dots, p_n\}$, find a line that fits the point set best.

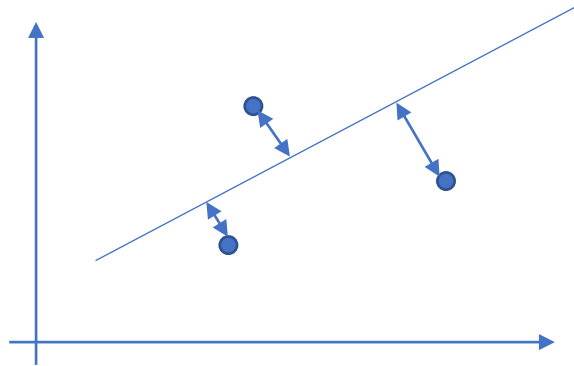


Consider 2-dimension, $p_i \in \mathbb{R}^2$, line model is $ax + by + c = 0$



Definition of “fit” – minimize the perpendicular distance

$$E = \sum_{i=1}^n (ax_i + by_i + c)^2$$





Least Square Fitting



Re-organize the problem into standard least-square optimization (sometimes called homogeneous equations):

$$\hat{\mathbf{x}} = [\hat{a}, \hat{b}, \hat{c}]^T = \min_{\mathbf{x}} \|A\mathbf{x}\|_2^2, \text{ s.t. } \|\mathbf{x}\|_2 = 1$$

$$A = \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix}, \mathbf{x} = [a, b, c]^T$$



The solution is obvious: $[a, b, c]^T$ is the **eigenvector of the smallest eigenvalues of A**

- Given A is full column rank, i.e., $n \geq 3$



Least Square Fitting

Many model fitting problem can be formulated as least square (LSQ) optimization problem.

Linear LSQ problem $A\mathbf{x} = 0$

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \|A\mathbf{x}\|_2^2, \text{ s.t., } \|\mathbf{x}\|_2 = 1, A \in \mathbb{R}^{n \times m}, \mathbf{x} \in \mathbb{R}^m$$

- Solution given by eigenvector of the smallest eigenvalue of A

Linear LSQ problem $A\mathbf{x} = \mathbf{b}$:

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2, A \in \mathbb{R}^{n \times m}, \mathbf{x} \in \mathbb{R}^m, \mathbf{b} \in \mathbb{R}^n$$

- In the case that $n \geq m$, the solution is given by $\hat{\mathbf{x}} = (A^T A)^{-1} A^T \mathbf{b}$

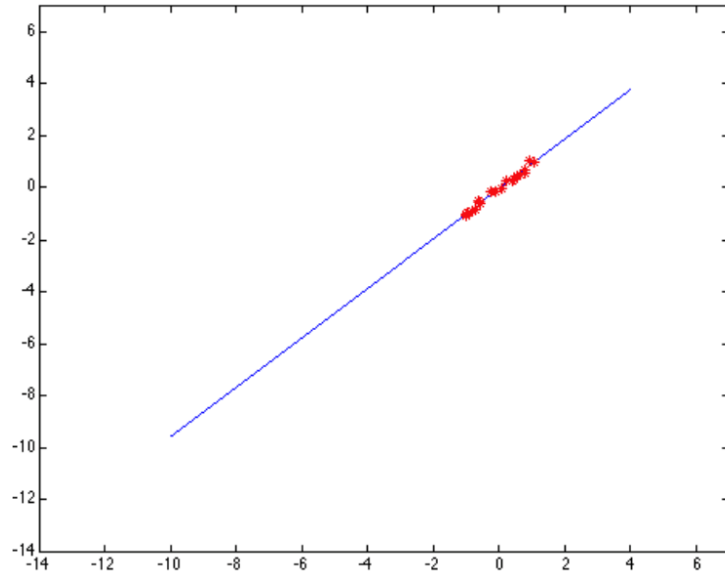
Linear LSQ problem $A\mathbf{x} = \mathbf{b}, \text{ s.t., } C\mathbf{x} = 0$



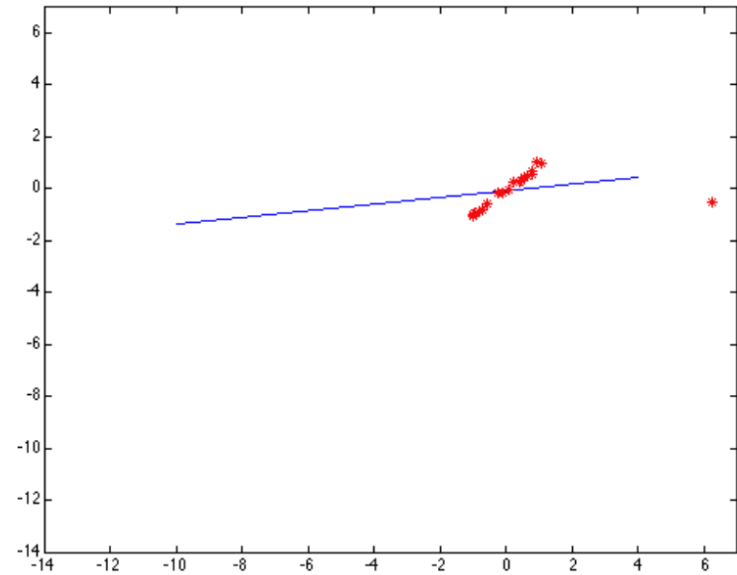
Limitations of LSQ



Sensitive to Outlier



LSQ with No Outlier



LSQ with One Outlier

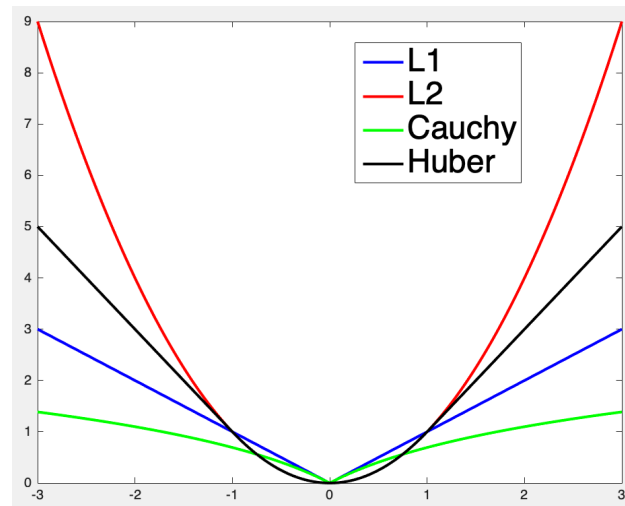


Loss functions



Typical loss functions

- L1. $\rho = |s|$
- L2. $\rho = s^2$
- Cauchy. $\rho = \log(1 + |s|)$
- Huber. $\rho = \begin{cases} s^2, & |s| < \delta \\ 2\delta(|s| - \frac{1}{2}\delta), & \text{otherwise} \end{cases}$
- etc.



Robust loss functions like Huber, Cauchy reduce the effect of outliers



However, the problem becomes *non-linear*!



Non-Linear LSQ



A general formulation of LSQ

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \|f(\mathbf{x})\|^2$$



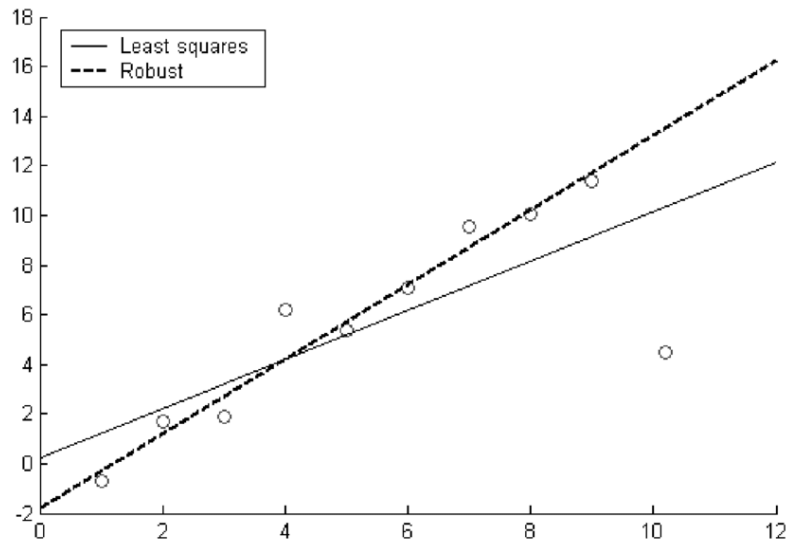
Function f is the non-linear function

- e.g., coupling the robust loss function with linear LSQ



Optimization methods

- Gradient descent
- Gauss-Newton
- Levenberg-Marquardt





Model Fitting



Least Square / Robust Least Square

- No outlier / few outlier
- Simple and fast



What if:

- Lots of outliers
- More than one models in the data, e.g., more than one line.



Hough Transform



Random Sample Consensus (RANSAC)



Hough Transform

- Discretize parameter spaces into bins
- For each data point, vote the bins that can generate this data point
- Find the bins with most votes

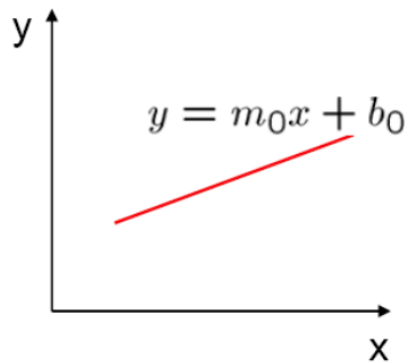
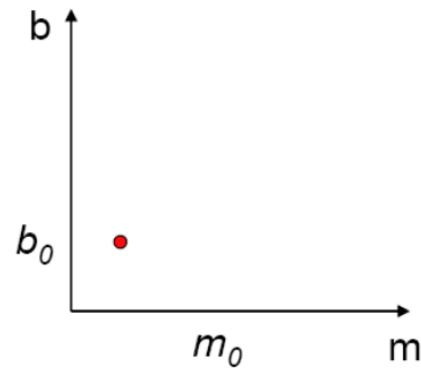


Image space



Line parameter space



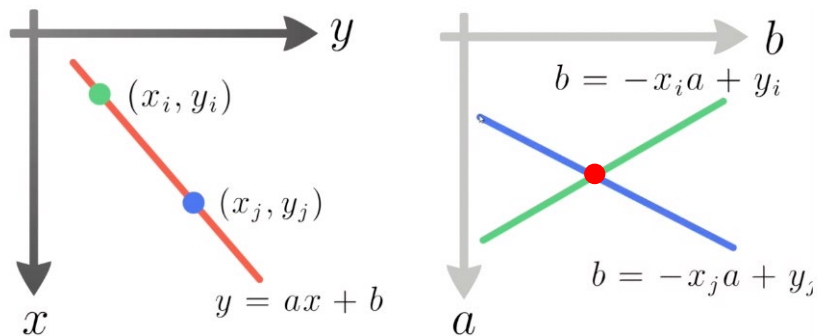
Hough Transform



A point in the Euclidean space \rightarrow A line in the parameter space



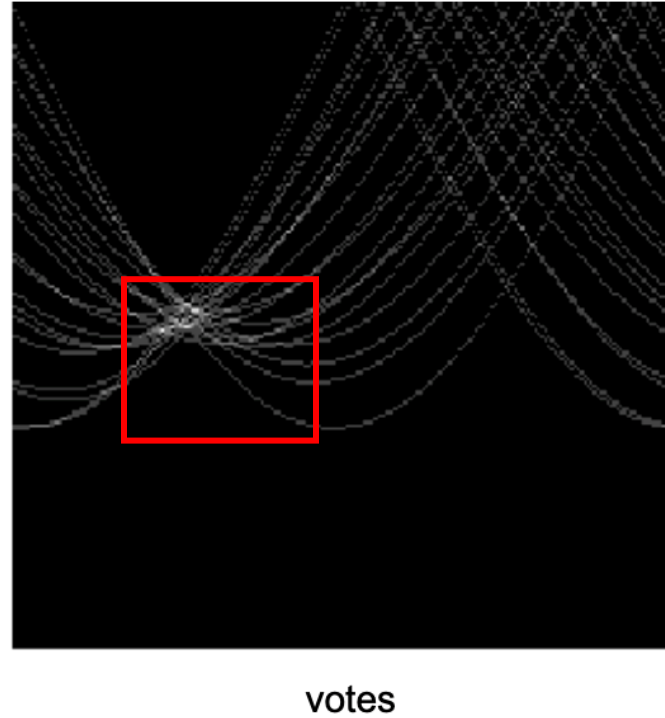
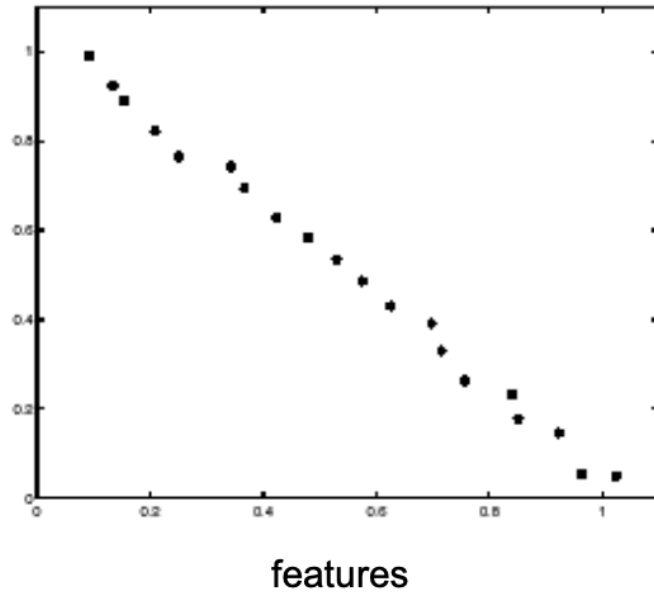
Select the bin with most votes





Hough Transform – Effect of Noise

Blurred peak
– in accurate parameter estimation





Hough Transform



Model parameterization. E.g., for a line

- $y = ax + b$ is non-uniform, can't represent vertical lines (a is infinity)
- $x\cos\theta + y\sin\theta = r$ is a better model with parameters $\{\theta, r\}$



Selection of resolution

- Tradeoff between speed and precision



Apply smoothing at the parameter space before searching for the highest vote

- E.g., Gaussian smooth
- Reduce the effect of noise



Hough Transform – Extension for Circles



The circle model with parameters $\{a, b, r\}$

$$(x - a)^2 + (y - b)^2 = r^2$$



Each point (x, y) fills a set of parameter bins for $\{a, b, r\}$ that fulfills the above function.

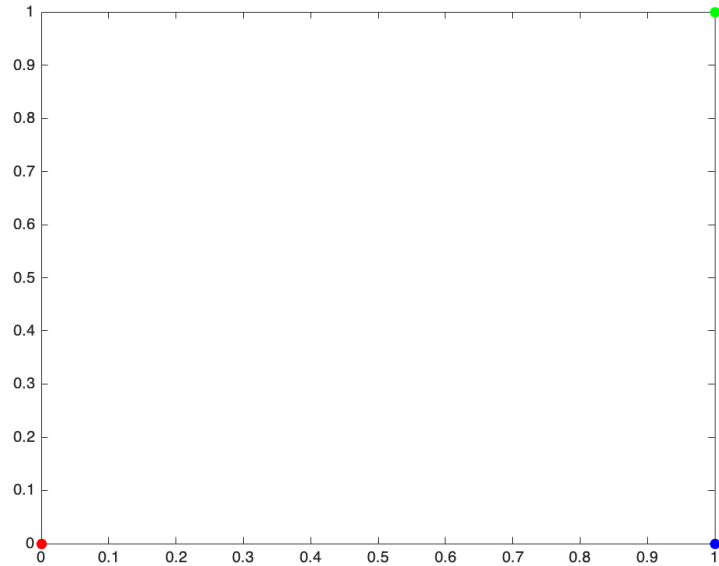


How to find those bins?

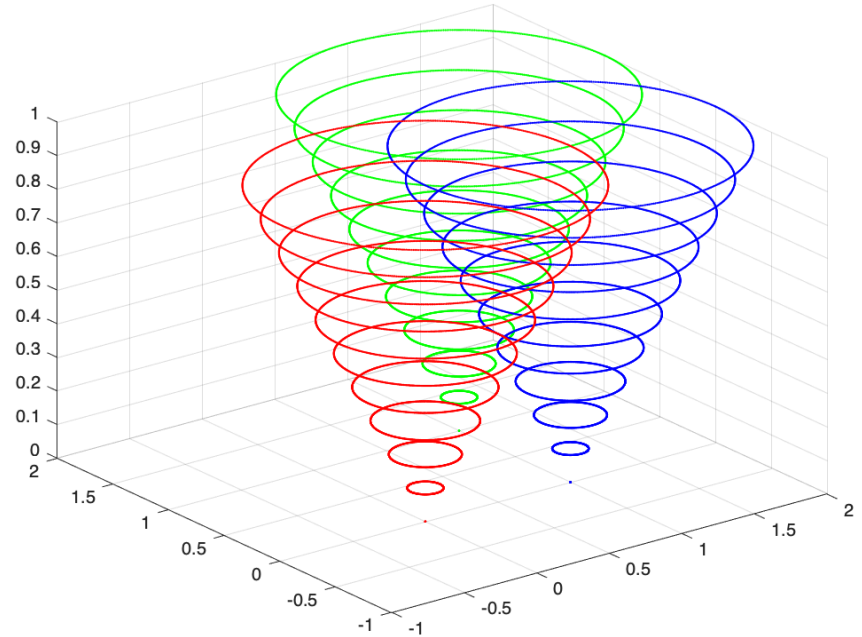
- Fix $r = r_i$
- Uniformly sample a set of $\{\theta_1, \dots, \theta_k\}$
- Each θ_j generates a set of $\{a, b\}$
 - $a = x - r_i \cos \theta_j$
 - $b = y - r_i \sin \theta_j$



Hough Transform – Extension for Circles



3 points in Euclidean space: $[0, 0]$, $[1, 1]$, $[1, 0]$



Each point corresponds to a "cone" in Hough space



Hough Transform – Summary



Advantage

- Robust to noise
- Robust to missing points of the shape
- Can be extended to lots of models



Disadvantage

- Doesn't scale well with complicated models
 - Usually works for models with less than 3 unknown parameters



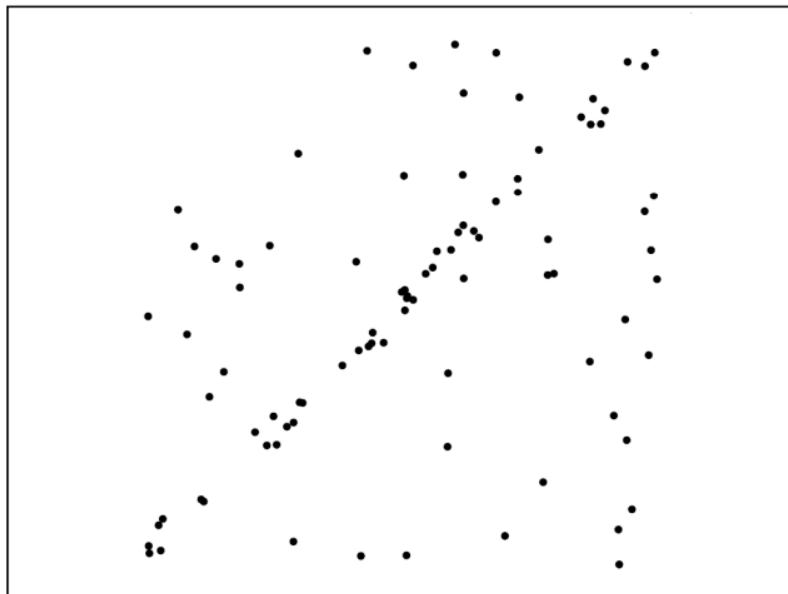
RANSAC



Simple and works well in practice



Works well with complicated models





RANSAC – Line Fitting

1. Randomly select a sample (minimal subset of points required to solve the model)

$$p_0 = (x_0, y_0), p_1 = (x_1, y_1)$$

2. Solve the line model

$$x = x_0 + at$$

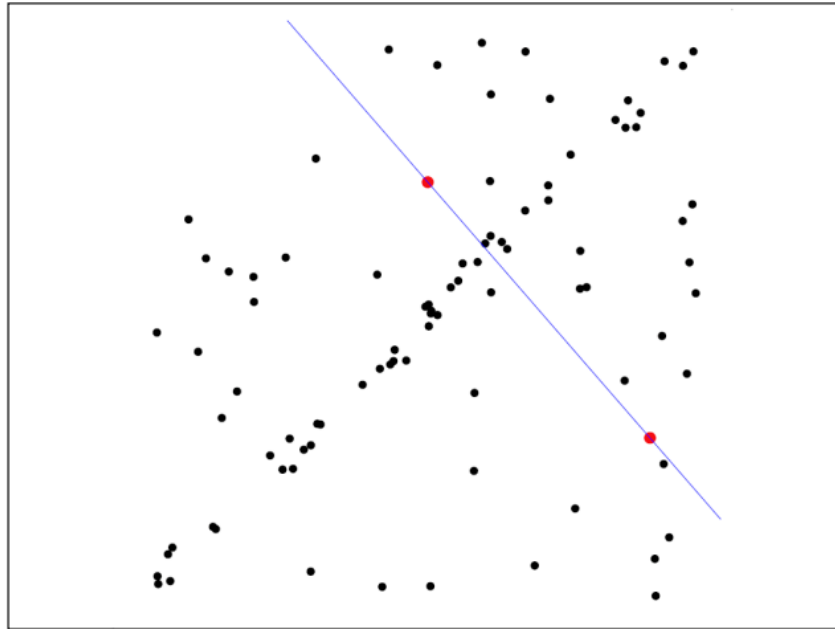
$$y = y_0 + bt$$

where,

$$p_0 = [x_0, y_0]^T, n = [a, b]^T$$

$$\Delta x = x_1 - x_0, \Delta y = y_1 - y_0$$

$$\Delta x = at, \Delta y = bt \rightarrow \frac{\Delta x}{\Delta y} = \frac{a}{b}$$



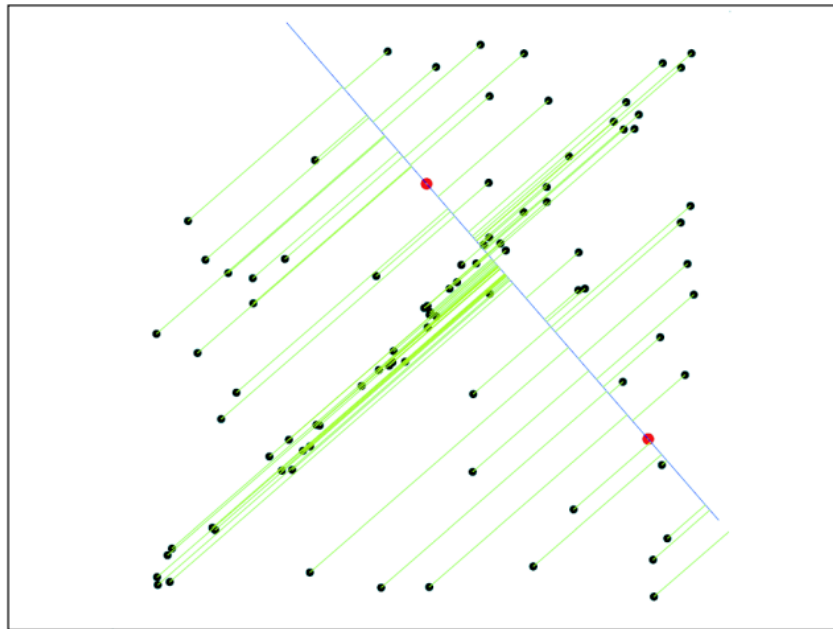


RANSAC – Line Fitting

1. Randomly select a minimal subset of points required to solve the model
2. Solve the model
3. Compute error function for each point

$$p_i = (x_i, y_i)$$

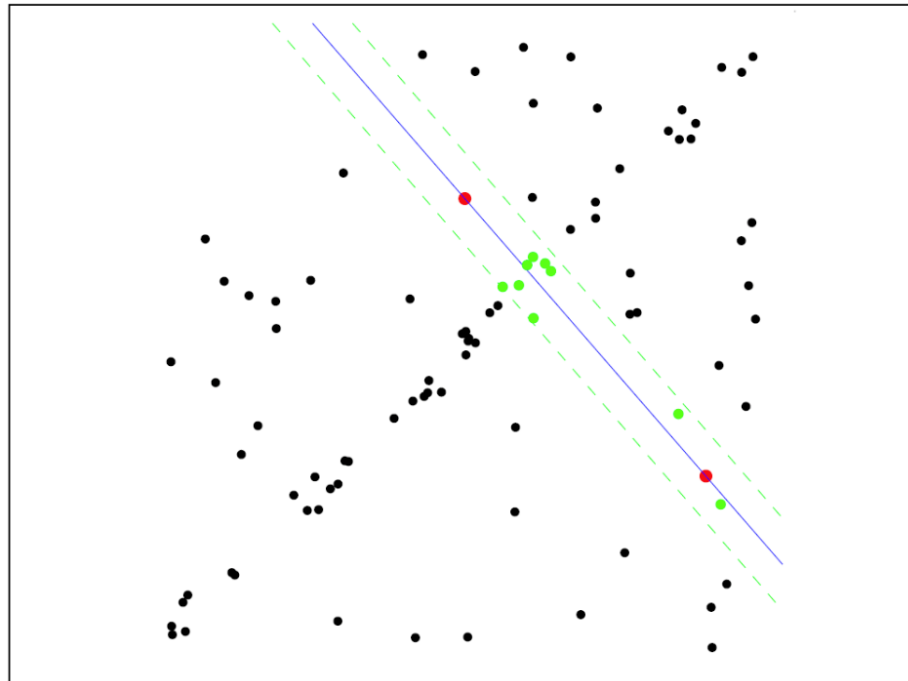
$$d_i = \frac{n^T (p_i - p_0)}{\|n\|_2}$$





RANSAC – Line Fitting

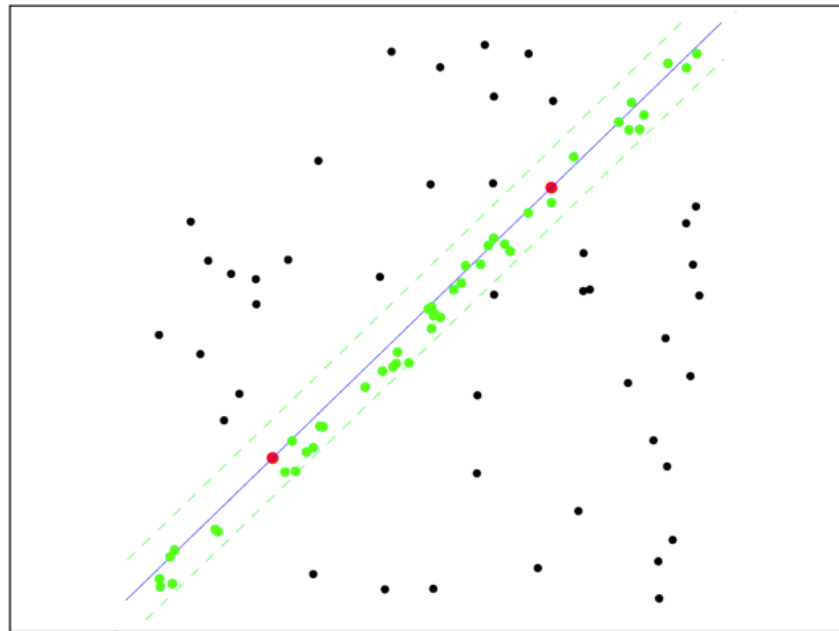
1. Randomly select a minimal subset of points required to solve the model
2. Solve the model
3. Compute error function for each point $p_i = (x_i, y_i)$
4. Count the points consistent with the model, $d_i < \tau$ (inlier)





RANSAC – Line Fitting

1. Randomly select a minimal subset of points required to solve the model
2. Solve the model
3. Compute error function for each point $p_i = (x_i, y_i)$
4. Count the points consistent with the model, $d_i < \tau$
5. Repeat step 1-4 for N iterations, choose the model with most inlier points





RANSAC – Line Fitting



Distance threshold τ

- Usually chosen empirically
- Chi-square distribution χ^2



Number of iterations N

- Choose N so that with probability p , as least one random sample is free from outliers, e.g., $p = 0.99$



RANSAC – Threshold τ by χ^2



Assume the error between data point and model is Gaussian distribution $d \sim \mathcal{N}(0, \sigma^2)$



χ^2 distribution – sum of squares of k independent standard normal distribution. Assume 95% confidence that the point is an inlier

- 1DoF χ_1^2 for 2D/3D line fitting or 3D plane fitting. The error is perpendicular distance
 - $\tau = \sqrt{3.84\sigma^2}$
- 2DoF χ_2^2 for 2D point distance, because the distance is $\Delta x^2 + \Delta y^2$
 - $\tau = \sqrt{5.99\sigma^2}$
- 3DoF χ_3^2 for 3D point distance, because the distance is $\Delta x^2 + \Delta y^2 + \Delta z^2$
 - $\tau = \sqrt{7.81\sigma^2}$



RANSAC – Threshold τ by χ^2

Degrees of freedom (df)	χ^2 value ^[19]											
1	0.004	0.02	0.06	0.15	0.46	1.07	1.64	2.71	3.84	6.63	10.83	
2	0.10	0.21	0.45	0.71	1.39	2.41	3.22	4.61	5.99	9.21	13.82	
3	0.35	0.58	1.01	1.42	2.37	3.66	4.64	6.25	7.81	11.34	16.27	
4	0.71	1.06	1.65	2.20	3.36	4.88	5.99	7.78	9.49	13.28	18.47	
5	1.14	1.61	2.34	3.00	4.35	6.06	7.29	9.24	11.07	15.09	20.52	
6	1.63	2.20	3.07	3.83	5.35	7.23	8.56	10.64	12.59	16.81	22.46	
7	2.17	2.83	3.82	4.67	6.35	8.38	9.80	12.02	14.07	18.48	24.32	
8	2.73	3.49	4.59	5.53	7.34	9.52	11.03	13.36	15.51	20.09	26.12	
9	3.32	4.17	5.38	6.39	8.34	10.66	12.24	14.68	16.92	21.67	27.88	
10	3.94	4.87	6.18	7.27	9.34	11.78	13.44	15.99	18.31	23.21	29.59	
P value (Probability)	0.95	0.90	0.80	0.70	0.50	0.30	0.20	0.10	0.05	0.01	0.001	

https://en.wikipedia.org/wiki/Chi-squared_distribution



RANSAC – Number of Iterations N



e : outlier ratio (probability that a point is an outlier)



s : number of points in a sample (e.g., in line fitting a sample contains 2 points)



N : sample number N (number of RANSAC iteration)



p : confidence we get at least a good sample that is free from outliers

$$\underbrace{(1 - (1 - e)^s)}^{\text{Probability of choosing } s \text{ inliers in a row}}^N = 1 - p \quad \underbrace{(1 - (1 - e)^s)}^{\text{Probability that one or more points are outliers}}^N = 1 - p \quad \underbrace{(1 - (1 - e)^s)}^{\text{Probability that } N \text{ samples are contaminated}}^N = 1 - p$$

Probability of choosing s
inliers in a row

Probability that one or more
points are outliers

Probability that N samples are
contaminated



RANSAC – Number of Iterations N



Iteration number N is given by

$$N = \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$



Table for $p = 0.99$

proportion of outliers e							
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177



RANSAC - Practical Tricks



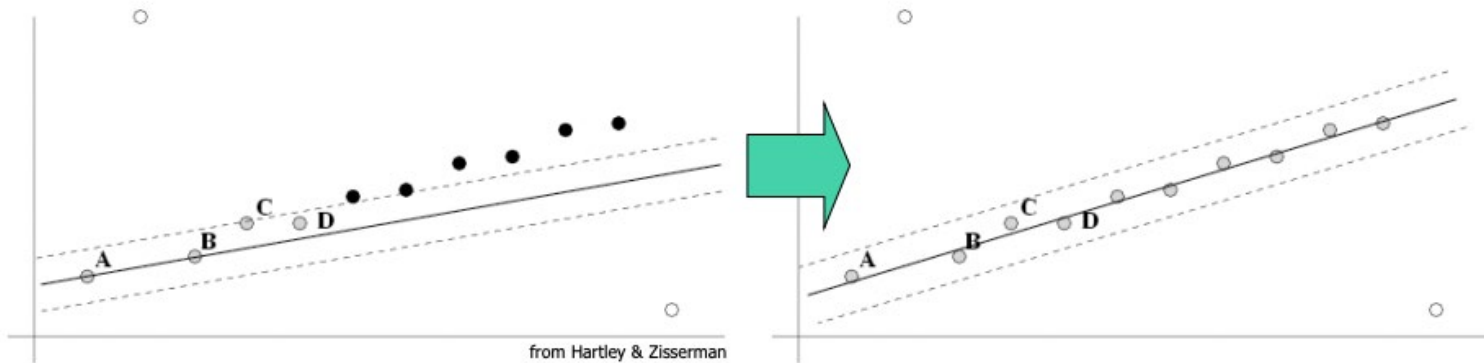
Don't need to perform N samples/iterations

- Terminate when the inlier ratio reach the expected inlier ratio

$$T = (1 - e) \cdot total_num_of_data_points$$



Run LSQ to refine the model after selecting the final model and inlier points





RANSAC - Summary



Advantages

- Simple and general
- Usually works well in practice, even with low inlier ratio like 10%



Disadvantages

- Need to determine the inlier threshold τ
- Need large number of samples when inlier ratio is low



Homework



Object detection pipeline for lidar

- Use KITTI 3D object detection dataset, select 3 point clouds, do the followings.
- Step 1. Remove the ground from the lidar points. Visualize ground as blue.
 - Any method you want – LSQ, Hough, RANSAC
- Step 2. Clustering over the remaining points. Visualize the clusters with random colors.
 - Any method you want
- Step 3. Classification over the clusters
 - Homework of Lecture 5
- Step 4. Report the detection precision-recall for three categories: vehicle, pedestrian, cyclist
 - Homework of Lecture 5