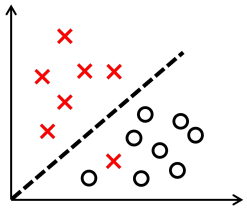
计算机应用数学-2025 秋

Lecture 1.7: 聚类

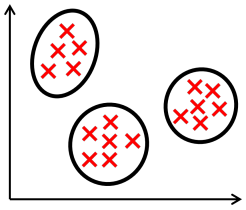
2025.11.26

Lecturer: 宋骐 *Scribe:* 王亦涵, 赵奇, 王馨语, 郑岭

Supervised learning vs. Unsupervised learning



Data-labled



Data-Unlabled

例子：社交网络
基因表达分析
异常检测（信用卡）

Need training/testing data.

Classification.

A classifier that can be used to distinguished x and o.

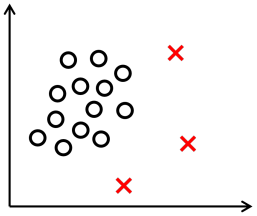
其他类型：强化学习等.

-

Clustering.

Put the data points that are similar into the same group.

PARTI. Clustering.



clustering: partitioning a set of objects into clusters, or simply finding some clusters.

cluster: a group of objects such that objects inside the group are more similar (is some sense) to each other that to those in other groups.

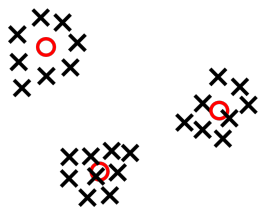
n: # of data points. K: $\#$ of desired clusters.

$A = \{a_1, \cdots, a_n\}$ n data points, a_i - row vector

$a_i \in \mathbb{R}^d$ (sometimes we assume $a_i \in \{0,1\}^d$)

1. Center-based clustering

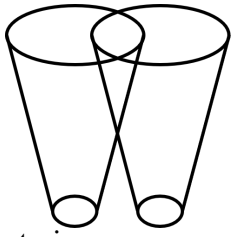
Each cluster is represented as a central vector, which is not necessarily a member of the dataset.



Examples: K-means / K-median / K-centers.

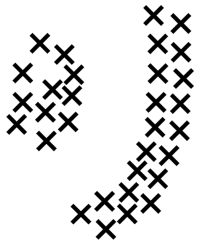
2. Spectral clustering

Project the data into a new space and run K-means $\cdots$



clusters in the full space and their projections.

3. Density-based clustering



clusters are defined as areas of higher density than the reminder of the dataset.

4. Hierarchical clustering

No need to specify K.

The cluster at each level of the hierarchy are created by merging clusters at the next lower level.

Preliminaries: Distance between objects.

Suppose the data points are from $M \subseteq \mathbb{R}^d$ or $M \subseteq \{0,1\}^d$

Metric.

D. $M \times M \to \mathbb{R}$ if for all $x, y, z \in M$

1. $D(x,y) = 0 \Leftrightarrow x = y$
2. $D(x,y) = D(y,x)$
3. $D(x,z) \leq D(x,y) + D(y,z) - triangle\ inequality$

Note that $D(x,y) \geq 0$

Proof: $D(x,y) + D(y,x) \geq D(x,x)$

which gives $2D(x,y) \geq D(x,x) \geq 0$ and thus $D(x,y) \geq 0$

We also call D Distance function.

Example: $D_{l_2}(x,y) = \|x-y\|_2 = \left(\sum_{i=1}^{d} |x_i - y_i|^2\right)^{1/2}$

Euclidean distance

$D_{l_1}(x, y) = \sum_{i=1}^{d} |x_i - y_i|$
Manhattan Distance

$D_{l_p}(x, y) = \|x - y\|_p = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}$
$p \geq 1$. Minkowski distances l_p-norms.
A generalized version of Euclidean and Manhattan distance.

1 Center-based clustering

1.1 K-median / K-means / K-center definition

I. K-median: Given $A \subset M, K \geq 1$, find $C = \{C_1, \cdots, C_K\} \subseteq M$
that minimize

$$\sum_{a \in A} \min_{1 \leq i \leq K} D(a, C_i)$$

where C_i's are called centers.
- Given $C = \{C_1, \cdots, C_K\}$ define
$C_i = \{a \in A \mid \forall j, D(a, C_i) \leq D(a, C_j)\}$
- If ties are broken, $C = \{C_1, \cdots, C_K\}$ is a partition of A. Then equivalently, we are minimizing

$$\sum_{j=1}^{K} \sum_{a \in C_j} D(a, C_j)$$

Some notations:
1. If $C \subseteq M$, $|C| < \infty$, $D(x, C) := \min_{c \in C} D(x, c)$
2. If $A, C \subseteq M$, $|A|, |C| < \infty$, $D(A, C) := \sum_{a \in A} D(a, C)$
Called (D-) cost of A with respect to C.
3. If $K \in \mathbb{N}$, $\text{cost}_K^D(A) := \min_{C \subseteq M,\ |C|=K} D(A, C)$
Called K-median cost of A.

Thus, K-median problem is the following:
Given $A \subseteq M$, $K \subseteq \mathbb{N}$, find $C = \{C_1, \cdots, C_K\} \subseteq M$, s.t. $D(A, C) = cost_K^D(A)$.
For K-median, we use Manhattan Distance.

II. K-means: Given $A \subset M, K \geq 1$, find $C = \{C_1, \cdots, C_K\} \subseteq M$

that minimize

$$\sum_{a\in A} \min_{1\le i\le K} D^2(a, C_i)$$

where C_i's are called centers.

$\star$ That is, the difference from the K-median includes:

1. replace "$D(a, C_i)$" by "$D^2(a, C_i)$" in the definition. i.e., use Euclidean distance.

2. For K-median, we always let $M = A$, so the centers can only be chosen from the input A.

While for K-means, we always let $M = \mathbb{R}^d$, so the centers can be points outside of A.

If we define cluster C_i as before, then it is equivalent to minimize

$$\sum_{j=1}^{K} \sum_{a\in C_i} D^2(a, C_i)$$

For K-means, we often consider $D = D_{l_2}$, so that we are minimizing

$$\sum_{a\in A} \min_{1\le i\le K} \|a - C_i\|^2 = \sum_{j=1}^{K} \sum_{a\in C_j} \|a - C_j\|^2$$

It is also called Euclidean K-means problem.

III. K-center. Given $A \subset M, K \ge 1$, find $C = \{C_1, \cdots, C_K\} \subseteq M$

that minimized

$$\max_{a\in A} \min_{1\le i\le K} D(a, C_i)$$

where C_i's are called centers.

the difference from K-median

Minimize the maximum distance from each item to its nearest cluster centers, while for K-median, minimize the total distance.

1.2 Lloyd's algorithm for K-means clustering

Assume $M = \mathbb{R}^d, A \subseteq M, A = \{a_1, \cdots, a_n\}$

Lemma(*). The centroid of A with respect to $D = D_{l_2{}^2}$ is given by

$$C(A) = \frac{1}{|A|}\sum_{i=1}^{n} a_i$$

More precisely, for any $x \in \mathbb{R}^d$,

$$\sum_i \|a_i - x\|^2 = \sum_i \|a_i - C(A)\|^2 + |A| \cdot \|C(A) - x\|^2$$

Proof:

$$\begin{aligned}\sum_i \|a_i - x\|^2 &= \sum_i \|a_i - C(A) + C(A) - x\|^2 \\ &= \sum_i \|a_i - C(A)\|^2 + 2(C(A) - x) \cdot \sum_i (a_i - C(A)) + |A| \cdot \|C(A) - x\|^2\end{aligned}$$

By definition of $C(A)$, $\sum_i (C(A) - a_i) = 0$

Thus. $\sum_i \|a_i - x\|^2 = \sum_i \|a_i - C(A)\|^2 + |A| \cdot \|C(A) - x\|^2$. □

Corollary (推论): Let $\{a_1, \cdots, a_n\}$ be a set of points. The sum of squared distances of the a_i to a point x is minimized when x is the centroid, i.e., $x = \frac{1}{|A|}\sum_i a_i$

Proof: By the above lemma. □

I. The idea of Lloyd's algorithm:

1. Choose K initial centers $c_1, \cdots, c_K$.
2. Repeat the following until there is no improvement in the cost function.
   a) $C_i :=$ set of the points closest to c_i
   b) update c_i to be the centroid of C_i

The formal version:

K-means ($A \subseteq M = \mathbb{R}^d, K \in \mathbb{N}^{\geq 1}$)

1. Choose K initial centers $c_1, \cdots, c_k \in \mathbb{R}^d$ arbitrarily.
2. Repeat
   for $i = 1, \cdots, K$ do
   Let $C_i :=$ set of points in A closest to c_i.
   Called the assignment step

for $i = 1, \cdots, K$ do

Let $c_i := C(C_i) = \frac{1}{|C_i|} \cdot \sum_{a \in C_i} a$

Called the update step

3. Until convergence (e.g. when quality of solution no longer improves)

4. Return $c_1, \cdots, c_K$ and $C_1, \cdots, C_K$

II. Some properties

i.

> Lemma. The algorithm K-means always halt after a finite number of steps.

Proof: Prove that T = the sum of squares of distances of each point to its cluster center always improves (until convergence)

1) after update step:

by Lemma(*), for each cluster, the sum of squares of distances of each point to its cluster center improves, i.e, it becomes smaller.

2) after assignment step:

T also improves, as for each point p, the distance between p and its center is improving.

$\|p - C_{old}\| \to \|p - C_{new}\|$

The running time: $O(nKd \cdot R)$, where R is the number of assignment and update steps until convergence.

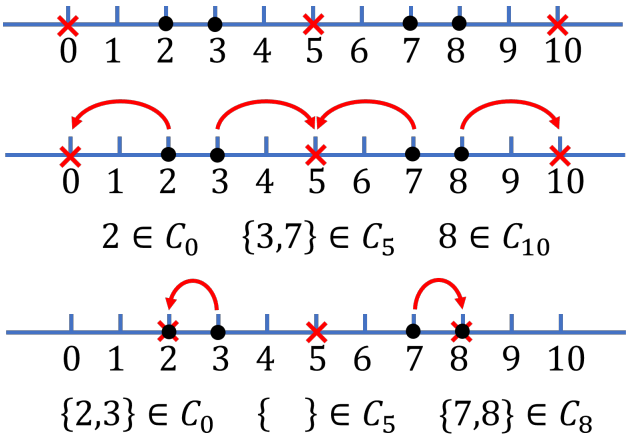
- Good in practice

- Worst-case (in theory) $R = 2^{\Omega(\sqrt{n})}$, super-polynomial.

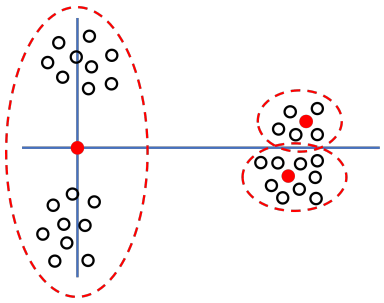
ii. Lloyd's algorithm can get stuck in arbitrary poor local minima.

Example 1. $K = 3$ $1-d$ data $\{2, 3, 7, 8\}$

start with center $\{0, 5, 10\}$

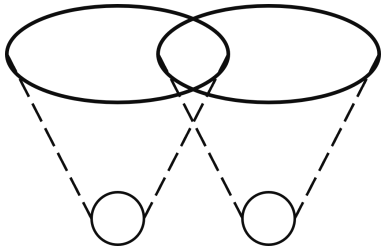


Example 2. $K = 3$ $2-d$ data



2 Spectral clustering

Make use of the spectrum(eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions.



2.1 Similarity Graphs

Def. We use graphs to represent the similarity matrix.

Graph notation

Let $G = (V, E)$ be a graph with vertex set $V = \{v_1, \ldots, v_n\}$. For weighted graph, each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$.

Weighted adjacency matrix $W = (w_{ij}), i, j = 1, \ldots, n$. For undirected graph, $w_{ij} = w_{ji}$. The degree of a vertex $v_i \in V$ is defined as $d_i = \sum_{j=1}^{n} w_{ij}$.

Thus, W- (weighted) Adjacency matrix 【对角线全 0】

$$\begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1n} \\ \ldots & \ldots & \ldots & \ldots \\ w_{n1} & \ldots & \ldots & w_{nn} \end{bmatrix}$$

Degree matrix D: a diagonal matrix with the degrees $d_1, \ldots, d_n$ on the diagonal

$$\begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ldots & \\ & & & d_n \end{bmatrix}$$

Given a set of data points $A = \{a_1, \ldots, a_n\}$ with pairwise similarities s_{ij} or distances d_{ij}, how to construct a graph?

Treat

i. ε-neighborhood graph. Connect all points whose pairwise distances are smaller than ε.

ii. k-nearest neighbor graphs. Connect vertex v_i with vertex v_j if v_j is among the k-nearest neighbors of v_i ($\rightarrow$ This leads to a directed graph)

① k-nearest neighbor graph: ignore the directions of the edges.

② mutual k-nearest neighbor graph: connect v_i and v_j if both $v_i \rightarrow v_j$ and $v_j \rightarrow v_i$ (both directions exist).

iii. Fully connected graph. Simply connect all points with weights defined by the similarities.

2.2 Graph Laplacians

I. The unnormalized graph Laplacian

Defined as $L = D - W$, where D is Degree Matrix, and W is Weighted Adjacency Matrix

Proposition 1

L satisfies the following properties:

1. For every vector $f \in \mathbb{R}^n$ we have $f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$ 【半正定】
2. L is symmetric and positive semi-definite 【特征值】
3. The smallest eigenvalue of L is 0, and the corresponding eigenvector is constant one vector $\mathbb{1}$. 【特征向量】
4. L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \cdots \leq \lambda_n$

Proof. 1.

$$\begin{aligned} f'Lf &= f'Df - f'Wf \\ &= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} \\ &= \frac{1}{2}(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2) \\ &= \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} f_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji} f_j^2) \\ &= \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(f_i - f_j)^2) \end{aligned}$$

注意：$\sum_{i=1}^{n} d_i f_i^2 = \sum_{j=1}^{n} d_j f_j^2$

2. As D and W are both symmetric, L is symmetric. From 1 we have $f^T Lf \geq 0$, thus L is positive semi-definite.

3. Consider vector $\mathbb{1} = (1, 1, \ldots, 1)$, we have the k-th entry of the vector $L\mathbb{1}$ equal to

$$\begin{aligned} \sum_{i=1}^{n} L_{ki} &= \sum_{i=1}^{n} D_{ki} - W_{ki} \\ &= D_{kk} - \sum_{i=1}^{n} W_{ki} \end{aligned}$$

Since D_{kk} is the degree of k-th vector, we have the above $= 0$. Thus, we have $L\mathbb{1} = 0\mathbb{1}$

Sine L is semi-definite, it has only non-negative eigenvalues, thus the smallest eigenvalue is 0, and the corresponding eigenvector is $\mathbb{1}$.

4. A direct consequence of $1, 2, 3$.

□

Proposition 2

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

Proof. ① case $k = 1$, the graph is connected.

Assume f is an eigenvector with eigenvalue 0, then we know

$0 = f'Lf = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$.

As w_{ij} are non-negative, the sum can only vanish(=0) if all terms $w_{ij}(f_i - f_j)^2$ vanish(=0).

Thus, if v_i and v_j are connected (i.e. $w_{ij} > 0$), then $f_i = f_j$.

$\rightarrow f$ needs to be constant for all vertices which can be connected by a path in the graph.

Moreover, as all vertices of a connected component in an undirected graph can be connected by a path, f needs to be constant on the whole connected component.

In a graph consisting of only one connected component, we thus only have the constant on vector $\mathbb{1}$ as eigenvector with eigenvalue 0, which obviously is the indicator vector of the connected component.

② case k connected components. 作业。

□

Normalized Laplacian 有兴趣可以看 paper

2.3 Spectral Clustering Algorithm

Input. Data points and similarity function

$\rightarrow$ Construct a similarity graph G. (use above mentioned method)

$\rightarrow$ Compute unnormalized Laplacian L.

$\rightarrow$ Compute the first k eigenvectors $\mu_1, \ldots, \mu_k$ of L.

$\rightarrow$ Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $\mu_1, \ldots, \mu_k$ as columns.

$\rightarrow$ For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.

$\rightarrow$ Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$

Output: cluster $A_1, \ldots, A_k$ with $A_i = \{j | y_j \in C_i\}$

Δ change the representation of the abstract data point x_i to $y_i \in \mathbb{R}^k$

2.4 Graph cut point of view

Restate the clustering problem: find a partition of the graph such that the edges between different groups have a very low weight, and the edges within a group have high weight.

Spectral clustering $\rightarrow$ an approximation to such graph partitioning problem.

Δ Mincut: minimize $cut(A_1, \ldots, A_k) := \frac{1}{2}\sum_{i=1}^{k} W(A_i, \overline{A_i})$

mincut in many cases separates on individual vertex from the rest of the graph $\rightarrow$ clusters should be reasonably large.

$\rightarrow$ RatioCut $(A_i, \ldots, A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{|A_i|}$

$\rightarrow$ NCut $(A_i, \ldots, A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{vol(A_i)}$

RatioCut use number of vertices $|A|$ as the size, while NCut use the weights of its edges $vol(A)$.

Solve RatioCut and NCut is NP-hard.

Relax NCut leads to normalized spectral clustering, while relaxing RatioCut leads to unnormalized spectral clustering.

i. Approximating RatioCut for $k = 2$

Objective:

$$min_{A \subset V}\ RatioCut(A, \overline{A}) \tag{1}$$

We define the vector $f = (f_1, \ldots, f_n)' \in \mathbb{R}^n$ with entries

$$f_i = \begin{cases} \sqrt{|\overline{A}|/|A|} & if\ v_i \in A \\ -\sqrt{|A|/|\overline{A}|} & if\ v_i \in \overline{A} \end{cases} \tag{2}$$

Rewrite the objective function using the unnormalized graph Laplacian,

$$\begin{aligned} f'Lf &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \\ &= \frac{1}{2}\sum_{i\in A, j\in\overline{A}} w_{ij}(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}})^2 + \frac{1}{2}\sum_{i\in\overline{A}, j\in A} w_{ij}(-\sqrt{\frac{|\overline{A}|}{|A|}} - \sqrt{\frac{|A|}{|\overline{A}|}})^2 \\ &= cut(A, \overline{A}) \cdot (\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2) \\ &= cut(A, \overline{A}) \cdot (\frac{|A|+|\overline{A}|}{|A|} + \frac{|A|+|\overline{A}|}{|\overline{A}|}) \\ &= |V| \cdot RatioCut(A, \overline{A}) \end{aligned}$$

【$\frac{|\overline{A}|}{|A|}+\frac{|A|}{|\overline{A}|}+2$ 与 $(\sqrt{\frac{|\overline{A}|}{|A|}}+\sqrt{\frac{|A|}{|\overline{A}|}})^2$ 相等】

Additionally, we have

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

【f is orthogonal to the constant one vector $\mathbb{1}$】

At the same time, f satisfies

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

putting all above together. $min_{A\subset C} RatioCut(A,\overline{A})$ can be equivalently rewritten as (*) $\min_{A\subset C, ||f||=\sqrt{n}} f'Lf$ subject to $f \perp \mathbb{1}$. f_i is defined in Eq(2).

This is still NP-hard.

Discard the discreteness condition, f_i takes arbitrary values in $\mathbb{R}$.

$$(*)\ min_{f\in\mathbb{R}^n} f'Lf\ subject\ to f\perp\mathbb{1}, ||f|| = \sqrt{n}$$

Rayleigh-Ritz theorem

Let $M \in \mathbb{R}^{n\times n}$ be a symmetric matrix with ordered eigenvalues $\lambda_1 \le \cdots \le \lambda_n$ and corresponding eigenvectors $v_1, \ldots, v_n$.

The minimum/minimizer of the Rayleigh quotient ($r(x) = \frac{x^T M x}{x^T x}$) is related to:

$$\begin{aligned}
\lambda_1 &= \min_{x\in\mathbb{R}^n} r(x)\\
v_1 &= argmin_{x\in\mathbb{R}^n} r(x)\\
\lambda_2 &= \min_{x\in\mathbb{R}^n} r(x)\ subject\ to\ v_1^T x = 0\\
v_2 &= argmin_{x\in\mathbb{R}^n} r(x)\ subject\ to\ v_1^T x = 0\\
\lambda_3 &= \min_{x\in\mathbb{R}^n} r(x)\ subject\ to\ v_1^T x = 0, v_2^T x = 0\\
v_3 &= argmin_{x\in\mathbb{R}^n} r(x)\ subject\ to\ v_1^T x = 0, v_2^T x = 0\\
&\ldots
\end{aligned}$$

Rayleigh-Ritz theorem states that (the smallest eigenvalue of L is 0 with eigenvalues $\mathbb{1}$) f is the eigenvector corresponding to the second smallest eigenvalue of L.

Affer we have f_i, we cluster them into two groups $C, \overline{C}$ by the k-means Algo.

$$\begin{cases} v_i \in A & if\ f_i \in C \\ V_i \notin A & if\ f_i \in \overline{C} \end{cases}$$

this is equivalent to the unnormalized spectral clustering Algo.

ii. Approximating RatioCut for arbitrary k. Given a partition of V into k sets $A_1, ..., A_k$, we define k indicator rectors $h_j = (h_{1,j}, ..., h_{n,j})'$ by

$$h_{i,j} = \begin{cases} 1/\sqrt{|A_j|} & if\ v_i \in A_j \\ 0 & otherwise \end{cases} \qquad (i = 1, ..., n; j = 1, ..., k)$$

Then we set the matrix $H \in \mathbb{R}^{n \times k}$ as the matrix containing those k indicator vectors as columns. Observe that the colunms in H are orthonormal to each other, thus $H^{'}H = I$.

$$\rightarrow h_i^{'} L h_i = \frac{cut(A_i, \overline{A_i})}{|A_i|}$$

Moreover

$$h_i^{'} L h_i = (H^{'} L H)_{ii}$$

Combining above

$$\begin{aligned} RatioCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} h_i{}' L h_i &= \sum_{i=1}^{k} (H^{'} L H)_{ii} \\ &= T_r(H^{'} L H) \end{aligned}$$

T_r denotes the trace of a matrix.

So the problem of minimizing $RatioCut(A_1, ..., A_k)$ can be rewritten as

$$\min_{A_1, ..., A_k} Tr(H^{'} L H) \quad subject \quad to \quad H^{'} H = I$$

This is the standard form of a trace minimization prblem.

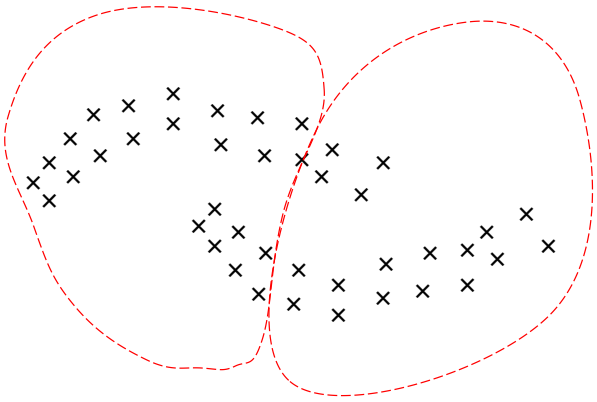
Rayleigh-Ritz theorem $\rightarrow$ solution is given by choosing H as the matrix which contains the first k eigencevtors of L as columns.

Then we use k-means on the rows of V.(第四章算法中的 V)

This leads to the general unnormalized spectral clustering algo.

3 Density-based clustering

why density-based clustering? Core idea: put data points in the same dense area as a cluster.



3.1 Preliminary

△$\varepsilon - neighborhood$. Given a data point p, its $\varepsilon - neighborhood$ is defined as follows:

$$N\varepsilon(p) = \{q \in D | d(p,q) \leq \varepsilon\}$$

i.e. the suprasphere with p as the center and $varepsilon$ as the radius.

超球体

△$Core - point$. Given a datapoint $p \in D$ and an integer M, if $|N_\varepsilon(p)| \geq M$, we call p the core-point under condition (ε, M)

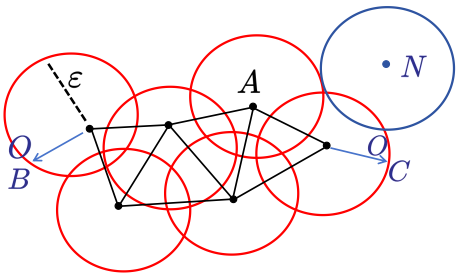
△$Border - point$. p is a boder-point if p is not a core-point, and p falls into the $\varepsilon - neighborhood$ of q.

△$Directly \quad reachable$. A datapoint q is directly reachable from p if $d(p,q) \leq \varepsilon$, i.e. q falls into the $\varepsilon - neighborhood$ of p.

△$Reachable$. A datapoint q is reachable from p, if there is a path $p_1, ..., p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i.

△$Outliers$. All data points not reachable from any other point are outliers ot noise points.

Example:



point A and all black points are core points. B/C are reachable from A, they are not core points. N is an outlier. B and C are density-connected

$\triangle \underline{Connectedness}$: Two data points p and q are density-connected if there is a point o such that both p and q are reachable from o. Density-connectedness is symmetric.

3.2 ★ Idea of DBSCAN.

1. Find the points in the $\varepsilon - neighborhood$ of every point, and identify the core points with more than M neighbors.

2. Find the connected components of core points on the neighbor graph, ignoring all non-core points.

3. Assign each non-core point to a nearby cluster, if the cluster is an $\varepsilon - neighborhood$, otherwise assign to outliers.

★ A. query-based algo.

Input: D, ε, M.

Step1: random select an un-visited node p.

Step2: Search $p^{'}$s $\varepsilon - neighborhood$.

if $|N \in (P) < M|$, mark p as outlier

else

use p as the core for the first cluster, mark it as visited.

Step3: find nodes that fall into the $\varepsilon - neighborhood$ of p, put them in the first cluster and mark them as visited.

Step4: For all nodes in the first cluster, execute step 2 and 3. until there is no node that can be added.

Random select an un-visited node and execute step 2-4. until all nodes are visited.

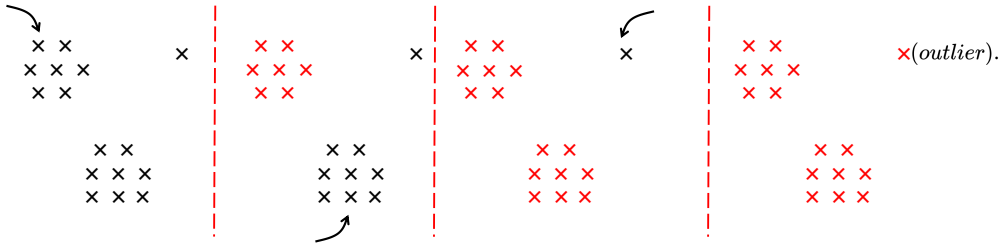
pros. → No need to specify k.

→ Not sensitive to outliers.

→ can cluster non-linear data.

cons. → sensitive to ε and M.

→ can not cluster high-dimension data.



3.3 DBSCAN revisited.

Fun fact

DBSCAN-KDD'96 original paper

DBSCAN-KDD'14 test of time award

DBSCAN revisited-SIGMOD'15 Best paper.

Yufei Tao.

Highlight of the DBSACAN Revisited paper.

i. The original paper claimed DBSCAN with $O(nlogn)$ running time. But it runs in $O(n^2)$ worst case.

ii. There exist a $O(nlogn)$ algo. in 2D space.

iii. For $d \geq 3$, the DBSCAN problem requires $\Omega(n^{4/3})$ time to solve.

I. Algorithm in 2D space.

Core idea: imposes an arbitrary grid T on the data space $\mathbb{R}^2$, where each cell of T is a $\frac{\varepsilon}{\sqrt{2}} \times \frac{\varepsilon}{\sqrt{2}}$ square.

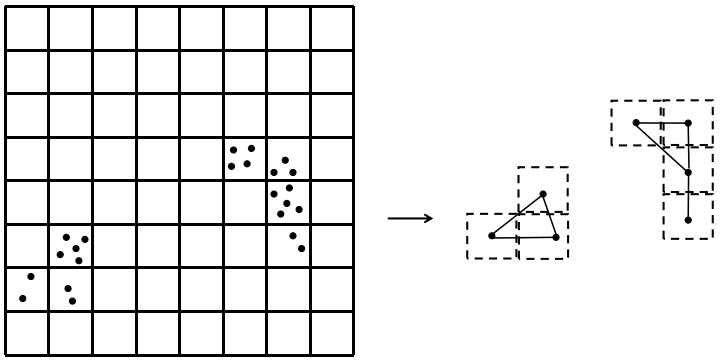
A cell is a core cell if $P(c)$(set of points of P covered by c.) contains at least on core point.

Construct a graph $G = (V, E)$, as follows:

- Each $v \in V$ corresponds to a distinct core cell in Score.
- Given two different cells $c_1, c_2 \in Score$, E contains and edge between c_1 and c_2 if there exist core points $p_1 \in p(c_1), p_2 \in p(c_2)$ such that $d(p_1, p_2) \leq \varepsilon$.

Next find all the connected components of G. Then —

The computation of G requires $O(n)$ nearest neighbor queries, each of which can be answered in $O(logn)$ time after building a voronoi diagram for each core cell. Total time cost $O(nlogn)$.

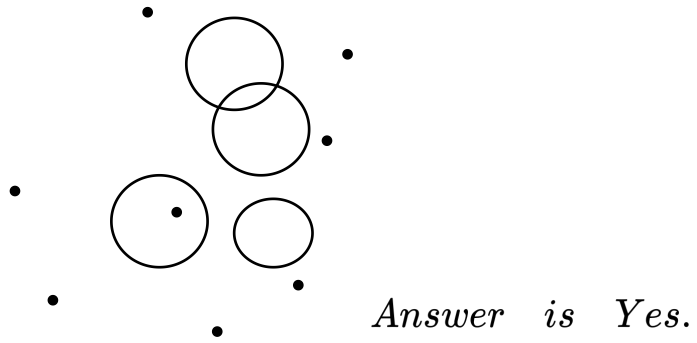


G

II. DBSCAN In ≥ 3 dimensions. i. Relevant geometric problems.

Unit-spherical emptiness checking **(USEC)** problem:

Let S_{pt} be a set of points, and S_{ball} be a set of balls with the same radius, all in data space $\mathbb{R}^d$, where the dimensionality d is a constant. The objective is to determine whether there is a point of S_{pt} that is covered by some ball in S_{ball}.



Hopcroft's problem:

Let S_{pt} be a set of points, S_{line} be a set of lines, all in data space $\mathbb{R}^2$(d=2). The goal is to determine whether there is a point in S_{pt} that lies on some lines of S_{line}.

The Hopcroft's problem can be settled in time slightly higher than $O(n^{4/3})$ time. A problem X is Hopcroft hard if an algorithm solving X in $O(n^{4/3})$ time implies an algorithm solving the Hopcroft's problem in $O(n^{4/3})$ time.

$\Leftrightarrow$ A lower bound $\Omega(n^{4/3})$ on the time of solving the Hopcroft's problem implies the same lower bound on X.

Lemma1:

The USEC problem in any dimensionality $d \geq 5$ is Hopcroft hard.

ii. Hardness of DBSCAN

Theorem:

$\triangle$ It is Hopcroft hard in any $d \geq 5$ for DBSCAN.
Namely, the problem requires $\Omega(n^{4/3})$ time to solve, unless the Hopcroft problem can be settled in $O(n^{4/3})$ time.

$\triangle$ when $d = 3$(and hence, $d = 4$), the problem requires $\Omega(n^{4/3})$ time to solve, unless the USEC problem can be settled in $O(n^{4/3})$ time.

Next we prove Theorem.

Lemma2:

For any dimensionality d, if we can solve the DBSCAN problem in $T(n)$ time, then we can solve the USEC problem in $T(n) + O(n)$ time.

Proof: Denote by $\mathcal{A}$ a DBSCAN algorithm in $\mathbb{R}^d$ that runs in $T(m)$ time on m points. Next, we describe an algorithm that deploys $\mathcal{A}$ as a black box to solve the USEC problem in $T(n)+O(n)$ time, where $n = |S_{pt}| + |S_{ball}|$.

Algorithm:

1. Obtain P, which is the union of S_{pt} and the set of the balls in S_{ball}
2. Set ε to the identical radius of the balls in S_{ball}
3. Run $\mathcal{A}$ to solve the DBSCAN algorithm on P with ε and $M = 1$
4. If any point in S_{pt} and any center of S_{ball} belong to the same cluster, then return Yes for the USEC problem. Otherwise return No.

The above algorithm runs in $O(n) + T(n)$. We next prove its correctness.

Case 1: We return Yes.

Yes means a point $p \in S_{pt}$ and the center q of some ball in S_{ball} have been placed in the same cluster(denoted by c), which means there exists a point $z \in c$ such that both p and q are density-reachable from z.

By setting $M = 1$, we ensure that all the points in P are core points. In general, if a core point p_1 is density-reachable from p_z(also a core point), then p_z is also density-reachable from p_1

This means z is density-reachable from $p, q$is d-reachable from z, q is d-reachable from p.

Then there is a sequence of points $p_1, p_2, ..., p_t \in P$ s.t.

1. $p_1 = p, p_t = q$,
2. $dist(p_i, p_{i+1}) \leq \epsilon$for each$i \in [1, t-1]$

Let k be the smallest $i \in [2, t]$ s.t. p_i is the center of a ball in S_{ball}. Note that k definitely exists because p_t is such a center. It thus follows that p_{k-1} is a point from S_{pt}, and p_{k-1} is covered by the ball in S_{ball} centered at p_k

Case 2: We return No.

Suppose on the contrary that a point $p \in S_{pt}$ is covered by a ball of S_{ball} centered at q. Thus, $dist(p, q) \leq \varepsilon$, namely, q is d-reachable from p. Then q must be in the cluster of p (recall that all points of p are core points). This contradicts the fact that we returned no.

This shows that in this case no point of S_{pt} is covered by any ball in S_{ball}.

Put Lemma 1 and 2, we have Theorem proved.

There is a approximation algo introduced in the paper.

4 Hierarchical Clustering

4.1 Basic Definitions

$A = \{a_1, \ldots, a_n\}$, $a_i \in M \subseteq \mathbb{R}^d$, D: distance function.

Def: A sequence of clustering $\mathbb{C}_n, \ldots, \mathbb{C}_1$ with $|\mathbb{C}_k| = k$ is called hierarchical clustering of A if for all $A \in \mathbb{C}_k$:

1. either $A \in \mathbb{C}_{k+1}$, or
2. $\exists B, C \in \mathbb{C}_{k+1} : \ A = B \cup C$ and $\mathbb{C}_k = \mathbb{C}_{k+1} \setminus \{B, C\} \cup \{A\}$.

Dendrogram (树状图):

A dendrogram on n nodes is a rooted binary tree $T = (V, E)$ with an index function $\mathcal{X} : V \setminus \{\text{leaves of } T\} \to \{1, \ldots, n\}$ such that:

1. $\forall v \neq w$, $\mathcal{X}(v) \neq \mathcal{X}(w)$,
2. $\mathcal{X}(\text{root}) = n$,
3. $\forall u, v$, if v is the parent of u, then $\mathcal{X}(v) > \mathcal{X}(u)$.

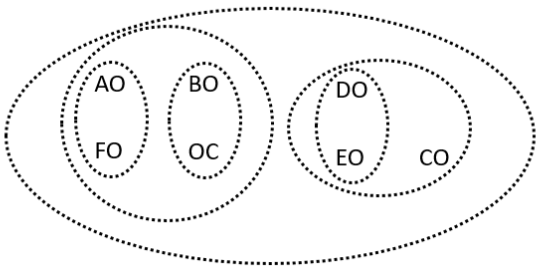
From hierarchical clustering to dendrogram.

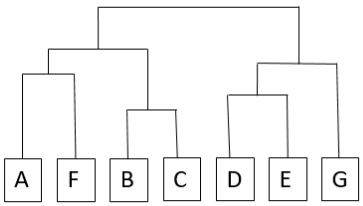
Construction of dendrogram:

1. Create leaf for each point $a \in A$.
2. Interior nodes correspond to union of clusters.
3. If k-th cluster is obtained by union of clusters B, C, create new node with index k and with children B, C.

Example:

层次聚类分为聚合型与分离性，可理解为自底向上和自顶向下





4.2 Agglomerative clustering(聚合型)

Basic idea:

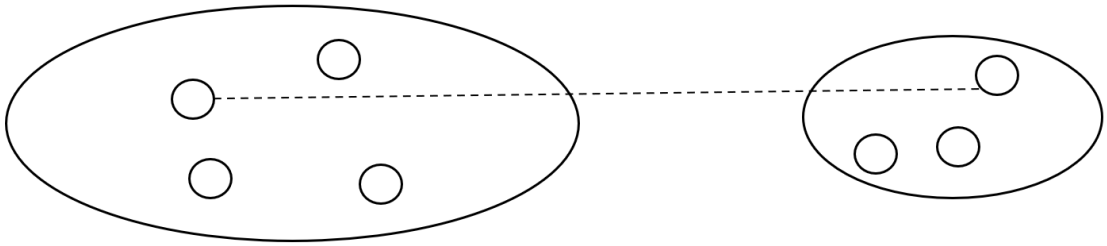
1. Start with n clusters C_i, $1 \leq i \leq n$, $C_i = \{a_i\}$.
2. In each step, replace two clusters C_i, C_j that are "closest" by their union $C_i \cup C_j$.

3. Until single cluster is left.

Two ways to measure the closeness.

I. Complete Linkage

Def: For $C_1, C_2 \subseteq M$, $D_{CL}(C_1, C_2) := \max_{x \in C_1, y \in C_2} D(x, y)$ called the complete linkage cost of C_1 and C_2.



Agglomerative Complete Linkage(A)

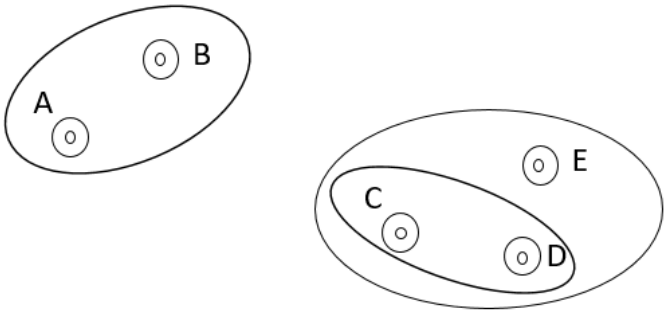
$C_n = \{\{a_i\} | a_i \in A\}$

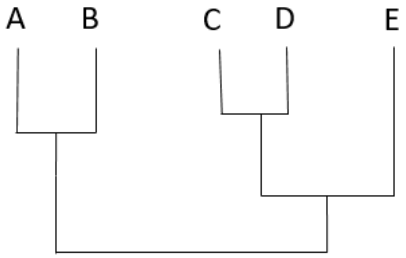
for $i = n - 1, ..., 1$ do

find distinct clusters $A, B \in C_{i+1}$ minimizing $D_{CL}(A, B)$

$C_i = (C_{i+1} \backslash \{A, B\}) \cup \{A \cup B\}$

return $C_1, ...C_n$





Analysis:

Let $diam^D(S) := \max_{x,y \in S} D(x, y)$ be diameter of S

$cost^D diam(c) := \max_{1 \le i \le k} diam^D(c_i))$ be diameter cost

$opt_k^{diam}(A) := \min_{|\mathbb{C}|=k} cost_{diam}^D(\mathbb{C})$

Theorem: The algorithm Agglomerative Complete Linkage computes a k-clustering $\mathbb{C}_k$ with

$cost_{diam}^D(\mathbb{C}_k) \le O_d(\log k) \cdot opt_k^{diam}(A)$

for each $k \le |A|$

II. Single Linkage

Def: For $C_1, C_2 \subseteq M$

$D_{SL}(C_1, C_2) = \min_{x \in C_1, y \in C_2} D(x, y)$

is called single linkage cost of C_1, C_2

Algorithm-AggSL(A)

$\mathbb{C}_n = \{\{a_i\} | a_i \in A\}$

for $i = n-1, ..., 1$ do

find distinct clusters $A, B \in \mathbb{C}_{i+1}$ minimizing $D_{SL}(A < B)$

$\mathbb{C}_i = (C_{i+1} \backslash \{A, B\}) \cup \{A \cup B\}$

return $\mathbb{C}_1, ..., \mathbb{C}_n$

Analysis:

Theorem: AggSL(A) find a partition $\mathbb{C}_k = \{C_1, ..., C_k\}$ of A that maximizes

$\min_{x \in C_i, y \in C_j, i \ne j} D(x, y)$

i.e. a partition that maximizes the minimum distance between points in different clusters.

It holds for any $k \le n$

4.3 Divisive clustering

Initialization: All points stay in one cluster.

Iteration: Select a cluster and split it into two sub-clusters.

Until each leaf contains only one point.

4.4 HC based on Gonzalez algorithm

Let $D(A, C) = \max_{a \in A} \min_{c \in C} D(a, c), C \subseteq A$ (k-center objective)

Full-farthest-first traversal(A,D)

1. Choose a center c_1 arbitrarily from A, set $C^1 = \{c_1\}$
2. For $i = 2, ..., |A|$ do
   - Set $R_i := \max_{a \in A} D(a, C^{i-1})$
   - Choose a c_i as a point with $D(c_i, C^{i-1}) = R_i$
   - Set $C^i = C^{i-1} \cup \{c_i\}$
3. Return $c_1, ..., c_{|A|}$ and $R_2, ..., R_{|A|}$

Lemma: The above algo computes $\{c_i\}$ and $\{R_i\}$ s.t. $R_k \leq 2 \cdot D(A, C_k^*)$ for all $k = 2, ..., |A|$, where C_k^* is the optimal solution for (A, k)

What to do next?

Define levels for each point: Let $R = R_2$(longest)

$L_0 = \{c_1\}, L_j = \{c_i | R_i \in (\frac{R}{2^j}, \frac{R}{2^{j+1}}]\}$ for all $j \geq 1$

Define $L(x) = j$ where $x \in L_j$

Define the parent for each point:

parent$(c_i) = argmin\{D(x_i, y) | y \in \bigcup_{j=0}^{L(x_i)-1} L_j\}$ for $i = 2, ..., |A|$

Lemma: $\forall x \in A$
$D(x, \bigcup_{j'=0}^{j} L_{j'}) \leq \frac{R}{2^j}$, and $D(x, parent(x) \leq \frac{R}{2^{L(x)-1}}$

Hierarchical-k-center(A,D)

1. Compute $x_1, ..., x_{|A|}$ and $R_2, ..., R_{|A|}$ by calling full-farthest-first-traversal(A,D)
2. Set $R := R_2$ and set $L_0 = \{x_1\}, L(x_1) = 0$
3. For all $j \geq 1$, set $L_j = \{x_i | R_i \in (\frac{R}{2^j}, \frac{R}{2^{j-1}})\}$ and $L(x_i) = j$ iff $x_i \in L_j$
4. For all $x \in A$, set $parent(x) = argmin\{D(x, y) | y \in \bigcup_{j=0}^{L(x)-1} L_j\}$

5. Set $c_i = \{x_i\}$ and $\mathcal{H}_{|A|} = \{c_1, ..., c_{|A|}\}$

6. For $k = |A| - 1$ to 1 do (Notice x_{k+1} is the center of the cluster c_{k+1} to be reassigned)

    - Let $x_p = parent(x_{k+1})$ be the parent of x_{k+1}, notice $x_p \in c_p$
    - Set $c_p = c_p \cup c_k$ and $\mathcal{H}_k = \{c_i | i \in [k]\}$

7. Return $\mathcal{H}_1, ..., \mathcal{H}_{|A|}$

Theorem: The algorithm computes a hierarchical clustering $\mathcal{H}_1, ..., \mathcal{H}_{|A|}$ s.t. for all $k \in [|A|]$, and optimum $C_k^*, D(A, \mathcal{H}_k) \leq \delta D(A, C_k^*)$