Chapter 14

Clustering

Clustering

Unsupervised learning

- Requires data, but no labels
- Detect patterns e.g. in
  - Group emails or search results
  - Customer shopping patterns
  - Regions of images
- Useful when don't know what you're looking for
- But: can get gibberish

Clustering examples

Image segmentation

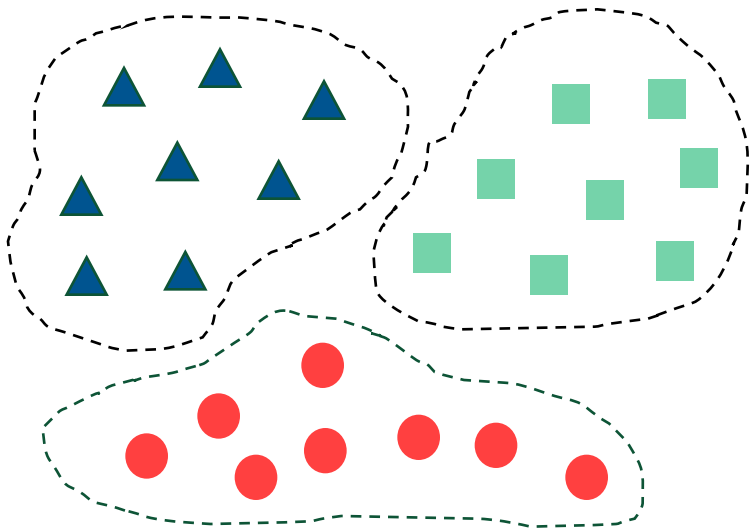
Goal: Break up the image into meaningful or perceptually similar regions

Outlines

- Clustering Problem
- Performance Measure
- Distance Calculation
- Prototype Clustering
- Density Clustering
- Hierarchical Clustering

Clustering Problem

- It is mostly studied and widely used in *unsupervised learning*.
- **Goal:** partitions the dataset into several disjoint subsets (clusters).
- Clustering can be used by itself to identify the inherent structure of data, while it can also serve as a pre-processing technique for other learning tasks such as classification.



Clustering Problem

Formalization

Given dataset $D = \{x_1, x_2, \cdots, x_m\}$ containing m unlabeled samples, where each sample $x_i = (x_{i1}; x_{i2}; \cdots; x_{in})$ is a n-dimensional vector. Then, a clustering algorithm partitions the data set D into k clusters $\{C_l | l = 1, 2, ..., k\}$, where $C_{l'} \bigcap_{l' \neq l} C_l = \phi$ and $D = \bigcup_{l=1}^{k} C_l$.

Accordingly, we denote $\lambda_j \in \{1, 2, \cdots, k\}$ as the *cluster label* of sample x_j (*i.e.*, $x_j \in C_{\lambda_j}$). Then the clustering result can be represented as a cluster label vector $\lambda = \{\lambda_1; \lambda_2; \cdots; \lambda_m\}$ with m elements.

Outlines

- Clustering Problem
- **Performance Measure**
- Distance Calculation
- Prototype Clustering
- Density Clustering
- Hierarchical Clustering

Performance Measure

- Performance measure for clustering are also called *validity indices*.

- Intuitively, we wish *things of a kind come together*; that is, samples in the same cluster should be as similar as possible while samples from different clusters should be as different as possible. In other words, we seek clusters with high *intra-cluster similarity* and low *inter-cluster similarity*.

Performance Measure

Performance measure for clustering:

- **External index**
  - Comparing the clustering result against a *reference model.*
- **Internal index**
  - Evaluating the clustering result without using any reference model

Performance Measure–External index

Given a data set $D = \{x_1, x_2, \ldots, x_m\}$, suppose a clustering algorithm produces the clusters $C = \{C_1, C_2, \ldots, C_k\}$, and a reference model gives the clusters $C^* = \{C_1^*, C_2^*, \ldots, C_s^*\}$. Accordingly, let λ and λ^* denote the clustering label vectors of C and C^*.

For each pair of samples we define the following four terms

$$a = |SS|, SS = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$$

$$b = |SD|, SD = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$$

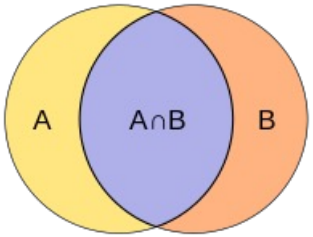
$$c = |DS|, DS = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$$

$$d = |DD|, DD = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$$

Performance Measure–External index

- Jaccard Coefficient, JC (雅卡尔系数)

交并比



$$JC = \frac{a}{a+b+c}$$

- Fowlkes and Mallows Index, FMI

准确率和召回率的几何平均数

$$FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}$$

- Rand Index, RI (兰德系数)

$$\mathrm{RI} = \frac{2(a+d)}{m(m-1)}$$

values between [0,1], the larger the better

Performance Measure–External index

```
>>> from sklearn.metrics.cluster import fowlkes_mallows_score
>>> fowlkes_mallows_score([0, 0, 1, 1], [0, 0, 1, 1])
1.0
>>> fowlkes_mallows_score([0, 0, 1, 1], [1, 1, 0, 0])
1.0
```

```
>>> fowlkes_mallows_score([0, 0, 0, 0], [0, 1, 2, 3])
0.0
```

Performance Measure–Internal index

- Given the generated clusters $C = \{C_1, C_2, \ldots, C_k\}$, define the following four terms:
  - The average distance between the samples in cluster C

$$avg(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \le i \le j \le |C|} dist(x_i, x_j)$$

  - The largest distance between samples in cluster C

$$diam(C) = max_{1 \le i \le j \le |C|} dist(x_i, x_j)$$

  - The distance between two nearest samples in clusters C_i and C_j

$$d_{min}(C) = min_{x_i \in C_i, x_j \in C_j} dist(x_i, x_j)$$

  - The distance between the centroids of clusters C_i and C_j

$$d_{cen}(C) = dist(\mu_i, \mu_j)$$

Performance Measure–Internal index

- Davies-Bouldin Index, DBI

$$DBI = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{avg(C_i)+avg(C_j)}{d_{cen}(C_i,C_j)}\right)$$

The smaller the better.

- Dunn Index, DI

$$DI = \min_{1\leq i\leq k}\left\{\min_{j\neq i}\left(\frac{d_{min}(C_i,C_j)}{\max_{1\leq l\leq k} diam(C_l)}\right)\right\}$$

The bigger the better.

Outlines

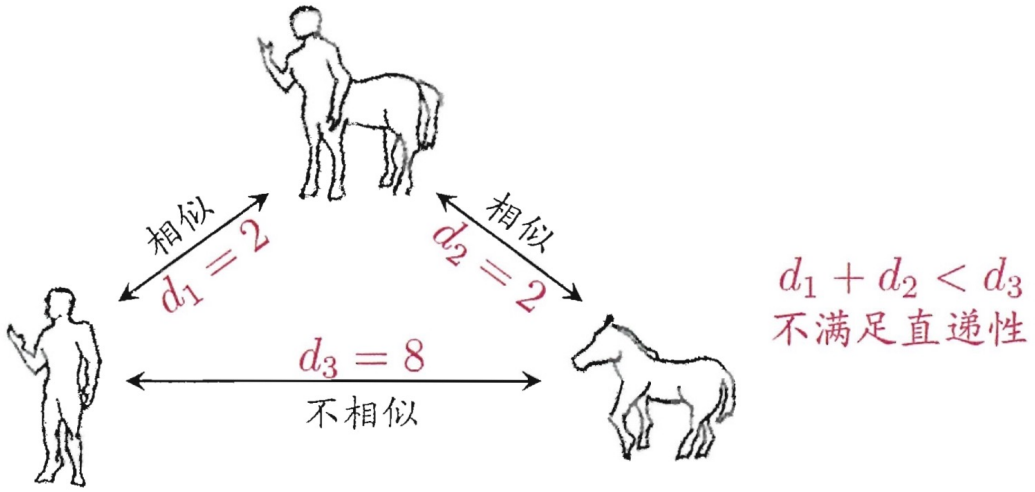
- Clustering Problem
- Performance Measure
- **Distance Calculation**
- Prototype Clustering
- Density Clustering
- Hierarchical Clustering

Distance Calculation

The axioms (公理) of *distance metric*

- Non-negativity (非负性)： $dist(x_i, x_j) \geq 0$
- Identity of indiscernible (不可分者同一性原理)：

  $dist(x_i, x_j) = 0$ if and only if $x_i = x_j$
- Symmetry (对称性)： $dist(x_i, x_j) = dist(x_j, x_i)$
- Subadditivity （ 直递性)： $dist(x_i, x_j) \leq dist(x_i, x_k) + dist(x_k, x_j)$



Distance Calculation

- A commonly used distance metric:

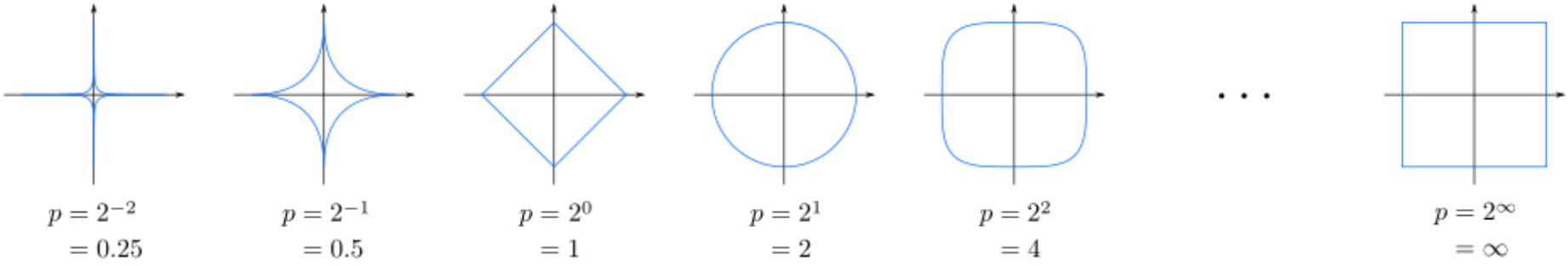
  Minkowski distance:

$$dist(x_i, x_j) = \left(\sum_{u=1}^{n} |x_{iu} - x_{ju}|^p \right)^{\frac{1}{p}}$$

  p=2: Euclidean distance.

  p=1: Manhattan distance.

The figure shows unit circles (all points are at the unit distance from the center) with various values of p



Outlines

- Clustering Problem
- Performance Measure
- Distance Calculation
- **Prototype Clustering**
- Density Clustering
- Hierarchical Clustering

Prototype Clustering

- Prototype Clustering

Also known as *prototype-based clustering*, assumes the clustering structure can be represented by as set of prototypes.

- Algorithm:

Typically, such algorithms start with some initial prototypes, and then iteratively update and optimize the prototypes.

- Next, we discuss several well-known prototype-based clustering algorithms.
  - K-means Clustering
  - Learning Vector Quantization (supervised)
  - Mixture-of-Gaussian Clustering

Prototype Clustering – *k*-means Clustering

Given a data set $D = \{x_1, x_2, \cdots, x_m\}$, the *k*-means algorithm minimizes the squared error of clusters $C = \{C_1, C_2, \cdots, C_k\}$:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2$$

where μ_i is the mean vector of cluster C_i.

Intuitively, E represents the closeness between the mean vector of a cluster and the samples within that cluster, where a smaller E indicates higher intra-cluster similarity.

Prototype Clustering – *k*-means Clustering

Given a data set $D = \{x_1, x_2, \cdots, x_m\}$, the *k*-means algorithm minimizes the squared error of clusters $C = \{C_1, C_2, \cdots, C_k\}$:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2$$

where μ_i is the mean vector of cluster C_i.

Intuitively, E represents the closeness between the mean vector of a cluster and the samples within that cluster, where a smaller E indicates higher intra-cluster similarity.

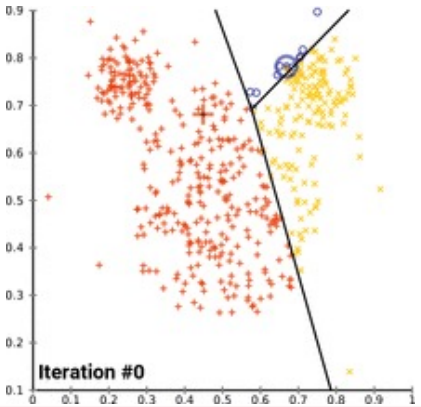
- Algorithm (iterative optimization)

  initializes the mean vectors of clusters

  repeat

  1. (update) the clusters
  2. calculate the mean vectors

  until clusters do not change



Prototype Clustering – *k*-means Clustering

Algorithm 9.1 k-means Clustering.

Input: Data set $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
Number of clusters k.

Process:

1: Randomly select k samples as the initial mean vectors $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\}$;
2: **repeat**
3: $C_i = \varnothing (1 \leqslant i \leqslant k)$;
4: **for** $j = 1, 2, \ldots, m$ **do**
5: Compute the distance between sample $\boldsymbol{x}_j$ and each mean vector $\boldsymbol{\mu}_i (1 \leqslant i \leqslant k)$: $d_{ji} = \|\boldsymbol{x}_j - \boldsymbol{\mu}_i\|_2$;
6: According to the nearest mean vector, decide the cluster label of $\boldsymbol{x}_j$: $\lambda_j = \arg\min_{i \in \{1,2,\ldots,k\}} d_{ji}$;
7: Move $\boldsymbol{x}_j$ to the corresponding cluster: $C_{\lambda_j} = C_{\lambda_j} \cup \{\boldsymbol{x}_j\}$;
8: **end for**
9: **for** $i = 1, 2, \ldots, k$ **do**
10: Compute the updated mean vectors: $\boldsymbol{\mu}'_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$;
11: **if** $\boldsymbol{\mu}'_i \neq \boldsymbol{\mu}_i$ **then**
12: Update the current mean vector $\boldsymbol{\mu}_i$ to $\boldsymbol{\mu}'_i$;
13: **else**
14: Leave the current mean vector unchanged.
15: **end if**
16: **end for**
17: **until** All mean vectors remain unchanged

Output: Clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.

Properties of *k*-means algorithm

- Guaranteed to converge in a finite number of iterations

- Running time per iteration:
  - 1. Assign data points to closest cluster center *O(kN)*
  - 2. Change the cluster center to the average of its assigned points *O(N)*

k-means Convergence

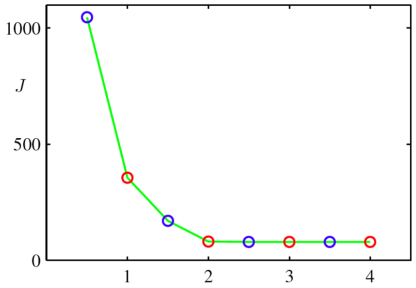
Objective

$$\min_{\mu}\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix μ, optimize C:

$$\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 = \min_{c} \sum_{i}^{n} \left|x_i - \mu_{x_i}\right|^2$$

Step 1 of kmeans



2. Fix C, optimize μ:

$$\min_{\mu} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

   – Take partial derivative of μ_i and set to zero, we have

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Step 2 of kmeans

Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

[Slide from Alan Fern]

Example: *k*-means for segmentation

K=2

Goal of Segmentation is to partition an image into regions each of which has reasonably homogenous visual appearance.

Original

Example: *k*-means for segmentation

K=2

K=3

Original

Example: *k*-means for segmentation

Prototype Clustering – *k*-means Clustering

- An example for *k*-means algorithm

We take the watermelon data set in the following table as an example to demonstrate the *k*-means algorithm. For ease of discussion, let x_i represent the sample with the ID i

ID	density	sugar	ID	density	sugar	ID	density	sugar
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

Prototype Clustering – *k*-means Clustering

- An example for *k*-means algorithm

Suppose we set k =3, then the algorithm randomly picks up three samples x_6, x_{12}, x_{24} as the initial mean vectors, that is, $\mu_1 = (0.403; 0.237), \mu_2 = (0.343; 0.099), \mu_3 = (0.478; 0.437)$

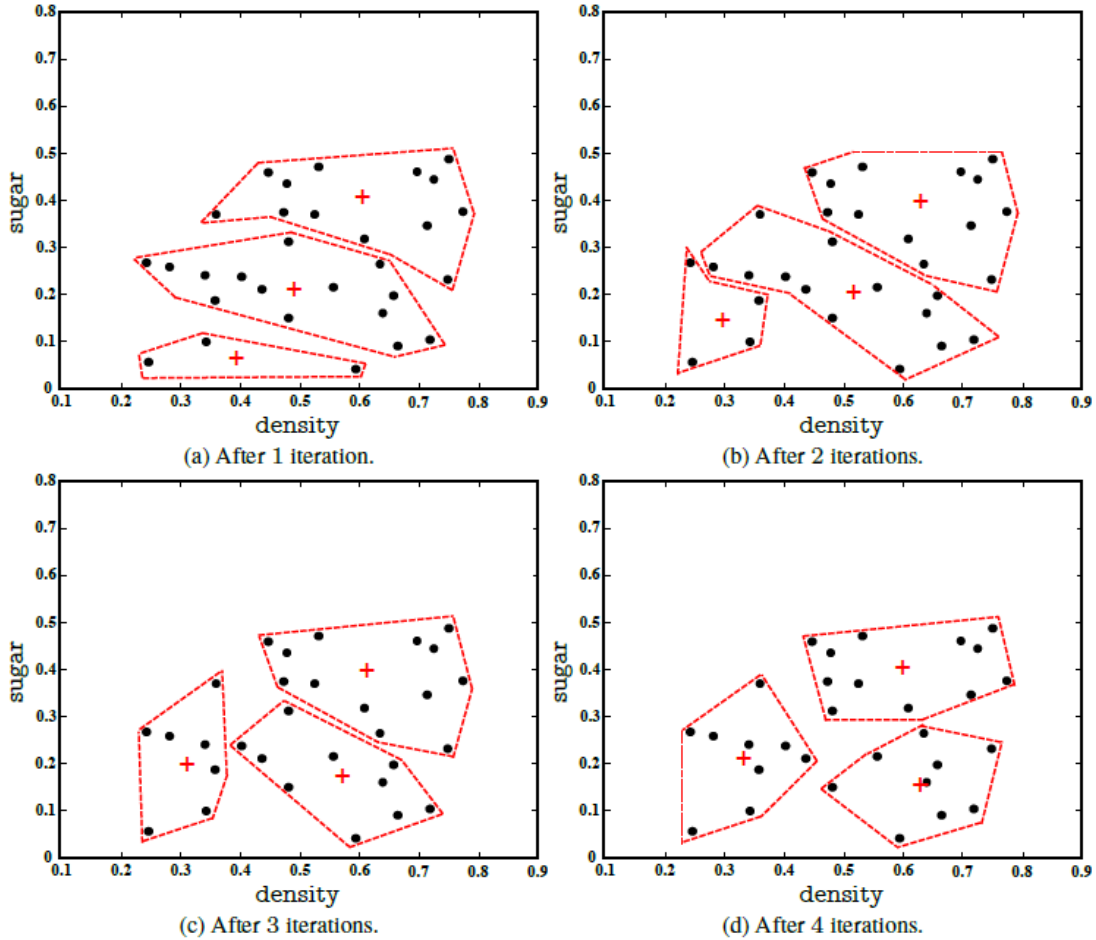
Then, for the sample $x_1 = (0.697; 0.460)$, its distances to the three current mean vectors μ_1, μ_2, μ_3 are 0.369, 0.506, and 0.220, respectively. Thus x_1 is assigned to cluster C_3. Similarly, we evaluate all samples in the data set and find the following cluster assignments:

$$C_1 = \{x_3, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{13}, x_{14}, x_{17}, x_{18}, x_{19}, x_{20}, x_{23}\}$$
$$C_2 = \{x_{11}, x_{12}, x_{16}\}$$
$$C_3 = \{x_1, x_2, x_4, x_{15}, x_{21}, x_{22}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}\}$$

$$\mu_1' = (0.493; 0.207), \mu_2' = (0.394; 0.066), \mu_3' = (0.602; 0.396)$$

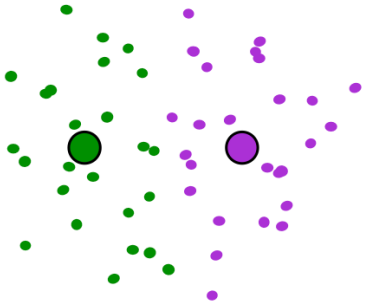
Prototype Clustering – *k*-means Clustering

- Results of the *k*-means algorithm

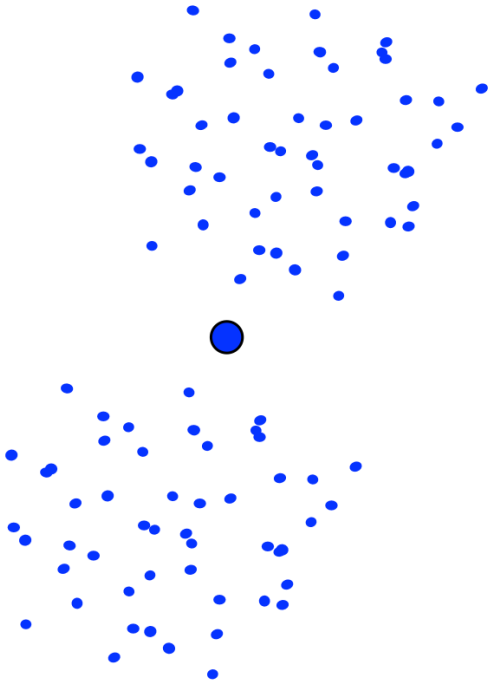


k-Means Getting Stuck

A local optimum:



Would be better to have one cluster here

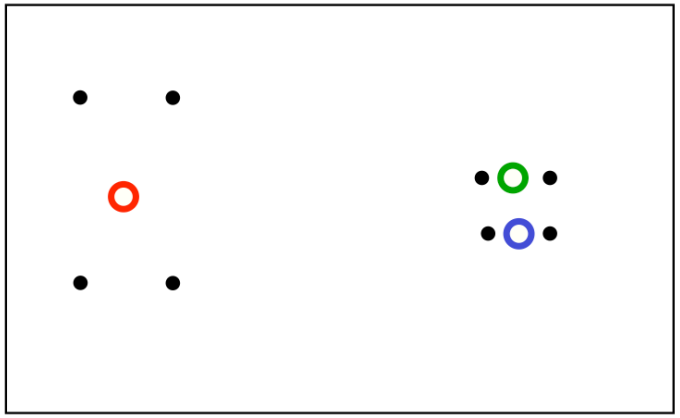


… and two clusters here

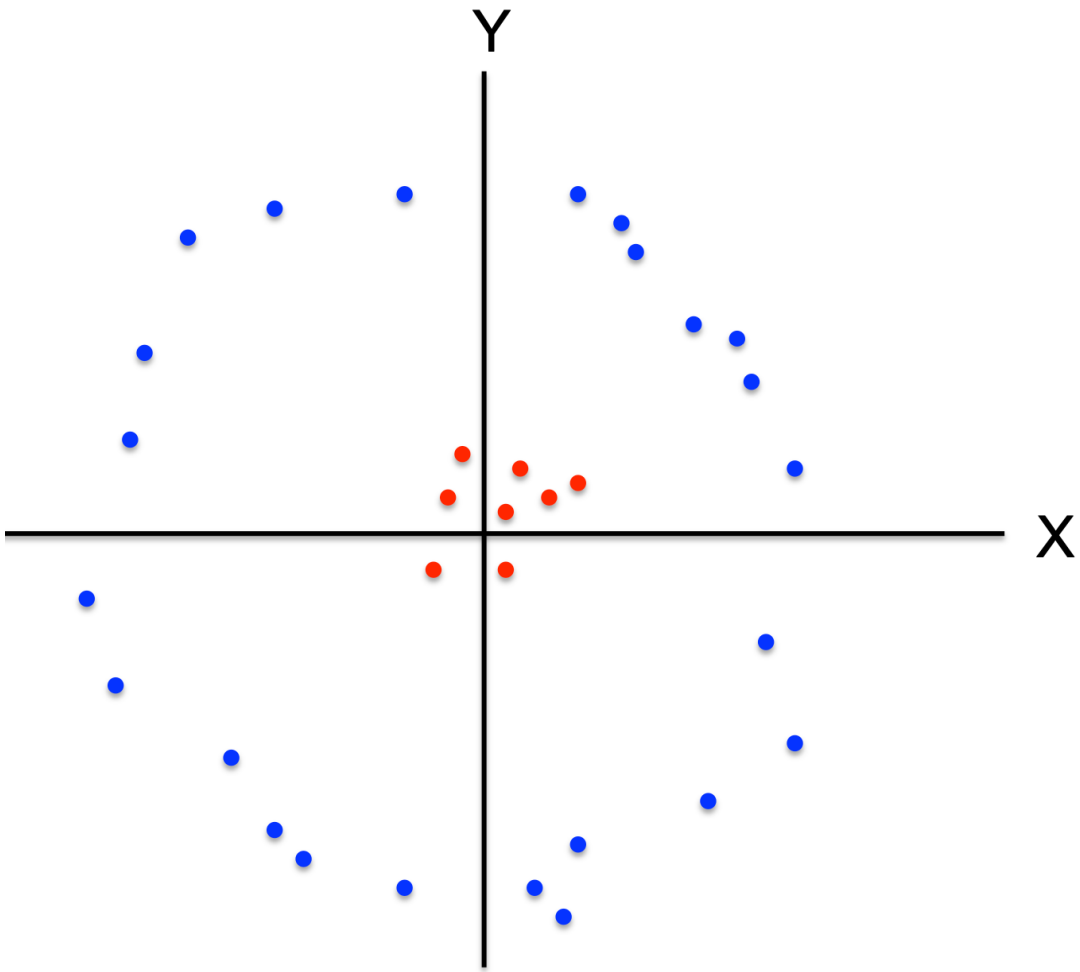
Local Minima

- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points
- We could try non-local split-and-merge moves:
  - Simultaneously merge two nearby clusters
  - and split a big cluster into two

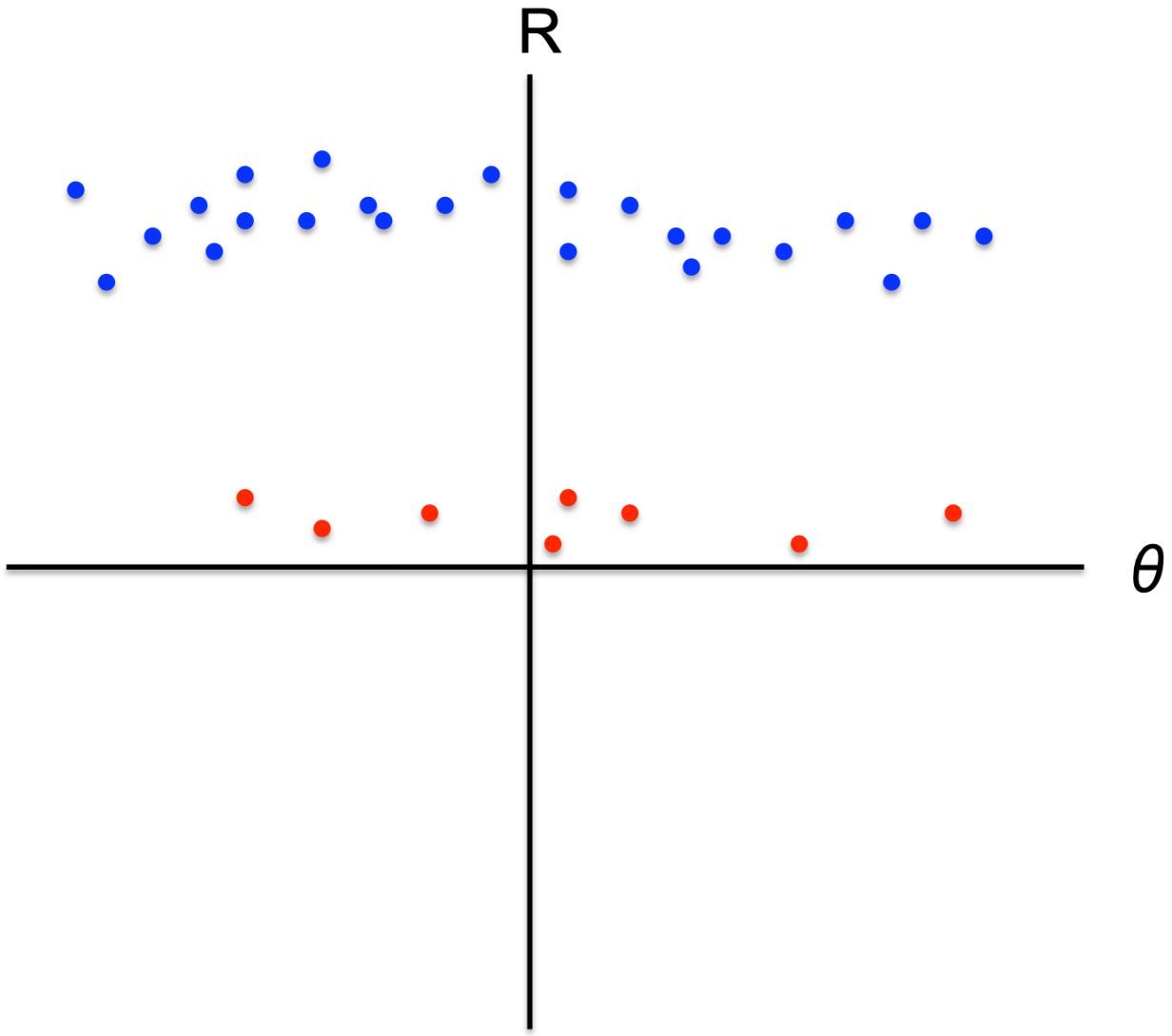
A bad local optimum



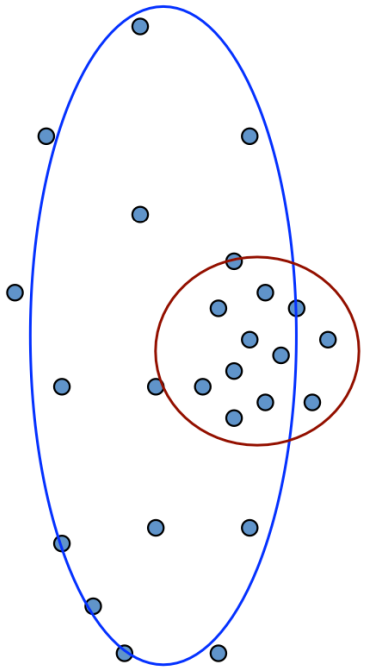
k-means not able to properly cluster



Changing the features (distance function) can help



Reconsidering "hard assignments"?



- Clusters may overlap
- Some clusters may be "wider" than others
- Distances can be deceiving!

Prototype Clustering – Learning Vector Quantization

Learning Vector Quantization, LVQ

Unlike typical clustering algorithm, LVQ assumes data samples are labeled, and the clustering process is assisted by supervised information.

Given a data set $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$, LVQ aims to learn a set of n-dimensional prototype vectors $\{p_1, p_2, \cdots, p_q\}$ where each prototype vector represents one cluster.

Prototype Clustering – Learning Vector Quantization

Algorithm 9.2 Learning Vector Quantization.

Input: Training set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Number of prototype vectors q;
Initial labels of prototype vectors $\{t_1, t_2, \ldots, t_q\}$;
Learning rate η.

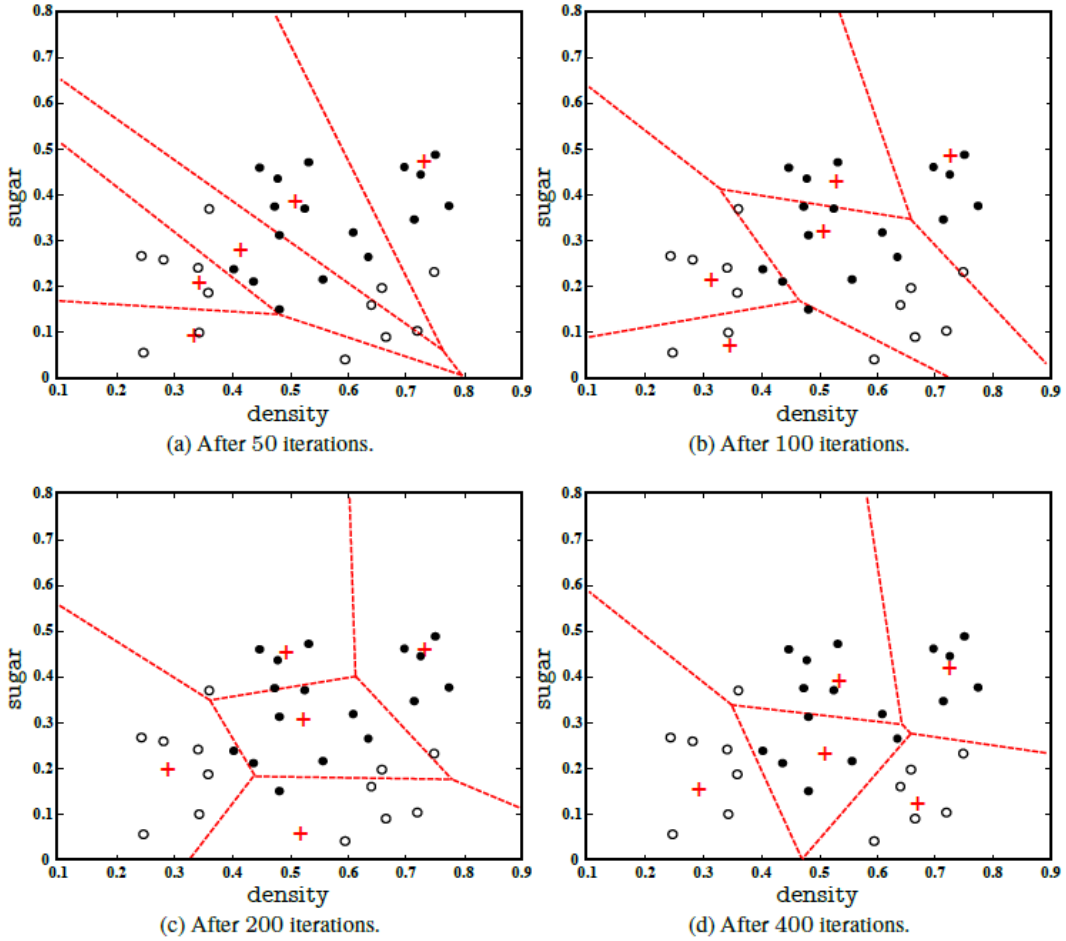
Process:

1: Initialize a set of prototype vectors $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_q\}$;
2: **repeat**
3: Randomly pickup a sample $(\boldsymbol{x}_j, y_j)$ from the data set D;
4: Compute the distance between $\boldsymbol{x}_j$ and $\boldsymbol{p}_i (1 \leqslant i \leqslant q)$: $d_{ji} = \|\boldsymbol{x}_j - \boldsymbol{p}_i\|_2$;
5: Find the nearest prototype vector p_{i^*} for $\boldsymbol{x}_j$, where $i^* = \arg\min_{i \in \{1,2,\ldots,q\}} d_{ji}$;
6: **if** $y_j = t_{i^*}$ **then**
7: $\boldsymbol{p}' = \boldsymbol{p}_{i^*} + \eta \cdot (\boldsymbol{x}_j - \boldsymbol{p}_{i^*})$;
8: **else**
9: $\boldsymbol{p}' = \boldsymbol{p}_{i^*} - \eta \cdot (\boldsymbol{x}_j - \boldsymbol{p}_{i^*})$;
10: **end if**
11: Update the prototype vector $\boldsymbol{p}_{i^*}$ to $\boldsymbol{p}'$.
12: **until** The termination condition is met

Output: Prototype vectors $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_q\}$.

Prototype Clustering – Learning Vector Quantization

- Clustering results



(a) After 50 iterations.

(b) After 100 iterations.

(c) After 200 iterations.

(d) After 400 iterations.

Prototype Clustering – Mixture-of-Gaussian Clustering

Unlike k-means and LVQ, Mixture-of-Gaussian clustering does not use prototype vectors but probabilistic models to represent clustering structures.

- Definition of multivariate Gaussian distribution

  For a random vector x in an n-dimensional sample space $\mathcal{X}$,

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

  where μ is an n-dimensional mean vector and Σ is an $n \times n$ covariance matrix. We write the probability density function as $p(x|\mu, \Sigma)$

Prototype Clustering – Mixture-of-Gaussian Clustering

- Definition of the Mixture-of-Gaussian distribution

$$p_M(x) = \sum_{i=1}^{k} \alpha_i p(x|\mu_i, \Sigma_i)$$

which consists of k mixture components and each corresponds to a Gaussian distribution. μ_i and Σ_i are the parameters of the ith mixture component, and $\alpha_i > 0$ are the corresponding *mixture coefficients*, where $\sum_{i=1}^{k} \alpha_i = 1$

Prototype Clustering – Mixture-of-Gaussian Clustering

- Suppose that the samples are generated from a Mixture-of-Gaussian distribution:

  Firstly, select the Gaussian mixture components using the prior distribution defined by $\alpha_1, \alpha_2, \cdots, \alpha_k$, where α_i is the probability of selecting the ith mixture component

  Then, generate samples by sampling from the probability density functions of the selected mixture components.

Prototype Clustering – Mixture-of-Gaussian Clustering

- Optimization of the model parameters: maximum the likelihood

$$LL(D) = \ln\left(\prod_{j=1}^{m} p_M(x_j)\right)$$

$$= \sum_{j=1}^{m} \ln\left(\sum_{i=1}^{k} \alpha_i \cdot p\left(x_j|\mu_i, \Sigma_i\right)\right)$$

Let:

$$\frac{\partial LL(D)}{\partial \mu_i} = 0 \quad \Rightarrow \quad \mu_i = \frac{\sum_{j=1}^{m} \gamma_{ji} x_j}{\sum_{j=1}^{m} \gamma_{ji}}$$

Mixture-of-Gaussian Clustering–Optimization (Continued)

Let:

$$\frac{\partial LL(D)}{\partial \Sigma_i} = 0 \Rightarrow \Sigma_i = \frac{\sum_{j=1}^{m} \gamma_{ji}(x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^{m} \gamma_{ji}}$$

Lagrange multiplier: $\Rightarrow \alpha_i = \frac{1}{m} \sum_{j=1}^{m} \gamma_{ji}$

Mixture-of-Gaussian Clustering

Algorithm 9.3 Mixture-of-Gaussian Clustering.

Input: Data set $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
Number of Gaussian mixture components k.

Process:

Initialize the parameters $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$ of the Mixture-of-Gaussian distribution;
repeat
  for $j = 1, 2, \ldots, m$ **do**
    According to (9.30), compute the posterior probabilities that $\boldsymbol{x}_j$ is generated by each Gaussian mixture component, i.e., $\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \boldsymbol{x}_j)(1 \leqslant i \leqslant k)$;
  end for
  for $i = 1, 2, \ldots, k$ **do**
    Compute the updated mean vector: $\boldsymbol{\mu}_i' = \frac{\sum_{j=1}^{m} \gamma_{ji} \boldsymbol{x}_j}{\sum_{j=1}^{m} \gamma_{ji}}$;
    Compute the updated covariance matrix: $\boldsymbol{\Sigma}_i' = \frac{\sum_{j=1}^{m} \gamma_{ji} (\boldsymbol{x}_j - \boldsymbol{\mu}_i')(\boldsymbol{x}_j - \boldsymbol{\mu}_i')^{\mathrm{T}}}{\sum_{j=1}^{m} \gamma_{ji}}$;
    Compute the updated mixture coefficients: $\alpha_i' = \frac{1}{m} \sum_{j=1}^{m} \gamma_{ji}$;
  end for
  Update the model parameters $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$ to $\{(\alpha_i', \boldsymbol{\mu}_i', \boldsymbol{\Sigma}_i') \mid 1 \leqslant i \leqslant k\}$;
until The termination condition is met
$C_i = \varnothing (1 \leqslant i \leqslant k)$;
for $j = 1, 2, \ldots, m$ **do**
  Determine the cluster label λ_j of $\boldsymbol{x}_j$ according to (9.31);
  Move $\boldsymbol{x}_j$ to the corresponding cluster: $C_{\lambda_j} = C_{\lambda_j} \cup \{\boldsymbol{x}_j\}$.
end for

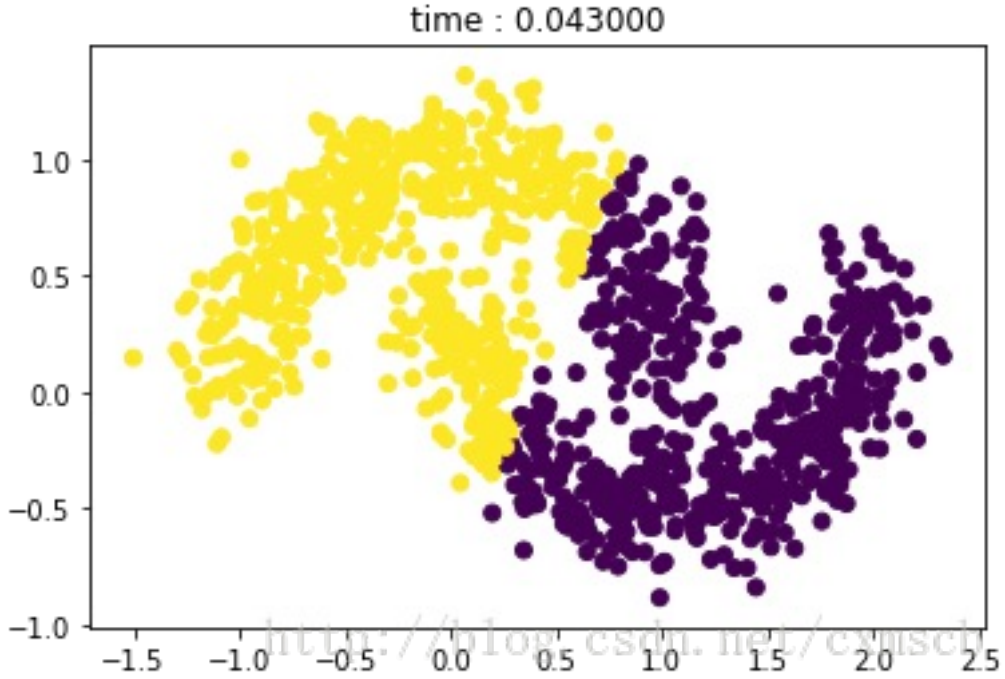
Output: Clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.

Outlines

- Clustering Problem
- Performance Measure
- Distance Calculation
- Prototype Clustering
- **Density Clustering**
- Hierarchical Clustering

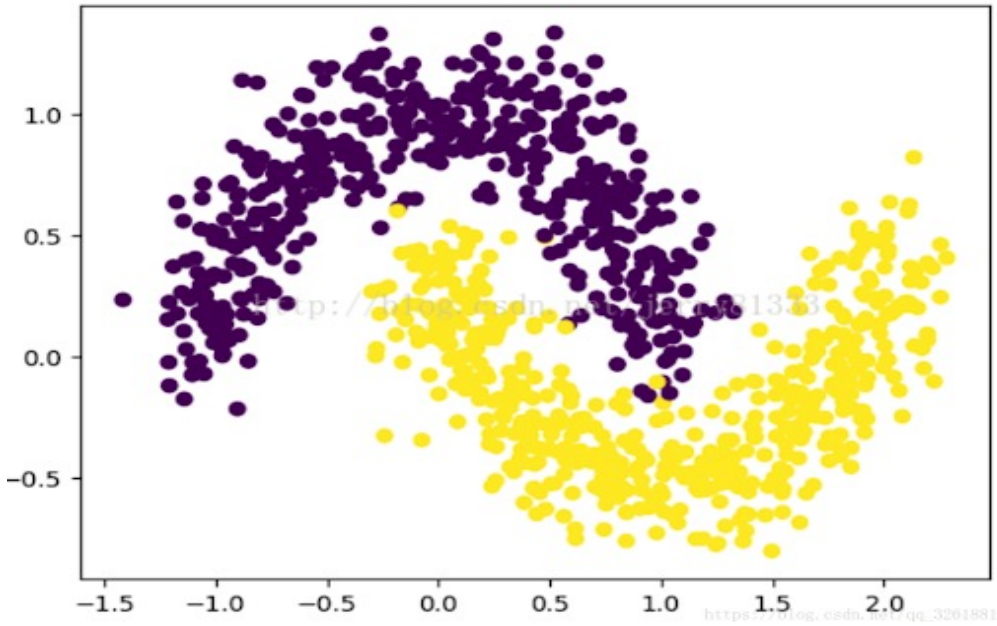
Density Clustering

Applying *k*-means clustering to following data, it is hard for prototype-based clustering to find the density information, which leads to a further results from expected:



Density Clustering

Density-based clustering: evaluate the connectivity between samples from the density perspective and expand the clusters by adding connectable samples.



Density Clustering

Definition of Density Clustering

Density Clustering is also known as *density-based clustering*.

Assuming the clustering structure can be determined by the densities of sample distributions.

Typically, density clustering algorithms evaluate the connectivity between samples from the density perspective and expand the clusters by adding connectable samples.

Next we introduce DBSCAN (Density-Based Spatial Clustering of Applications with Noise), which is a representative density clustering algorithm.

Density Clustering

- DBSCAN algorithm characterizes the density of sample distributions by a pair of *neighborhood* parameters $(\epsilon, MinPts)$
- Define the basic concepts:
  - ϵ-neighborhood: for $x_j \in D$, its ϵ-neighborhood includes all samples in D that have a distance to x_j no larger than ϵ;
  - Core object: if the ϵ-neighborhood of x_j includes at least MinPts samples, then x_j is called a core object; （核心对象）
  - Directly density-reachable: x_j is said to be directly density-reachable by x_i if x_i is a core object and x_j is in the ϵ-neighborhood of x_i; （密度直达）
  - Density-reachable: x_j is said to be density-reachable by x_i if there exists a sequence of samples $p_1, p_2, \cdots, p_n$, where $p_1 = x_i, p_n = x_j$ and p_{i+1} is directly density-reachable by p_i;
  - Density-connected: x_i and x_j are density-connected if there exists x_k such that both are density-reachable by x_k

Density Clustering

- An example

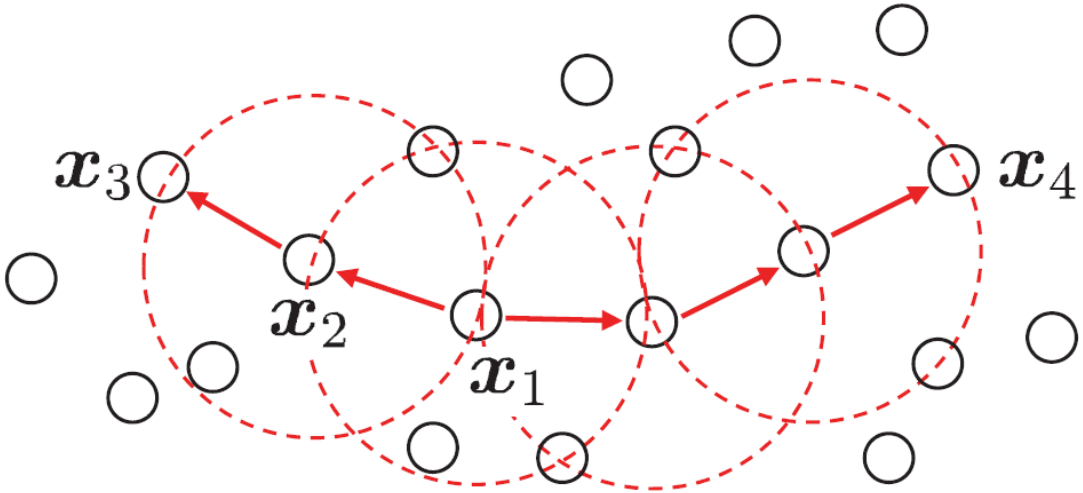
Let $MinPts = 3$:the dashed circles show theϵ -neighborhood

x_1 is a core object

x_2 is directly density-reachable by x_1

x_3 is density-reachable by x_1

x_3 and x_4 density-connected.



Density Clustering

- Definition of a cluster

The largest set of density-connected samples derived by density-reachable relationships.

- Formalization

Given the neighborhood parameters, a cluster is a non-empty subset with following properties:

Connectivity: $x_i \in C, x_j \in C \Rightarrow x_i$ and x_j are density-connected

Maximality: $x_i \in C, x_j$ is density-reachable by $x_i \Rightarrow x_j \in C$

Actually, if x is a core object and let $X = \{x' \in D \mid x' \; is \; density - reachable \; by \; x\}$ denote the set of samples density-reachable by x, then it can be proved that X is a cluster that satisfies both the connectivity and the maximality.

（连 接 性、最大性）

DBSCAN

① Find all core objects

② Find connected component

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
邻域参数 $(\epsilon, MinPts)$.

过程:

初始化核心对象集合: $\Omega = \varnothing$
for $j = 1, 2, \ldots, m$ **do**
  确定样本 $\boldsymbol{x}_j$ 的 ϵ-邻域 $N_\epsilon(\boldsymbol{x}_j)$;
  if $|N_\epsilon(\boldsymbol{x}_j)| \geqslant MinPts$ **then**
    将样本 $\boldsymbol{x}_j$ 加入核心对象集合: $\Omega = \Omega \bigcup \{\boldsymbol{x}_j\}$
  end if
end for
初始化聚类簇数: $k = 0$
初始化未访问样本集合: $\Gamma = D$
while $\Omega \neq \varnothing$ **do**
  记录当前未访问样本集合: $\Gamma_{\text{old}} = \Gamma$;
  随机选取一个核心对象 $\boldsymbol{o} \in \Omega$, 初始化队列 $Q =< \boldsymbol{o} >$;
  $\Gamma = \Gamma \setminus \{\boldsymbol{o}\}$;
  while $Q \neq \varnothing$ **do**
    取出队列 Q 中的首个样本 $\boldsymbol{q}$;
    if $|N_\epsilon(\boldsymbol{q})| \geqslant MinPts$ **then**
      令 $\Delta = N_\epsilon(\boldsymbol{q}) \bigcap \Gamma$;
      将 Δ 中的样本加入队列 Q;
      $\Gamma = \Gamma \setminus \Delta$;
    end if
  end while
  $k = k + 1$, 生成聚类簇 $C_k = \Gamma_{\text{old}} \setminus \Gamma$;
  $\Omega = \Omega \setminus C_k$
end while

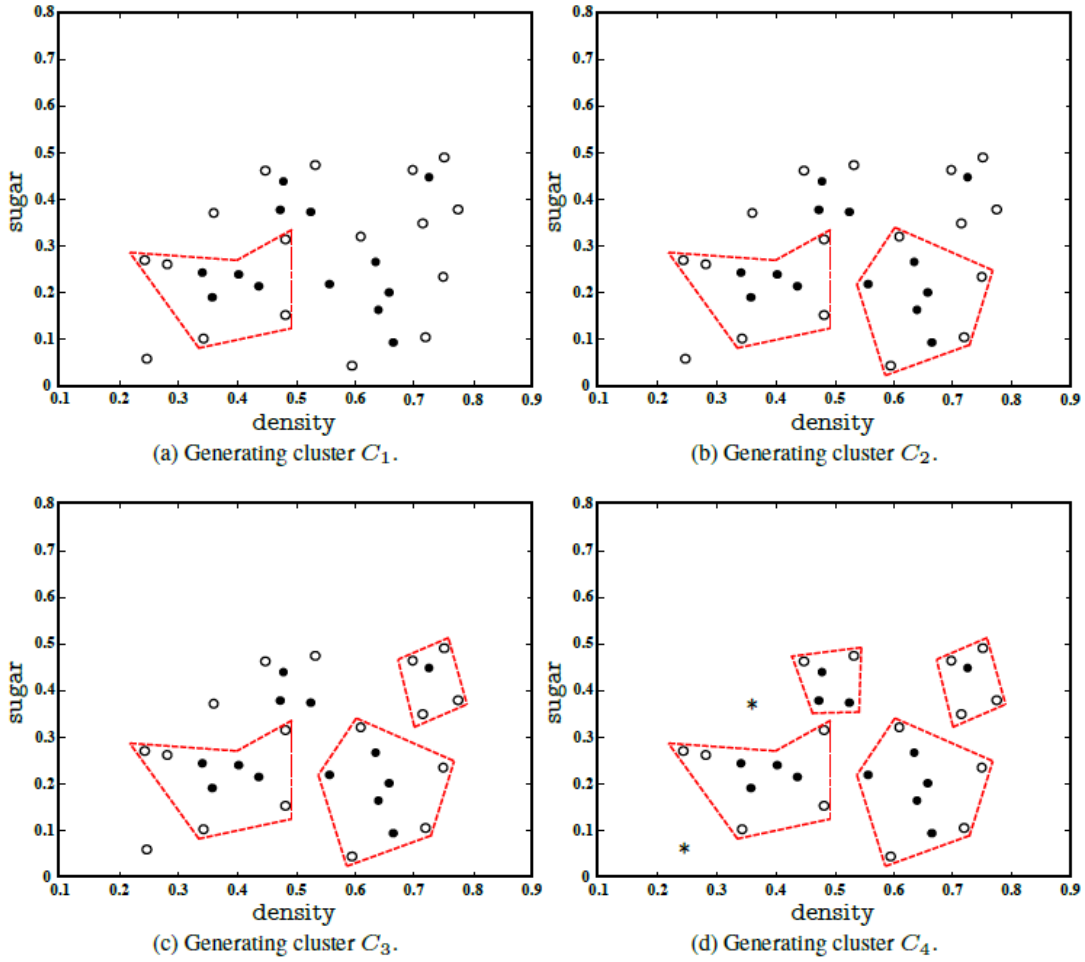
输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

DBSCAN

```
DBSCAN(DB, distFunc, eps, minPts) {
    C := 0                                                  /* Cluster counter */
    for each point P in database DB {
        if label(P) ≠ undefined then continue               /* Previously processed in inner loop */
        Neighbors N := RangeQuery(DB, distFunc, P, eps)     /* Find neighbors */
        if |N| < minPts then {                              /* Density check */
            label(P) := Noise                               /* Label as Noise */
            continue
        }
        C := C + 1                                          /* next cluster label */
        label(P) := C                                       /* Label initial point */
        SeedSet S := N \ {P}                                /* Neighbors to expand */
        for each point Q in S {                             /* Process every seed point Q */
            if label(Q) = Noise then label(Q) := C          /* Change Noise to border point */
            if label(Q) ≠ undefined then continue           /* Previously processed (e.g., border point) */
            label(Q) := C                                   /* Label neighbor */
            Neighbors N := RangeQuery(DB, distFunc, Q, eps) /* Find neighbors */
            if |N| ≥ minPts then {                          /* Density check (if Q is a core point) */
                S := S ∪ N                                  /* Add new neighbors to seed set */
            }
        }
    }
}
```

Density Clustering

- Clustering results:



Outlines

- Clustering Problem
- Performance Measure
- Distance Calculation
- Prototype Clustering
- Density Clustering
- **Hierarchical Clustering**

Hierarchical Clustering

- Hierarchical Clustering aims to create a tree-like clustering structure by dividing a data set at different layers. The hierarchy of clusters can be formed by taking either a *bottom-up* strategy (**Agglomerative，聚集**) or a *top-down* strategy (**Divisive，分裂**).
- **AGNES algorithm** (*bottom-up Hierarchical Clustering*)

starts by considering each sample in the data set as an initial cluster. Then, in each round, two nearest clusters are merged as a new cluster, and this process repeats until the number of clusters meets the pre-specified value.

We define the distances of given clusters C_i and C_j in different forms.

Hierarchical Clustering

Minimum distance (single-linkage , “单链接”):

$$d_{\min}(C_i, C_j) = \min_{\boldsymbol{x} \in C_i, \boldsymbol{z} \in C_j} \operatorname{dist}(\boldsymbol{x}, \boldsymbol{z})$$

Maximum distance (complete-linkage , “全链接”):

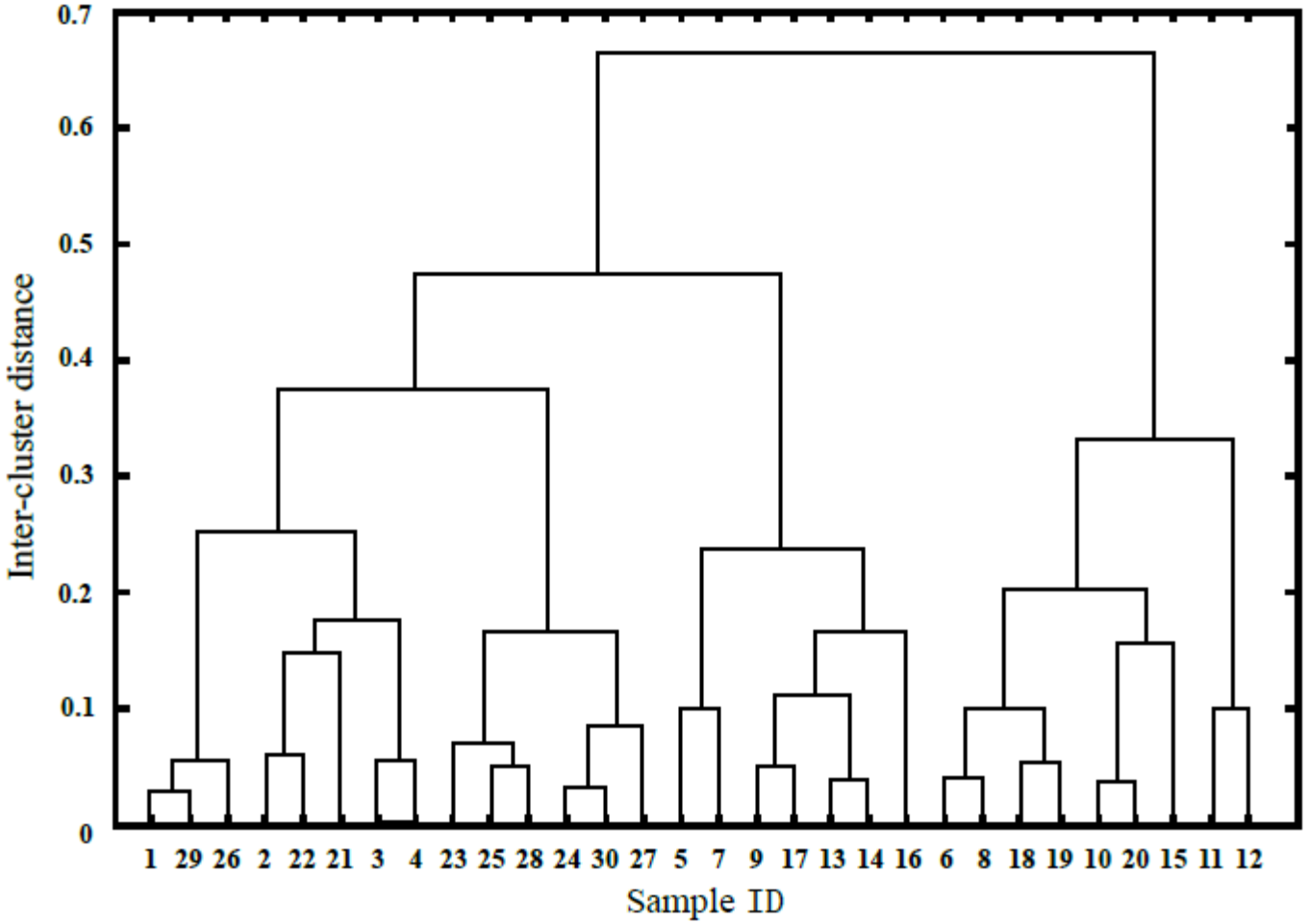
$$d_{\max}(C_i, C_j) = \max_{\boldsymbol{x} \in C_i, \boldsymbol{z} \in C_j} \operatorname{dist}(\boldsymbol{x}, \boldsymbol{z})$$

Average distance (average-linkage , “均链接 “) :

$$d_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\boldsymbol{x} \in C_i} \sum_{\boldsymbol{z} \in C_j} \operatorname{dist}(\boldsymbol{x}, \boldsymbol{z})$$

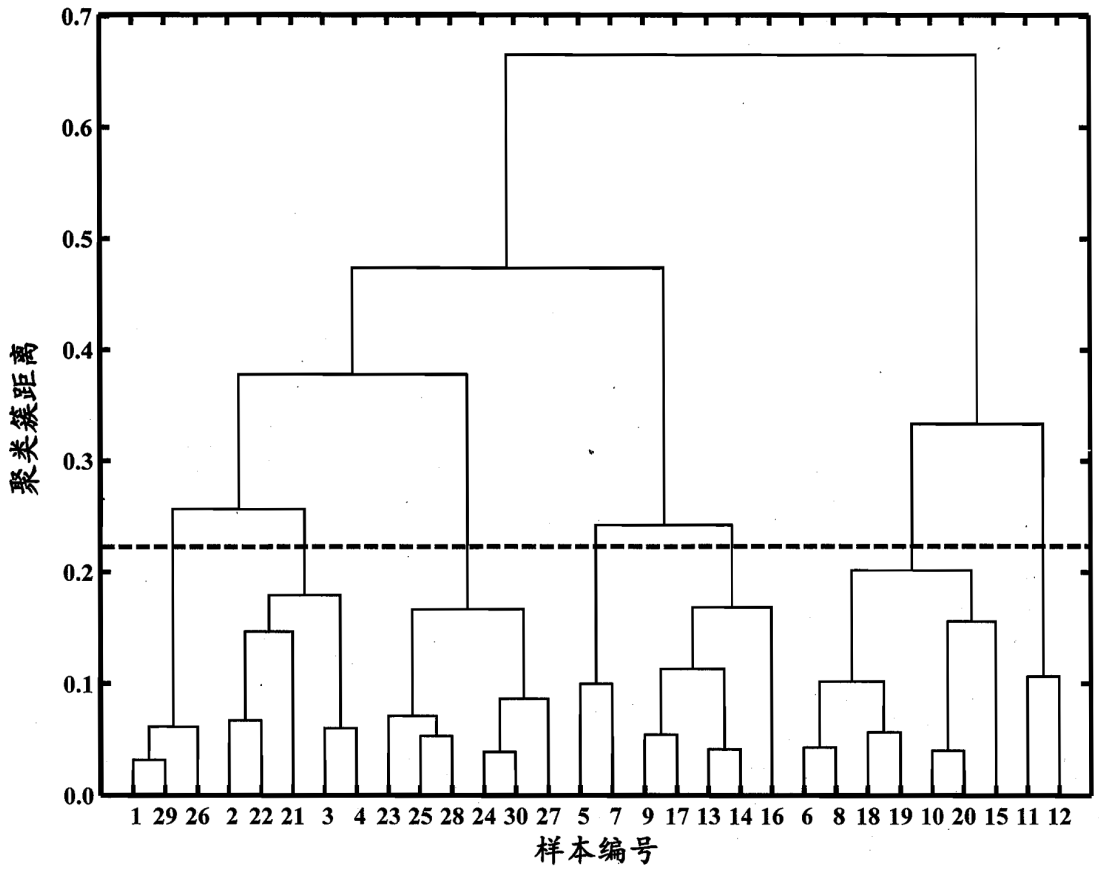
Hierarchical Clustering – dendrogram

- The dendrogram (树状图) of AGNES :



Hierarchical Clustering – dendrogram

- The dendrogram (树状图) of AGNES :



Hierarchical Clustering – AGNES Algorithm

① Initialize distance matrix

② merge clusters and update distance matrix

Algorithm 9.5 AGNES.

Input: Data set $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
Cluster distance metric function d;
Number of clusters k.

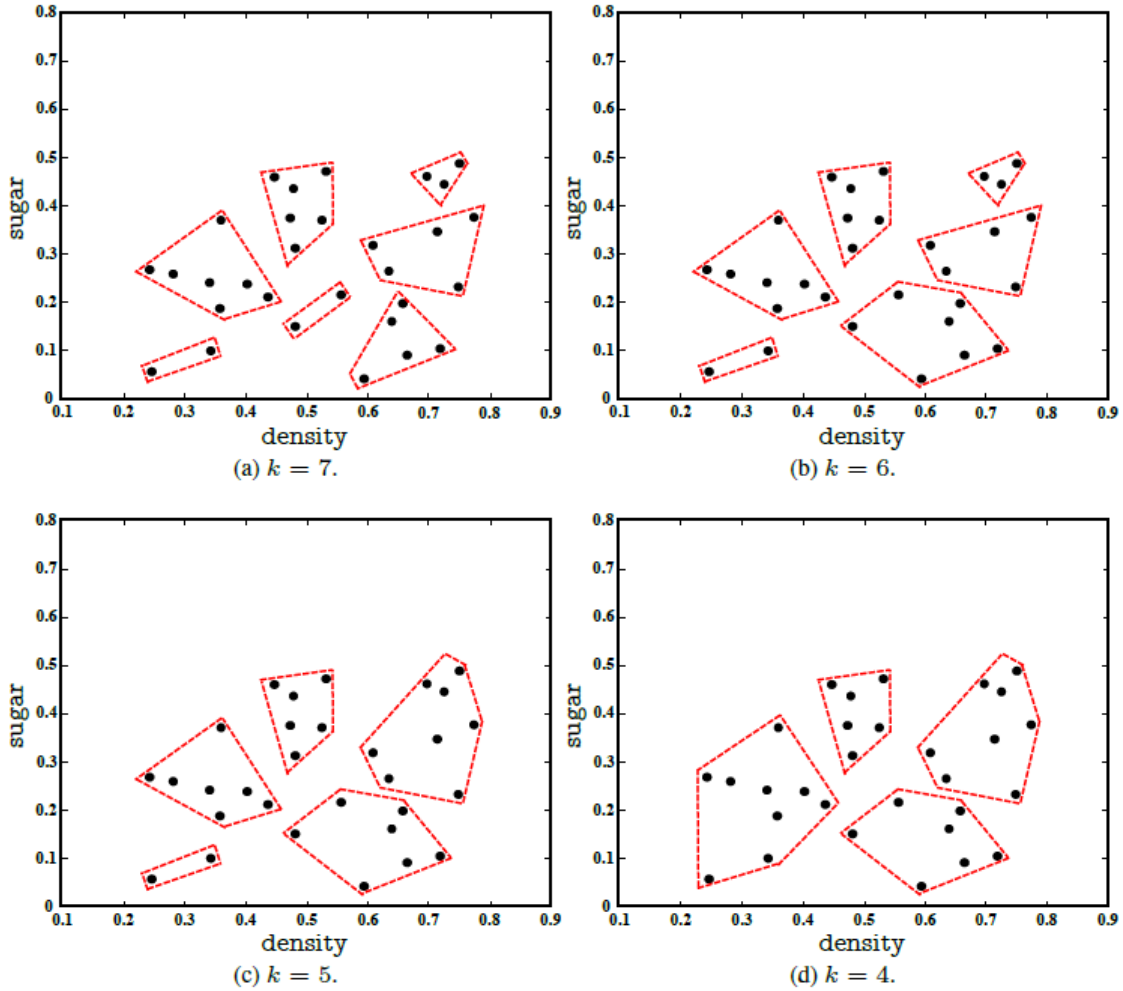
Process:

```
for j = 1, 2, ..., m do
    C_j = {x_j};
end for
for i = 1, 2, ..., m do
    for j = i + 1, ..., m do
        M(i, j) = d(C_i, C_j);
        M(j, i) = M(i, j);
    end for
end for
Set the current number of clusters: q = m;
while q > k do
    Find two clusters C_i* and C_j* that have the shortest distance;
    Merge C_i* and C_j*: C_i* = C_i* ∪ C_j*;
    for j = j* + 1, j* + 2, ..., q do
        Change the index of C_j to C_{j-1};
    end for
    Delete the j*th row and j*th column of the distance matrix M;
    for j = 1, 2, ..., q − 1 do
        M(i*, j) = d(C_i*, C_j);
        M(j, i*) = M(i*, j);
    end for
    q = q − 1.
end while
```

Output: Clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.

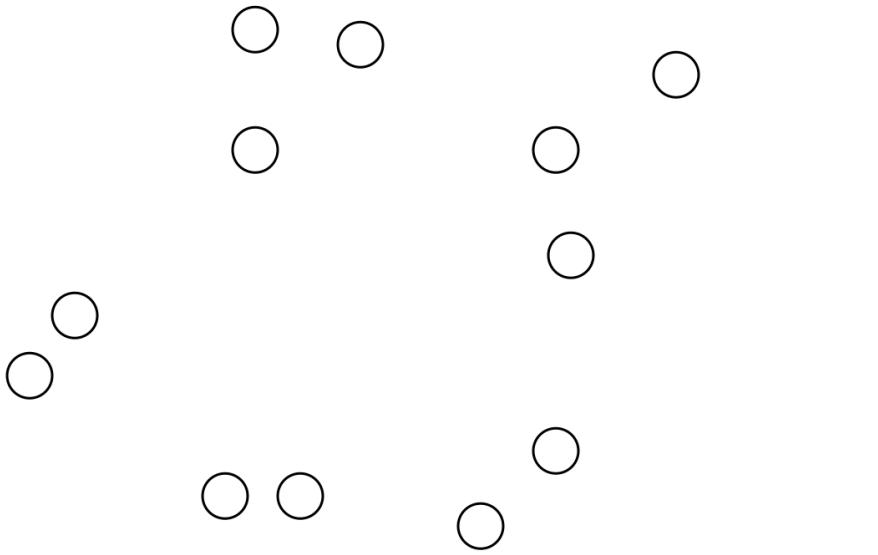
Hierarchical Clustering

- AGNES Clustering results :



Input/ Initial setting

- Start with clusters of individual points and a distance/proximity matrix

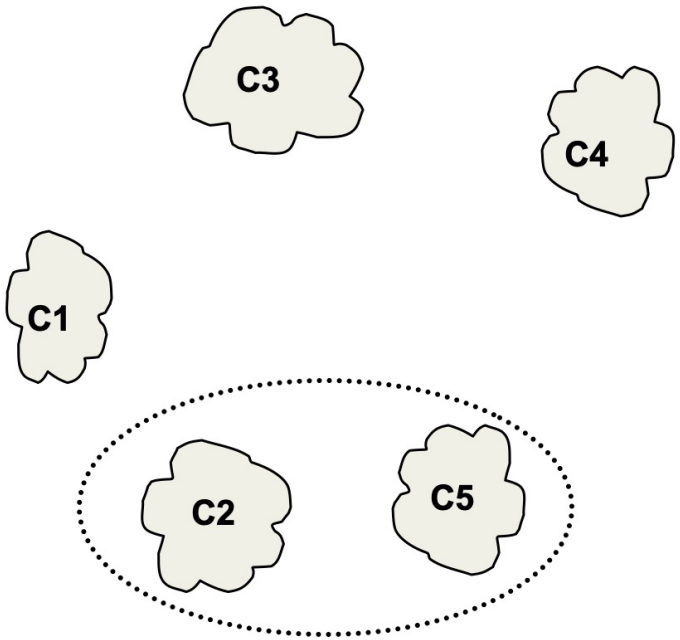


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
. . .						

Distance/Proximity Matrix

Intermediate State

- Merge the two closest clusters (C2 and C5) and update the distance matrix.

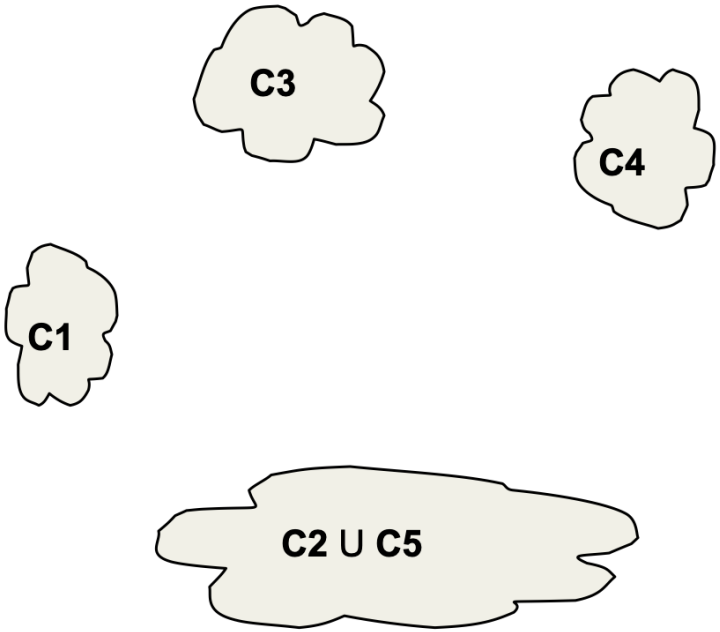


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix

After Merging

- "How do we update the distance matrix?"



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

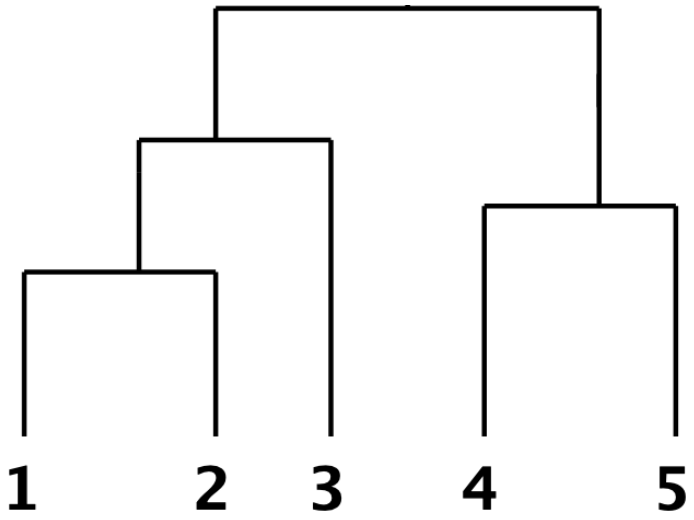
Example

Apply the Hierarchical Clustering to the following proximity matrix with **single** linkage.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00

Example

Apply the Hierarchical Clustering to the following proximity matrix with **single** linkage.



Updating Distance Matrix

Let us assume that we have five samples (a, b, c, d, e) and the following matrix of pairwise distances between them:

	a	b	c	d	e
a	0	17	21	31	23
b	17	0	30	34	21
c	21	30	0	28	39
d	31	34	28	0	43
e	23	21	39	43	0

In this example, $D_1(a, b) = 17$ is the lowest value of D_1 so we cluster samples a and b.

Updating Distance Matrix

We then proceed to update the initial distance matrix D_1 into a new matrix D_2, reduced in size by one row and one column. Let's consider the **single-linkage** clustering:

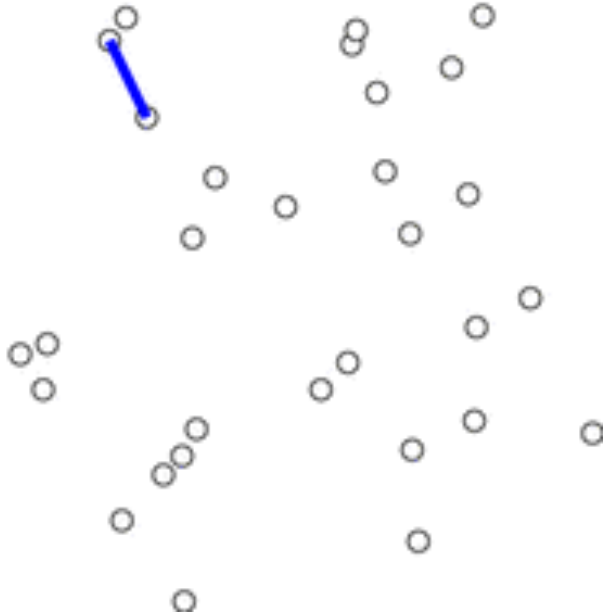
$$
\begin{aligned}
D_2((a,b),c) &= \min(D_1(a,c), D_1(b,c)) = \min(21, 30) = 21 \\
D_2((a,b),d) &= \min(D_1(a,d), D_1(b,d)) = \min(31, 34) = 31 \\
D_2((a,b),e) &= \min(D_1(a,e), D_1(b,e)) = \min(23, 21) = 21
\end{aligned}
$$

	(a,b)	c	d	e
(a,b)	0	**21**	**31**	**21**
c	**21**	0	*28*	*39*
d	**31**	*28*	0	*43*
e	**21**	*39*	*43*	0

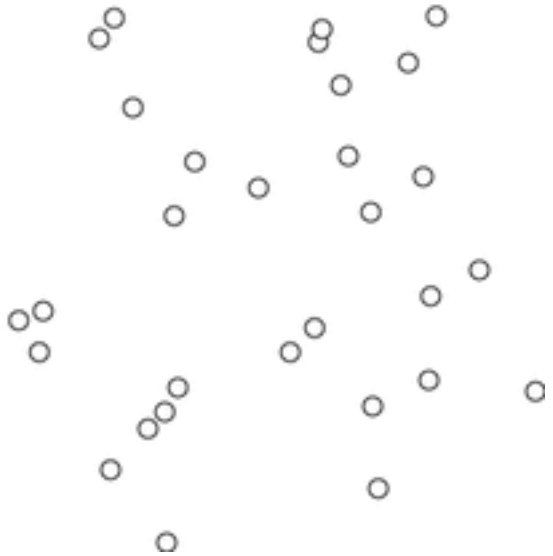
What if we adopt the complete-linkage clustering?

Time complexity

time complexity $O(n^2)$ and space complexity $O(n)$



Prim's algorithm
minimum spanning tree



Kruskal's algorithm
(disjoint-set)

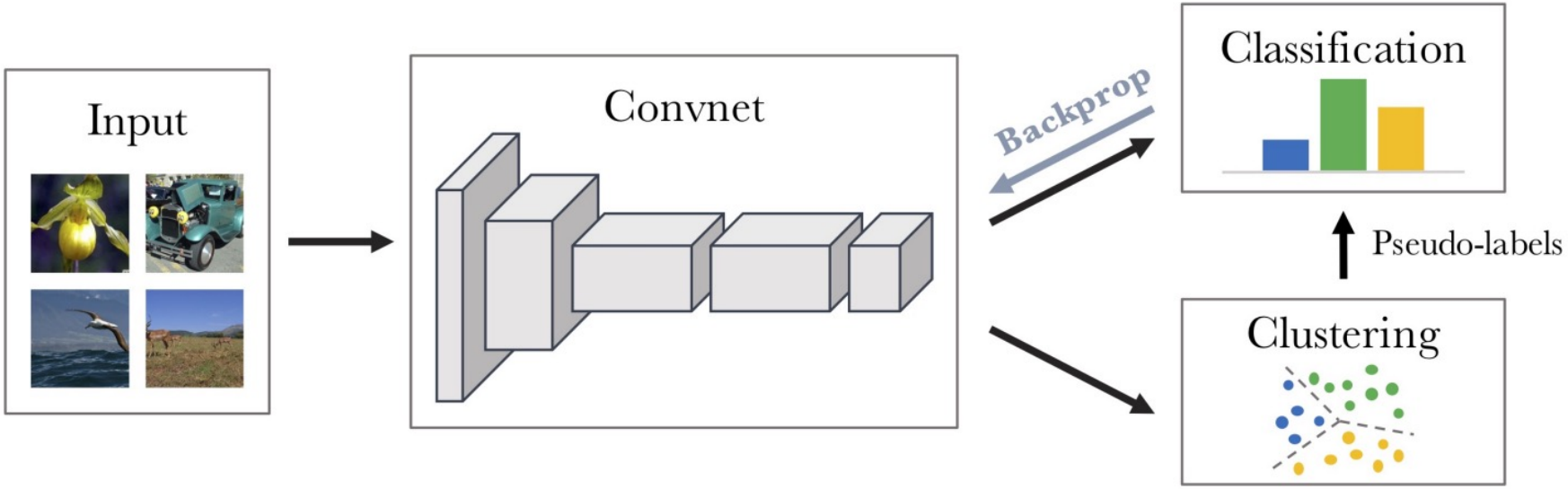
Summary

- Clustering Problem
- Performance Measure
- Distance Calculation
- Prototype Clustering
- Density Clustering
- Hierarchical Clustering

Recent Progress

DeepCluster

DeepCluster is a novel method for the end-to-end learning of convnets that works with any standard clustering algorithm.

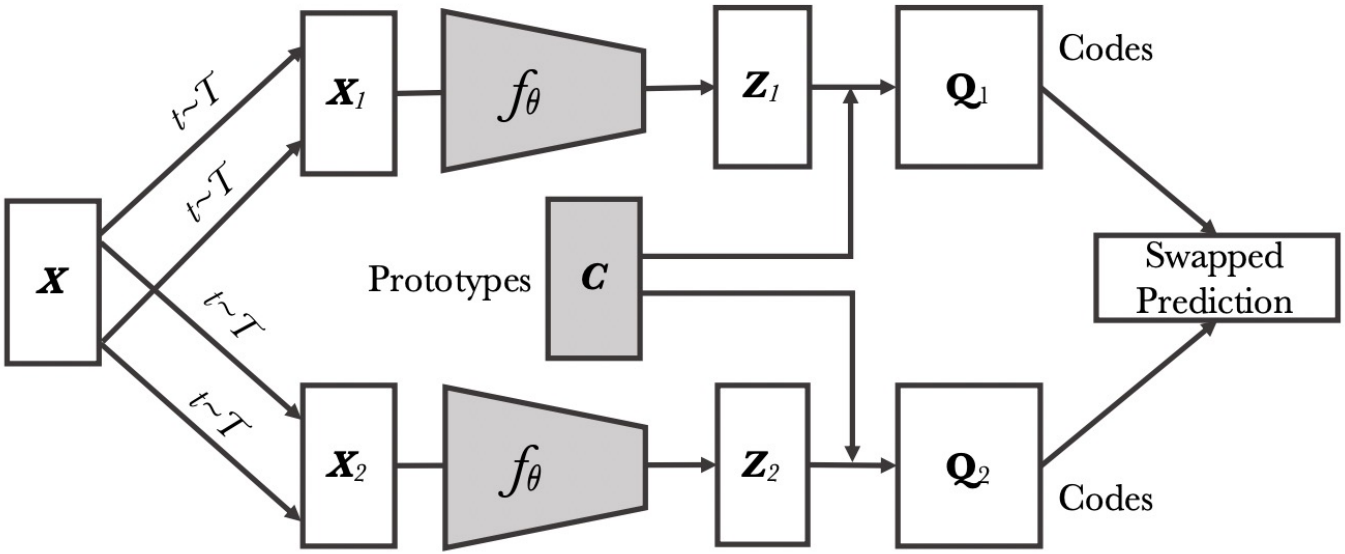


$$\min_{C \in \mathbb{R}^{d \times k}} \frac{1}{N} \sum_{n=1}^{N} \min_{y_n \in \{0,1\}^k} \|f_\theta(x_n) - C y_n\|_2^2 \quad \text{such that} \quad y_n^\top 1_k = 1.$$

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. "Deep Clustering for Unsupervised Learning of Visual Features." Proc. ECCV (2018). [2200+ citation, May 25, 2023]

SwAV - Online clustering

- SwAV: **Sw**apping **A**ssignments between multiple **V**iews of the same image.
- SwAV uses trainable prototypes vectors

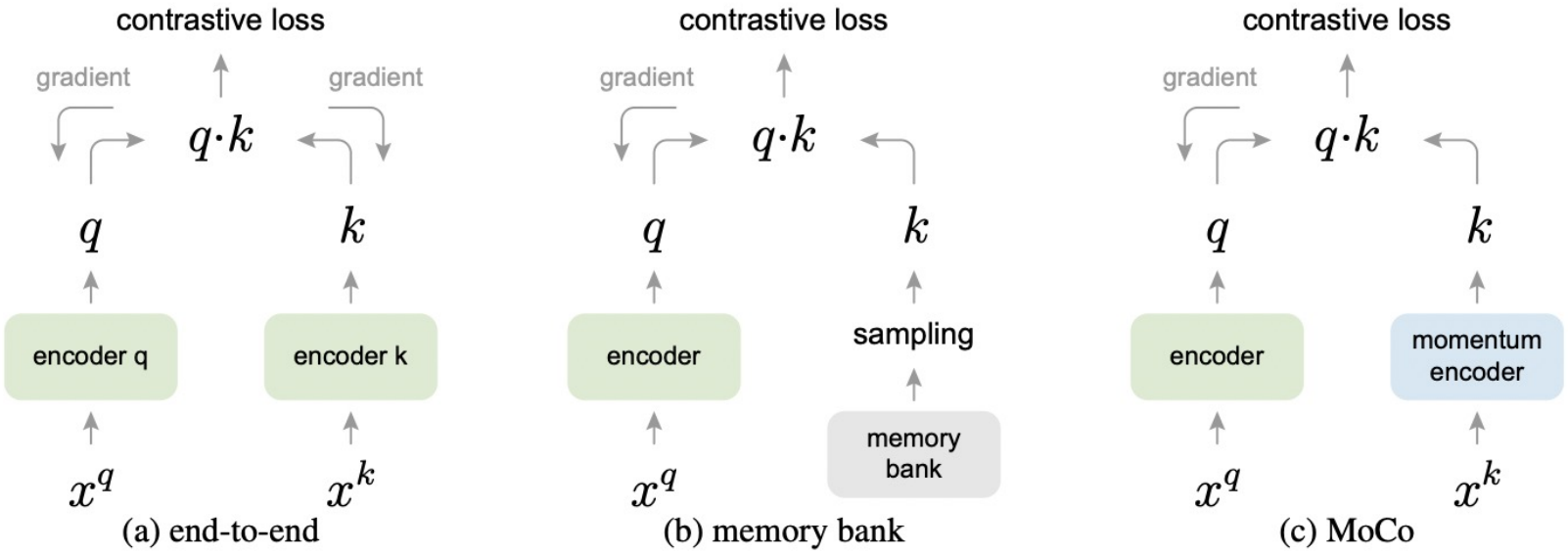


$$-\frac{1}{N}\sum_{n=1}^{N}\sum_{s,t\sim\mathcal{T}}\left[\frac{1}{\tau}\mathbf{z}_{nt}^{\top}\mathbf{C}\mathbf{q}_{ns}+\frac{1}{\tau}\mathbf{z}_{ns}^{\top}\mathbf{C}\mathbf{q}_{nt}-\log\sum_{k=1}^{K}\exp\left(\frac{\mathbf{z}_{nt}^{\top}\mathbf{c}_k}{\tau}\right)-\log\sum_{k=1}^{K}\exp\left(\frac{\mathbf{z}_{ns}^{\top}\mathbf{c}_k}{\tau}\right)\right]$$

Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. NeurIPS 2020 [2100+ citation, May 25, 2023]

Representation Learning

- Clustering can be a way of self-supervised learning（自监督学习）. How?
- Now, the most popular representation learning methods are based on self-supervised learning, e.g., MoCo, SimCLR.



Kaiming He et al., Momentum Contrast for Unsupervised Visual Representation Learning. CVPR 2020 [7000+ citation, May 25, 2023]