Data mining and Machine learning

Part 13. Representative-based Clustering

Representative-based Clustering

Given a dataset with n points in a d-dimensional space, $\boldsymbol{D} = \{\boldsymbol{x}_i\}_{i=1}^n$, and given the number of desired clusters k, the goal of representative-based clustering is to partition the dataset into k groups or clusters, which is called a *clustering* and is denoted as $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.

For each cluster C_i there exists a representative point that summarizes the cluster, a common choice being the mean (also called the *centroid*) $\boldsymbol{\mu}_i$ of all points in the cluster, that is,

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{x_j \in C_i} \boldsymbol{x}_j$$

where $n_i = |C_i|$ is the number of points in cluster C_i.

A brute-force or exhaustive algorithm for finding a good clustering is simply to generate all possible partitions of n points into k clusters, evaluate some optimization score for each of them, and retain the clustering that yields the best score. However, this is clearly infeasilbe, since there are $O(k^n/k!)$ clusterings of n points into k groups.

K-means Algorithm: Objective

The *sum of squared errors* scoring function is defined as

$$SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} \|\boldsymbol{x}_j - \boldsymbol{\mu}_i\|^2$$

The goal is to find the clustering that minimizes the SSE score:

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} \{SSE(\mathcal{C})\}$$

K-means employs a greedy iterative approach to find a clustering that minimizes the SSE objective. As such it can converge to a local optima instead of a globally optimal clustering.

K-means Algorithm: Objective

K-means initializes the cluster means by randomly generating k points in the data space. Each iteration of K-means consists of two steps: (1) cluster assignment, and (2) centroid update.

Given the k cluster means, in the cluster assignment step, each point $\boldsymbol{x}_j \in \boldsymbol{D}$ is assigned to the closest mean, which induces a clustering, with each cluster C_i comprising points that are closer to $\boldsymbol{\mu}_i$ than any other cluster mean. That is, each point $\boldsymbol{x}_j$ is assigned to cluster C_{i^*}, where

$$i^* = \arg\min_{i=1}^{k} \left\{ \|\boldsymbol{x}_j - \boldsymbol{\mu}_i\|^2 \right\}$$

Given a set of clusters C_i, $i = 1, \ldots, k$, in the centroid update step, new mean values are computed for each cluster from the points in C_i.

The cluster assignment and centroid update steps are carried out iteratively until we reach a fixed point or local minima.

K-Means Algorithm

K-means $(\boldsymbol{D}, k, \epsilon)$**:**

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

repeat

  $t \leftarrow t + 1$

  $C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$

  `// Cluster Assignment Step`

  foreach $\boldsymbol{x}_j \in \boldsymbol{D}$ **do**

    $i^* \leftarrow \arg\min_i \left\{ \|\boldsymbol{x}_j - \boldsymbol{\mu}_i^t\|^2 \right\}$ `// Assign` $\boldsymbol{x}_j$ `to closest centroid`

    $C_{i^*} \leftarrow C_{i^*} \cup \{\boldsymbol{x}_j\}$

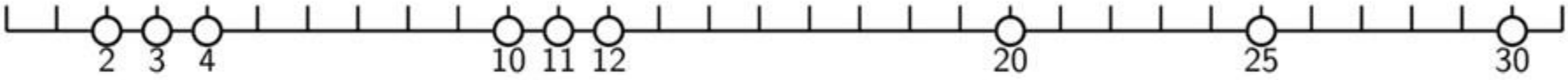
  `// Centroid Update Step`

  foreach $i = 1$ *to* k **do**

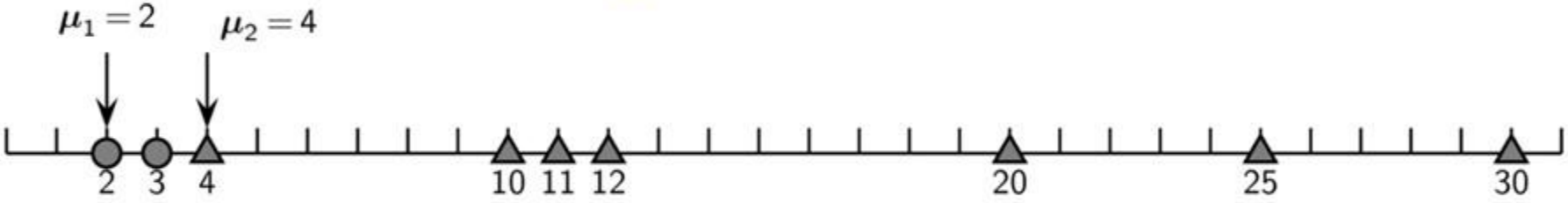
    $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$

until $\sum_{i=1}^k \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$

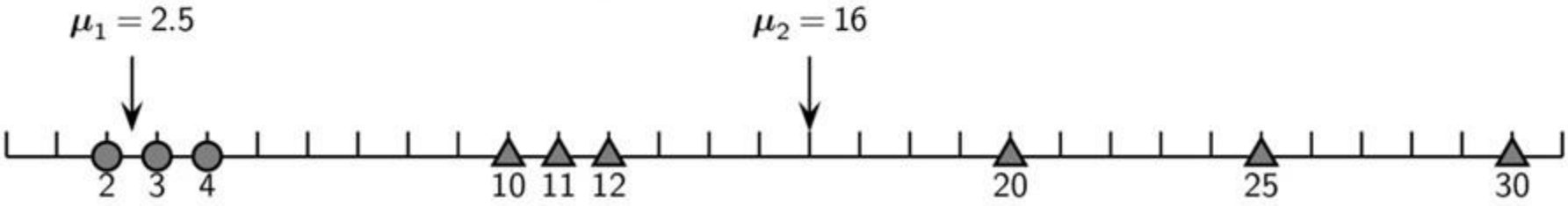
K-means in One Dimension



(a) Initial dataset

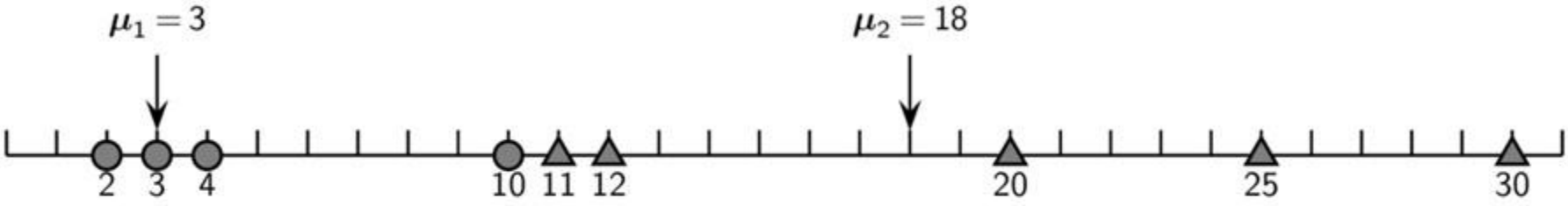


(b) Iteration: $t = 1$

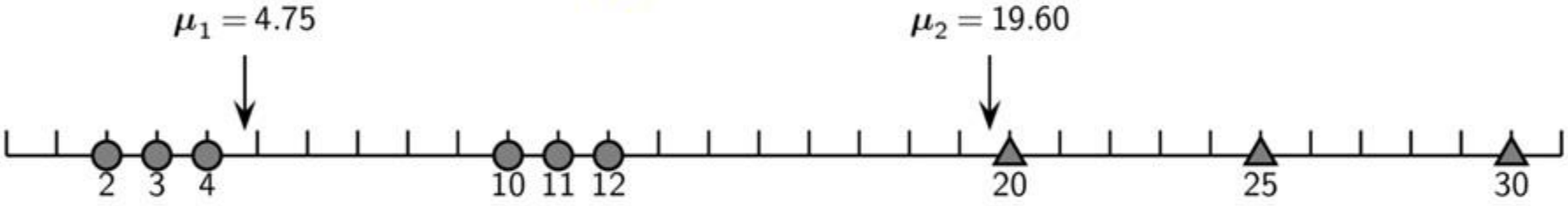


(c) Iteration: $t = 2$

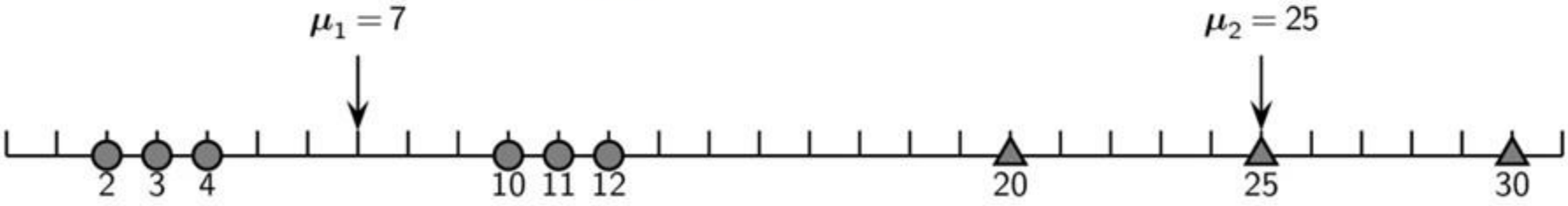
K-means in One Dimension (contd.)



(d) Iteration: $t = 3$

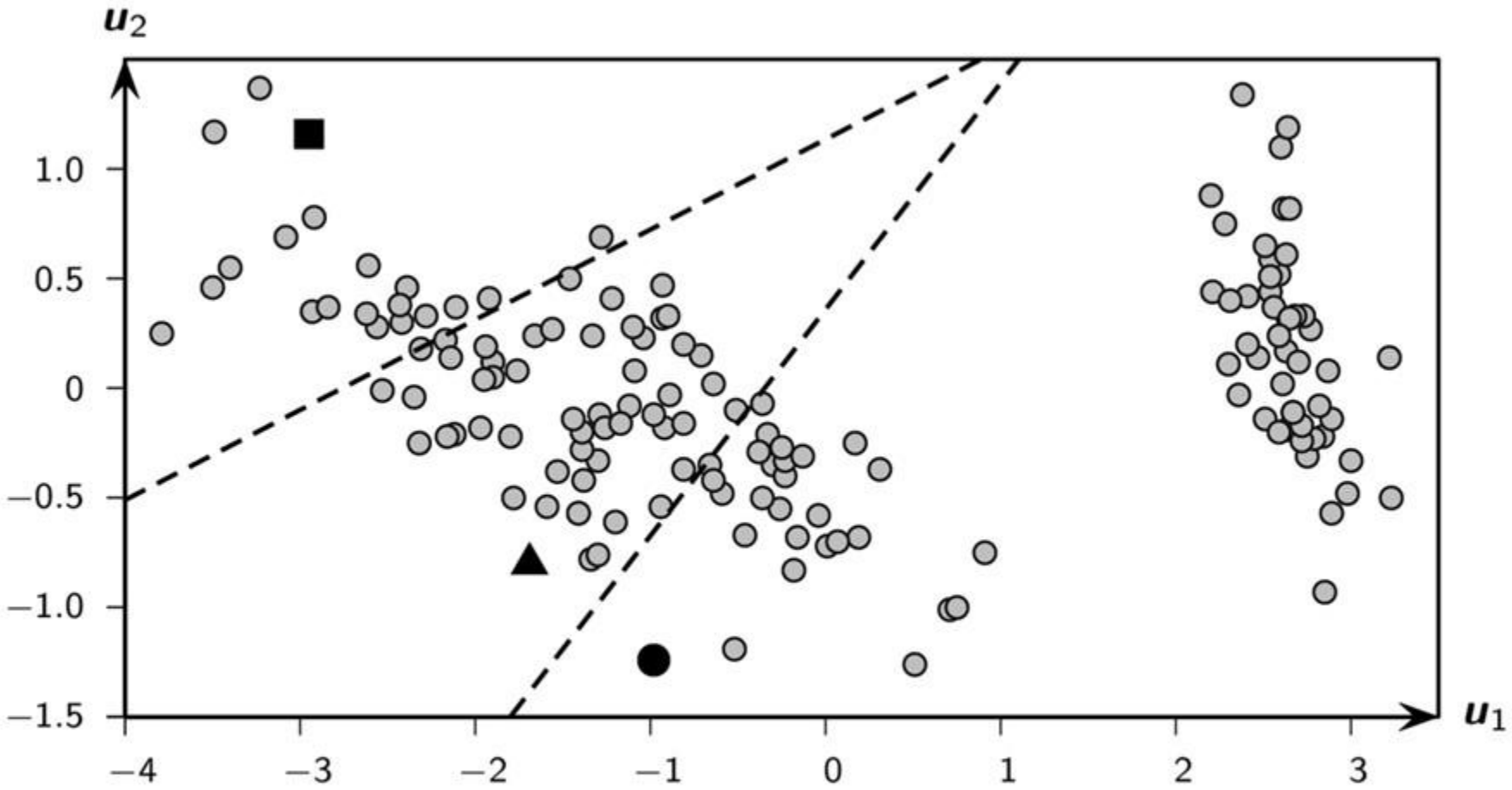


(e) Iteration: $t = 4$



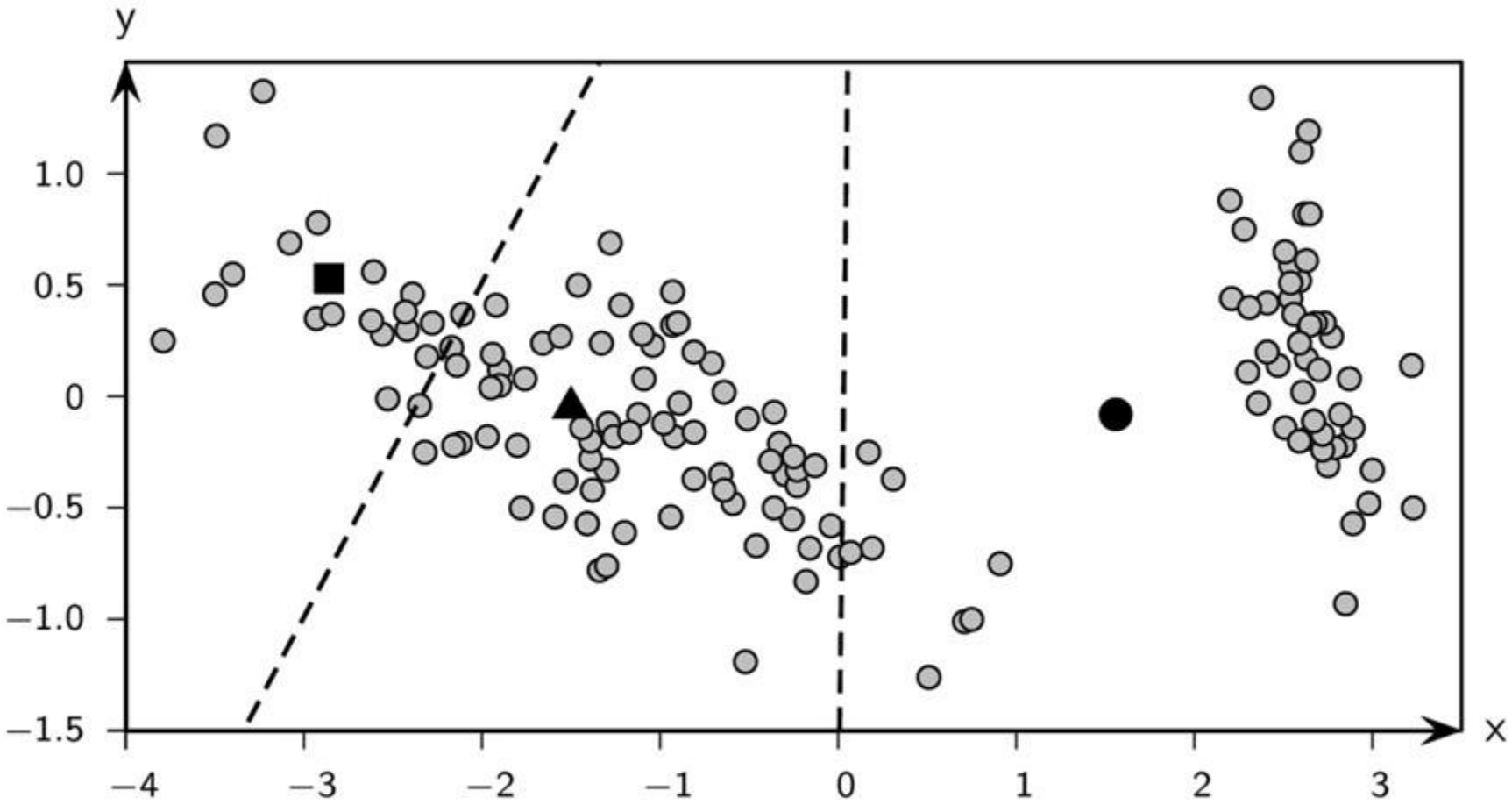
(f) Iteration: $t = 5$ (converged)

K-means in 2D: Iris Principal Components



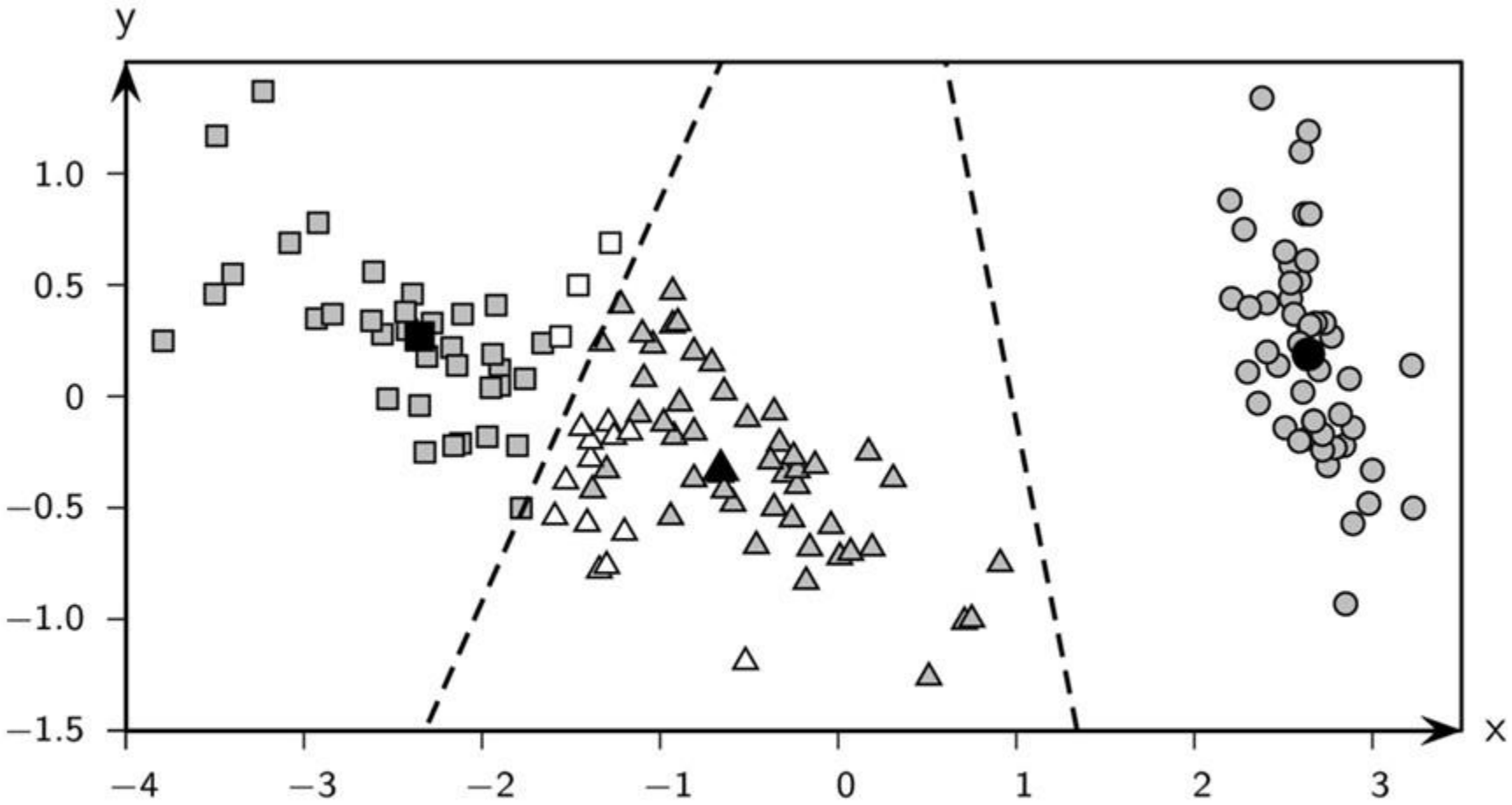
(a) Random initialization: $t = 0$

K-means in 2D: Iris Principal Components



(b) Iteration: $t = 1$

K-means in 2D: Iris Principal Components



(c) Iteration: $t = 8$ (converged)

Kernel K-means

In K-means, the separating boundary between clusters is linear. Kernel K-means allows one to extract nonlinear boundaries between clusters via the use of the kernel trick, i.e., we show that all operations involve only the kernel value between a pair of points.

Let $\boldsymbol{x}_i \in \boldsymbol{D}$ be mapped to $\phi(\boldsymbol{x}_i)$ in feature space. Let $\boldsymbol{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{i,j=1,\ldots,n}$ denote the $n \times n$ symmetric kernel matrix, where $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$.

The cluster means in feature space are $\{\boldsymbol{\mu}_1^\phi, \ldots, \boldsymbol{\mu}_k^\phi\}$, where $\boldsymbol{\mu}_i^\phi = \frac{1}{n_i} \sum_{\boldsymbol{x}_j \in C_i} \phi(\boldsymbol{x}_j)$.

The sum of squared errors in feature space is

$$\min_{\mathcal{C}} SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} \left\| \phi(\boldsymbol{x}_j) - \boldsymbol{\mu}_i^\phi \right\|^2 = \sum_{j=1}^{n} K(\boldsymbol{x}_j, \boldsymbol{x}_j) - \sum_{i=1}^{k} \frac{1}{n_i} \sum_{\boldsymbol{x}_a \in C_i} \sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b)$$

Thus, the kernel K-means SSE objective function can be expressed purely in terms of the kernel function.

Kernel K-means: Cluster Reassignment

Consider the distance of a point $\phi(\boldsymbol{x}_j)$ to the mean $\boldsymbol{\mu}_i^\phi$ in feature space

$$\begin{aligned}\left\|\phi(\boldsymbol{x}_j) - \boldsymbol{\mu}_i^\phi\right\|^2 &= \|\phi(\boldsymbol{x}_j)\|^2 - 2\phi(\boldsymbol{x}_j)^T\boldsymbol{\mu}_i^\phi + \left\|\boldsymbol{\mu}_i^\phi\right\|^2 \\ &= K(\boldsymbol{x}_j, \boldsymbol{x}_j) - \frac{2}{n_i}\sum_{\boldsymbol{x}_a \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j) + \frac{1}{n_i^2}\sum_{\boldsymbol{x}_a \in C_i}\sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b)\end{aligned}$$

Kernel K-means assign a point to the closest cluster mean as follows:

$$\begin{aligned}C^*(\boldsymbol{x}_j) &= \arg\min_i \left\{\left\|\phi(\boldsymbol{x}_j) - \boldsymbol{\mu}_i^\phi\right\|^2\right\} \\ &= \boxed{\arg\min_i \left\{\frac{1}{n_i^2}\sum_{\boldsymbol{x}_a \in C_i}\sum_{\boldsymbol{x}_b \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b) - \frac{2}{n_i}\sum_{\boldsymbol{x}_a \in C_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j)\right\}}\end{aligned}$$

Kernel-Kmeans Algorithm

Kernel-Kmeans(K, k, ϵ)**:**

$t \leftarrow 0$

$\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$// Randomly partition points into k clusters

repeat

  $t \leftarrow t+1$

  foreach $C_i \in \mathcal{C}^{t-1}$ **do** // Squared norm of cluster means

    $\text{sqnorm}_i \leftarrow \frac{1}{n_i^2} \sum_{x_a \in C_i} \sum_{x_b \in C_i} K(x_a, x_b)$

  foreach $x_j \in D$ **do** // Average kernel value for x_j and C_i

    foreach $C_i \in \mathcal{C}^{t-1}$ **do**

      $\text{avg}_{ji} \leftarrow \frac{1}{n_i} \sum_{x_a \in C_i} K(x_a, x_j)$

  // Find closest cluster for each point

  foreach $x_j \in D$ **do**

    foreach $C_i \in \mathcal{C}^{t-1}$ **do**

      $d(x_j, C_i) \leftarrow \text{sqnorm}_i - 2 \cdot \text{avg}_{ji}$

    $j^* \leftarrow \arg\min_i \{d(x_j, C_i)\}$

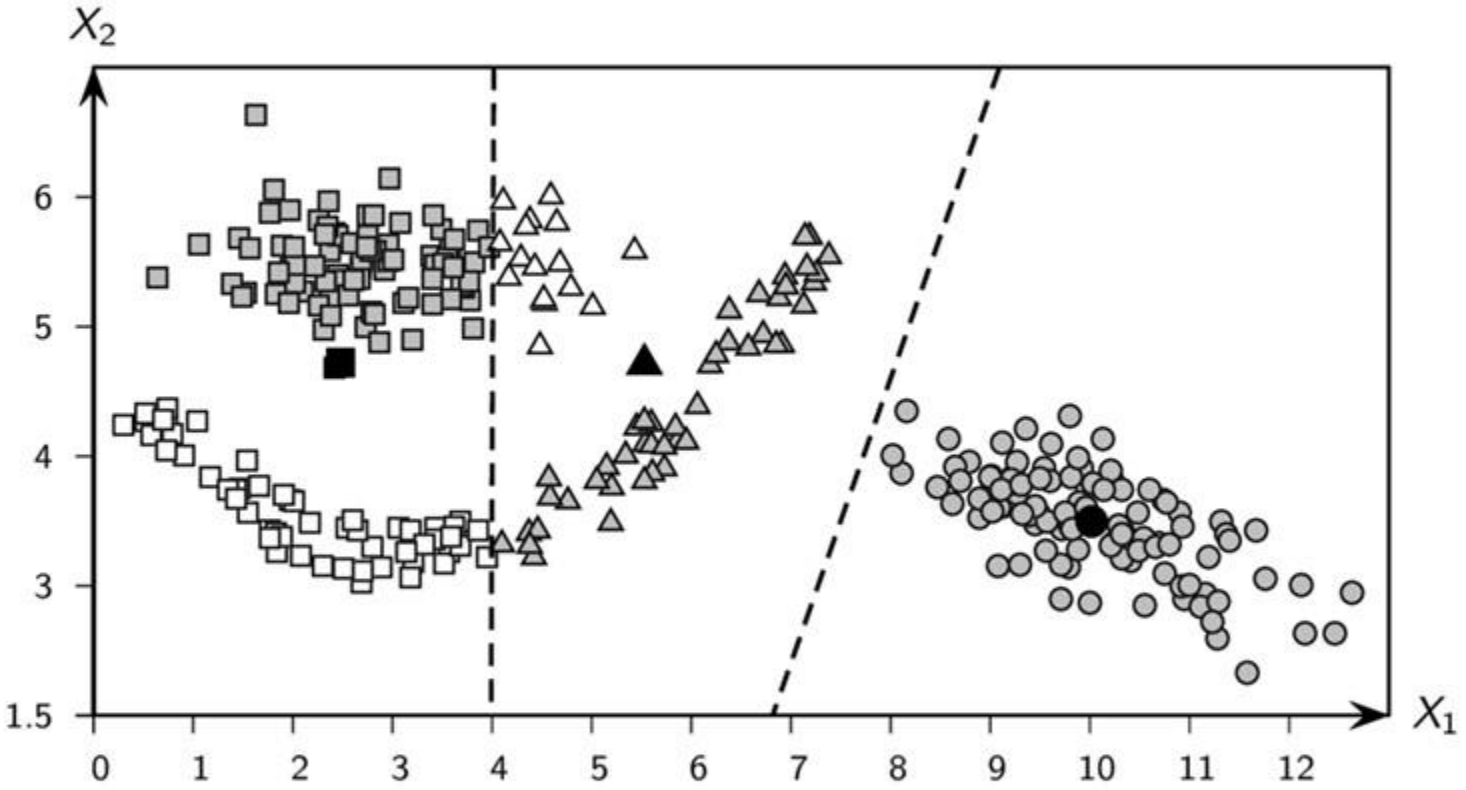
    $C_{j^*}^t \leftarrow C_{j^*}^t \cup \{x_j\}$ // Cluster reassignment

  $\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$

until $1 - \frac{1}{n} \sum_{i=1}^{k} \left| C_i^t \cap C_i^{t-1} \right| \leq \epsilon$

K-Means or Kernel K-means with Linear Kernel

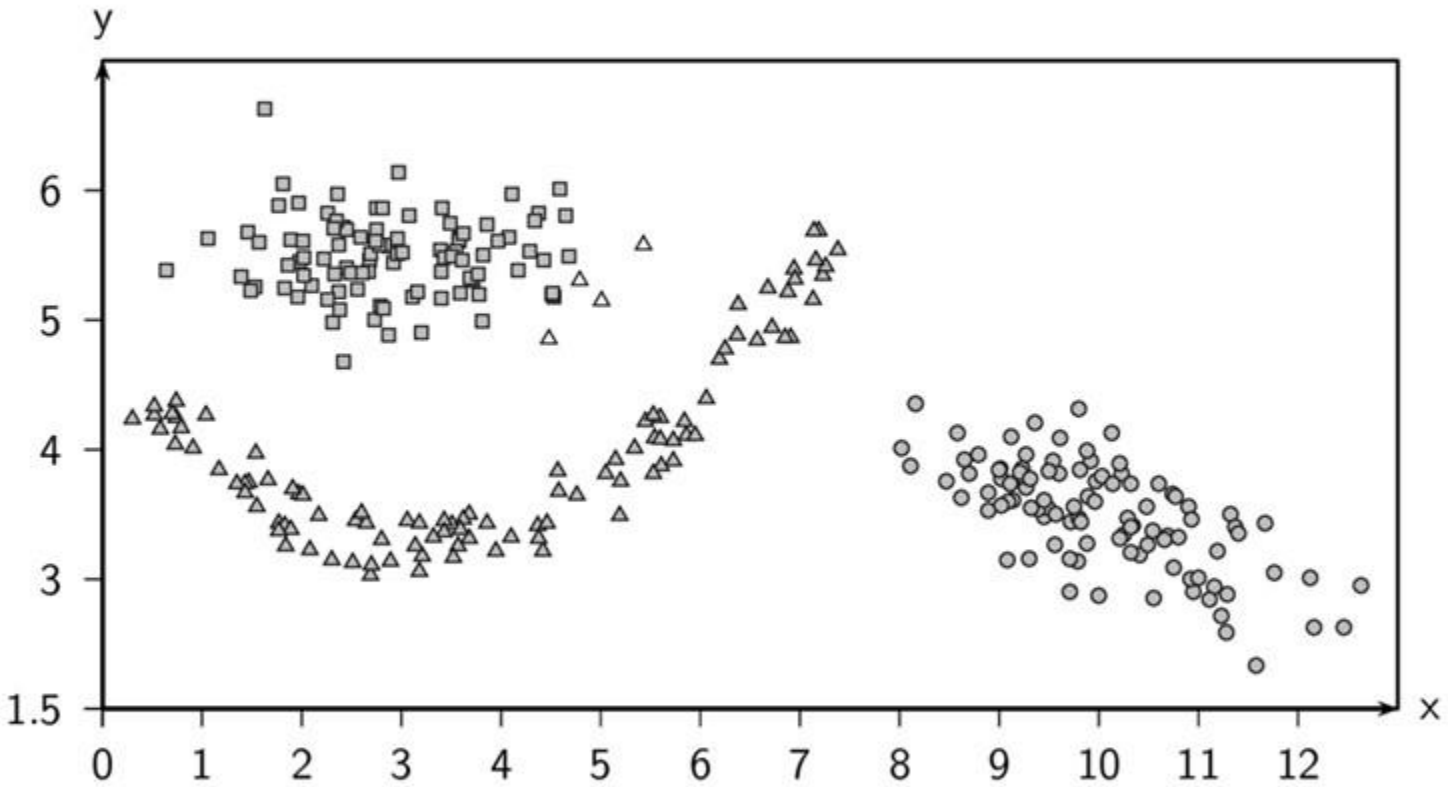
Using linear kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$ is equivalent to the K-means algorithm.



(a) Linear kernel: $t = 5$ iterations

Kernel K-means: Gaussian Kernel

Using the Gaussian kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right\}$, with $\sigma = 1.5$.



(b) Gaussian kernel: $t = 4$ Iterations

Expectation-Maximization Clustering

Gaussian Mixture Model

Let X_a denote the random variable corresponding to the ath attribute. Let $\boldsymbol{X} = (X_1, X_2, \ldots, X_d)$ denote the vector random variable across the d-attributes, with $\boldsymbol{x}_j$ being a data sample from $\boldsymbol{X}$.

We assume that each cluster C_i is characterized by a multivariate normal distribution

$$f_i(\boldsymbol{x}) = f(\boldsymbol{x}|\boldsymbol{\mu}_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left\{-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)}{2}\right\}$$

where the cluster mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and covariance matrix $\Sigma_i \in \mathbb{R}^{d \times d}$ are both unknown parameters.

The probability density function of $\boldsymbol{X}$ is given as a *Gaussian mixture model* over all the k clusters

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} f_i(\boldsymbol{x}) P(C_i) = \sum_{i=1}^{k} f(\boldsymbol{x}|\boldsymbol{\mu}_i, \Sigma_i) P(C_i)$$

where the prior probabilities $P(C_i)$ are called the *mixture parameters*, which must satisfy the condition $\sum_{i=1}^{k} P(C_i) = 1$.

Expectation-Maximization Clustering

Maximum Likelihood Estimation

We write the set of all the model parameters compactly as

$$\boldsymbol{\theta} = \{\boldsymbol{\mu}_1, \Sigma_1, P(C_1) \ldots, \boldsymbol{\mu}_k, \Sigma_k, P(C_k)\}$$

Given the dataset $\boldsymbol{D}$, we define the *likelihood* of $\boldsymbol{\theta}$ as the conditional probability of the data $\boldsymbol{D}$ given the model parameters $\boldsymbol{\theta}$

$$P(\boldsymbol{D}|\boldsymbol{\theta}) = \prod_{j=1}^{n} f(\boldsymbol{x}_j)$$

The goal of maximum likelihood estimation (MLE) is to choose the parameters $\boldsymbol{\theta}$ that maximize the likelihood. We do this by maximizing the log of the likelihood function

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{\ln P(\boldsymbol{D}|\boldsymbol{\theta})\}$$

where the *log-likelihood* function is given as

$$\ln P(\boldsymbol{D}|\boldsymbol{\theta}) = \sum_{j=1}^{n} \ln f(\boldsymbol{x}_j) = \sum_{j=1}^{n} \ln \left(\sum_{i=1}^{k} f(\boldsymbol{x}_j|\boldsymbol{\mu}_i, \Sigma_i) P(C_i) \right)$$

Expectation-Maximization Clustering

Directly maximizing the log-likelihood over $\boldsymbol{\theta}$ is hard. Instead, we can use the expectation-maximization (EM) approach for finding the maximum likelihood estimates for the parameters $\boldsymbol{\theta}$.

EM is a two-step iterative approach that starts from an initial guess for the parameters $\boldsymbol{\theta}$. Given the current estimates for $\boldsymbol{\theta}$, in the *expectation step* EM computes the cluster posterior probabilities $P(C_i|\boldsymbol{x}_j)$ via the Bayes theorem:

$$P(C_i|\boldsymbol{x}_j) = \frac{P(C_i \text{ and } \boldsymbol{x}_j)}{P(\boldsymbol{x}_j)} = \frac{P(\boldsymbol{x}_j|C_i)P(C_i)}{\sum_{a=1}^{k} P(\boldsymbol{x}_j|C_a)P(C_a)} = \frac{f_i(\boldsymbol{x}_j) \cdot P(C_i)}{\sum_{a=1}^{k} f_a(\boldsymbol{x}_j) \cdot P(C_a)}$$

In the *maximization step*, using the weights $P(C_i|\boldsymbol{x}_j)$ EM re-estimates $\boldsymbol{\theta}$, that is, it re-estimates the parameters $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $P(C_i)$ for each cluster C_i. The re-estimated mean is given as the weighted average of all the points, the re-estimated covariance matrix is given as the weighted covariance over all pairs of dimensions, and the re-estimated prior probability for each cluster is given as the fraction of weights that contribute to that cluster.

EM in One Dimension: Expectation Step

Let $\boldsymbol{D}$ comprise of a single attribute X, with each point $x_j \in \mathbb{R}$ a random sample from X. For the mixture model, we use univariate normals for each cluster:

$$f_i(x) = f(x|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\}$$

with the cluster parameters μ_i, σ_i^2, and $P(C_i)$.

Initialization: For each cluster C_i, with $i = 1, 2, \ldots, k$, we can randomly initialize the cluster parameters μ_i, σ_i^2, and $P(C_i)$.

Expectation Step: Given the mean μ_i, variance σ_i^2, and prior probability $P(C_i)$ for each cluster, the cluster posterior probability is computed as

$$w_{ij} = P(C_i|x_j) = \frac{f(x_j|\mu_i, \sigma_i^2) \cdot P(C_i)}{\sum_{a=1}^{k} f(x_j|\mu_a, \sigma_a^2) \cdot P(C_a)}$$

EM in One Dimension: Maximization Step

Given w_{ij} values, the re-estimated cluster mean is

$$\mu_i = \frac{\sum_{j=1}^{n} w_{ij} \cdot x_j}{\sum_{j=1}^{n} w_{ij}}$$

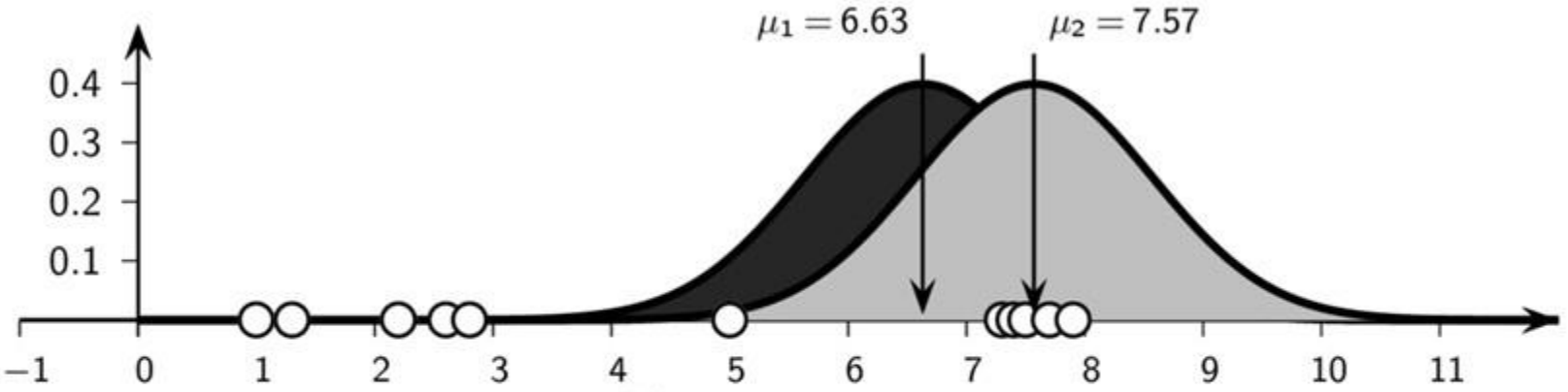
The re-estimated value of the cluster variance is computed as the weighted variance across all the points:

$$\sigma_i^2 = \frac{\sum_{j=1}^{n} w_{ij}(x_j - \mu_i)^2}{\sum_{j=1}^{n} w_{ij}}$$

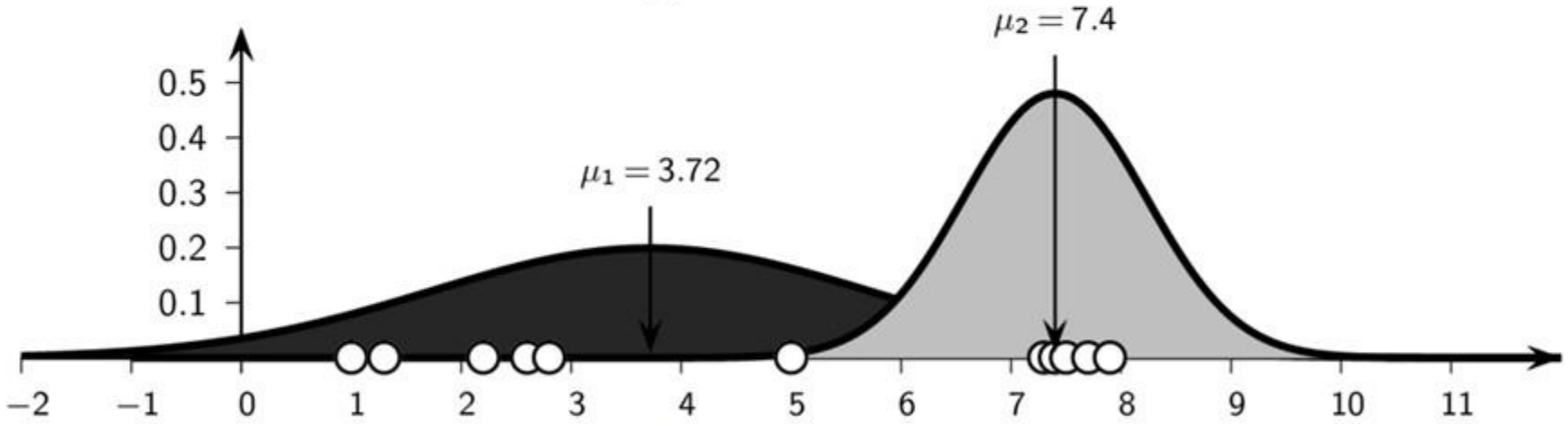
The prior probability of cluster C_i is re-estimated as

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{n}$$

EM in One Dimension

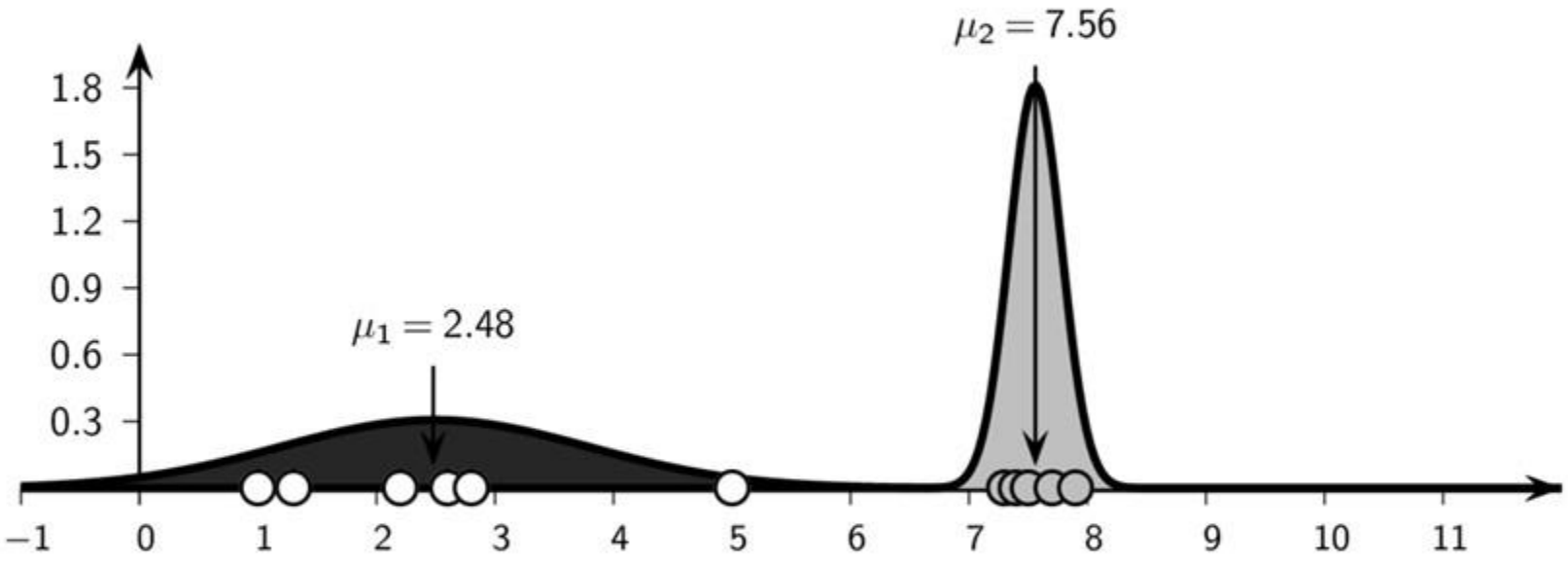


(a) Initialization: $t = 0$



(b) Iteration: $t = 1$

EM in One Dimension: Final Clusters



(c) Iteration: $t = 5$ (converged)

EM in d Dimensions

Each cluster we have to reestimate the $d \times d$ covariance matrix:

$$\Sigma_i = \begin{pmatrix} (\sigma_1^i)^2 & \sigma_{12}^i & \dots & \sigma_{1d}^i \\ \sigma_{21}^i & (\sigma_2^i)^2 & \dots & \sigma_{2d}^i \\ \vdots & \vdots & \ddots & \\ \sigma_{d1}^i & \sigma_{d2}^i & \dots & (\sigma_d^i)^2 \end{pmatrix}$$

It requires $O(d^2)$ parameters, which may be too many for reliable estimation. A simplification is to assume that all dimensions are independent, which leads to a diagonal covariance matrix:

$$\Sigma_i = \begin{pmatrix} (\sigma_1^i)^2 & 0 & \dots & 0 \\ 0 & (\sigma_2^i)^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \dots & (\sigma_d^i)^2 \end{pmatrix}$$

EM in d Dimensions

Expectation Step: Given $\boldsymbol{\mu}_i$, Σ_i, and $P(C_i)$, the posterior probability is given as

$$w_{ij} = P(C_i|\boldsymbol{x}_j) = \frac{f_i(\boldsymbol{x}_j) \cdot P(C_i)}{\sum_{a=1}^{k} f_a(\boldsymbol{x}_j) \cdot P(C_a)}$$

Maximization Step: Given the weights w_{ij}, in the maximization step, we re-estimate Σ_i, $\boldsymbol{\mu}_i$ and $P(C_i)$.
The mean $\boldsymbol{\mu}_i$ for cluster C_i can be estimated as

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{n} w_{ij} \cdot \boldsymbol{x}_j}{\sum_{j=1}^{n} w_{ij}}$$

The covariance matrix Σ_i is re-estimated via the outer-product form

$$\Sigma_i = \frac{\sum_{j=1}^{n} w_{ij}(\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n} w_{ij}}$$

The prior probability $P(C_i)$ for each cluster is

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{n}$$

Expectation-Maximization Clustering Algorithm

Expectation-Maximization $(\boldsymbol{D}, k, \epsilon)$**:**

$t \leftarrow 0$

Randomly initialize $\boldsymbol{\mu}_1^t, \ldots, \boldsymbol{\mu}_k^t$

$\Sigma_i^t \leftarrow \boldsymbol{I}, \forall i = 1, \ldots, k$

repeat

  $t \leftarrow t + 1$

  for $i = 1, \ldots, k$ *and* $j = 1, \ldots, n$ **do**

    $w_{ij} \leftarrow \frac{f(\boldsymbol{x}_j | \boldsymbol{\mu}_i, \Sigma_i) \cdot P(C_i)}{\sum_{a=1}^{k} f(\boldsymbol{x}_j | \boldsymbol{\mu}_a, \Sigma_a) \cdot P(C_a)}$ // posterior probability $P^t(C_i | \boldsymbol{x}_j)$

  for $i = 1, \ldots, k$ **do**

    $\boldsymbol{\mu}_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij} \cdot \boldsymbol{x}_j}{\sum_{j=1}^{n} w_{ij}}$ // re-estimate mean

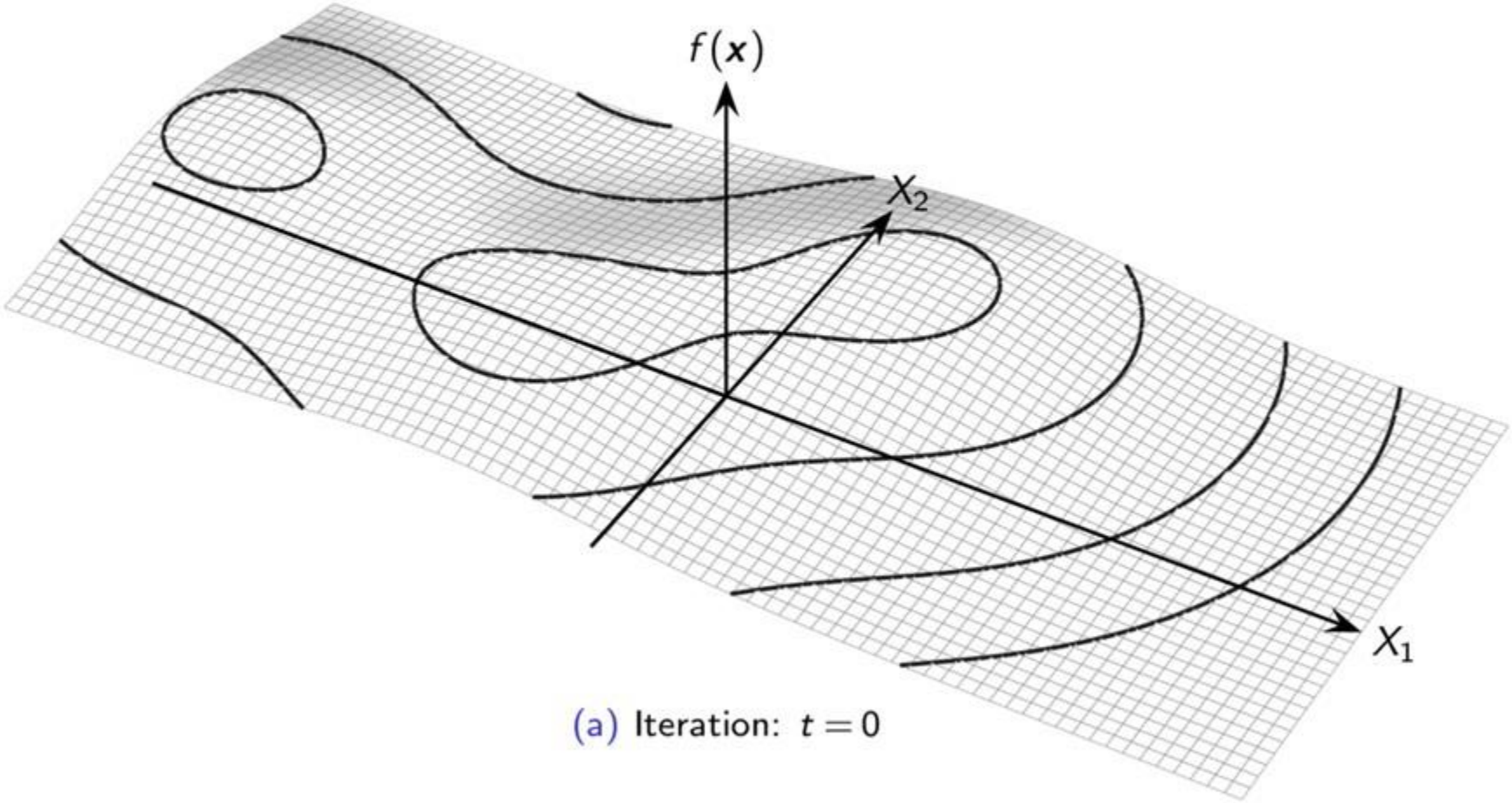
    $\Sigma_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij} (\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n} w_{ij}}$ // re-estimate covariance matrix

    $P^t(C_i) \leftarrow \frac{\sum_{j=1}^{n} w_{ij}}{n}$ // re-estimate priors

until $\sum_{i=1}^{k} \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$

EM Clustering in 2D

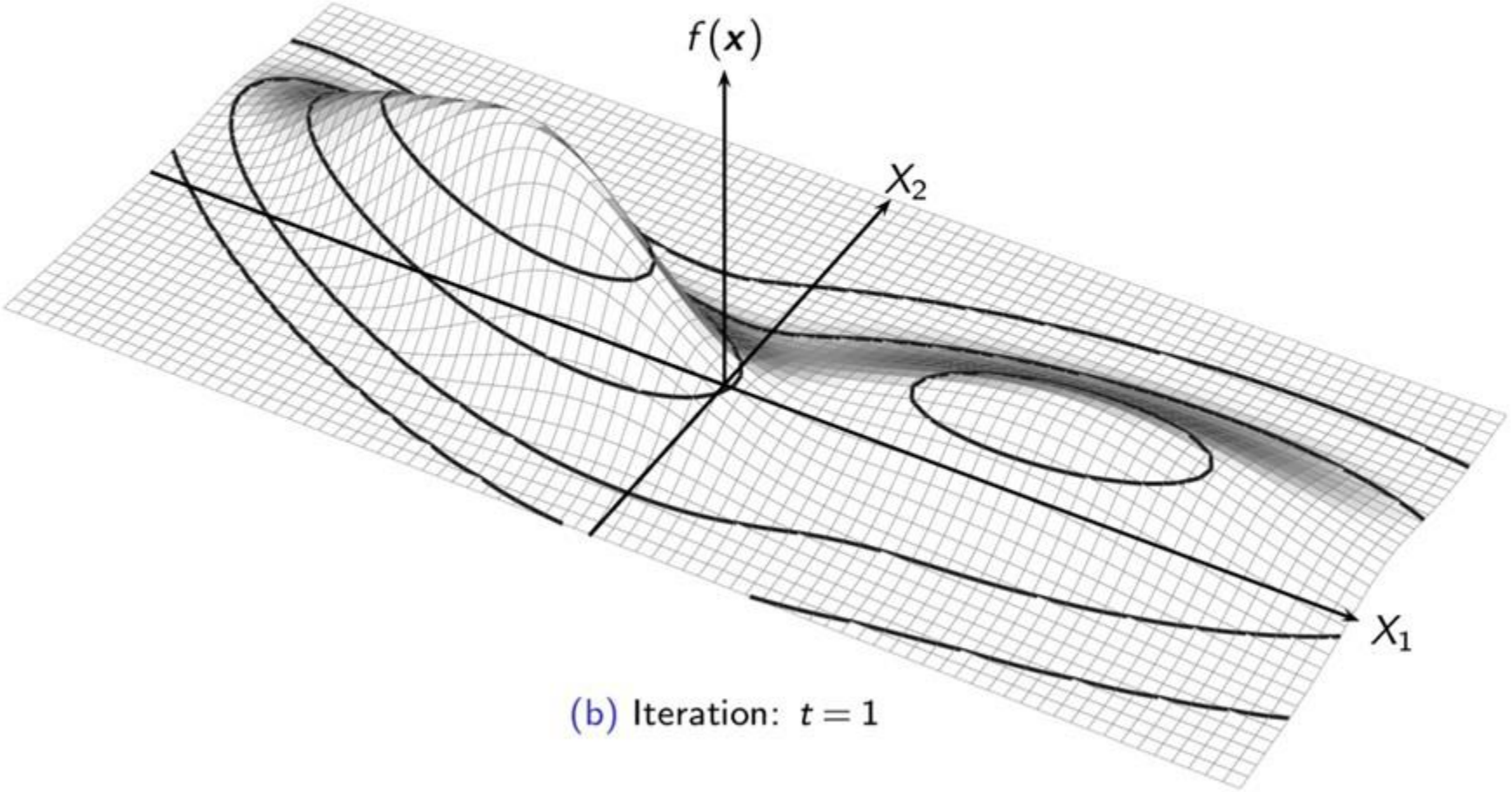
Mixture of $k = 3$ Gaussians



(a) Iteration: $t = 0$

EM Clustering in 2D

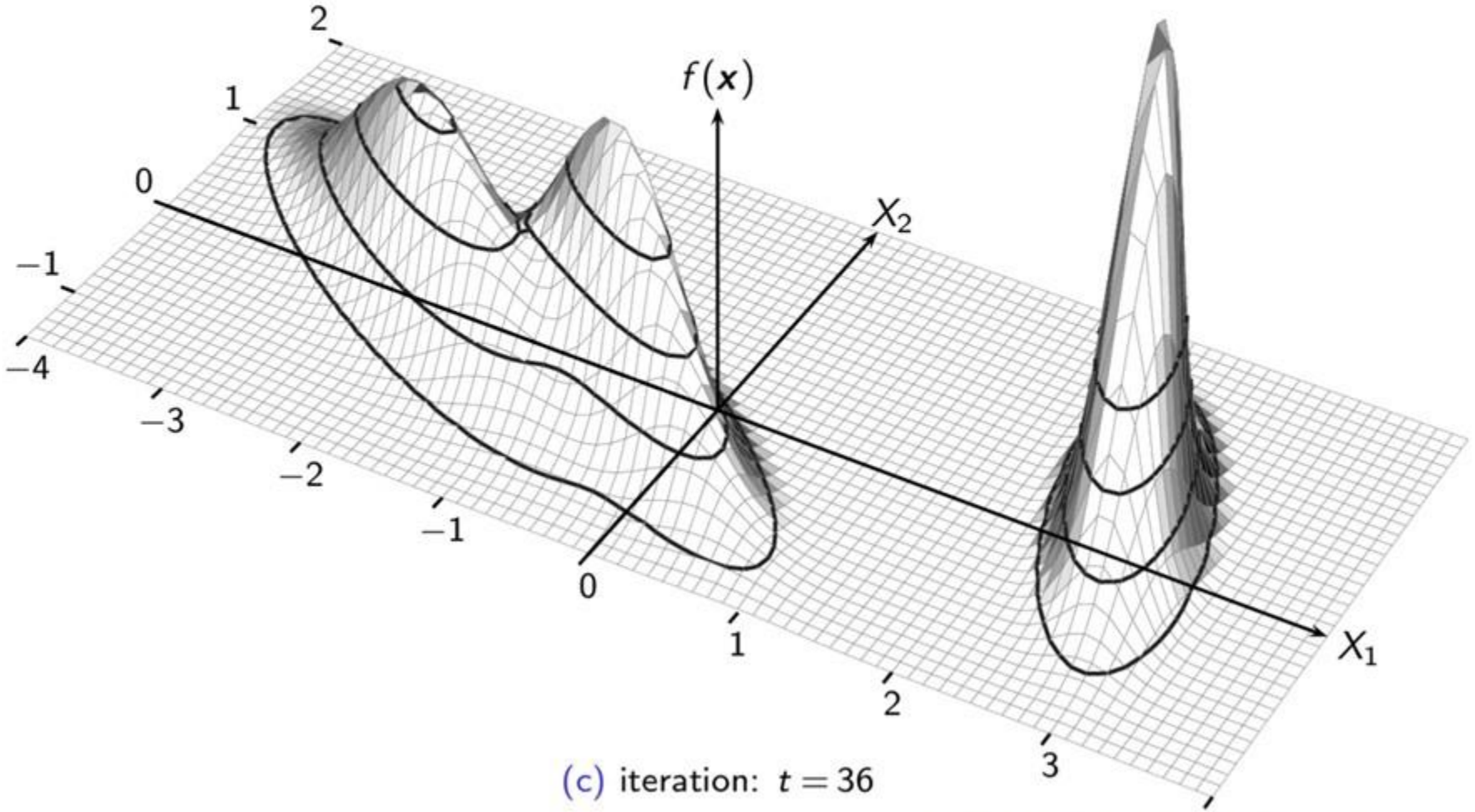
Mixture of $k = 3$ Gaussians



(b) Iteration: $t = 1$

EM Clustering in 2D

Mixture of $k = 3$ Gaussians



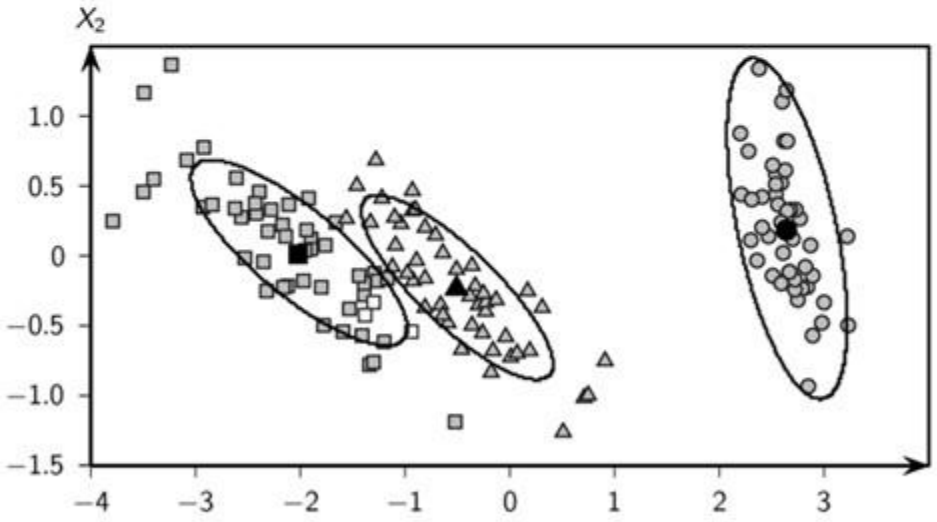
(c) iteration: $t = 36$

Iris Principal Components Data

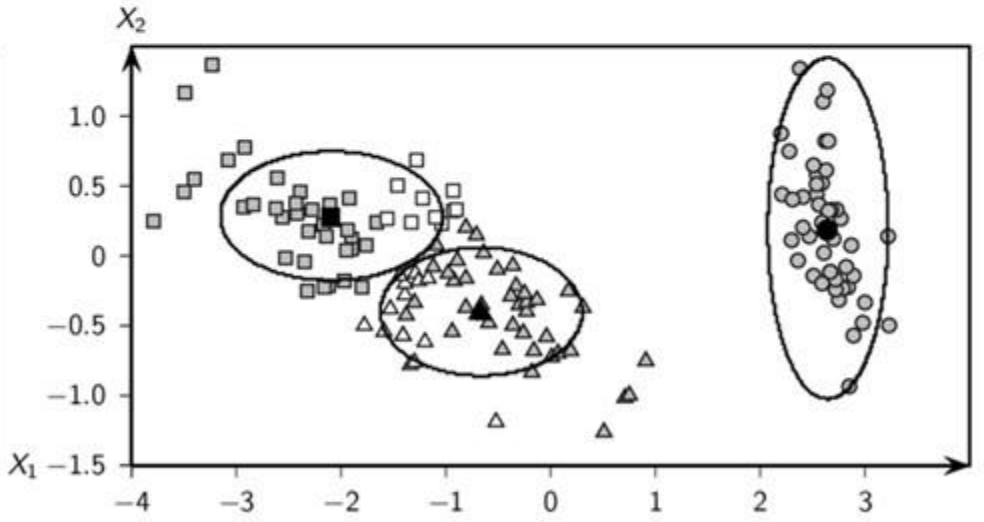
Full vs. Diagonal Covariance Matrix

The diagonal assumption leads to axis parallel contours for the normal density, contrasted with the rotated contours for the full covariance matrix.

The full matrix yields much better clustering, since the full covariance matrix results in 3 wrongly clustered points, whereas the diagonal covariance matrix results in 25.



(a) Full covariance matrix ($t = 36$)



(b) Diagonal covariance matrix ($t = 29$)

K-means as Specialization of EM

K-means can be considered as a special case of the EM algorithm, as follows:

$$P(\boldsymbol{x}_j|C_i) = \begin{cases} 1 & \text{if } C_i = \arg\min_{C_a} \left\{ \|\boldsymbol{x}_j - \boldsymbol{\mu}_a\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$

The posterior probability $P(C_i|\boldsymbol{x}_j)$ is given as

$$P(C_i|\boldsymbol{x}_j) = \frac{P(\boldsymbol{x}_j|C_i)P(C_i)}{\sum_{a=1}^{k} P(\boldsymbol{x}_j|C_a)P(C_a)}$$

If $P(\boldsymbol{x}_j|C_i) = 0$, then $P(C_i|\boldsymbol{x}_j) = 0$. Otherwise, if $P(\boldsymbol{x}_j|C_i) = 1$, then $P(\boldsymbol{x}_j|C_a) = 0$ $\forall a \neq i$, and thus $P(C_i|\boldsymbol{x}_j) = \frac{1 \cdot P(C_i)}{1 \cdot P(C_i)} = 1$.

$$P(C_i|\boldsymbol{x}_j) = \begin{cases} 1 & \text{if } \boldsymbol{x}_j \in C_i, \text{i.e., if } C_i = \arg\min_{C_a} \left\{ \|\boldsymbol{x}_j - \boldsymbol{\mu}_a\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$

The parameters are $\boldsymbol{\mu}_i$ and $P(C_i)$ and the covariance matrix is not used.

Data mining and Machine learning

Part 14. Hierarchical Clustering

Hierarchical Clustering

The goal of hierarchical clustering is to create a sequence of nested partitions, which can be conveniently visualized via a tree or hierarchy of clusters, also called the cluster *dendrogram*.

The clusters in the hierarchy range from the fine-grained to the coarse-grained – the lowest level of the tree (the leaves) consists of each point in its own cluster, whereas the highest level (the root) consists of all points in one cluster.

Agglomerative hierarchical clustering methods work in a bottom-up manner. Starting with each of the n points in a separate cluster, they repeatedly merge the most similar pair of clusters until all points are members of the same cluster.

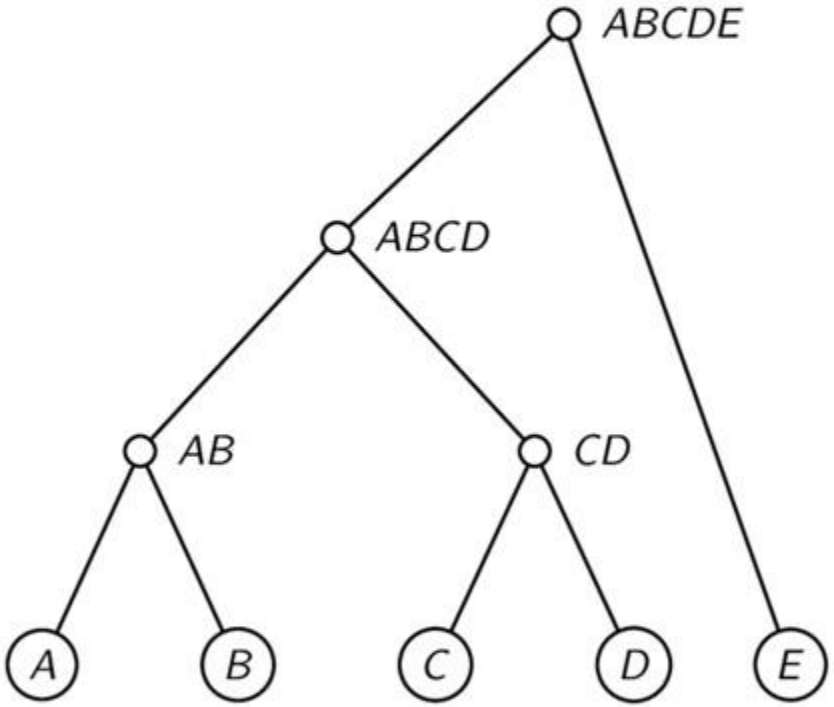
Hierarchical Clustering: Nested Partitions

Given a dataset $\boldsymbol{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ is a partition of $\boldsymbol{D}$.

A clustering $\mathcal{A} = \{A_1, \ldots, A_r\}$ is said to be nested in another clustering $\mathcal{B} = \{B_1, \ldots, B_s\}$ if and only if $r > s$, and for each cluster $A_i \in \mathcal{A}$, there exists a cluster $B_j \in \mathcal{B}$, such that $A_i \subseteq B_j$.

Hierarchical clustering yields a sequence of n nested partitions $\mathcal{C}_1, \ldots, \mathcal{C}_n$. The clustering $\mathcal{C}_{t-1}$ is nested in the clustering $\mathcal{C}_t$. The cluster dendrogram is a rooted binary tree that captures this nesting structure, with edges between cluster $C_i \in \mathcal{C}_{t-1}$ and cluster $C_j \in \mathcal{C}_t$ if C_i is nested in C_j, that is, if $C_i \subset C_j$.

Hierarchical Clustering Dendrogram



The dendrogram represents the following sequence of nested partitions:

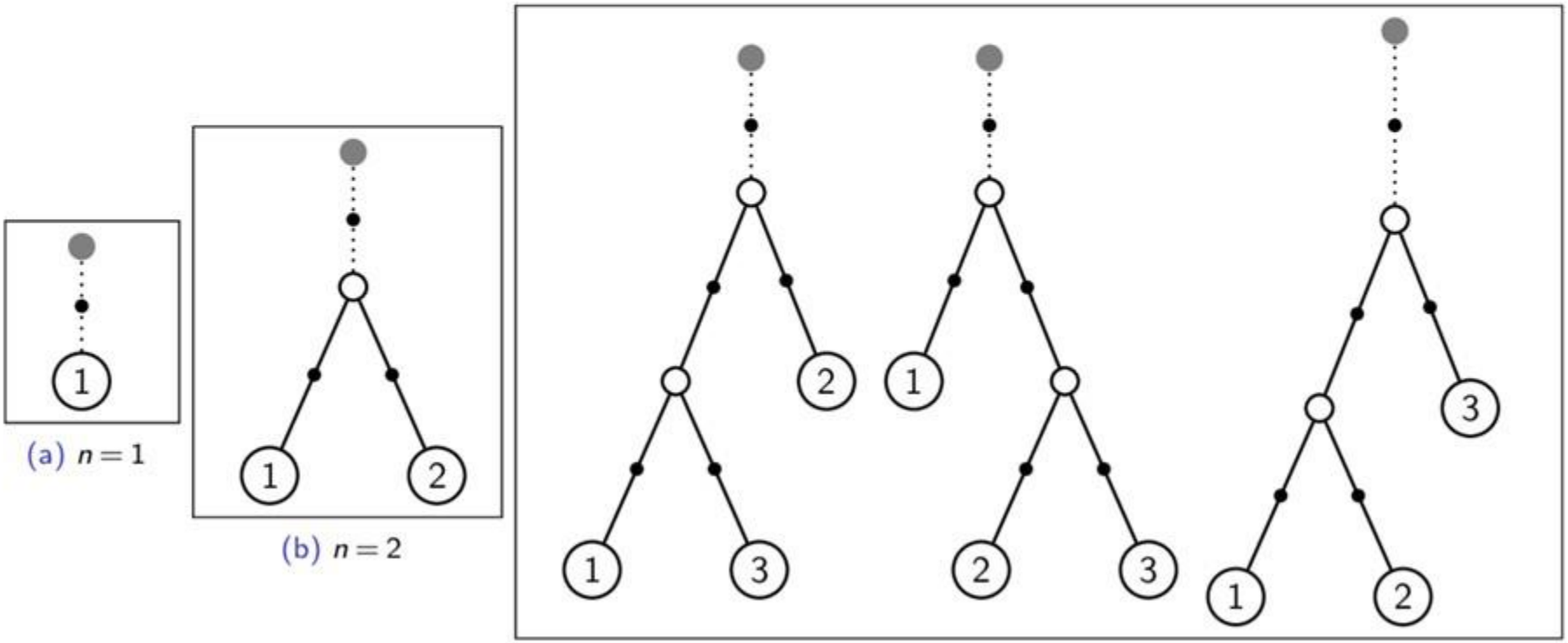
Clustering	Clusters
$\mathcal{C}_1$	$\{A\},\{B\},\{C\},\{D\},\{E\}$
$\mathcal{C}_2$	$\{AB\},\{C\},\{D\},\{E\}$
$\mathcal{C}_3$	$\{AB\},\{CD\},\{E\}$
$\mathcal{C}_4$	$\{ABCD\},\{E\}$
$\mathcal{C}_5$	$\{ABCDE\}$

with $\mathcal{C}_{t-1} \subset \mathcal{C}_t$ for $t = 2, \ldots, 5$. We assume that A and B are merged before C and D.

Number of Hierarchical Clusterings

The total number of different dendrograms with n leaves is given as:

$$\prod_{m=1}^{n-1}(2m-1) = 1 \times 3 \times 5 \times 7 \times \cdots \times (2n-3) = (2n-3)!!$$



(a) $n = 1$

(b) $n = 2$

(c) $n = 3$

Agglomerative Hierarchical Clustering

In agglomerative hierarchical clustering, we begin with each of the n points in a separate cluster. We repeatedly merge the two closest clusters until all points are members of the same cluster.

Given a set of clusters $\mathcal{C} = \{C_1, C_2, .., C_m\}$, we find the *closest* pair of clusters C_i and C_j and merge them into a new cluster $C_{ij} = C_i \cup C_j$.

Next, we update the set of clusters by removing C_i and C_j and adding C_{ij}, as follows $\mathcal{C} = \left(\mathcal{C} \setminus \{C_i, C_j\}\right) \cup \{C_{ij}\}$.

We repeat the process until $\mathcal{C}$ contains only one cluster. If specified, we can stop the merging process when there are exactly k clusters remaining.

Agglomerative Hierarchical Clustering Algorithm

AgglomerativeClustering$(\boldsymbol{D}, k)$**:**

1 $\mathcal{C} \leftarrow \{C_i = \{\boldsymbol{x}_i\} \mid \boldsymbol{x}_i \in \boldsymbol{D}\}$ // Each point in separate cluster
2 $\Delta \leftarrow \{\delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \colon \boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{D}\}$ // Compute distance matrix
3 **repeat**
4 Find the closest pair of clusters $C_i, C_j \in \mathcal{C}$
5 $C_{ij} \leftarrow C_i \cup C_j$ // Merge the clusters
6 $\mathcal{C} \leftarrow \big(\mathcal{C} \setminus \{C_i, C_j\}\big) \cup \{C_{ij}\}$ // Update the clustering
7 Update distance matrix Δ to reflect new clustering
8 **until** $|\mathcal{C}| = k$

Distance between Clusters

Single, Complete and Average

A typical distance between two points is the Euclidean distance or L_2-*norm*

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 = \left(\sum_{i=1}^{d} (x_i - y_i)^2\right)^{1/2}$$

Single Link: The minimum distance between a point in C_i and a point in C_j

$$\delta(C_i, C_j) = \min\{\|\boldsymbol{x} - \boldsymbol{y}\| \mid \boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j\}$$

Complete Link: The maximum distance between points in the two clusters:

$$\delta(C_i, C_j) = \max\{\|\boldsymbol{x} - \boldsymbol{y}\| \mid \boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j\}$$

Group Average: The average pairwise distance between points in C_i and C_j:

$$\delta(C_i, C_j) = \frac{\sum_{\boldsymbol{x} \in C_i} \sum_{\boldsymbol{y} \in C_j} \|\boldsymbol{x} - \boldsymbol{y}\|}{n_i \cdot n_j}$$

Distance between Clusters: Mean and Ward's

Mean Distance: The distance between two clusters is defined as the distance between the means or centroids of the two clusters:

$$\delta(C_i, C_j) = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$$

Minimum Variance or Ward's Method: The distance between two clusters is defined as the increase in the sum of squared errors (SSE) when the two clusters are merged, where the SSE for a given cluster C_i is given as

$$\delta(C_i, C_j) = \Delta SSE_{ij} = SSE_{ij} - SSE_i - SSE_j$$

where $SSE_i = \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$. After simplification, we get:

$$\delta(C_i, C_j) = \left(\frac{n_i n_j}{n_i + n_j}\right) \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2$$

Ward's measure is therefore a weighted version of the mean distance measure.

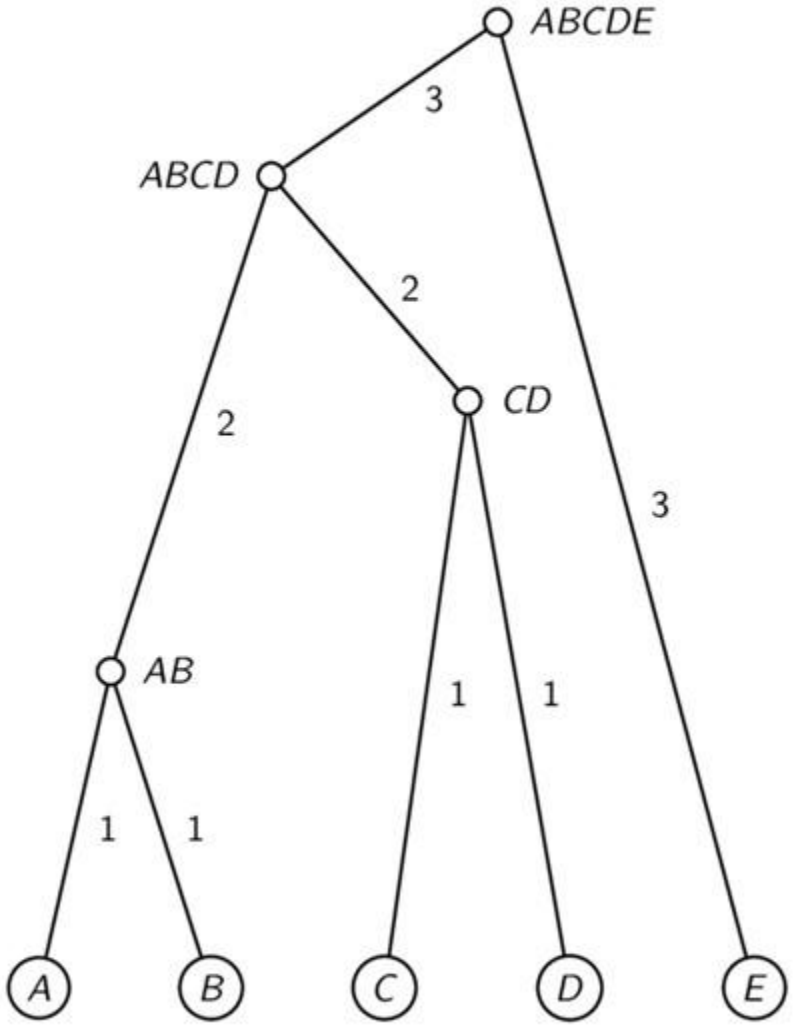
Single Link Agglomerative Clustering

δ	E
$ABCD$	(3)

δ	CD	E
AB	(2)	3
CD		3

δ	C	D	E
AB	3	2	3
C		(1)	3
D			5

δ	B	C	D	E
A	(1)	3	2	4
B		3	2	3
C			1	3
D				5



Lance–Williams Formula

Whenever two clusters C_i and C_j are merged into C_{ij}, we need to update the distance matrix by recomputing the distances from the newly created cluster C_{ij} to all other clusters C_r ($r \neq i$ and $r \neq j$).

The Lance–Williams formula provides a general equation to recompute the distances for all of the cluster proximity measures

$$\delta(C_{ij}, C_r) = \alpha_i \cdot \delta(C_i, C_r) + \alpha_j \cdot \delta(C_j, C_r) + \beta \cdot \delta(C_i, C_j) + \gamma \cdot \left|\delta(C_i, C_r) - \delta(C_j, C_r)\right|$$

The coefficients $\alpha_i, \alpha_j, \beta$, and γ differ from one measure to another.

Lance–Williams Formulas for Cluster Proximity

Measure	α_i	α_j	β	γ
Single link	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete link	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
Group average	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Mean distance	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$\frac{-n_i \cdot n_j}{(n_i+n_j)^2}$	0
Ward's measure	$\frac{n_i+n_r}{n_i+n_j+n_r}$	$\frac{n_j+n_r}{n_i+n_j+n_r}$	$\frac{-n_r}{n_i+n_j+n_r}$	0

Lance–Williams Formulas for Cluster Proximity

Single link: Arithmetical trick to find the minimum.

$$\delta(C_{ij}, C_r) = \frac{\delta(C_i, C_r)}{2} + \frac{\delta(C_j, C_r)}{2} - \frac{\left|\delta(C_i, C_r) - \delta(C_j, C_r)\right|}{2}$$

Complete link: Arithmetical trick to find the maximum.

$$\delta(C_{ij}, C_r) = \frac{\delta(C_i, C_r)}{2} + \frac{\delta(C_j, C_r)}{2} + \frac{\left|\delta(C_i, C_r) - \delta(C_j, C_r)\right|}{2}$$

Group average: Weight the distance by the cluster size.

$$\delta(C_{ij}, C_r) = \frac{n_i}{n_i + n_j} \cdot \delta(C_i, C_r) + \frac{n_j}{n_i + n_j} \cdot \delta(C_j, C_r)$$

Lance–Williams Formulas for Cluster Proximity

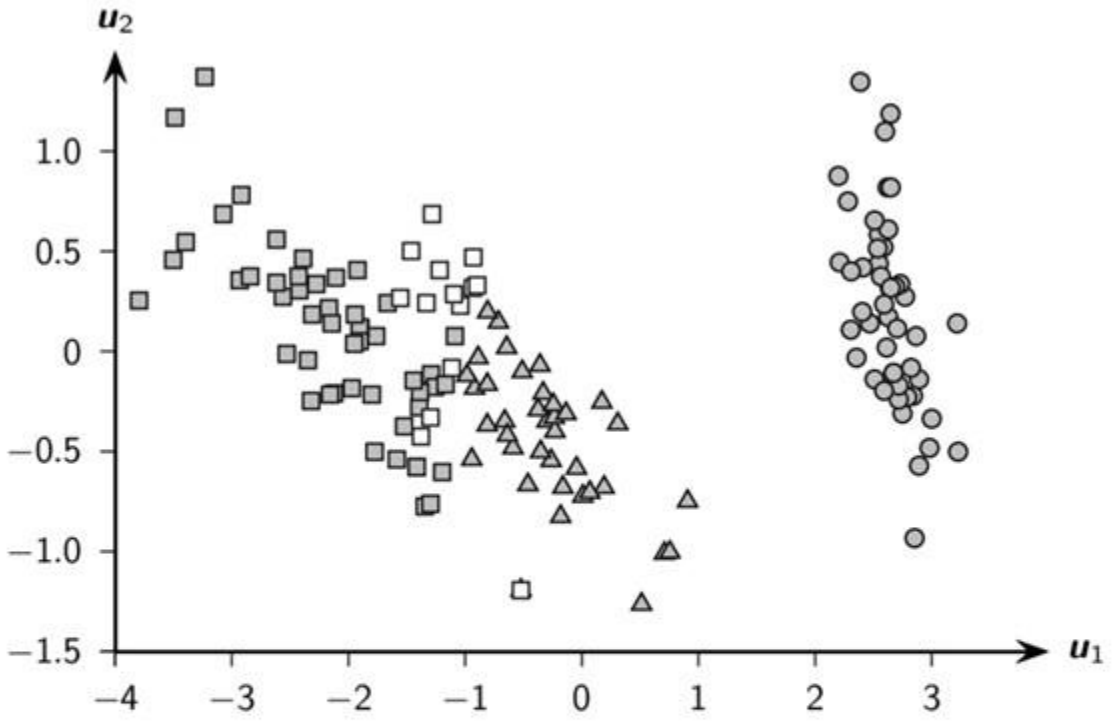
Mean distance: The new centroid is in the line defined by μ_i and μ_j, and its distance to μ_r has to be adjusted by $\frac{n_i \cdot n_j}{(n_i+n_j)^2}$.

$$\delta(C_{ij}, C_r) = \frac{n_i}{n_i + n_j} \cdot \delta(C_i, C_r) + \frac{n_j}{n_i + n_j} \cdot \delta(C_j, C_r) + \frac{-n_i \cdot n_j}{(n_i + n_j)^2} \cdot \delta(C_i, C_j)$$

Ward's measure: The ΔSSE of the new cluster is a weigthed sum of the $\Delta SSEs$ of the original clusters, adjusted by the fact that n_r was considered twice.

$$\delta(C_{ij}, C_r) = \frac{n_i + n_r}{n_i + n_j + n_r} \cdot \delta(C_i, C_r) + \frac{n_j + n_r}{n_i + n_j + n_r} \cdot \delta(C_j, C_r) + \frac{-n_r}{n_i + n_j + n_r} \cdot \delta(C_i, C_j)$$

Iris Dataset: Complete Link Clustering



Contingency Table:

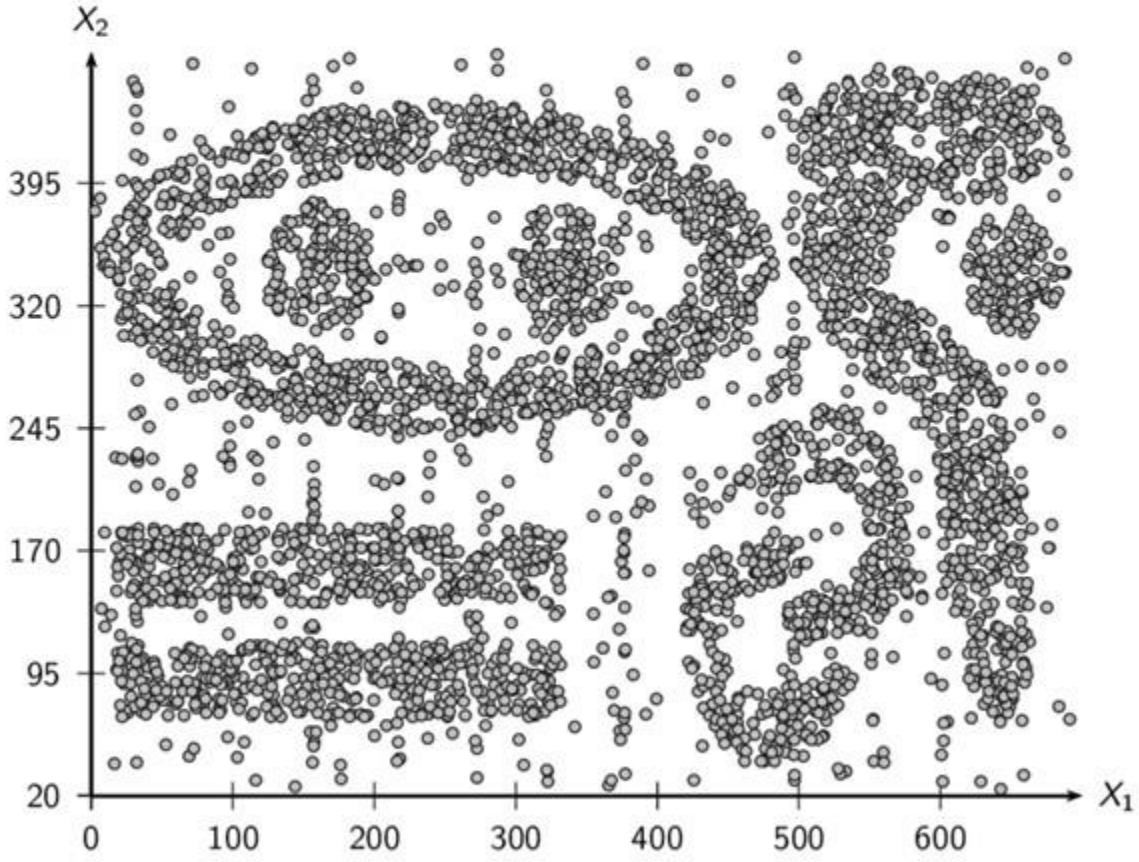
	`iris-setosa`	`iris-virginica`	`iris-versicolor`
C_1 (circle)	50	0	0
C_2 (triangle)	0	1	36
C_3 (square)	0	49	14

Data mining and Machine learning

Part 15. Density-based Clustering

Density-based Clustering

Density-based methods are able to mine nonconvex clusters, where distance-based methods may have difficulty.



The DBSCAN Approach

Neighborhood and Core Points

Define a ball of radius ϵ around a point $\boldsymbol{x} \in \mathbb{R}^d$, called the ϵ-*neighborhood* of $\boldsymbol{x}$:

$$N_\epsilon(\boldsymbol{x}) = B_d(\boldsymbol{x}, \epsilon) = \{\boldsymbol{y} \mid \delta(\boldsymbol{x}, \boldsymbol{y}) \leq \epsilon\}$$

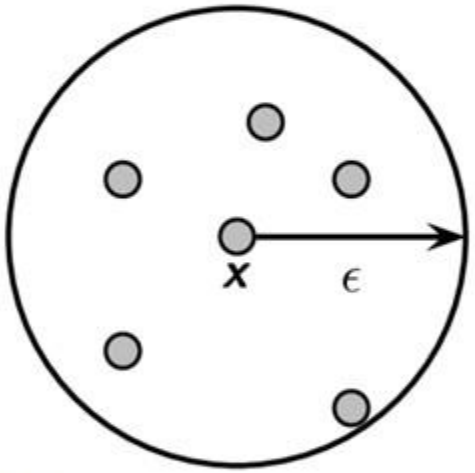
Here $\delta(\boldsymbol{x}, \boldsymbol{y})$ represents the distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$. which is usually assumed to be the Euclidean

We say that $\boldsymbol{x}$ is a *core point* if there are at least *minpts* points in its ϵ-neighborhood, i.e., if $|N_\epsilon(\boldsymbol{x})| \geq minpts$.

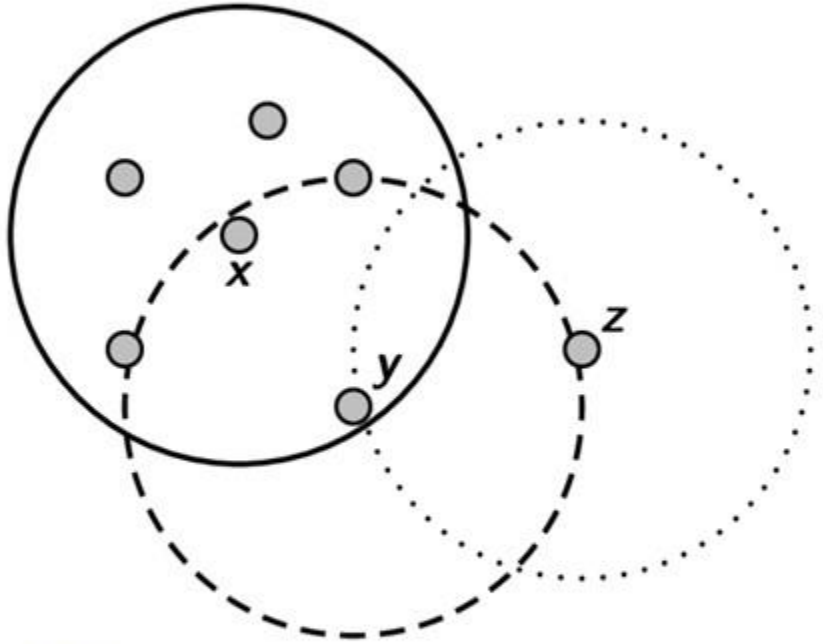
A *border point* does not meet the *minpts* threshold, i.e., $|N_\epsilon(\boldsymbol{x})| < minpts$, but it belongs to the ϵ-neighborhood of some core point $\boldsymbol{z}$, that is, $\boldsymbol{x} \in N_\epsilon(\boldsymbol{z})$.

If a point is neither a core nor a border point, then it is called a *noise point* or an outlier.

Core, Border and Noise Points



(a) Neighborhood of a Point



(b) Core, Border, and Noise Points

The DBSCAN Approach

Reachability and Density-based Cluster

A point $\boldsymbol{x}$ is *directly density reachable* from another point $\boldsymbol{y}$ if $\boldsymbol{x} \in N_\epsilon(\boldsymbol{y})$ and $\boldsymbol{y}$ is a core point.

A point $\boldsymbol{x}$ is *density reachable* from $\boldsymbol{y}$ if there exists a chain of points, $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_l$, such that $\boldsymbol{x} = \boldsymbol{x}_0$ and $\boldsymbol{y} = \boldsymbol{x}_l$, and $\boldsymbol{x}_i$ is directly density reachable from $\boldsymbol{x}_{i-1}$ for all $i = 1, \ldots, l$. In other words, there is set of core points leading from $\boldsymbol{y}$ to $\boldsymbol{x}$.

Two points $\boldsymbol{x}$ and $\boldsymbol{y}$ are *density connected* if there exists a core point $\boldsymbol{z}$, such that both $\boldsymbol{x}$ and $\boldsymbol{y}$ are density reachable from $\boldsymbol{z}$.

A *density-based cluster* is defined as a maximal set of density connected points.

DBSCAN Density-based Clustering Algorithm

DBSCAN computes the ϵ-neighborhood $N_\epsilon(\boldsymbol{x}_i)$ for each point $\boldsymbol{x}_i$ in the dataset $\boldsymbol{D}$, and checks if it is a core point. It also sets the cluster id $id(\boldsymbol{x}_i) = \emptyset$ for all points, indicating that they are not assigned to any cluster.

Starting from each unassigned core point, the method recursively finds all its density connected points, which are assigned to the same cluster.

Some border point may be reachable from core points in more than one cluster; they may either be arbitrarily assigned to one of the clusters or to all of them (if overlapping clusters are allowed).

Those points that do not belong to any cluster are treated as outliers or noise.

Each DBSCAN cluster is a maximal connected component over the core point graph.

DBSCAN is sensitive to the choice of ϵ, in particular if clusters have different densities. The overall complexity of DBSCAN is $O(n^2)$.

DBSCAN Algorithm

dbscan ($\boldsymbol{D}$, ϵ, *minpts*):

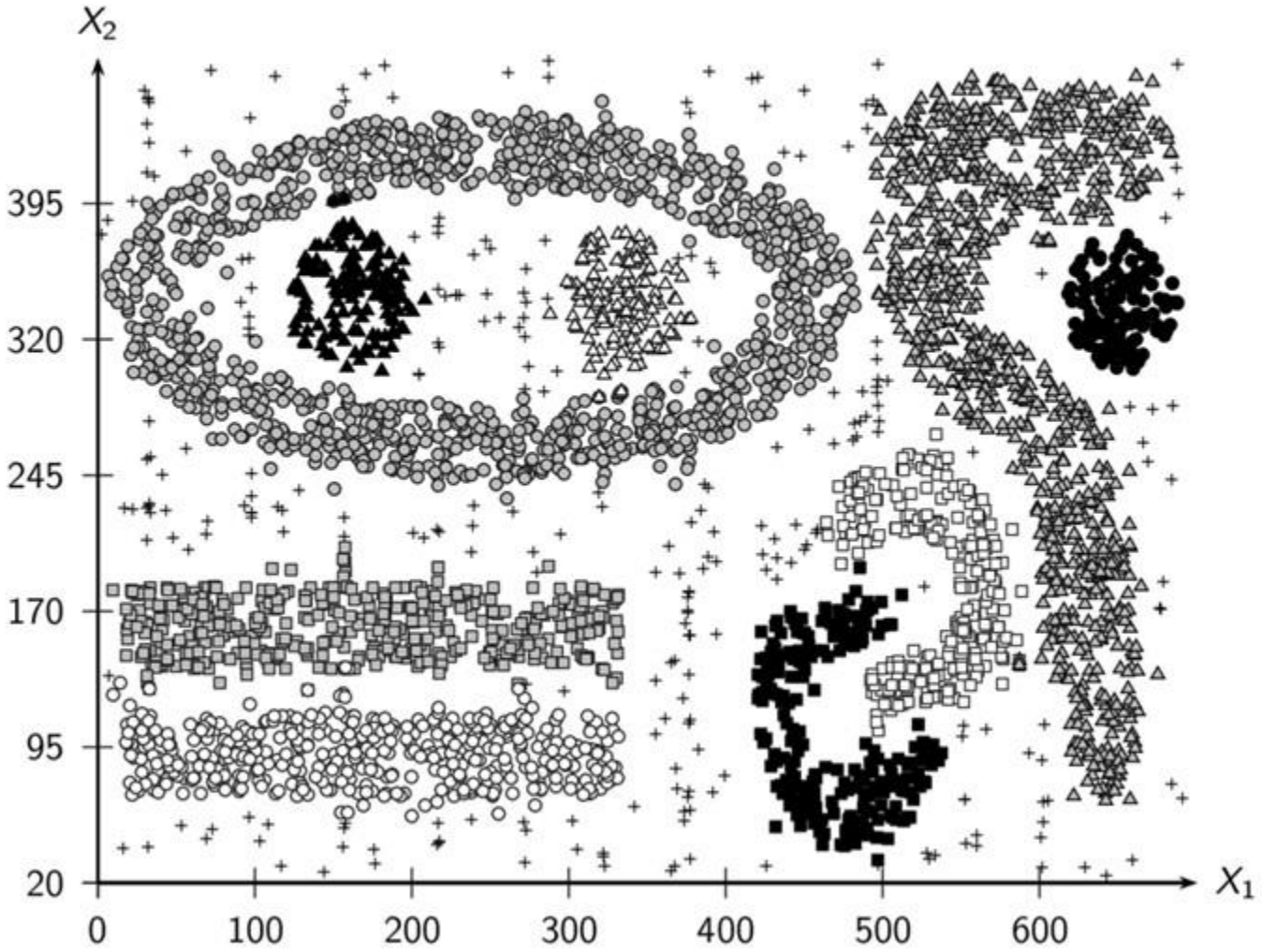
1 $Core \leftarrow \emptyset$
2 **foreach** $\boldsymbol{x}_i \in \boldsymbol{D}$ **do** // Find the core points
3 Compute $N_\epsilon(\boldsymbol{x}_i)$
4 $id(\boldsymbol{x}_i) \leftarrow \emptyset$ // cluster id for $\boldsymbol{x}_i$
5 **if** $N_\epsilon(\boldsymbol{x}_i) \geq minpts$ **then** $Core \leftarrow Core \cup \{\boldsymbol{x}_i\}$
6 $k \leftarrow 0$ // cluster id
7 **foreach** $\boldsymbol{x}_i \in Core$, *such that* $id(\boldsymbol{x}_i) = \emptyset$ **do**
8 $k \leftarrow k + 1$
9 $id(\boldsymbol{x}_i) \leftarrow k$ // assign $\boldsymbol{x}_i$ to cluster id k
10 DensityConnected $(\boldsymbol{x}_i, k)$
11 $\mathcal{C} \leftarrow \{C_i\}_{i=1}^{k}$, where $C_i \leftarrow \{\boldsymbol{x} \in \boldsymbol{D} \mid id(\boldsymbol{x}) = i\}$
12 $Noise \leftarrow \{\boldsymbol{x} \in \boldsymbol{D} \mid id(\boldsymbol{x}) = \emptyset\}$
13 $Border \leftarrow \boldsymbol{D} \setminus \{Core \cup Noise\}$
14 **return** $\mathcal{C}, Core, Border, Noise$

DensityConnected ($\boldsymbol{x}$, k):

15 **foreach** $\boldsymbol{y} \in N_\epsilon(\boldsymbol{x})$ **do**
16 $id(\boldsymbol{y}) \leftarrow k$ // assign $\boldsymbol{y}$ to cluster id k
17 **if** $\boldsymbol{y} \in Core$ **then** DensityConnected $(\boldsymbol{y}, k)$

Density-based Clusters

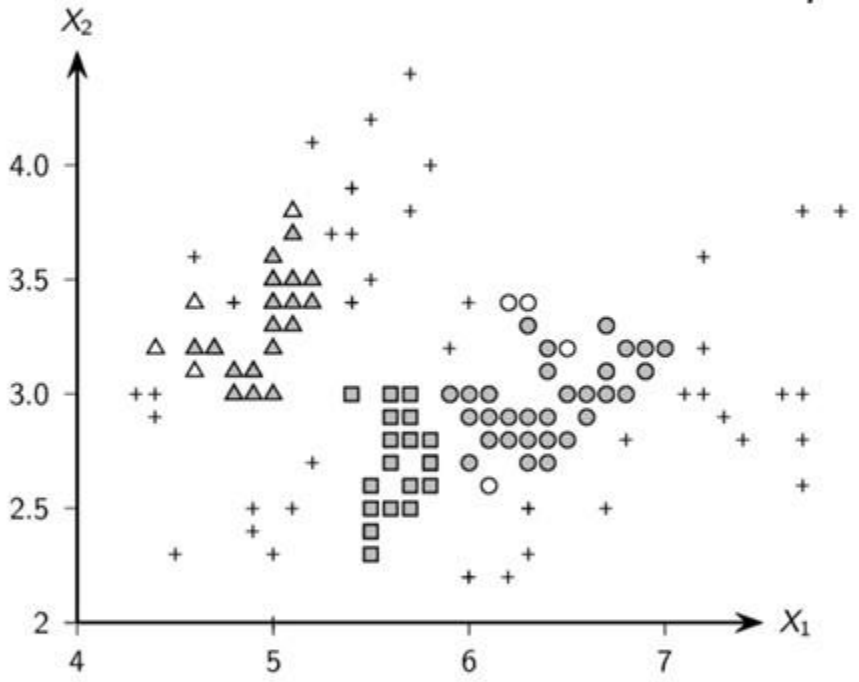
$\epsilon = 15$ and $minpts = 10$



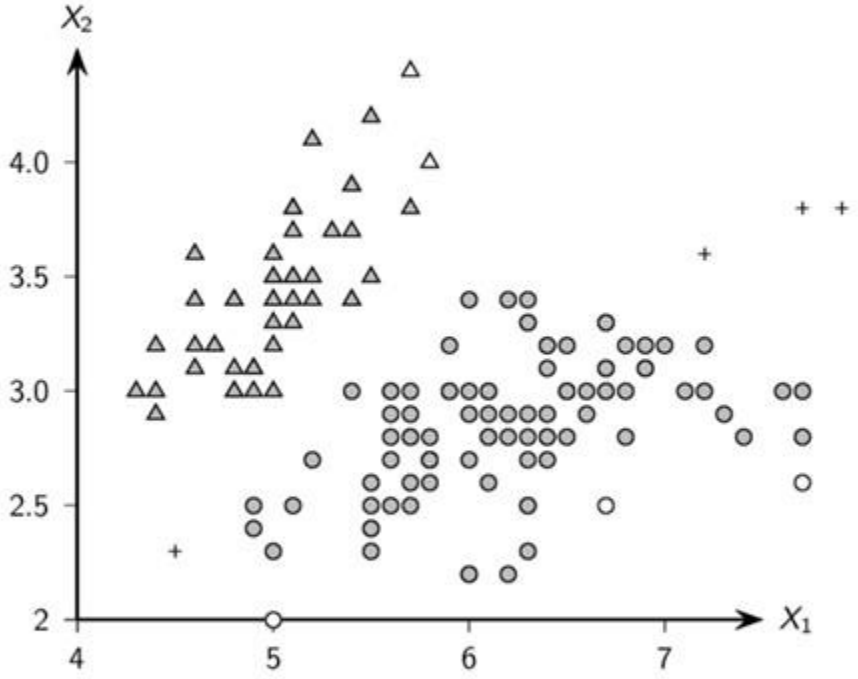
DBSCAN Clustering

Iris Dataset

There is a trade off between ϵ and *minpts*.



(a) $\epsilon = 0.2$, *minpts* $= 5$



(b) $\epsilon = 0.36$, *minpts* $= 3$

Kernel Density Estimation

There is a close connection between density-based clustering and density estimation. The goal of density estimation is to determine the unknown probability density function by finding the dense regions of points, which can in turn be used for clustering.

Kernel density estimation is a nonparametric technique that does not assume any fixed probability model of the clusters. Instead, it tries to directly infer the underlying probability density at each point in the dataset.

Univariate Density Estimation

Assume that X is a continuous random variable, and let $x_1, x_2, \ldots, x_n$ be a random sample. We directly estimate the cumulative distribution function from the data by counting how many points are less than or equal to x:

$$\hat{F}(x) = \frac{1}{n} \sum_{i=1}^{n} I(x_i \leq x)$$

where I is an indicator function.

We estimate the density function by taking the derivative of $\hat{F}(x)$

$$\hat{f}(x) = \frac{\hat{F}\left(x + \frac{h}{2}\right) - \hat{F}\left(x - \frac{h}{2}\right)}{h} = \frac{k/n}{h} = \frac{k}{nh}$$

where k is the number of points that lie in the window of width h centered at x. The density estimate is the ratio of the fraction of the points in the window (k/n) to the volume of the window (h).

Kernel Estimator

Kernel density estimation relies on a *kernel function* K that is non-negative, symmetric, and integrates to 1, that is, $K(x) \geq 0$, $K(-x) = K(x)$ for all values x, and $\int K(x)dx = 1$.

Discrete Kernel Define the **discrete kernel** function K, that computes the number of points in a window of width h

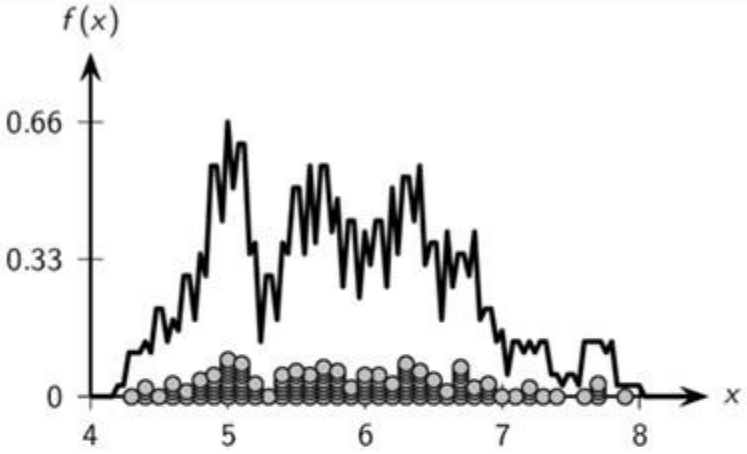
$$K(z) = \begin{cases} 1 & \text{If } |z| \leq \frac{1}{2} \\ 0 & \text{Otherwise} \end{cases}$$

The density estimate $\hat{f}(x)$ can be rewritten in terms of the kernel function as follows:

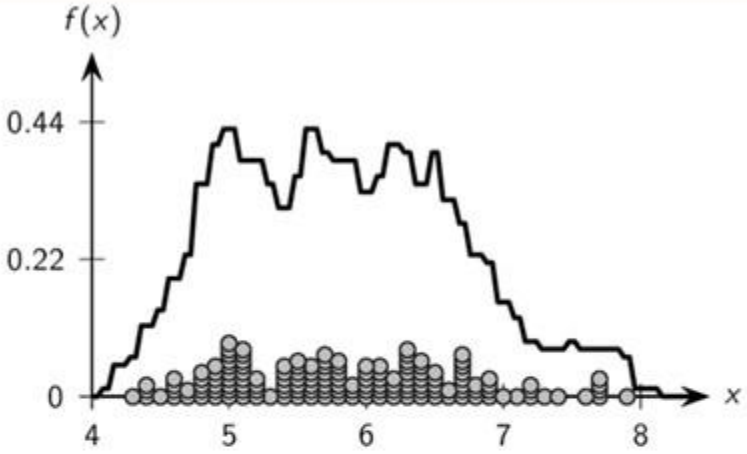
$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Kernel Density Estimation

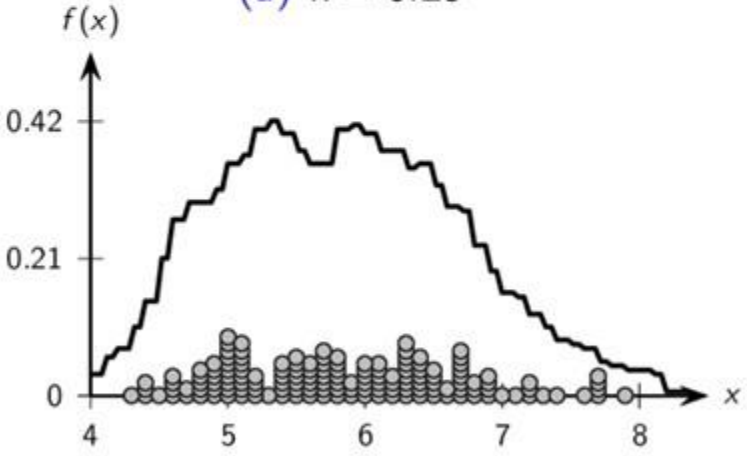
Discrete Kernel (Iris 1D)



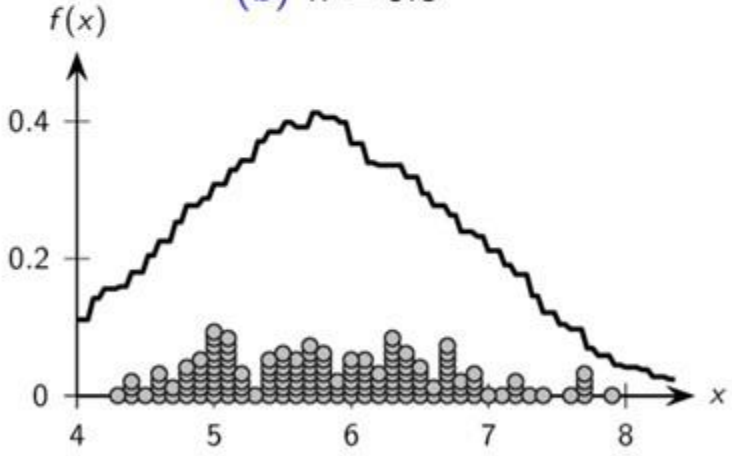
(a) $h = 0.25$



(b) $h = 0.5$



(c) $h = 1.0$



(d) $h = 2.0$

The discrete kernel yields a non-smooth (or jagged) density function.

Kernel Density Estimation

Gaussian Kernel

The width h is a parameter that denotes the spread or smoothness of the density estimate. The discrete kernel function has abrupt changes.

Define a more smooth transition of influence via a Gaussian kernel:

$$K(z) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{z^2}{2}\right\}$$

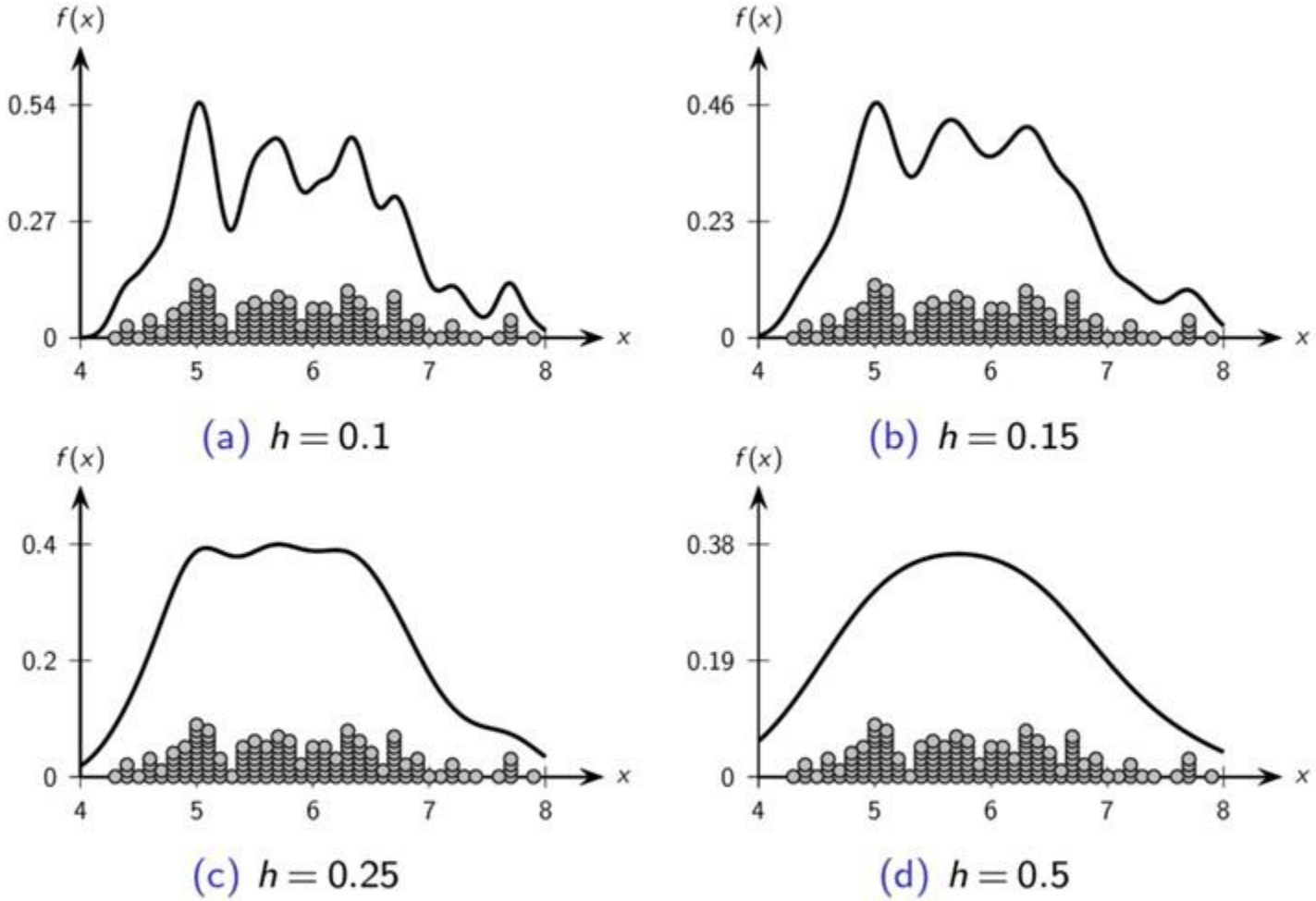
Thus, we have

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(x - x_i)^2}{2h^2}\right\}$$

Here x, which is at the center of the window, plays the role of the mean, and h acts as the standard deviation.

Kernel Density Estimation

Gaussian Kernel (Iris 1D)



When h is small the density function has many local maxima. A large h results in a unimodal distribution.

Multivariate Density Estimation

To estimate the probability density at a d-dimensional point $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$, we define the d-dimensional "window" as a hypercube in d dimensions, that is, a hypercube centered at $\boldsymbol{x}$ with edge length h. The volume of such a d-dimensional hypercube is given as

$$\text{vol}(H_d(h)) = h^d$$

The density is estimated as the fraction of the point weight lying within the d-dimensional window centered at $\boldsymbol{x}$, divided by the volume of the hypercube:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right)$$

where the multivariate kernel function K satisfies the condition $\int K(\boldsymbol{z}) d\boldsymbol{z} = 1$.

Discrete Kernel: For any d-dimensional vector $\boldsymbol{z} = (z_1, z_2, \ldots, z_d)^T$, the discrete kernel function in d-dimensions is given as

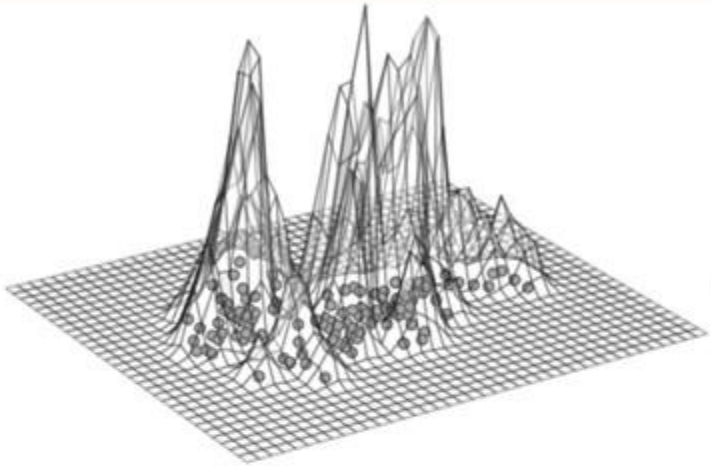
$$K(\boldsymbol{z}) = \begin{cases} 1 & \text{If } |z_j| \leq \frac{1}{2}, \text{ for all dimensions } j = 1, \ldots, d \\ 0 & \text{Otherwise} \end{cases}$$

Gaussian Kernel: The d-dimensional Gaussian kernel is given as

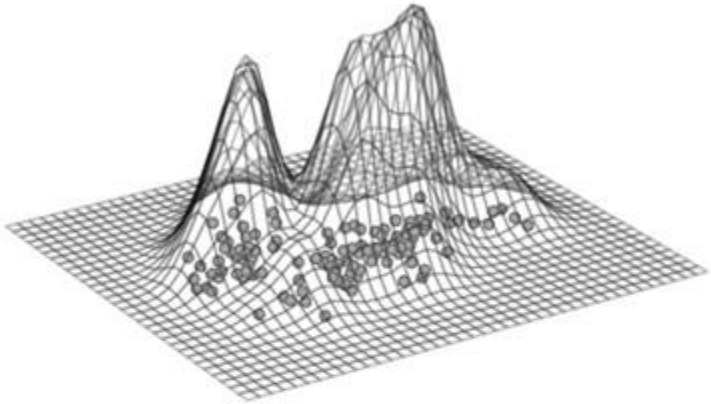
$$K(\boldsymbol{z}) = \frac{1}{(2\pi)^{d/2}} \exp\left\{-\frac{\boldsymbol{z}^T\boldsymbol{z}}{2}\right\}$$

Density Estimation

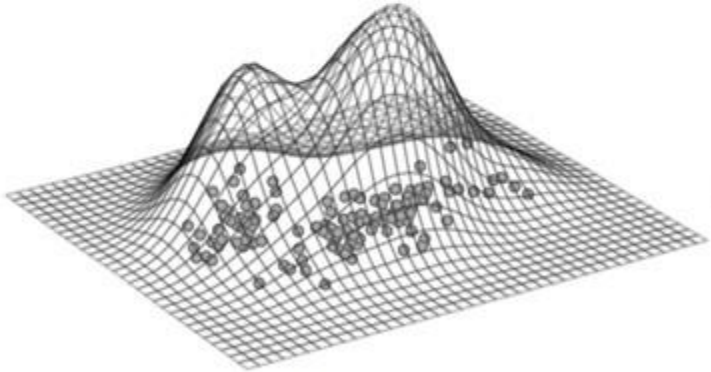
Iris 2D Data (Gaussian Kernel)



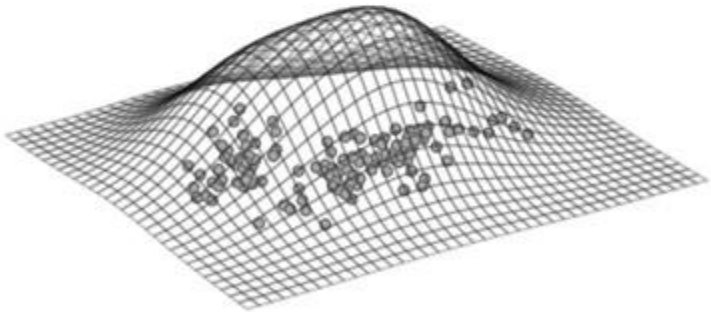
(a) $h = 0.1$



(b) $h = 0.2$



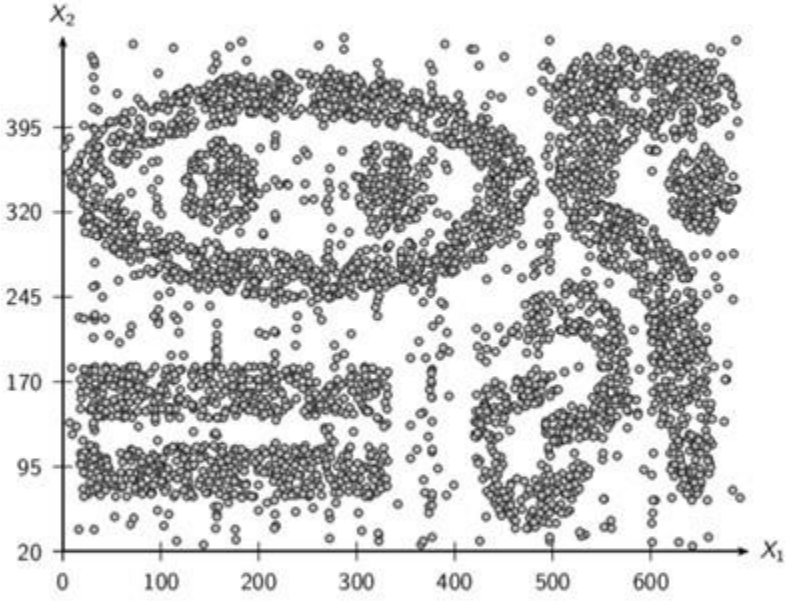
(c) $h = 0.35$



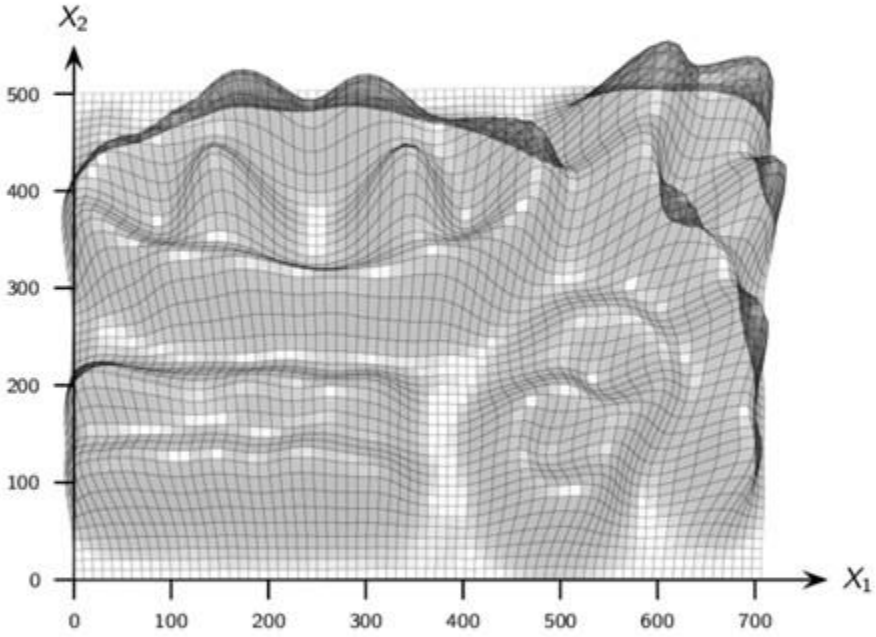
(d) $h = 0.6$

Density Estimation

Gaussian kernel, $h = 20$



(a) Original Points



(b) Gaussian Density Estimation

Nearest Neighbor Density Estimation

In kernel density estimation we implicitly fixed the volume by fixing the width h, and we used the kernel function to find out the number or weight of points that lie inside the fixed volume region.

An alternative approach to density estimation is to fix k, the number of points required to estimate the density, and allow the volume of the enclosing region to vary to accommodate those k points. This approach is called the k nearest neighbors (KNN) approach to density estimation.

Given k, the number of neighbors, we estimate the density at $\boldsymbol{x}$ as follows:

$$\hat{f}(\boldsymbol{x}) = \frac{k}{n \text{vol}(S_d(h_{\boldsymbol{x}}))}$$

where $h_{\boldsymbol{x}}$ is the distance from $\boldsymbol{x}$ to its kth nearest neighbor, and $\text{vol}(S_d(h_{\boldsymbol{x}}))$ is the volume of the d-dimensional hypersphere $S_d(h_{\boldsymbol{x}})$ centered at $\boldsymbol{x}$, with radius $h_{\boldsymbol{x}}$.

DENCLUE Density-based Clustering

Attractor and Gradient

A point $\boldsymbol{x}^*$ is called a *density attractor* if it is a local maxima of the probability density function f.

The density gradient at a point $\boldsymbol{x}$ is the multivariate derivative of the probability density estimate

$$\nabla \hat{f}(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}} \hat{f}(\boldsymbol{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} \frac{\partial}{\partial \boldsymbol{x}} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right)$$

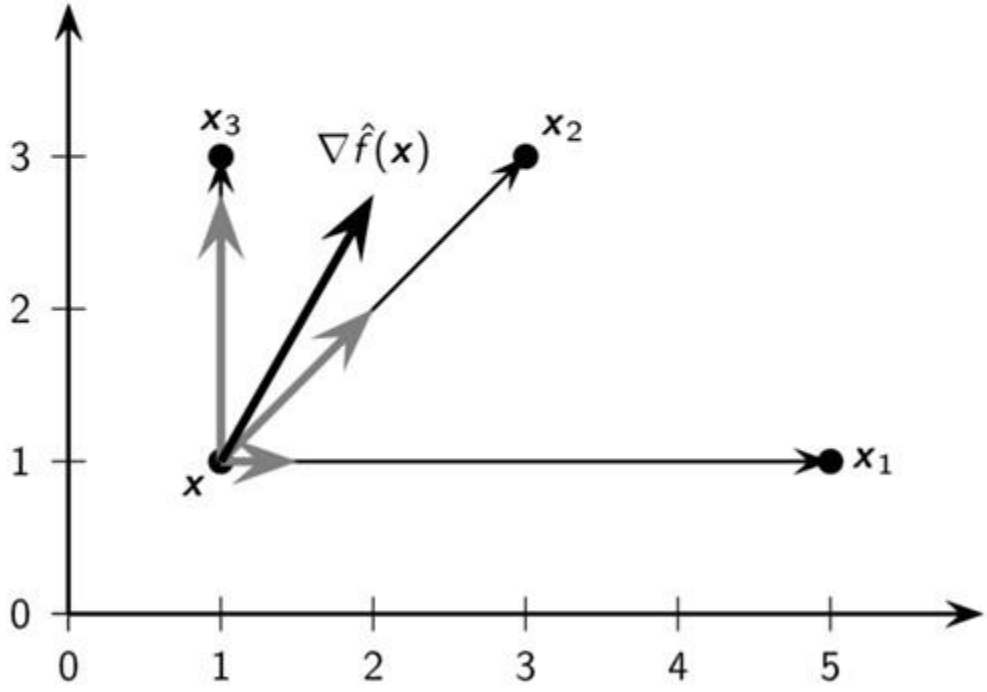
For the Gaussian kernel the gradient at a point $\boldsymbol{x}$ is given as

$$\nabla \hat{f}(\boldsymbol{x}) = \frac{1}{nh^{d+2}} \sum_{i=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right) \cdot (\boldsymbol{x}_i - \boldsymbol{x})$$

This equation can be thought of as having two parts for each point: a vector $(\boldsymbol{x}_i - \boldsymbol{x})$ and a scalar *influence* value $K(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h})$.

The Gradient Vector

We first compute the direction away from $\boldsymbol{x}$ to $\boldsymbol{x}_i$, i.e., the vector $(\boldsymbol{x}_i - \boldsymbol{x})$.
Next, we scale it using the Gaussian kernel value as the weight $K\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right)$.
$\nabla \hat{f}(\boldsymbol{x})$ is the net influence at $\boldsymbol{x}$, i.e., the weighted sum of the difference vectors.



DENCLUE: Density Attractor

We say that $\boldsymbol{x}^*$ is a *density attractor* for $\boldsymbol{x}$, or alternatively that $\boldsymbol{x}$ is *density attracted* to $\boldsymbol{x}^*$, if a hill climbing process started at $\boldsymbol{x}$ converges to $\boldsymbol{x}^*$.

That is, there exists a sequence of points $\boldsymbol{x} = \boldsymbol{x}_0 \rightarrow \boldsymbol{x}_1 \rightarrow \ldots \rightarrow \boldsymbol{x}_m$, starting from $\boldsymbol{x}$ and ending at $\boldsymbol{x}_m$, such that $\|\boldsymbol{x}_m - \boldsymbol{x}^*\| \leq \epsilon$, that is, $\boldsymbol{x}_m$ converges to the attractor $\boldsymbol{x}^*$.

Setting the gradient to the zero vector leads to the following *mean-shift* update rule:

$$\boldsymbol{x}_{t+1} = \frac{\sum_{i=1}^{n} K\left(\frac{\boldsymbol{x}_t - \boldsymbol{x}_i}{h}\right) \boldsymbol{x}_i}{\sum_{i=1}^{n} K\left(\frac{\boldsymbol{x}_t - \boldsymbol{x}_i}{h}\right)}$$

where t denotes the current iteration and $\boldsymbol{x}_{t+1}$ is the updated value for the current vector $\boldsymbol{x}_t$.

The DENCLUE Algorithm: Find Attractor

FindAttractor (x, D, h, ϵ)**:**

$t \leftarrow 0$

$x_t \leftarrow x$

repeat

  $x_{t+1} \leftarrow \frac{\sum_{i=1}^{n} K\left(\frac{x_t - x_i}{h}\right) \cdot x_t}{\sum_{i=1}^{n} K\left(\frac{x_t - x_i}{h}\right)}$

  $t \leftarrow t + 1$

until $\|x_t - x_{t-1}\| \leq \epsilon$

return x_t

DENCLUE: Density-based Cluster

A cluster $C \subseteq \boldsymbol{D}$, is called a *center-defined cluster* if all the points $\boldsymbol{x} \in C$ are density attracted to a unique density attractor $\boldsymbol{x}^*$, such that $\hat{f}(\boldsymbol{x}^*) \geq \xi$, where ξ is a user-defined minimum density threshold.

An arbitrary-shaped cluster $C \subseteq \boldsymbol{D}$ is called a *density-based cluster* if there exists a set of density attractors $\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \ldots, \boldsymbol{x}_m^*$, such that

1. Each point $\boldsymbol{x} \in C$ is attracted to some attractor $\boldsymbol{x}_i^*$.
2. Each density attractor has density above ξ.
3. Any two density attractors $\boldsymbol{x}_i^*$ and $\boldsymbol{x}_j^*$ are *density reachable*, that is, there exists a path from $\boldsymbol{x}_i^*$ to $\boldsymbol{x}_j^*$, such that for all points $\boldsymbol{y}$ on the path, $\hat{f}(\boldsymbol{y}) \geq \xi$.

The DENCLUE Algorithm

denclue $(\boldsymbol{D}, h, \xi, \epsilon)$**:**

1 $\mathcal{A} \leftarrow \emptyset$

2 **foreach** $\boldsymbol{x} \in \boldsymbol{D}$ **do** // `find density attractors`

4 $\quad \boldsymbol{x}^* \leftarrow \text{FindAttractor}(\boldsymbol{x}, \boldsymbol{D}, h, \epsilon)$

5 $\quad$ **if** $\hat{f}(\boldsymbol{x}^*) \geq \xi$ **then**

7 $\quad\quad \mathcal{A} \leftarrow \mathcal{A} \cup \{\boldsymbol{x}^*\}$

9 $\quad\quad R(\boldsymbol{x}^*) \leftarrow R(\boldsymbol{x}^*) \cup \{\boldsymbol{x}\}$

11 $\mathcal{C} \leftarrow \{\text{maximal } C \subseteq \mathcal{A} \mid \forall \boldsymbol{x}_i^*, \boldsymbol{x}_j^* \in C, \boldsymbol{x}_i^* \text{ and } \boldsymbol{x}_j^* \text{ are density reachable}\}$

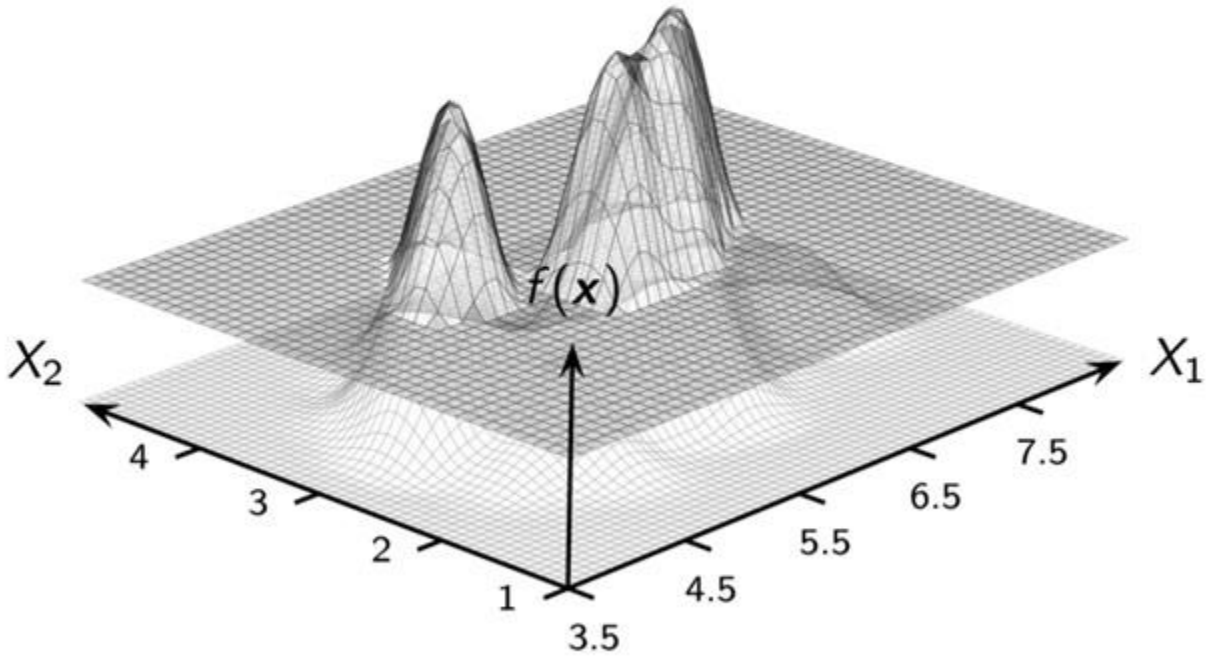
12 **foreach** $C \in \mathcal{C}$ **do** // `density-based clusters`

13 $\quad$ **foreach** $\boldsymbol{x}^* \in C$ **do** $C \leftarrow C \cup R(\boldsymbol{x}^*)$

14 **return** $\mathcal{C}$

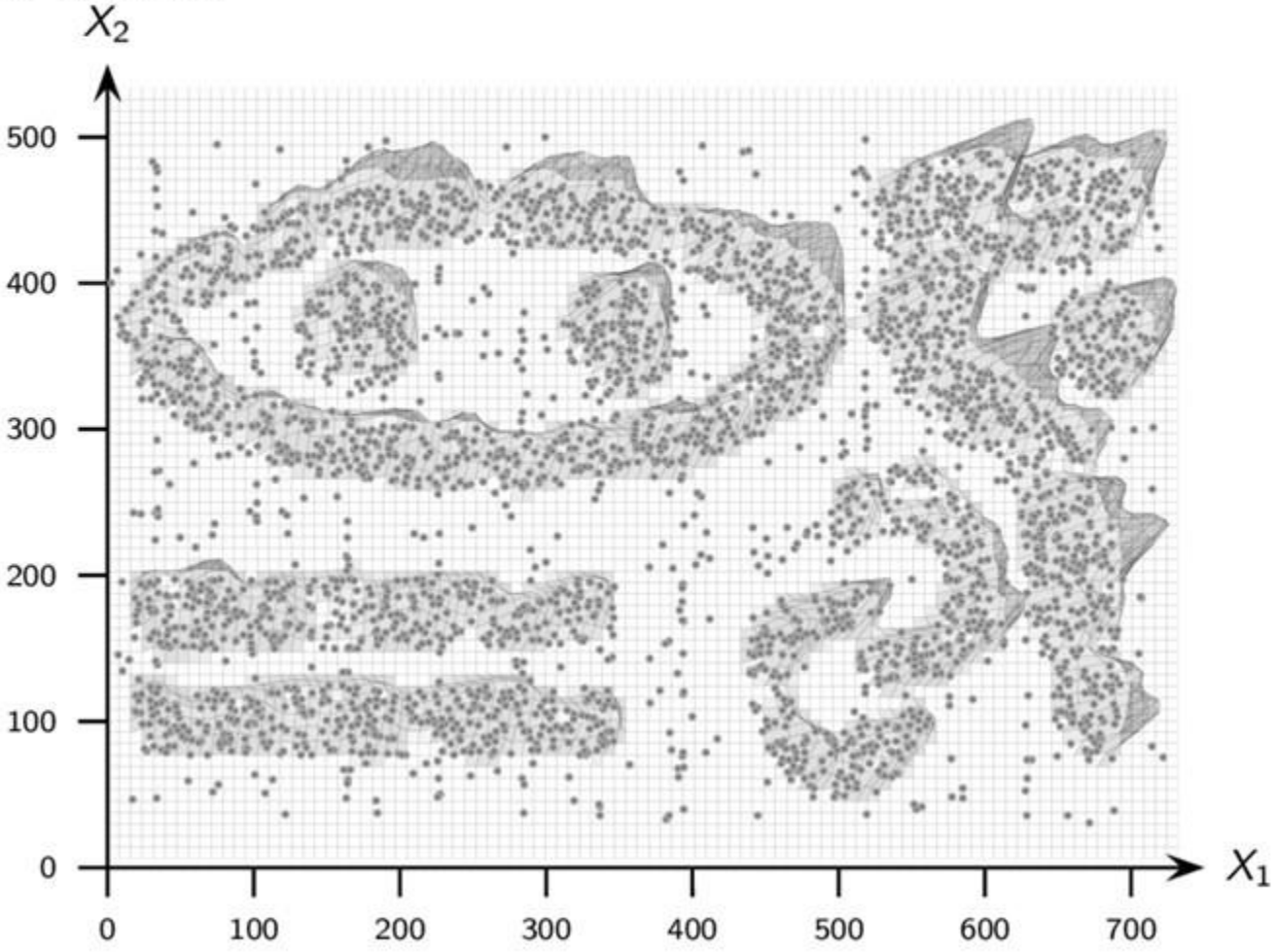
DENCLUE: Iris 2D Data

Iris 2D dataset comprising the `sepal length` and `sepal width` attributes. The results were obtained with $h = 0.2$ and $\xi = 0.08$, using a Gaussian kernel.



DENCLUE: Density-based Dataset

Using the parameters $h = 10$ and $\xi = 9.5 \times 10^{-5}$, with a Gaussian kernel, we obtain eight clusters.



Data mining and Machine learning

Part 16. Spectral & Graph Clustering

Graphs and Matrices: Adjacency Matrix

Given a dataset $\boldsymbol{D} = \{\boldsymbol{x}_i\}_{i=1}^n$ consisting of n points in $\mathbb{R}^d$, let $\boldsymbol{A}$ denote the $n \times n$ symmetric *similarity matrix* between the points, given as

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

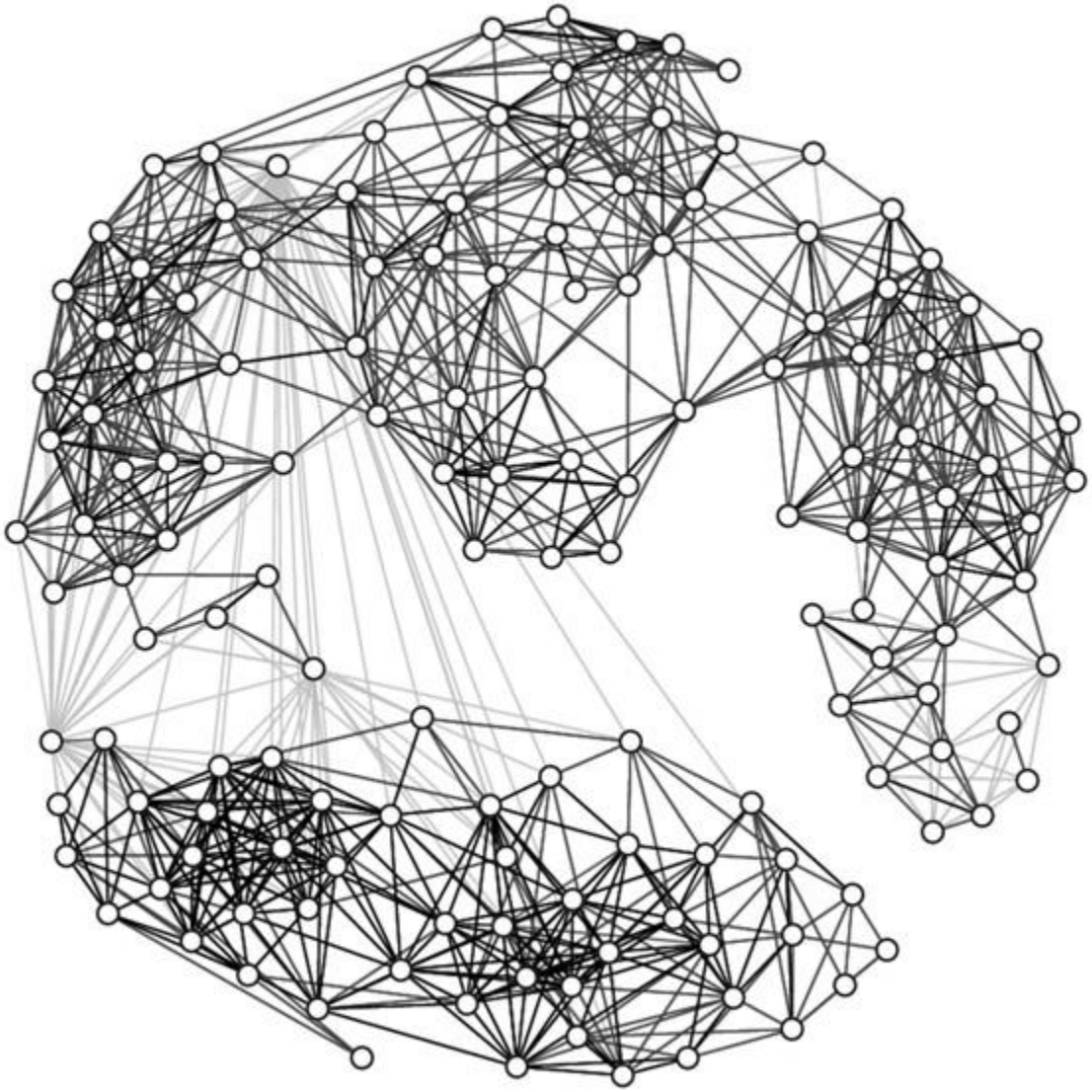
where $\boldsymbol{A}(i,j) = a_{ij}$ denotes the similarity or affinity between points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

We require the similarity to be symmetric and non-negative, that is, $a_{ij} = a_{ji}$ and $a_{ij} \geq 0$, respectively.

The matrix $\boldsymbol{A}$ is the *weighted adjacency matrix* for the data graph. If all affinities are 0 or 1, then $\boldsymbol{A}$ represents the regular adjacency relationship between the vertices.

Iris Similarity Graph: Mutual Nearest Neighbors

$|V| = n = 150$, $|E| = m = 1730$



Graphs and Matrices: Degree Matrix

For a vertex $\boldsymbol{x}_i$, let d_i denote the *degree* of the vertex, defined as

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

We define the *degree matrix* Δ of graph G as the $n \times n$ diagonal matrix:

$$\Delta = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n} a_{1j} & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{n} a_{2j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{j=1}^{n} a_{nj} \end{pmatrix}$$

Δ can be compactly written as $\Delta(i,i) = d_i$ for all $1 \leq i \leq n$.

Graphs and Matrices: Normalized Adjacency Matrix

The normalized adjacency matrix is obtained by dividing each row of the adjacency matrix by the degree of the corresponding node. Given the weighted adjacency matrix $\boldsymbol{A}$ for a graph G, its normalized adjacency matrix is defined as

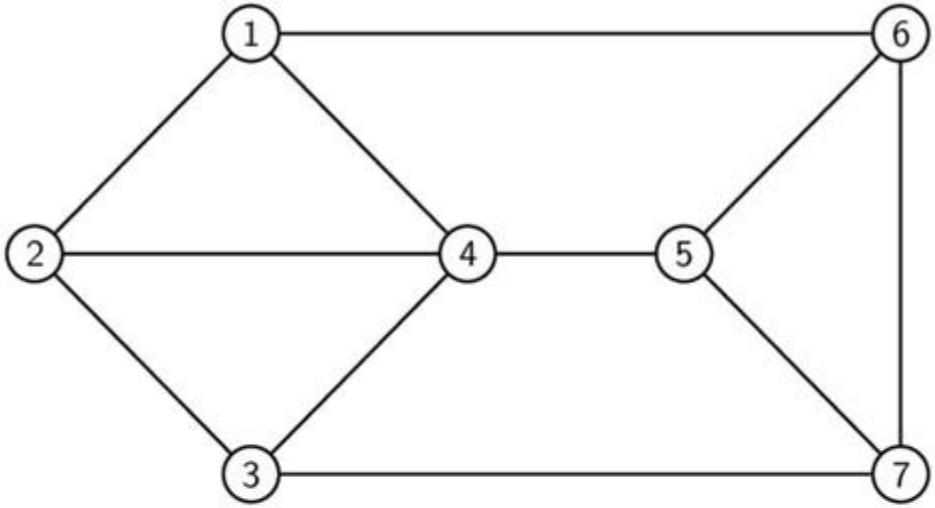
$$\boldsymbol{M} = \Delta^{-1}\boldsymbol{A} = \begin{pmatrix} \frac{a_{11}}{d_1} & \frac{a_{12}}{d_1} & \cdots & \frac{a_{1n}}{d_1} \\ \frac{a_{21}}{d_2} & \frac{a_{22}}{d_2} & \cdots & \frac{a_{2n}}{d_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{d_n} & \frac{a_{n2}}{d_n} & \cdots & \frac{a_{nn}}{d_n} \end{pmatrix}$$

Because $\boldsymbol{A}$ is assumed to have non-negative elements, this implies that each element of $\boldsymbol{M}$, namely m_{ij} is also non-negative, as $m_{ij} = \frac{a_{ij}}{d_i} \geq 0$.

Each row in $\boldsymbol{M}$ sums to 1, which implies that 1 is an eigenvalue of $\boldsymbol{M}$. In fact, $\lambda_1 = 1$ is the largest eigenvalue of $\boldsymbol{M}$, and the other eigenvalues satisfy the property that $|\lambda_i| \leq 1$. Because $\boldsymbol{M}$ is not symmetric, its eigenvectors are not necessarily orthogonal.

Example Graph

Adjacency and Degree Matrices

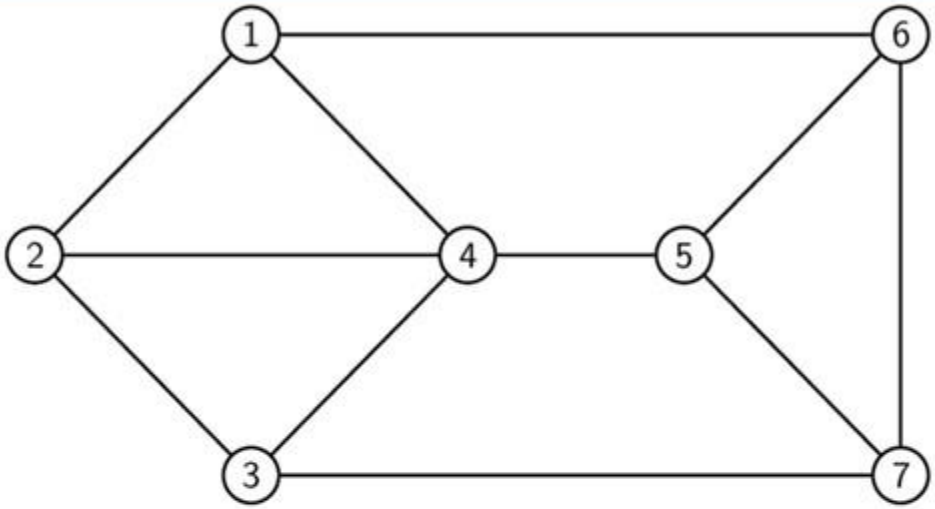


Its adjacency and degree matrices are given as

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \qquad \Delta = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

Example Graph

Normalized Adjacency Matrix



The normalized adjacency matrix is as follows:

$$\boldsymbol{M} = \Delta^{-1}\boldsymbol{A} = \begin{pmatrix} 0 & 0.33 & 0 & 0.33 & 0 & 0.33 & 0 \\ 0.33 & 0 & 0.33 & 0.33 & 0 & 0 & 0 \\ 0 & 0.33 & 0 & 0.33 & 0 & 0 & 0.33 \\ 0.25 & 0.25 & 0.25 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0.33 & 0 & 0.33 & 0.33 \\ 0.33 & 0 & 0 & 0 & 0.33 & 0 & 0.33 \\ 0 & 0 & 0.33 & 0 & 0.33 & 0.33 & 0 \end{pmatrix}$$

The eigenvalues of $\boldsymbol{M}$ are: $\lambda_1 = 1$, $\lambda_2 = 0.483$, $\lambda_3 = 0.206$, $\lambda_4 = -0.045$, $\lambda_5 = -0.405$, $\lambda_6 = -0.539$, $\lambda_7 = -0.7$

Graph Laplacian Matrix

The *Laplacian matrix* of a graph is defined as

$$\boldsymbol{L} = \Delta - \boldsymbol{A}$$

$$= \begin{pmatrix} \sum_{j=1}^{n} a_{1j} & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{n} a_{2j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{j=1}^{n} a_{nj} \end{pmatrix} - \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{j \neq 1} a_{1j} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \sum_{j \neq 2} a_{2j} & \cdots & -a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \sum_{j \neq n} a_{nj} \end{pmatrix}$$

$\boldsymbol{L}$ is a symmetric, positive semidefinite matrix. This means that $\boldsymbol{L}$ has n real, non-negative eigenvalues, which can be arranged in decreasing order as follows: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Because $\boldsymbol{L}$ is symmetric, its eigenvectors are orthonormal. The rank of $\boldsymbol{L}$ is at most $n-1$, and the smallest eigenvalue is $\lambda_n = 0$.

Graph Laplacian Matrix

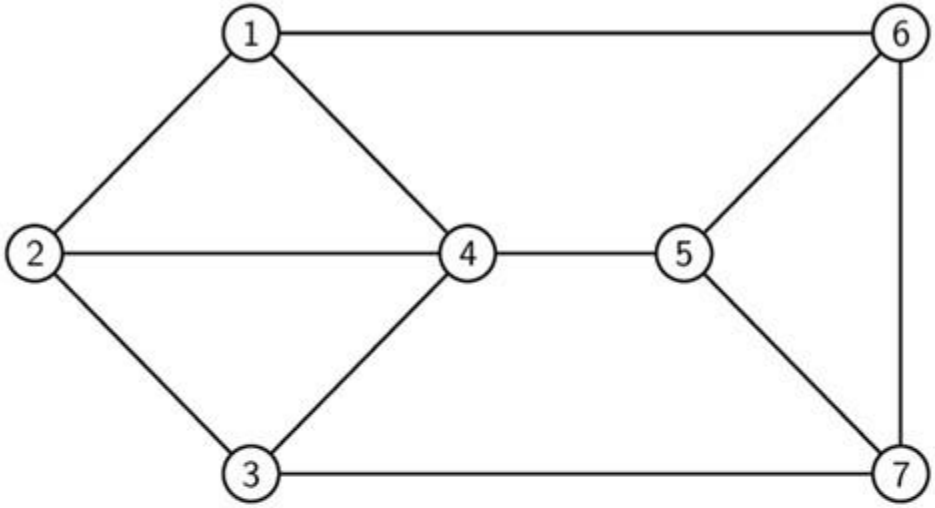
The Laplacian matrix is used to compute some properties of its corresponding graph. For example:

1. The number of connected components of graph G is equal to the multiplicity of the Laplacian eigenvalue 0.
2. The cofactor of any element of L(G) is equal to the number of spanning trees of G.

Laplacian matrix is usually used when the graph is irregular. Its main advantage is that it enables us to study certain properties of irregular graphs (unlike adjacent matrix).

Given a graph function, the Laplacian measures how much the function on a graph differs at a vertex from the average of the values of the same function over the neighbors of the vertex.

Example Graph: Laplacian Matrix



The graph Laplacian is given as

$$\boldsymbol{L} = \Delta - \boldsymbol{A} = \begin{pmatrix} 3 & -1 & 0 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & -1 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 & 3 \end{pmatrix}$$

The eigenvalues of $\boldsymbol{L}$ are as follows: $\lambda_1 = 5.618$, $\lambda_2 = 4.618$, $\lambda_3 = 4.414$, $\lambda_4 = 3.382$, $\lambda_5 = 2.382$, $\lambda_6 = 1.586$, $\lambda_7 = 0$

Normalized Laplacian Matrices

The *normalized symmetric Laplacian matrix* of a graph is defined as

$$\boldsymbol{L}^s = \Delta^{-1/2} \boldsymbol{L} \Delta^{-1/2} = \begin{pmatrix} \frac{\sum_{j \neq 1} a_{1j}}{\sqrt{d_1 d_1}} & -\frac{a_{12}}{\sqrt{d_1 d_2}} & \cdots & -\frac{a_{1n}}{\sqrt{d_1 d_n}} \\ -\frac{a_{21}}{\sqrt{d_2 d_1}} & \frac{\sum_{j \neq 2} a_{2j}}{\sqrt{d_2 d_2}} & \cdots & -\frac{a_{2n}}{\sqrt{d_2 d_n}} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{\sqrt{d_n d_1}} & -\frac{a_{n2}}{\sqrt{d_n d_2}} & \cdots & \frac{\sum_{j \neq n} a_{nj}}{\sqrt{d_n d_n}} \end{pmatrix}$$

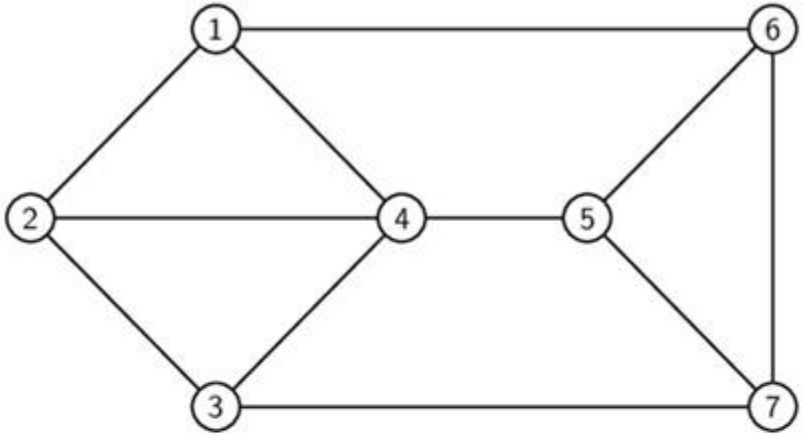
$\boldsymbol{L}^s$ is a symmetric, positive semidefinite matrix, with rank at most $n-1$. The smallest eigenvalue $\lambda_n = 0$. The *normalized asymmetric Laplacian* matrix is defined as

$$\boldsymbol{L}^a = \Delta^{-1} \boldsymbol{L} = \begin{pmatrix} \frac{\sum_{j \neq 1} a_{1j}}{d_1} & -\frac{a_{12}}{d_1} & \cdots & -\frac{a_{1n}}{d_1} \\ -\frac{a_{21}}{d_2} & \frac{\sum_{j \neq 2} a_{2j}}{d_2} & \cdots & -\frac{a_{2n}}{d_2} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{d_n} & -\frac{a_{n2}}{d_n} & \cdots & \frac{\sum_{j \neq n} a_{nj}}{d_n} \end{pmatrix}$$

$\boldsymbol{L}^a$ is also a positive semi-definite matrix with n real eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n = 0$.

Example Graph

Normalized Symmetric Laplacian Matrix



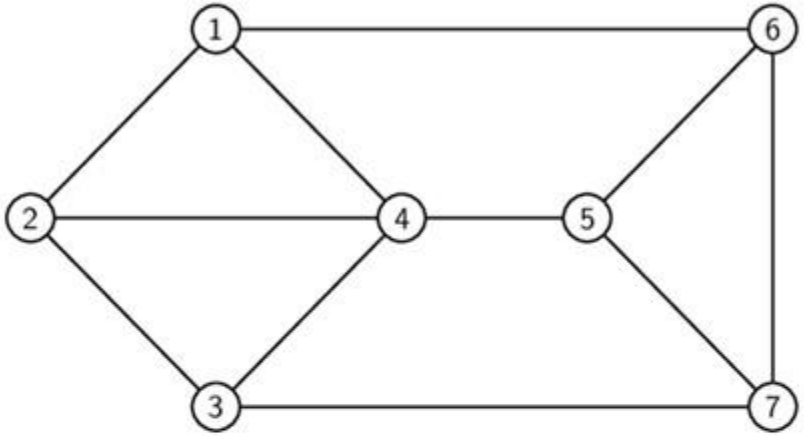
The normalized symmetric Laplacian is given as

$$\boldsymbol{L}^s = \begin{pmatrix} 1 & -0.33 & 0 & -0.29 & 0 & -0.33 & 0 \\ -0.33 & 1 & -0.33 & -0.29 & 0 & 0 & 0 \\ 0 & -0.33 & 1 & -0.29 & 0 & 0 & -0.33 \\ -0.29 & -0.29 & -0.29 & 1 & -0.29 & 0 & 0 \\ 0 & 0 & 0 & -0.29 & 1 & -0.33 & -0.33 \\ -0.33 & 0 & 0 & 0 & -0.33 & 1 & -0.33 \\ 0 & 0 & -0.33 & 0 & -0.33 & -0.33 & 1 \end{pmatrix}$$

The eigenvalues of $\boldsymbol{L}^s$ are as follows: $\lambda_1 = 1.7$, $\lambda_2 = 1.539$, $\lambda_3 = 1.405$, $\lambda_4 = 1.045$, $\lambda_5 = 0.794$, $\lambda_6 = 0.517$, $\lambda_7 = 0$

Example Graph

Normalized Asymmetric Laplacian Matrix



The normalized asymmetric Laplacian matrix is given as

$$\boldsymbol{L}^a = \Delta^{-1}\boldsymbol{L} = \begin{pmatrix} 1 & -0.33 & 0 & -0.33 & 0 & -0.33 & 0 \\ -0.33 & 1 & -0.33 & -0.33 & 0 & 0 & 0 \\ 0 & -0.33 & 1 & -0.33 & 0 & 0 & -0.33 \\ -0.25 & -0.25 & -0.25 & 1 & -0.25 & 0 & 0 \\ 0 & 0 & 0 & -0.33 & 1 & -0.33 & -0.33 \\ -0.33 & 0 & 0 & 0 & -0.33 & 1 & -0.33 \\ 0 & 0 & -0.33 & 0 & -0.33 & -0.33 & 1 \end{pmatrix}$$

The eigenvalues of $\boldsymbol{L}^a$ are identical to those for $\boldsymbol{L}^s$, namely $\lambda_1 = 1.7$, $\lambda_2 = 1.539$, $\lambda_3 = 1.405$, $\lambda_4 = 1.045$, $\lambda_5 = 0.794$, $\lambda_6 = 0.517$, $\lambda_7 = 0$

Clustering as Graph Cuts

A *k-way cut* in a graph is a partitioning or clustering of the vertex set, given as $\mathcal{C} = \{C_1, \ldots, C_k\}$. We require $\mathcal{C}$ to optimize some objective function that captures the intuition that nodes within a cluster should have high similarity, and nodes from different clusters should have low similarity.

Given a weighted graph G defined by its similarity matrix $\boldsymbol{A}$, let $S, T \subseteq V$ be any two subsets of the vertices. We denote by $W(S, T)$ the sum of the weights on all edges with one vertex in S and the other in T, given as

$$W(S, T) = \sum_{v_i \in S} \sum_{v_j \in T} a_{ij}$$

Given $S \subseteq V$, we denote by $\overline{S}$ the complementary set of vertices, that is, $\overline{S} = V - S$. A *(vertex) cut* in a graph is defined as a partitioning of V into $S \subset V$ and $\overline{S}$. The *weight of the cut* or *cut weight* is defined as the sum of all the weights on edges between vertices in S and $\overline{S}$, given as $W(S, \overline{S})$.

Cuts and Matrix Operations

Given a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ comprising k clusters. Let $\boldsymbol{c}_i \in \{0,1\}^n$ be the *cluster indicator vector* that records the cluster membership for cluster C_i, defined as

$$c_{ij} = \begin{cases} 1 & \text{if } v_j \in C_i \\ 0 & \text{if } v_j \notin C_i \end{cases}$$

The cluster size can be written as

$$|C_i| = \boldsymbol{c}_i^T \boldsymbol{c}_i = \|\boldsymbol{c}_i\|^2$$

The *volume* of a cluster C_i is defined as the sum of all the weights on edges with one end in cluster C_i:

$$vol(C_i) = W(C_i, V) = \sum_{v_r \in C_i} d_r = \sum_{v_r \in C_i} c_{ir} d_r c_{ir} = \sum_{r=1}^{n} \sum_{s=1}^{n} c_{ir} \Delta_{rs} c_{is} = \boldsymbol{c}_i^T \Delta \boldsymbol{c}_i$$

Cuts and Matrix Operations

The sum of weights of all internal edges is:

$$W(C_i, C_i) = \sum_{v_r \in C_i} \sum_{v_s \in C_i} a_{rs} = \sum_{r=1}^{n} \sum_{s=1}^{n} c_{ir} a_{rs} c_{is} = \boldsymbol{c}_i^T \boldsymbol{A} \boldsymbol{c}_i$$

We can get the sum of weights for all the external edges as follows:

$$W(C_i, \overline{C_i}) = \sum_{v_r \in C_i} \sum_{v_s \in V - C_i} a_{rs} = W(C_i, V) - W(C_i, C_i) = \boldsymbol{c}_i(\Delta - \boldsymbol{A})\boldsymbol{c}_i = \boldsymbol{c}_i^T \boldsymbol{L} \boldsymbol{c}_i$$

Clustering Objective Functions: Ratio Cut

The clustering objective function can be formulated as an optimization problem over the k-way cut $\mathcal{C} = \{C_1, \ldots, C_k\}$.
The *ratio cut* objective is defined over a k-way cut as follows:

$$\min_{\mathcal{C}} J_{rc}(\mathcal{C}) = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{|C_i|} = \sum_{i=1}^{k} \frac{\boldsymbol{c}_i^T \boldsymbol{L} \boldsymbol{c}_i}{\boldsymbol{c}_i^T \boldsymbol{c}_i} = \sum_{i=1}^{k} \frac{\boldsymbol{c}_i^T \boldsymbol{L} \boldsymbol{c}_i}{\|\boldsymbol{c}_i\|^2}$$

Ratio cut tries to minimize the sum of the similarities from a cluster C_i to other points not in the cluster $\overline{C_i}$, taking into account the size of each cluster.
Unfortunately, for binary cluster indicator vectors $\boldsymbol{c}_i$, the ratio cut objective is NP-hard. An obvious relaxation is to allow $\boldsymbol{c}_i$ to take on any real value. In this case, we can rewrite the objective as

$$\min_{\mathcal{C}} J_{rc}(\mathcal{C}) = \sum_{i=1}^{k} \frac{\boldsymbol{c}_i^T \boldsymbol{L} \boldsymbol{c}_i}{\|\boldsymbol{c}_i\|^2} = \sum_{i=1}^{k} \left(\frac{\boldsymbol{c}_i}{\|\boldsymbol{c}_i\|}\right)^T \boldsymbol{L} \left(\frac{\boldsymbol{c}_i}{\|\boldsymbol{c}_i\|}\right) = \sum_{i=1}^{k} \boldsymbol{u}_i^T \boldsymbol{L} \boldsymbol{u}_i$$

The optimal solution comprises the eigenvectors corresponding to the k smallest eigenvalues of $\boldsymbol{L}$, i.e., the eigenvectors $\boldsymbol{u}_n, \boldsymbol{u}_{n-1}, \ldots, \boldsymbol{u}_{n-k+1}$ represent the relaxed cluster indicator vectors.

Clustering Objective Functions: Normalized Cut

Normalized cut is similar to ratio cut, except that it divides the cut weight of each cluster by the volume of a cluster instead of its size. The objective function is given as

$$\min_{\mathcal{C}} J_{nc}(\mathcal{C}) = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{vol(C_i)} = \sum_{i=1}^{k} \frac{\boldsymbol{c}_i^T \boldsymbol{L} \boldsymbol{c}_i}{\boldsymbol{c}_i^T \Delta \boldsymbol{c}_i}$$

We can obtain an optimal solution by allowing $\boldsymbol{c}_i$ to be an arbitrary real vector.

The optimal solution comprise the eigenvectors corresponding to the k smallest eigenvalues of either the normalized symmetric or asymmetric Laplacian matrices, $\boldsymbol{L}^s$ and $\boldsymbol{L}^a$.

Spectral Clustering Algorithm

The spectral clustering algorithm takes a dataset $\boldsymbol{D}$ as input and computes the similarity matrix $\boldsymbol{A}$. For normalized cut we chose either $\boldsymbol{L}^s$ or $\boldsymbol{L}^a$, whereas for ratio cut we choose $\boldsymbol{L}$. Next, we compute the k smallest eigenvalues and corresponding eigenvectors of the chosen matrix.

The main problem is that the eigenvectors $\boldsymbol{u}_i$ are not binary, and thus it is not immediately clear how we can assign points to clusters.

One solution to this problem is to treat the $n \times k$ matrix of eigenvectors as a new data matrix:

$$\boldsymbol{U} = \begin{pmatrix} | & | & & | \\ \boldsymbol{u}_n & \boldsymbol{u}_{n-1} & \cdots & \boldsymbol{u}_{n-k+1} \\ | & | & & | \end{pmatrix} \rightarrow \text{ normalize rows } \rightarrow \begin{pmatrix} — & \boldsymbol{y}_1^T & — \\ — & \boldsymbol{y}_2^T & — \\ & \vdots & \\ — & \boldsymbol{y}_n^T & — \end{pmatrix} = \boldsymbol{Y}$$

We then cluster the new points in $\boldsymbol{Y}$ into k clusters via the K-means algorithm or any other fast clustering method to obtain binary cluster indicator vectors $\boldsymbol{c}_i$.

Spectral Clustering Algorithm

Spectral Clustering $(\boldsymbol{D}, k)$**:**

1 Compute the similarity matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$

2 **if** *ratio cut* **then** $\boldsymbol{B} \leftarrow \boldsymbol{L}$

3 **else if** *normalized cut* **then** $\boldsymbol{B} \leftarrow \boldsymbol{L}^s$ or $\boldsymbol{L}^a$

4 Solve $\boldsymbol{B}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$ for $i = n, \ldots, n-k+1$, where $\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_{n-k+1}$

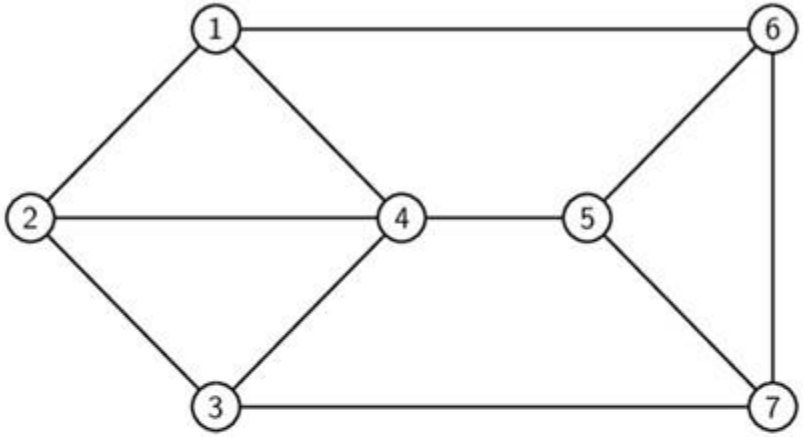
5 $\boldsymbol{U} \leftarrow \begin{pmatrix} \boldsymbol{u}_n & \boldsymbol{u}_{n-1} & \cdots & \boldsymbol{u}_{n-k+1} \end{pmatrix}$

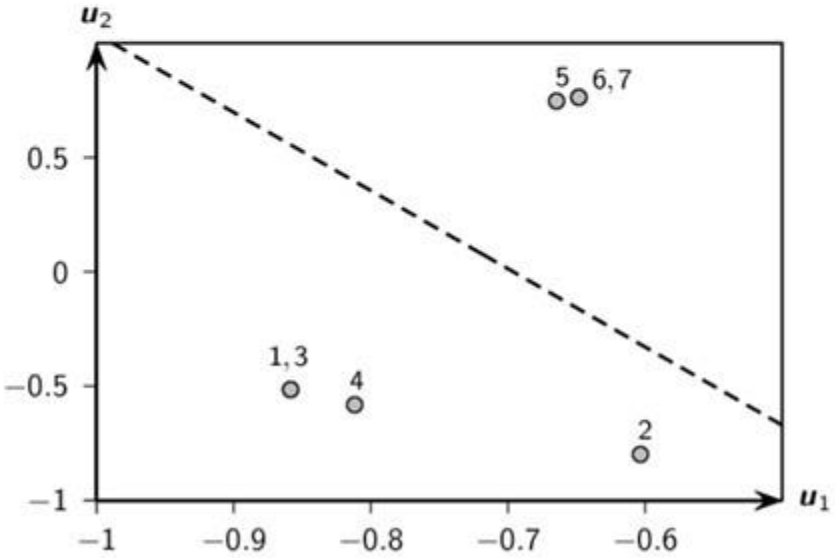
6 $\boldsymbol{Y} \leftarrow$ normalize rows of $\boldsymbol{U}$

7 $\mathcal{C} \leftarrow \{C_1, \ldots, C_k\}$ via K-means on $\boldsymbol{Y}$

Spectral Clustering on Example Graph

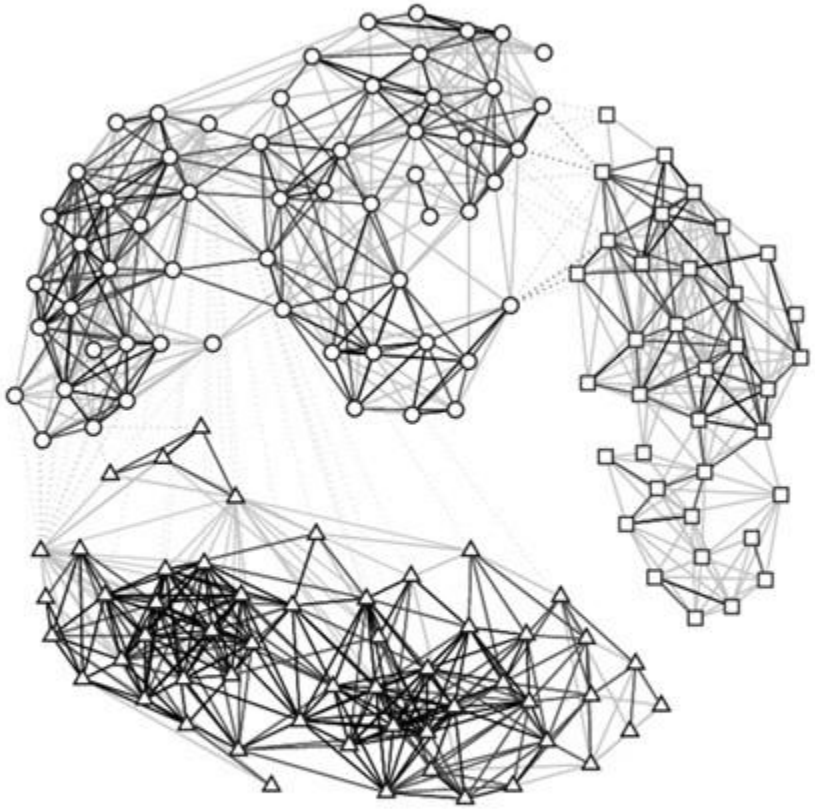
$k = 2$, normalized cut (normalized asymmetric Laplacian)





Normalized Cut on Iris Graph

$k = 3$, normalized asymmetric Laplacian



	setosa	virginica	versicolor
C_1 (triangle)	50	0	4
C_2 (square)	0	36	0
C_3 (circle)	0	14	46

Maximization Objectives: Average Cut

The *average weight* objective is defined as

$$\max_{\mathcal{C}} J_{aw}(\mathcal{C}) = \sum_{i=1}^{k} \frac{W(C_i, C_i)}{|C_i|} = \sum_{i=1}^{k} \frac{\boldsymbol{c}_i^T \boldsymbol{A} \boldsymbol{c}_i}{\boldsymbol{c}_i^T \boldsymbol{c}_i} = \sum_{i=1}^{k} \boldsymbol{u}_i^T \boldsymbol{A} \boldsymbol{u}_i$$

where $\boldsymbol{u}_i$ is an arbitrary real vector, which is a relaxation of the binary cluster indicator vectors $\boldsymbol{c}_i$.

We can maximize the objective by selecting the k largest eigenvalues of $\boldsymbol{A}$, and the corresponding eigenvectors.

$$\begin{aligned} \max_{\mathcal{C}} J_{aw}(\mathcal{C}) &= \boldsymbol{u}_1^T \boldsymbol{A} \boldsymbol{u}_1 + \cdots + \boldsymbol{u}_k^T \boldsymbol{A} \boldsymbol{u}_k \\ &= \lambda_1 + \cdots + \lambda_k \end{aligned}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. In general, while $\boldsymbol{A}$ is symmetric, it may not be positive semidefinite. This means that $\boldsymbol{A}$ can have negative eigenvalues, and to maximize the objective we must consider only the positive eigenvalues and the corresponding eigenvectors.

Maximization Objectives: Modularity

Given $\boldsymbol{A}$, the weighted adjacency matrix, the modularity of a clustering is the difference between the observed and expected fraction of weights on edges within the clusters. The clustering objective is given as

$$\max_{\mathcal{C}} J_Q(\mathcal{C}) = \sum_{i=1}^{k} \left(\frac{\boldsymbol{c}_i^T \boldsymbol{A} \boldsymbol{c}_i}{tr(\Delta)} - \frac{(\boldsymbol{d}_i^T \boldsymbol{c}_i)^2}{tr(\Delta)^2} \right) = \sum_{i=1}^{k} \boldsymbol{c}_i^T \boldsymbol{Q} \boldsymbol{c}_i$$

where $\boldsymbol{Q}$ is the *modularity matrix*:

$$\boldsymbol{Q} = \frac{1}{tr(\Delta)} \left(\boldsymbol{A} - \frac{\boldsymbol{d} \cdot \boldsymbol{d}^T}{tr(\Delta)} \right)$$

The optimal solution comprises the eigenvectors corresponding to the k largest eigenvalues of $\boldsymbol{Q}$. Since $\boldsymbol{Q}$ is symmetric, but not positive semidefinite, we use only the positive eigenvalues.

Markov Chain Clustering

A Markov chain is a discrete-time stochastic process over a set of states, in our case the set of vertices V.

The Markov chain makes a transition from one node to another at discrete timesteps $t = 1, 2, \ldots$, with the probability of making a transition from node i to node j given as m_{ij}.

Let the random variable X_t denote the state at time t. The Markov property means that the probability distribution of X_t over the states at time t depends only on the probability distribution of X_{t-1}, that is,

$$P(X_t = i | X_0, X_1, \ldots, X_{t-1}) = P(X_t = i | X_{t-1})$$

Further, we assume that the Markov chain is *homogeneous*, that is, the transition probability

$$P(X_t = j | X_{t-1} = i) = m_{ij}$$

is independent of the time step t.

Markov Chain Clustering: Markov Matrix

The normalized adjacency matrix $\boldsymbol{M} = \Delta^{-1}\boldsymbol{A}$ can be interpreted as the $n \times n$ *transition matrix* where the entry $m_{ij} = \frac{a_{ij}}{d_i}$ is the probability of transitioning or jumping from node i to node j in the graph G.

The matrix $\boldsymbol{M}$ is thus the transition matrix for a *Markov chain* or a Markov random walk on graph G. That is, given node i the transition matrix $\boldsymbol{M}$ specifies the probabilities of reaching any other node j in one time step.

In general, the transition probability matrix for t time steps is given as

$$\boldsymbol{M}^{t-1} \cdot \boldsymbol{M} = \boldsymbol{M}^t$$

Markov Chain Clustering: Random Walk

A random walk on G thus corresponds to taking successive powers of the transition matrix $\boldsymbol{M}$.

Let $\boldsymbol{\pi}_0$ specify the initial state probability vector at time $t = 0$. The state probability vector after t steps is

$$\boldsymbol{\pi}_t^T = \boldsymbol{\pi}_{t-1}^T \boldsymbol{M} = \boldsymbol{\pi}_{t-2}^T \boldsymbol{M}^2 = \cdots = \boldsymbol{\pi}_0^T \boldsymbol{M}^t$$

Equivalently, taking transpose on both sides, we get

$$\boldsymbol{\pi}_t = (M^t)^T \boldsymbol{\pi}_0 = (\boldsymbol{M}^T)^t \boldsymbol{\pi}_0$$

The state probability vector thus converges to the dominant eigenvector of $\boldsymbol{M}^T$.

Markov Clustering Algorithm

Consider a variation of the random walk, where the probability of transitioning from node i to j is inflated by taking each element m_{ij} to the power $r \geq 1$. Given a transition matrix $\boldsymbol{M}$, define the inflation operator Υ as follows:

$$\Upsilon(\boldsymbol{M}, r) = \left\{ \frac{(m_{ij})^r}{\sum_{a=1}^{n} (m_{ia})^r} \right\}_{i,j=1}^{n}$$

The net effect of the inflation operator is to increase the higher probability transitions and decrease the lower probability transitions.

The Markov clustering algorithm (MCL) is an iterative method that interleaves matrix expansion and inflation steps. Matrix expansion corresponds to taking successive powers of the transition matrix, leading to random walks of longer lengths. On the other hand, matrix inflation makes the higher probability transitions even more likely and reduces the lower probability transitions.

MCL takes as input the inflation parameter $r \geq 1$. Higher values lead to more, smaller clusters, whereas smaller values lead to fewer, but larger clusters.

Markov Clustering Algorithm: MCL

The final clusters are found by enumerating the weakly connected components in the directed graph induced by the converged transition matrix $\boldsymbol{M}_t$, where the edges are defined as:

$$E = \{(i,j) \mid \boldsymbol{M}_t(i,j) > 0\}$$

A directed edge (i,j) exists only if node i can transition to node j within t steps of the expansion and inflation process.

A node j is called an *attractor* if $\boldsymbol{M}_t(j,j) > 0$, and we say that node i is attracted to attractor j if $\boldsymbol{M}_t(i,j) > 0$. The MCL process yields a set of attractor nodes, $V_a \subseteq V$, such that other nodes are attracted to at least one attractor in V_a.

To extract the clusters from G_t, MCL first finds the strongly connected components $S_1, S_2, \ldots, S_q$ over the set of attractors V_a. Next, for each strongly connected set of attractors S_j, MCL finds the weakly connected components consisting of all nodes $i \in V_t - V_a$ attracted to an attractor in S_j. If a node i is attracted to multiple strongly connected components, it is added to each such cluster, resulting in possibly overlapping clusters.

Algorithm Markov Clustering

Markov Clustering $(\boldsymbol{A}, r, \epsilon)$**:**

$t \leftarrow 0$

Add self-edges to $\boldsymbol{A}$ if they do not exist

$\boldsymbol{M}_t \leftarrow \Delta^{-1}\boldsymbol{A}$

repeat

  $t \leftarrow t+1$

  $\boldsymbol{M}_t \leftarrow \boldsymbol{M}_{t-1} \cdot \boldsymbol{M}_{t-1}$

  $\boldsymbol{M}_t \leftarrow \Upsilon(\boldsymbol{M}_t, r)$

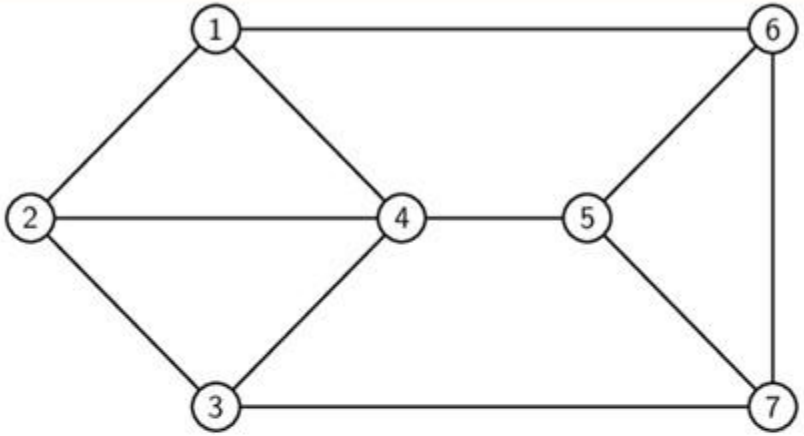
until $\|\boldsymbol{M}_t - \boldsymbol{M}_{t-1}\|_F \leq \epsilon$

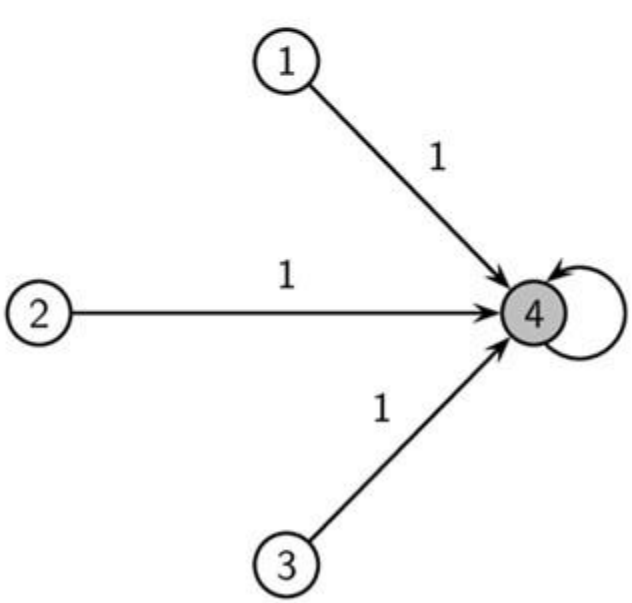
$G_t \leftarrow$ directed graph induced by $\boldsymbol{M}_t$

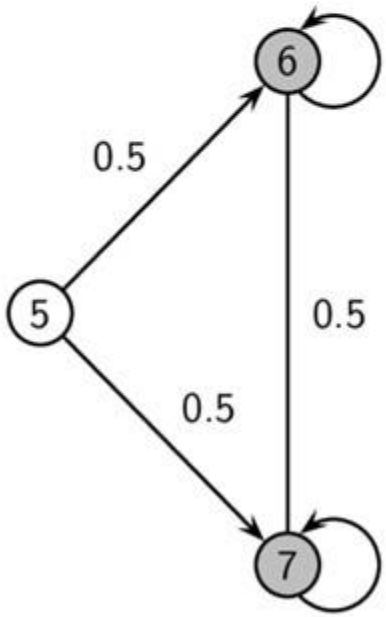
$\mathcal{C} \leftarrow \{$weakly connected components in $G_t\}$

MCL Attractors and Clusters

$r = 2.5$

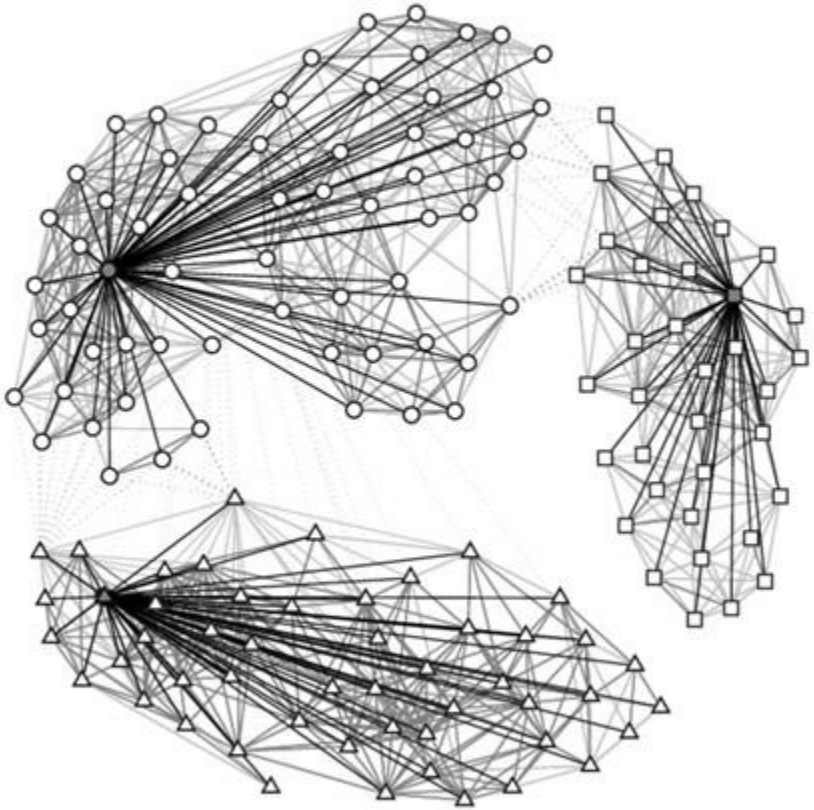




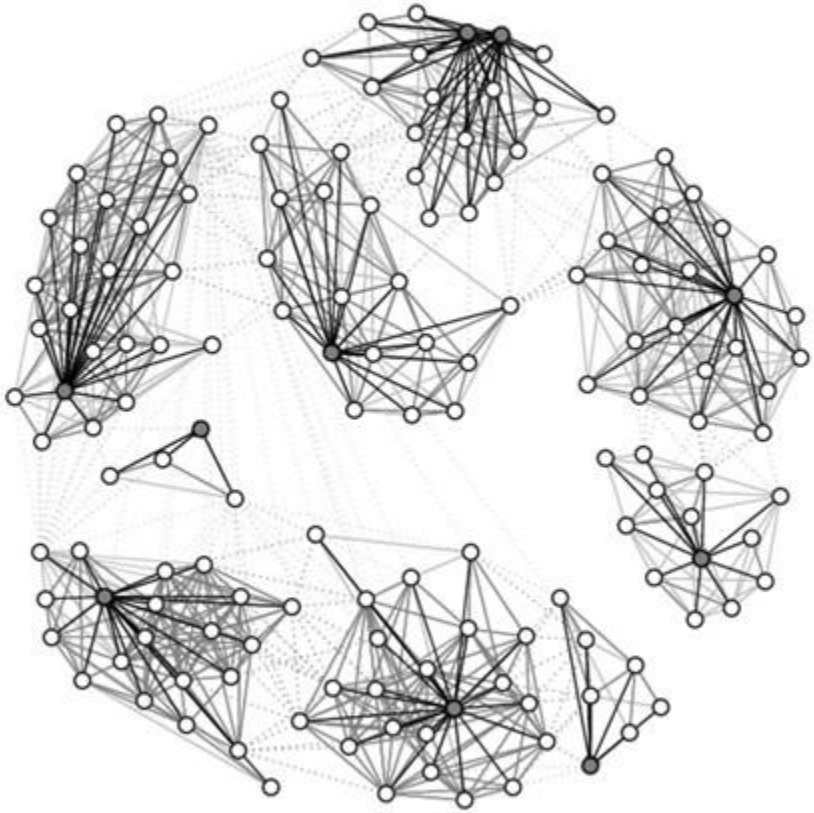


MCL on Iris Graph

Comparison between two values of r



(a) $r = 1.3$



(b) $r = 2$

Contingency Table: MCL Clusters versus Iris Types

$r = 1.3$

There is a quite significant number of `iris-virginica` misplaced.

	`iris-setosa`	`iris-virginica`	`iris-versicolor`
C_1 (triangle)	50	0	1
C_2 (square)	0	36	0
C_3 (circle)	0	14	49

Data mining and Machine learning

Part 17. Clustering Validation

Clustering Validation and Evaluation

Cluster validation and assessment encompasses three main tasks: *clustering evaluation* seeks to assess the goodness or quality of the clustering, *clustering stability* seeks to understand the sensitivity of the clustering result to various algorithmic parameters, for example, the number of clusters, and *clustering tendency* assesses the suitability of applying clustering in the first place, that is, whether the data has any inherent grouping structure.

Validity measures can be divided into three main types:

External: External validation measures employ criteria that are not inherent to the dataset, e.g., class labels.

Internal: Internal validation measures employ criteria that are derived from the data itself, e.g., intracluster and intercluster distances.

Relative: Relative validation measures aim to directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm.

External Measures

External measures assume that the correct or ground-truth clustering is known *a priori*, which is used to evaluate a given clustering.

Let $\boldsymbol{D} = \{\boldsymbol{x}_i\}_{i=1}^n$ be a dataset consisting of n points in a d-dimensional space, partitioned into k clusters. Let $y_i \in \{1, 2, \ldots, k\}$ denote the ground-truth cluster membership or label information for each point.

The ground-truth clustering is given as $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$, where the cluster T_j consists of all the points with label j, i.e., $T_j = \{\boldsymbol{x}_i \in \boldsymbol{D} | y_i = j\}$. We refer to $\mathcal{T}$ as the ground-truth *partitioning*, and to each T_i as a *partition*.

Let $\mathcal{C} = \{C_1, \ldots, C_r\}$ denote a clustering of the same dataset into r clusters, obtained via some clustering algorithm, and let $\hat{y}_i \in \{1, 2, \ldots, r\}$ denote the cluster label for $\boldsymbol{x}_i$.

External Measures

External evaluation measures try capture the extent to which points from the same partition appear in the same cluster, and the extent to which points from different partitions are grouped in different clusters.

All of the external measures rely on the $r \times k$ *contingency table* $\boldsymbol{N}$ that is induced by a clustering $\mathcal{C}$ and the ground-truth partitioning $\mathcal{T}$, defined as follows

$$\boldsymbol{N}(i,j) = n_{ij} = |C_i \cap T_j|$$

The count n_{ij} denotes the number of points that are common to cluster C_i and ground-truth partition T_j.

Let $n_i = |C_i|$ denote the number of points in cluster C_i, and let $m_j = |T_j|$ denote the number of points in partition T_j.

The contingency table can be computed from $\mathcal{T}$ and $\mathcal{C}$ in $O(n)$ time by examining the partition and cluster labels, y_i and $\hat{y}_i$, for each point $\boldsymbol{x}_i \in \boldsymbol{D}$ and incrementing the corresponding count $n_{y_i \hat{y}_i}$.

Matching Based Measures: Purity

Purity quantifies the extent to which a cluster C_i contains entities from only one partition:

$$purity_i = \frac{1}{n_i} \max_{j=1}^{k} \{n_{ij}\}$$

The purity of clustering $\mathcal{C}$ is defined as the weighted sum of the clusterwise purity values:

$$purity = \sum_{i=1}^{r} \frac{n_i}{n} purity_i = \frac{1}{n} \sum_{i=1}^{r} \max_{j=1}^{k} \{n_{ij}\}$$

where the ratio $\frac{n_i}{n}$ denotes the fraction of points in cluster C_i.

Matching Based Measures: Maximum Matching

The maximum matching measure selects the mapping between clusters and partitions, such that the sum of the number of common points (n_{ij}) is maximized, provided that only one cluster can match with a given partition.

Let G be a bipartite graph over the vertex set $V = \mathcal{C} \cup \mathcal{T}$, and let the edge set be $E = \{(C_i, T_j)\}$ with edge weights $w(C_i, T_j) = n_{ij}$. A *matching* M in G is a subset of E, such that the edges in M are pairwise nonadjacent, that is, they do not have a common vertex.

The *maximum weight matching* in G is given as:

$$match = \arg\max_{M} \left\{ \frac{w(M)}{n} \right\}$$

where $w(M)$ is the sum of the sum of all the edge weights in matching M, given as $w(M) = \sum_{e \in M} w(e)$

Matching Based Measures: F-measure

Given cluster C_i, let j_i denote the partition that contains the maximum number of points from C_i, that is, $j_i = \max_{j=1}^{k}\{n_{ij}\}$.

The *precision* of a cluster C_i is the same as its purity:

$$prec_i = \frac{1}{n_i}\max_{j=1}^{k}\{n_{ij}\} = \frac{n_{ij_i}}{n_i}$$

The *recall* of cluster C_i is defined as

$$recall_i = \frac{n_{ij_i}}{|T_{j_i}|} = \frac{n_{ij_i}}{m_{j_i}}$$

where $m_{j_i} = |T_{j_i}|$.

Matching Based Measures: F-measure

The F-measure is the harmonic mean of the precision and recall values for each C_i

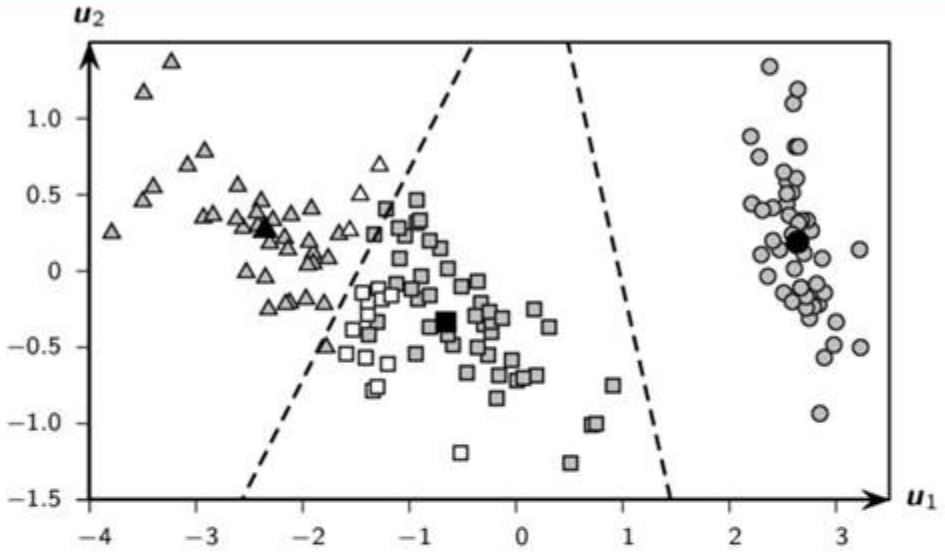
$$F_i = \frac{2}{\frac{1}{prec_i} + \frac{1}{recall_i}} = \frac{2 \cdot prec_i \cdot recall_i}{prec_i + recall_i} = \frac{2\, n_{ij_i}}{n_i + m_{j_i}}$$

The F-measure for the clustering $\mathcal{C}$ is the mean of clusterwise F-meaure values:

$$F = \frac{1}{r} \sum_{i=1}^{r} F_i$$

K-means: Iris Principal Components Data

Good Case



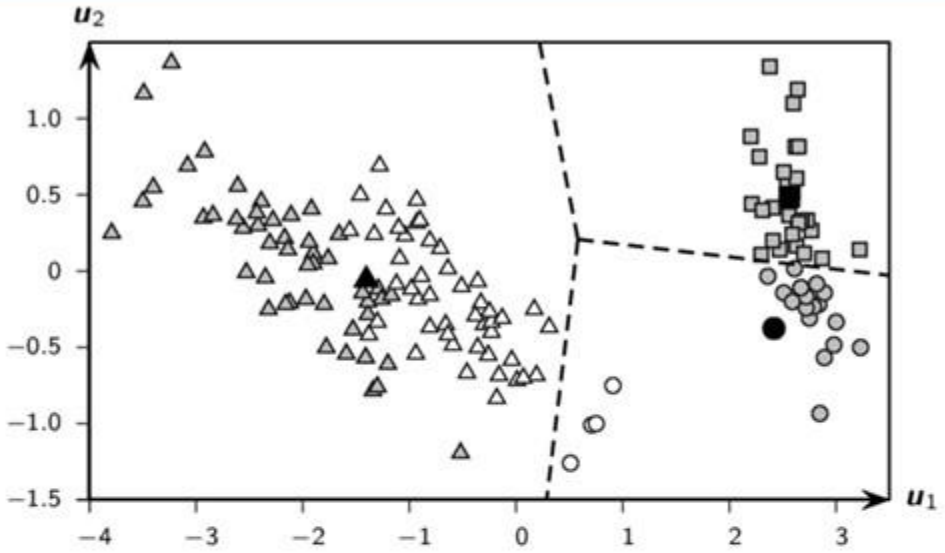
Contingency table:

	iris-setosa T_1	iris-versicolor T_2	iris-virginica T_3	n_i
C_1(squares)	0	47	14	61
C_2(circles)	50	0	0	50
C_3(triangles)	0	3	36	39
m_j	50	50	50	$n = 100$

$purity = 0.887$, $match = 0.887$, $F = 0.885$.

K-means: Iris Principal Components Data

Bad Case



Contingency table:

	iris-setosa T_1	iris-versicolor T_2	iris-virginica T_3	n_i
C_1(*squares*)	30	0	0	30
C_2(*circles*)	20	4	0	24
C_3(*triangles*)	0	46	50	96
m_j	50	50	50	$n = 150$

$purity = 0.667$, $match = 0.560$, $F = 0.658$

Entropy-based Measures: Conditional Entropy

The entropy of a clustering $\mathcal{C}$ and partitioning $\mathcal{T}$ is given as

$$H(\mathcal{C}) = -\sum_{i=1}^{r} p_{C_i} \log p_{C_i} \qquad H(\mathcal{T}) = -\sum_{j=1}^{k} p_{T_j} \log p_{T_j}$$

where $p_{C_i} = \frac{n_i}{n}$ and $p_{T_j} = \frac{m_j}{n}$ are the probabilities of cluster C_i and partition T_j.

The cluster-specific entropy of $\mathcal{T}$, that is, the conditional entropy of $\mathcal{T}$ with respect to cluster C_i is defined as

$$H(\mathcal{T}|C_i) = -\sum_{j=1}^{k} \left(\frac{n_{ij}}{n_i}\right) \log \left(\frac{n_{ij}}{n_i}\right)$$

Entropy-based Measures: Conditional Entropy

The conditional entropy of $\mathcal{T}$ given clustering $\mathcal{C}$ is defined as the weighted sum:

$$H(\mathcal{T}|\mathcal{C}) = \sum_{i=1}^{r} \frac{n_i}{n} H(\mathcal{T}|C_i) = \boxed{-\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log\left(\frac{p_{ij}}{p_{C_i}}\right)}$$

$$= H(\mathcal{C},\mathcal{T}) - H(\mathcal{C})$$

where $p_{ij} = \frac{n_{ij}}{n}$ is the probability that a point in cluster i also belongs to partition and where $H(\mathcal{C},\mathcal{T}) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log p_{ij}$ is the joint entropy of $\mathcal{C}$ and $\mathcal{T}$.

$H(\mathcal{T}|\mathcal{C}) = 0$ if and only if $\mathcal{T}$ is completely determined by $\mathcal{C}$, corresponding to the ideal clustering. If $\mathcal{C}$ and $\mathcal{T}$ are independent of each other, then $H(\mathcal{T}|\mathcal{C}) = H(\mathcal{T})$.

Entropy-based Measures: Normalized Mutual Information

The *mutual information* tries to quantify the amount of shared information between the clustering $\mathcal{C}$ and partitioning $\mathcal{T}$, and it is defined as

$$I(\mathcal{C},\mathcal{T}) = \sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log\left(\frac{p_{ij}}{p_{C_i} \cdot p_{T_j}}\right)$$

When $\mathcal{C}$ and $\mathcal{T}$ are independent then $p_{ij} = p_{C_i} \cdot p_{T_j}$, and thus $I(\mathcal{C},\mathcal{T}) = 0$. However, there is no upper bound on the mutual information.

The *normalized mutual information* (NMI) is defined as the geometric mean:

$$NMI(\mathcal{C},\mathcal{T}) = \sqrt{\frac{I(\mathcal{C},\mathcal{T})}{H(\mathcal{C})} \cdot \frac{I(\mathcal{C},\mathcal{T})}{H(\mathcal{T})}} = \frac{I(\mathcal{C},\mathcal{T})}{\sqrt{H(\mathcal{C}) \cdot H(\mathcal{T})}}$$

The NMI value lies in the range $[0,1]$. Values close to 1 indicate a good clustering.

Entropy-based Measures: Variation of Information

This criterion is based on the mutual information between the clustering $\mathcal{C}$ and the ground-truth partitioning $\mathcal{T}$, and their entropy; it is defined as

$$\begin{aligned} VI(\mathcal{C},\mathcal{T}) &= (H(\mathcal{T}) - I(\mathcal{C},\mathcal{T})) + (H(\mathcal{C}) - I(\mathcal{C},\mathcal{T})) \\ &= H(\mathcal{T}) + H(\mathcal{C}) - 2I(\mathcal{C},\mathcal{T}) \end{aligned}$$

Variation of information (VI) is zero only when $\mathcal{C}$ and $\mathcal{T}$ are identical. Thus, the lower the VI value the better the clustering $\mathcal{C}$.

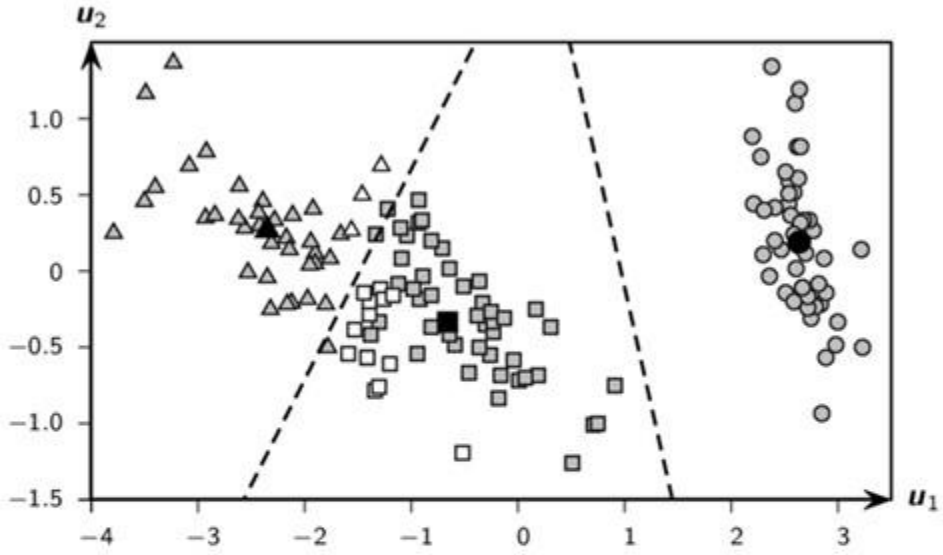
VI can also be expressed as:

$$VI(\mathcal{C},\mathcal{T}) = H(\mathcal{T}|\mathcal{C}) + H(\mathcal{C}|\mathcal{T})$$

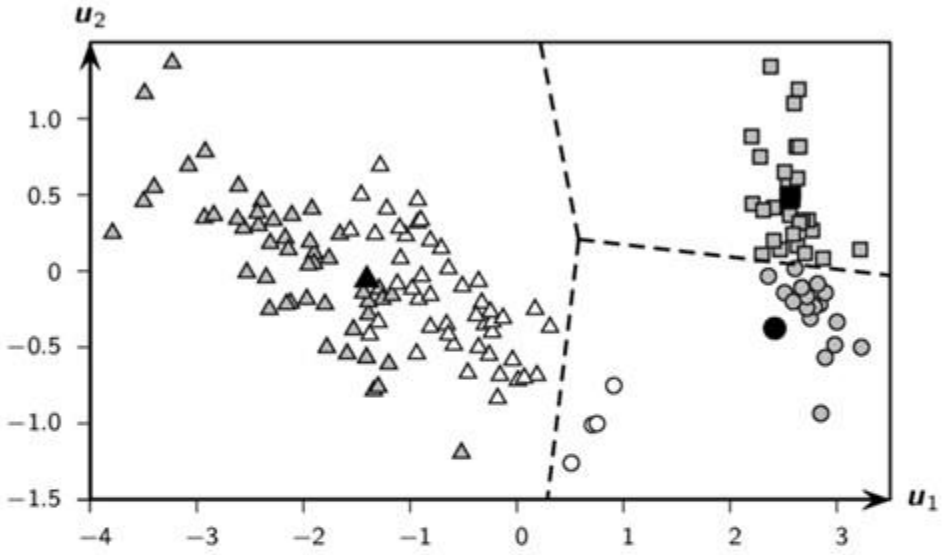
$$\boxed{VI(\mathcal{C},\mathcal{T}) = 2H(\mathcal{T},\mathcal{C}) - H(\mathcal{T}) - H(\mathcal{C})}$$

K-means: Iris Principal Components Data

Good Case



(a) K-means: good



(b) K-means: bad

	purity	*match*	*F*	$H(\mathcal{T}\|\mathcal{C})$	*NMI*	*VI*
(a) Good	0.887	0.887	0.885	0.418	0.742	0.812
(b) Bad	0.667	0.560	0.658	0.743	0.587	1.200

Pairwise Measures

Given clustering $\mathcal{C}$ and ground-truth partitioning $\mathcal{T}$, let $\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{D}$ be any two points, with $i \neq j$. Let y_i denote the true partition label and let $\hat{y}_i$ denote the cluster label for point $\boldsymbol{x}_i$.

If both $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same cluster, that is, $\hat{y}_i = \hat{y}_j$, we call it a *positive* event, and if they do not belong to the same cluster, that is, $\hat{y}_i \neq \hat{y}_j$, we call that a *negative* event. Depending on whether there is agreement between the cluster labels and partition labels, there are four possibilities to consider:

True Positives: $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same partition in $\mathcal{T}$, and they are also in the same cluster in $\mathcal{C}$. The number of true positive pairs is given as

$$TP = \left|\{(\boldsymbol{x}_i, \boldsymbol{x}_j) : y_i = y_j \text{ and } \hat{y}_i = \hat{y}_j\}\right|$$

False Negatives: $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same partition in $\mathcal{T}$, but they do not belong to the same cluster in $\mathcal{C}$. The number of all false negative pairs is given as

$$FN = \left|\{(\boldsymbol{x}_i, \boldsymbol{x}_j) : y_i = y_j \text{ and } \hat{y}_i \neq \hat{y}_j\}\right|$$

Pairwise Measures

False Positives: $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ do not belong to the same partition in $\mathcal{T}$, but they do belong to the same cluster in $\mathcal{C}$. The number of false positive pairs is given as

$$FP = \left|\{(\boldsymbol{x}_i, \boldsymbol{x}_j) : y_i \neq y_j \text{ and } \hat{y}_i = \hat{y}_j\}\right|$$

True Negatives: $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ neither belong to the same partition in $\mathcal{T}$, nor do they belong to the same cluster in $\mathcal{C}$. The number of such true negative pairs is given as

$$TN = \left|\{(\boldsymbol{x}_i, \boldsymbol{x}_j) : y_i \neq y_j \text{ and } \hat{y}_i \neq \hat{y}_j\}\right|$$

Because there are $N = \binom{n}{2} = \frac{n(n-1)}{2}$ pairs of points, we have the following identity:

$$N = TP + FN + FP + TN$$

Pairwise Measures: TP, TN, FP, FN

They can be computed efficiently using the contingency table $\boldsymbol{N} = \{n_{ij}\}$. The number of true positives is given as

$$TP = \frac{1}{2}\left(\left(\sum_{i=1}^{r}\sum_{j=1}^{k} n_{ij}^2\right) - n\right)$$

The false negatives can be computed as

$$FN = \frac{1}{2}\left(\sum_{j=1}^{k} m_j^2 - \sum_{i=1}^{r}\sum_{j=1}^{k} n_{ij}^2\right)$$

The number of false positives are:

$$FP = \frac{1}{2}\left(\sum_{i=1}^{r} n_i^2 - \sum_{i=1}^{r}\sum_{j=1}^{k} n_{ij}^2\right)$$

Finally, the number of true negatives can be obtained via

$$TN = N - (TP + FN + FP) = \frac{1}{2}\left(n^2 - \sum_{i=1}^{r} n_i^2 - \sum_{j=1}^{k} m_j^2 + \sum_{i=1}^{r}\sum_{j=1}^{k} n_{ij}^2\right)$$

Pairwise Measures: Jaccard Coefficient, Rand Statistic

Jaccard Coefficient: measures the fraction of true positive point pairs, but after ignoring the true negative:

$$Jaccard = \frac{TP}{TP + FN + FP}$$

Rand Statistic: measures the fraction of true positives and true negatives over all point pairs:

$$Rand = \frac{TP + TN}{N}$$

Pairwise Measures: FM Measure

Fowlkes-Mallows Measure: Define the overall *pairwise precision* and *pairwise recall* values for a clustering $\mathcal{C}$, as follows:

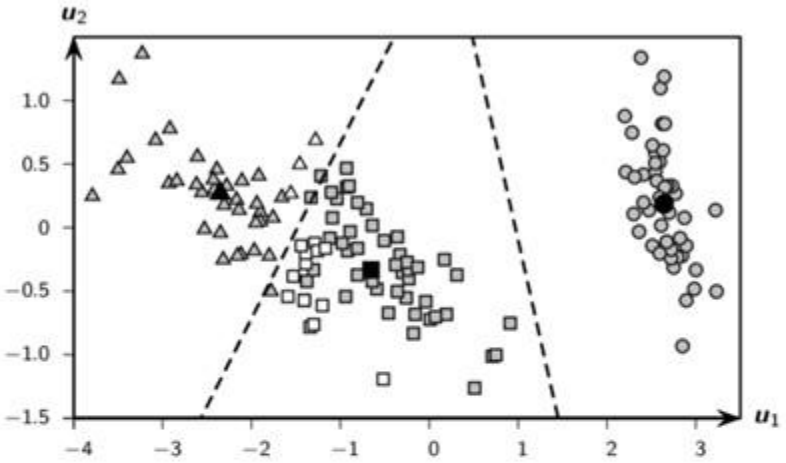
$$prec = TP/TP + FP \qquad\qquad recall = TP/TP + FN$$

The Fowlkes–Mallows (FM) measure is defined as the geometric mean of the pairwise precision and recall

$$FM = \sqrt{prec \cdot recall} = \frac{TP}{\sqrt{(TP + FN)(TP + FP)}}$$

K-means: Iris Principal Components Data

Good Case



Contingency table:

	setosa T_1	**versicolor** T_2	**virginica** T_3
C_1	0	47	14
C_2	50	0	0
C_3	0	3	36

The number of true positives is:

$$TP = \binom{47}{2} + \binom{14}{2} + \binom{50}{2} + \binom{3}{2} + \binom{36}{2} = 3030$$

Likewise, we have $FN = 645$, $FP = 766$, $TN = 6734$, and $N = \binom{150}{2} = 11175$.
We therefore have: $Jaccard = 0.682$, $Rand = 0.887$, $FM = 0.811$.
For the "bad" clustering, we have: $Jaccard = 0.477$, $Rand = 0.717$, $FM = 0.657$.

Correlation Measures: Hubert statistic

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two symmetric $n \times n$ matrices, and let $N = \binom{n}{2}$. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$ denote the vectors obtained by linearizing the upper triangular elements (excluding the main diagonal) of $\boldsymbol{X}$ and $\boldsymbol{Y}$.

Let μ_X denote the element-wise mean of $\boldsymbol{x}$, given as

$$\mu_X = \frac{1}{N} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \boldsymbol{X}(i,j) = \frac{1}{N} \boldsymbol{x}^T \boldsymbol{x}$$

and let $\boldsymbol{z}_x$ denote the centered $\boldsymbol{x}$ vector, defined as $\boldsymbol{z}_x = \boldsymbol{x} - 1 \cdot \mu_X$

The Hubert statistic is defined as

$$\Gamma = \frac{1}{N} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \boldsymbol{X}(i,j) \cdot \boldsymbol{Y}(i,j) = \frac{1}{N} \boldsymbol{x}^T \boldsymbol{y}$$

The normalized Hubert statistic is defined as the element-wise correlation

$$\Gamma_n = \frac{\boldsymbol{z}_x^T \boldsymbol{z}_y}{\|\boldsymbol{z}_x\| \cdot \|\boldsymbol{z}_y\|} = \cos\theta$$

Correlation-based Measure: Discretized Hubert Statistic

Let $\boldsymbol{T}$ and $\boldsymbol{C}$ be the $n \times n$ matrices defined as

$$\boldsymbol{T}(i,j) = \begin{cases} 1 & \text{if } y_i = y_j, i \neq j \\ 0 & \text{otherwise} \end{cases} \qquad \boldsymbol{C}(i,j) = \begin{cases} 1 & \text{if } \hat{y}_i = \hat{y}_j, i \neq j \\ 0 & \text{otherwise} \end{cases}$$

Let $\boldsymbol{t}, \boldsymbol{c} \in \mathbb{R}^N$ denote the N-dimensional vectors comprising the upper triangular elements (excluding the diagonal) of $\boldsymbol{T}$ and $\boldsymbol{C}$. Let $\boldsymbol{z}_t$ and $\boldsymbol{z}_c$ denote the centered $\boldsymbol{t}$ and $\boldsymbol{c}$ vectors.

The discretized Hubert statistic is computed by setting $\boldsymbol{x} = \boldsymbol{t}$ and $\boldsymbol{y} = \boldsymbol{c}$:

$$\Gamma = \frac{1}{N} \boldsymbol{t}^T \boldsymbol{c} = \frac{TP}{N}$$

The normalized version of the discretized Hubert statistic is simply the correlation between $\boldsymbol{t}$ and $\boldsymbol{c}$

$$\Gamma_n = \frac{\boldsymbol{z}_t^T \boldsymbol{z}_c}{\|\boldsymbol{z}_t\| \cdot \|\boldsymbol{z}_c\|} = \frac{\frac{TP}{N} - \mu_T \mu_C}{\sqrt{\mu_T \mu_C (1-\mu_T)(1-\mu_C)}}$$

where $\mu_T = \frac{TP+FN}{N}$ and $\mu_C = \frac{TP+FP}{N}$.

Internal Measures

Internal evaluation measures do not have recourse to the ground-truth partitioning. To evaluate the quality of the clustering, internal measures therefore have to utilize notions of intracluster similarity or compactness, contrasted with notions of intercluster separation, with usually a trade-off in maximizing these two aims.

The internal measures are based on the $n \times n$ *distance matrix*, also called the *proximity matrix*, of all pairwise distances among the n points:

$$\boldsymbol{W} = \left\{ \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \right\}_{i,j=1}^{n} \tag{1}$$

where $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ is the Euclidean distance between $\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{D}$.

The proximity matrix $\boldsymbol{W}$ is the adjacency matrix of the weighted complete graph G over the n points, that is, with nodes $V = \{\boldsymbol{x}_i \mid \boldsymbol{x}_i \in \boldsymbol{D}\}$, edges $E = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid \boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{D}\}$, and edge weights $w_{ij} = \boldsymbol{W}(i,j)$ for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{D}$.

Internal Measures

The clustering $\mathcal{C}$ can be considered as a k-way cut in G. Given any subsets $S, R \subset V$, define $W(S, R)$ as the sum of the weights on all edges with one vertex in S and the other in R, given as

$$W(S, R) = \sum_{x_i \in S} \sum_{x_j \in R} w_{ij}$$

We denote by $\overline{S} = V - S$ the complementary set of vertices.

The sum of all the intracluster and intercluster weights are given as

$$W_{in} = \frac{1}{2} \sum_{i=1}^{k} W(C_i, C_i) \qquad W_{out} = \frac{1}{2} \sum_{i=1}^{k} W(C_i, \overline{C_i}) = \sum_{i=1}^{k-1} \sum_{j>i} W(C_i, C_j)$$

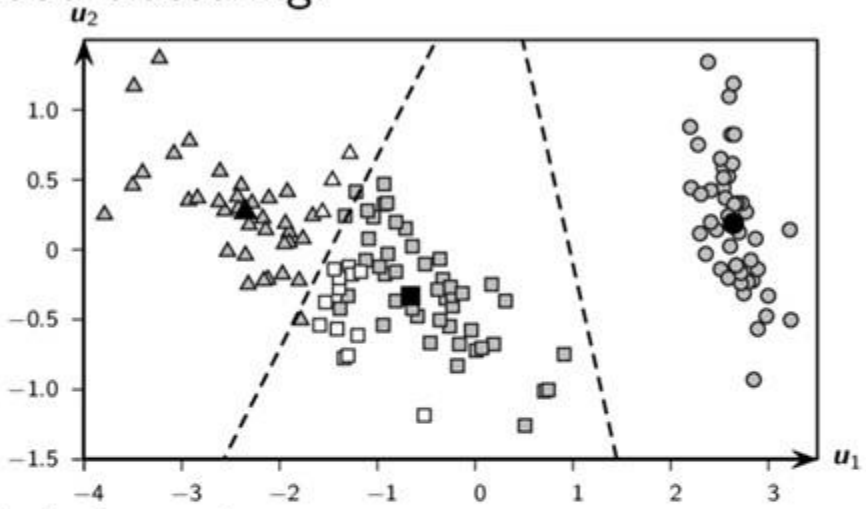
The number of distinct intracluster and intercluster edges is given as

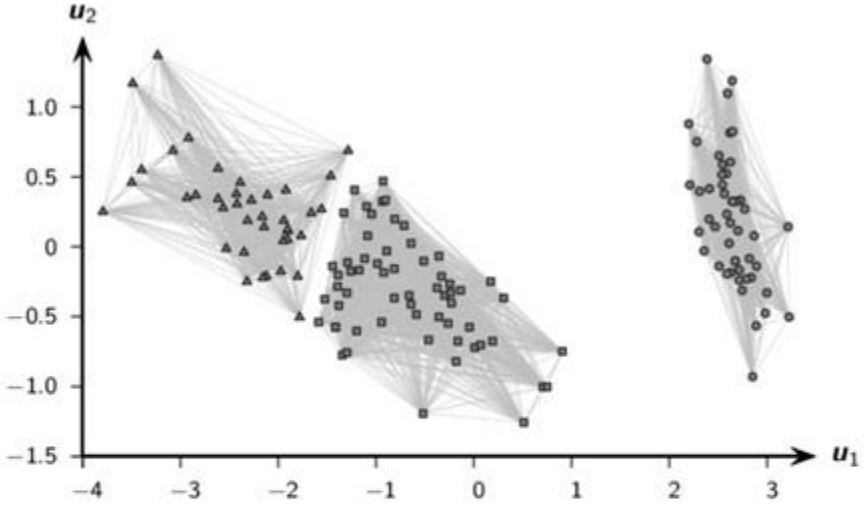
$$N_{in} = \sum_{i=1}^{k} \binom{n_i}{2} \qquad N_{out} = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} n_i \cdot n_j$$

Clusterings as Graphs: Iris

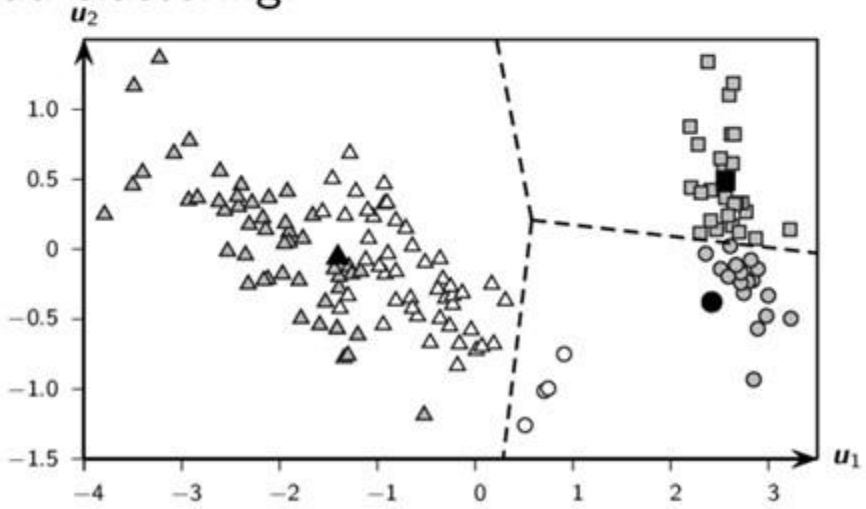
Only intracluster edges shown.

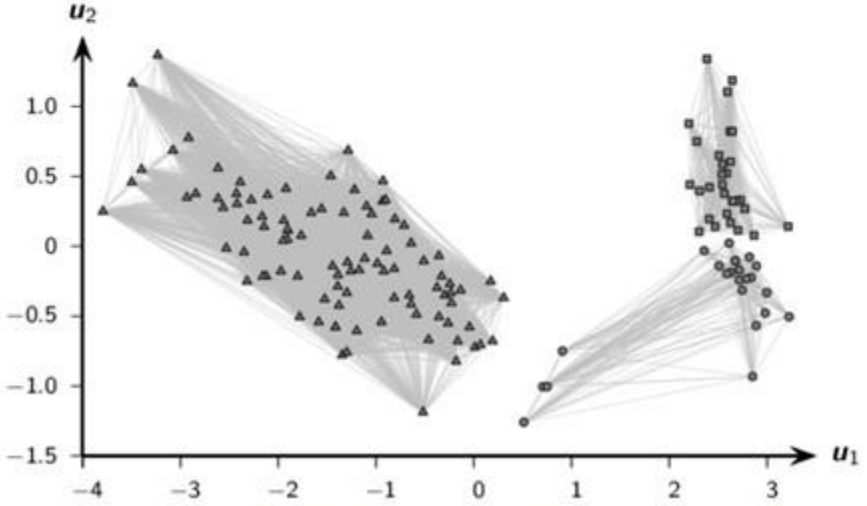
Good clustering.





Bad clustering.





Internal Measures: BetaCV and C-index

BetaCV Measure: The BetaCV measure is the ratio of the mean intracluster distance to the mean intercluster distance:

$$BetaCV = \frac{W_{in}/N_{in}}{W_{out}/N_{out}} = \frac{N_{out}}{N_{in}} \cdot \frac{W_{in}}{W_{out}} = \frac{N_{out}}{N_{in}} \frac{\sum_{i=1}^{k} W(C_i, C_i)}{\sum_{i=1}^{k} W(C_i, \overline{C_i})}$$

The smaller the BetaCV ratio, the better the clustering.

C-index: Let $W_{\min}(N_{in})$ be the sum of the smallest N_{in} distances in the proximity matrix $\boldsymbol{W}$, where N_{in} is the total number of intracluster edges, or point pairs. Let $W_{\max}(N_{in})$ be the sum of the largest N_{in} distances in $\boldsymbol{W}$.

The C-index measures to what extent the clustering puts together the N_{in} points that are the closest across the k clusters. It is defined as

$$Cindex = \frac{W_{in} - W_{\min}(N_{in})}{W_{\max}(N_{in}) - W_{\min}(N_{in})}$$

The C-index lies in the range $[0, 1]$. The smaller the C-index, the better the clustering.

Internal Measures: Normalized Cut and Modularity

Normalized Cut Measure: The normalized cut objective for graph clustering can also be used as an internal clustering evaluation measure:

$$NC = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{vol(C_i)} = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{W(C_i, V)}$$

where $vol(C_i) = W(C_i, V)$ is the volume of cluster C_i. The higher the normalized cut value the better.

Modularity: The modularity objective is given as

$$Q = \sum_{i=1}^{k} \left(\frac{W(C_i, C_i)}{W(V, V)} - \left(\frac{W(C_i, V)}{W(V, V)} \right)^2 \right)$$

The smaller the modularity measure the better the clustering.

Internal Measures: Dunn Index

The Dunn index is defined as the ratio between the minimum distance between point pairs from different clusters and the maximum distance between point pairs from the same cluster

$$Dunn = \frac{W_{out}^{\min}}{W_{in}^{\max}}$$

where $W_{out}^{\min}$ is the minimum intercluster distance:

$$W_{out}^{\min} = \min_{i,j>i} \left\{ w_{ab} | \boldsymbol{x}_a \in C_i, \boldsymbol{x}_b \in C_j \right\}$$

and $W_{in}^{\max}$ is the maximum intracluster distance:

$$W_{in}^{\max} = \max_{i} \left\{ w_{ab} | \boldsymbol{x}_a, \boldsymbol{x}_b \in C_i \right\}$$

The larger the Dunn index the better the clustering because it means even the closest distance between points in different clusters is much larger than the farthest distance between points in the same cluster.

Internal Measures: Davies-Bouldin Index

Let μ_i denote the cluster mean

$$\mu_i = \frac{1}{n_i} \sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$$

Let σ_{μ_i} denote the dispersion or spread of the points around the cluster mean

$$\sigma_{\mu_i} = \sqrt{\frac{\sum_{\boldsymbol{x}_j \in C_i} \delta(\boldsymbol{x}_j, \mu_i)^2}{n_i}} = \sqrt{var(C_i)}$$

The Davies–Bouldin measure for a pair of clusters C_i and C_j is defined as the ratio

$$DB_{ij} = \frac{\sigma_{\mu_i} + \sigma_{\mu_j}}{\delta(\mu_i, \mu_j)}$$

DB_{ij} measures how compact the clusters are compared to the distance between the cluster means. The Davies–Bouldin index is then defined as

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \{DB_{ij}\}$$

The smaller the DB value the better the clustering

Silhouette Coefficient

Define the silhoutte coefficient of a point x_i as

$$s_i = \frac{\mu_{out}^{\min}(x_i) - \mu_{in}(x_i)}{\max\left\{\mu_{out}^{\min}(x_i), \mu_{in}(x_i)\right\}}$$

where $\mu_{in}(x_i)$ is the mean distance from x_i to points in its own cluster $\hat{y}_i$:

$$\mu_{in}(x_i) = \frac{\sum_{x_j \in C_{\hat{y}_i}, j \neq i} \delta(x_i, x_j)}{n_{\hat{y}_i} - 1}$$

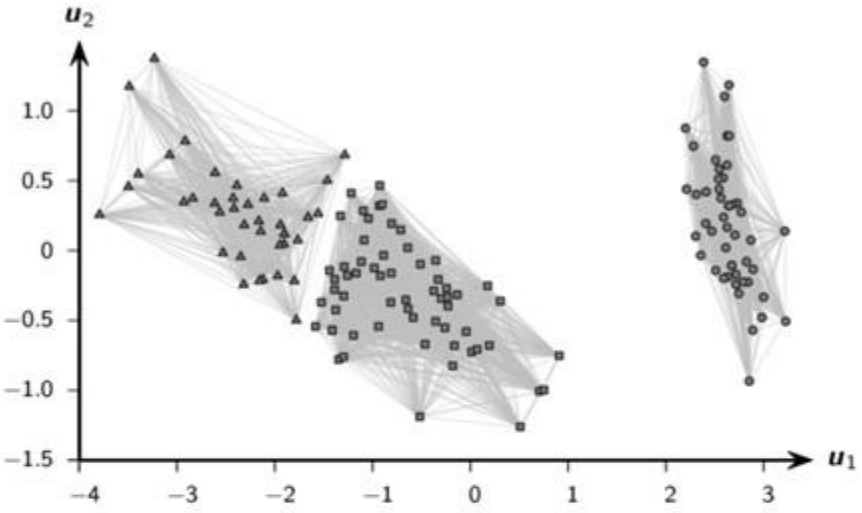
and $\mu_{out}^{\min}(x_i)$ is the mean of the distances from x_i to points in the closest cluster:

$$\mu_{out}^{\min}(x_i) = \min_{j \neq \hat{y}_i} \left\{ \frac{\sum_{y \in C_j} \delta(x_i, y)}{n_j} \right\}$$

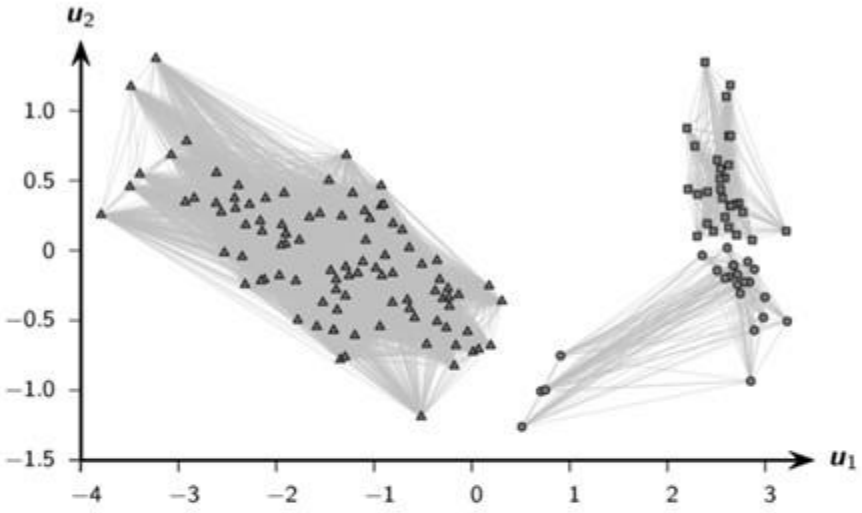
The s_i value lies in the interval $[-1, +1]$. A value close to $+1$ indicates that x_i is much closer to points in its own cluster, a value close to zero indicates x_i is close to the boundary, and a value close to -1 indicates that x_i is much closer to another cluster, and therefore may be mis-clustered.

The silhouette coefficient is the mean s_i value: $SC = \frac{1}{n} \sum_{i=1}^{n} s_i$. A value close to $+1$ indicates a good clustering.

Iris Data: Good vs. Bad Clustering



(a) Good



(b) Bad

	Lower better				Higher better				
	BetaCV	*Cindex*	*Q*	*DB*	*NC*	*Dunn*	*SC*	Γ	Γ_n
(a) Good	0.24	0.034	−0.23	0.65	2.67	0.08	0.60	8.19	0.92
(b) Bad	0.33	0.08	−0.20	1.11	2.56	0.03	0.55	7.32	0.83

Relative Measures: Silhouette Coefficient

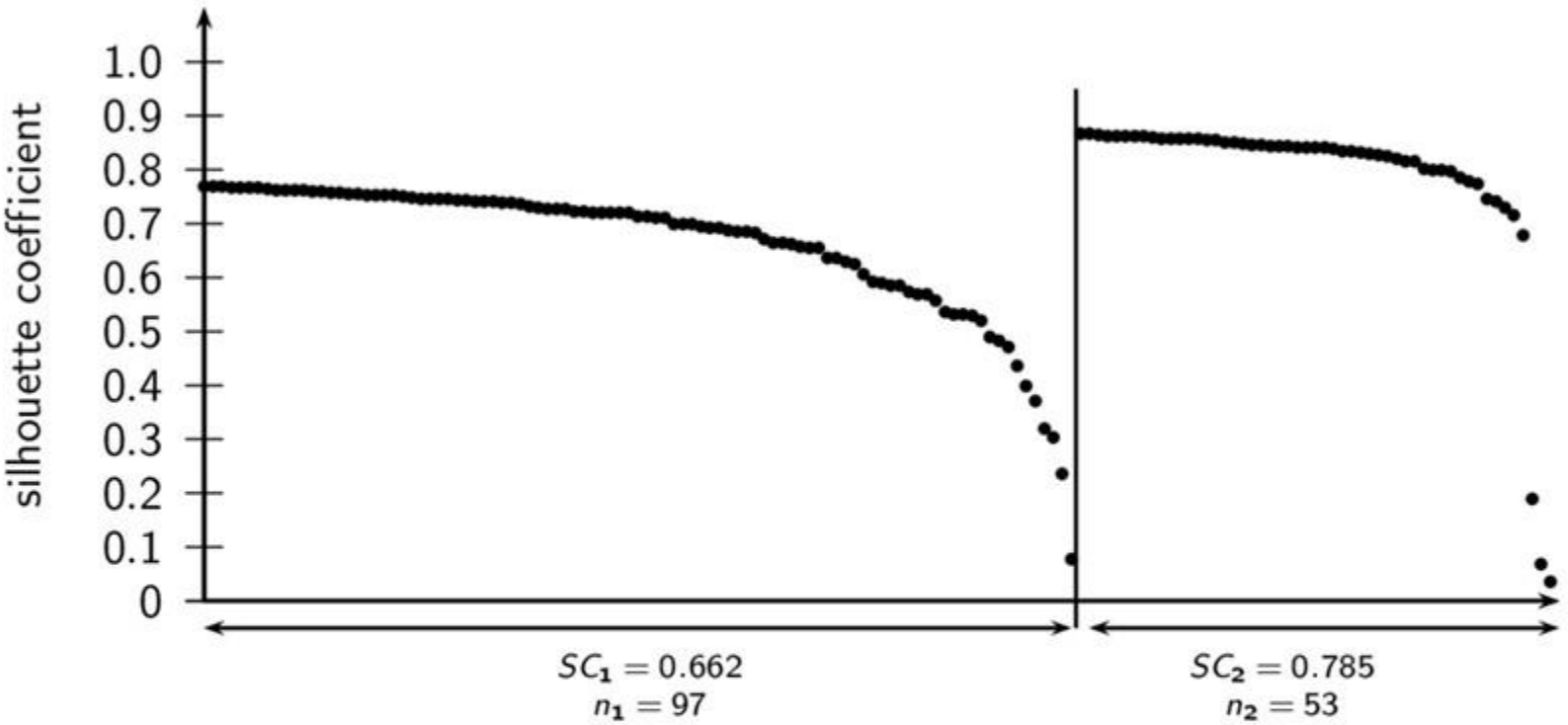
The silhouette coefficient for each point s_j, and the average SC value can be used to estimate the number of clusters in the data.

The approach consists of plotting the s_j values in descending order for each cluster, and to note the overall SC value for a particular value of k, as well as clusterwise SC values:

$$SC_i = \frac{1}{n_i} \sum_{x_j \in C_i} s_j$$

We then pick the value k that yields the best clustering, with many points having high s_j values within each cluster, as well as high values for SC and SC_i $(1 \leq i \leq k)$.

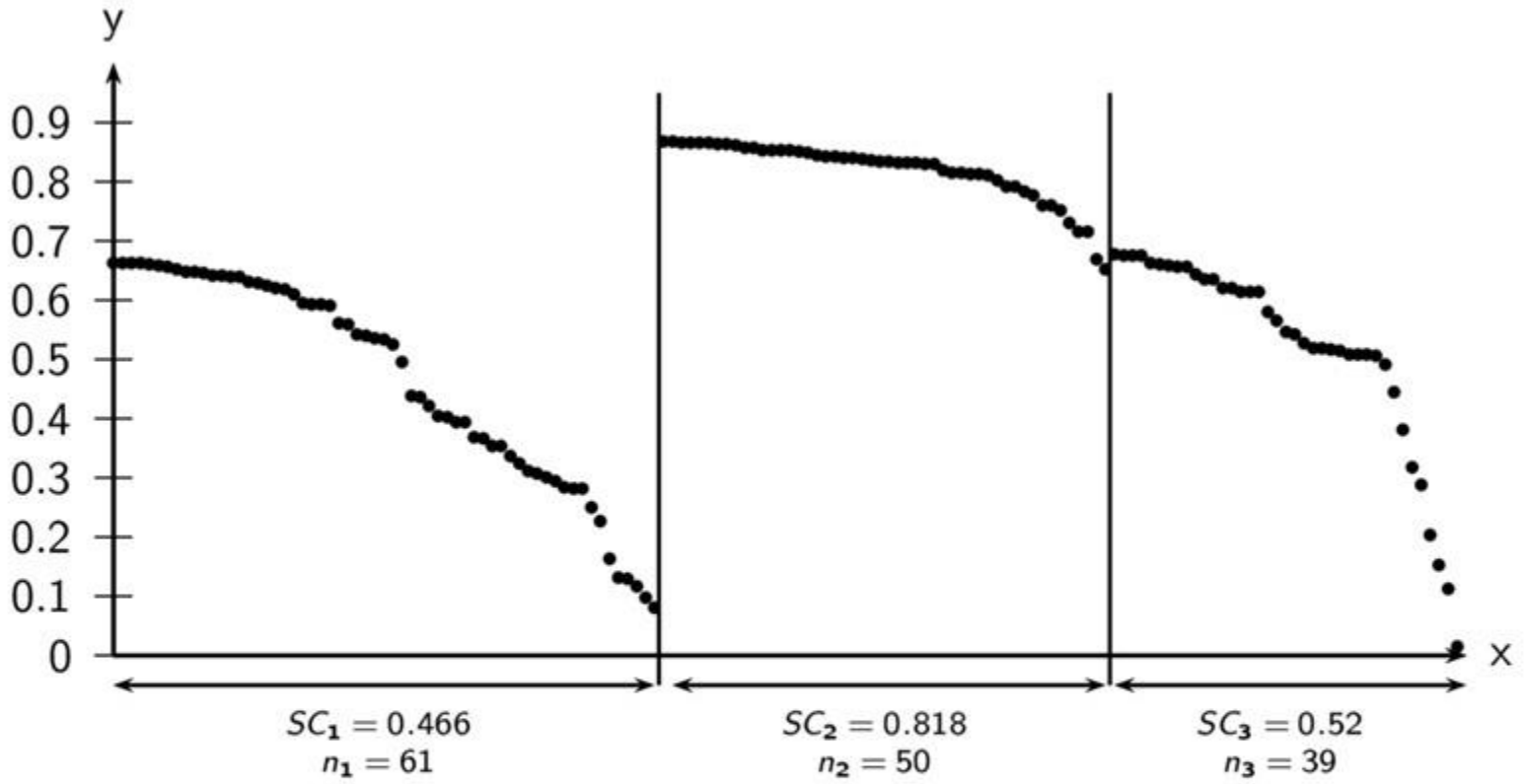
Iris K-means: Silhouette Coefficient Plot ($k = 2$)



(a) $k = 2$, $SC = 0.706$

$k = 2$ yields the highest silhouette coefficient, with the two clusters essentially well separated. C_1 starts out with high s_i values, which gradually drop as we get to border points. C_2 is even better separated, since it has a higher silhouette coefficient and the pointwise scores are all high, except for the last three points.

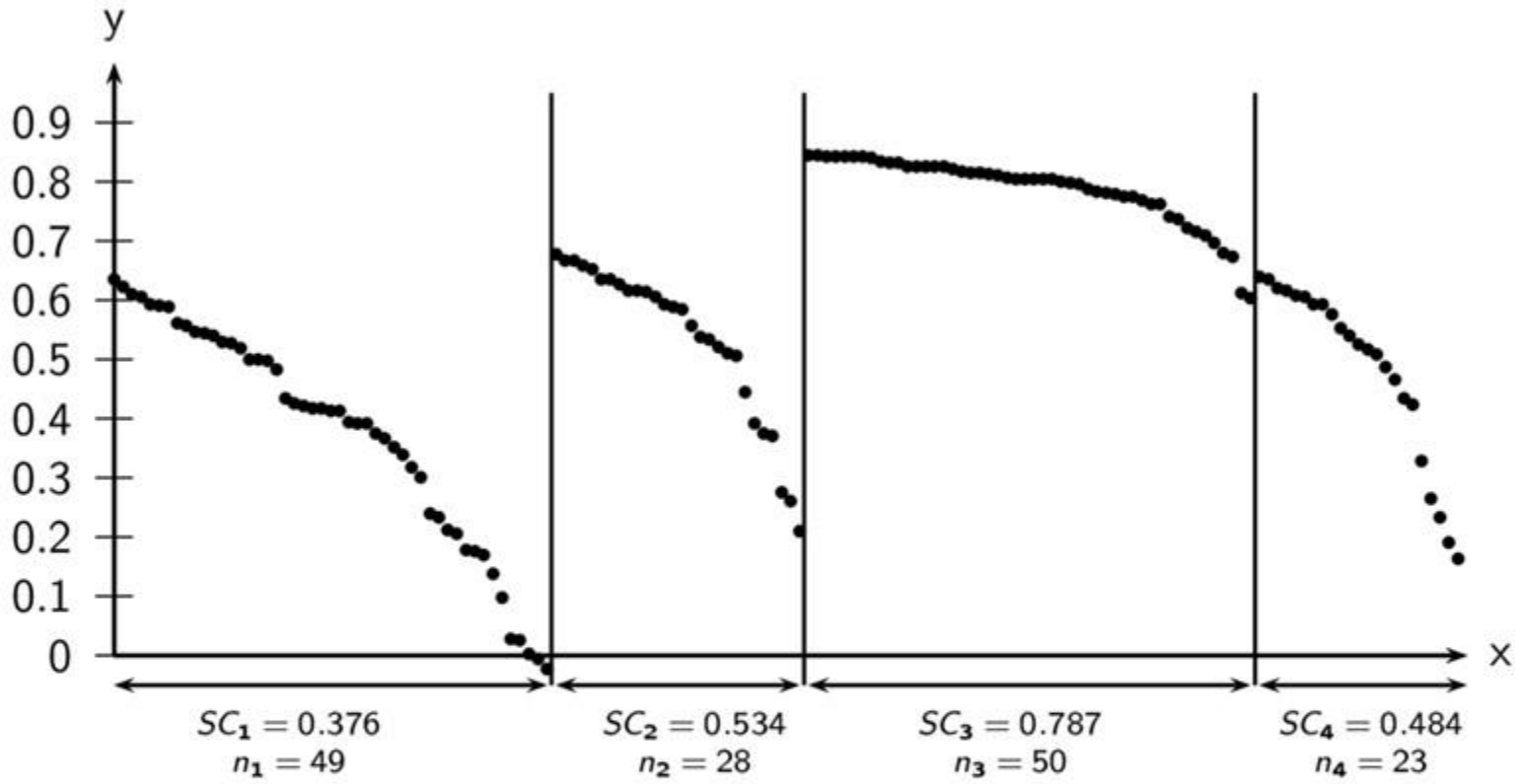
Iris K-means: Silhouette Coefficient Plot ($k = 3$)



(b) $k = 3$, $SC = 0.598$

C_1 from $k = 2$ has been split into two clusters for $k = 3$, namely C_1 and C_3. Both of these have many bordering points, whereas C_2 is well separated with high silhouette coefficients across all points.

Iris K-means: Silhouette Coefficient Plot ($k = 4$)



(c) $k = 4$, $SC = 0.559$

C_3 is the well separated cluster, corresponding to C_2 (in $k = 2$ and $k = 3$), and the remaining clusters are essentially subclusters of C_1 for $k = 2$. Cluster C_1 also has two points with negative s_i values, indicating that they are probably misclustered.

Relative Measures: Calinski–Harabasz Index

Given the dataset $\boldsymbol{D} = \{\boldsymbol{x}_i\}_{i=1}^n$, the scatter matrix for $\boldsymbol{D}$ is given as

$$\boldsymbol{S} = n\Sigma = \sum_{j=1}^{n} (\boldsymbol{x}_j - \boldsymbol{\mu})(\boldsymbol{x}_j - \boldsymbol{\mu})^T$$

where $\boldsymbol{\mu} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{x}_j$ is the mean and Σ is the covariance matrix. The scatter matrix can be decomposed into two matrices $\boldsymbol{S} = \boldsymbol{S}_W + \boldsymbol{S}_B$, where $\boldsymbol{S}_W$ is the within-cluster scatter matrix and $\boldsymbol{S}_B$ is the between-cluster scatter matrix, given as

$$\boldsymbol{S}_W = \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} (\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T$$

$$\boldsymbol{S}_B = \sum_{i=1}^{k} n_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

where $\boldsymbol{\mu}_i = \frac{1}{n_i}\sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$ is the mean for cluster C_i.

Relative Measures: Calinski–Harabasz Index

The Calinski–Harabasz (CH) variance ratio criterion for a given value of k is defined as follows:

$$CH(k) = \frac{tr(\boldsymbol{S}_B)/(k-1)}{tr(\boldsymbol{S}_W)/(n-k)} = \frac{n-k}{k-1} \cdot \frac{tr(\boldsymbol{S}_B)}{tr(\boldsymbol{S}_W)}$$

where tr is the trace of the matrix.

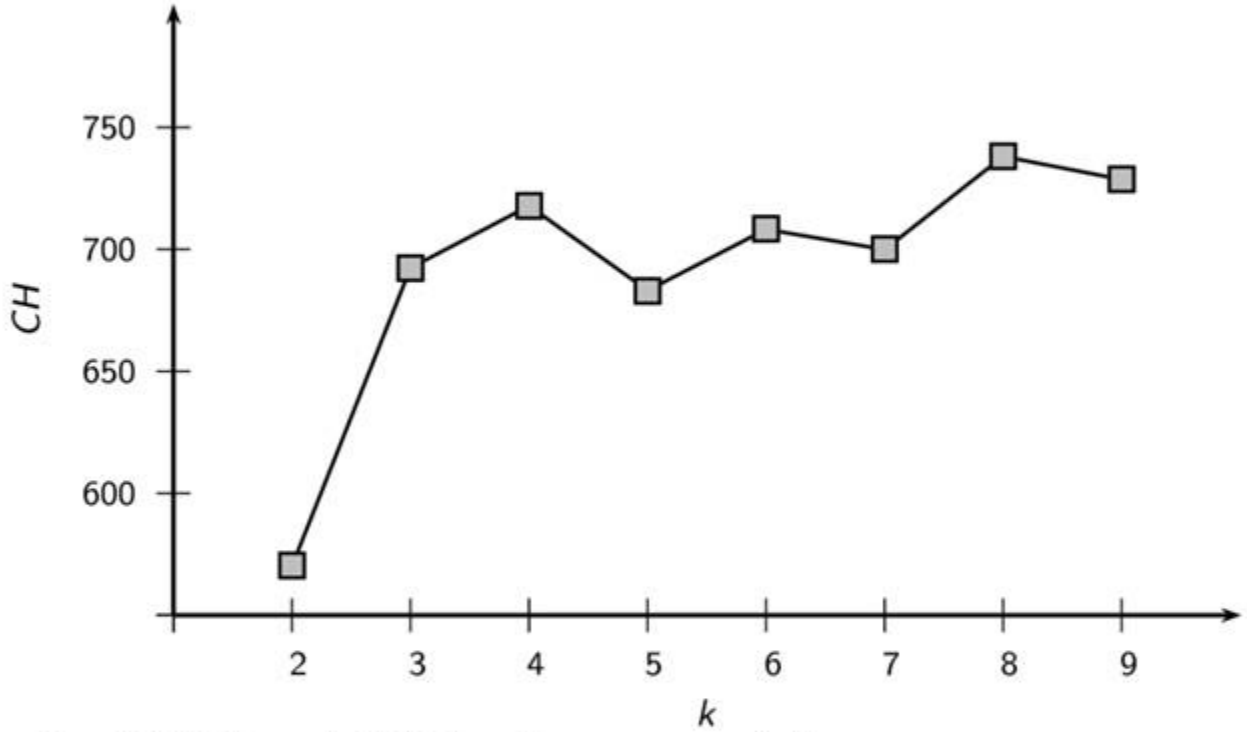
We plot the CH values and look for a large increase in the value followed by little or no gain. We choose the value $k > 3$ that minimizes the term

$$\Delta(k) = \Big(CH(k+1) - CH(k)\Big) - \Big(CH(k) - CH(k-1)\Big)$$

The intuition is that we want to find the value of k for which $CH(k)$ is much higher than $CH(k-1)$ and there is only a little improvement or a decrease in the $CH(k+1)$ value.

Calinski–Harabasz Variance Ratio

CH ratio for various values of k on the Iris principal components data, using the K-means algorithm, with the best results chosen from 200 runs.



The successive $CH(k)$ and $\Delta(k)$ values are as follows:

k	2	3	4	5	6	7	8	9
$CH(k)$	570.25	692.40	717.79	683.14	708.26	700.17	738.05	728.63
$\Delta(k)$	–	−96.78	−60.03	59.78	−33.22	45.97	−47.30	–

$\Delta(k)$ suggests $k = 3$ as the best (lowest) value.

Relative Measures: Gap Statistic

The gap statistic compares the sum of intracluster weights W_{in} for different values of k with their expected values assuming no apparent clustering structure, which forms the null hypothesis.

Let $\mathcal{C}_k$ be the clustering obtained for a specified value of k. Let $W_{in}^k(\boldsymbol{D})$ denote the sum of intracluster weights (over all clusters) for $\mathcal{C}_k$ on the input dataset $\boldsymbol{D}$.

We would like to compute the probability of the observed W_{in}^k value under the null hypothesis. To obtain an empirical distribution for W_{in}, we resort to Monte Carlo simulations of the sampling process.

Relative Measures: Gap Statistic

We generate t random samples comprising n points. Let $\boldsymbol{R}_i \in \mathbb{R}^{n \times d}$, $1 \leq i \leq t$ denote the ith sample. Let $W_{in}^k(\boldsymbol{R}_i)$ denote the sum of intracluster weights for a given clustering of $\boldsymbol{R}_i$ into k clusters.

From each sample dataset $\boldsymbol{R}_i$, we generate clusterings for different values of k, and record the intracluster values $W_{in}^k(\boldsymbol{R}_i)$.

Let $\mu_W(k)$ and $\sigma_W(k)$ denote the mean and standard deviation of these intracluster weights for each value of k. The *gap statistic* for a given k is then defined as

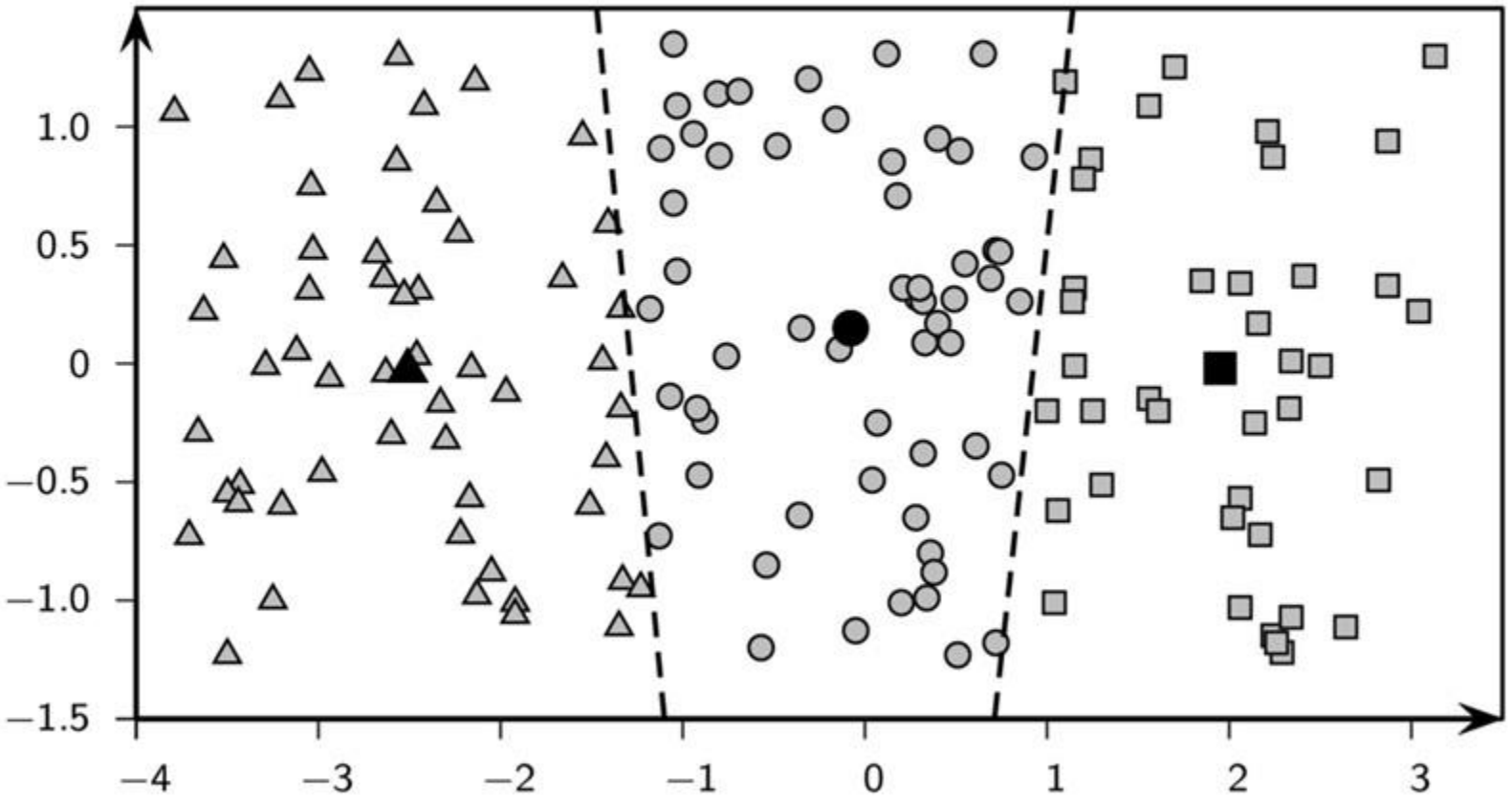
$$gap(k) = \mu_W(k) - \log W_{in}^k(\boldsymbol{D})$$

Choose k as follows:

$$k^* = \arg\min_k \left\{ gap(k) \geq gap(k+1) - \sigma_W(k+1) \right\}$$

Gap Statistic: Randomly Generated Data

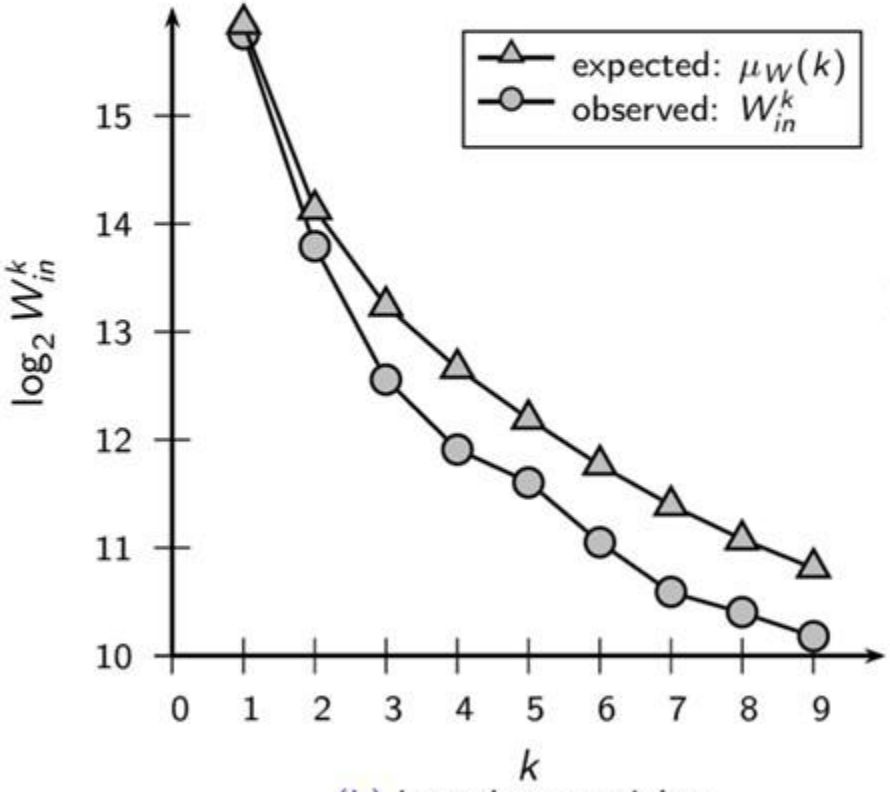
A random sample of $n = 150$ points, which does not have any apparent cluster structure.



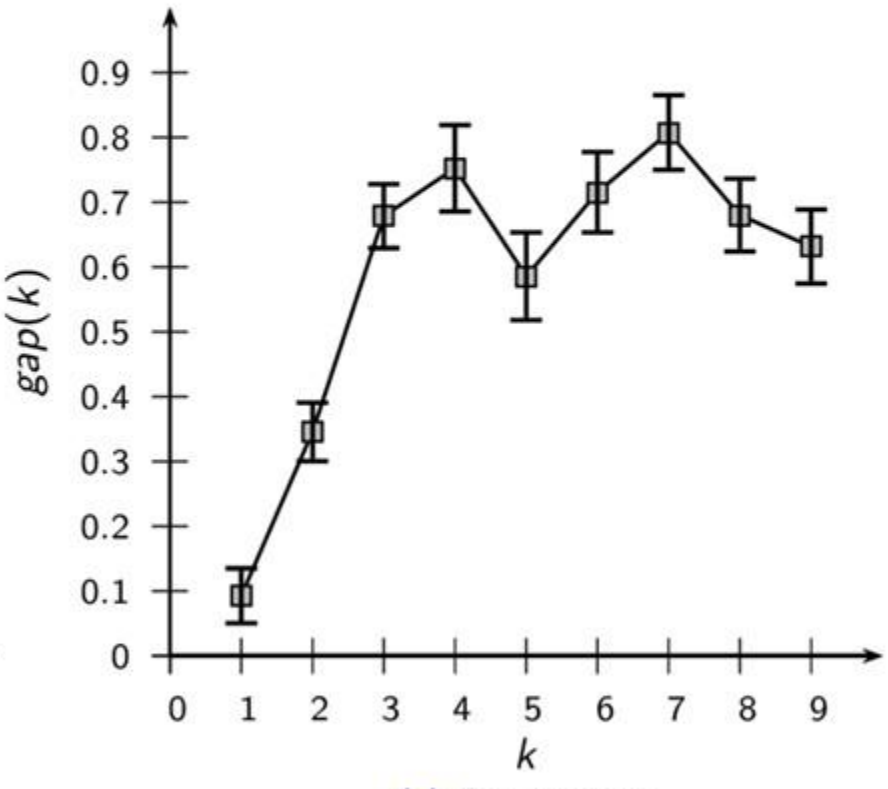
(a) Randomly generated data ($k = 3$)

Gap Statistic: Intracluster Weights and Gap Values

We generate $t = 200$ random datasets, and compute both the expected and the observed (Iris) intracluster weight $\mu_W(k)$, for each value of k. The observed $W_{in}^k(\boldsymbol{D})$ values are smaller than the expected values $\mu_W(k)$.



(b) Intracluster weights



(c) Gap statistic

Gap Statistic as a Function of k

k	$gap(k)$	$\sigma_W(k)$	$gap(k) - \sigma_W(k)$
1	0.093	0.0456	0.047
2	0.346	0.0486	0.297
3	0.679	0.0529	0.626
4	0.753	0.0701	0.682
5	0.586	0.0711	0.515
6	0.715	0.0654	0.650
7	0.808	0.0611	0.746
8	0.680	0.0597	0.620
9	0.632	0.0606	0.571

The optimal value for the number of clusters is $k = 4$ because

$$gap(4) = 0.753 > gap(5) - \sigma_W(5) = 0.515$$

However, if we relax the gap test to be within two standard deviations, then the optimal value is $k = 3$ because

$$gap(3) = 0.679 > gap(4) - 2\sigma_W(4) = 0.753 - 2 \cdot 0.0701 = 0.613$$

Cluster Stability

The main idea behind cluster stability is that the clusterings obtained from several datasets sampled from the same underlying distribution as $\boldsymbol{D}$ should be similar or "stable."

Stability can be used to find a good value for k, the correct number of clusters.

We generate t samples of size n by sampling from $\boldsymbol{D}$ with replacement. Let $\mathcal{C}_k(\boldsymbol{D}_i)$ denote the clustering obtained from sample $\boldsymbol{D}_i$, for a given value of k.

Next, we compare the distance between all pairs of clusterings $\mathcal{C}_k(\boldsymbol{D}_i)$ and $\mathcal{C}_k(\boldsymbol{D}_j)$ using several of the external cluster evaluation measures. From these values we compute the expected pairwise distance for each value of k. Finally, the value k^* that exhibits the least deviation between the clusterings obtained from the resampled datasets is the best choice for k because it exhibits the most stability.

Clustering Stability Algorithm

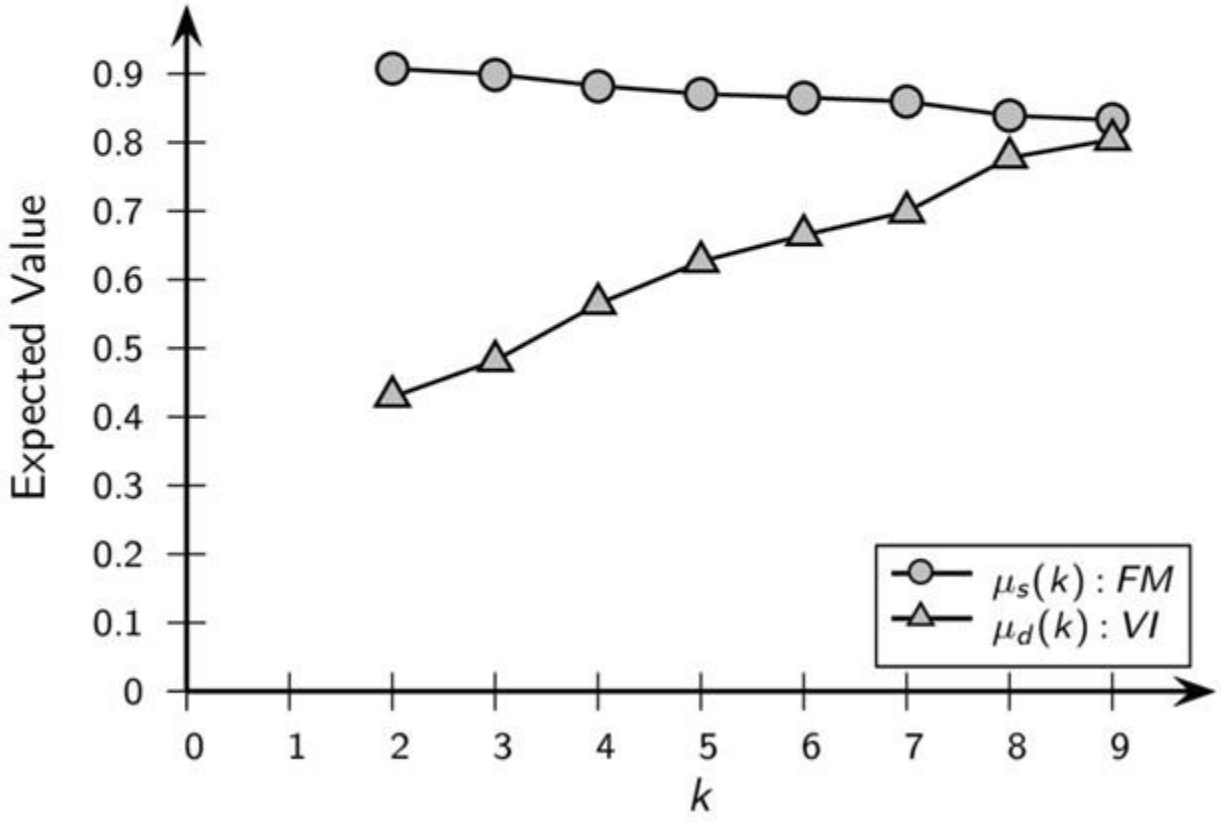
ClusteringStability $(A, t, k^{\max}, \boldsymbol{D})$**:**

$n \leftarrow |\boldsymbol{D}|$
for $i = 1, 2, \ldots, t$ **do**
  $\boldsymbol{D}_i \leftarrow$ sample n points from $\boldsymbol{D}$ with replacement
for $i = 1, 2, \ldots, t$ **do**
  for $k = 2, 3, \ldots, k^{\max}$ **do**
    $\mathcal{C}_k(\boldsymbol{D}_i) \leftarrow$ cluster $\boldsymbol{D}_i$ into k clusters using algorithm A
foreach *pair* $\boldsymbol{D}_i, \boldsymbol{D}_j$ *with* $j > i$ **do**
  $\boldsymbol{D}_{ij} \leftarrow \boldsymbol{D}_i \cap \boldsymbol{D}_j$ `// create common dataset`
  for $k = 2, 3, \ldots, k^{\max}$ **do**
    $d_{ij}(k) \leftarrow d\big(\mathcal{C}_k(\boldsymbol{D}_i), \mathcal{C}_k(\boldsymbol{D}_j), \boldsymbol{D}_{ij}\big)$ `// distance between clusterings`
for $k = 2, 3, \ldots, k^{\max}$ **do**
  $\mu_d(k) \leftarrow \frac{2}{t(t-1)} \sum_{i=1}^{t} \sum_{j>i} d_{ij}(k)$
$k^* \leftarrow \arg\min_k \{\mu_d(k)\}$

Clustering Stability: Iris Data

$t = 500$ bootstrap samples; best K-means from 100 runs

Both the Variation of Information and the Fowlkes-Mallows measures indicate that $k = 2$ is the best value. VI indicates the least expected distance between pairs of clusterings, and FM indicates the most expected similarity between clusterings.



Clustering Tendency: Spatial Histogram

Clustering tendency or clusterability aims to determine whether the dataset $\boldsymbol{D}$ has any meaningful groups to begin with.

Let $X_1, X_2, \ldots, X_d$ denote the d dimensions. Given b, the number of bins for each dimension, we divide each dimension X_j into b equi-width bins, and simply count how many points lie in each of the b^d d-dimensional cells.

From this spatial histogram, we can obtain the empirical joint probability mass function (EPMF) for the dataset $\boldsymbol{D}$

$$f(\boldsymbol{i}) = P(\boldsymbol{x}_j \in \text{cell } \boldsymbol{i}) = \frac{\left|\{\boldsymbol{x}_j \in \text{cell } \boldsymbol{i}\}\right|}{n}$$

where $\boldsymbol{i} = (i_1, i_2, \ldots, i_d)$ denotes a cell index, with i_j denoting the bin index along dimension X_j.

Clustering Tendency: Spatial Histogram

We generate t random samples, each comprising n points within the same d-dimensional space as the input dataset $\boldsymbol{D}$. Let $\boldsymbol{R}_j$ denote the jth such random sample. We then compute the corresponding EPMF $g_j(\boldsymbol{i})$ for each $\boldsymbol{R}_j$, $1 \leq j \leq t$.

We next compute how much the distribution f differs from g_j (for $j = 1, \ldots, t$), using the Kullback–Leibler (KL) divergence from f to g_j, defined as

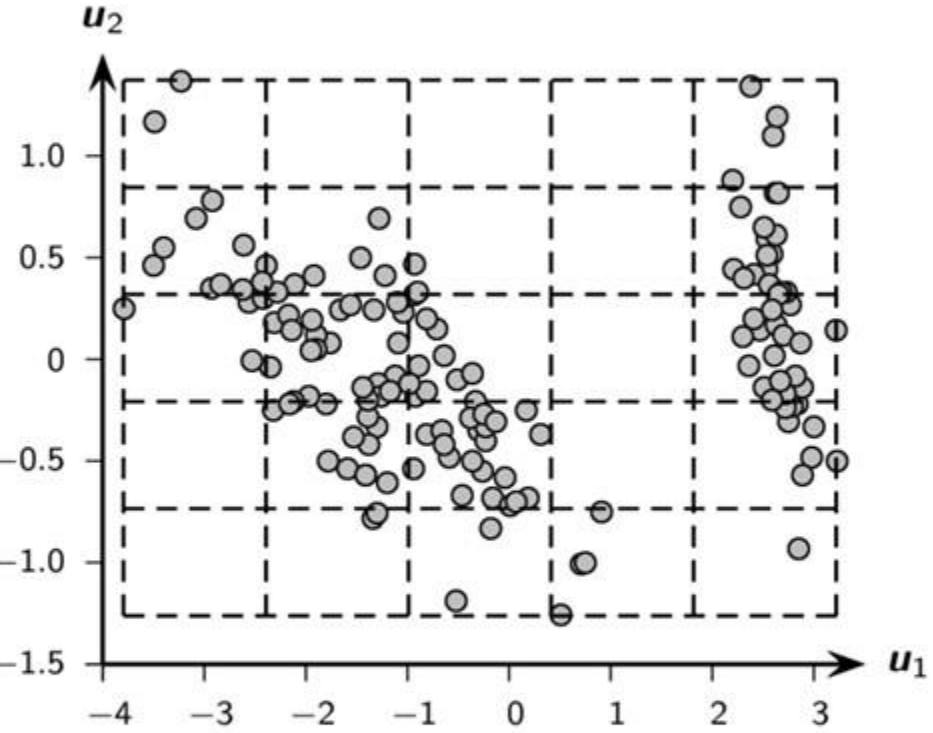
$$KL(f|g_j) = \sum_{\boldsymbol{i}} f(\boldsymbol{i}) \log \left(\frac{f(\boldsymbol{i})}{g_j(\boldsymbol{i})} \right)$$

The KL divergence is zero only when f and g_j are the same distributions. Using these divergence values, we can compute how much the dataset $\boldsymbol{D}$ differs from a random dataset.

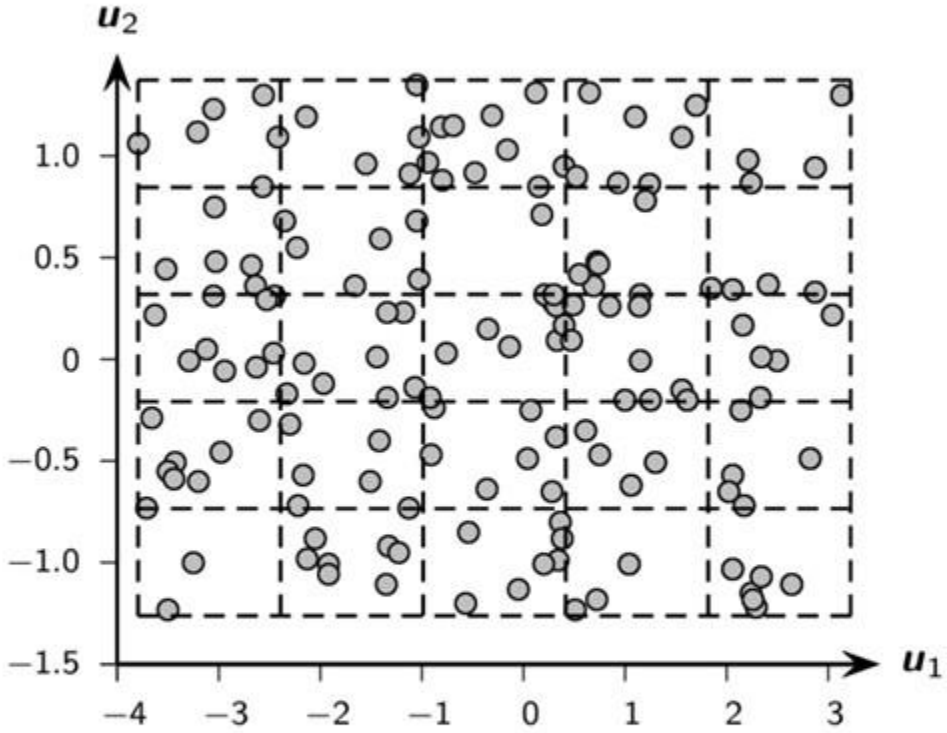
Its main limitation is that the number of cells (b^d) increases exponentially with the dimensionality, and, with a fixed sample size n, most of the cells will have none or one point, making it hard to estimate the divergence. The method is also sensitive to the choice of parameter b.

Spatial Histogram: Iris PCA Data versus Uniform

Uniform has $n = 150$ points



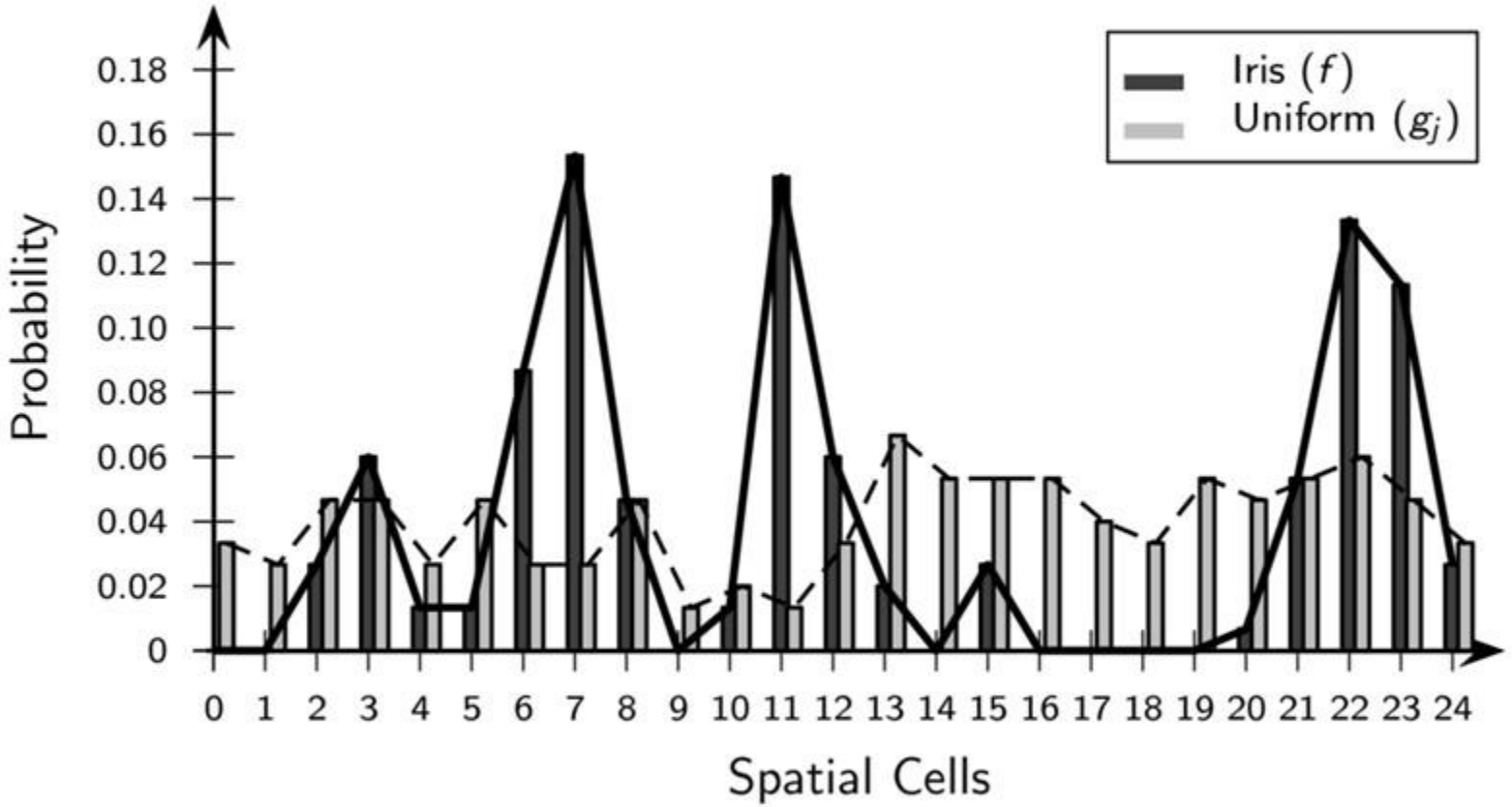
(a) Iris: spatial cells



(b) Uniform: spatial cells

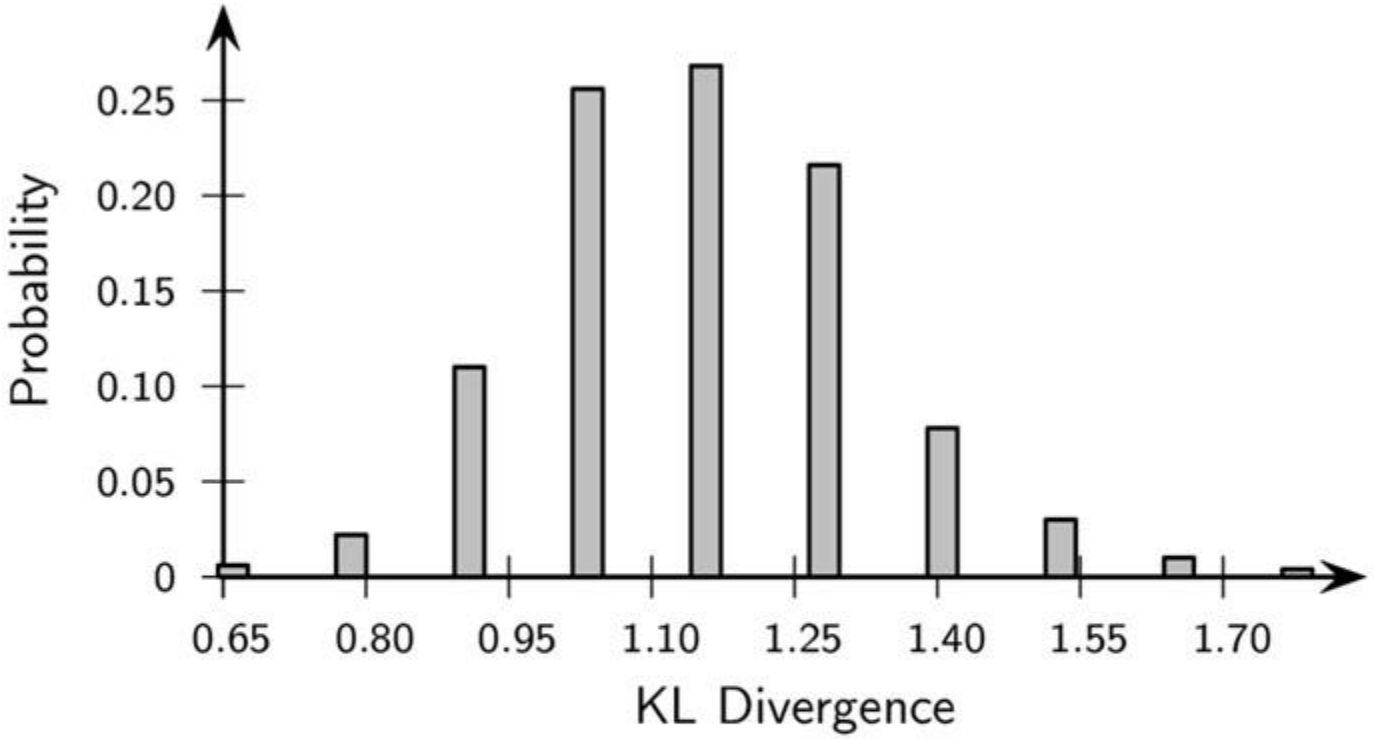
Spatial Histogram: Empirical PMF

5 bins results in 25 spatial cells



(c) Empirical probability mass function

Spatial Histogram: KL Divergence Distribution



(d) KL-divergence distribution

We generated $t = 500$ random samples from the null distribution, and computed the KL divergence from f to g_j for each $1 \leq j \leq t$.

The mean KL value is $\mu_{KL} = 1.17$, with a standard deviation of $\sigma_{KL} = 0.18$, that is, Iris PCA is clusterable.

Clustering Tendency: Distance Distribution

We can compare the pairwise point distances from $\boldsymbol{D}$, with those from the randomly generated samples $\boldsymbol{R}_i$ from the null distribution.

We create the EPMF from the proximity matrix $\boldsymbol{W}$ for $\boldsymbol{D}$ by binning the distances into b bins:

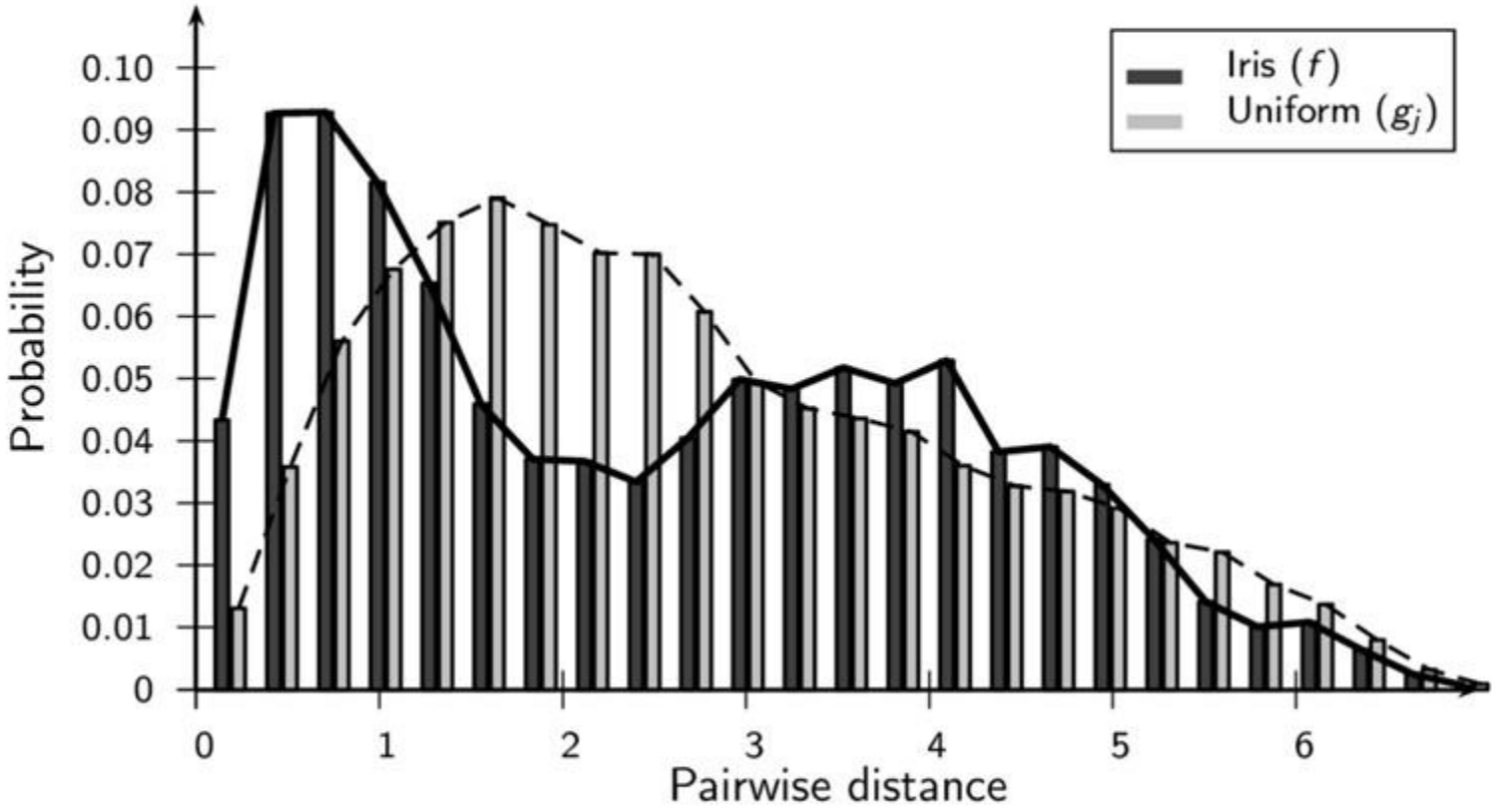
$$f(i) = P(w_{pq} \in \text{ bin } i \mid \boldsymbol{x}_p, \boldsymbol{x}_q \in \boldsymbol{D}, p < q) = \frac{\left|\{w_{pq} \in \text{ bin } i\}\right|}{n(n-1)/2}$$

Likewise, for each of the samples $\boldsymbol{R}_j$, we determine the EPMF for the pairwise distances, denoted g_j.

Finally, we compute the KL divergences between f and g_j. The expected divergence indicates the extent to which $\boldsymbol{D}$ differs from the null (random) distribution.

Iris PCA Data × Uniform: Distance Distribution

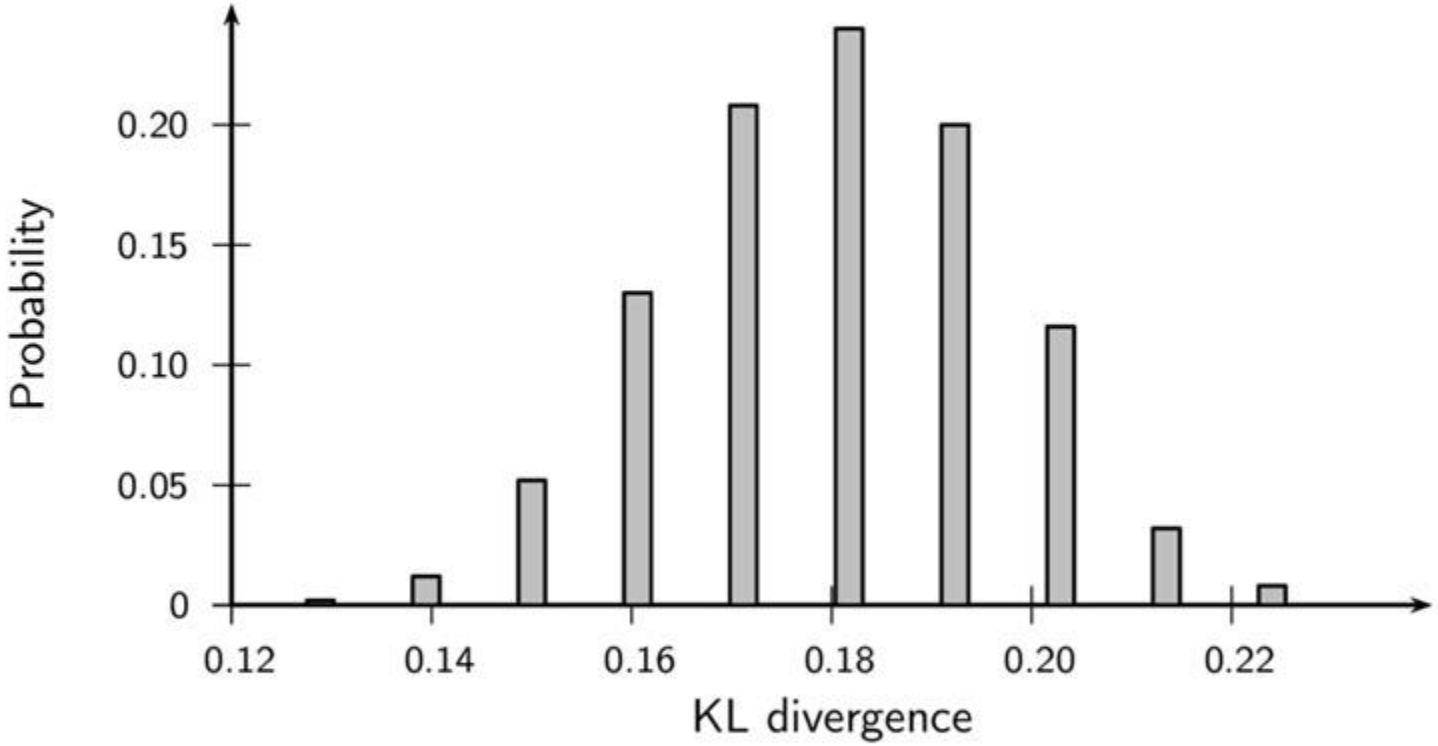
The distance distribution is obtained by binning the edge weights between all pairs of points using $b = 25$ bins.



(a)

Iris PCA Data × Uniform: Distance Distribution

We compute the KL divergence from $\boldsymbol{D}$ to each $\boldsymbol{R}_j$, over $t = 500$ samples. The mean divergence is $\mu_{KL} = 0.18$, with standard deviation $\sigma_{KL} = 0.017$. Even though the Iris dataset has a good clustering tendency, the KL divergence is not very large.



(b)

We conclude that, at least for the Iris dataset, the distance distribution is not as discriminative as the spatial histogram approach for clusterability analysis.

Clustering Tendency: Hopkins Statistic

Given a dataset $\boldsymbol{D}$ comprising n points, we generate t uniform subsamples $\boldsymbol{R}_i$ of m points each, sampled from the same dataspace as $\boldsymbol{D}$.

We also generate t subsamples of m points directly from $\boldsymbol{D}$, using sampling without replacement. Let $\boldsymbol{D}_i$ denote the ith direct subsample.

Next, we compute the minimum distance between each point $\boldsymbol{x}_j \in \boldsymbol{D}_i$ and points in $\boldsymbol{D}$

$$\delta_{\min}(\boldsymbol{x}_j) = \min_{\boldsymbol{x}_i \in \boldsymbol{D}, \boldsymbol{x}_i \neq \boldsymbol{x}_j} \left\{ \|\boldsymbol{x}_j - \boldsymbol{x}_i\| \right\}$$

We also compute the minimum distance $\delta_{\min}(\boldsymbol{y}_j)$ between a point $\boldsymbol{y}_j \in \boldsymbol{R}_i$ and points in $\boldsymbol{D}$.

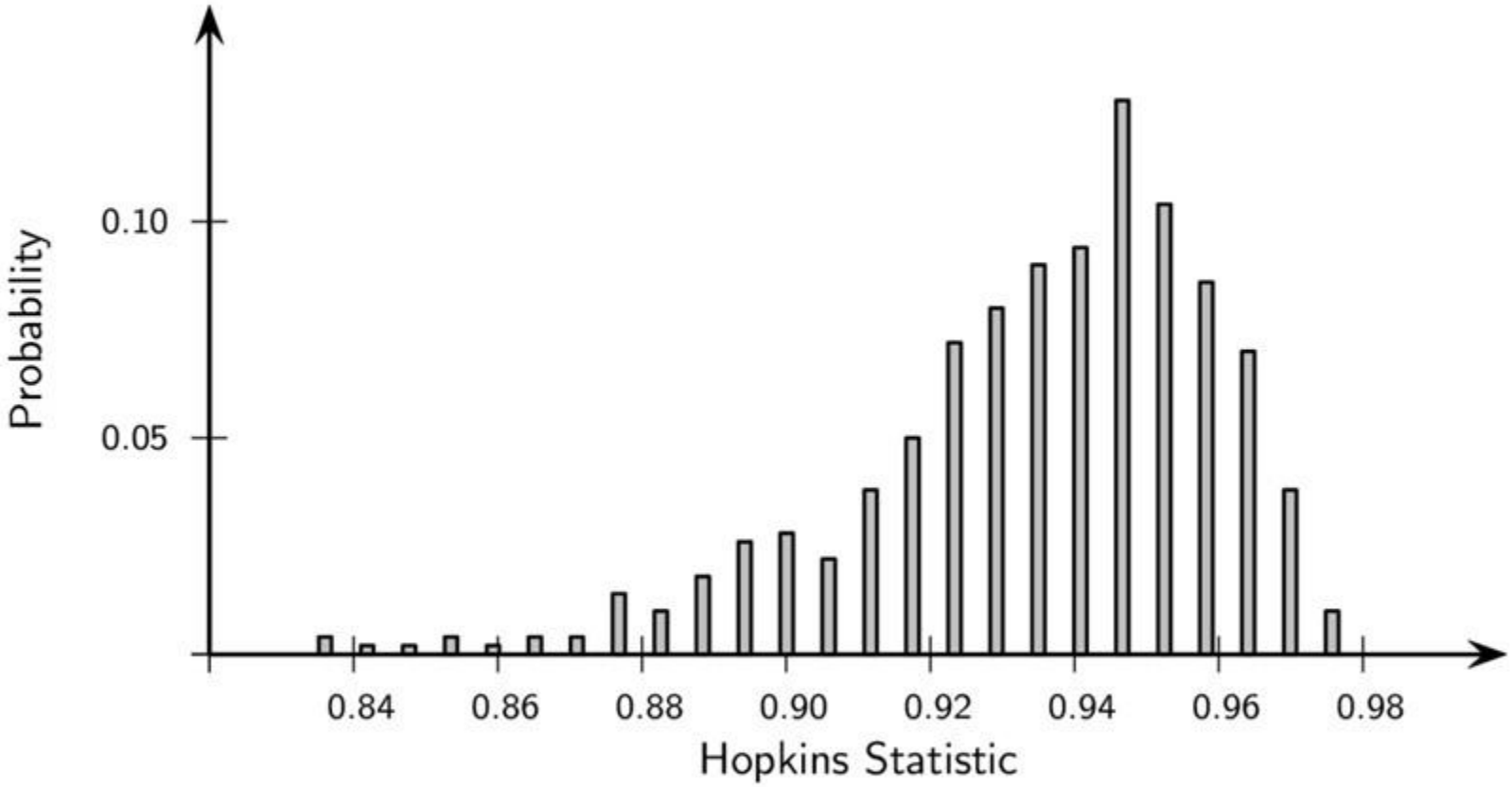
The Hopkins statistic (in d dimensions) for the ith pair of samples $\boldsymbol{R}_i$ and $\boldsymbol{D}_i$ is then defined as

$$HS_i = \frac{\sum_{\boldsymbol{y}_j \in \boldsymbol{R}_i} \left(\delta_{\min}(\boldsymbol{y}_j)\right)^d}{\sum_{\boldsymbol{y}_j \in \boldsymbol{R}_i} \left(\delta_{\min}(\boldsymbol{y}_j)\right)^d + \sum_{\boldsymbol{x}_j \in \boldsymbol{D}_i} \left(\delta_{\min}(\boldsymbol{x}_j)\right)^d}$$

If the data is well clustered we expect $\delta_{\min}(\boldsymbol{x}_j)$ values to be smaller compared to the $\delta_{\min}(\boldsymbol{y}_j)$ values, and in this case HS_i tends to 1.

Iris PCA Data × Uniform: Hopkins Statistic Distribution

Number of sample pairs $t = 500$, subsample size $m = 30$.



The Hopkins statistic has $\mu_{HS} = 0.935$ and $\sigma_{HS} = 0.025$.

Given the high value of the statistic, we conclude that the Iris dataset has a good clustering tendency.

Q&A