✱ Representative-Based Clustering

- For each cluster, there is a point (or parameter vector) that summarizes it (common choice: mean)
- Given a dataset of N instances and desired # of clusters K, this class of algorithms generates a partition $C = \{C_1, \ldots, C_k\}$

$$\mu_i = \frac{1}{n_i} \cdot \sum_{x_j \in C_j} x_j$$

• Brute-Force Approach

- Generate all possible clustering $C = \{C_1, \ldots, C_k\}$ → Then select the best one (But there are $O(k^n/k!)$ partitions)

• K-means Algorithm

- Greedy iterative approach to find a clustering that minimizes the SSE objective. (Then it can converge local optimal instead of global one)

$$SSE(C) = \sum_{i=1}^{k} \sum_{x_j \in C_j} \|x_j - \mu_i\|^2 \xrightarrow{\text{Goal is to find:}} C^* = \arg\min_C SSE(C)$$

Algorithm 13.1: K-means Algorithm

```
K-MEANS (D, k, ε):
t = 0
Randomly initialize k centroids: μ_1^t, μ_2^t, ..., μ_k^t ∈ R^d
repeat
    t ← t + 1
    C_i ← ∅ for all i = 1, ..., k
    // Cluster Assignment Step
    foreach x_j ∈ D do
        i* ← arg min_i { ||x_j − μ_i^{t−1}||^2 }
        C_{i*} ← C_{i*} ∪ {x_j} // Assign x_j to closest centroid
    // Centroid Update Step
    foreach i = 1, ..., k do
        μ_i^t ← (1/|C_i|) Σ_{x_j ∈ C_i} x_j
until Σ_{i=1}^{k} ||μ_i^t − μ_i^{t−1}||^2 ≤ ε
```

→ Algorithm Complexity:
O(tnkd) time
n points, k clusters,
d operations
t iterations

» Centroid Initialization:

Solution 1) Pick points that are as far away from one another as possible

Solution 2) Pick first point random → while there are fewer than k points; add the point whose min. distance from the selected points is as large as possible

Solution 3) Cluster a sample of data (hierarchically?) → k clusters → pick a point from each cluster, perhaps that point closest to the centroid of the cluster

❗ Selecting best initial centroid is difficult:

$$\frac{\text{ways to select one centroid for each cluster}}{\text{ways to select k centroids}} = \frac{k! \cdot n^k}{(kn)^k} = \frac{k!}{k^k}$$

* for 10 clusters: $\frac{10!}{10^{10}} \sim 0.00036$

Algorithm 3 Bisecting K-means Algorithm.

```
1: Initialize the list of clusters to contain the cluster containing all points.
2: repeat
3:     Select a cluster from the list of clusters
4:     for i = 1 to number_of_iterations do
5:         Bisect the selected cluster using basic K-means
6:     end for
7:     Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: until Until the list of clusters contains K clusters
```

→ selected cluster'ı ortadan ikiye ayır.

» Dealing with Initial Centroids Issue:

- Multiple runs helps → but probability is not on your side.
- Sample and use another clustering method (hierarchical?) to determine initial centroids.
- Select more than k initial centroids → then select among these
- Post-processing
- Bisecting K-means → not sensitive to initialization issues.

✱ Centroids are updated after all points are assigned to a centroid (alternative) → Update centroids after each assignment (More expensive, introduces order dependency, never get empty cluster)

» Preprocessing

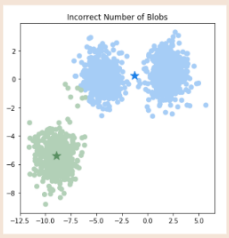
- Normalize the data
- Eliminate outliers

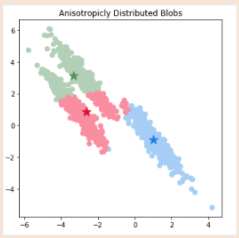
» Post processing

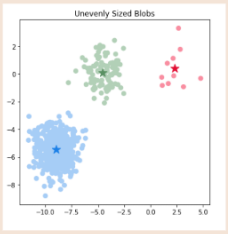
- Eliminate small clusters that may represent outliers
- Split 'loose' clusters (clusters with high SSE)
- Merge clusters that are 'close' and have relatively low SSE

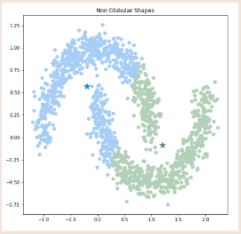
» Limitations of K-means:

- K-means has problems when clusters are differing → non-globular shape, size, densities
- It also has problems with outliers









» K-means Clustering Summary:

Strength
- Efficient
- Often terminates at local optimal
- Global optimal can be found using; deterministic annealing, genetic algorithms

Weakness
- Applicable only when mean defined (categorical data?)
- Need to specify K in advance
- Unable to handle noise, outliers
- Not suitable for non-convex shapes

Advantages
- Simple, understandable
- Items automatically assigned to clusters

Disadvantages
- Must pick K before
- All items forced into a cluster
- Too sensitive to outliers

» Variants of K-means:

- K-means differ in → selection of initial K means; dissimilarity calculations; strategies to calculate cluster mean

- Handling categorical data: (K-modes)
  - Replace means with modes
  - Use new dissimilarity measure to deal with categorical objects
  - Use a frequency-based method to update modes.
  - Mixture of numerical and categorical data: k-prototype method

- How do we choose K?
  - Knee and Elbow analysis → Not enough!

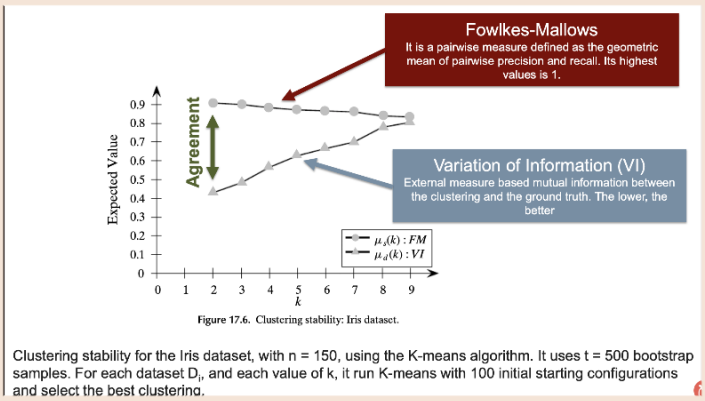
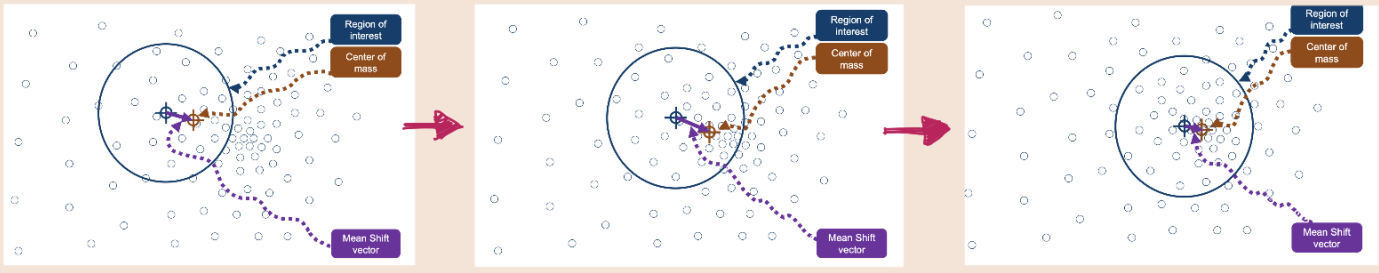


Figure 17.6. Clustering stability: Iris dataset.

Clustering stability for the Iris dataset, with n = 150, using the K-means algorithm. It uses t = 500 bootstrap samples. For each dataset D_i, and each value of k, it run K-means with 100 initial starting configurations and select the best clustering.

Mean-Shift Algorithm

- It searches for the mode of a data distribution (nonparametric, iterative, versatile algorithm)



» Mean-Shift Pseudocode

1) Choose a seach window size
2) For each point:
   a) Center a window on that point
   b) Compute mean in the search window
   c) Center the search window at new mean location
   d) Repeat (b,c) until convergence
3) Assign points that lead to nearby modes to the same cluster.

» K-Means Clustering vs. Mean-Shift Clustering

- Assumes # of clusters already known and clusters shaped spherically
- Sensitive to initialization
- Sensitive to outliers
- Fast and has amplexity $O(knT)$ (k: # of cluster, n: # of points)

- Does not assume anything about # of clusters, instead, # of modes give the # of cluster. Since it based on density estimation, it can handle arbitrarily shaped clusters
- Robust to initializations
- Less-sensitive to outliers
- Computationaly expersive, has time complexity $O(Tn^2)$ (T: # of iterations)

» Mean-Shift Clustering Strength & Weakness

Strength:

- Does not assume prior shape on clusters
- Can handle arbitrary feature spaces
- Only window size to choose
- Window size has a physical meaning

Weakness

- Window size selection is not trivial (Inappropriate window size can cause modes to be merged, or generate additional shallow modes → Adaptive window size can help)
- Not suitable for high-dimensional features

• Expectation - Maximization (EM) Clustering

- We assume that each cluster C_i is characterized by a multivariate normal distribution and thus, identified by the mean vector μ_i and covariance matrix Σ_i
- A clustering identified by a vector of parameter $\Theta = \{\mu_i, \Sigma_i, P(C_i)\}$ ($P(C_i)$: prior prob. of cluster C_i) * $\sum P(C_i) = 1$
- Goal of Maximum Likelihood Estimation (MLE) is choose the parameters Θ that maximize the likelihood

$$\Theta^* = \arg\max_{\Theta} P(D \mid \Theta)$$

- General Idea:
  - Starts with an initial estimate of parameter vector Θ
  - Iteratively rescores the patterns against mixture density produced by the parameter vector Θ
  - The rescored patterns are used to update the parameters
  - Patterns belonging to same cluster if they're placed by their scores in particular component.

>> EM in One Direction

- Consider a dataset consisting of single attribute $X = \{x_1, \ldots, x_n\}$
- Clusters represented using Univariate Normal ; $f(x) = \frac{1}{\sqrt{2\pi}\sigma_i} \cdot e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$
- Each cluster represented by the parameters $\{\mu_i, \sigma_i^2, P(C_i)\}$
- Initialize parameters; μ_i selected uniformly at random, $\sigma_i^2 = 1$, $P(C_i) = 1/k$ → # of clusters
- For each cluster, using current estimate of parameters, we can compute posterior probabilities

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

→ denote; $w_{ij} = P(C_i \mid x_j)$

w_i : weight vector for C_i over all n points $(w_{i1}, \ldots, w_{in})^T$

- Given all posterior values w_{ij}, maximization step computes MLE of cluster parameters $(\mu_i, \sigma_i^2, P(C_i))$
  - Mean updated as: $\mu_i = \frac{\sum_{j=1}^{n} w_{ij} x_j}{\sum_{j=1}^{n} w_{ij}}$
  - $\sigma_i^2 = \frac{\sum_{j=1}^{n} w_{ij} (x_j - \mu_i)^2}{\sum_{j=1}^{n} w_{ij}}$
  - $P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{a=1}^{k} \sum_{j=1}^{n} w_{aj}} = \frac{\sum_{j=1}^{n} w_{ij}}{n}$ ($\sum_{a=1}^{k} w_{aj} = 1$)
- Expectation and Maximization steps are repeated until convergence