k-Means Algorithm

A centroid-based partitioning method

K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

The k-means algorithm

Input:

- k: the number of clusters,
- D: a data set containing n objects.

Output: A set of k clusters.

Method:

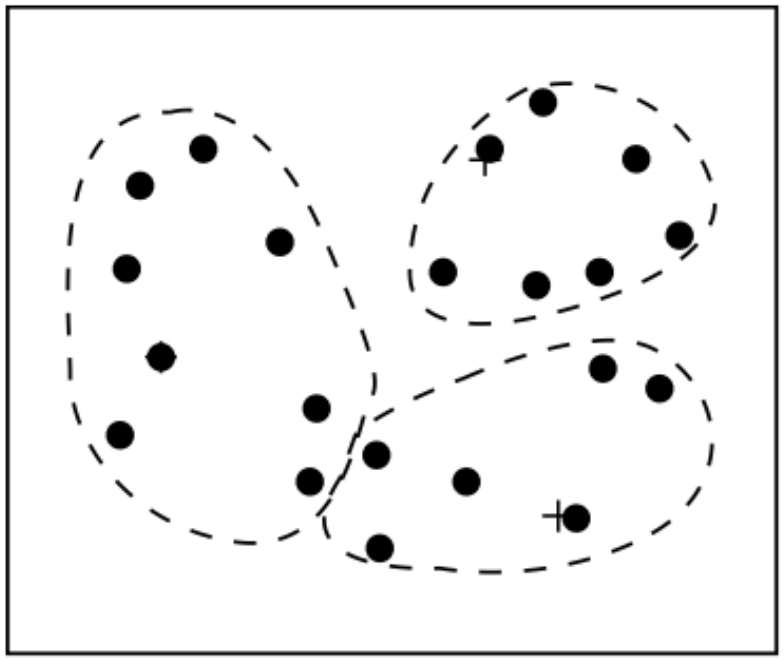
(1) arbitrarily choose k objects from D as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar;
(4) update the cluster centers, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

Often Chosen Randomly

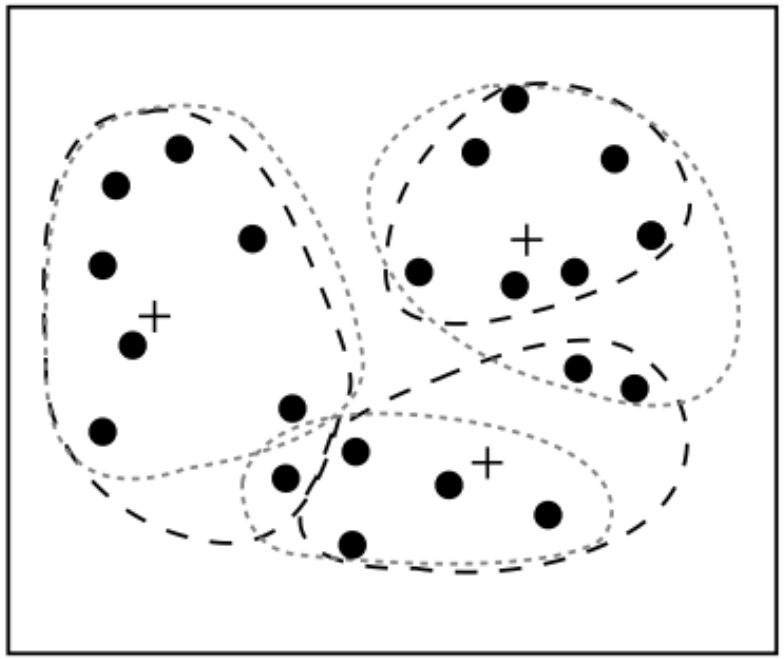
Measured by Euclidean distance, cosine similarity,

Note: Each cluster's center is represented by the mean value of the objects in the cluster.

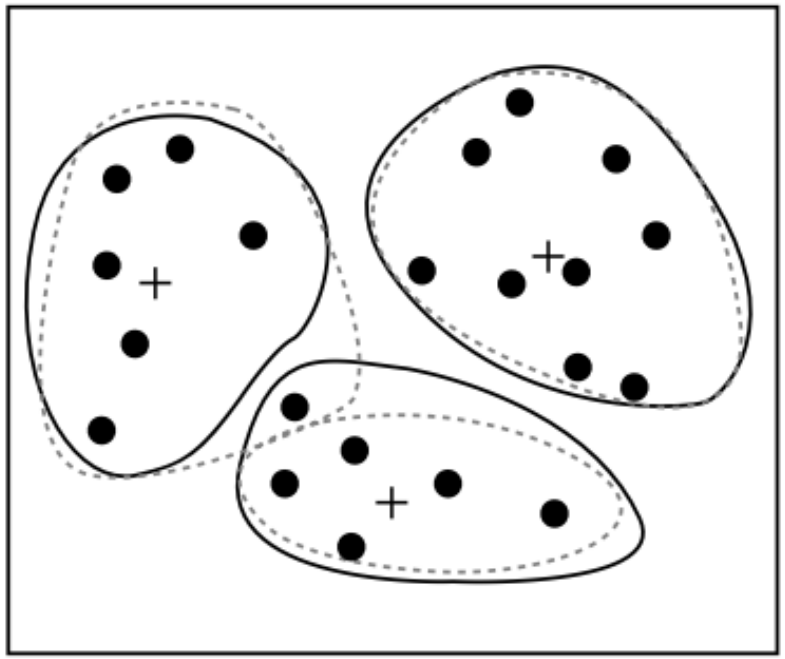
The k-means algorithm



(a) Initial clustering



(b) Iterate

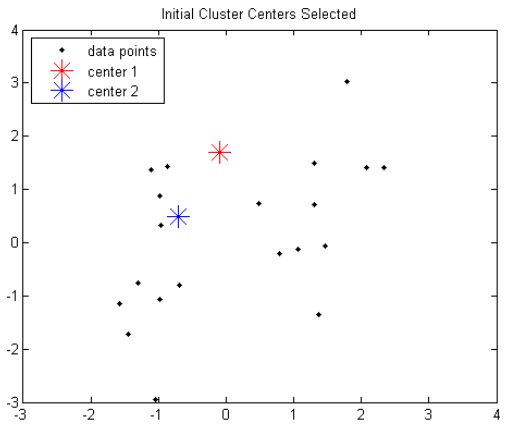


(c) Final clustering

More on k-means algorithm

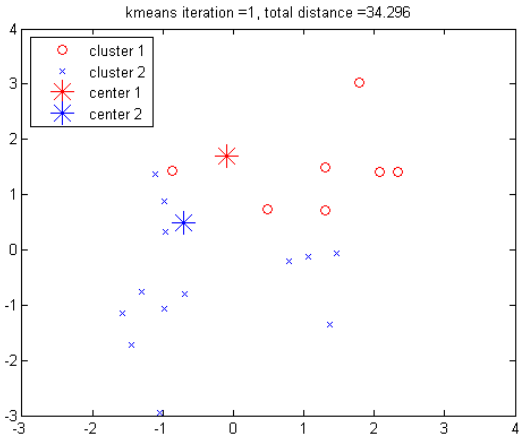
- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is O(n * K * t)
  - n = number of points,
  - K = number of clusters,
  - t = number of iterations

Example: K-Means Clustering

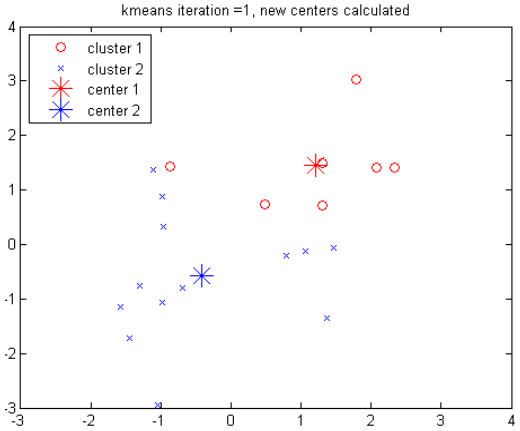


The original data points & randomly select $K = 2$ centroids

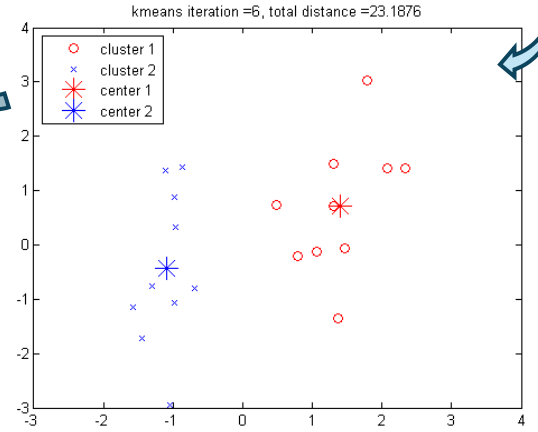
Assign points to clusters



Re-compute cluster centers



Redo point assignment



Iterate

Execution of the K-Means Clustering Algorithm

Select K points as initial centroids

Repeat

- Form K clusters by assigning each point to its closest centroid
- Re-compute the centroids (i.e., *mean point*) of each cluster

Until convergence criterion is satisfied

Pre-processing and Post-processing in K-means

- Pre-processing:
  - Normalize Data: Standardizes data for consistent scaling.
  - Remove Outliers: Helps prevent skewing of cluster centers.
- Post-processing:
  - Eliminate Small Clusters: Small clusters may represent noise.
  - Split Loose Clusters: For clusters with high SSE, split into smaller clusters.
  - Merge Close Clusters: Combine clusters that are close together to reduce SSE.

Challenges with K-Means

- Local vs. Global Optimum
  - K-means may not reach the global optimum.
  - Often terminates at a local optimum.
  - Initial cluster center selection affects results.
- Practical Approach (sometimes useful when less data, small *k*)
  - Run multiple times with different initial centers.
  - Choose clustering with the lowest within-cluster variation.

Efficiency of K-Means

- Time Complexity
- Complexity: $O(nkt)$
  - n : Number of data objects
  - k: Number of clusters
  - t : Number of iterations
- Scalable for large data sets
- Efficient due to typically small k and t values.

Advantages of K-Means

- Simplicity
  - Intuitive and easy to implement.
  - Available in most data mining and machine learning toolkits.
- Scalability
  - Handles large datasets efficiently.
  - Linear runtime with respect to data size, clusters, and iterations.
- Convergence
  - Guaranteed to converge to some local optimum.
  - Avoids very poor results.
- Flexibility
  - Allows for warm-starts (manual initial mean setup based on domain knowledge).
- Adaptable to new data:
  - Can integrate new data after certain iterations.
  - Adapts clustering to new data in next iteration.

Limitations of K-Means

- User-Specified Cluster Count
  - Requires manual setting of the number of clusters.
  - Difficult if user is unfamiliar with the dataset.
- Sensitivity to Initial Means
  - Choice of initial means affects clustering quality.
  - Running multiple times helps only with small cluster counts.
- Challenges with Diverse Cluster Sizes and Density
  - Struggles to identify clusters with different sizes or densities.
  - Sensitive to outliers, which can skew cluster centers.
- High Dimensionality Issues
  - Uses Euclidean distance; becomes dominated by noise in high dimensions.
  - Performance degrades as dimensionality increases.

Evaluating K-means Clustering with Sum of Squared Error (SSE)

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(x, c_i)$$

  - x is a data point in cluster C_i and c_i is the representative point (e.g., center) of cluster C_i

Evaluating K-means Clustering with Sum of Squared Error (SSE)

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them
  - Using Euclidean Distance

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} (\|x - c_i\|^2)$$

  - x is a data point in cluster C_i and c_i is the representative point (e.g., center) of cluster C_i
  - Note the uppercase C is the Cluster and the lowercase c is the centroid

Evaluating K-means Clustering with Sum of Squared Error (SSE)

- Given two clusters, we can choose the one with the smaller error
- However, one easy way to reduce SSE is to increase K, the number of clusters
  - A good clustering with smaller K can have a larger SSE than a poor clustering with larger K
  - Think of the extreme case when K = number of data points