# Clustering
# Partitioning Methods

# Major Clustering Approaches

- **Partitioning approach:**
  - Construct k partitions (k <= n) and then evaluate them by some criterion, e.g., minimizing the sum of square errors
    - Each group has at least one **object, each object** belongs to one group
    - Iterative Relocation Technique
    - Avoid Enumeration by storing the centroids
  - Typical methods: k-means, k-medoids, CLARANS
- **Hierarchical approach:**
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
    - Agglomerative Vs Divisive
    - Rigid – Cannot undo
      - Perform Analysis of linkages
      - Integrate with iterative relocation
  - Typical methods: Diana, Agnes, BIRCH

# Major Clustering Approaches

- Density Based Methods
  - Distance based methods – Spherical Clusters
  - Density – For each data point within a given cluster the neighbourhood should contain a minimum number of points
  - DBSCAN, OPTICS
- Grid Based Methods
  - Object space – finite number of cells forming grid structure
  - Fast processing time
  - Typical methods: STING, WaveCluster, CLIQUE

# Major Clustering Approaches

- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: pCluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD, constrained clustering

# Partitioning Algorithms: Basic Concept

- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters, s.t., min sum of squared distance

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in Ci} (p - m_i)^2$$

- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) Each cluster is represented by one of the objects in the cluster

# K-Means Clustering Method

- Given *k*, the *k-means* algorithm is implemented in 4 steps:
  - Partition objects into *k* non-empty subsets
  - Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
  - Assign each object to the cluster with the nearest seed point.
  - Go back to Step 2, stop when no more new assignment.

# K-Means Clustering Method

- *k-means* algorithm is implemented as below:
- Input: Number of clusters – k, database of n objects
- Output: Set of k clusters that minimize the squared error
  - Choose k objects as the initial cluster centers
  - Repeat
    - (Re)assign each object to the cluster to which the object is most similar based on the mean value of the objects in the cluster
    - Update the cluster means

    Until no change

# K-Means Clustering Method

**Algorithm:** ***k*-means.** The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

**Input:**

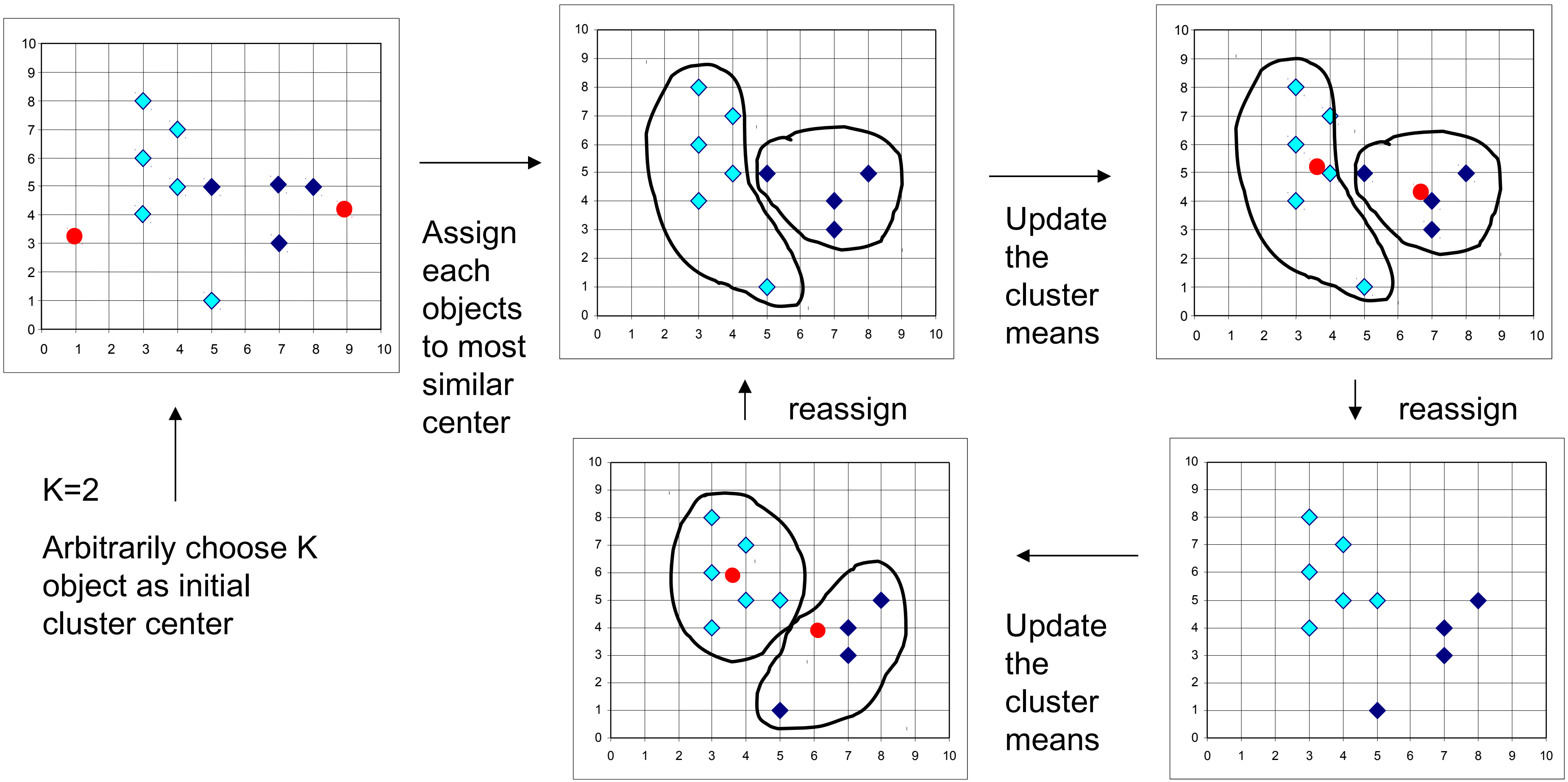
- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

**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, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) **until** no change;

# K-Means Clustering Method



# K-Means Method

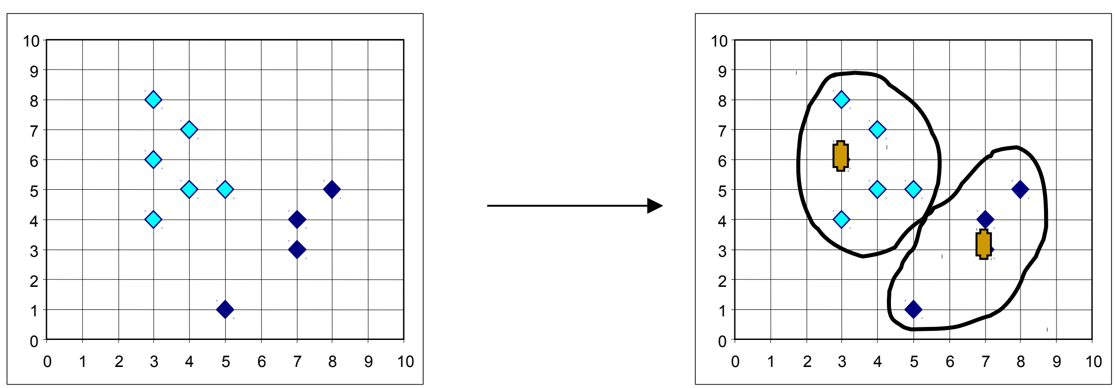
- Strength: *Relatively efficient*: $O(tkn)$, where $n$ is # objects, $k$ is # clusters, and $t$ is # iterations. Normally, $k$, $t \ll n$.
- Comment: Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Weakness
  - Applicable only when *mean* is defined – Categorical data
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

# Variations of the K-Means Method

- A few variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - A mixture of categorical and numerical data: *k-prototype* method
- Expectation Maximization
  - Assigns objects to clusters based on the probability of membership
- Scalability of k-means
  - Compressible, Discardable, To be maintained in main memory
  - Clustering Features

# Problem of the K-Means Method

- The k-means algorithm is sensitive to outliers
  - Since an object with an extremely large value may substantially distort the distribution of the data.
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster.



# K-Medoids Clustering Method

- *PAM* (Partitioning Around Medoids)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - All pairs are analyzed for replacement
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA*
- *CLARANS*

# K-Medoids

- Input: k, and database of n objects
- Output: A set of k clusters
- Method:
  - Arbitrarily choose k objects as initial medoids
  - Repeat
    - Assign each remaining object to cluster with nearest medoid
    - Randomly select a non-medoid $o_{random}$
    - Compute cost S of swapping $o_j$ with $o_{random}$
    - If S < 0 swap to form new set of k medoids

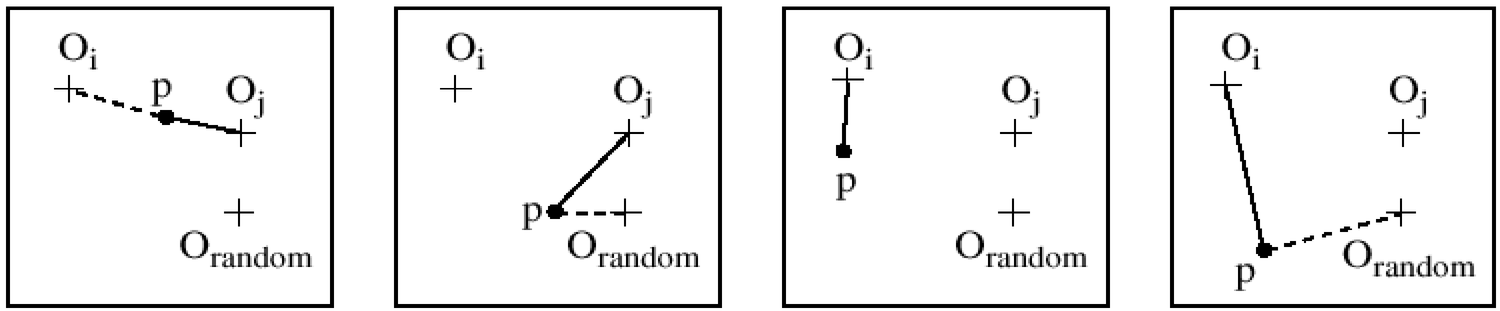
  Until no change

Working Principle: Minimize sum of the dissimilarities between each object and its corresponding reference point. That is, an absolute-error criterion is used

$$E = \Sigma_{j=1}^{k} \Sigma_{p \in Cj} | p - o_j |$$

# K-medoids

- **Case 1:** p currently belongs to medoid $o_j$. If $o_j$ is replaced by $o_{random}$ as a medoid and p is closest to one of $o_i$ where i < > j then **p is reassigned to $o_i$.**
- **Case 2:** p currently belongs to medoid $o_j$. If $o_j$ is replaced by $o_{random}$ as a medoid and p is closest to $o_{random}$ then **p is reassigned to $o_{random}$**.




1. Reassigned to $O_i$
2. Reassigned to $O_{random}$
3. No change
4. Reassigned to $O_{random}$

- **Case 3:** p currently belongs to medoid $o_i$(i< >j) If $o_j$ is replaced by $o_{random}$ as a medoid and p is still closest to **$o_i$ assignment does not change**
- **Case 4:** p currently belongs to medoid $o_i$ (i < > j). If $o_j$ is replaced by $o_{random}$ as a medoid and p is closest to $o_{random}$ then **p is reassigned to $o_{random}$**.

# K-medoids

- After reassignment difference in squared error E is calculated. Total cost of swapping – Sum of costs incurred by all non-medoid objects
- If total cost is negative, $o_j$ is replaced with $o_{random}$ as E will be reduced

# K-medoids Algorithm

**Algorithm:** ***k*-medoids.** PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

**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 in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, $\boldsymbol{o}_{\text{random}}$;
(5) compute the total cost, $S$, of swapping representative object, $\boldsymbol{o}_j$, with $\boldsymbol{o}_{\text{random}}$;
(6) **if** $S < 0$ **then** swap $\boldsymbol{o}_j$ with $\boldsymbol{o}_{\text{random}}$ to form the new set of *k* representative objects;
(7) **until** no change;

# Problem with PAM

- PAM is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- PAM works efficiently for small data sets but does not **scale well** for large data sets.

# CLARA

- Clustering LARge Applications
- Choose a representative set of data
- Choose medoids from this
- Cluster
- Draw multiple such samples and apply PAM on each
- Returns best Clustering
- Effectiveness depends on Sample Size

# CLARANS

- Clustering Large Applications based on RANdomized Search
- Uses Sampling and PAM
- Doesn't restrict itself to any particular sample
- Performs a graph search with each node acting as a potential solution-( k medoids)
- Clustering got after replacement – Neighbor
- Number of neighbors to be tried is limited
- Moves to better neighbour
- Silhouette Coefficient
- Complexity – $O(n^2)$