So Far…

- We have discussed
  - Supervised learning
    - Goal: learn a mapping from inputs $\boldsymbol{x}$ to outputs y
    - Training data: a labeled set of input-output pairs
  - Various methods to learn this mapping functions

- It's time for
  - Unsupervised learning
    - We are only given inputs
    - Goal: find "interesting patterns"

    - Discovering clusters: Clustering

Clustering

Deng Cai (蔡登)

College of Computer Science
Zhejiang University

dengcai@gmail.com

What is Clustering (Cluster Analysis)?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters

- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

Measure the Quality of Clustering

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, typically metric: *d(i, j)*
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

Distance Measures for Different Kinds of Data

- Numerical (interval)-based:
  - Minkowski Distance:
  - Special cases: Euclidean (L2-norm), Manhattan (L1-norm)
- Vectors: cosine measure
- Binary variables:
  - symmetric vs. asymmetric (Jaccard coeff.)
- Nominal variables: # of mismatches
- Ordinal variables: treated like interval-based
- Ratio-scaled variables: apply log-transformation first
- Mixed variables: weighted combinations

- More important:
  - Distance Metric

Aspects in Clustering Methods

- Partitioning requirement: one level versus hierarchical partitioning
- Separation of clusters: exclusive versus non-exclusive
- Similarity measure: distance versus connectivity based on density or contiguity
- Clustering space: full space versus subspaces

Algorithms

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: **k-means**, **k-medoids**
- 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: **GMM**
- Dimensionality reduction approach
  - First dimensionality reduction, then clustering
  - Typical methods: **Spectral clustering**, Ncut

Partitioning Algorithms: Basic Concept

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

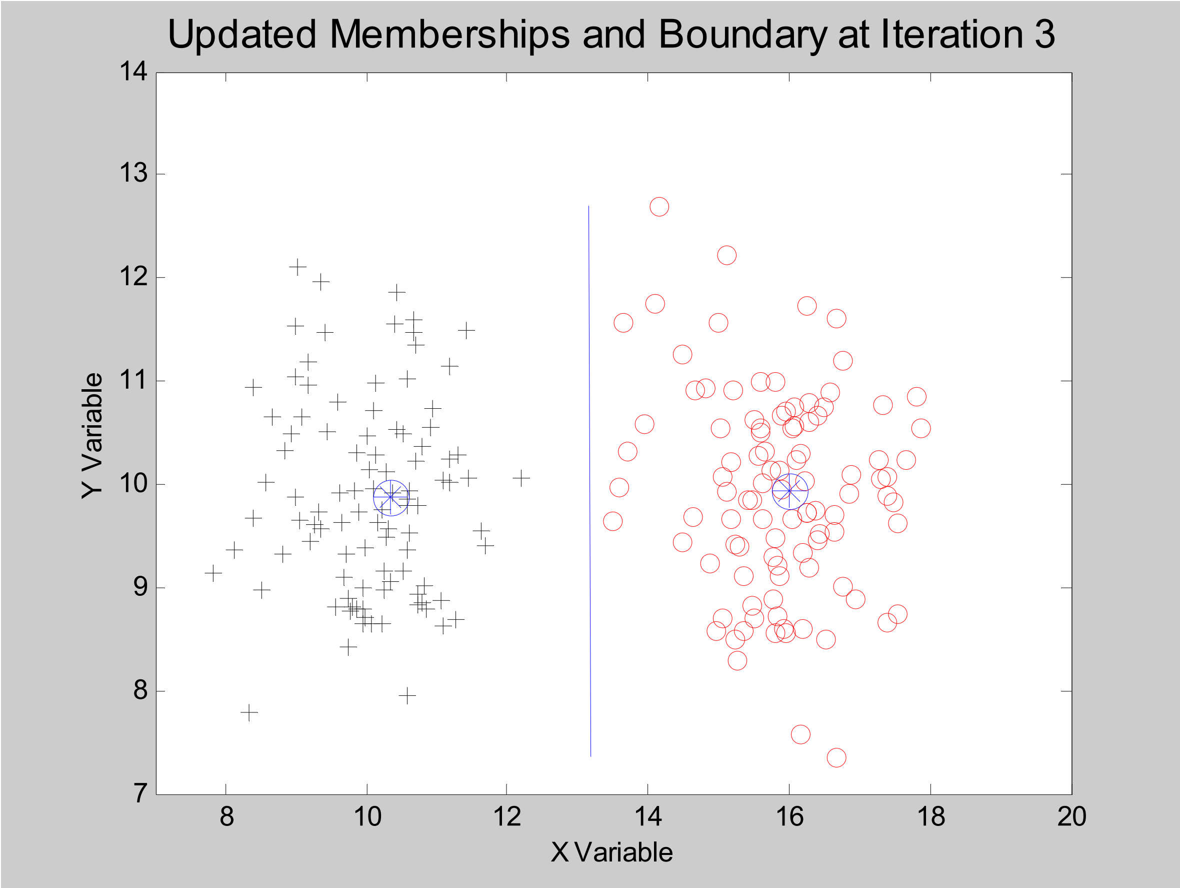
$$\arg\min_{S} \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in c_i} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2$$

- Given 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 (MacQueen'67): Each cluster is represented by the center of the cluster
  - k-medoids or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The K-Means Clustering Method

- Given k, the k-means algorithm is implemented in four steps:
  1. Partition objects into k nonempty subsets
  2. Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., mean point, of the cluster)
  3. Assign each object to the cluster with the nearest seed point
  4. Go back to Step 2, stop when the assignment does not change

K-Means Example



K-Means: Time Complexity

- $O(kndt)$ where t is the iteration upper bond
  - Computing distance between two points: $O(d)$
  - Assignment step: $O(kn)$ distance computations, totaled $O(knd)$
  - Update step: each vector gets added once to corresponding centroid, $O(nd)$
  - Multiply the iteration upper bond t: $O(kndt)$

K-Means: Local Optimum

- TSD (Totoal Squared Distance) decreases at each iteration
  - Global minimum of TSE?
  - No, not necessarily.
  - in a sense it is doing "steepest descent" from a random initial starting point, thus, results will be sensitive to the starting point
  - in practice, we can run it from multiple starting points and pick the solution with the lowest TSD

K-Means: Comments

- Implicit assumptions about the "shapes" of clusters
  - Spherical in vector space
  - Sensitive to coordinate changes, weighting
  - Solution: spectral clustering
- Have to manually pick the number of clusters
  - Try and error? Unfortunately not feasible
  - Solution: Hierarchical clustering
- All items forced into a cluster – Hard clustering
  - Small shift of a data point can flip it to a different cluster
  - Solution: soft probabilistic assignments (GMM)
- Doesn't have a notion of "outliers"
  - Other objective functions
  - Solution: K-Medoids

K-Medoids Clustering Method

- 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.
    - Medoid: a chosen, centrally located object in the cluster
    - Centroid: the "middle" of a cluster

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \boldsymbol{x}_i^{(k)}$$

    - not necessarily inside a cluster

How to find the medoid

- Method 1:
  - Compute the pairwise distances matrix D
  - Compute the row (or, column) sum of D, d
  - Finds the smallest entry in d

- Method 2:
  - Compute the centroid μ
  - Finds the point which is closest to μ

How to find the medoid

- For cluster c_i, min sum of squared distance

$$\arg\min_{x_k \in c_i} \sum_{x_j \in c_i} ||x_j - x_k||^2$$

- Let $\mu_i = \frac{1}{n_i} \sum_{x_j \in c_i} x_j$,

$$\sum_{x_j \in c_i} ||x_j - x_k||^2 = \sum_{x_j \in c_i} ||x_j - \mu_i + \mu_i - x_k||^2$$

$$= \sum_{x_j \in c_i} ||x_j - \mu_i||^2 + n_i ||\mu_i - x_k||^2$$

- Choose x_k be the point nearest to μ_i in cluster c_i

K-Medoids Algorithm

- Given k, the k-medoids algorithm is implemented in five steps:
  1. Partition objects into k nonempty subsets
  2. Compute the centroids of the clusters of the current partitioning
  3. Choose the nearest points of the centroids of the clusters as seed points
  4. Assign each object to the cluster with the nearest seed point
  5. Go back to Step 2, stop when the assignment does not change

K-Medoids: Time Complexity

- $O(kndt)$ the same as k-means
  - Computing distance between two points: $O(d)$
  - Assignment step: $O(kn)$ distance computations, totaled $O(knd)$
  - Computing centroid step: each vector gets added once to corresponding centroid, $O(nd)$
  - Update step: each vector gets added once to corresponding seed points, $O(nd)$
  - Multiply the iteration upper bond t: $O(kndt)$

K-Medoids Clustering Method

- One advantage over K-means
  - Sometimes only the distance matrix are provided and the value $||\mu_i - x_k||^2$ couldn't be calculated directly
  - In the case k-means fails, but k-medoids still works after slightly changing its algorithm

K-Medoids Algorithm

- Partitioning Around Medoids (PAM) algorithm

0. Calculate the pair-wise distance matrix W
1. Initialize: randomly select *k* of the *n* data points as the medoids
2. Associate each data point to the closest medoid
3. For each cluster, compute its medoid
4. Repeat 2-3 until there is no change in the medoids

K-Medoids: Time Complexity

- $O(n^2dt)$
  - Calculate the pair-wise distance: $O\left(n^2d\right)$
  - Assignment step: $O(knd)$ to pick the closest medoid
  - Update medoid step: $O(n)$
  - Multiply the iteration upper bond t: $O(n^2dt)$