Clustering:
Overview and
K-means algorithm

Informal goal

- Given set of objects and measure of similarity between them, group similar objects together
- What mean by “similar”? 
- What is good grouping?
- Computation time / quality tradeoff

General types of clustering

- “Soft” versus “hard” clustering
  - Hard: partition the objects
    - each object in exactly one partition
  - Soft: assign degree to which object in cluster
    - view as probability or score
- flat versus hierarchical clustering
  - hierarchical = clusters within clusters

Applications:

- Many
  - biology
  - astronomy
  - computer aided design of circuits
  - information organization
  - marketing
  - …

Clustering in information search and analysis

- Group information objects
  - ⇒ discover topics
  - other groupings desirable
- Clustering versus classifying
  - classifying: have pre-determined classes with example members
  - clustering:
    - get groups of similar objects
    - added problem of labeling clusters by topic
      - e.g. common terms within cluster of docs.

Example applications in search

- Query evaluation: cluster pruning (§7.1.6)
  - cluster all documents
  - choose representative for each cluster
  - evaluate query w.r.t. cluster reps.
  - evaluate query for docs in cluster(s) having most similar cluster rep.(s)
- Results presentation: labeled clusters
  - cluster only query results
  - e.g. Clusty.com (metasearch)

hard / soft? flat / hier?
Issues

- What attributes represent items for clustering purposes?
- What is measure of similarity between items?
  - General objects and matrix of pairwise similarities
  - Objects with specific properties that allow other specifications of measure
    - Most common:
      - Euclidean distance
      - Cosine similarity
- What is measure of similarity between clusters?

Issues continued

- Cluster goals?
  - Number of clusters?
  - Flat or hierarchical clustering?
  - Cohesiveness of clusters?
- How evaluate cluster results?
  - Relates to measure of closeness between clusters
- Efficiency of clustering algorithms
  - Large data sets => external storage
- Maintain clusters in dynamic setting?
- Clustering methods? - MANY!

General types of clustering methods

- **agglomerative** versus **divisive** algorithms
  - **agglomerative** = bottom-up
    - Build up clusters from single objects
  - **divisive** = top-down
    - Break up cluster containing all objects into smaller clusters
    - Both agglomerative and divisive give hierarchies
    - Hierarchy can be trivial:

      1. ( . . . )
      2. ( . . . )
      3. ( . . . )
      4. ( . . . )

General types of clustering methods cont.

- **constructive** versus **iterative improvement**
  - **constructive**: decide in what cluster each object belongs and don’t change
    - Often faster
  - **iterative improvement**: start with a clustering and move objects around to see if can improve clustering
    - Often slower but better

Quality of clustering

- In applications quality of clustering depends on how well solves problem at hand
- Algorithm uses measure of quality that can be optimized, but that may or may not do a good job of capturing application needs.
- Underlying graph-theoretic problems usually NP-complete
  - E.g. graph partitioning
- Usually algorithm not finding optimal clustering

Distance between clusters

Possible definitions:

I. Distance between closest pair of objects with one in each cluster
   - Called single link
     - . . . .
   - .

II. Distance between furthest pair of objects, one from each cluster
    - Called complete linkage
      - . . . .
      - .

- Measure of similarity between clusters?
Distance between clusters, cont.

Possible definitions:

III. average of pairwise distance between all pairs of objects, one from each
   - more computation

  - Generally no representative point for a cluster;
  - If Euclidean distance
    - centroid
    - bounding box

Vector model:
K-means algorithm

- Well known, well used
- Flat clustering
- Number of clusters picked ahead of time
- Iterative improvement
- Uses notion of centroid
- Typically uses Euclidean distance

K-means overview

- Choose k points among set to cluster
  - Call them k centroids
- For each point not selected, assign it to its closest centroid
  - All assignment give initial clustering
- Until “happy” do:
  - Recompute centroids of clusters
    - New centroids may not be points of original set
  - Reassign all points to closest centroid
  - Updates clusters

An Example
start: choose centroids and cluster

An Example
recompute centroids

An Example
re-cluster around new centroids
An Example
2nd recompute centroids and re-cluster

An Example
3rd (final) recompute and re-cluster

Details for K-means
• Need definition of centroid
  \[ c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \] for i-th cluster \( C_i \) containing objects \( x \)
  notion of \( \text{sum of objects} \) ?
• Need definition of distance to (similarity to) centroid
• Typically vector model with Euclidean distance
• minimizing sum of squared distances of each point to its centroid = Residual Sum of Squares
  \[ \text{RSS} = \sum_{i=1}^{K} \sum_{x \in C_i} \text{dist}(c_i, x)^2 \]

K-means performance
• Can prove RSS decreases with each iteration, so converge
• Can achieve local optimum
  – No change in centroids
• Running time depends on how demanding stopping criteria
• Works well in practice
  – speed
  – quality

Time Complexity of K-means
• Let \( t_{\text{dist}} \) be the time to calculate the distance between two objects
• Each iteration time complexity:
  \[ O(Kn^2 t_{\text{dist}}) \]
  \( n \) = number of objects
• Bound number of iterations \( I \) giving
  \[ O(I^2 K^2 n^2 t_{\text{dist}}) \]
• for m-dimensional vectors:
  \[ O(I^2 K^2 n^m) \]
  \( m \) large and centroids not sparse

Space Complexity of K-means
• Store points and centroids
  – vector model: \( O(n + Km) \)
• External algorithm versus internal?
  – store k centroids in memory
  – run through points each iteration
Choosing Initial Centroids

Bad initialization leads to poor results

Optimal  Not Optimal

Choosing Initial Centroids

Many people spent much time examining how to choose seeds

- Random
  - Fast and easy, but often poor results
- Run random multiple times, take best
  - Slower, and still no guarantee of results
- Pre-conditioning
  - remove outliers
- Choose seeds algorithmically
  - run hierarchical clustering on sample points and use resulting centroids
  - Works well on small samples and for few initial centroids

K-means weakness

Non-globular clusters

K-means weakness

Wrong number of clusters

K-means weakness

Outliers and empty clusters

Real cases tend to be harder

- Different attributes of the feature vector have vastly different sizes
  - size of star versus color
- Can weight different features
  - how weight greatly affects outcome
- Difficulties can be overcome