Clustering Algorithms
for general similarity measures

Types of general 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. (((...)...)...)...

Similarity between clusters

Possible definitions:

I. similarity between most similar pair of objects with one in each cluster
   - called **single link**

II. similarity between least similar pair objects, one from each cluster
    - called **complete linkage**

Similarity between clusters, cont.

Possible definitions:

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

- Generally no representative point for a cluster;
  - compare K-means
- If using Euclidean distance as metric
  - centroid
  - bounding box

General **Agglomerative**

- Uses any computable cluster similarity measure \( \text{sim}(C_i, C_j) \)
- For \( n \) objects \( v_1, \ldots, v_n \), assign each to a singleton cluster \( C_i = \{v_i\} \)
- repeat \{
  - identify two most similar clusters \( C_i \) and \( C_k \) (could be ties – chose one pair)
  - delete \( C_i \) and \( C_k \) and add \( (C_i \cup C_k) \) to the set of clusters
- \} until only one cluster
- Dendrograms diagram the sequence of cluster merges.

Agglomerative: remarks

- *Intro. to IR* discusses in great detail for cluster similarity:
  - single-link, complete-link, avg. of all pairs, centroid
- Uses priority queues to get time complexity
  \( O((n^2 \log n)^* (\text{time to compute cluster similarity})) \)
  - one priority queue for each cluster: contains similarities to all other clusters plus bookkeeping info
  - time complexity more precisely:
    \( O((n^2)^* (\text{time to compute object-object similarity}) + (n \log n)^* (\text{time to compute sim(cluster, cluster, cluster)})) \)
- Problem with priority queue?
Single pass agglomerative-like

Given arbitrary order of objects to cluster: \(v_1, \ldots, v_n\) and threshold \(\tau\)

Put \(v_1\) in cluster \(C_1\) by itself

For \(i = 2\) to \(n\) {
  for all existing clusters \(C_j\)
    calculate \(\text{sim}(v_i, C_j)\);
    record most similar cluster to \(v_i\) as \(C_{\text{max}(i)}\)
    if \(\text{sim}(v_i, C_{\text{max}(i)}) > \tau\)
      add \(v_i\) to \(C_{\text{max}(i)}\)
    else create new cluster \(\{v_i\}\)
}

Issues

• put \(v_i\) in cluster after seeing only \(v_1, \ldots, v_{i-1}\)
• not hierarchical
• tends to produce large clusters
  – depends on \(\tau\)
• depends on order of \(v_i\)

Alternate perspective for single-link algorithm

• Build a minimum spanning tree (MST) - graph alg.
  – edge weights are pair-wise similarities
  – since in terms of similarities, not distances, really want maximum spanning tree
• For some threshold \(\tau\), remove all edges of similarity < \(\tau\)
• Tree falls into pieces => clusters
• Not hierarchical, but get hierarchy for sequence of \(\tau\)

Hierarchical Divisive: Template

1. Put all objects in one cluster
2. Repeat until all clusters are singletons
   a) choose a cluster to split
      • what criterion?
   b) replace the chosen cluster with the sub-clusters
      • split into how many?
      • how split?
      • “reversing” agglomerative => split in two
• cutting operation: cut-based measures seem to be a natural choice.
  – focus on similarity across cut - lost similarity
• not necessary to use a cut-based measure

An Example

An Example: 1\textsuperscript{st} cut
An Example: 2nd cut

An Example: stop at 3 clusters

Compare k-means result

Cut-based optimization

• weaken the connection between objects in different clusters rather than strengthening connection between objects within a cluster

• Are many cut-based measures
• We will look at one

Inter / Intra cluster costs

Given:
• \( V = \{v_1, \ldots, v_n\} \), the set of all objects
• A partitioning clustering \( C_1, C_2, \ldots, C_k \) of the objects: \( V = \bigcup_{i=1}^{k} C_i \).

Define:
• \( \text{cutcost}(C_p) = \sum_{v_i \in C_p} \sum_{v_j \in V \setminus C_p} \text{sim}(v_i, v_j) \).
• \( \text{intracost}(C_p) = \sum_{v_i, v_j \in C_p} \text{sim}(v_i, v_j) \).

Cost of a clustering

\[
\text{total relative cut cost}(C_1, \ldots, C_k) = \sum_{p=1}^{k} \frac{\text{cutcost}(C_p)}{\text{intracost}(C_p)}
\]

• contribution each cluster: ratio external similarity to internal similarity

Optimization

Find clustering \( C_1, \ldots, C_k \) that minimizes total relative cut cost(C_1, \ldots, C_k)
Simple example

- six objects
- similarity 1 if edge shown
- similarity 0 otherwise
- choice 1: cost UNDEFINED + 1/4
- choice 2: cost 1/1 + 1/3 = 4/3
- choice 3: cost 1/2 + 1/2 = 1 *prefer balance

Hierarchical divisive revisited

- can use one of cut-based algorithms to split a cluster
- how choose cluster to split next?
  - if building entire tree, doesn’t matter
  - if stopping a certain point, choose next cluster based on measure optimizing
    - e.g. for total relative cut cost, choose \( C_i \) with largest cutcost\( (C_i) / \) intracost\( (C_i) \)

Divisive Algorithm:
Iterative Improvement; no hierarchy
1. Choose initial partition \( C_1, \ldots, C_k \)
2. repeat {
   unlock all vertices
   repeat {
     choose some \( C_i \) at random
     choose an unlocked vertex \( v_j \) in \( C_i \)
     move \( v_j \) to that cluster, if any, such that move gives maximum decrease in cost
     lock vertex \( v_j \)
   } until all vertices locked
} until converge

Observations on algorithm

- heuristic
- uses randomness
- convergence usually improvement < some chosen threshold between outer loop iterations
- vertex “locking” insures that all vertices are examined before examining any vertex twice
- there are many variations of algorithm
- can use at each division of hierarchical divisive algorithm with \( k=2 \)
  - more computation than an agglomerative merge

Compare to k-means

- Similarities:
  - number of clusters, \( k \), is chosen in advance
  - an initial clustering is chosen (possibly at random)
  - iterative improvement is used to improve clustering

- Important difference:
  - divisive algorithm can minimize a cut-based cost
    - total relative cut cost uses external and internal measures
  - k-means maximizes only similarity within a cluster
    - ignores cost of cuts

Eigenvalues and clustering

General class of techniques for clustering a graph using eigenvectors of adjacency matrix (or similar matrix) called Spectral clustering

First described in 1973
Spectral clustering: brief overview

Given:
- \( k \): number of clusters
- \( nxn \) object-object sim. matrix \( S \) of non-neg. val.s

Compute:
1. Derive matrix \( L \) from \( S \) (straightforward computation)
   - e.g. Laplacian: are variations in def.
2. Find eigenvectors correspond. to \( k \) smallest eigenval.s of \( L \)
3. Use eigenvectors to define clusters
   - variety of ways to do this
   - all involve another, simpler, clustering
     - e.g. points on a line

Spectral clustering optimizes a cut measure
similar to total relative cut cost

Comparing clusterings

- Define external measure to
  - comparing two clusterings as to similarity
    - if one clustering "correct", one clustering by an algorithm, measures how well algorithm doing
      - refer to "correct" clusters as classes
        - "gold standard"
      - refer to computed clusters as clusters
- External measure independent of cost function optimized by algorithm

One measure: motivated by F-score in IR

- Given:
  - a set of classes \( S_1, \ldots, S_k \) of the objects
    use to define relevance
  - a computed clustering \( C_1, \ldots, C_k \) of the objects
    use to define retrieval
- Consider pairs of objects
  - pair in same class, call "similar pair" \( \equiv \) relevant
  - pair in different classes \( \equiv \) irrelevant
  - pair in same clusters \( \equiv \) retrieved
  - pair in different clusters \( \equiv \) not retrieved
- Use to define precision and recall

Properties of cluster F-score

- always \( \leq 1 \)
- Perfect match computed clusters to classes gives F-score = 1
- Symmetric
  - Two clusterings \( \{C_i\} \) and \( \{K_j\} \), neither "gold standard"
  - treat \( \{C_i\} \) as if are classes and compute F-score of \( \{K_j\} \) w.r.t. \( \{C_i\} \) = \( \text{F-score}_{Ci}([K_j]) \)
  - treat \( \{K_j\} \) as if are classes and compute F-score of \( \{C_i\} \) w.r.t. \( \{K_j\} \) = \( \text{F-score}_{Kj}([C_i]) \)
  - \( \implies \text{F-score}_{Ci}([K_j]) = \text{F-score}_{Kj}([C_i]) \)

Clustering f-score

Precision of the clustering w.r.t the gold standard = \[ \frac{\# \text{ similar pairs in the same cluster}}{\# \text{ pairs in the same cluster}} \]

Recall of the clustering w.r.t the gold standard = \[ \frac{\# \text{ similar pairs in the same cluster}}{\# \text{ similar pairs}} \]

F-score of the clustering w.r.t the gold standard = \[ \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]

Clustering f-score

- many applications
  - application determines similarity between objects
- menu of
  - cost functions to optimizes
  - similarity measures between clusters
  - types of algorithms
    - flat/hierarchical
    - constructive/iterative
    - algorithms within a type