Clustering Algorithms for general similarity measures

**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 \( C_j \) and \( C_k \) (could be ties – chose one pair)
  - delete \( C_j \) and \( C_k \) and add \( (C_j \cup C_k) \) 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 \( O((n^2 \log n) \times \text{(time to compute object-object similarity)}) \)
  - one priority queue for each cluster: contains similarities to all other clusters plus bookkeeping into
  - time complexity more precisely:
    \( O(n^2) \times \text{(time to compute object-object similarity)} + (n^2 \log n) \times \text{(time to compute cluster similarity)} \)
  - if know \( \text{sim}(C_z, C_j \cup C_k) \), then
    - if \( \text{sim}(v_i, C_j) > \tau \) and \( \text{sim}(v_i, C_k) > \tau \) add \( v_i \) to \( C_j \cup C_k \)
    - else create new cluster \( \{ v_i \} \)
- Problem with priority queue?

**Single pass agglomerative-like**
Given arbitrary order of objects to cluster: \( v_1, \ldots, v_n \) and threshold \( \tau \)
- Put \( v_i \) in cluster \( C_i \) 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 $\rightarrow$ 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: 1\(^{\text{st}}\) cut

An Example: 2\(^{\text{nd}}\) 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:
- \( \text{cutcost}(C_p) = \sum_{v_i \in C_p, v_j \in V \setminus C_p} \text{sim}(v_i, v_j). \)
- \( \text{intracost}(C_p) = \sum_{v_i \in C_p, v_j \in C_p} \text{sim}(v_i, v_j). \)

Cost of a clustering

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

- contribution each cluster: ratio external similarity to internal similarity

min-max cut optimization

Find clustering \( C_1, \ldots, C_k \) that minimizes 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 min-max cut, choose \( C_i \) with largest \( \frac{\text{cutcost}(C_i)}{\text{intracost}(C_i)} \)

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 improves < 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:**
  - min-max cut algorithm minimizes a cut-based cost
  - 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. values

**Compute:**
1. Derive matrix L from S (straightforward computation)
   - e.g. Laplacian: are variations in def.
2. eigenvectors corresponding to k smallest eigenvalues
   - 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 min-max cut

### HITS and clustering

- Non-principal eigenvectors of EE^T and E^TE have positive and negative component values
  - Denote a_e1, a_e2, ... matching h_e1, h_e2, ...
  - E is adjacency matrix
- For a matched pair of eigenvectors a_ej and h_ek
  - Denote k^{th} component of j^{th} pair: a_{ej}(k) and h_{ek}(k)
  - Make a "community" of size c (chosen constant):
    - Choose c pages with most positive h_{ek}(k) - hubs
    - Choose c pages with most positive a_{ej}(k) - authorities
  - Make another "community" of size c:
    - Choose c pages with most negative h_{ek}(k) - hubs
    - Choose c pages with most negative a_{ej}(k) - authorities

### 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
- External measure independent of cost function optimized by algorithm

one measure motivated by F-score in IR: combining precision and recall

- Given:
  a “correct” clustering S_1, ..., S_m of the objects (≡ relevant)
a computed clustering C_1, ..., C_n of the objects (≡ retrieved)
- Define:
  \[ \text{precision of } C_i \text{ w.r.t } S_q = \frac{|S_q \cap C_i|}{|C_i|} \]
  fraction of computed cluster that is “correct”
  \[ \text{recall of } C_i \text{ w.r.t } S_q = \frac{|S_q \cap C_i|}{|S_q|} \]
  fraction of a “correct” cluster found in a computed cluster
Fscore of $C_x$ w.r.t $S_q = F(x,q) = \frac{2r(x,q)p(x,q)}{r(x,q) + p(x,q)}$

combine precision and recall (Harmonic mean)

Fscore of $\{C_1, C_2, \ldots, C_k\}$ w.r.t $S_q = F(q) = \max_{x=1,\ldots,k} F(x,q)$

score of best computed cluster for $S_q$

Fscore of $\{C_1, C_2, \ldots, C_k\}$ w.r.t $\{S_1, S_2, \ldots, S_k\} = \sum_{q=1}^{k} \frac{(S_q / n) \cdot F(q)}{n}$ for $n$ items overall

weighted average of best scores over all correct clusters

- always $\leq 1$
- Perfect match computed clusters to correct clusters gives Fscore = 1
- Not symmetric: $\{C_i\}$ with respect to $\{S_j\}$

---

Clustering: wrap-up

- 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