Clustering Algorithms for general similarity measures

general similarity measure: specified by object X object similarity matrix

Types of general clustering methods

• constructive algorithms
• agglomerative versus divisive construction
  – 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:

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
    – ^ ^

General Agglomerative

• Uses any computable cluster similarity measure sim(C_i, C_j)
• For n objects v_1, ..., 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 U C_k) to the set of clusters
} until only one cluster
• Dendrograms diagram the sequence of cluster merges.

Similarity between clusters, cont.

Possible definitions:

III. average of pairwise similarity between all pairs of objects, one from each cluster
    – “centroid” similarity
    • Generally no representative point for a cluster;
    – compare K-means
    • If using Euclidean distance as metric
      – centroid
      – bounding box

Agglomerative: remarks

• \textit{Intro. to IR} discusses in great detail for cluster similarity:
  – single-link, complete-link, group average, centroid
• Uses priority queues to get time complexity \( O(n^2 \log n) \) (time to compute cluster similarity)
  – one priority queue for each cluster: contains similarities to all other clusters plus bookkeeping into
  – time complexity more precisely:
    \[
    O(n^2 + (n^2 \log n) + (n^2 \log n))
    \]
    \[
    = (n^2 \log n) + (n^2 \log n)
    \]
    \[
    \text{(time to compute \( \text{sim(cluster}_i, \text{cluster}_j U \text{cluster}_k) \) if know \( \text{sim(cluster}_i, \text{cluster}_j \) and \( \text{sim(cluster}_k, \text{cluster}_j \) )}
    \]
  – Problem with priority queue?
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. Yippy.com (metasearch)

Single pass agglomerative-like

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

Put \( v_i \) in cluster \( C_1 \) 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?**

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 algorithm
  - 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 \( \Rightarrow \) 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
An Example: 1\textsuperscript{st} cut

An Example: result of 1\textsuperscript{st} cut

An Example: 2\textsuperscript{nd} cut

An Example: stop at 3 clusters

Compare k-means result

Cut-based optimization

• focus on weak connections between objects in different clusters \textit{rather than strong connections} between objects within a cluster

• Are many cut-based measures
• We will look at two
Inter / Intra cluster costs

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

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

Cost of a clustering

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

Second cut-based measure:
Conductance

• define:
  \( s\text{-degree}(C_p) = \text{cutcost}(C_p) + 2 \times \text{intracost}(C_p) \)
  – model as graph, similarity = edge weights
  – \( s\text{-degree} \) is sum over all vertices in \( C_p \) of weights of edges touching vertex

• conductance \( (C_p) = \frac{\text{cutcost}(C_p)}{\min\{s\text{-degree}(C_p), s\text{-degree}(V-C_p)\}} \)

Optimization using conductance

• Choices:
  – minimize \( \sum_{p=1}^{k} \text{conductance}(C_p) \)
  – minimize \( \max_{p=1}^{k} \text{conductance}(C_p) \)

• Observations
  – conductance \( (C_p) = \text{conductance}(V-C_p) \)
  – Finding a cut \((C, V-C)\) with minimum conductance is NP-hard

Simple example

• six objects
• similarity 1 if edge shown
• similarity 0 otherwise

• choice 1: conductance 1/min(1,9) = 1
• choice 2: conductance 1/min(3, 7) = 1/3
• choice 3: conductance1/min(5, 5) = 1/5 *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 \( \text{cutcost}(C_i) / \text{intracost}(C_i) \)

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, 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, conductance use 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

- spectrum of a graph is list of eigenvalues, with multiplicity, of its adjacency matrix

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)
   - variety of definitions of \( L \)
     • e.g. Laplacian \( L=I-E \) if similarity is edge/no edge
2. find eigenvectors corresponding to \( k \) smallest eigenvals 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
    - 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

Clustering f-score

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

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

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

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. $\{C_i\}$ $\text{F-score}_{\{C_i\}}(\{K_j\})$
  - treat $\{K_j\}$ as if are classes and compute F-score of $\{C_i\}$ w.r.t. $\{K_j\}$ $\text{F-score}_{\{K_j\}}(\{C_i\})$
  - $\text{F-score}_{\{C_i\}}(\{K_j\}) = \text{F-score}_{\{K_j\}}(\{C_i\})$

another related external measure

Rand index

\[ \left( \frac{\# \text{ similar pairs in the same cluster} + \# \text{ dissimilar pairs in the different clusters}}{N(N-1)/2} \right) \]

percentage pairs that are correct

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