Clustering Algorithms: Divisive hierarchical and flat

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: 1\textsuperscript{st} cut

An Example: 2\textsuperscript{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:
- \( U = \{v_1, \ldots, v_n\} \), the set of all objects
- A partitioning clustering \( C_1, C_2, \ldots, C_k \) of the objects: \( U = \bigcup_{i=1, \ldots, k} C_i \).

Define:
- \( \text{cutcost}(C_p) = \sum_{v_i \in C_p, v_j \in U-C_p} \text{sim}(v_i, v_j) \).
- \( \text{intracost}(C_p) = \sum_{v_i, v_j \in C_p} \text{sim}(v_i, v_j) \).
Cost of a clustering

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
Iterative Improvement Algorithm

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:
  – min-max cut algorithm minimizes a cut-based cost
  – k-means maximizes only similarity within a cluster
    • ignores cost of cuts

Another method: Spectral clustering

Brief overview
Given:
• k: number of clusters
• nxn object-object sim. matrix S of non-neg. values
Compute:
1. Laplacian matrix L from S (straightforward computation)
   – are variations in def. Laplacian
2. eigenvectors corresponding to k smallest eigenvalues
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 min-max cut
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 $\text{cutcost}(C_i) / \text{intracost}(C_i)$

External measures

• comparing two clusterings as to similarity
• if one clustering “correct”, one clustering by an algorithm, measures how well algorithm doing
one measure motivated by F-score in IR: combining *precision* and *recall*

- Given:
  a “correct” clustering \( S_1, \ldots, S_k \) of the objects (\( \Xi \) relevant)
  a computed clustering \( C_1, \ldots, C_k \) of the objects (\( \Xi \) retrieved)

- Define:
  precision of \( C_x \) w.r.t \( S_q \):
  \[ p(x,q) = \frac{|S_q \cap C_x|}{|C_x|} \]
  fraction of computed cluster that is “correct”

  recall of \( C_x \) w.r.t \( S_q \):
  \[ r(x,q) = \frac{|S_q \cap C_x|}{|S_q|} \]
  fraction of a “correct” cluster found in a computed cluster

\[
\text{F-score of } C_x \text{ w.r.t } S_q = F(x,q) = \frac{2r(x,q) \cdot p(x,q)}{r(x,q) + p(x,q)}
\]
combine precision and recall (Harmonic mean)

\[
\text{F-score of } \{C_1, C_2, \ldots, C_k\} \text{ w.r.t } S_q = F(q) = \max_{x = 1, \ldots, k} F(x,q)
\]

\[
\text{F-score of } \{C_1, C_2, \ldots, C_k\} \text{ w.r.t } \{S_1, S_2, \ldots, S_k\} = \sum_{q = 1, \ldots, k} \left( \frac{|S_q|}{n} \right) * F(q)
\]
weighted average of best scores over all correct clusters
always \( \leq 1 \)

- Perfect match of computed clusters to correct clusters gives Fscore of 1