CS276A
Text Retrieval and Mining

Lecture 14

Recap

- Why cluster documents?
  - For improving recall in search applications
  - For speeding up vector space retrieval
  - Navigation
  - Presentation of search results
- *k*-means basic iteration
  - At the start of the iteration, we have *k* centroids.
  - Each doc assigned to the nearest centroid.
  - All docs assigned to the same centroid are averaged to compute a new centroid;
    - thus have *k* new centroids.

"The Curse of Dimensionality"

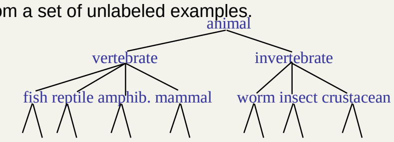
- Why document clustering is difficult
  - While clustering looks intuitive in 2 dimensions, many of our applications involve 10,000 or more dimensions…
  - High-dimensional spaces look different: the probability of random points being close drops quickly as the dimensionality grows.
  - One way to look at it: in large-dimension spaces, random vectors are almost all almost perpendicular. Why?
- Next class we will mention methods of dimensionality reduction … important for text

Today's Topics: Clustering 2

- Hierarchical clustering
  - Agglomerative clustering techniques
- Evaluation
- Term vs. document space clustering
- Multi-lingual docs
- Feature selection
- Labeling

Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of unlabeled examples.



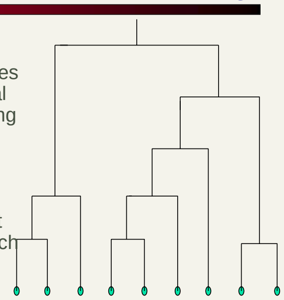
- One option to produce a hierarchical clustering is recursive application of a partitional clustering algorithm to produce a hierarchical clustering.

Hierarchical Agglomerative Clustering (HAC)

- Assumes a similarity function for determining the similarity of two instances.
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

A Dendogram: Hierarchical Clustering

- Dendrogram: Decomposes data objects into a several levels of nested partitioning (tree of clusters).
- Clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



HAC Algorithm

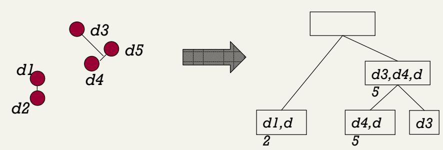
Start with all instances in their own cluster.
Until there is only one cluster:
  Among the current clusters, determine the two clusters, c_i and c_j, that are most similar.
  Replace c_i and c_j with a single cluster $c_i \cup c_j$

Hierarchical Clustering algorithms

- **Agglomerative (bottom-up):**
  - Start with each document being a single cluster.
  - Eventually all documents belong to the same cluster.
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster.
  - Eventually each node forms a cluster on its own.
- Does not require the number of clusters ***k*** in advance
- Needs a termination/readout condition
  - The final mode in both Agglomerative and Divisive is of no use.

Dendrogram: Document Example

- As clusters *agglomerate*, docs likely to fall into a hierarchy of “topics” or concepts.



“Closest pair” of clusters

- Many variants to defining closest pair of clusters
- “Center of gravity”
  - Clusters whose centroids (centers of gravity) are the most cosine-similar
- Average-link
  - Average cosine between pairs of elements
- Single-link
  - Similarity of the most cosine-similar (single-link)
- Complete-link
  - Similarity of the “furthest” points, the least cosine-similar

Hierarchical Clustering

- Key problem: as you build clusters, how do you represent the location of each cluster, to tell which pair of clusters is closest?
- Euclidean case: each cluster has a *centroid* = average of its points.
  - Measure intercluster distances by distances of centroids.

Single Link Agglomerative Clustering

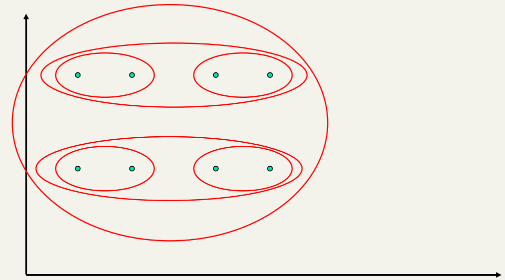
- Use maximum similarity of pairs:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- Can result in "straggly" (long and thin) clusters due to chaining effect.
  - Appropriate in some domains, such as clustering islands: "Hawai'i clusters"
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

Single Link Example



Complete Link Agglomerative Clustering

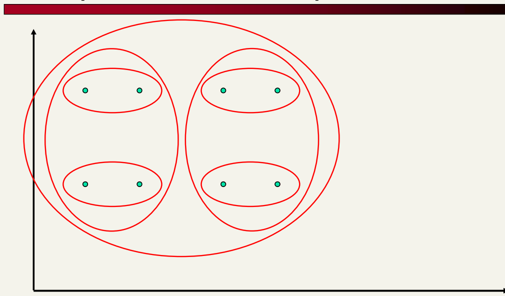
- Use minimum similarity of pairs:

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

Complete Link Example



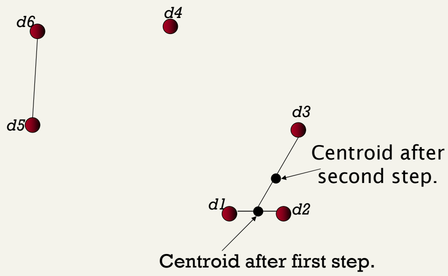
Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances which is $O(n^2)$.
- In each of the subsequent n–2 merging iterations, it must compute the distance between the most recently created cluster and all other existing clusters.
  - Since we can just store unchanged similarities
- In order to maintain an overall $O(n^2)$ performance, computing similarity to each other cluster must be done in constant time.
  - Else $O(n^2 \log n)$ or $O(n^3)$ if done naively

Key notion: *cluster representative*

- We want a notion of a representative point in a cluster
- Representative should be some sort of "typical" or central point in the cluster, e.g.,
  - point inducing smallest radii to docs in cluster
  - smallest squared distances, etc.
  - point that is the "average" of all docs in the cluster
    - Centroid or center of gravity

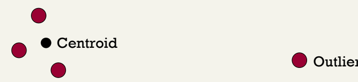
Example: *n*=6, *k*=3, closest pair of centroids



Outliers in centroid computation

- Can ignore outliers when computing centroid.
- What is an outlier?
  - Lots of statistical definitions, e.g.
  - *moment* of point to centroid > M × some cluster *moment*.

Say 10.



Group Average Agglomerative Clustering

- Use average similarity across all pairs within the merged cluster to measure the similarity of two clusters.

$$sim(c_i, c_j) = \frac{1}{|c_i \cup c_j|(|c_i \cup c_j| - 1)} \sum_{\vec{x} \in (c_i \cup c_j)} \sum_{\vec{y} \in (c_i \cup c_j): \vec{y} \neq \vec{x}} sim(\vec{x}, \vec{y})$$

- Compromise between single and complete link.
- Two options:
  - Averaged across all ordered pairs in the merged cluster
  - Averaged over all pairs *between* the two original clusters
- Some previous work has used one of these options; some the other. No clear difference in efficacy

Computing Group Average Similarity

- Assume cosine similarity and normalized vectors with unit length.
- Always maintain sum of vectors in each cluster.

$$\vec{s}(c_j) = \sum_{\vec{x} \in c_j} \vec{x}$$

- Compute similarity of clusters in constant time:

$$sim(c_i, c_j) = \frac{(\vec{s}(c_i) + \vec{s}(c_j)) \bullet (\vec{s}(c_i) + \vec{s}(c_j)) - (|c_i| + |c_j|)}{(|c_i| + |c_j|)(|c_i| + |c_j| - 1)}$$

Efficiency: Medoid As Cluster Representative

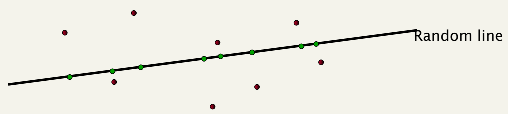
- The centroid does not have to be a document.
- Medoid: A cluster representative that is one of the documents
- For example: the document closest to the centroid
- One reason this is useful
  - Consider the representative of a large cluster (>1000 documents)
  - The centroid of this cluster will be a dense vector
  - The medoid of this cluster will be a sparse vector
- Compare: mean/centroid vs. median/medoid

Exercise

- Consider agglomerative clustering on *n* points on a line. Explain how you could avoid n^3 distance computations - how many will your scheme use?

Efficiency: “Using approximations”

- In standard algorithm, must find closest pair of centroids at each step
- Approximation: instead, find nearly closest pair
  - use some data structure that makes this approximation easier to maintain
  - simplistic example: maintain closest pair based on distances in projection on a random line



Term vs. document space

- So far, we clustered docs based on their similarities in term space
- For some applications, e.g., topic analysis for inducing navigation structures, can “dualize”:
  - use docs as axes
  - represent (some) terms as vectors
  - proximity based on co-occurrence of terms in docs
  - now clustering terms, *not* docs

Term vs. document space

- Cosine computation
  - Constant for docs in term space
  - Grows linearly with corpus size for terms in doc space
- Cluster labeling
  - clusters have clean descriptions in terms of noun phrase co-occurrence
  - Easier labeling?
- Application of term clusters
  - Sometimes we want term clusters (example?)
  - If we need doc clusters, left with problem of binding docs to these clusters

Multi-lingual docs

- E.g., Canadian government docs.
- Every doc in English and equivalent French.
  - Must cluster by concepts rather than language
- Simplest: pad docs in one language with dictionary equivalents in the other
  - thus each doc has a representation in both languages
- Axes are terms in both languages

Feature selection

- Which terms to use as axes for vector space?
- Large body of (ongoing) research
- IDF is a form of feature selection
  - Can exaggerate noise e.g., mis-spellings
- Better is to use highest weight *mid-frequency* words – the most discriminating terms
- Pseudo-linguistic heuristics, e.g.,
  - drop stop-words
  - stemming/lemmatization
  - use only nouns/noun phrases
- Good clustering should “figure out” some of these

Major issue - labeling

- After clustering algorithm finds clusters - how can they be useful to the end user?
- Need pithy label for each cluster
  - In search results, say “Animal” or “Car” in the ***jaguar*** example.
  - In topic trees (Yahoo), need navigational cues.
    - Often done by hand, a posteriori.

How to Label Clusters

- Show titles of typical documents
  - Titles are easy to scan
  - Authors create them for quick scanning!
  - But you can only show a few titles which may not fully represent cluster
- Show words/phrases prominent in cluster
  - More likely to fully represent cluster
  - Use distinguishing words/phrases
    - Differential labeling
  - But harder to scan

Labeling

- Common heuristics - list 5-10 most frequent terms in the centroid vector.
  - Drop stop-words; stem.
- Differential labeling by frequent terms
  - Within a collection "Computers", clusters all have the word ***computer*** as frequent term.
  - Discriminant analysis of centroids.
- Perhaps better: distinctive noun phrase

Evaluation of clustering

- Perhaps the most substantive issue in data mining in general:
  - how do you measure goodness?
- Most measures focus on computational efficiency
  - Time and space
- For application of clustering to search:
  - Measure retrieval effectiveness

Approaches to evaluating

- Anecdotal
- User inspection
- Ground "truth" comparison
  - Cluster retrieval
- Purely quantitative measures
  - Probability of generating clusters found
  - Average distance between cluster members
- Microeconomic / utility

Anecdotal evaluation

- Probably the commonest (and surely the easiest)
  - "I wrote this clustering algorithm and look what it found!"
- No benchmarks, no comparison possible
- Any clustering algorithm will pick up the easy stuff like partition by languages
- Generally, unclear scientific value.

User inspection

- Induce a set of clusters or a navigation tree
- Have subject matter experts evaluate the results and score them
  - some degree of subjectivity
- Often combined with search results clustering
- Not clear how reproducible across tests.
- Expensive / time-consuming

Ground "truth" comparison

- Take a union of docs from a taxonomy & cluster
  - Yahoo!, ODP, newspaper sections …
- Compare clustering results to baseline
  - e.g., 80% of the clusters found map "cleanly" to taxonomy nodes
  - How would we measure this? "Subjective"
- But is it the "right" answer?
  - There can be several equally right answers
- For the docs given, the static prior taxonomy may be incomplete/wrong in places
  - the clustering algorithm may have gotten right things not in the static taxonomy

Ground truth comparison

- Divergent goals
- Static taxonomy designed to be the "right" navigation structure
  - somewhat independent of corpus at hand
- Clusters found have to do with vagaries of corpus
- Also, docs put in a taxonomy node may not be the most representative ones for that topic
  - cf Yahoo!

Microeconomic viewpoint

- Anything - including clustering - is only as good as the economic utility it provides
- For clustering: net economic gain produced by an approach (vs. another approach)
- Strive for a concrete optimization problem
- Examples
  - recommendation systems
  - clock time for interactive search
    - expensive

Evaluation example: Cluster retrieval

- Ad-hoc retrieval
- Cluster docs in returned set
- Identify best cluster & only retrieve docs from it
- How do various clustering methods affect the quality of what's retrieved?
- Concrete measure of quality:
  - Precision as measured by user judgements for these queries
- Done with TREC queries

Evaluation

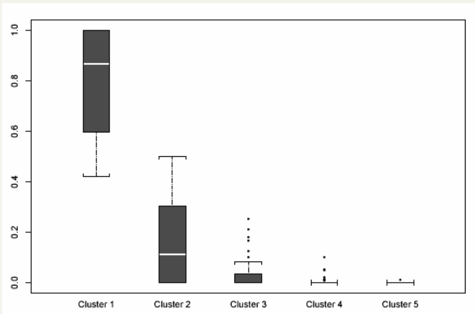
- Compare two IR algorithms
  - 1. send query, present ranked results
  - 2. send query, cluster results, present clusters
- Experiment was simulated (no users)
  - Results were clustered into 5 clusters
  - Clusters were ranked according to percentage relevant documents
  - Documents within clusters were ranked according to similarity to query

Sim-Ranked vs. Cluster-Ranked

	Precision at Cutoffs		
CutOff	Sim-Ranked	Cluster-Ranked	% Increase
5	.342	.428	.252
10	.314	.401	.277
20	.276	.363	.312

Table 4: Precision at small document cutoff levels for the one-step algorithm.

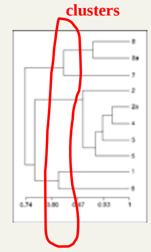
Relevance Density of Clusters



Buckshot Algorithm

- Another way to an efficient implementation:
  - Cluster a sample, then assign the entire set
- Buckshot combines HAC and K-Means clustering.
- First randomly take a sample of instances of size $\sqrt{n}$
- Run group-average HAC on this sample, which takes only O(n) time.
- Use the results of HAC as initial seeds for K-means.
- Overall algorithm is O(n) and avoids problems of bad seed selection.

Cut where
You have k
clusters



Uses HAC to bootstrap K-means

Bisecting K-means

- Divisive hierarchical clustering method using K-means
- For I=1 to k-1 do {
  - Pick a leaf cluster C to split
  - For J=1 to ITER do {
    - Use K-means to split C into two sub-clusters, C_1 and C_2
    - Choose the best of the above splits and make it permanent}

    }

  }
- Steinbach et al. suggest HAC is better than k-means but Bisecting K-means is better than HAC for their text experiments

Exercises

- Consider running 2-means clustering on a corpus, each doc of which is from one of two different languages. What are the two clusters we would expect to see?
- Is agglomerative clustering likely to produce different results to the above?
- Is the centroid of normalized vectors normalized?
- Suppose a run of agglomerative clustering finds $k=7$ to have the highest value amongst all k. Have we found the highest-value clustering amongst all clusterings with $k=7$?

Resources

- Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections (1992)
  - Cutting/Karger/Pedersen/Tukey
  - http://citeseer.ist.psu.edu/cutting92scattergather.html
- Data Clustering: A Review (1999)
  - Jain/Murty/Flynn
  - http://citeseer.ist.psu.edu/jain99data.html
- A Comparison of Document Clustering Techniques
  - Michael Steinbach, George Karypis and Vipin Kumar. TextMining Workshop. KDD. 2000.

Resources

- Initialization of iterative refinement clustering algorithms. (1998)
  - Fayyad, Reina, and Bradley
  - http://citeseer.ist.psu.edu/fayyad98initialization.html
- Scaling Clustering Algorithms to Large Databases (1998)
  - Bradley, Fayyad, and Reina
  - http://citeseer.ist.psu.edu/bradley98scaling.html