CSCI 5417
Information Retrieval Systems

Jim Martin

Lecture 15
10/13/2011

Today 10/13

- More Clustering
  - Finish flat clustering
  - Hierarchical clustering

K-Means

- Assumes documents are real-valued vectors.
- Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

$$\mu(c) = \frac{1}{|c|}\sum_{\vec{x}\in c}\vec{x}$$

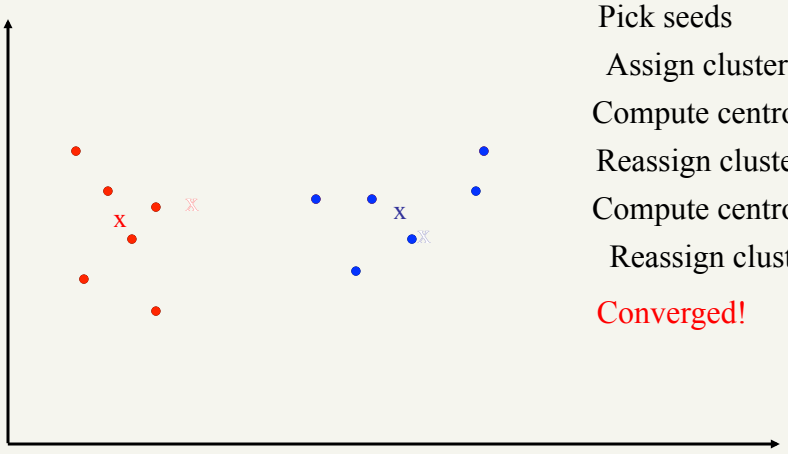
- Iterative reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)

K-Means Algorithm

Select K random docs $\{s_1, s_2, \ldots s_K\}$ as seeds.
Until stopping criterion:
  For each doc d_i:
    Assign d_i to the cluster c_j
    such that $dist(d_i, s_j)$ is minimal.

  For each cluster c_j
    $s\_j = m(c\_j)$

K Means Example (*K*=2)



Pick seeds
Assign clusters
Compute centroids
Reassign clusters
Compute centroids
Reassign clusters
Converged!

Termination conditions

- Several possibilities
  - A fixed number of iterations
  - Doc partition unchanged
  - Centroid positions don't change

Convergence

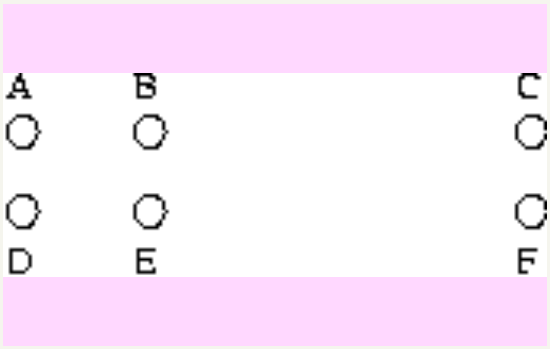
- Why should the *K*-means algorithm ever reach a *fixed point*?
  - A state in which clusters don't change.
- *K*-means is a special case of a general procedure known as the *Expectation Maximization (EM) algorithm*.
  - EM is known to converge.
  - Number of iterations could be large.

But in practice usually isn't

Seed Choice

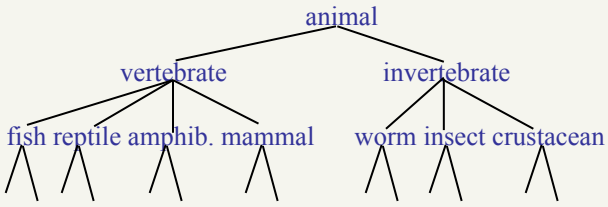
- Results can vary based on random seed selection.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
  - Select good seeds using a heuristic (e.g., doc least similar to any existing mean)
  - Try out multiple starting points
  - Initialize with the results of another method.

Do this with K=2



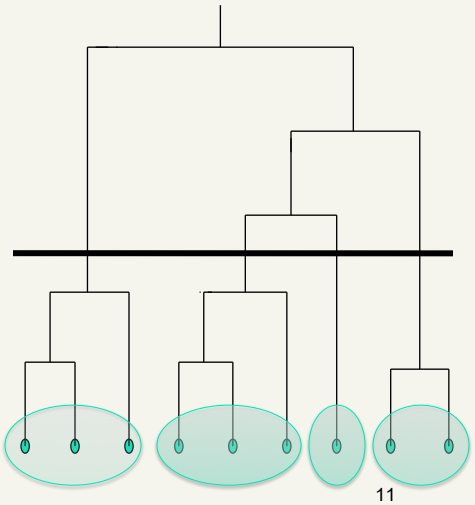
Hierarchical Clustering

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



Dendrogram: Hierarchical Clustering

- Traditional clustering partition is obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.



11

Break

- Past HW
  - Best score on part 2 is .437
  - Best approaches
    - Multifield indexing of title/keywords/abstract
    - Snowball (English), Porter
    - Tuning the stop list
    - Ensemble (voting)
  - Mixed results
    - Boosts
    - Relevance feedback

Descriptions

- For the most part, your approaches were pretty weak (or your descriptions were)
  - Failed to report R-Precision
  - Use of some kind of systematic approach
    - X didn't work
    - Interactions between approaches
    - Lack of details
      - Use relevance feedback and it gave me Z
      - I changed the stop list
      - Boosted the title field
      - Etc.

Next HW

- Due 10/25
- I have a new untainted test set
  - So don't worry about checking for the test document; it won't be there

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*** to be known in advance
  - But it does need a cutoff or threshold parameter condition

Hierarchical -> Partition

- Run the algorithm to completion
  - Take a slice across the tree at some level
    - Produces a partition
- Or insert an early stopping condition into either top-down or bottom-up

Hierarchical Agglomerative Clustering (HAC)

- Assumes a similarity function for determining the similarity of two instances and two clusters.
- Starts with all instances in separate clusters 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.

Hierarchical Clustering

- Key problem: as you build clusters, how do you represent each cluster, to tell which pair of clusters is closest?

"Closest pair" in Clustering

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

Single Link Agglomerative Clustering

- Use maximum similarity of pairs:

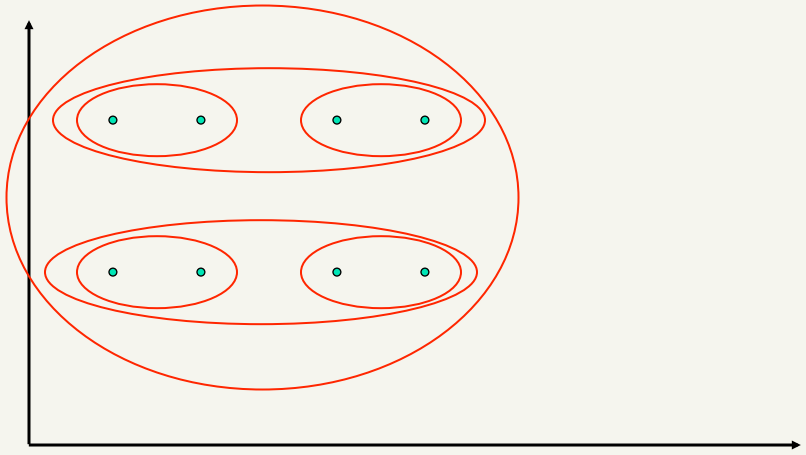
$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- Can result in "straggly" (long and thin) clusters due to chaining effect.

- 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

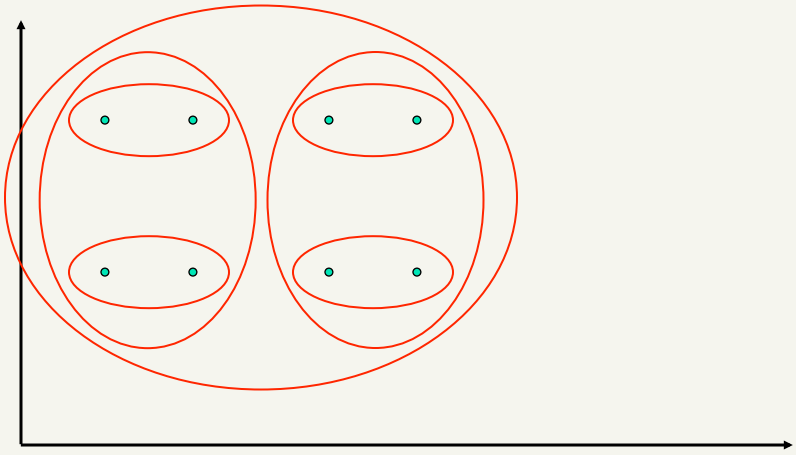
- Use minimum similarity of pairs:

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

- 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



Misc. Clustering Topics

- Clustering terms
- Clustering people
- Feature selection
- Labeling clusters

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, you can "dualize":
  - Use docs as axes
  - Represent (some) terms as vectors
  - Cluster terms, *not* docs

Clustering people

- Take documents (pages) containing mentions of ambiguous names and partition the documents into bins with identical referents.
  - SemEval competition
    - Web People Search Task: Given a name as a query to google, cluster the top 100 results so that each cluster corresponds to a real individual out in the world

Labeling clusters

- 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.

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 phrases
  - Requires NP chunking

Summary

- In clustering, clusters are inferred from the data without human input (unsupervised learning)
- In practice, it's a bit less clear. There are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, . . .