Document Clustering

CISC489/689-010, Lecture #17

Monday, April 20th

Ben Carterette

Classification Review

- Items (documents, web pages, emails) are represented with features
- Some items are assigned a class from a fixed set
- Classification goal: use known class assignments to "learn" a general function f(x) for classifying new instances
- Naïve Bayes classifier:

$$f(x) = \arg\max_j P(C_j|x) = \arg\max_j \prod_{i=1}^{n} P(t_i|C_j)P(C_j)$$

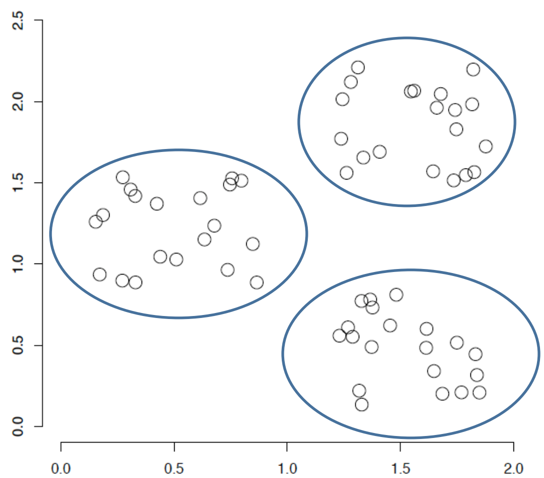
Clustering

- A set of algorithms that attempt to find latent (hidden) structure in a set of items
- Goal is to identify groups (clusters) of similar items
  - Two items in the same group should be similar to one another
  - An item in one group should be dissimilar to an item in another group

Clustering Example

- Suppose I gave you the shape, color, vitamin C content, and price of various fruits and asked you to cluster them
  - What criteria would you use?
  - How would you define similarity?
- Clustering is very sensitive to how items are represented and how similarity is defined!

Clustering in Two Dimensions



How would you cluster these points?

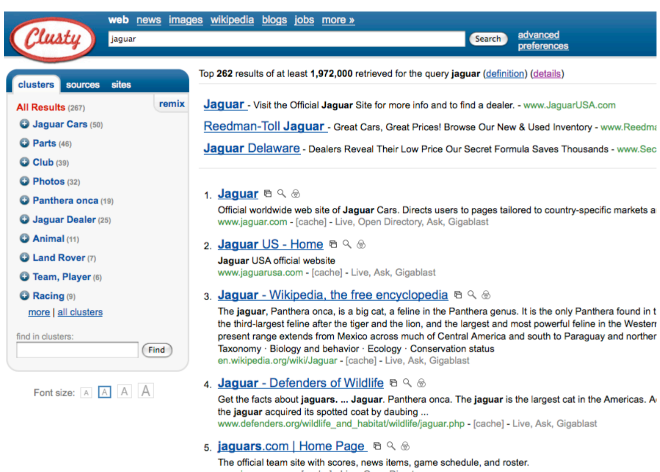
Classification vs Clustering

- Classification is *supervised*
  - You are given a fixed set of classes
  - You are given class labels for certain instances
  - This is data you can use to learn the classification function
- Clustering is *unsupervised*
  - You are not given any information about how documents should be grouped
  - You don't even know how many groups there should be
  - There is no training data to learn from
- One way to think of it: learning vs discovery

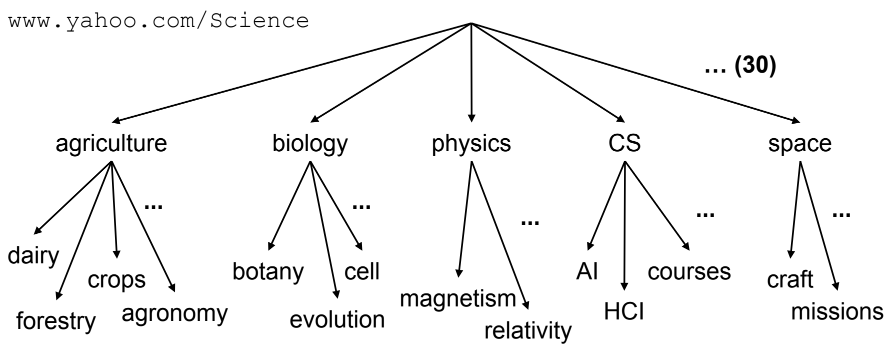
Clustering in IR

- Cluster hypothesis:
  – "Closely associated documents tend to be relevant to the same requests" – van Rijsbergen '79
- Document clusters may capture relevance better than individual documents
- Clusters may capture "subtopics"

Cluster-Based Search



Yahoo! Hierarchy



Not based on clustering approaches, but one possible use of clustering.

Example from "Introduction to IR" slides by Hinrich Schutze

Clustering Algorithms

- General outline of clustering algorithms
  1. Decide how items will be represented (e.g., feature vectors)
  2. Define similarity measure between pairs or groups of items (e.g., cosine similarity)
  3. Determine what makes a "good" clustering
  4. Iteratively construct clusters that are increasingly "good"
  5. Stop after a local/global optimum clustering is found
- Steps 3 and 4 differ the most across algorithms

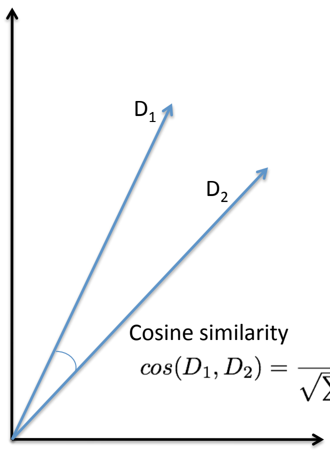
Item Representation

- Typical representation for documents in IR:
  - "Bag of words" – a vector of terms appearing in the document with associated weights
  - N-grams
  - etc.
- Any representation used in retrieval can (theoretically) be used in clustering or classification
  - Though specialized representations may be better for particular tasks

Item Similarity

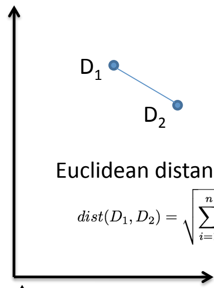
- Cluster hypothesis suggests that document similarity should be based on information content
  - Ideally semantic content, but we have already seen how hard that is
- Instead, use the same idea as in query-based retrieval
  - The score of a document to a query is based on how similar they are in the words they contain
    - Cosine angle between vectors; P(R | Q, D); P(Q | D)
  - The similarity of two documents will be based on how similar they are in the words they contain

Document Similarity



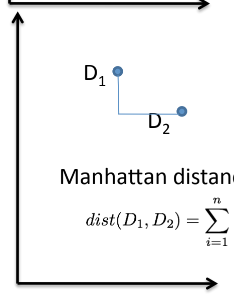
Cosine similarity

$$cos(D_1, D_2) = \frac{\sum_{i=1}^{n} d_{i1} d_{i2}}{\sqrt{\sum_{i=1}^{n} d_{i1}^2}\sqrt{\sum_{i=1}^{n} d_{i2}^2}}$$



Euclidean distance

$$dist(D_1, D_2) = \sqrt{\sum_{i=1}^{n} (d_{i1} - d_{i2})^2}$$



Manhattan distance

$$dist(D_1, D_2) = \sum_{i=1}^{n} |d_{i1} - d_{i2}|$$

“similarity” vs “distance”: in practice, you can use either

What Makes a Good Cluster?

- Large vs small?
  - Is it OK to have a cluster with one item?
  - Is it OK to have a cluster with 10,000 items?
- Similarity between items?
  - Is it OK for things in a cluster to be very far apart, as long as they are closer to each other than to things in other cluster?
  - Is it OK for things to be so close together that other similar things are excluded from the cluster?
- Overlapping vs non-overlapping?
  - Is it OK for two clusters to contain some items in common?
  - Should clusters “nest” within one another?

Example Approaches

- "Hard" clustering
  - Every item is in only one cluster
- "Soft" clustering
  - Items can belong to more than one cluster
  - Nested hierarchy: item belongs to a cluster, as well as the cluster's parent cluster, and so on
  - Non-nested: item belongs to two separate clusters
    - E.g. a document about jaguar cats riding in Jaguar cars might belong to the "animal" cluster and the "car" cluster

Example Approaches

- Flat clustering:
  - No overlap: every item in exactly one cluster
  - K clusters total
  - Start with random groups, then refine them until they are "good"
- Hierarchical clustering:
  - Clusters are nested: a cluster can be made up of two or more smaller clusters
  - No fixed number
  - Start with one group and split it until there are good clusters
  - Or start with N groups and agglomerate them until there are good clusters

Flat Clustering

- Goal: partition N documents into K clusters
- Given: N document feature vectors, a number K
- Optimal algorithm:
  - Try every possible clustering and take whichever one is the "best"
  - Computation time: $O(K^N)$
- Heuristic approach:
  - Split documents into K clusters randomly
  - Move documents from one cluster to another until the clusters seem "good"

K-Means Clustering

- K-means is a partitioning heuristic
- Documents are represented as vectors

$$D_1 = \begin{bmatrix} d_{11} & d_{12} & d_{13} & \dots & d_{1n} \end{bmatrix}$$
$$D_2 = \begin{bmatrix} d_{21} & d_{22} & d_{23} & \dots & d_{2n} \end{bmatrix}$$

- Clusters are represented as a *centroid vector*

$$C = \frac{1}{|C|} \sum_{D_i \in C} D_i = \frac{1}{|C|} \begin{bmatrix} d_{11} + d_{21} & d_{12} + d_{22} & d_{13} + d_{23} & \dots & d_{1n} + d_{2n} \end{bmatrix}$$

- Basic algorithm:
  - Step 0: Choose *K* docs to be initial cluster centroids
  - Step 1: Assign points to closet centroid
  - Step 2: Recompute cluster centroids
  - Step 3: Goto 1

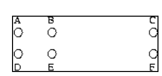
K-Means Clustering Algorithm

Input: N documents, a number K

- A[1], A[2], …, A[N] := 0
- C_1, C_2, …, C_K := initial cluster assignment (pick K docs)
- do
  - changed = false
  - for each document D_i, i = 1 to N
    - k = argmin_k dist(D_i, C_k)   (equivalently, k = argmax_k sim(D_i, C_k))
    - if A[i] != k then
      - A[i] = k
      - changed = true
  - if changed then C_1, C_2, …, C_K := cluster centroids
- until changed is false
- return A[1..N]

K-Means Decisions

- K – number of clusters
  - K=2? K=10? K=500?
- Cluster initialization
  - Random initialization often used
  - A bad initial assignment can result in bad clusters



- Distance measure
  - Cosine similarity most common
  - Euclidean distance, Manhattan distance, manifold distances
- Stopping condition
  - Until no documents have changed clusters
  - Until centroids do not change
  - Fixed number of iterations

K-Means Advantages

- Computationally efficient
  - Distance between two documents = O(V)
  - Distance of each doc to each centroid = O(KNV)
  - Calculating centroids = O(NV)
  - For m iterations, O(m(KNV+NV)) = O(mKNV)
- Tends to converge quickly (m is relatively small)
- Easy to implement

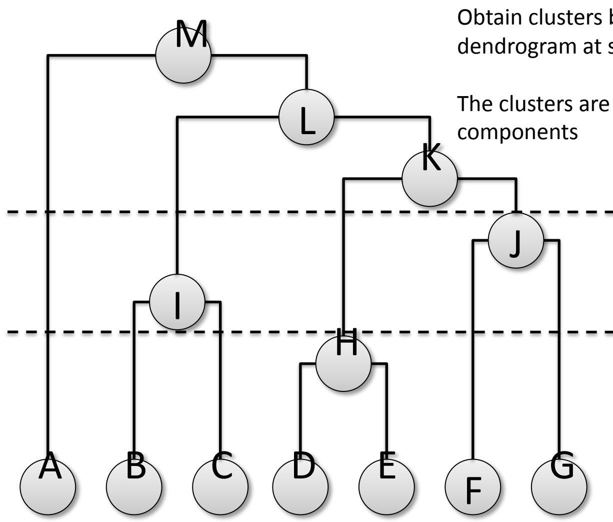
K-Means Disadvantages

- What should K be?
- Clusters have fixed geometric shape
  - Spherical
  - Very sensitive to dimensions and weights
- No notion of outliers
  - A document that's far away from everything will either be in a cluster on its own or in some very wide (geometrically speaking) cluster

Hierarchical Clustering

- Goal: construct a hierarchy of clusters
  - The top level of the hierarchy consists of a single cluster with all items in it
  - The bottom level of the hierarchy consists of *N* (# items) singleton clusters
- Two types of hierarchical clustering
  - Divisive ("top down")
  - Agglomerative ("bottom up")
- Hierarchy can be visualized as a *dendrogram*

Example Dendrogram



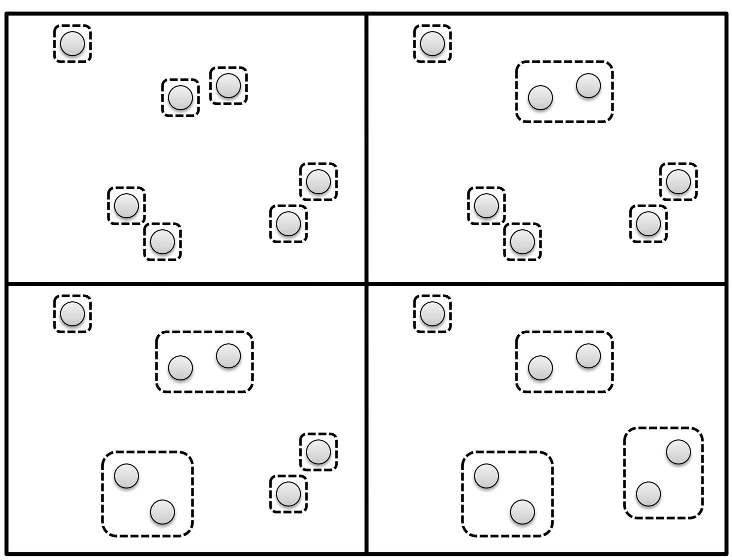
Obtain clusters by cutting dendrogram at some threshold

The clusters are the connected components

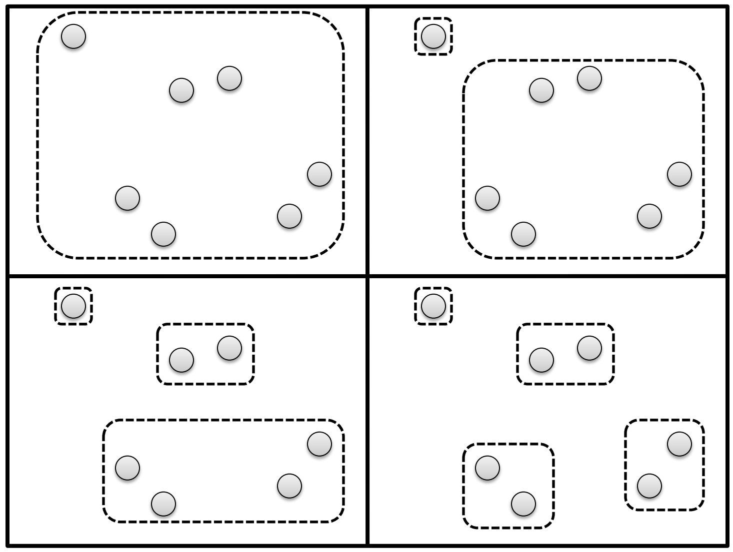
Divisive and Agglomerative Hierarchical Clustering

- Divisive
  - Start with a single cluster consisting of all of the items
  - Until only singleton clusters exist…
    - **Divide** an existing cluster into two new clusters
- Agglomerative
  - Start with *N* (# items) singleton clusters
  - Until a single cluster exists…
    - **Combine** two existing cluster into a new cluster
- How do we know how to divide or combine clusters?
  - Define a division or combination cost
  - Perform the division or combination with the lowest cost

Agglomerative Hierarchical Clustering



Divisive Hierarchical Clustering



Clustering Costs

- Similarity measured between two different clusters
- Single linkage

$$COST(C_i, C_j) = \min\{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}$$

- Complete linkage

$$COST(C_i, C_j) = \max\{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}$$

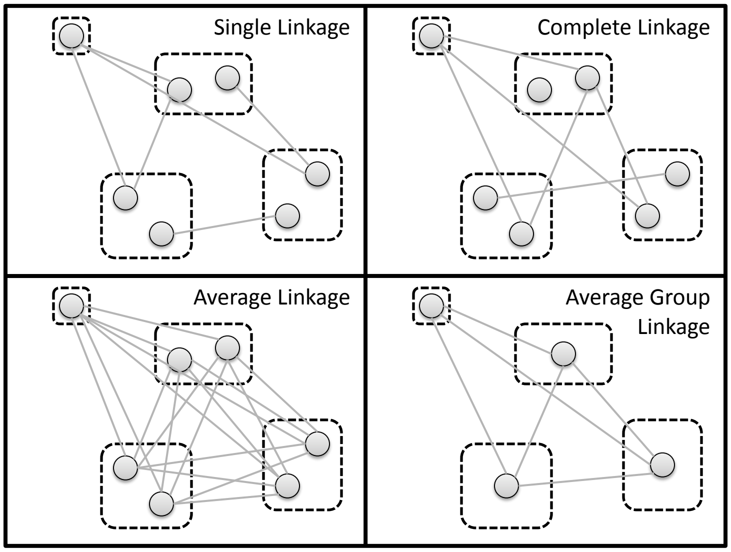
- Average linkage

$$COST(C_i, C_j) = \frac{\sum_{X_i \in C_i, X_j \in C_j} dist(X_i, X_j)}{|C_i||C_j|}$$

- Average group linkage

$$COST(C_i, C_j) = dist(\mu_{C_i}, \mu_{C_j})$$

Clustering Strategies



Single Linkage

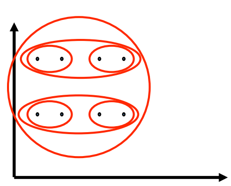
- Similarity between two clusters = minimum distance between all pairs of documents
  - (Or maximum similarity)

$$COST(C_i, C_j) = \min\{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}$$

- After merging two clusters,

$$COST((C_i \cup C_j), C_k) = \min\{COST(C_i, C_j), COST(C_j, C_k)\}$$

- Tends to produce “stringier” hierarchies
- Example:



-

Complete Linkage

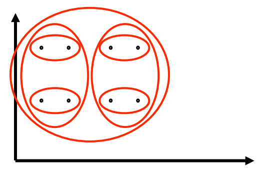
- Similarity between two clusters = maximum distance between all pairs of documents
  - (Or minimum similarity)

$$COST(C_i, C_j) = \max\{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}$$

- After merging two clusters,

$$COST((C_i \cup C_j), C_k) = \max\{COST(C_i, C_j), COST(C_j, C_k)\}$$

- Tends to produce more "spherical" clusters
- Example:



-

Hierarchical Clustering Advantages

- Flexibility
  - No fixed number of clusters
  - Can change threshold to get different clusters
    - Lower threshold: more specific clusters
    - Higher threshold: broader clusters
  - Can change cost function to get different clusters
- Hierarchical structure may be meaningful
  - E.g. articles about jaguar cats agglomerate together, articles about tigers agglomerate, then both agglomerate to articles about big cats

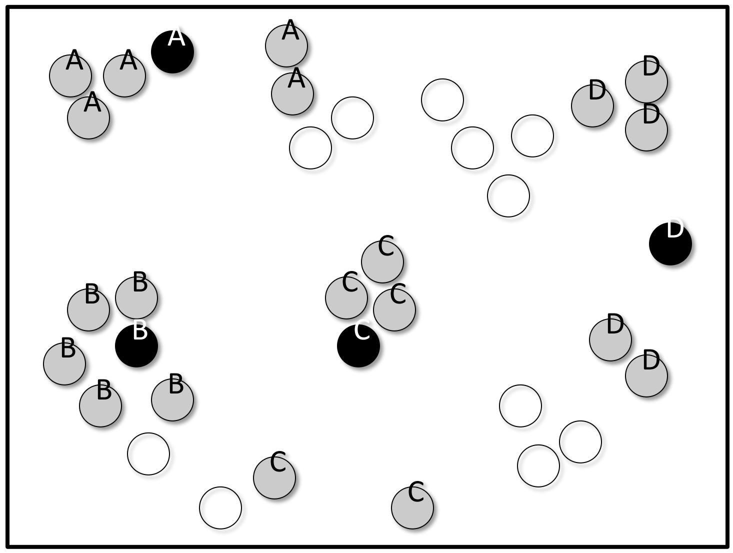
Hierarchical Clustering Disadvantages

- Computationally inefficient
  - Similarity between two documents = O(V)
  - Requires similarity between all pairs of documents = $O(VN^2)$
  - Then requires similarity between most recent cluster and all existing clusters, naïvely $O(N^3)$
    - $O(N^2 \log N)$ with a little cleverness

K-Nearest Neighbor Clustering

- K-means clustering partition items into clusters
- Hierarchical clustering creates nested clusters
- K-nearest neighbor clustering forms one cluster per item
  - The cluster for item *j* consists of *j* and *j*'s *K* nearest neighbors
  - Clusters now overlap
  - Some things don't get clustered

5-Nearest Neighbor Clustering



Evaluating Clustering

- Clustering will never be 100% accurate
  - Documents will be placed in clusters they don't belong in
  - Documents will be excluded from clusters they should be part of
  - A natural consequence of using term statistics to represent the information contained in documents
- Like retrieval and classification, clustering effectiveness must be evaluated
- Evaluating clustering is challenging, since it is an ***unsupervised*** learning task

Evaluating Clustering

- If labels exist, can use standard IR metrics, such as precision and recall
  - In this case we are evaluating the ability of our algorithm to discover the "true" latent information

	Class A	Class B	Class C	Class D
Cluster 1	A_1	B_1	C_1	D_1
Cluster 2	A_2	B_2	C_2	D_2
Cluster 3	A_3	B_3	C_3	D_3
Cluster 4	A_4	B_4	C_4	D_4

$$prec_{\text{cluster 1}} = \frac{A_1}{A_1 + B_1 + C_1 + D_1}$$
$$rec_{\text{cluster 1}} = \frac{A_1}{A_1 + A_2 + A_3 + A_4}$$

- This only works if you have some way to "match" clusters to classes
- What if there are fewer or more clusters than classes?

Evaluating Clusters

- "Purity": the ratio between the number of documents from the dominant class in C to the size of C

$$purity(C_i) = \frac{1}{|C_i|} \max_j |X \text{ s.t. } X \in C_i \text{ and } X \in K_j|$$

  - C_i is a cluster; K_j is a class
- Not such a great measure
  - Does not take into account coherence of the class
  - Optimized by making N clusters, one for each document

Evaluating Clusters

- With no labeled data even more difficult
- Best approach:
  - Evaluate the system that the clustering is part of
  - E.g. if clustering is used to aid retrieval, evaluate the cluster-aided retrieval
  - More on Wednesday