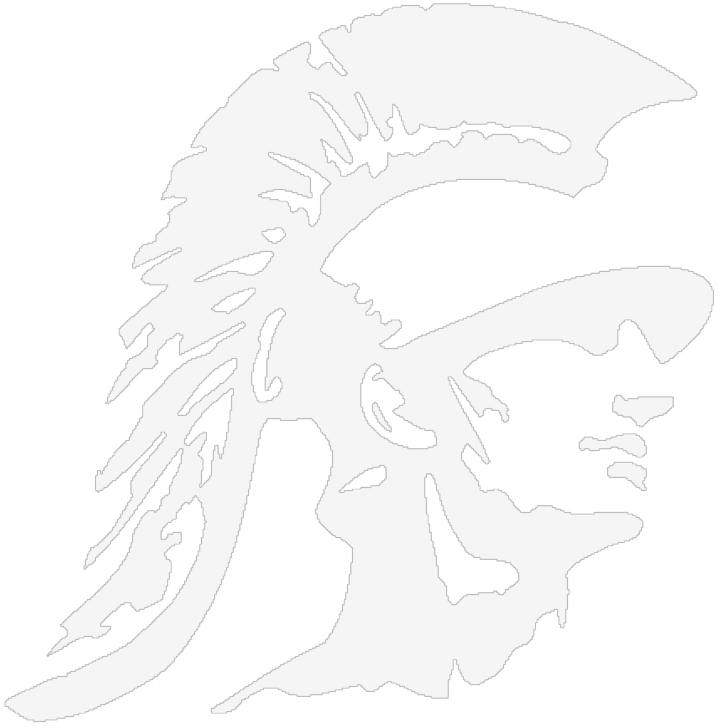
Clustering

Today's Topic: Clustering

- **Document clustering**
  - Motivations
  - Document representations
  - Success criteria
- **Clustering algorithms**
  - Partitional
  - Hierarchical



What is Clustering?

- **Clustering: the process of grouping a set of objects into classes of similar objects**
  - Documents within a cluster should be similar.
  - Documents from different clusters should be dissimilar.
- **Clustering is the most common form of *unsupervised learning***
    - Unsupervised learning = learning from raw data, as opposed to supervised learning where a classification of examples is given
- Clustering is a common and important task that finds many applications in IR and other places

Applications of Clustering 3 Scenarios

1. A telephone company needs to establish its network by putting its towers in a particular region it has acquired. The location of putting these towers can be found by clustering algorithm so that all its users receive optimum signal strength

2. The Miami DEA wants to make its law enforcement more stringent and hence have decided to make their patrol vans stationed across the area so that the areas of high crime rates are in vicinity to the patrol vans

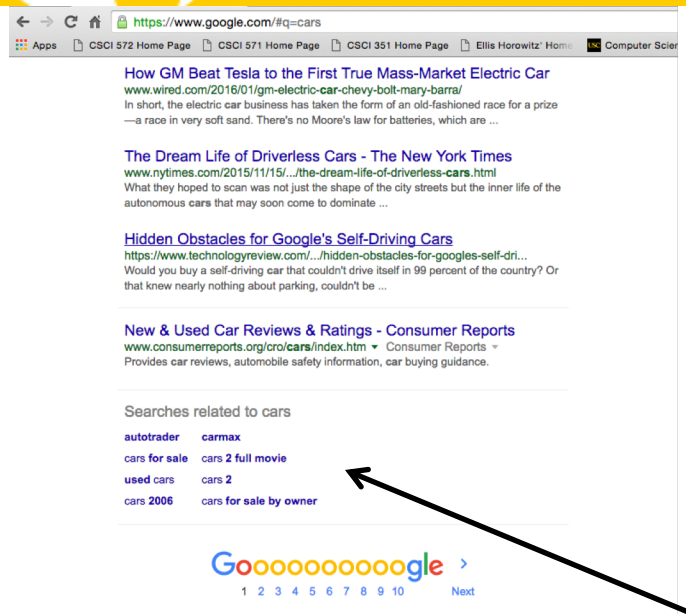
3. A hospital care chain wants to open a series of Emergency-Care wards, keeping in mind the factor of maximum accident prone areas in a region

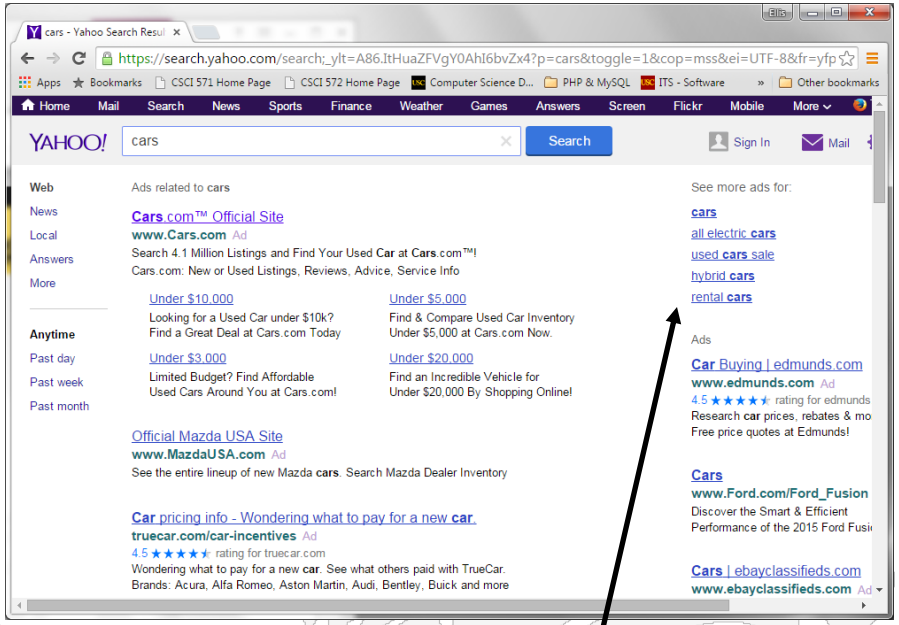
Why Search Engines Cluster Documents

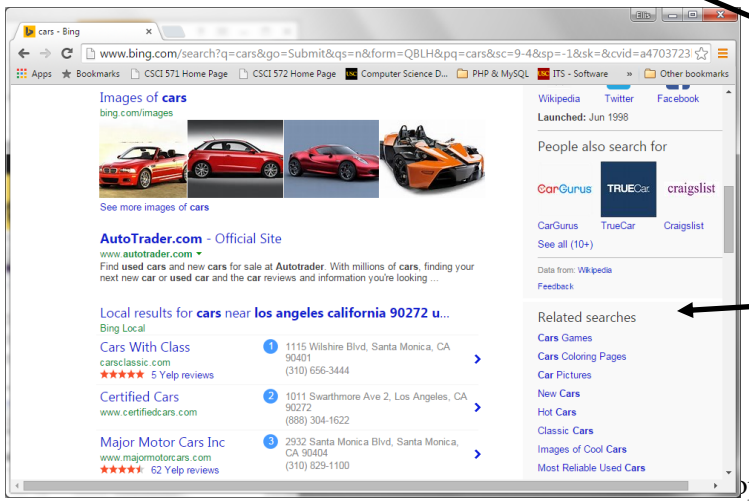
1. **For improving recall in search applications**
   - Better search results; similar documents are grouped
2. **For speeding up vector space retrieval**
   - Faster search if clustering occurs a priori
3. **Cleaner user interface**
4. **Automatic thesaurus generation by clustering related terms**

- ***Cluster hypothesis* - Documents with similar text are related**
- **Ergo, to improve search recall:**
  - We could cluster docs in our corpus a priori
  - When a query matches a doc *D*, also return other docs in the cluster containing *D*

Related Searches are a Form of Clustering







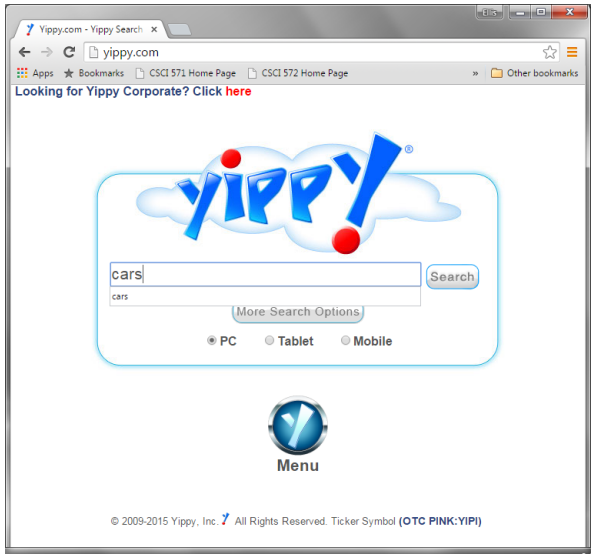
Google related searches

Yahoo does some clustering via alternate queries

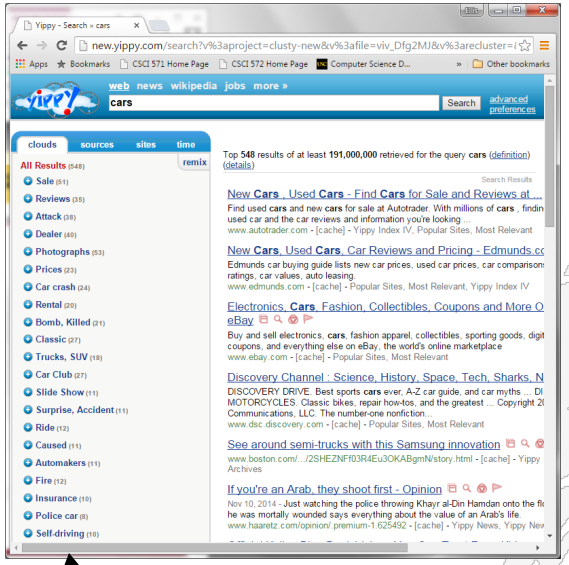
Bing does a little better

yippy.com Search Engine

- **Yippy** (formerly **Clusty**) is a metasearch engine developed by Vivísimo which emphasizes clusters of results.



initial screen
with query "cars"

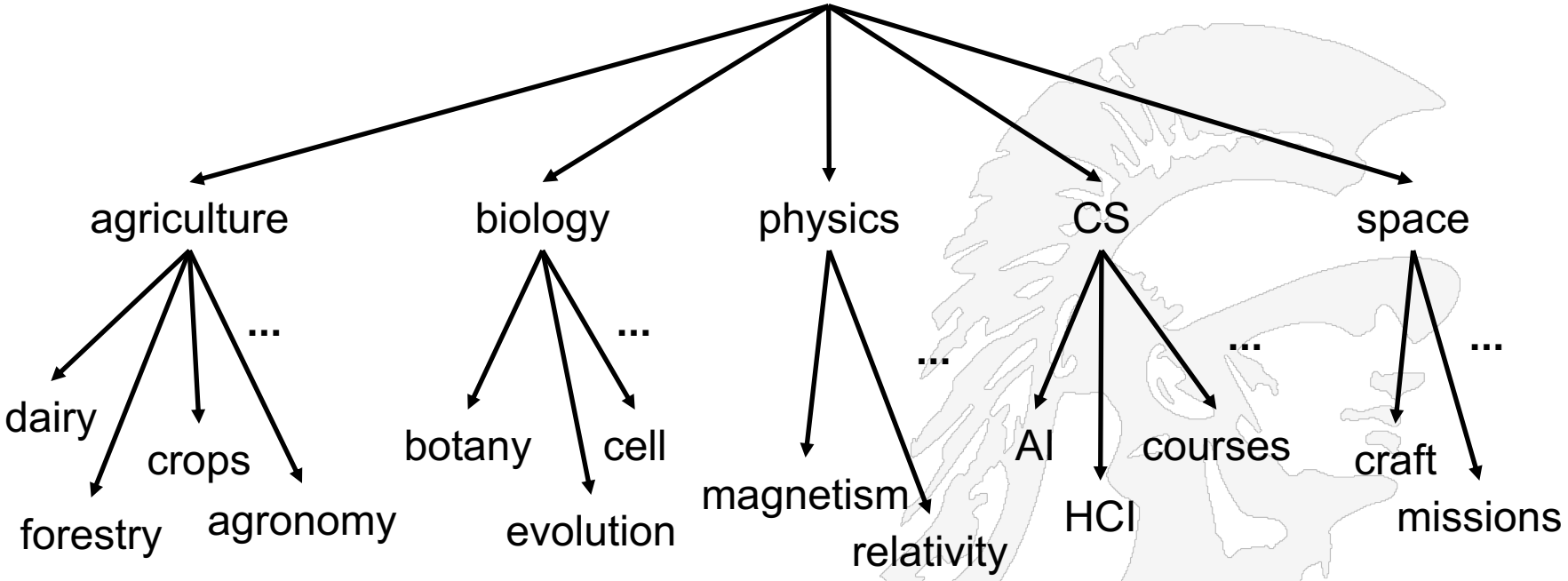


clustered results appear
on the left column: e.g.
sale
reviews
dealers
rentals

multiple level clusters:
car dealers
trucks
ebay

Yet Another Hierarchical Officious Oracle Acronym for Yahoo

Yahoo! Hierarchy *isn't* clustering but *is* the kind of output you want from clustering



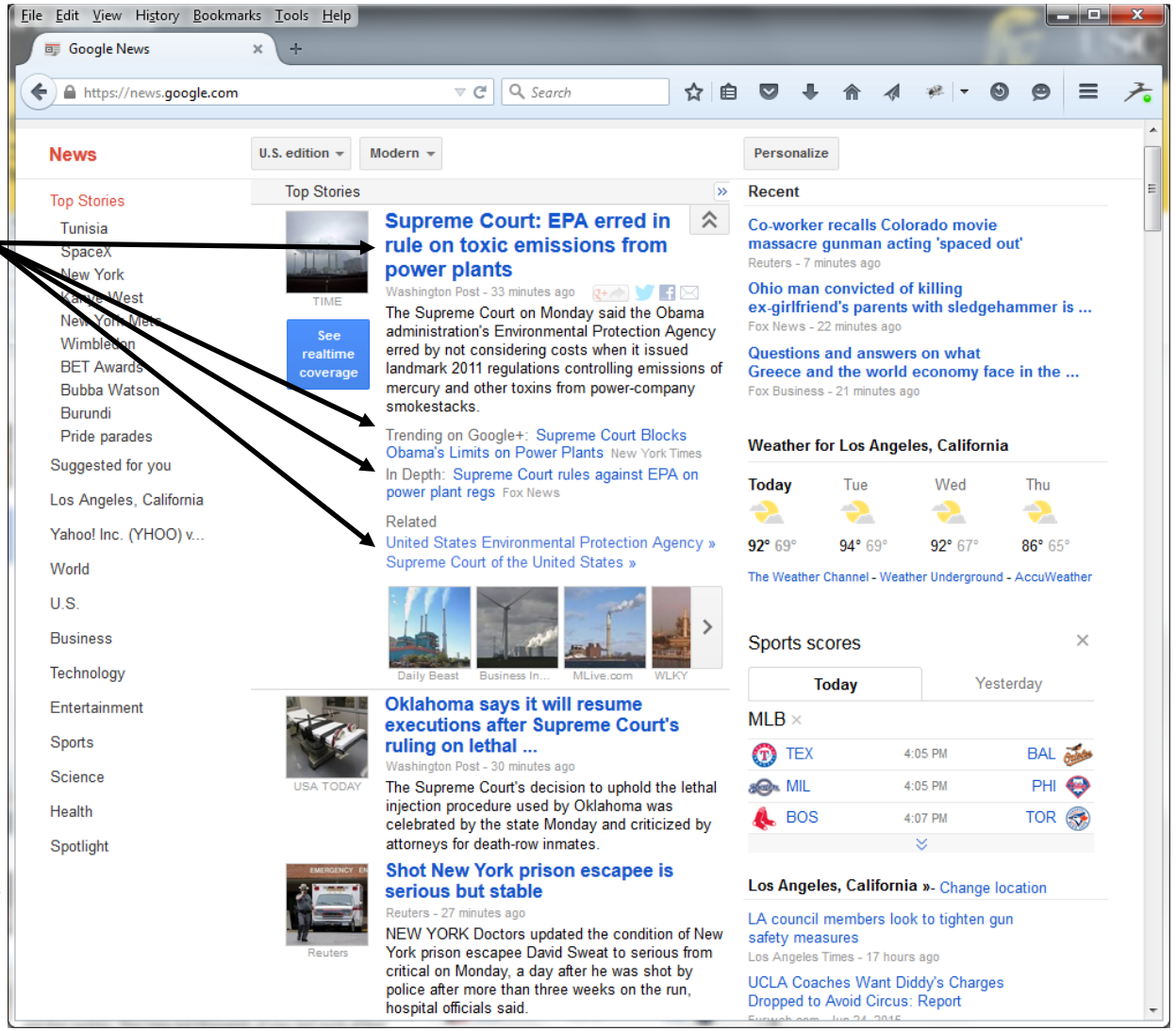
See www.yahoo.com/Science, or
https://searchengineland.com/yahoo-directory-close-204370

Google News: Automatic Clustering Gives an Effective News Presentation Metaphor

recent Supreme court decisions clustered together

Newspaper clusters: World, US, Business, Technology, Sports, etc

These clusters must be constantly re-computed to make sure the latest news is included

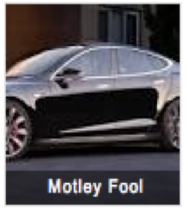


Clustering Examples from Google RSS Feeds

Two examples of Google feeds
There is a main article, some text and beneath that related or clustered articles

Tesla's Model 3 Market Opportunity Is Bigger Than You Think

Motley Fool



Tesla's (NASDAQ:TSLA) forthcoming Model 3 will be unveiled next month, go into production in late 2017, cost about $35,000 before incentives, and ...

Tesla Signs Lease for 40K-SF Red Hook Dealership - Commercial Observer
Advertising enters the equation for **Tesla** Motors - Seeking Alpha
Tesla Motors Finally Gets Its Paws on **Tesla**.com - Inverse
Full Coverage

There's one new **Tesla** car that nobody is talking about

Businessinsider India

These two **Teslas** are all anyone has been talking about lately - especially Wall Street analysts who want to figure out which way **Tesla's** extremely ...

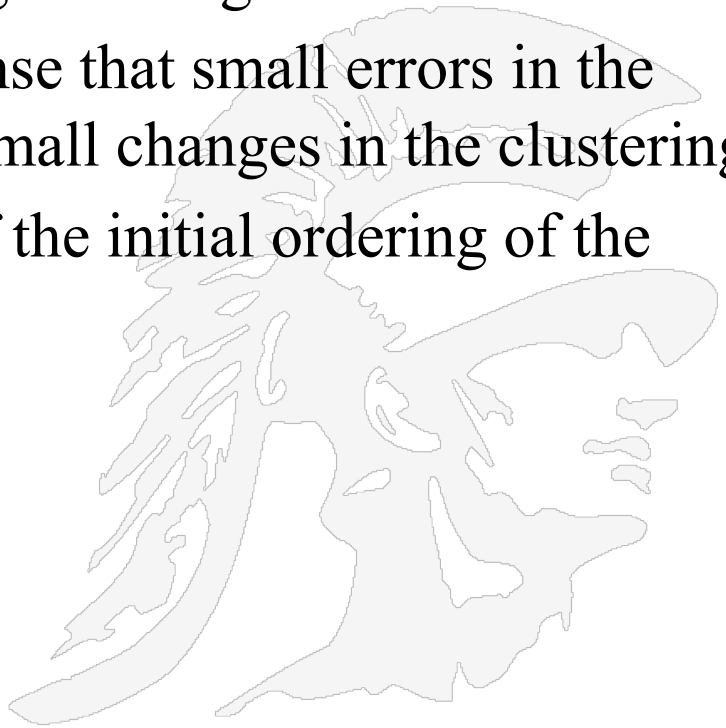
VIDEO: **Tesla** Drag Race! Model S vs. Model X In STUNNING Showdown - AutoSpies.com
Tesla Model S & Model X Comparison (Price, Range, Acceleration) After Removal Of 85 kWh Version - InsideEVs
Tesla Will Begin Taking Preorders on Its Make-or-Break Vehicle - GreatNews
Full Coverage

What Is A Good Clustering?

- **Internal criterion: A good clustering will produce high quality clusters in which:**
  - the **intra-class** (that is, intra-cluster) similarity is high
  - the **inter-class** similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used

Three Criteria of Adequacy for Clustering Methods

1. The method produces a clustering which is unlikely to be altered drastically when further objects are incorporated
   - i.e. it is stable even under significant growth
2. The method is **stable** in the sense that small errors in the description of objects lead to small changes in the clustering
3. The method is **independent** of the initial ordering of the objects

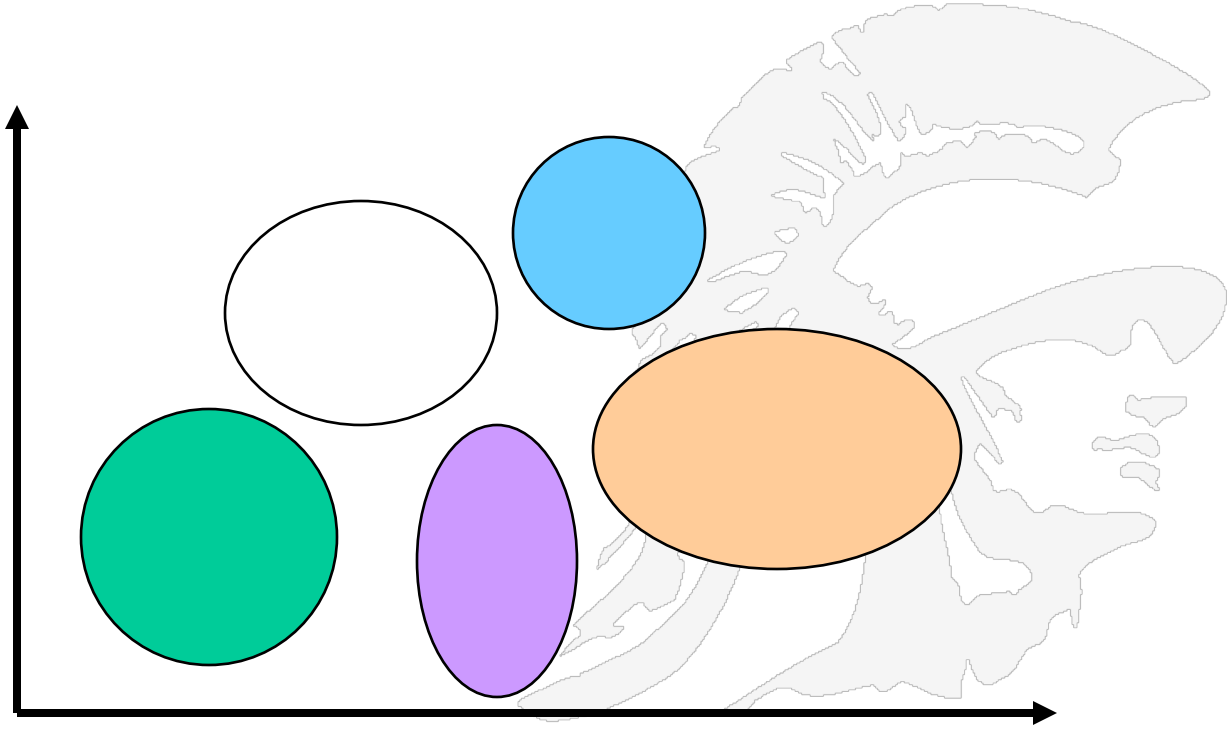


Classification is Different from Clustering

- In general, in ***classification*** you have a set of predefined classes and want to know which class a new object belongs to.
- ***Clustering*** tries to group a set of objects and find whether there is *some* relationship between the objects.
  - Clustering *precedes* classification
- In the context of machine learning, classification is *supervised learning* and clustering is *unsupervised learning*
  - **Clustering** requires a. an algorithm, b. a similarity measure, and c. a number of clusters
  - **classification** has each document labeled in a class and an algorithm that assigns new documents to one of the classes

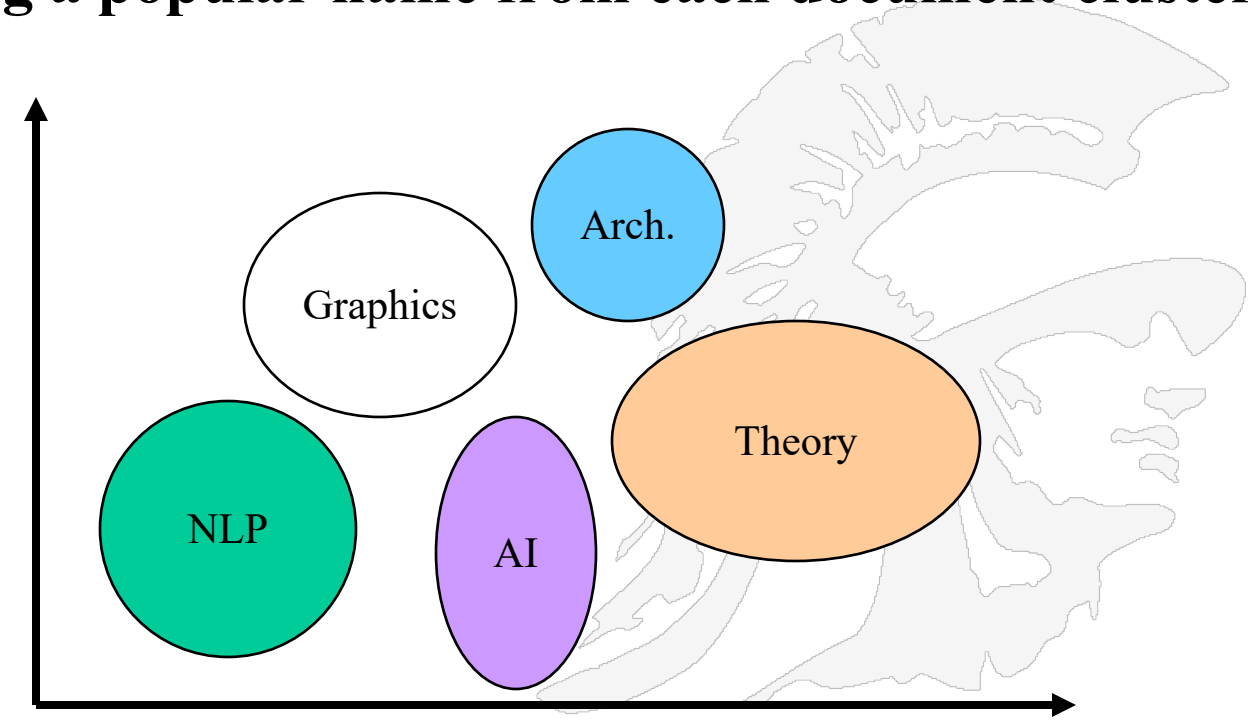
Clustering vs Classification

- **Step 1: Given a large set of computer science documents, first we cluster them**



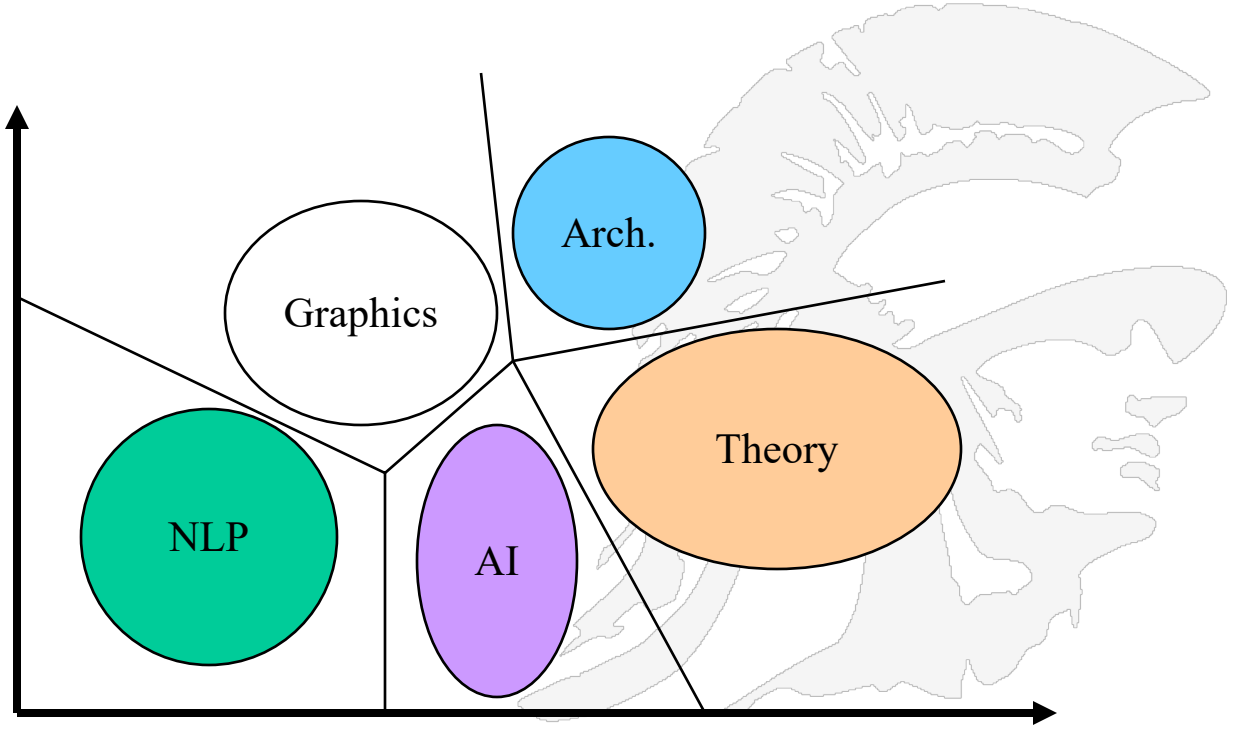
Still Clustering

- **Step 2: we label the clusters**
  - **choosing a popular name from each document cluster**



Still Clustering Decision boundaries

- **Step 3: we compute boundaries for the clusters that can be used as new documents appear; i.e. classification**



Classification is Supervised Learning

- **Definition: Supervised Learning, inferring a function from labeled training data**

1. **The documents in each cluster define the "training" docs for each category**
   - E.g in computer science named clusters would include: Algorithms, Theory, AI, Databases, Operating Systems, NLP, etc.
2. **Documents are in a cluster based upon the similarity measure used;**
   - generally a vector space with each doc viewed as a bag of words
3. **A classifier is an algorithm that will classify new docs**
   - Essentially, partition the decision space
4. **Given a new doc, figure out which partition it falls into**

Now Let's Return to the Earlier Problem: Clustering

- **Questions to consider when clustering**
  - How do we represent the document?
    - *Usually as a vector space*
  - How do we compute similarity/distance?
    - *Using cosine similarity*
  - How many clusters?
    - *will it be a fixed a priori number? or*
    - *completely data driven?*
  - Be careful to avoid "trivial" clusters - too large or small
    - *If a cluster is too large, then for navigation purposes you've wasted an extra user click without whittling down the set of documents much*

Issue: Hard vs. Soft Clustering

- ***Hard clustering*: Each document belongs to exactly one cluster**
  - More common and easier to do
- ***Soft clustering*: A document can belong to more than one cluster.**
  - Makes more sense for applications like creating browsable hierarchies; e.g. divide-and-conquer belongs in several clusters
  - E.g. you may want to put a pair of sneakers in two clusters: (i) sports apparel and (ii) shoes

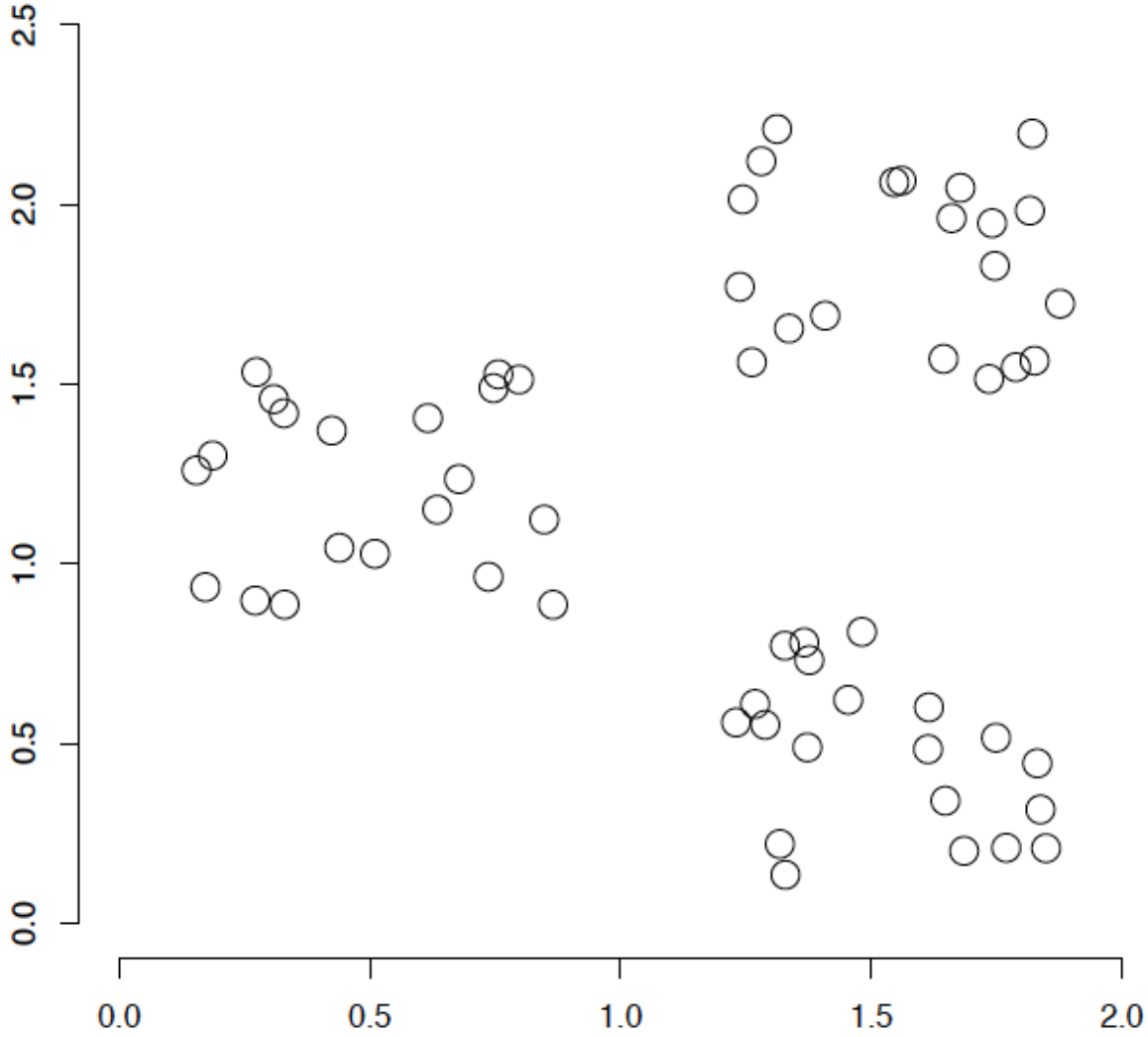
What Definition of Similarity/Distance Will Be Used

- **Typically one treats documents as vectors**
  - **Cosine similarity (seen before many times)**
    - Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. Range from -1 (dissimilar) to 1 exactly similar

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

  - **Most clustering implementations use cosine similarity**
  - **Euclidean distance is a close alternative that is also popular**

A Data Set with Clear Cluster Structure



- How would you design an algorithm for finding the three clusters in this case?

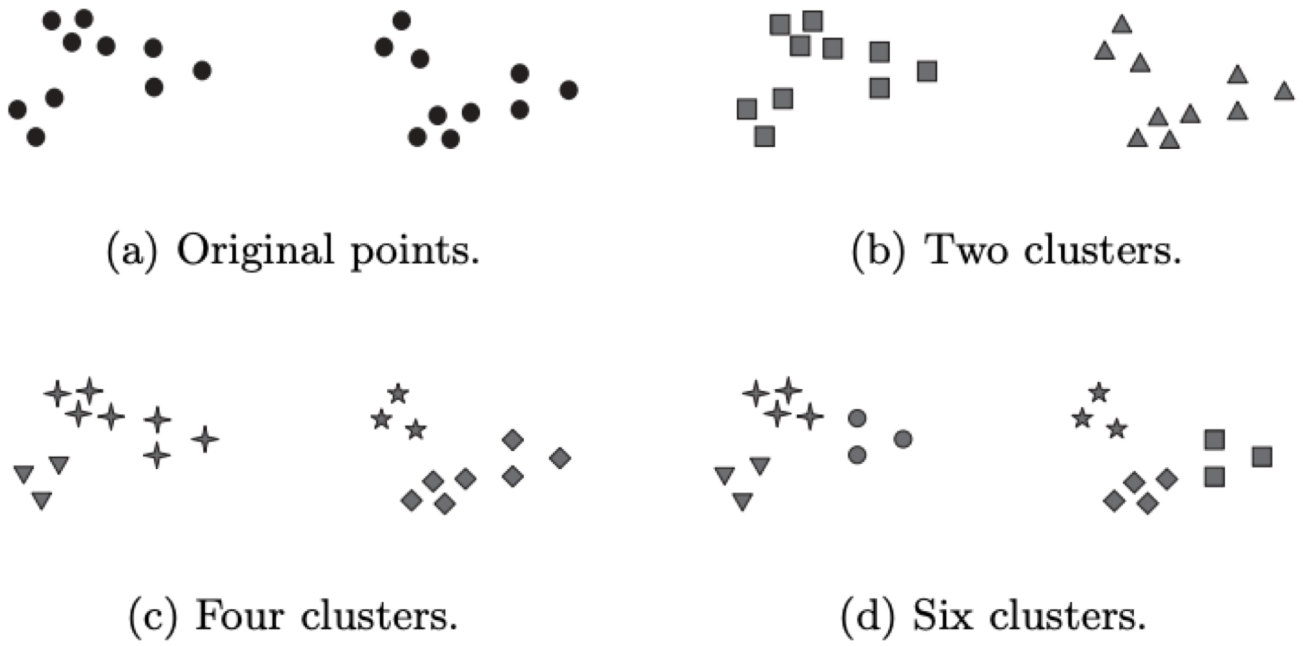
Clustering Algorithms

- **Two general methodologies**
  - Partitioning Based Algorithms
  - Hierarchical Algorithms
- **Partitioning Based**
  - divide a set of N items into K clusters (top-down)
- **Hierarchical**
  - agglomerative: pairs of items or clusters are successively linked to produce larger clusters
  - divisive: start with the whole set as a cluster and successively divide sets into smaller partitions

K-Means Clustering Algorithm

- **Clustering algorithm strategy**
  - Choose *k* random data items out of the *n* items; call these items the *means*; they designate the prototype or name of the cluster
  - **Refine it iteratively**
    - Associate each of the *n-k* items with one of the ***k* clusters** choosing the **cluster** that it is nearest to**;**
    - **This is called *K*-means clustering**
- **Recall**
  - The "***mean***" is the "average" where you add up all the numbers and then divide by the number of numbers.
  - The "***median***" is the "middle" value in the list of numbers. To find the median, you may have to sort
  - The "***mode***" is the value that occurs most often. If no number is repeated, then there is no mode for the list

Different Ways of Clustering the Same Set of Points



(a) Original points.

(b) Two clusters.

(c) Four clusters.

(d) Six clusters.

K-means clustering critically depends upon the value of k

"Optimal" K-Means Clustering

- The **optimal *k*-means clustering problem** calls for finding cluster centers that minimize the intra-class variance, i.e. the sum of squared distances from each data point being clustered to its cluster center;
- **The problem stated formally**:
  - Given a finite set *S* where each element is a vector of length *d*, find a subset *T* of size *k* that minimizes the sum of squares of the distances between elements in *S* and their closest element in *T*
- Finding an exact solution to the *k*-means problem for arbitrary input has been shown to be **NP-hard**
- **NP**-hardness (non-deterministic polynomial-time **hard**), in computational complexity theory, is a class of problems that are, informally, "at least as **hard** as the hardest problems in **NP**".
- finding a polynomial algorithm to solve any NP-hard problem would give polynomial algorithms for all the problems in NP, which is unlikely

A Popular Version of the K-Means Clustering Algorithm

Given an initial set of k means $m_1^{(1)},\ldots,m_k^{(1)}$, the algorithm proceeds by alternating between two steps:

Assignment step: Assign each observation to the cluster whose mean yields the least within-cluster sum of squares. Since the sum of squares is the squared Euclidean distance, this is intuitively the "nearest" mean

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \ \forall j, 1 \leq j \leq k\},$$

where each x_p is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them.

Update step: Calculate the new means to be the centroids of the observations in the new clusters.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

- The algorithm has converged when the assignments no longer change.
- The algorithm will converge to a (local) optimum.
- There is no guarantee that the global optimum is found using this algorithm.

The Previous Slide Expressed As An Algorithm

1. **Select K points as initial centroids**
2. **repeat**
   - form K clusters by assigning each point to its closest centroid
   - re-compute the centroid of each cluster
3. **until centroids do not change**

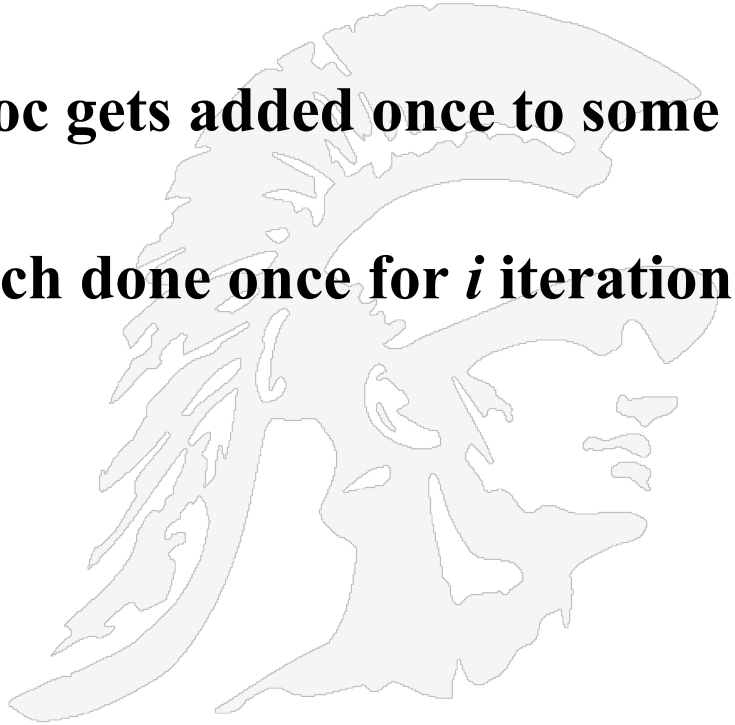
- **the algorithm will always terminate, however it does not always find the optimal solution**
- **this is an example of a greedy algorithm**

Some Adjustments to the Algorithm

- **How to pick the initial cluster means points**
  - **Multiple runs**
    - Choose different random points and see which yields the best result
  - **Select original set of points by methods other than random**
    - E.g., pick the most distant (from each other) points as cluster centers (this is called the *k-means*++ algorithm)
- **For termination conditions there are several possibilities, e.g.,**
  - After a fixed number of iterations
  - When the document partition is unchanged
  - When the centroid positions don't change

Time Complexity

- **Computing distance between two docs is O*(m)* where *m* is the dimensionality of the vectors**
- **Re-assigning clusters: *O(kn)* distance computations, or *O(knm)***
- **Computing centroids: Each doc gets added once to some centroid: *O(nm)***
- **Assume these two steps are each done once for *i* iterations: *O(iknm)***
- ***Note:***
- ***m is the size of the vector***
- ***n is the number of vectors (items)***
- ***k is the number of clusters***
- ***i depends upon convergence***



K-means Clustering – Summary Details

- **Initial centroids are often chosen randomly**
  - Clusters produced vary from one run to another
- **The centroid is (typically) the mean of the points in the cluster**
- **'Closeness' is measured by cosine similarity, a variation of Euclidean distance**
- **Most of the convergence happens in the first few iterations.**
  - Often the stopping condition is changed to 'Until relatively few points change clusters
- **Complexity is *O(i * k * n * m)***
  - n = number of points, k = number of clusters, i = number of iterations, m = number of attributes

Hierarchical Clustering Algorithms

- **Two main types of hierarchical clustering**
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- **Traditional hierarchical algorithms use a similarity or distance matrix**
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm - a Bottom Up Approach

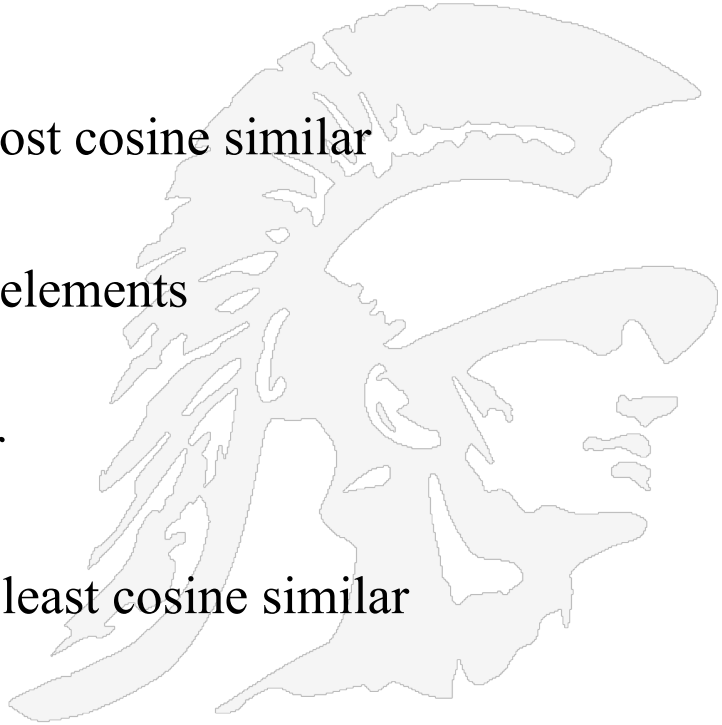
- **Basic Agglomerative Clustering Algorithm**
  1. Compute the distance matrix between the input data points (i.e. the distance between all pairs of points)
  2. Let each data point be a cluster unto itself
  3. Repeat
  4. Merge the two closest clusters
  5. Update the distance matrix
  6. Until only a single cluster remains

- **Key operation is the computation of the distance between two clusters**
  - Different definitions of the distance between clusters lead to somewhat different algorithms

How Can We Compute the Distance Between Two Clusters

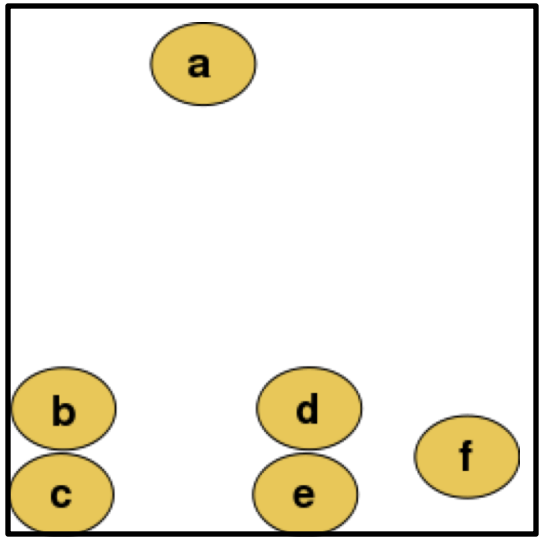
- As before, the **Centroid** of a cluster is the component-wise average of the vectors in a cluster, which is itself a vector
- Example, the Centroid of (1,2,3); (4,5,6); (7,2,6); is (4,3,5)
- **4 possible ways**

1. **Center of Gravity**
   - clusters whose centroids are the most cosine similar
2. **Average Link**
   - average distance between pairs of elements
3. **Single Link**
   - distance of the most cosine similar
4. **Complete Link**
   - distance of the furthest points, the least cosine similar



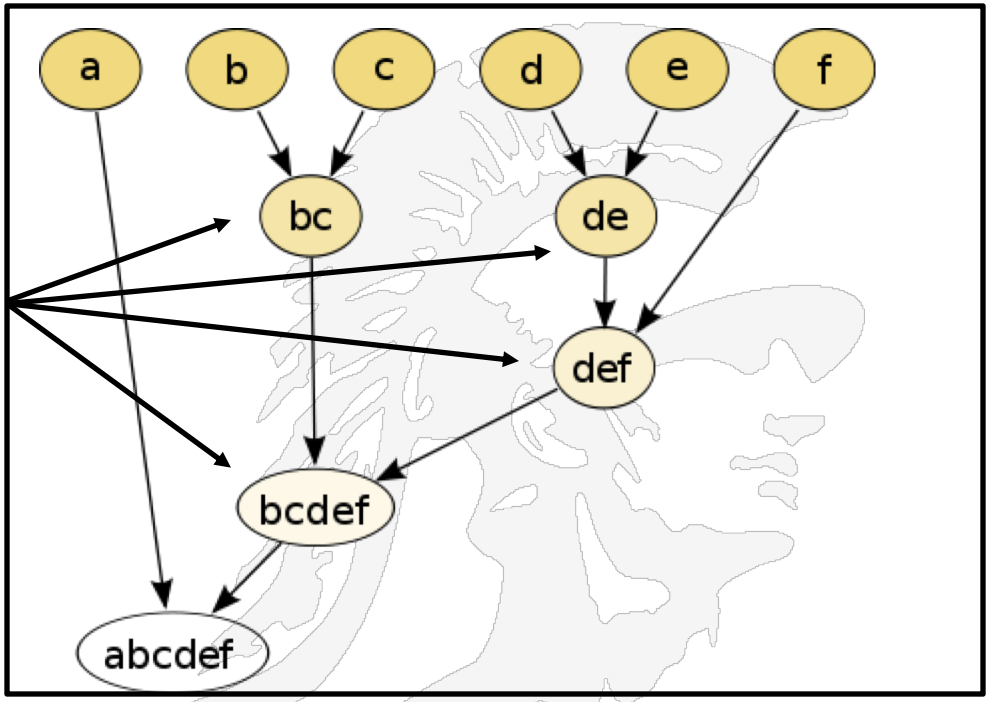
A Dendrogram is Used to Display Clusters

- A **dendrogram** is a tree diagram frequently used to illustrate the arrangement of the clusters produced by hierarchical clustering



original input

clusters

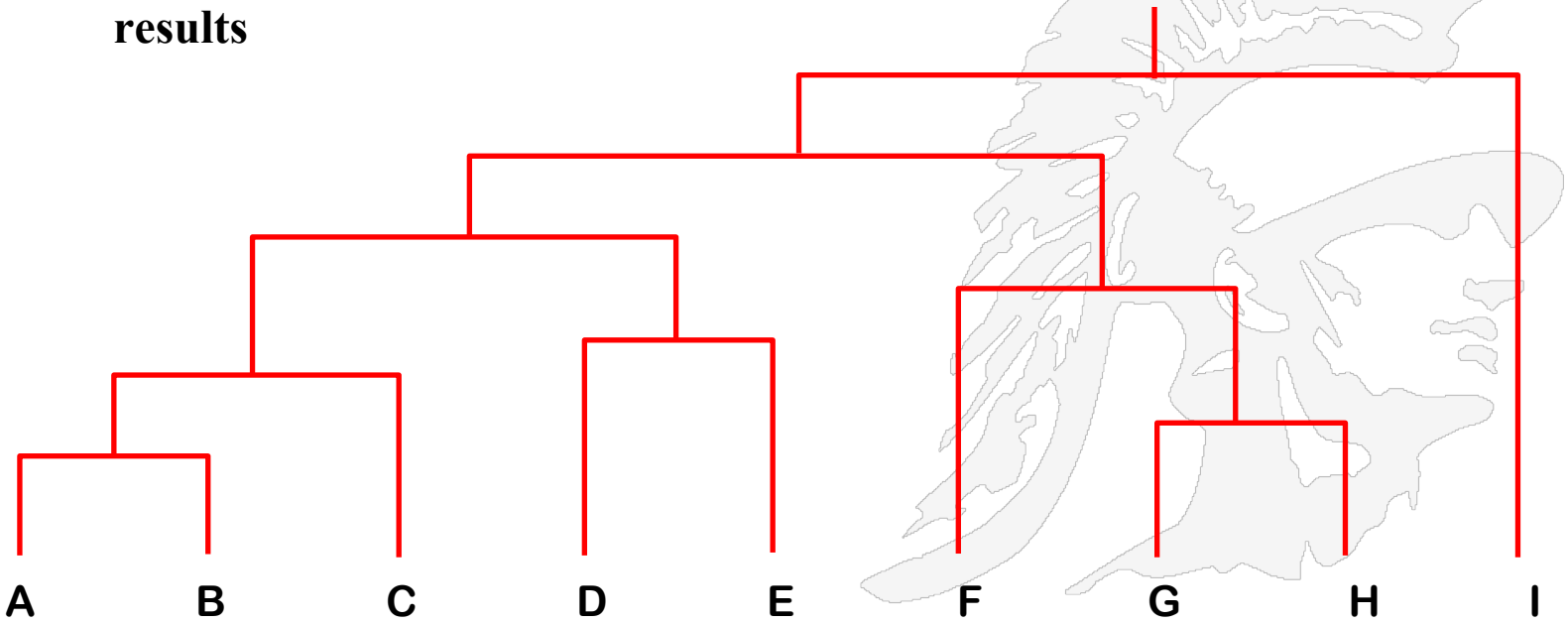


corresponding dendrogram

second row clusters are: {a}, {b c}, {d e} {f}
third row clusters are: {a}, {b c} {d e f}

Hierarchical Agglomerative Clustering

- HAC starts with unclustered data and performs successive pairwise joins among items (or previous clusters) to form larger ones
  - this results in a hierarchy of clusters which can be viewed as a dendrogram
  - Dendrograms are usually drawn as shown below
  - The height of an edge can sometimes refer to the degree of similarity
  - useful in pruning search in a clustered item set, or in browsing clustering results

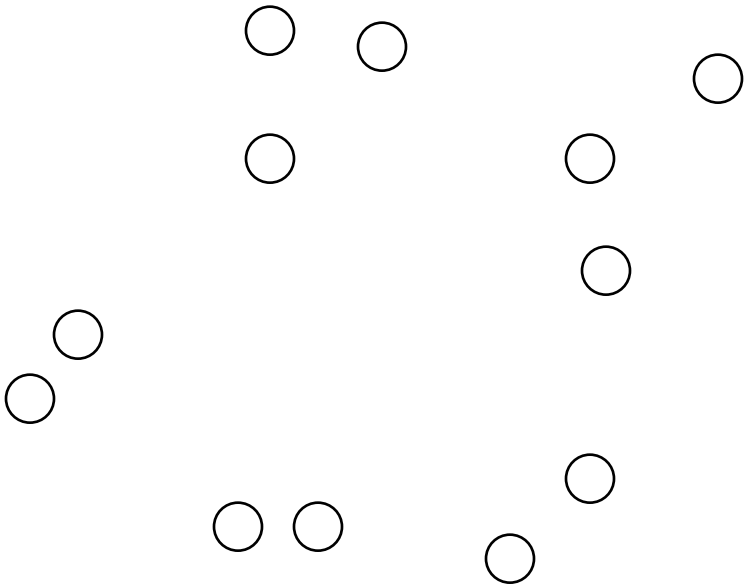


Hierarchical Agglomerative Clustering

- **Basic procedure**
  - **1.** Place each of N documents into a class of its own.
  - 2. Compute all pairwise document-document similarity coefficients
    - ***Total of N(N-1)/2 coefficients***
  - **3. Form a new cluster by combining the most similar pair of current clusters *i* and *j***
    - (use one of the methods described in a previous slide, e.g., complete link, etc.);
    - update similarity matrix by deleting the rows and columns corresponding to *i* and *j*;
    - calculate the entries in the row corresponding to the new cluster *i+j*.
  - **4. Repeat step 3 if the number of clusters left is great than 1.**

Input/ Initial setting

- **Start with clusters of individual points and a distance/proximity matrix**

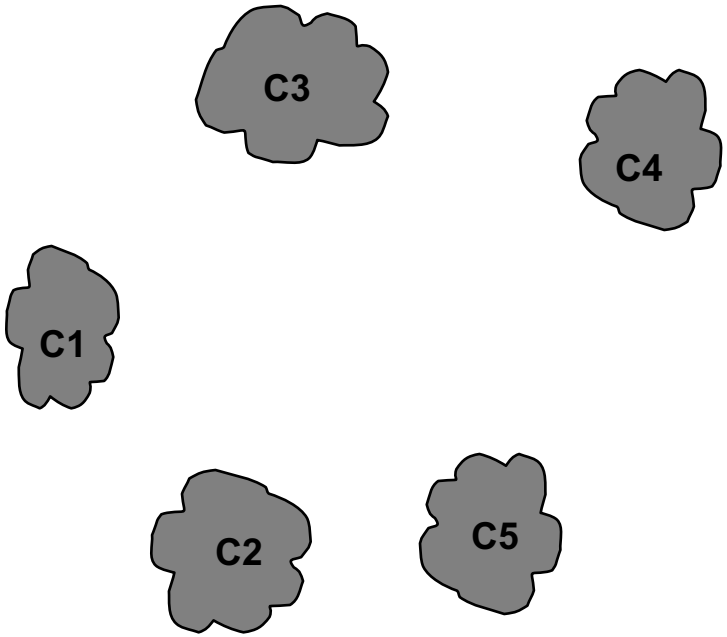


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						

Distance/Proximity Matrix

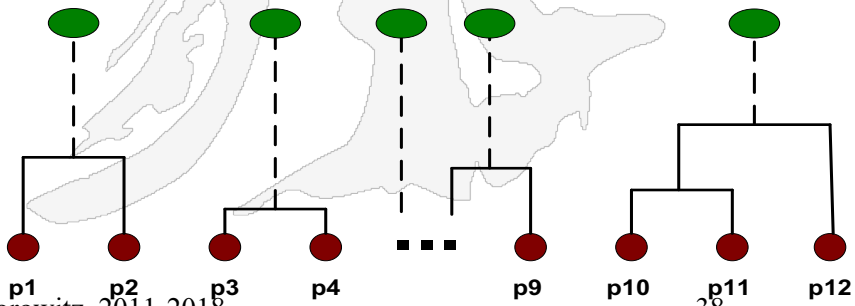
Intermediate State

- After some merging steps, we have some clusters



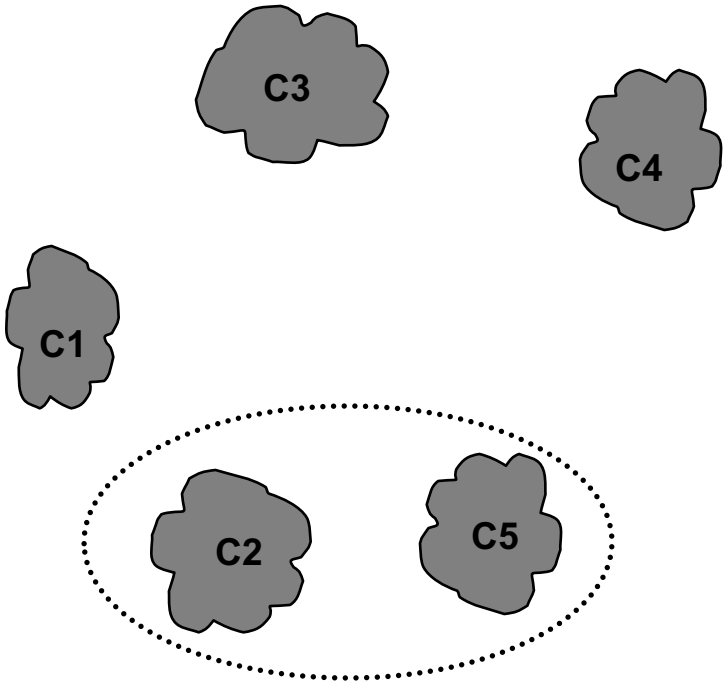
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix (above)
Dendrogram (below)



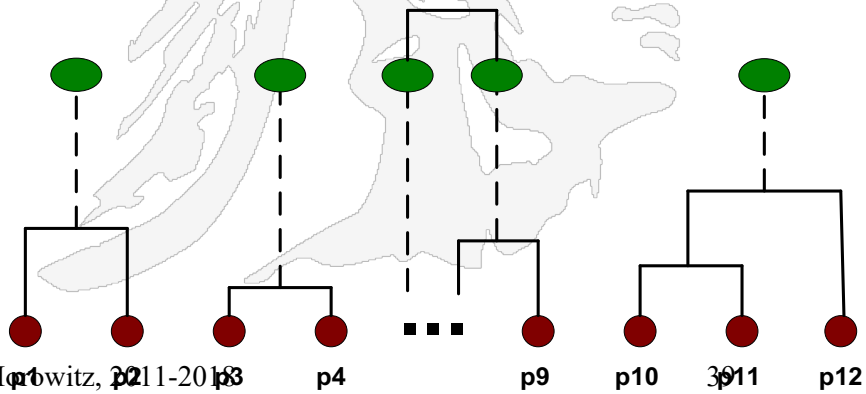
Intermediate State

- **Merge the two closest clusters (C2 and C5) and update the distance matrix.**



	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix



General Hierarchical Agglomerative Clustering Algorithm and Complexity

1. **Compute similarity between all pairs of documents** — $O(N^2)$
2. **Do N – 1 times**
   1. **Find closest pair of documents/clusters to merge**
      - Naïve: $O(N^2)$
      - Priority Queue: $O(N)$
      - Single link: $O(N)$
   1. **Update similarity of all documents/clusters to new cluster**
      - Naïve: $O(N)$
      - Priority Queue: $O(N \log N)$
      - Single link: $O(N)$

Divisive Clustering Algorithm

- Start at the top with all documents in one cluster.
- The cluster is split using a flat clustering algorithm.
  - Use the k-means clustering algorithm, which is linear in computing time whereas HAC algorithms are quadratic
- This procedure is applied recursively until each document is in its own singleton cluster
- Studies shown that the divisive algorithms produce more accurate hierarchies than bottom up
  - Bottom-up methods make clustering decisions based on local patterns without initially taking into account the global distribution. These early decisions cannot be undone.
  - Top-down clustering benefits from complete information about the global distribution when making top-level partitioning decisions.

How to Label Clusters Two Approaches

1. **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
2. **Show words/phrases prominent in cluster**
   - Use distinguishing words/phrases
   - But harder to scan

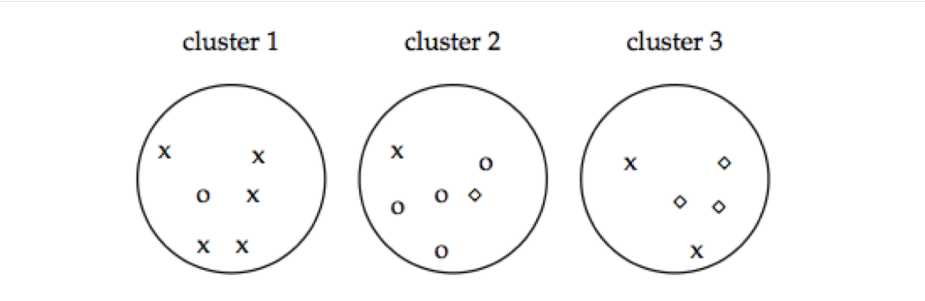
- **Common heuristics - list 5-10 most frequent terms in the centroid vector**
  - Drop stop-words;

Approaches to Evaluating Clustering Algorithms (from the weakest to the best)

- **Anecdotal**
  - Probably the commonest (and surely the easiest)
    - "I wrote this clustering algorithm and look what it found!"
  - No benchmarks, no comparison possible
- **User inspection**
  - Have subject matter experts evaluate the results and score them
  - Not clear how reproducible across tests.
  - Expensive / time-consuming
- **Ground "truth" comparison**
  - Take a union of docs from a taxonomy
  - Use Yahoo!, ODP (open directory project), newspaper sections …
  - Compare clustering results to results from prior organizations
  - e.g., this would be a good result
    - 80% of the clusters found to map "cleanly" to taxonomy nodes
- **Purely quantitative measures**
    - Compute the average distance between cluster members

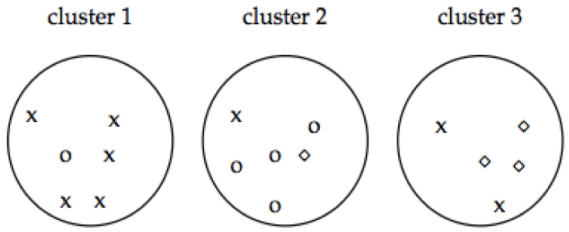
Purity and Rand Index for Cluster Evaluation

consider the three clusters,
cluster 1 has 5 related items;
cluster 2 has 4 related items and
cluster 3 has 3 related items;



- **Purity Measure - accuracy is measured by the number of correctly assigned documents divided by the total number of documents;**
- **Purity = (1/17) * (5 + 4 + 3) ~= 0.71**
- **Rand index**: We want to assign two documents to the same cluster if and only if they are similar.
  - A true positive (TP) decision assigns two similar documents to the same cluster,
  - a true negative (TN) decision assigns two dissimilar documents to different clusters.
  - There are two types of errors we can commit.
    - A false positive (FP) decision assigns two dissimilar documents to the same cluster.
    - A false negative (FN) decision assigns two similar documents to different clusters.
- The *Rand index* (RI) measures the percentage of decisions that are correct.

Computing the Rand Index



$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

As an example, we compute RI for Figure 16.4. We first compute TP + FP. The three clusters contain 6, 6, and 5 points, respectively, so the total number of "positives" or pairs of documents that are in the same cluster is:

$$\mathrm{TP} + \mathrm{FP} = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$$

Of these, the x pairs in cluster 1, the o pairs in cluster 2, the ⋄ pairs in cluster 3, and the x pair in cluster 3 are true positives:

$$\mathrm{TP} = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20$$

Thus, FP = 40 − 20 = 20.

FN and TN are computed similarly, resulting in the following contingency table:

	Same cluster	Different clusters
Same class	TP = 20	FN = 24
Different classes	FP = 20	TN = 72

RI is then $(20 + 72)/(20 + 20 + 24 + 72) \approx 0.68$.

See textbook for more details