Clustering

- Given:
  - a set of documents
  - no labels (→ unsupervised learning)

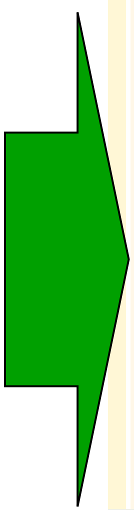
- Find:
  - a grouping of the examples into meaningful *clusters*
  - so that we have a **high**
    - **intra-class similarity**:
      - similarity between objects in same cluster
    - **inter-class dissimilarity**:
      - dissimilarity between objects in different clusters

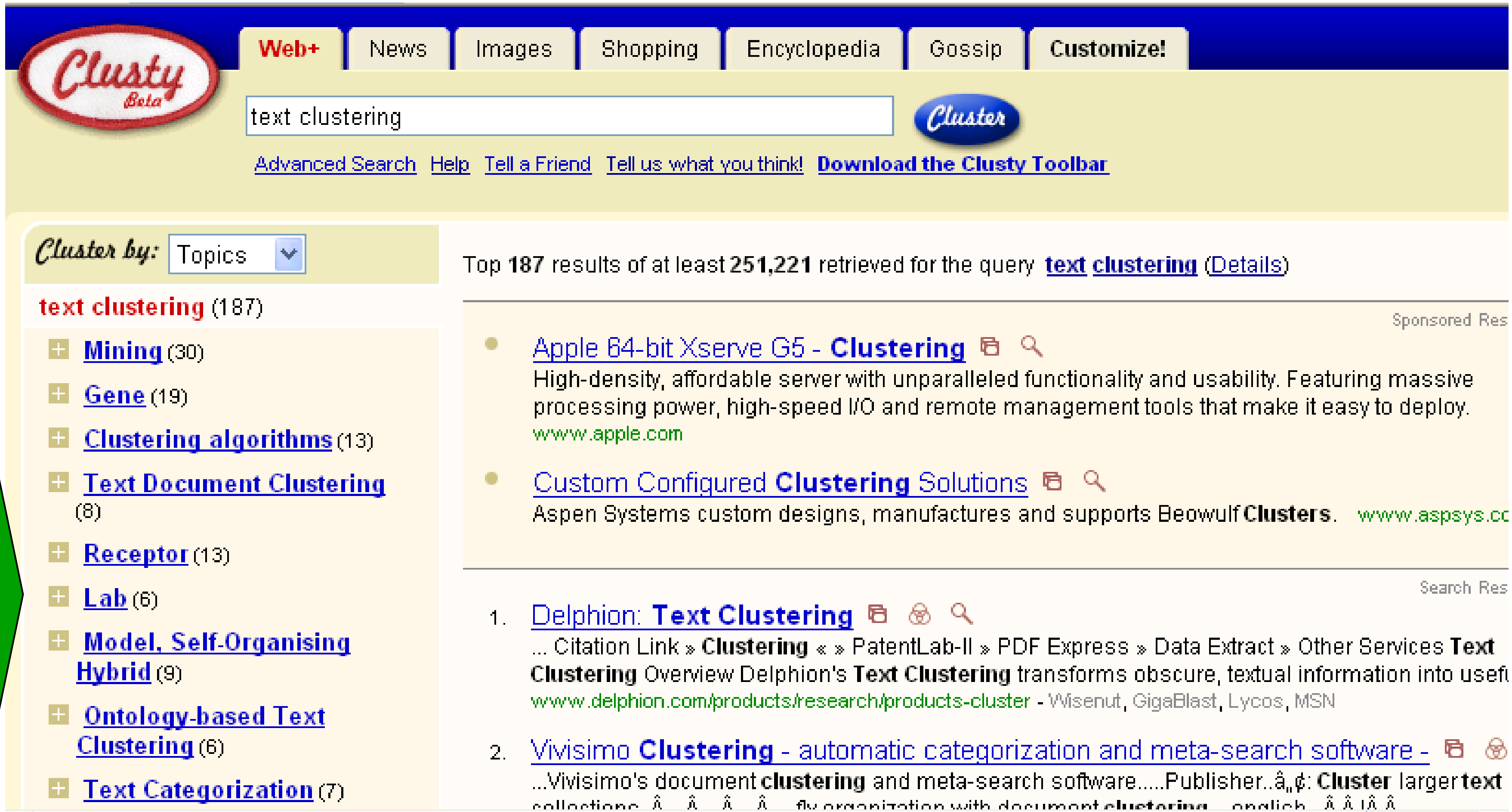
Some Applications of Clustering

- Query disambiguation
  - *Eg:* Query *“Star”* retrieves documents about *astronomy, plants, animals, movies* etc.
  - Solution:
    - Clustering document responses to queries
    - e.g., http://www.vivisimo.com/
- Manual construction of topic hierarchies and taxonomies
  - Solution:
    - Preliminary clustering of large samples of web documents.
- Speeding up similarity search
  - Solution:
    - Restrict the search for documents similar to a query to most representative cluster(s).

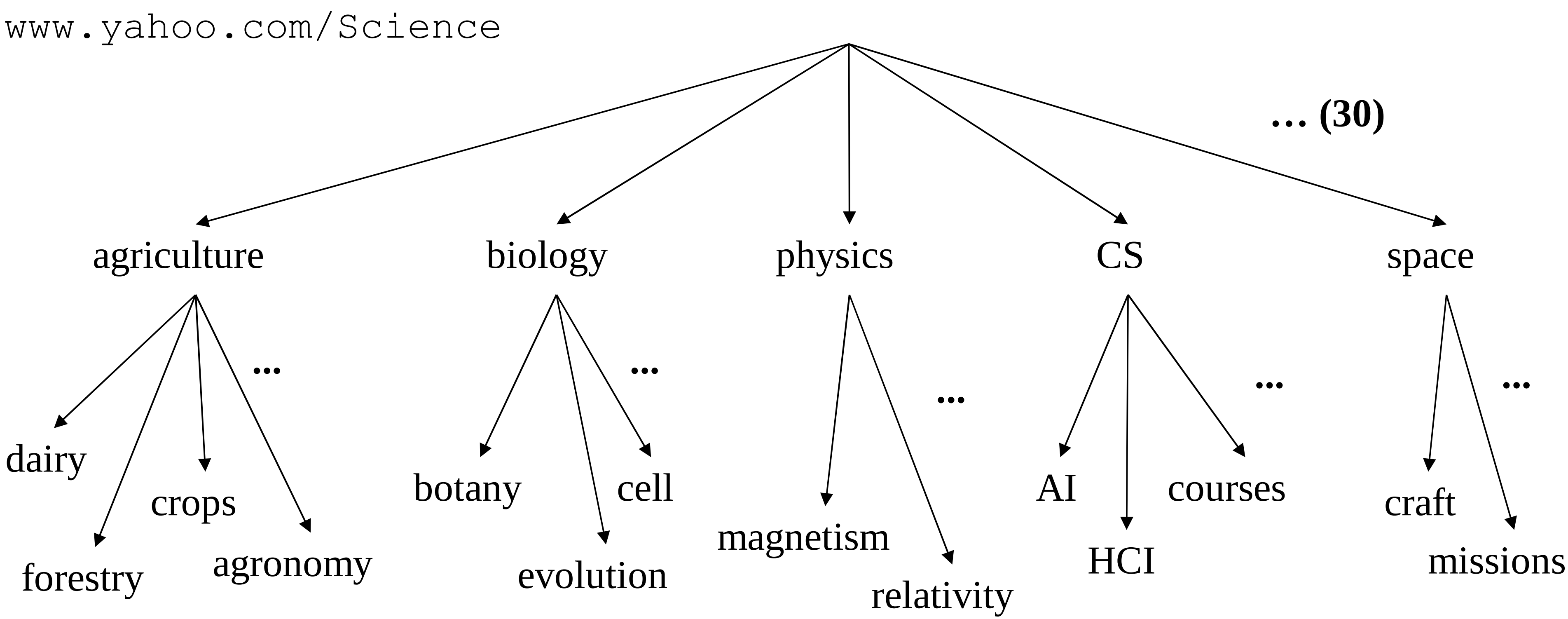
For better navigation of search results

- For grouping search results thematically
  - clusty.com / Vivisimo



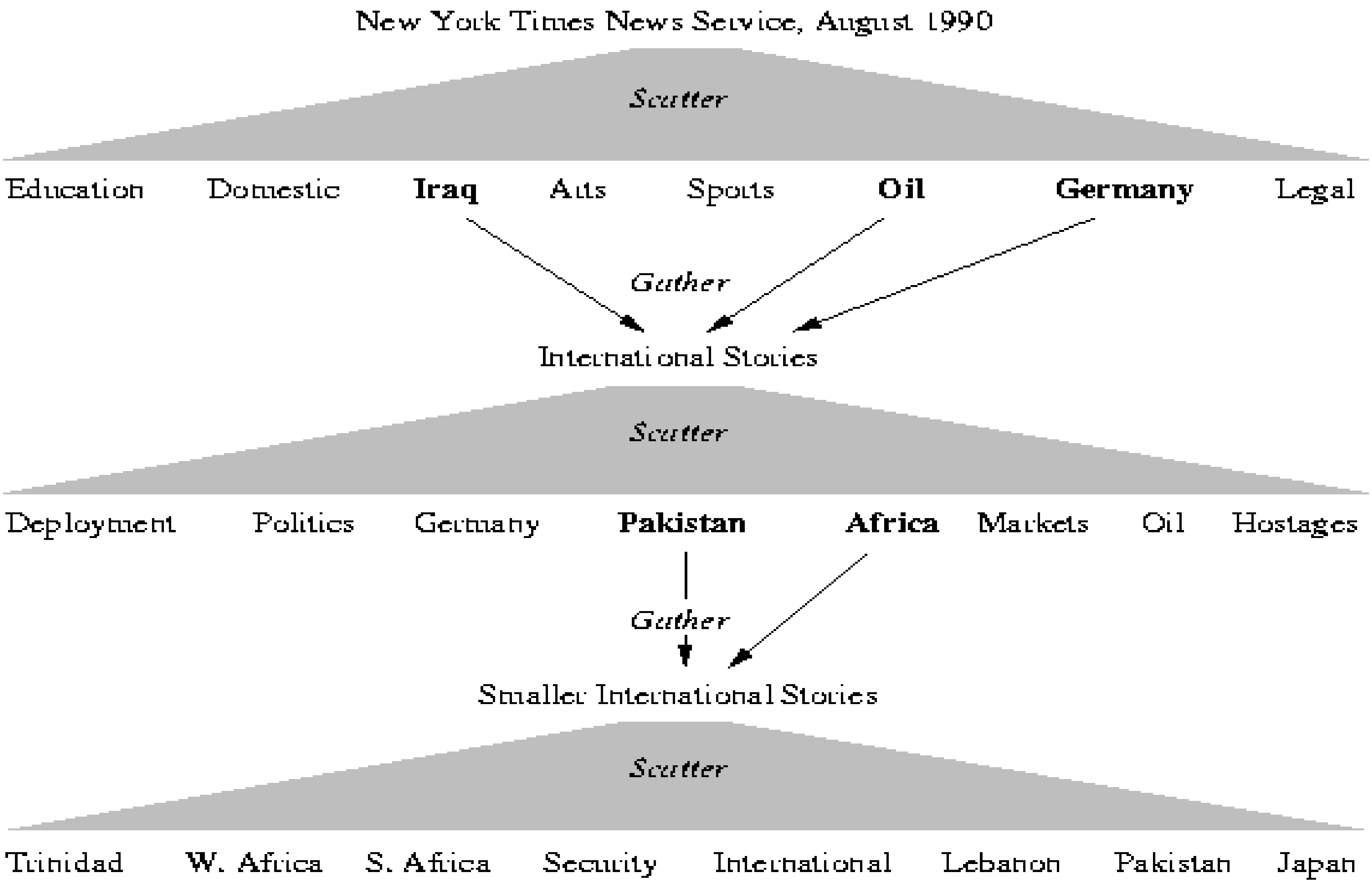


Application: Build up a Web Catalogue



Browsing Documents: Scatter/Gather

(Cutting, Karger, and Pedersen)



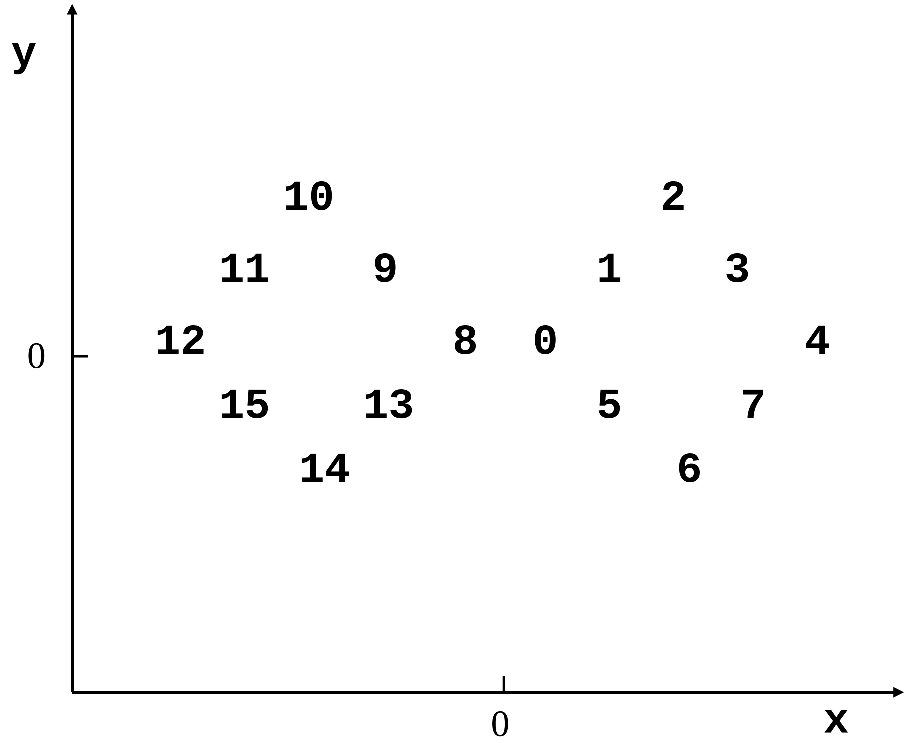
k-means Clustering

- Based on EM (Expectation Maximization) algorithm
- Efficiently find k clusters:

1. Randomly select k points as cluster centers
2. **E-Step:** Assign each example to the nearest cluster center
3. **M-Step:** Compute new cluster centers as the average of all points assigned to the cluster
4. Goto 2. unless no improvement

k-means: Example

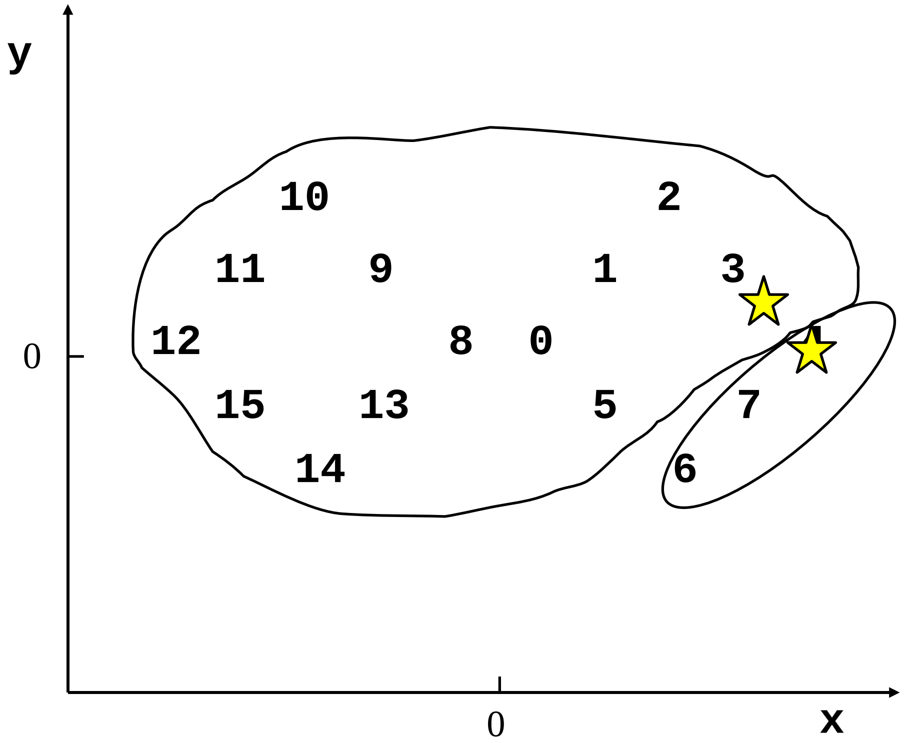
Id	x	y
0:	1.0	0.0
1:	3.0	2.0
2:	5.0	4.0
3:	7.0	2.0
4:	9.0	0.0
5:	3.0	-2.0
6:	5.0	-4.0
7:	7.0	-2.0
8:	-1.0	0.0
9:	-3.0	2.0
10:	-5.0	4.0
11:	-7.0	2.0
12:	-9.0	0.0
13:	-3.0	-2.0
14:	-5.0	-4.0
15:	-7.0	-2.0



- find the best 2 clusters

Seed: (9 0) (8 1)

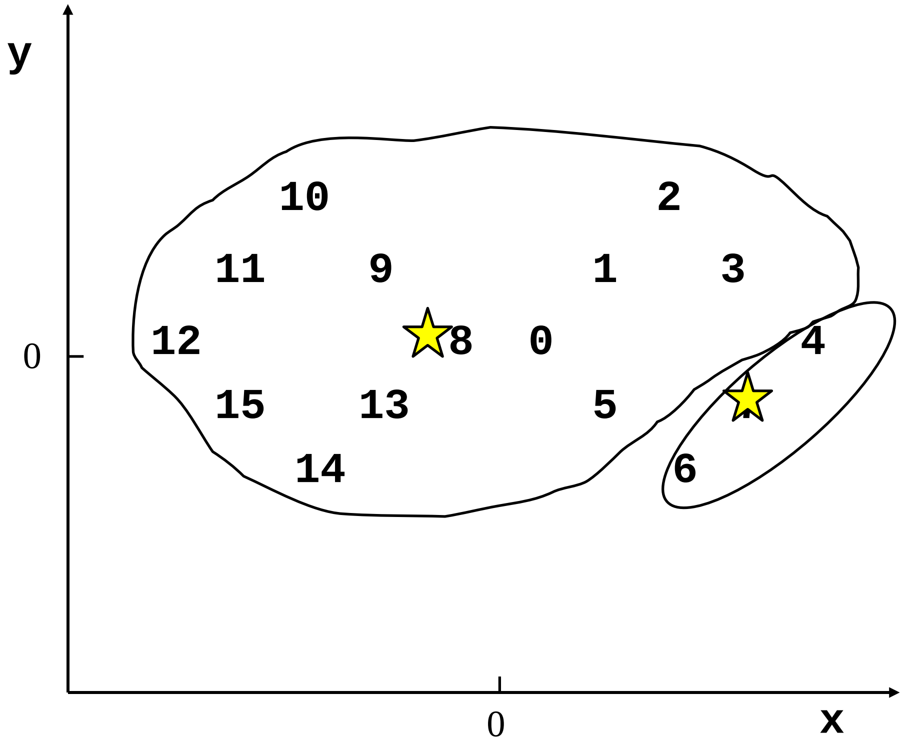
Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887



Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

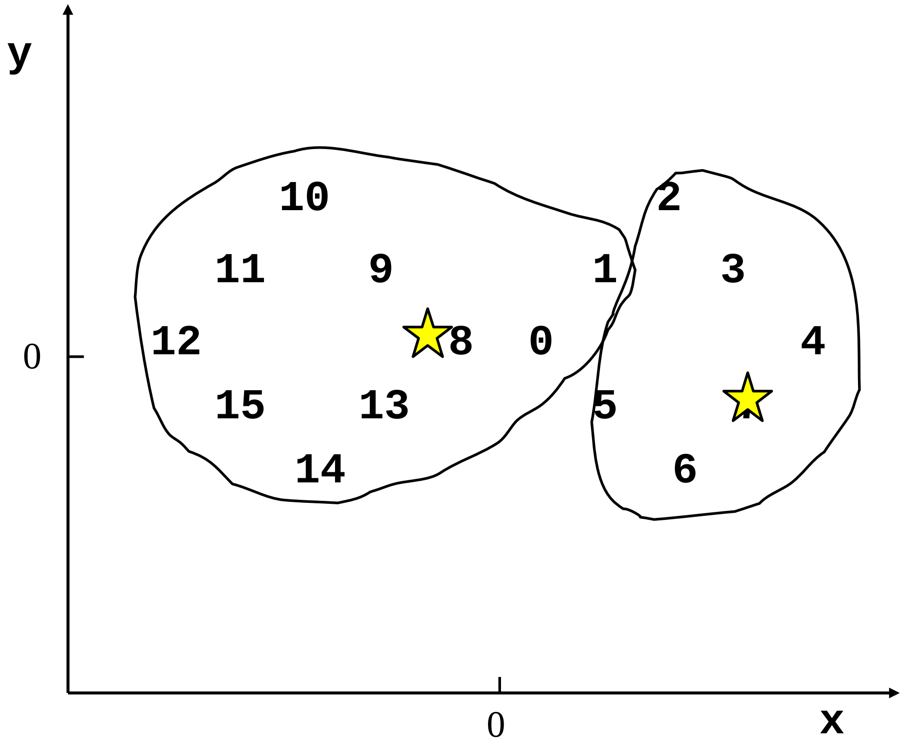
Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)



Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

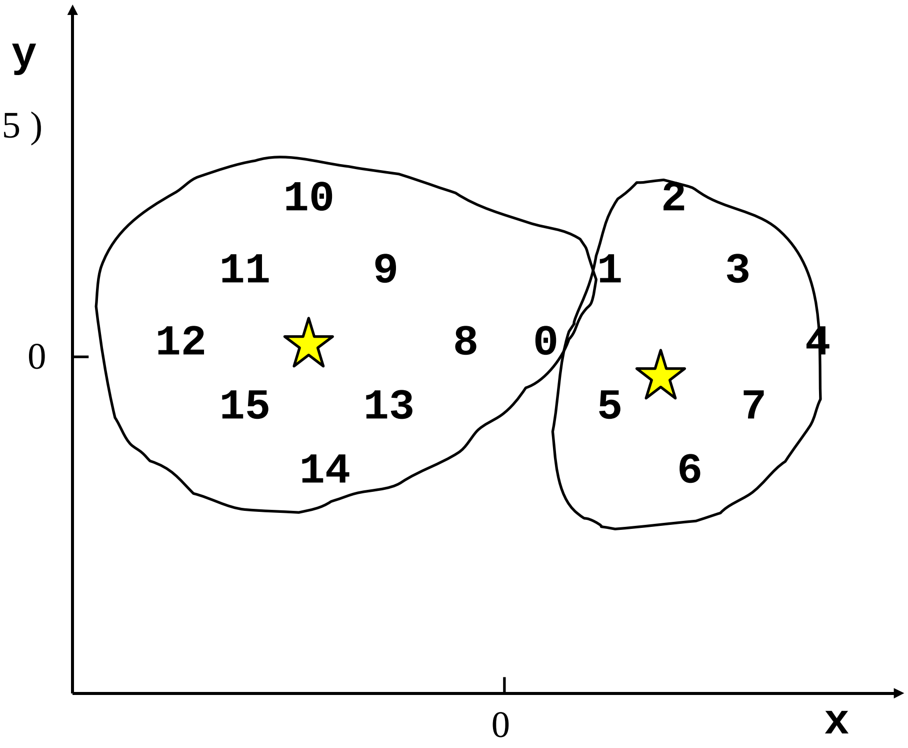


Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)

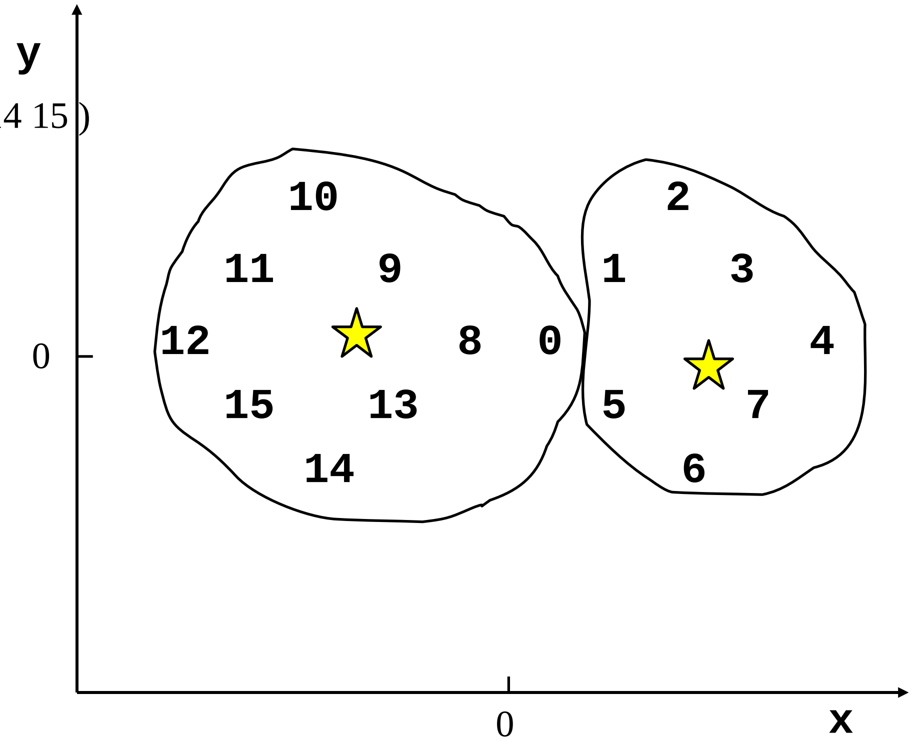


Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)
Average Distance: 3.49115



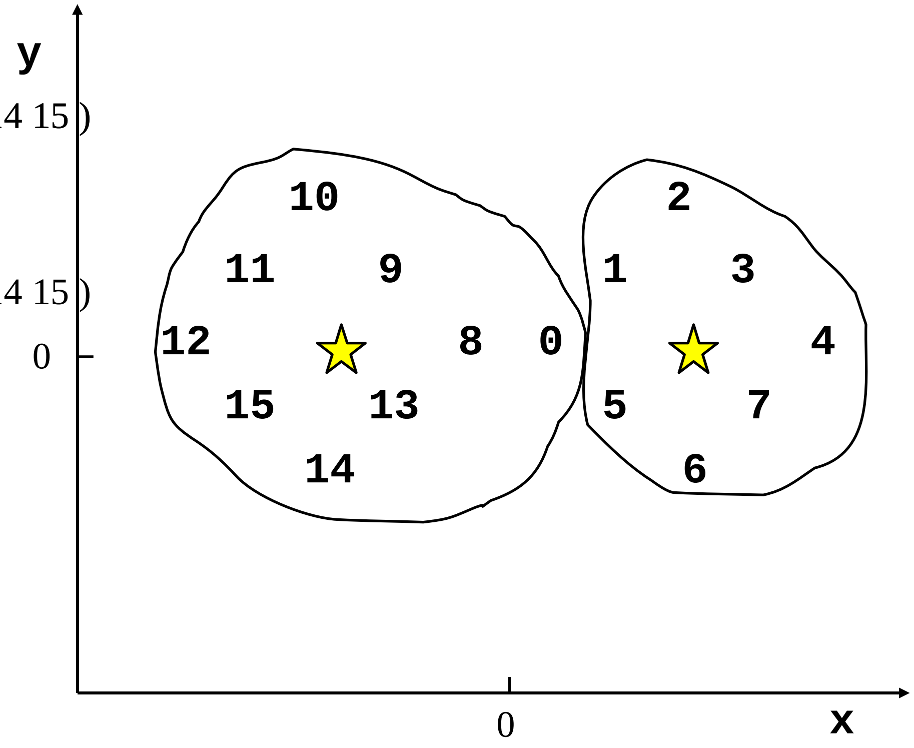
Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)
Average Distance: 3.49115

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)



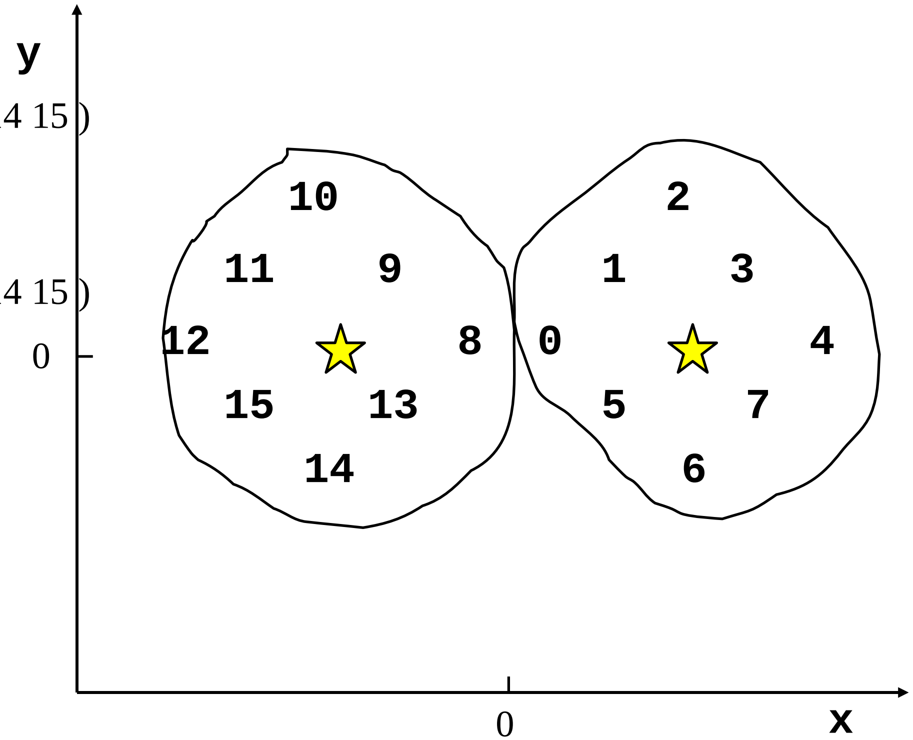
Seed: (9 0) (8 1)

Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)
Average Distance: 3.49115

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)
Cluster Centers: (5.0 0.0) (-5.0 0.0)
Average Distance: 3.41421



Seed: (9 0) (8 1)

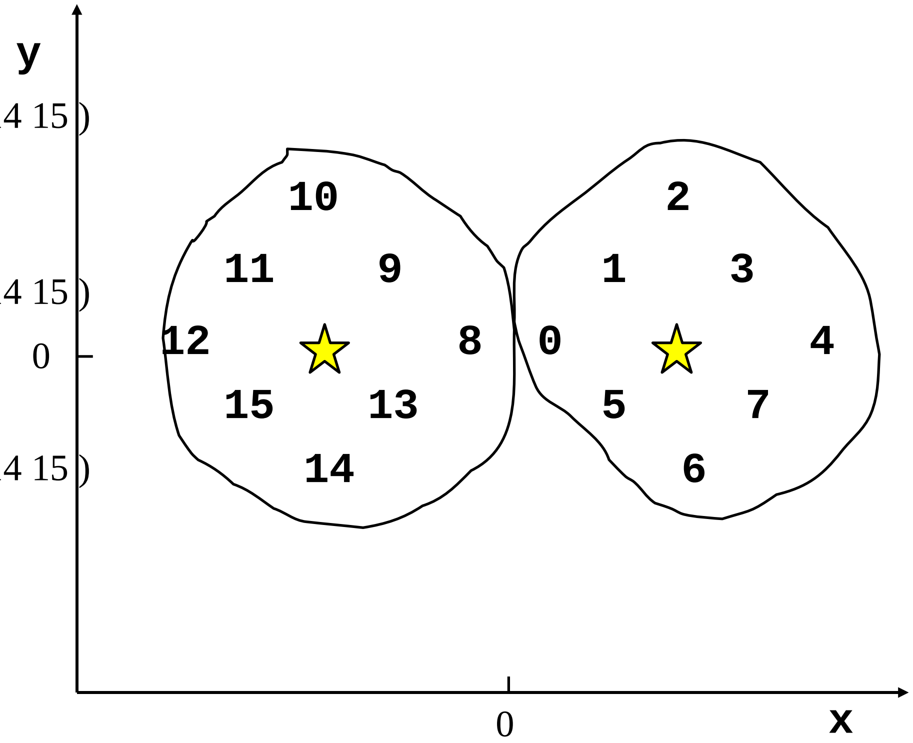
Clustering: (4 6 7) (0 1 2 3 5 8 9 10 11 12 13 14 15)
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)
Average Distance: 4.35887

Clustering: (2 3 4 5 6 7) (0 1 8 9 10 11 12 13 14 15)
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)
Average Distance: 3.6928

Clustering: (1 2 3 4 5 6 7) (0 8 9 10 11 12 13 14 15)
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)
Average Distance: 3.49115

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)
Cluster Centers: (5.0 0.0) (-5.0 0.0)
Average Distance: 3.41421

Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)
No improvement.



Termination Conditions and Convergence

- Several possibilities for termination conditions, e.g.,
  - repeat for a fixed number of iterations.
  - repeat until document partition unchanged
  - repeat until centroid positions unchanged
- Convergence
  - Why should the K-means algorithm ever reach a fixed point?
    - 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, but number of iterations could be large.
    - However, K-means typically converges quickly

Convergence of K-Means

- Define goodness measure of cluster k as sum of squared distances from cluster centroid c_k:
  - $G_k = \sum_i (d_i - c_k)^2$ (sum over all d_i in cluster k)
- and goodness measure for clustering as the sum
  - $G = \sum_k G_k$

- **E-Step** (reassignment) monotonically decreases G since each vector is assigned to the closest centroid
  - i.e., the distance to the cluster center cannot increase
- **M-Step** (recomputation) monotonically decreases each G_k because $x = \frac{1}{|G_k|} \sum_i d_i = c_k$ minimizes the function $f(x) = \sum_i (d_i - x)^2$
  - Proof:

$$f'(x) = \sum_i -2(d_i - x) = 0 \Leftrightarrow \sum_i x = \sum_i d_i \Leftrightarrow |G_k| x = \sum_i d_i$$

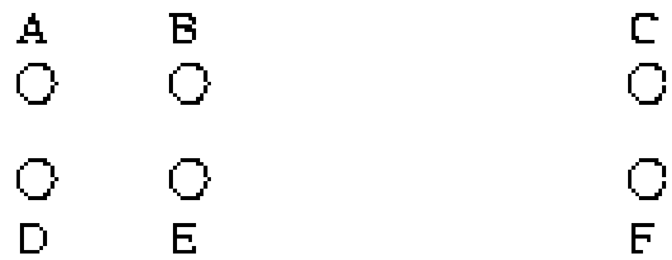
Time Complexity

- Computing distance between two docs:
  - $O(m)$ where m is the dimensionality of the vectors.
- Reassigning clusters:
  - $O(Kn)$ distance computations, in total $O(Knm)$
- Computing centroids:
  - Each doc gets added once to some centroid: $O(nm)$.
- Repeat this for I iterations:
  → Complexity is $O(IKnm)$ in total

Seed Choice

- Results can vary based on random seed selection.
  - Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
- Possible Strategies:
  - 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.

Example showing sensitivity to seeds



In the above, if you start with B and E as centroids you converge to {A,B,C} and {D,E,F}
If you start with D and F you converge to {A,B,D,E} {C,F}

How Many Clusters?

- The number of desired clusters *K* is not always given
- Finding the “right” *K* may be part of the problem
  - Given documents, partition into an “appropriate” number of subsets.
  - E.g., for query results - ideal value of *K* not known up front - though UI may impose limits.

- Simple Strategy:
  - Compute a clustering for various values of K
  - choose the best one
- But how can we measure Cluster Quality?
  - Why can't we use, e.g., the G-measure?

Trading Off Cluster Quality and Number of Clusters

- Measures that measure the quality of a clustering by average distances to cluster centers are easy to optimize
  - the optimum is always the largest K
    - see convergence proof
    - limiting case: for $K = N$, we have $G = 0$
- Strategy: Combine quality measures with a penalty for high number of clusters
  - For each cluster, we have a Cost *C*.
  - Thus for a clustering with *K* clusters, the Total Cost is *KC*.
  - Define the Value of a clustering to be =

    Average Distances + Total Cost.
  - Find the clustering of lowest value, over all choices of *K*.
    - Total benefit increases with increasing *K*. But can stop when it doesn't increase by "much". The Cost term enforces this.

K-means issues, variations, etc.

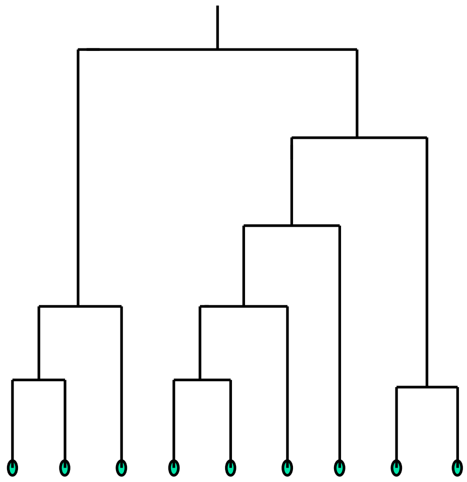
- Recomputing the centroid after every assignment (rather than after all points are re-assigned) can improve speed of convergence of K-means
- Assumes clusters are spherical in vector space
  - Sensitive to coordinate changes, weighting etc.
- Disjoint and exhaustive
  - Doesn't have a notion of "outliers"

Hierarchical Clustering

- Produces a tree hierarchy of clusters
  - *root:* all examples
  - *leaves:* single examples
  - *interior nodes:* subsets of examples
- Two approaches
  - **Top-down:**
    - start with maximal cluster (all examples)
    - successively split existing clusters
      - e.g., recursive application of k-means Clustering
  - **Bottom-up:**
    - start with minimal clusters (single examples)
    - successively merge existing clusters

Hierarchical Agglomerative Clustering

- Assumes a similarity function for determining
  - the similarity of two instances
    (and more generally the similarity of two clusters)
- Bottom-up strategy:
  - Starts with all instances in a separate cluster
  - 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 or dendrogram
  - a clustering can be obtained by cutting the dendrogram at a given level
  - all connected components form a cluster



Hierarchical Agglomerative Clustering

1. Start with one cluster for each example: $C = \{C_i\} = \{\{o_i\} \mid o_i \in O\}$
2. compute distance $d(C_i, C_j)$ between all pairs of Cluster C_i, C_j
3. Join clusters C_i und C_j with minimum distance into a new cluster C_p; make C_p the parent node of C_i and C_j :

   $$C_p = \{C_i, C_j\}$$
   $$C = (C \setminus \{C_i, C_j\}) \cup \{C_p\}$$

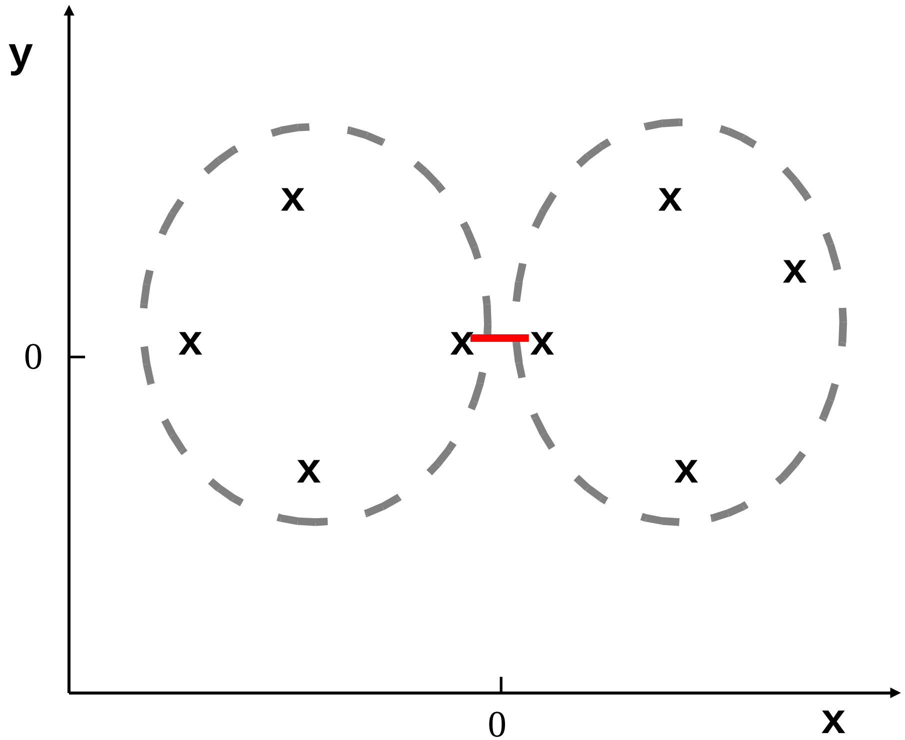
4. Compute distances between C_p and other clusteres in C
5. If $|C| > 1$, goto 3.

Similarity between Clusters

ways of computing a similarity/distance between clusters C_1 and C_2

- Single-link:
  - minimum distance between two elements of C_1 and C_2

    $d(C_1, C_2) = \min\{ d(x, y) \mid x \in C_1, y \in C_2 \}$

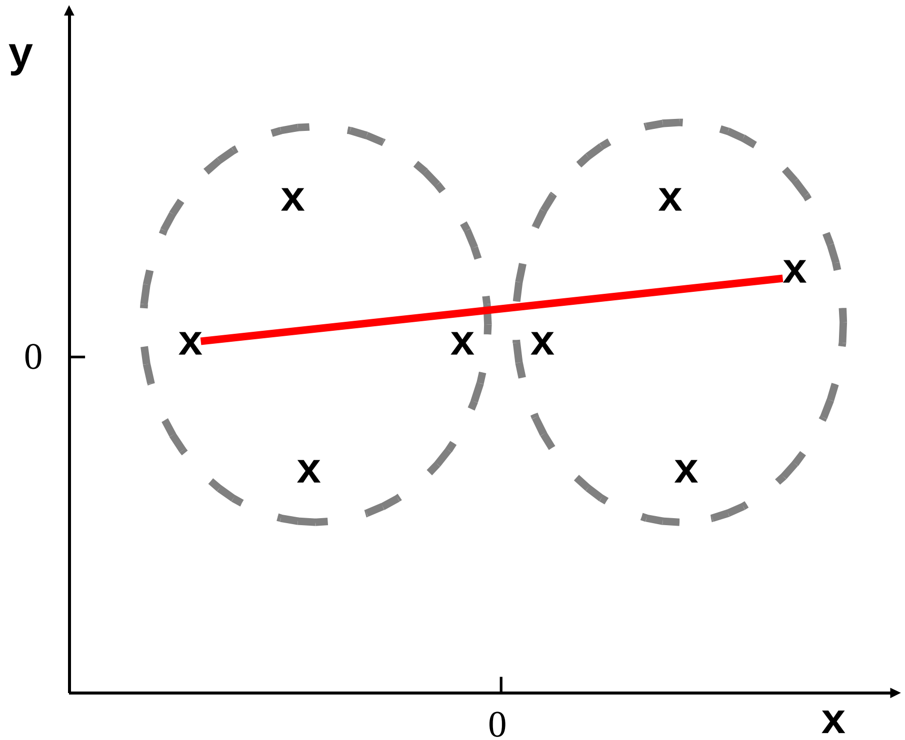


Similarity between Clusters

ways of computing a similarity/distance between clusters C_1 and C_2

- Complete-link:
  - maximum distance between two elements of C_1 and C_2

$$d(C_1, C_2) = \max\{\, d(x, y) \mid x \in C_1, y \in C_2 \,\}$$

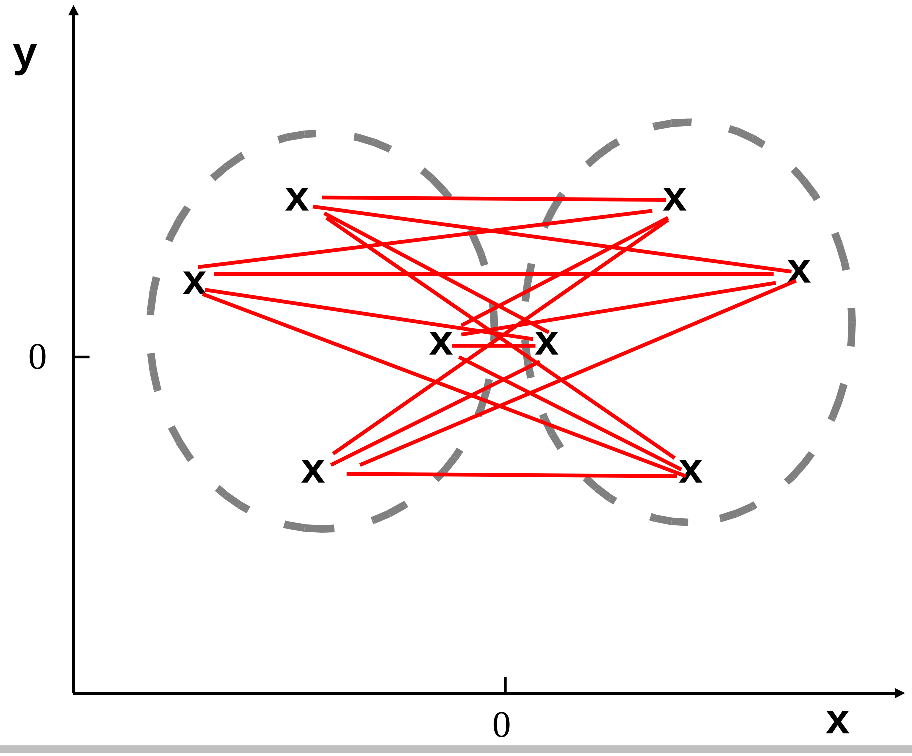


Similarity between Clusters

ways of computing a similarity/distance between clusters C_1 and C_2

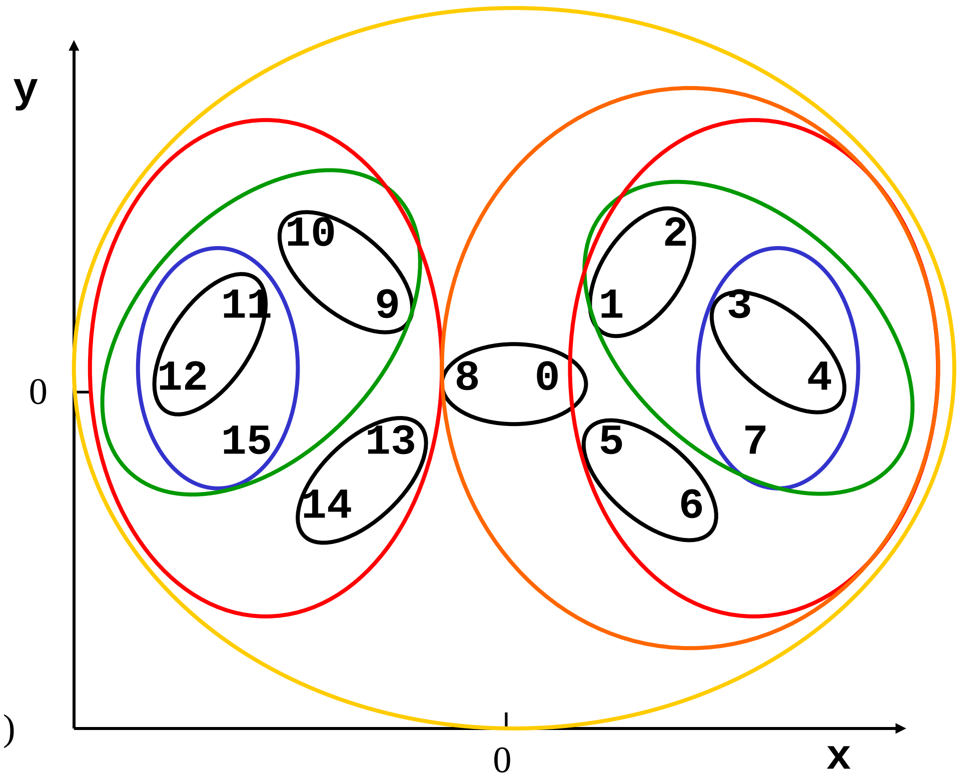
- Average-link:
  - average distance between two elements of C_1 and C_2

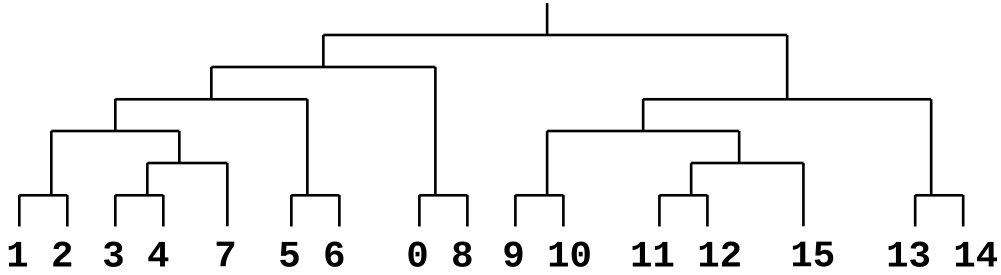
    $d(C_1, C_2) = \sum\{ d(x, y) \mid x \in C_1, y \in C_2 \} / |C_1| / |C_2|$



Bottom-up clustering (average-link):

min distance = 2.00000 (8) (0)
min distance = 2.82843 (2) (1)
min distance = 2.82843 (4) (3)
min distance = 2.82843 (6) (5)
min distance = 2.82843 (10) (9)
min distance = 2.82843 (12) (11)
min distance = 2.82843 (14) (13)
min distance = 3.16228 (7) (3 4)
min distance = 3.16228 (15) (11 12)
min distance = 4.73756 (3 4 7) (1 2)
min distance = 4.73756 (11 12 15) (9 10)
min distance = 4.74131 (1 2 3 4 7) (5 6)
min distance = 4.74131 (9 10 11 12 15) (13 14)
min distance = 5.57143 (0 8) (5 6 1 2 3 4 7)
min distance = 9.90476 (13 14 9 10 11 12 15) (5 6 1 2 3 4 7 0 8)





Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances
  - complexity 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.
  - can be obtained if, e.g., each cluster is represented with a single representative (a centroid)
- Else $O(n^2 \log n)$ or $O(n^3)$ if done naively

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
  - naïve approach:
    - use the 5-10 most frequent words in each cluster
    - Problem: clusters might have a uniform topic (e.g., computers)
  - Use distinguishing words/phrases
    - that appear more frequently in one class than in other classes
    - e.g., significance tests

Learning with Labelled and Unlabelled Data

- Supervised learning
  - Assign each example to a group (*class*)
  - Given: Training set with class labels
- Unsupervised learning
  - Find groups of examples that "belong together"
  - No class information is given in the training set
- On the Web
  - many tasks are supervised (require labeled examples)
  - there are many *unlabeled* documents
  - but labeling them is expensive

→ semi-supervised learning
  - augment unlabeled data with a (small) set of labeled data

Semi-Supervised Learning

- Goal:
  - Reduce the amount of labelled data needed by letting classifiers make use of additional unlabelled data

- Some Techniques:
  - **Active Learning:**
    - Classifier chooses examples that should be labelled
  - **Self-Training:**
    - Classifier labels its own examples
  - **Co-Training:**
    - Two classifier label each others examples
    - Multi-View Learning: Special case where the classifiers are identical, but trained on different features sets

Uncertainty Sampling
(Lewis, Catlett/Gale, 1994)

- The Learner decides which examples the teacher should label

1. Train a classifier on the labeled training set
2. Let the learner predict for each example in the unlabeled set
3. Choose the *n* examples where it has the *least* confidence in its predictions (is most uncertain about the classification)
4. Let the teacher label these examples
5. Goto 1. unless no improvement

- Properties:
  - Needs classifiers with (good) confidence estimates in its predictions
  - Reduces work-load for teacher
  - may oversample certain classes

Results Uncertainty Sampling

- data: AP newswire articles
- results show that uncertainty sampling (999 examples) is more efficient than random selection (10,000 examples)

Category	Reject All	3 + 996 uncertainty				3 + 9997 random			
		C4.5 (LR=5)		prob. (LR=1)		C4.5 (LR=1)		prob. (LR=1)	
		Average	SD	Average	SD	Average	SD	Average	SD
tickertalk	0.077	0.077	(0.000)	0.078	(0.001)	0.078	(0.003)	0.109	(0.044)
boxoffice	0.081	0.047	(0.002)	0.048	(0.008)	0.061	(0.018)	0.077	(0.021)
bonds	0.115	0.064	(0.002)	0.069	(0.006)	0.076	(0.020)	0.145	(0.069)
nielsens	0.167	0.094	(0.011)	0.062	(0.005)	0.107	(0.006)	0.100	(0.026)
burma	0.179	0.090	(0.008)	0.098	(0.006)	0.115	(0.040)	0.193	(0.046)
dukakis	0.206	0.197	(0.014)	0.208	(0.020)	0.210	(0.039)	0.235	(0.036)
ireland	0.225	0.188	(0.005)	0.189	(0.011)	0.220	(0.024)	0.228	(0.016)
quayle	0.256	0.161	(0.009)	0.222	(0.012)	0.143	(0.010)	0.263	(0.035)
budget	0.379	0.336	(0.010)	0.361	(0.009)	0.350	(0.014)	0.392	(0.016)
hostages	0.439	0.415	(0.024)	0.360	(0.016)	0.466	(0.039)	0.431	(0.018)

Table 2: Average and standard deviation of percentage error of various classifiers. *Reject all* is a classifier that deems all instances non-members of the category. Two types of training set were used: an uncertainty sample of size 999 and a random sample of size 10,000. Two types of classifier are built from each training set: a decision rule classifier trained using C4.5, and the probabilistic classifier described in the text. When C4.5 was used on the uncertainty sample, a loss ratio of 5 was used; for the random sample a loss ratio of 1 was used (original C4.5). Figures are averages over 20 runs for classifiers built from random samples using the probabilistic method, and over 10 runs for the other three combinations.

Self-Training
(Nigam, McCallum, Thrun &Mitchell, 2000)

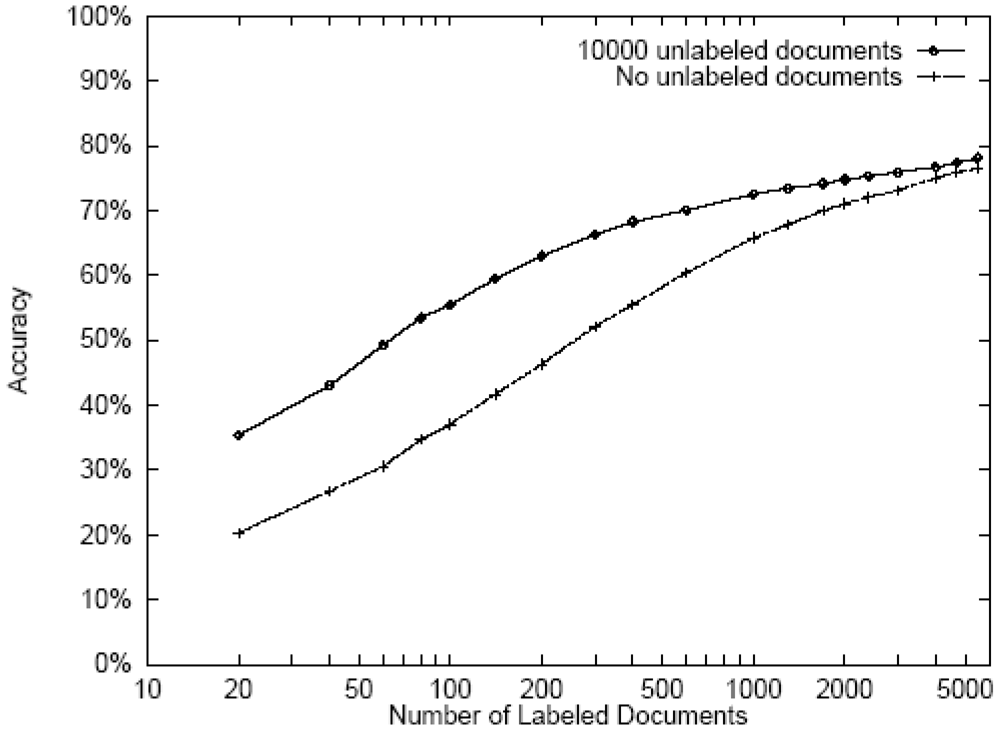
- Using EM (Expectation Maximization) algorithm

1. Train an initial classifier on the labeled documents
2. **E-Step:** Assign class labels to the unlabeled documents
3. **M-Step:** Train a classifier from all examples
4. Goto 2. unless no significant changes

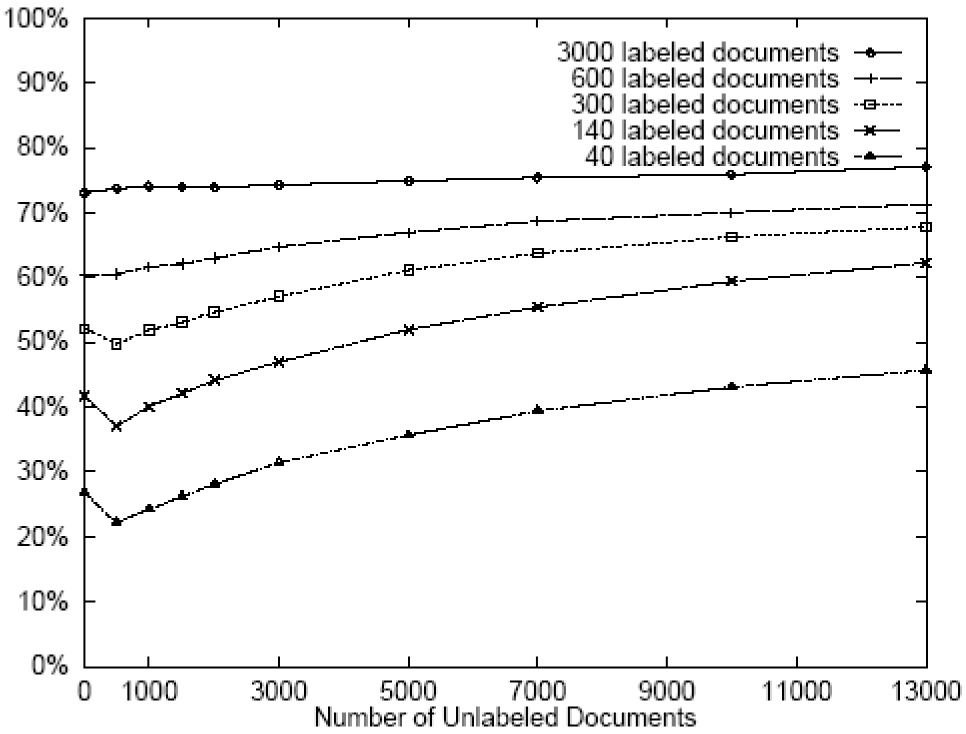
- Properties:
  - Works well for classifiers that use all of the features (e.g., Naïve Bayes)
    - Unlabelled data help to estimate the word probabilities
  - Does not work well for classifiers that use only a few features (e.g., decision trees, rule learners)
    - Subsequent iterations only reinforce the use of the same features as in the concept constructed in step 1.

Self-Training: Performance

unlabelled documents improve performance



the more unlabelled documents the better



Co-Training
(Blum & Mitchell, 1998)

- Using two classifier to label each other's data

1. Train Classifiers 1 and 2 on labelled data
2. Let Classifier *i* pick the n examples where it has the highest confidence in its predictions
3. Add the examples labelled by classifier 2 to the training set of classifier 1 and vice versa
4. Goto 2. as long as there is some improvement

- Properties:
  - Works well if the two classifiers
    - provide (good) confidence estimates in their own predictions
    - are diverse (tend to be correct on different regions of the example space)
  - Could be generalized to more than 2 classifiers

Multi-View Learning

- To obtain diverse and independent classifiers for co-training, use two different feature sets (two views)
  - T_D = bag of words in document D
  - T_A = bag of anchor texts from HREF tags that target D
  - alternatively, two random subsets of all available features could be used
- Co-training with multiple views reduces the error of each individual view (classifier)
- Further reduction can be obtained by combining the predictions of the two classifiers
  - e.g., pick a class c by maximizing $p(c|T_D)\,p(c|T_A)$ (assumes independence of T_A and T_D)
- Multi-View Learning is still a hot research topic

Results Multi-View Learning

Co-training reduces classification error

Shown is the reduction in error against the number of mutual training rounds.