

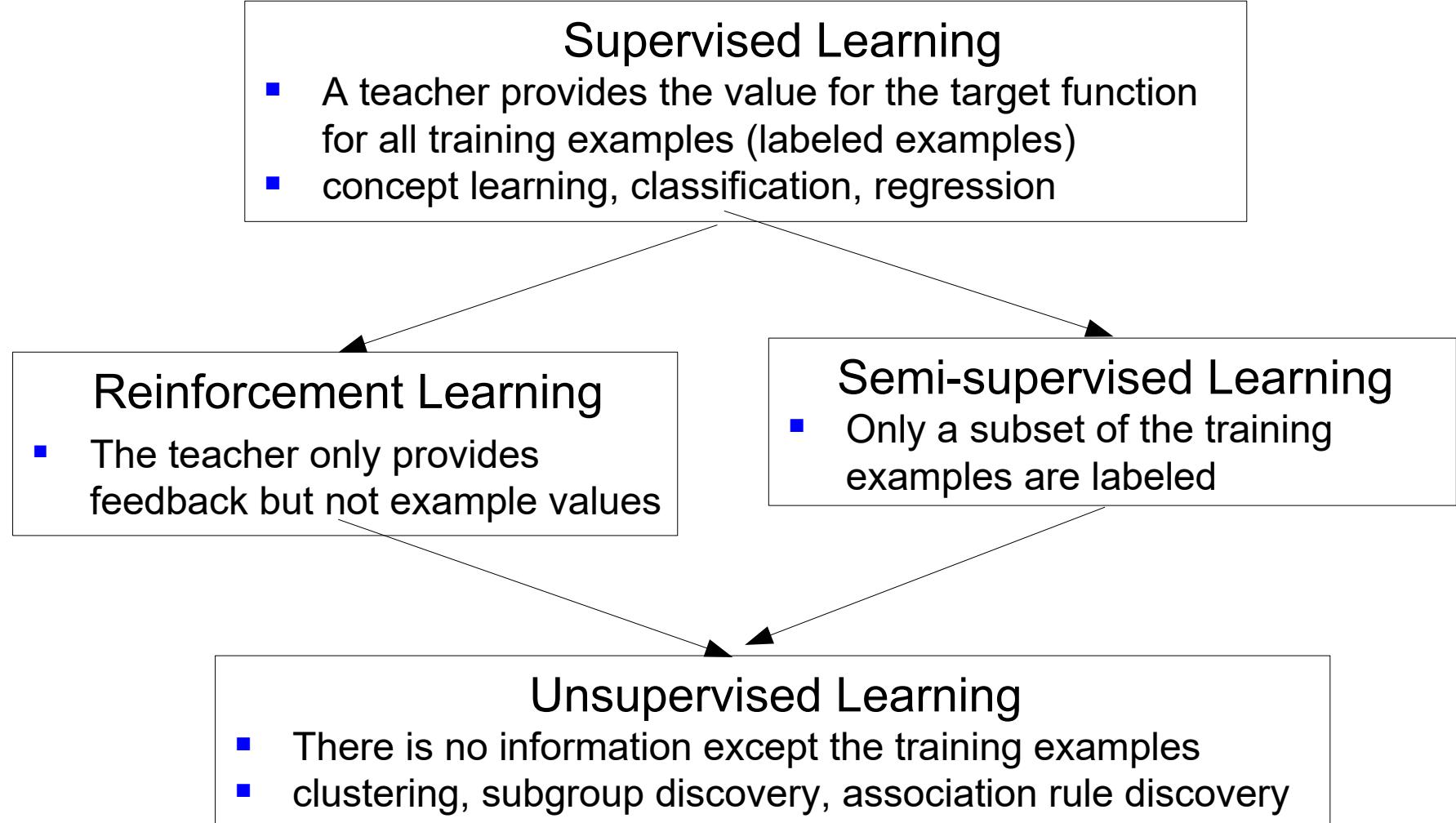


Unsupervised Learning

- Clustering
- Basic algorithms
 - k-means clustering
 - bottom-up hierarchical clustering
- Efficient algorithms
 - BIRCH
 - BUBBLE
 - CURE

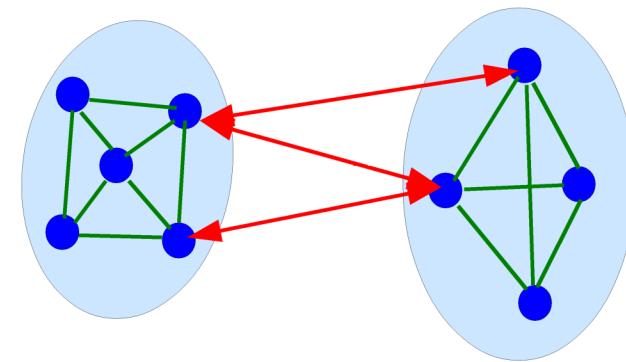


Different Learning Scenarios

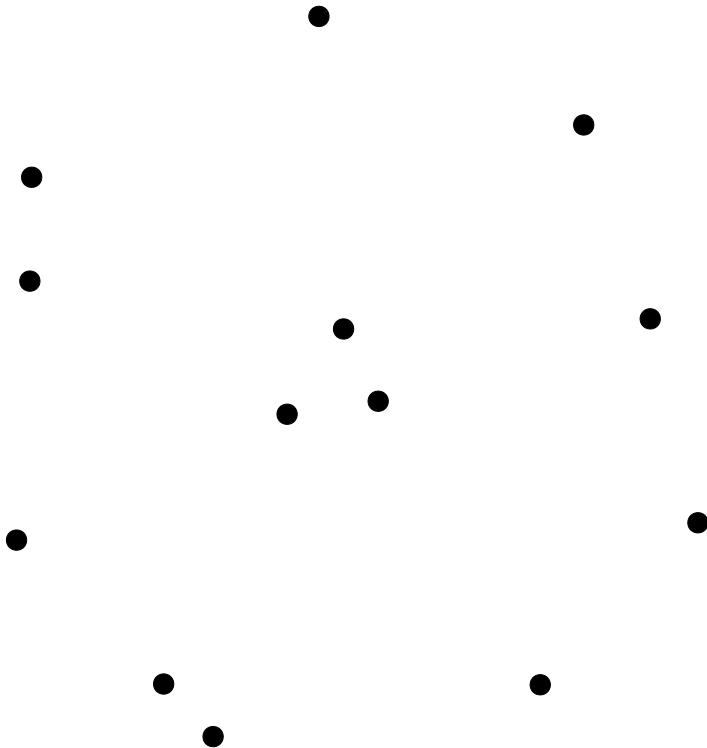


Clustering

- Given:
 - a set of examples (e.g., documents)
 - in some description language (e.g., attribute-value)
 - no labels (\rightarrow unsupervised)
- 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



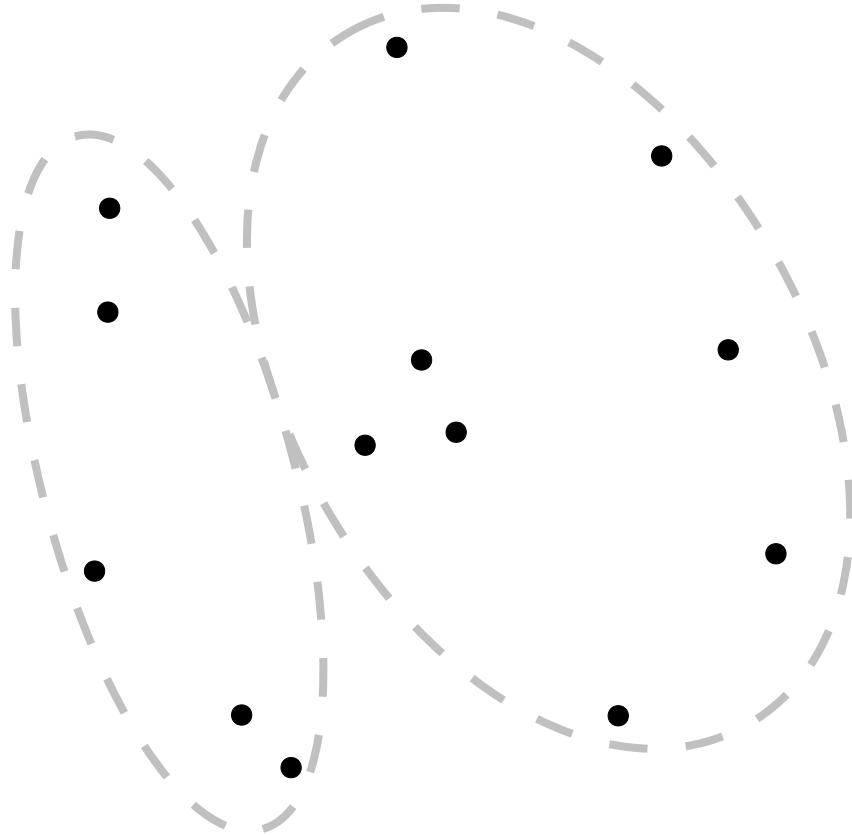
Clustering



Given a set of objects, divide it into groups (clusters) so that

- objects of same group are close to each other
(intra-class similarity)
- objects of different groups are far from each other
(inter-class dissimilarity)

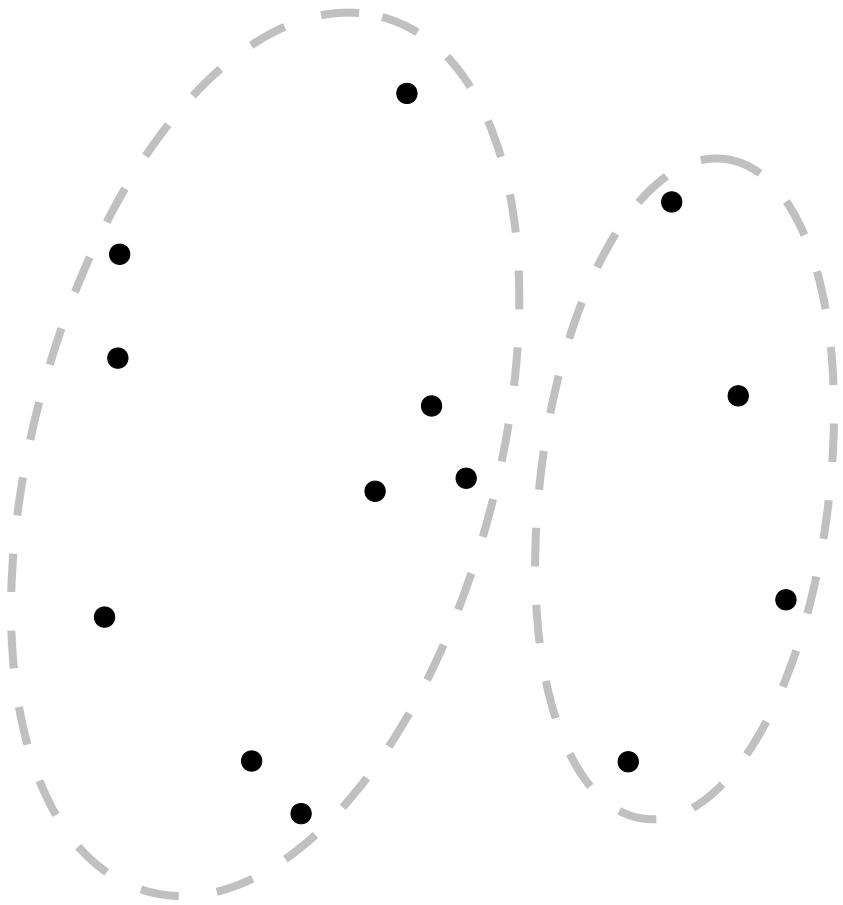
Clustering



Given a set of objects, divide it into groups (clusters) so that

- objects of same group are close to each other
(intra-class similarity)
- objects of different groups are far from each other
(inter-class dissimilarity)

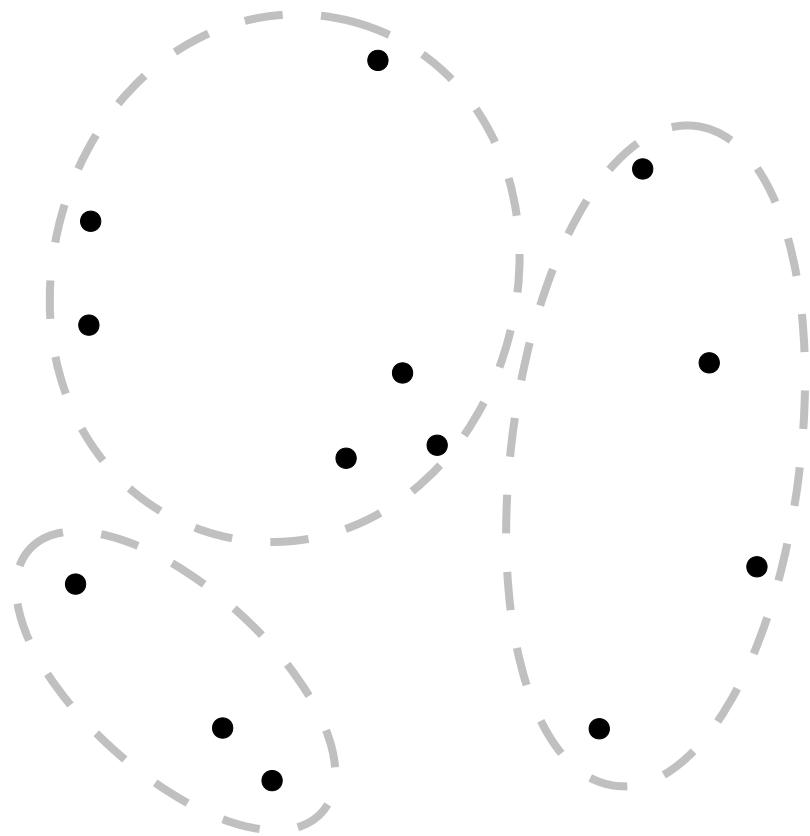
Clustering



Given a set of objects, divide it into groups (clusters) so that

- objects of same group are close to each other
(intra-class similarity)
- objects of different groups are far from each other
(inter-class dissimilarity)

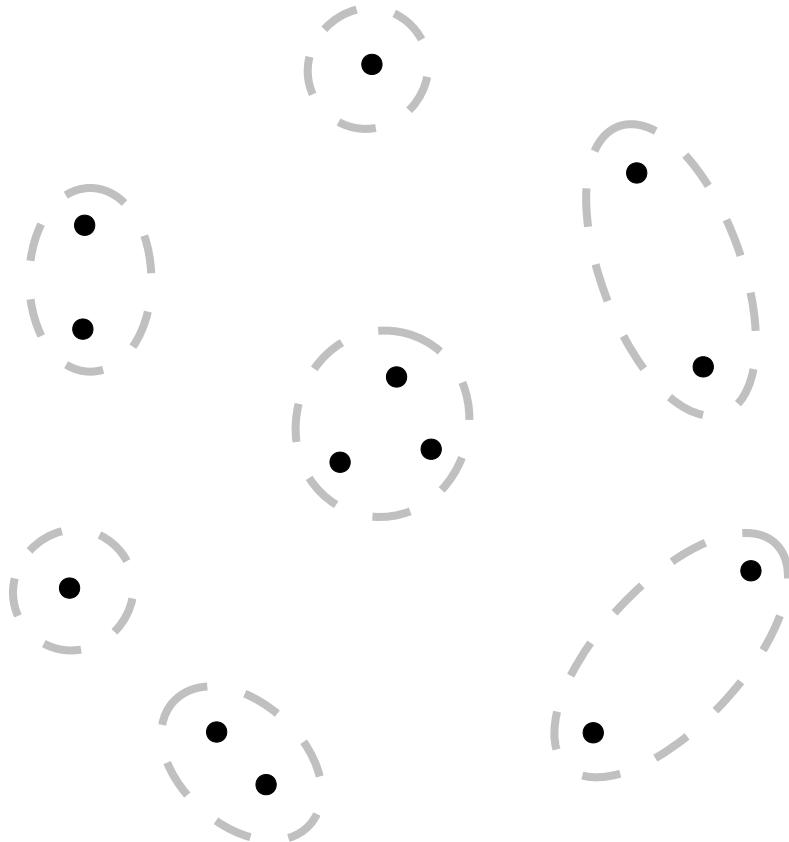
Clustering



Given a set of objects, divide it into groups (clusters) so that

- objects of same group are close to each other
(intra-class similarity)
- objects of different groups are far from each other
(inter-class dissimilarity)

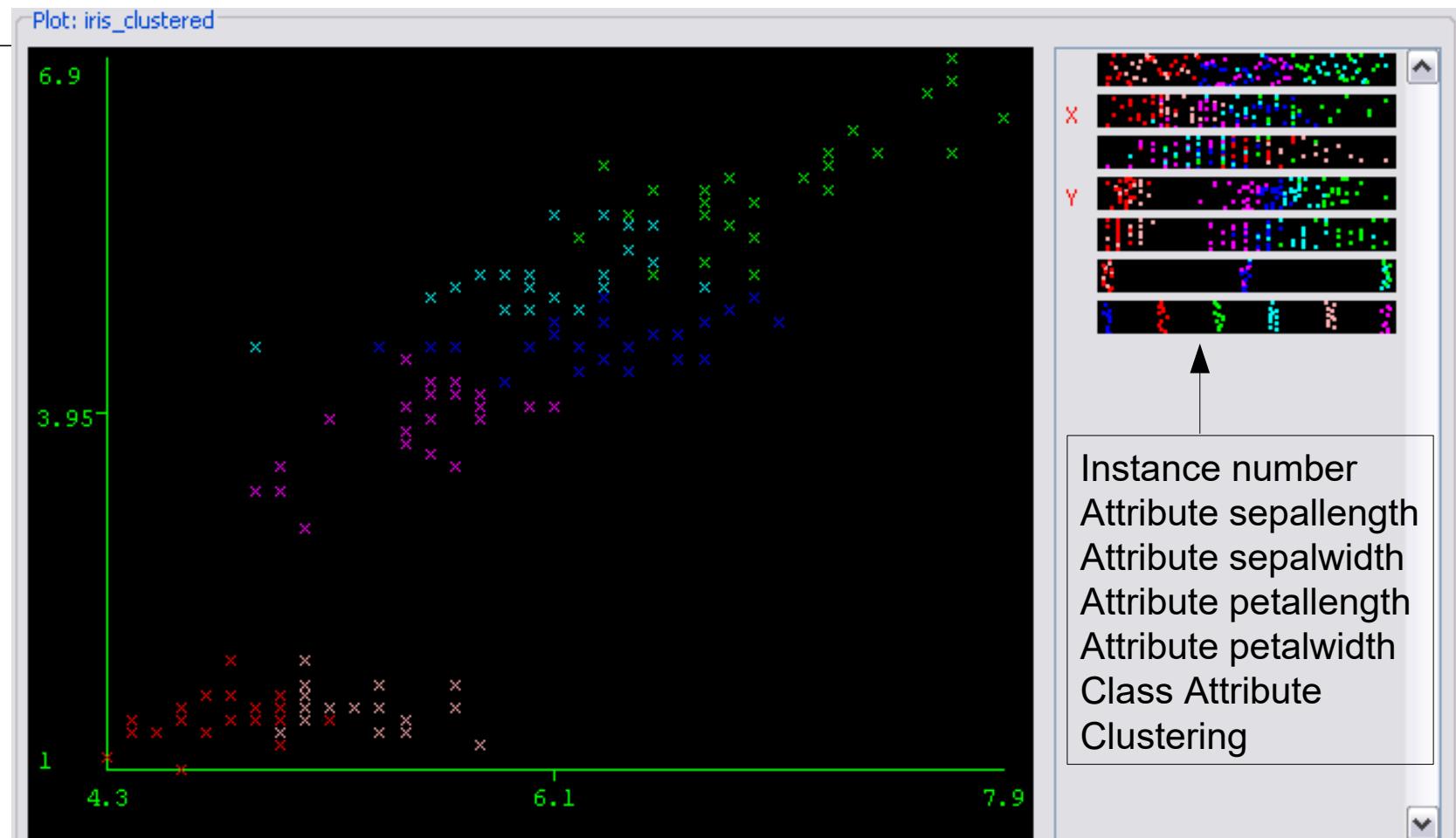
Clustering



Given a set of objects, divide it into groups (clusters) so that

- objects of same group are close to each other
(intra-class similarity)
- objects of different groups are far from each other
(inter-class dissimilarity)

6 clusters on Iris dataset



Basic Clustering Algorithms

- k-means clustering
 - given a similarity metric (like k-NN algorithms)
 - initialize k cluster centers
 - iteratively assign examples to closest neighbor
 - until procedure converges
- bottom-up hierarchical clustering
 - each example is a cluster
 - iteratively merge clusters, similar to chi-merge
- Cobweb
 - incrementally build up a tree structure
 - each node/cluster can estimate a probability that an example belongs to this cluster
 - examples are sorted into the tree in a top-down way



k-means Clustering

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

1. Randomly select k points c_k 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

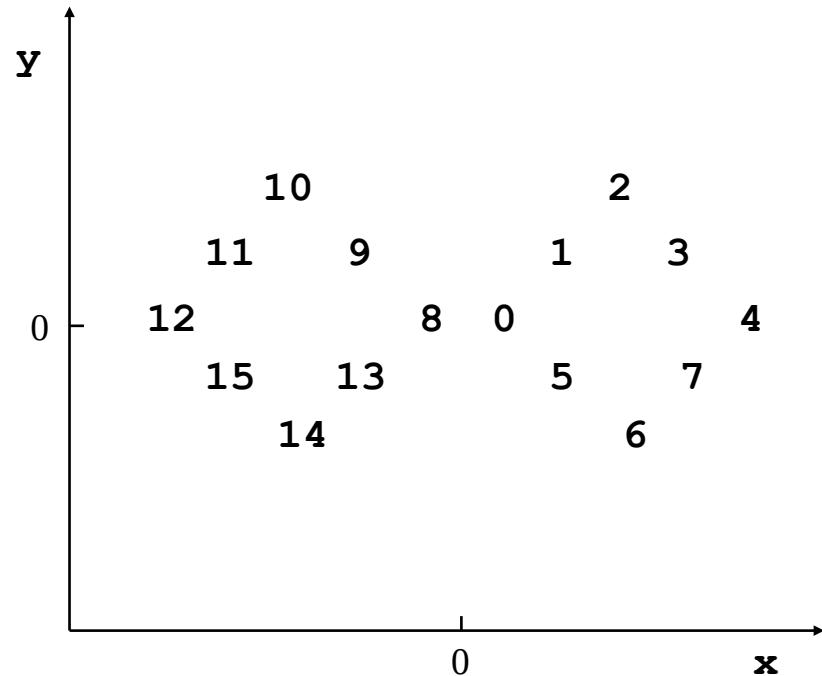
$$c_k \leftarrow \frac{1}{n_k} \sum_{i=1}^{n_k} d_i$$

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

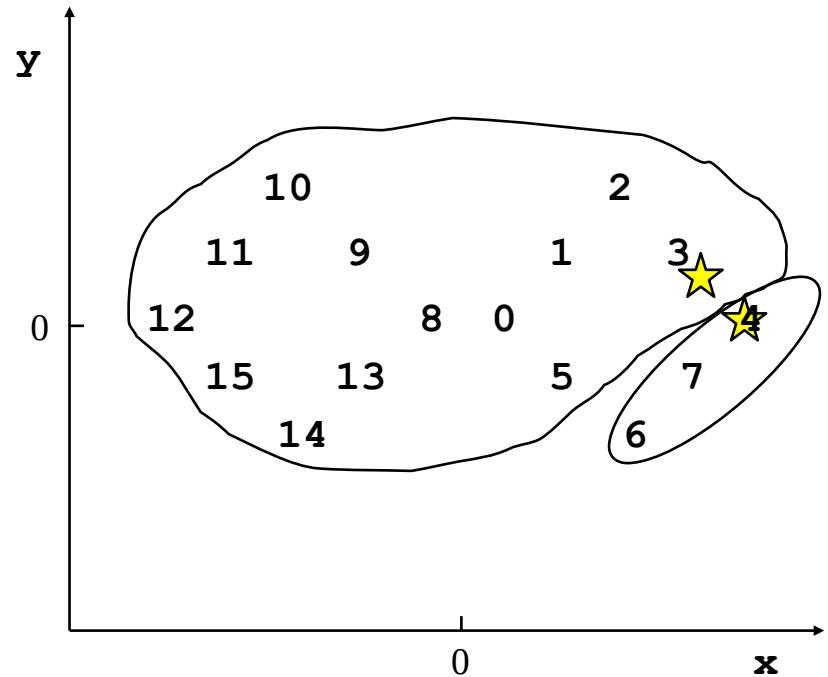
k-means-Clustering – Example

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



k-means-Clustering – Example

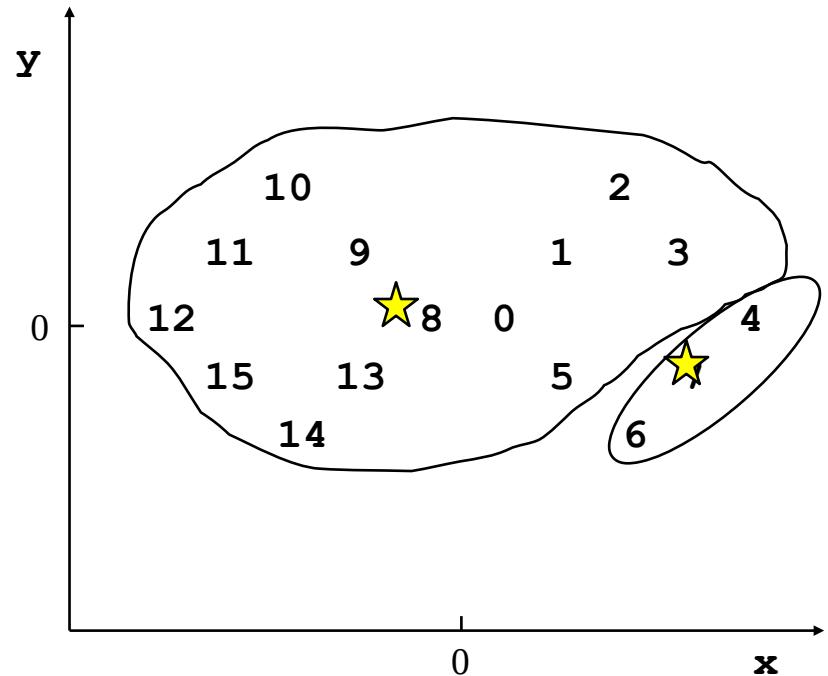
Seed: (9 0) (8 1)

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)



k-means-Clustering – Example

Seed: (9 0) (8 1)

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

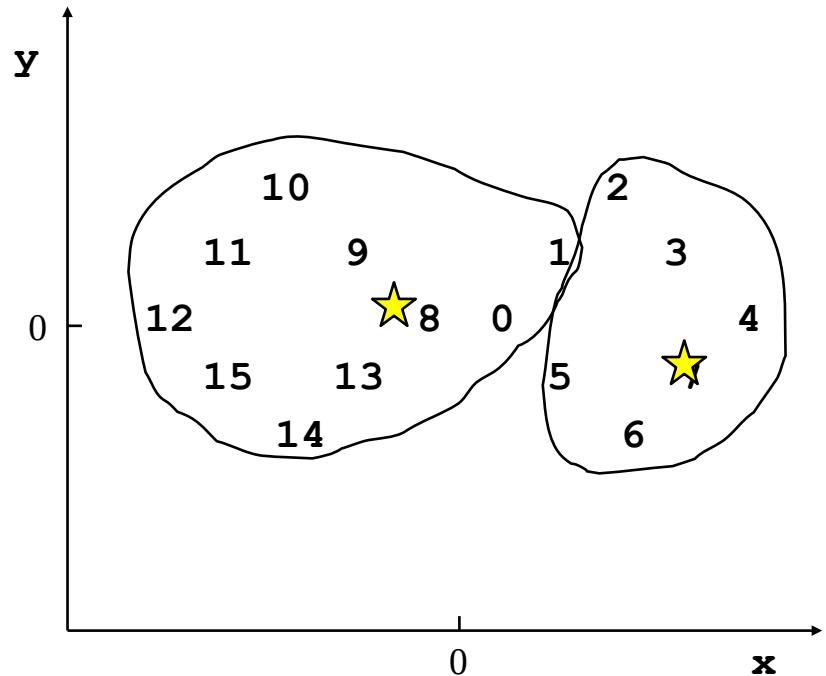
Cluster Centers: (7.0 -2.0) (-1.61538 0.46153)

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



k-means-Clustering – Example

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)

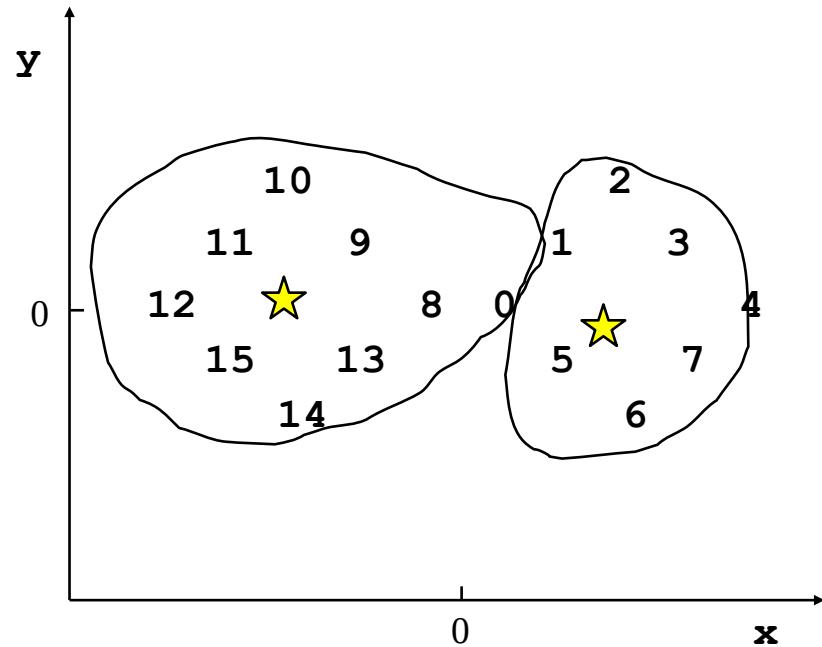
Average Distance: 4.35887

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)



k-means-Clustering – Example

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)

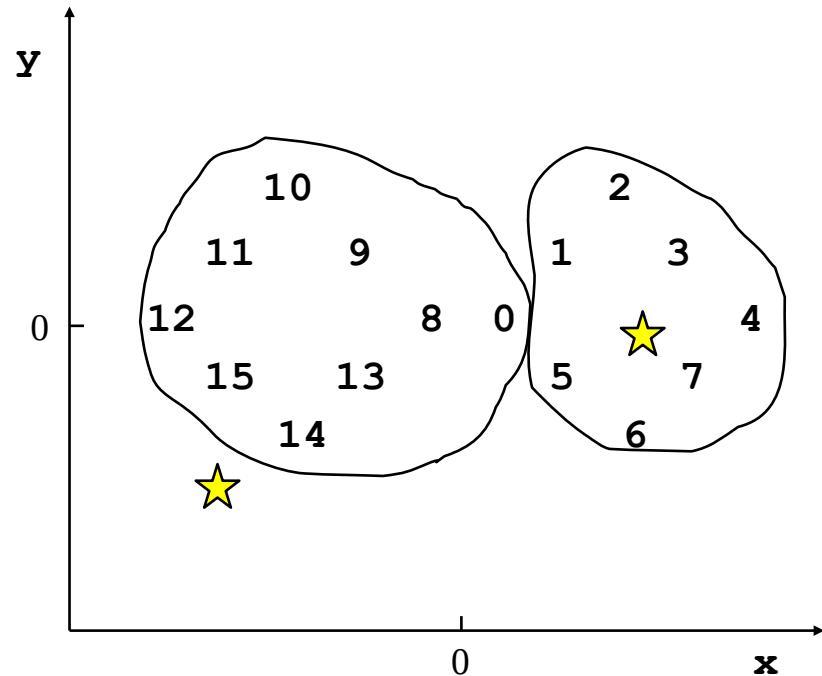
Cluster Centers: (6.0 -0.33334) (-3.6 0.2)

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



k-means-Clustering – Example

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)

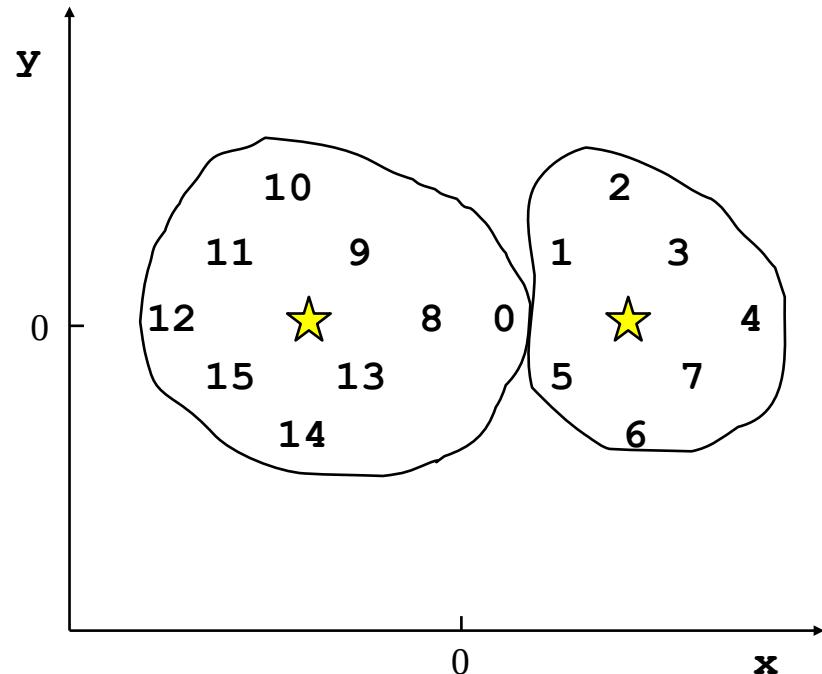
Average Distance: 3.6928

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)



k-means-Clustering – Example

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)

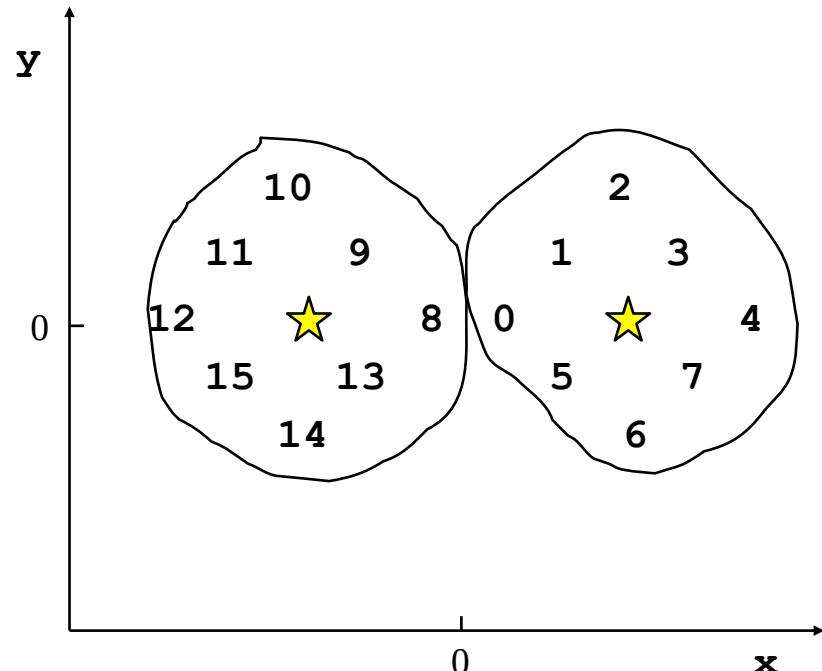
Cluster Centers: (5.57143 0.0) (-4.33334 0.0)

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



k-means-Clustering – Example

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

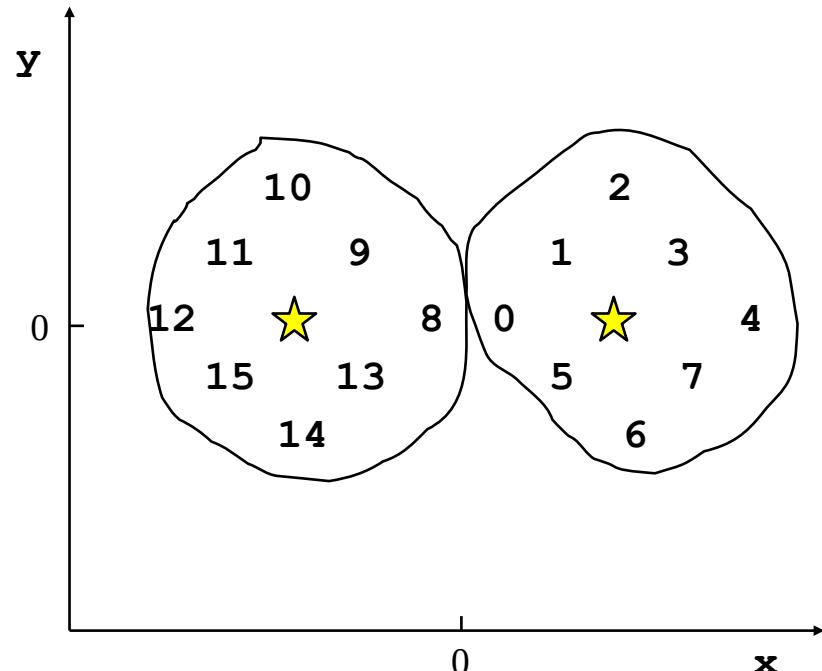
Clustering: (0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15)

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 fix point?
 - Fix 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 :
 - $G_k = \sum_{i=1}^{n_k} (d_i - c_k)^2$ (sum over all d_i in cluster k)
- and goodness measure for clustering as the sum
 - $G = \sum_{k=1}^K G_k$
- **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 $x = \frac{1}{n_k} \sum_{i=1}^{n_k} d_i = c_k$ minimizes the function $f(x) = \sum_i (d_i - x)^2$
 - Proof:
$$f'(x) = \sum_{i=1}^{n_k} -2(d_i - x) = 0 \Leftrightarrow \sum_{i=1}^{n_k} x = \sum_{i=1}^{n_k} d_i \Leftrightarrow n_k \cdot x = \sum_{i=1}^{n_k} d_i \Leftrightarrow x = c_k$$



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

A	B	C
○	○	○
○	○	○
D	E	F

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
 - Partition the instances into an “appropriate” number of subsets.
 - Ideal value of K not known up front
- 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 Cluster cost C .
 - Thus for K clusters, the Total Cluster Cost is KC .
 - Define the Value of a clustering to be =
Average Distances + Total Cluster 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.



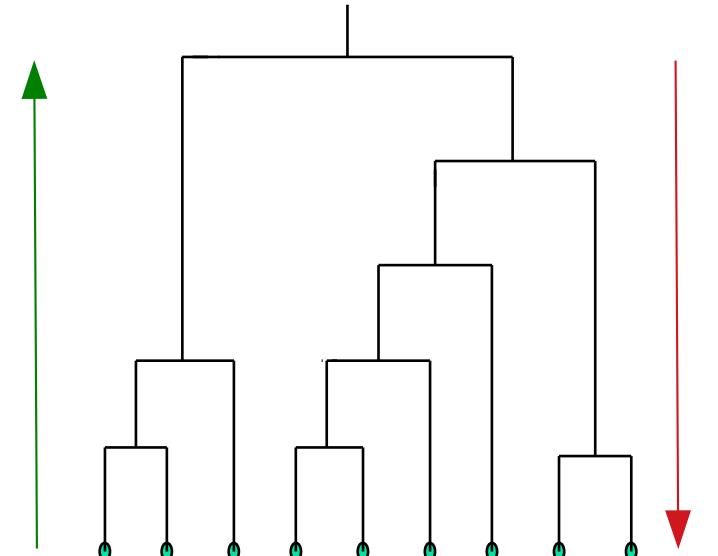
K-means issues, variations, etc.

- 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"



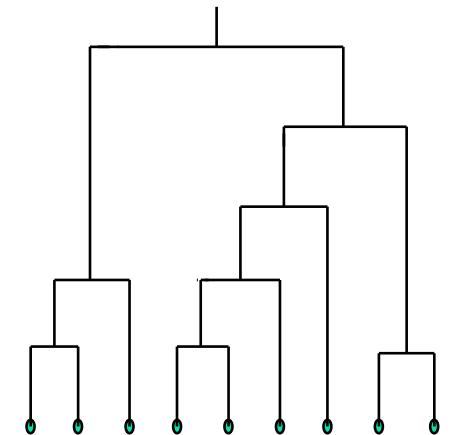
Hierarchical Clustering

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

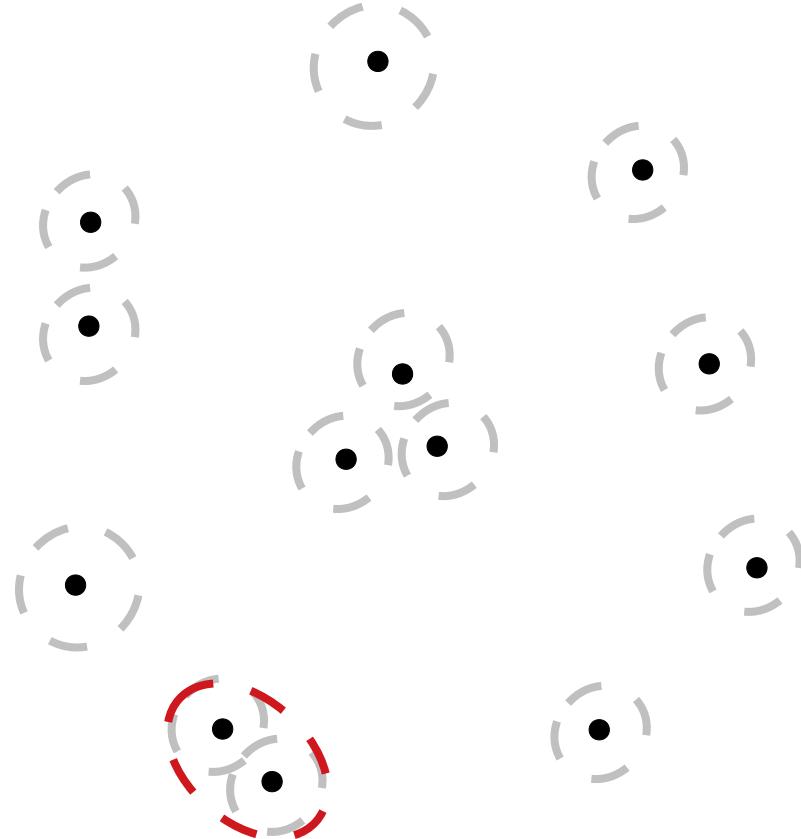


Hierarchical Agglomerative Clustering

- 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



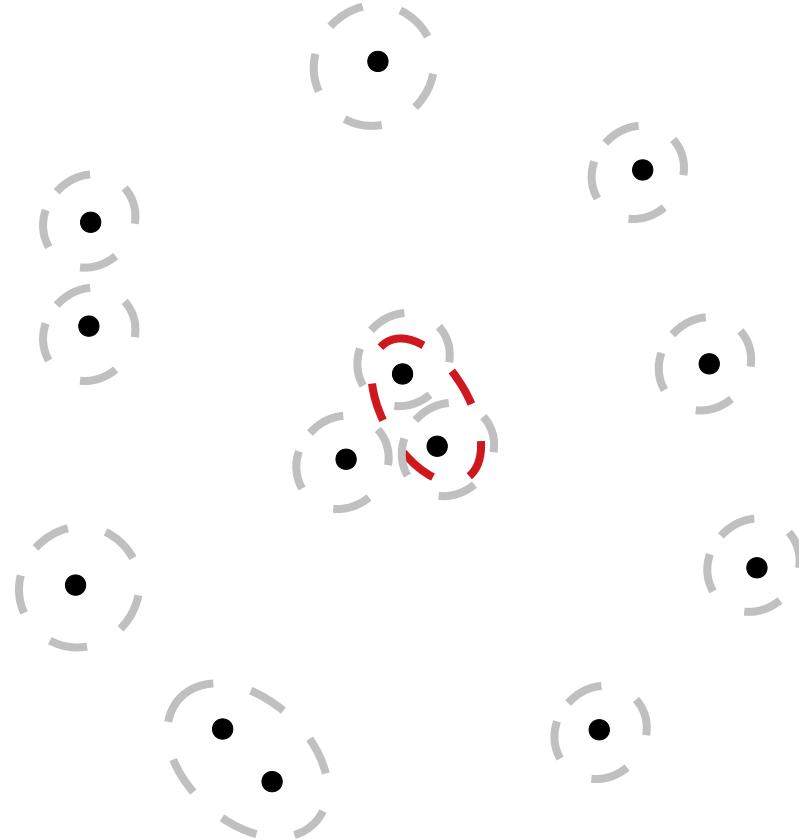
Hierarchical Agglomerative Clustering



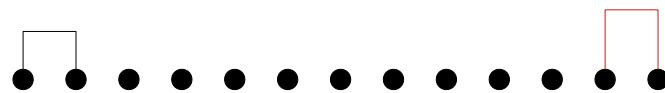
30



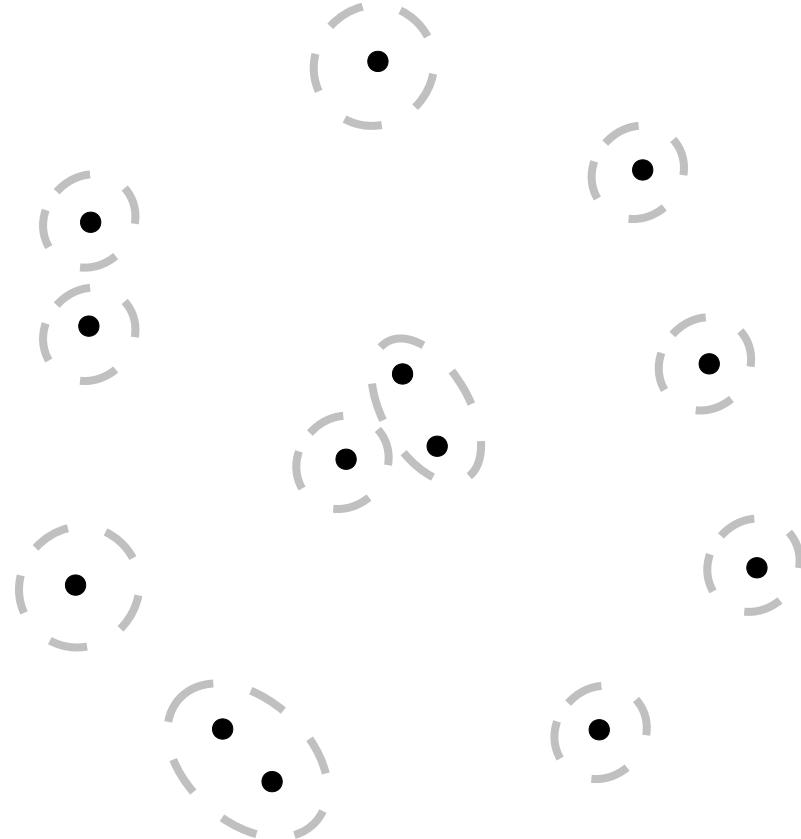
Hierarchical Agglomerative Clustering



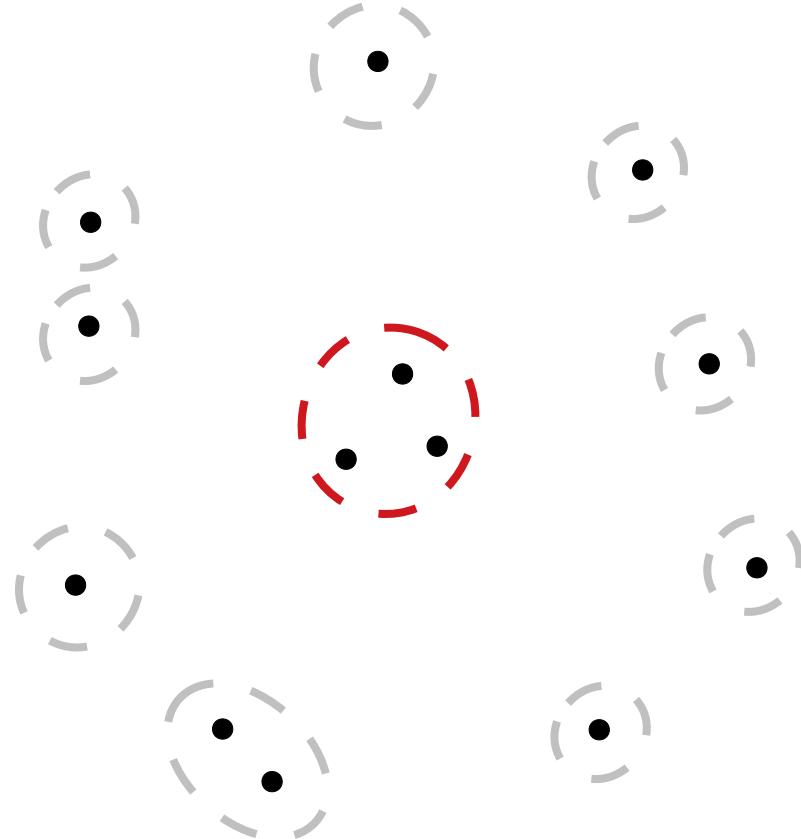
31



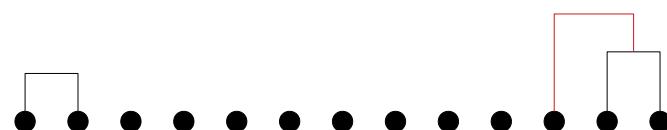
Hierarchical Agglomerative Clustering



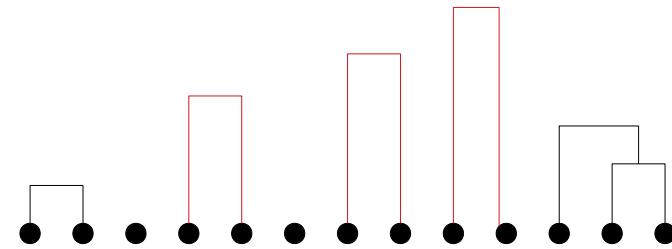
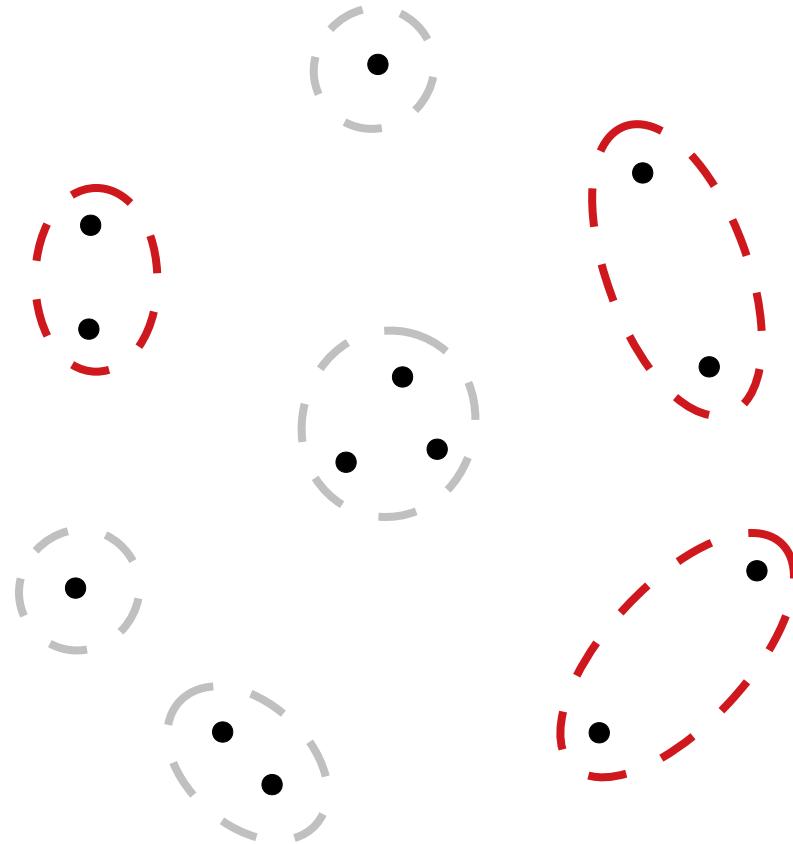
Hierarchical Agglomerative Clustering



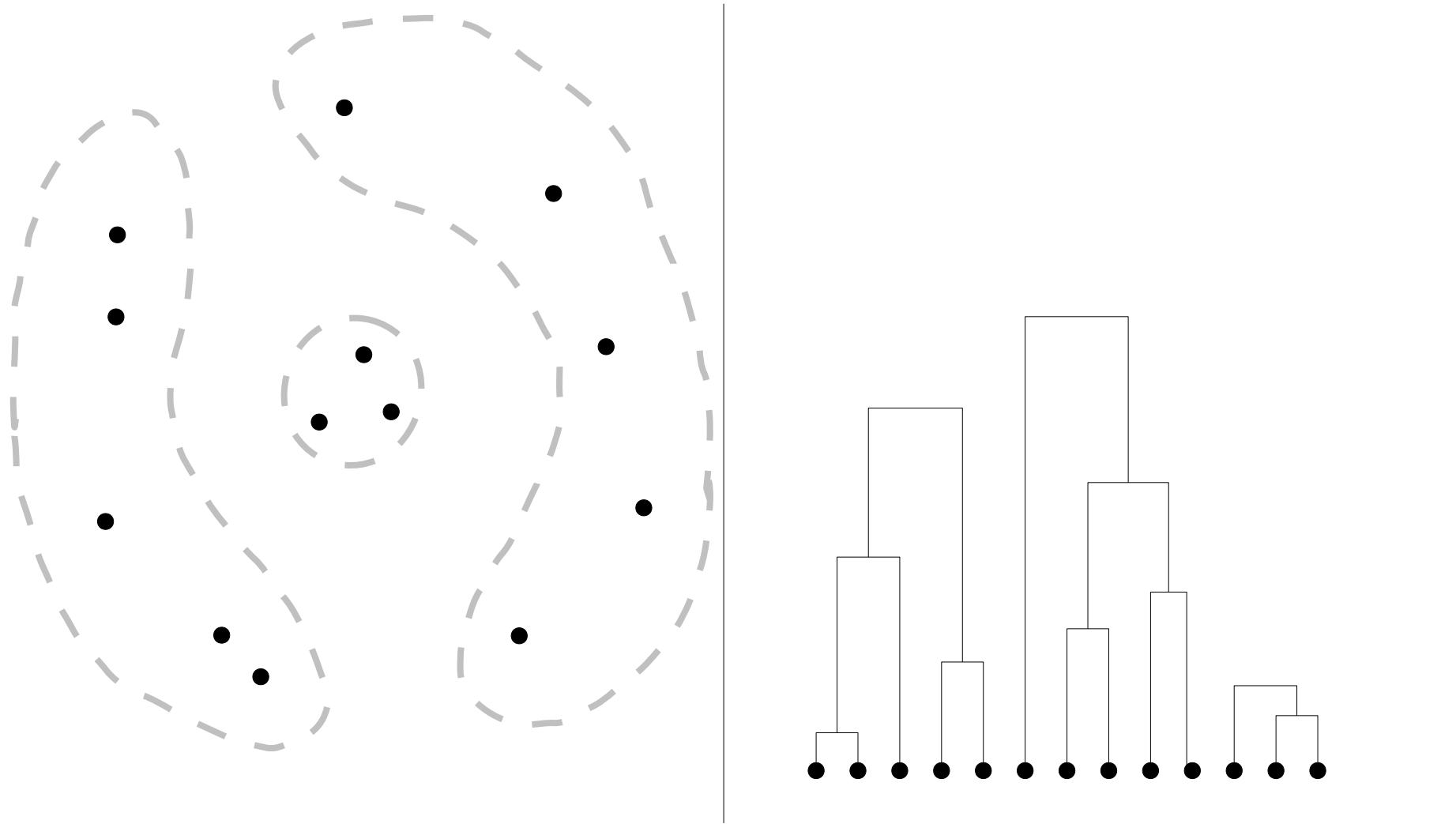
33



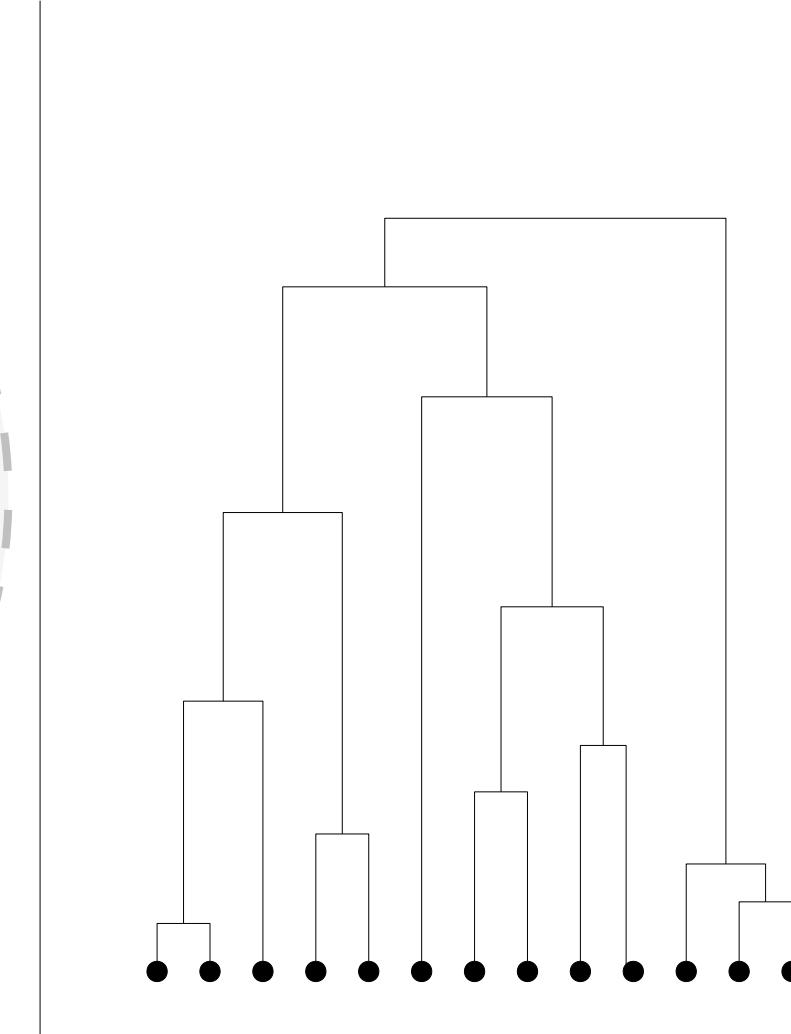
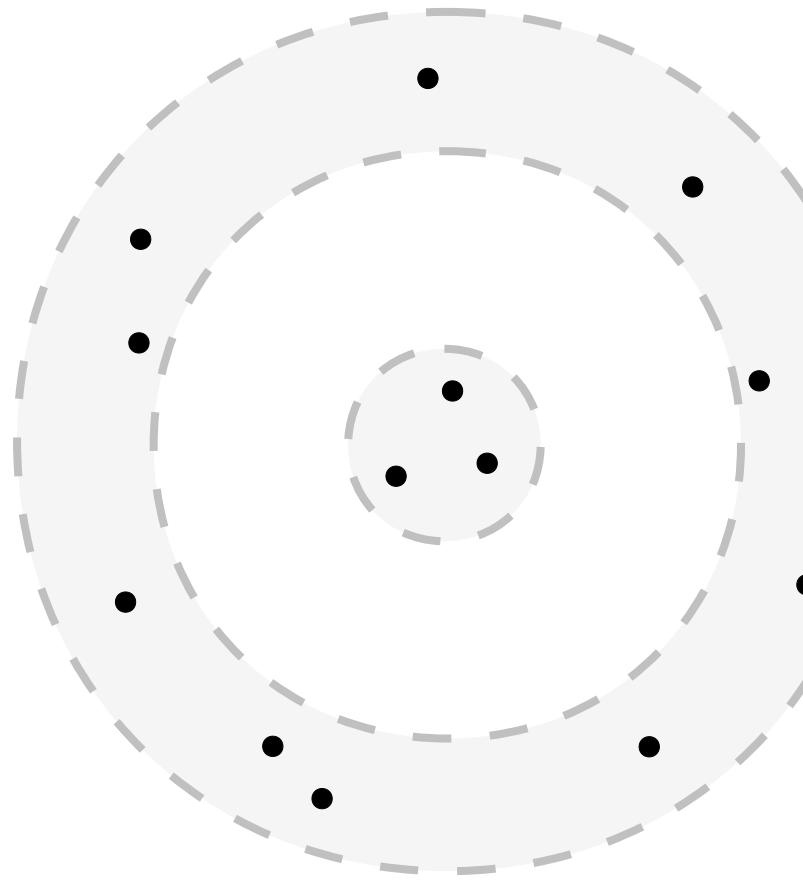
Hierarchical Agglomerative Clustering



Hierarchical Agglomerative Clustering



Hierarchical Agglomerative Clustering



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 Clusters 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 clusters in C
5. If $|C| > 1$, goto 3.



→ We need a method for computing distances between clusters!

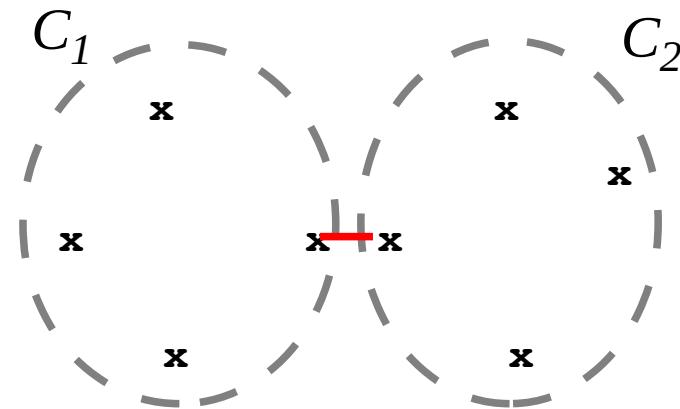
Similarity between Clusters

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

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



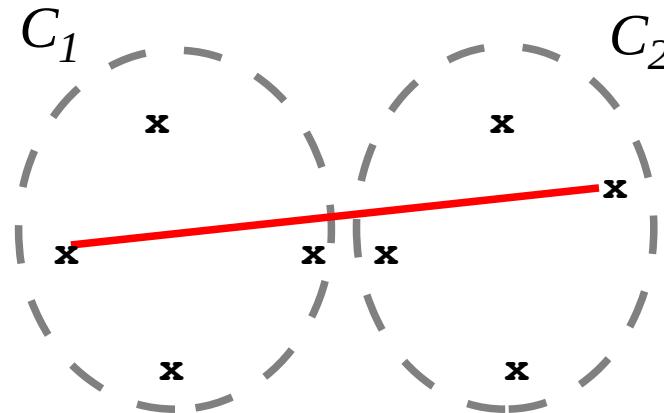
Similarity between Clusters

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

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



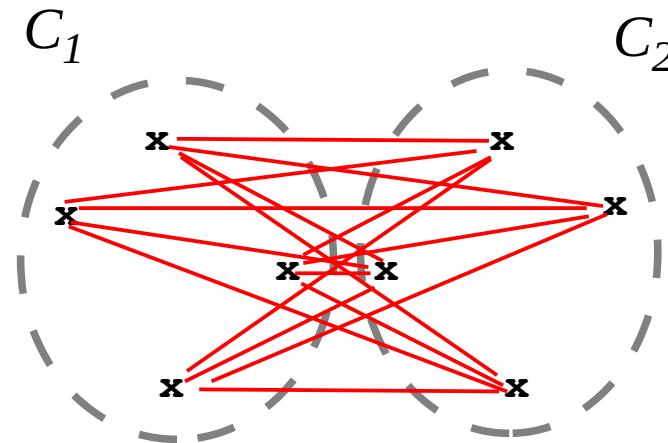
Similarity between Clusters

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

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

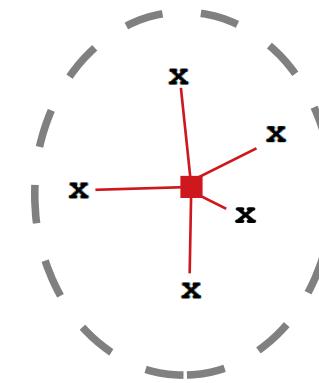


Representing Clusters with Centers

If a cluster could be represented with a single point, cluster similarity could be the distance of only two points (constant time)

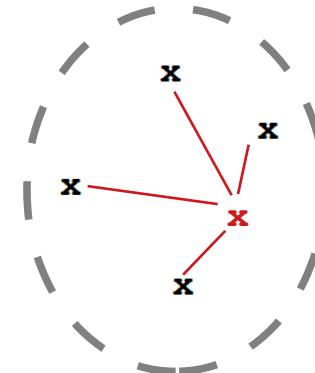
- **Centroids**

- **Mean/average point of all cluster points**
 - **point** that minimizes the distance to all cluster points
 - only in **Euclidian spaces!**



- **Clusteroids**

- **Median** point of all cluster points,
 - **cluster point** that minimize the distance to all other cluster points.



Computational Complexity

There are $n-1$ iterations

- in each iterations 2 clusters are merged

In each iteration, it compares all cluster distances to each other

- there are $(n-i+1)$ clusters at the start of the i -th iteration

$$\sum_{i=1}^{n-1} (n-i+1)^2 = \sum_{i=2}^n i^2 = \frac{1}{6}n(n+1)(2n+1) - 1 = O(n^3)$$



Computational Complexity (better)

There are $O(n)$ iterations

The distances from the previous iteration remain unchanged, we only need to recompute the $O(n)$ distances of the merged cluster to all other clusters

and keep them sorted in a priority queue $\rightarrow O(\log n)$

But $O(n^2 \log n)$ is still too much for big data...



Balanced Iterative Reducing and Clustering using Hierarchies

(Zhang, Ramakrishnan & Livny, 1996)

Efficient clustering algorithm

- does not assume that all examples are in main memory

Observation:

- Building trees bottom-up is useful
- but the lower levels of the tree (e.g., merging of single examples) are typically not interesting

Two Key Steps

1. **Compress the data into a tree structure** that fits into main memory
2. **Extract a clustering** from that tree structure

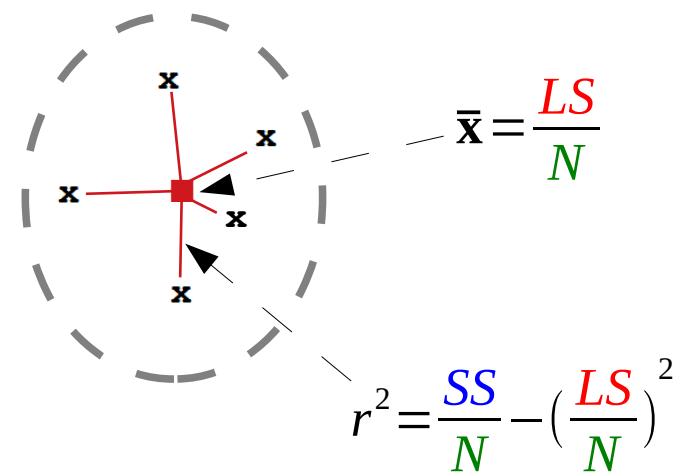


Cluster Feature

We cannot store all points of a cluster (in main memory)
→ we need to summarize information in each cluster

$$CF = \langle N, LS, SS \rangle$$

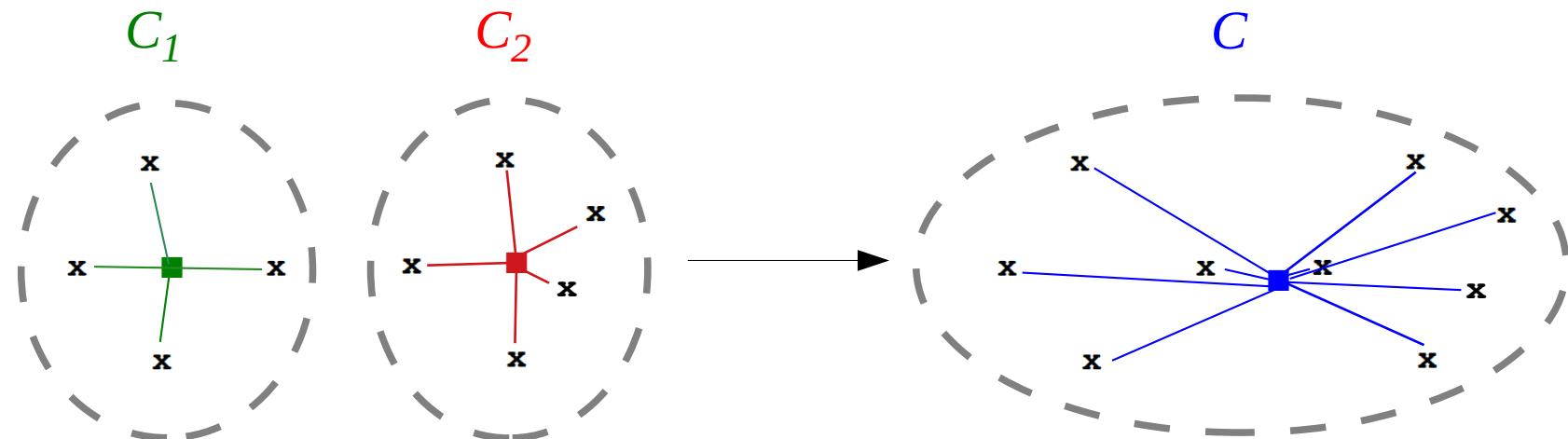
- N
 - number of points in the cluster
- $LS = \sum_{i=1}^N \mathbf{x}_i$
 - sum of points in the cluster
- $SS = \sum_{i=1}^N \mathbf{x}_i^2$
 - squared sum of points in the cluster



Cluster Merging

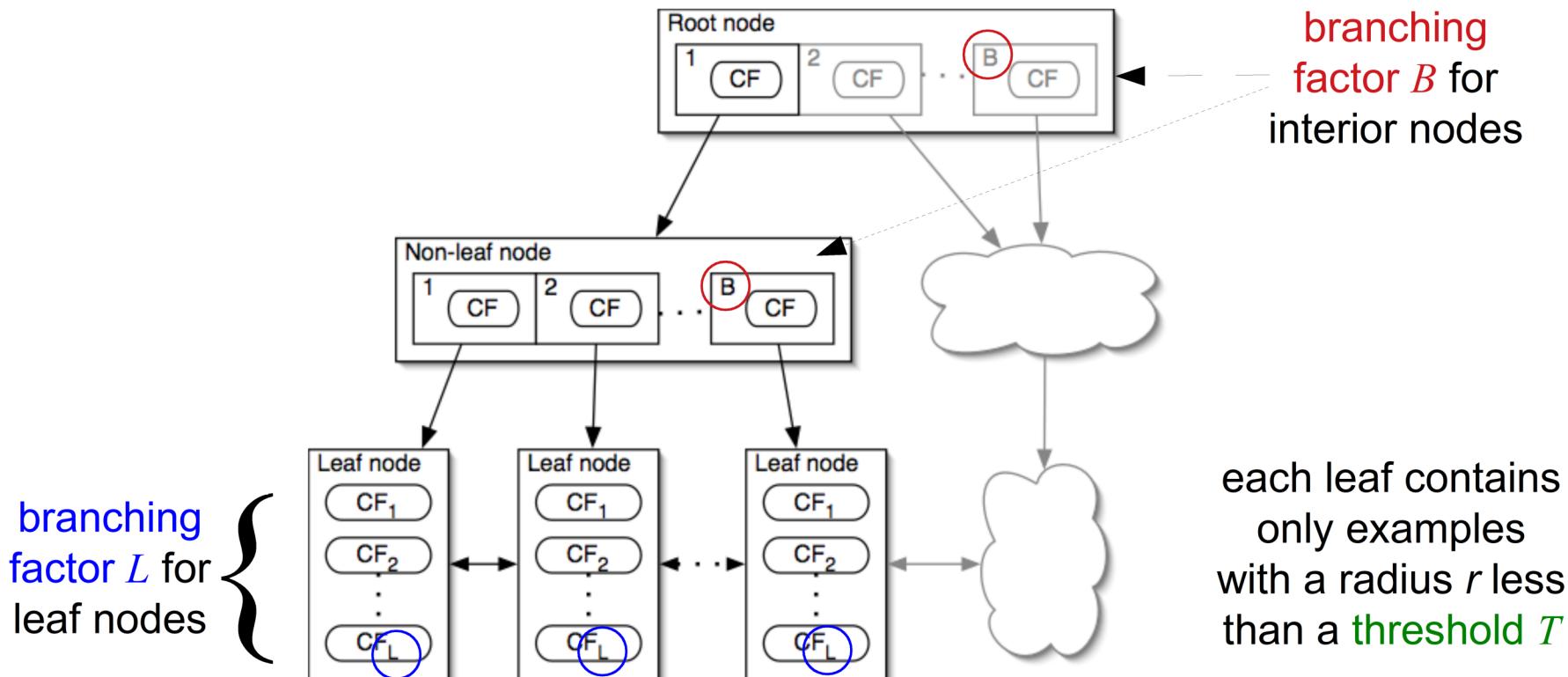
Merging clusters is trivial because of **Additivity Theorem**

$$\begin{aligned} C_1 \cup C_2 &= \langle N_1, LS_1, SS_1 \rangle \cup \langle N_2, LS_2, SS_2 \rangle \\ &= \langle N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2 \rangle = C \end{aligned}$$



BIRCH Cluster Tree

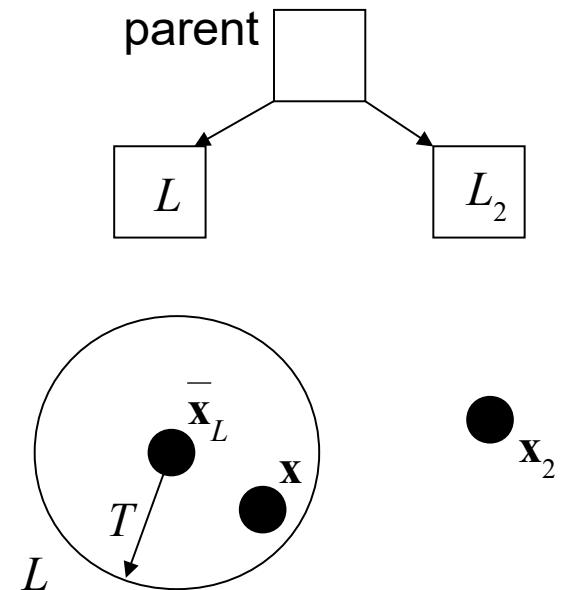
- balanced tree, similar to a B-tree



Data Compression by Building the CF Tree

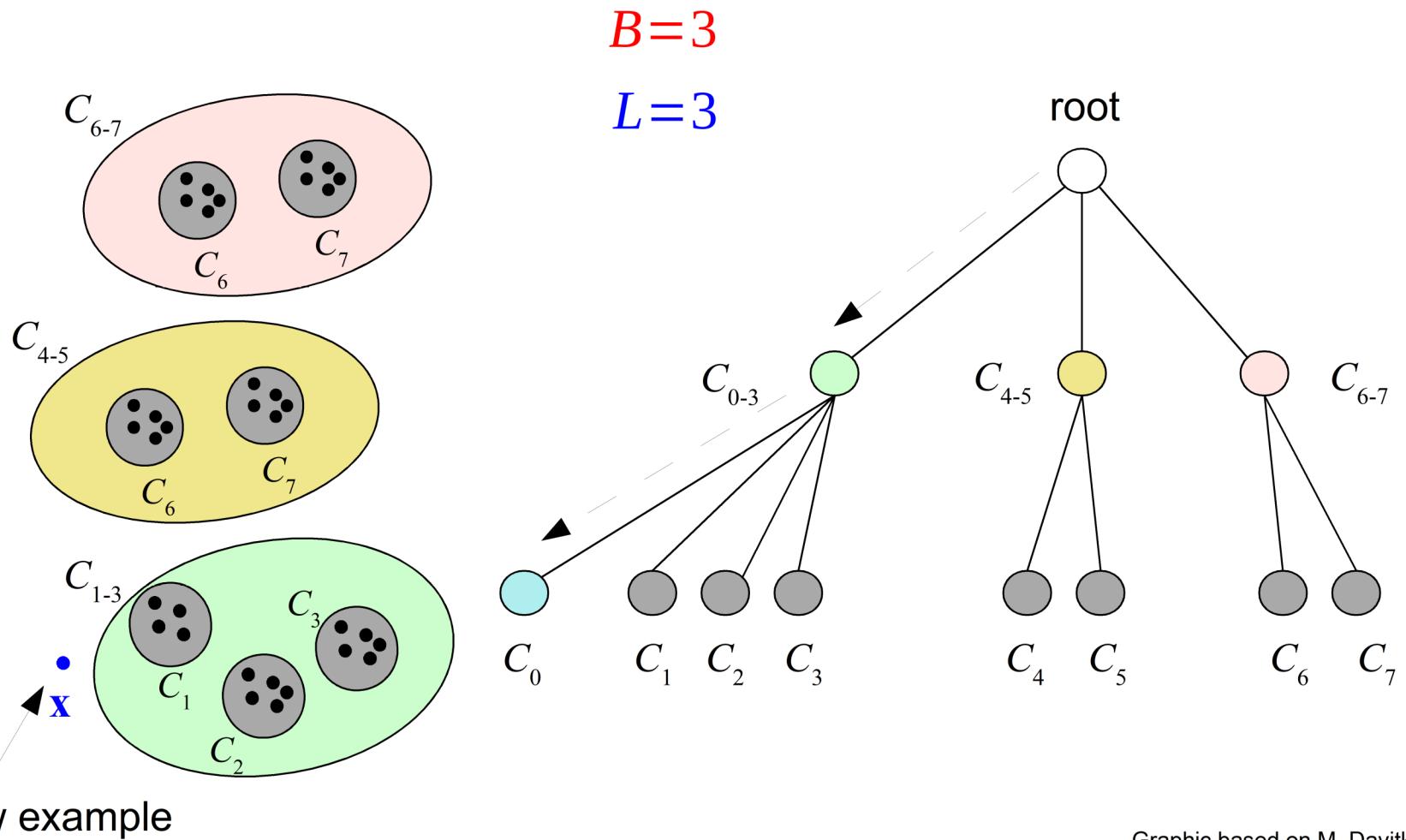
Phase 1: **Tree Construction:** For each example x do

- Identify the appropriate leaf L
 - recursively descend the CF tree and choose the child with the closest centroid until a leaf is reached
- If $d(x, \bar{x}_L) < T$
 - add the example to the node (update CF_L)
- Otherwise
 - split the node
- if number of nodes exceeds B
 - add a node at parent level
 - if necessary increase depth of tree
- Update the path
 - update CF information in every visited node

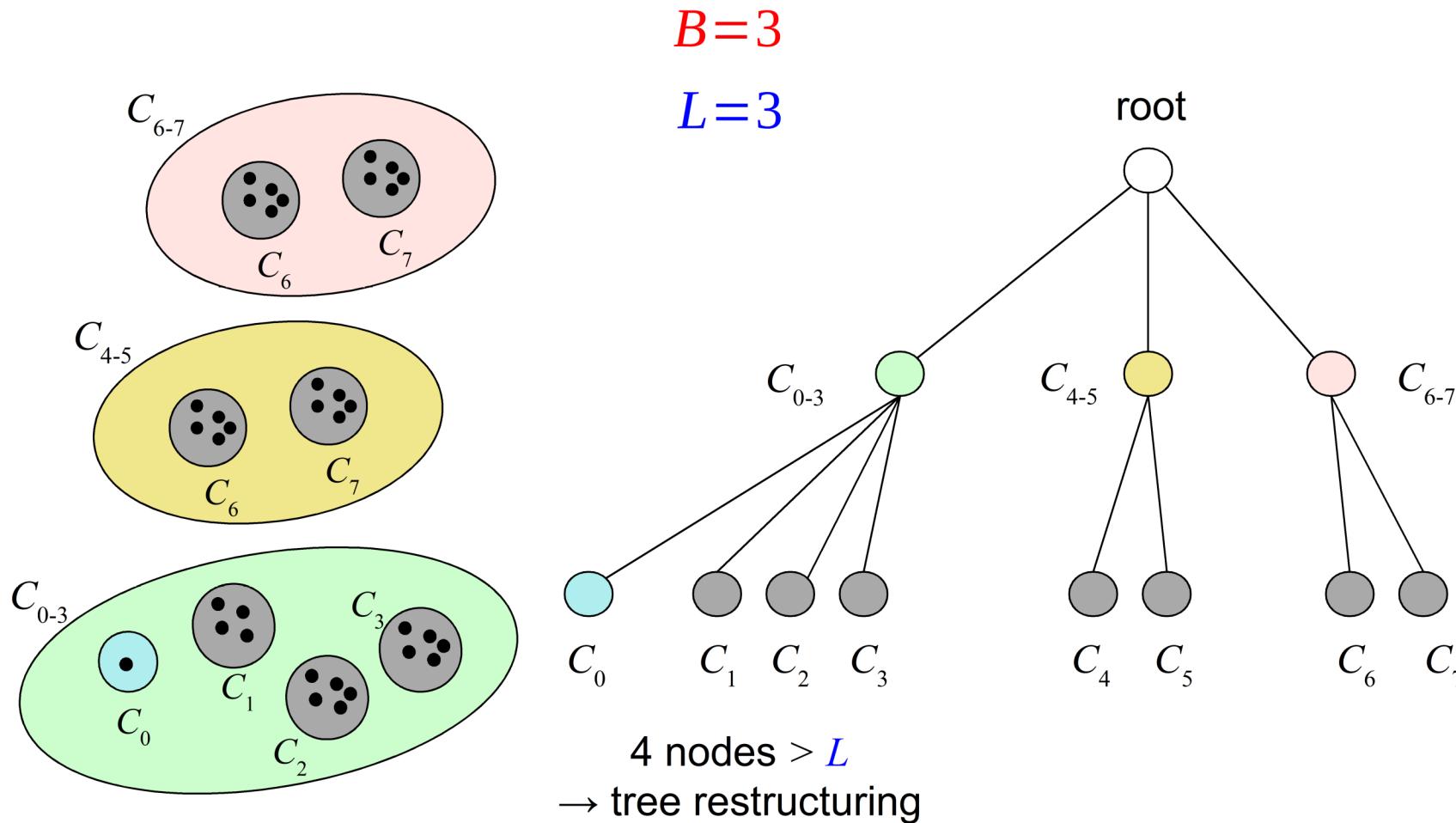


Graphic based on M. Davitkov

Restructuring of a CF Tree

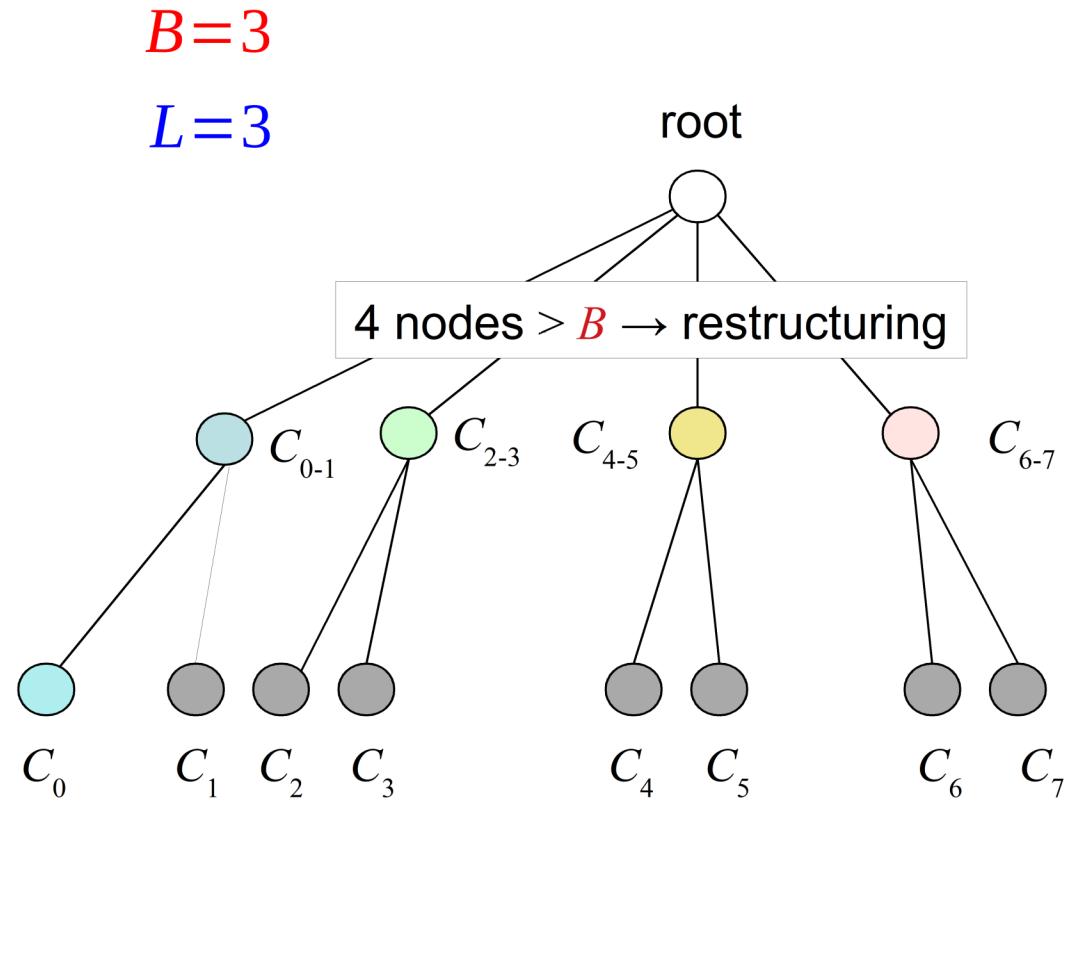
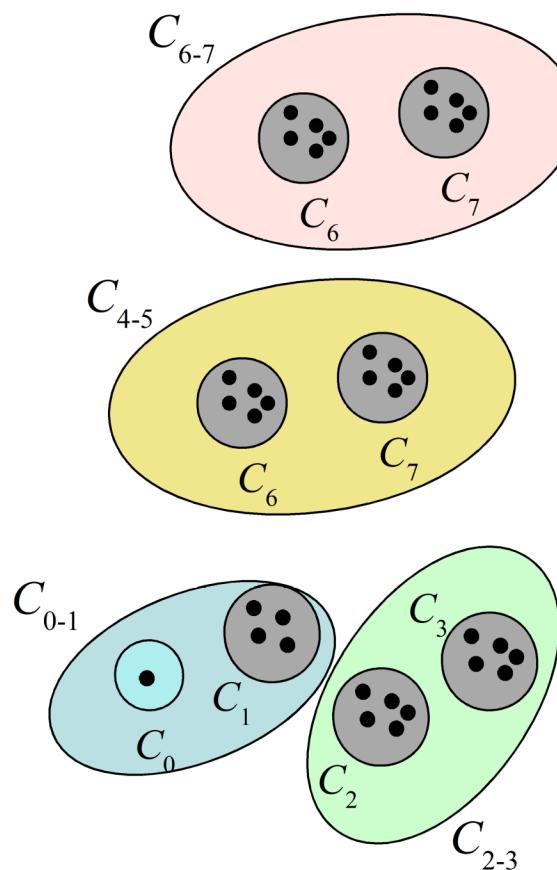


Restructuring of a CF Tree



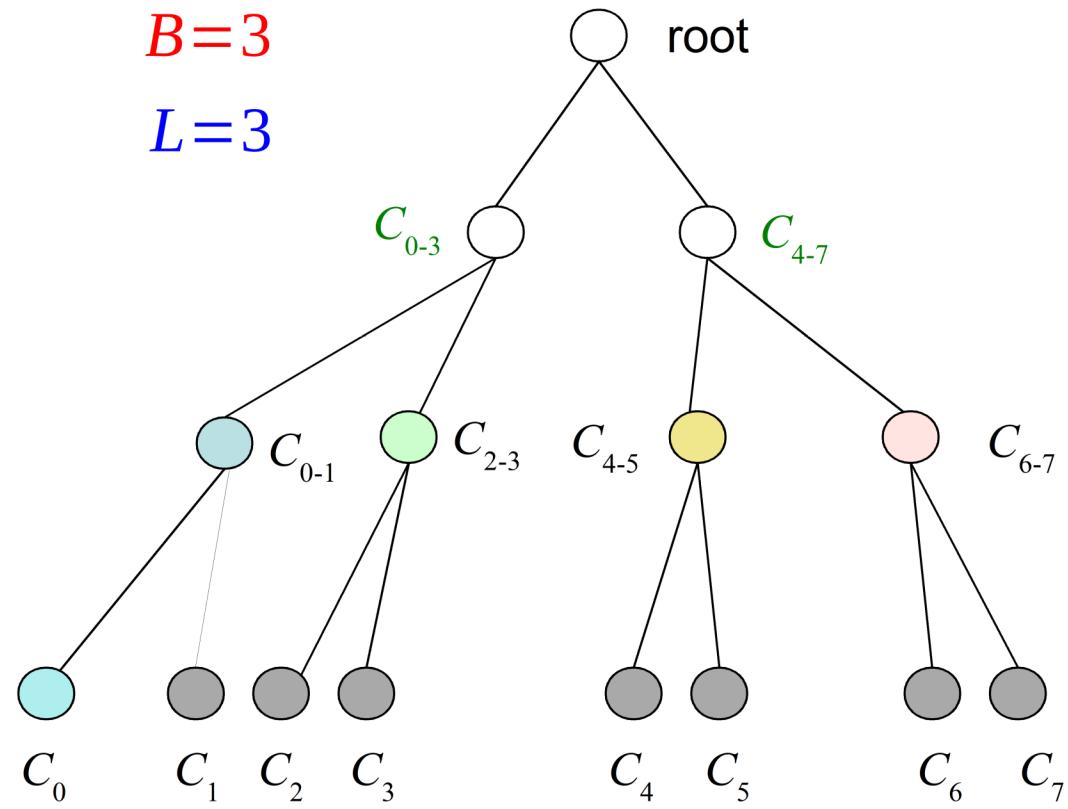
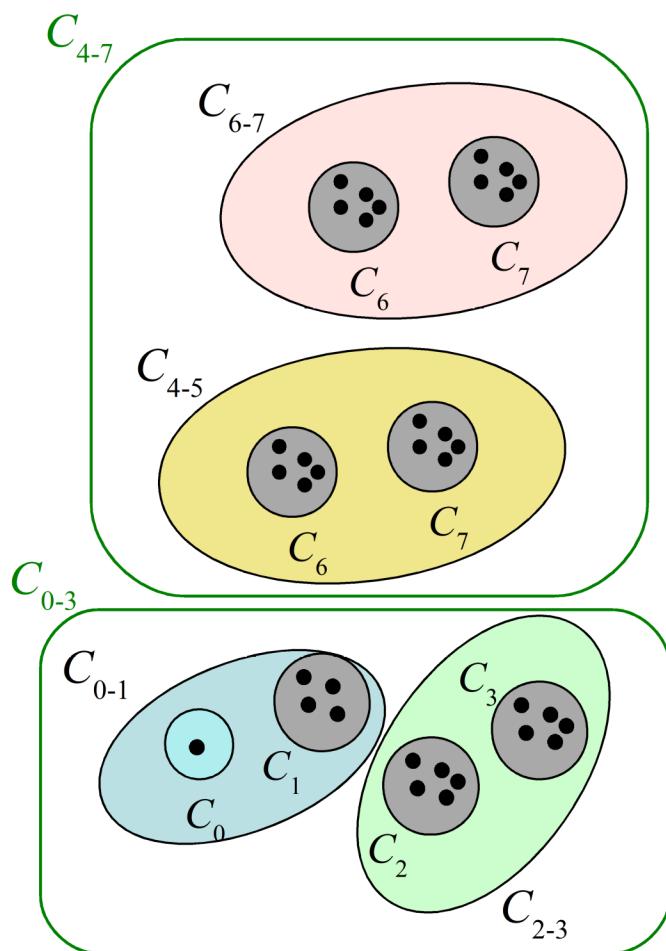
Graphic based on M. Davitkov

Restructuring of a CF Tree



Graphic based on M. Davitkov

Restructuring of a CF Tree



Graphic based on M. Davitkov

Threshold Parameter

- Crucial is the choice of the threshold T
 - If T is too large → initial clustering is very coarse
 - If T is too small → tree may not fit into memory

In the second case:

- If memory is exhausted, rebuild the tree with a larger T
 1. Start new empty tree
 2. Take every leaf of the old tree and insert it in the new tree
(→ some leafs may be joined in the process)
- Continue your pass through the examples

Choosing T :

- aim for doubling the number of processed examples
- crude approximation (d-dimensional sphere): $T_{i+1} = \sqrt[d]{2} \cdot T_i$



Further Compression

Phase 2: **Condense Tree** (optional)

- the tree resulting from phase 1 may be
 - too large
 - too fractured
 - inconsistent (e.g., the same example may end up in different leaves)
- Another pass for tree restructuring after the end of Phase 1 may help



Finding the Clustering

Phase 3: **Global Clustering**

- All clusters at the leaves are the input
- use any clustering algorithm for finding a final clustering
 - e.g., hierarchical agglomerative clustering
 - complexity not so bad, because the number of leaves is much smaller than N

Phase 4: **Iterative Improvement** (optional)

- use the clustering found in Phase 3 and perform EM-like improvement steps (analogous to k-means clustering)
 - assign each data point to nearest cluster center
 - compute new cluster centers
 - repeat ad libitum



Key Properties

Pros:

- Clustering decisions are local (should a node be split or not)
- First clustering can be obtained with a single pass through the data (incremental algorithm)

Neutral:

- Global optimization requires additional passes through the data

Cons:

- Assumes Euclidian space (centroids must exist)
- Assumes elliptic shapes of clusters
- Result depends on order of the examples



BUBBLE – Adapting BIRCH to general spaces

(Ganti et al. 1999)

We need to remember

- **points** in order to be able to compute a clusteroid
- **distances** between points in order to estimate cluster distances

BUBBLE stores in a node

- N : number of points in the cluster
- \mathbf{q} : the cluster center (clusteroid) of the cluster (corresponds to LS)
- $\text{RowSum}(\mathbf{q}) = \sum_{i=1}^N d^2(\mathbf{q}, \mathbf{x}_i)$
 - sum of squared distances to points in the cluster (corresponds to SS)
- a sample of points and their **RowSums**
 - **MinSet**: the k points closest to \mathbf{q}
 - **MaxSet**: the k points most distant from \mathbf{q}



Note: \mathbf{q} is the point with the minimum RowSum of all cluster points

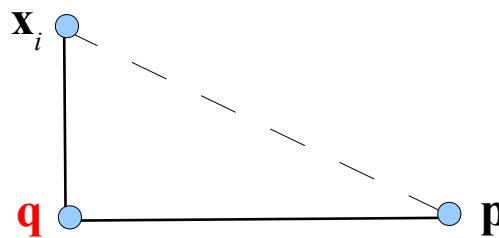
Node Update for a new instance p

1. approximate the RowSum for p:

$$\text{RowSum}(\mathbf{p}) \approx \text{RowSum}(\mathbf{q}) + N \cdot d^2(\mathbf{p}, \mathbf{q})$$

Intuition behind the approximation:

$$\begin{aligned}\text{RowSum}(\mathbf{p}) &= \sum_{i=1}^N d^2(\mathbf{p}, \mathbf{x}_i) \approx \sum_{i=1}^N (d^2(\mathbf{p}, \mathbf{q}) + d^2(\mathbf{q}, \mathbf{x}_i)) \\ &= N \cdot d^2(\mathbf{p}, \mathbf{q}) + \text{RowSum}(\mathbf{q})\end{aligned}$$



Note: in very high dimensional spaces, point angles are almost always close to orthogonal (**Curse of Dimensionality**).

2. increase the count:

$$N = N + 1$$

3. update the $2k + 1$ RowSums: $\text{RowSum}(\mathbf{x}) = \text{RowSum}(\mathbf{x}) + d^2(\mathbf{x}, \mathbf{p})$

Update of Cluster Structure

- Add point
 - if $\text{RowSum}(\mathbf{p})$ is small (large) enough, it replaces the max (min) distance point among the k **MinSet** (**MaxSet**) points
- New clusteroid
 - if after step 3. $\text{RowSum}(\mathbf{x}) < \text{RowSum}(\mathbf{q})$ for one of the k **MinSet** points \mathbf{x} , then \mathbf{x} and \mathbf{q} swap their roles
- Splitting of cluster
 - if cluster radius $r = \sqrt{\text{RowSum}(\mathbf{q})/N}$ too large, the cluster is split
 - divide the cluster so that the RowSums in each part are minimized
- Merging of clusters
 - if tree size exceeds memory, nodes need to be merged
 - for this the **MaxSet** points are considered as clusteroids



Characteristica of BUBBLE

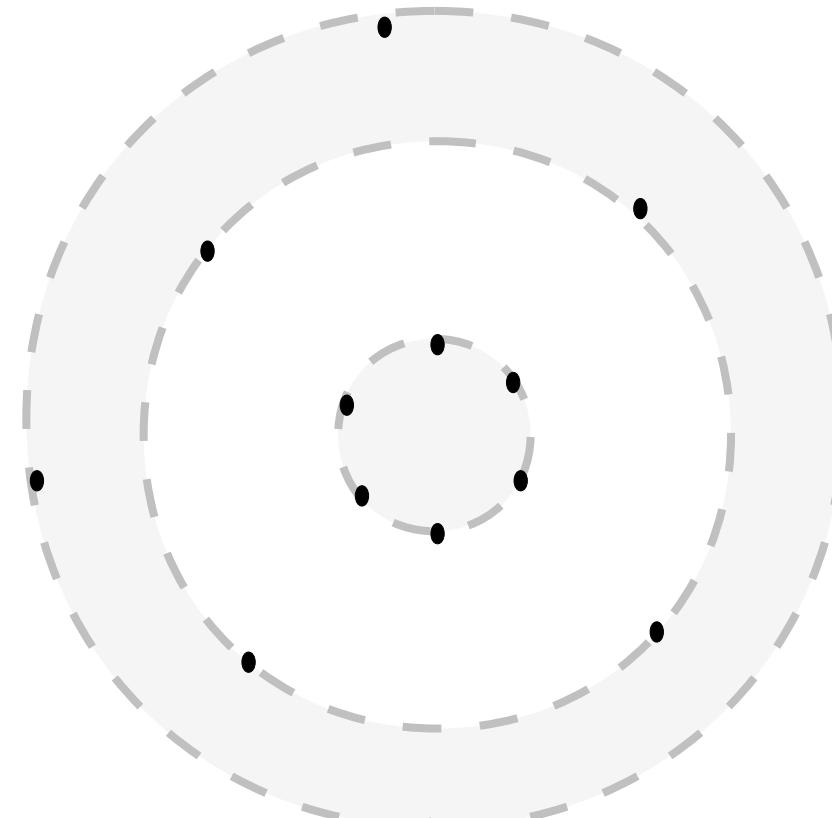
- Needs more storage space than BIRCH
 - In each cluster, we need to store $2k$ points (or pointers to disk space) plus the corresponding RowSums
 - but does not depend on shape of clusters or distances
 - only produces an approximate solution
 - Uses multiple points to represent a cluster (\rightarrow CURE)



Key idea of CURVE

(Guha, Rastogi & Shim, 1998)

- Represent each cluster with multiple points
 - which are chosen in a way to maximize distances

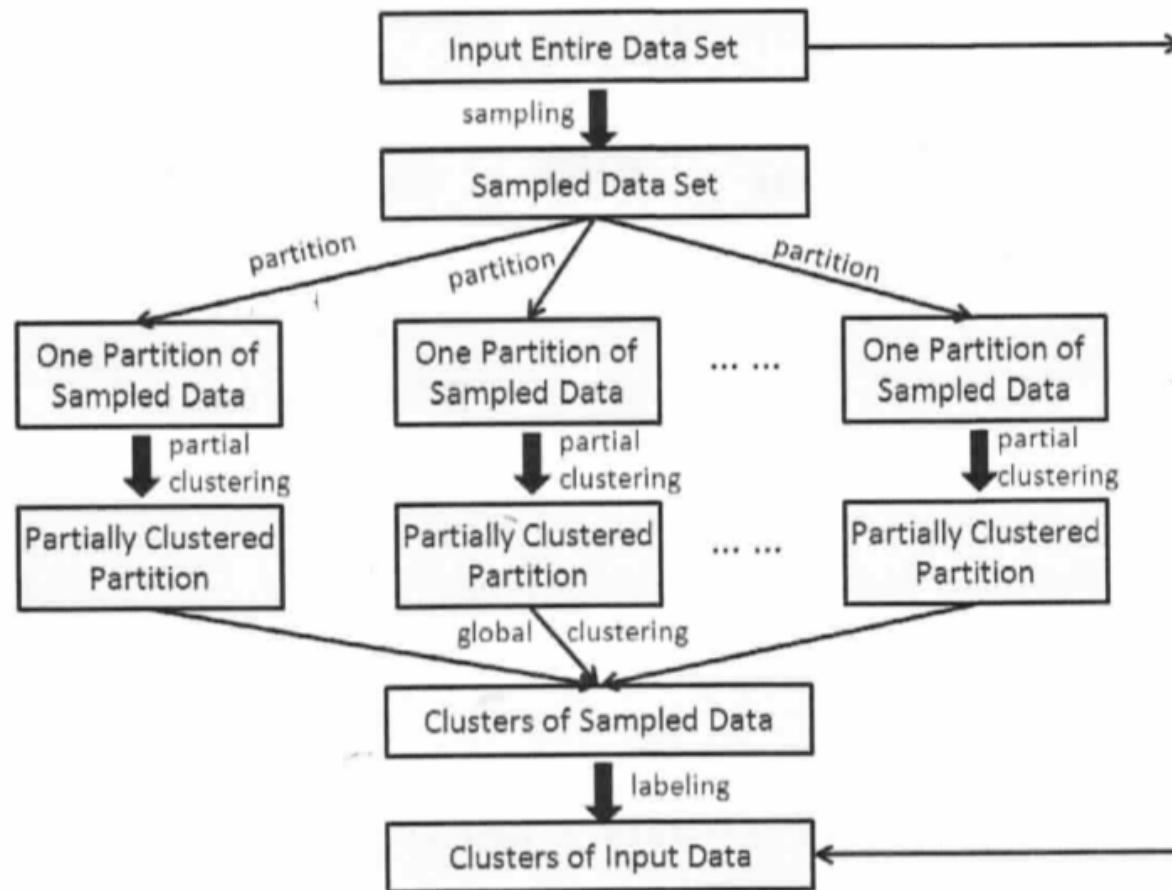


Other Ideas in CURE

- Shrinking:
 - representative points are contracted by a factor α towards the centroid:
 - $\alpha = 1$ corresponds to using all points as given
 - $\alpha = 0$ corresponds to replacing all points with a clusteroid
 - idea is to fight outliers
- Identifying outlier clusters:
 - Outliers are isolated points, so they will get few members → can be identified during growing (slow growth) or removed at the end
- Random Sampling / Partitioning
 - In order to make it more efficient, the algorithm uses a random sample or multiple partitions for finding the clusters
 - and then assigns the remaining points to the cluster (points are assigned to the cluster with the closest representative)



CURE algorithm



Picture taken from (Aggarwal & Reddy, 2014)



Properties of CURE

- use of multiple points for representing a cluster → allows to fit arbitrary shapes
- efficiency via the use of sampling or partitioning
- outliers are handled quite well



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.clusty.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

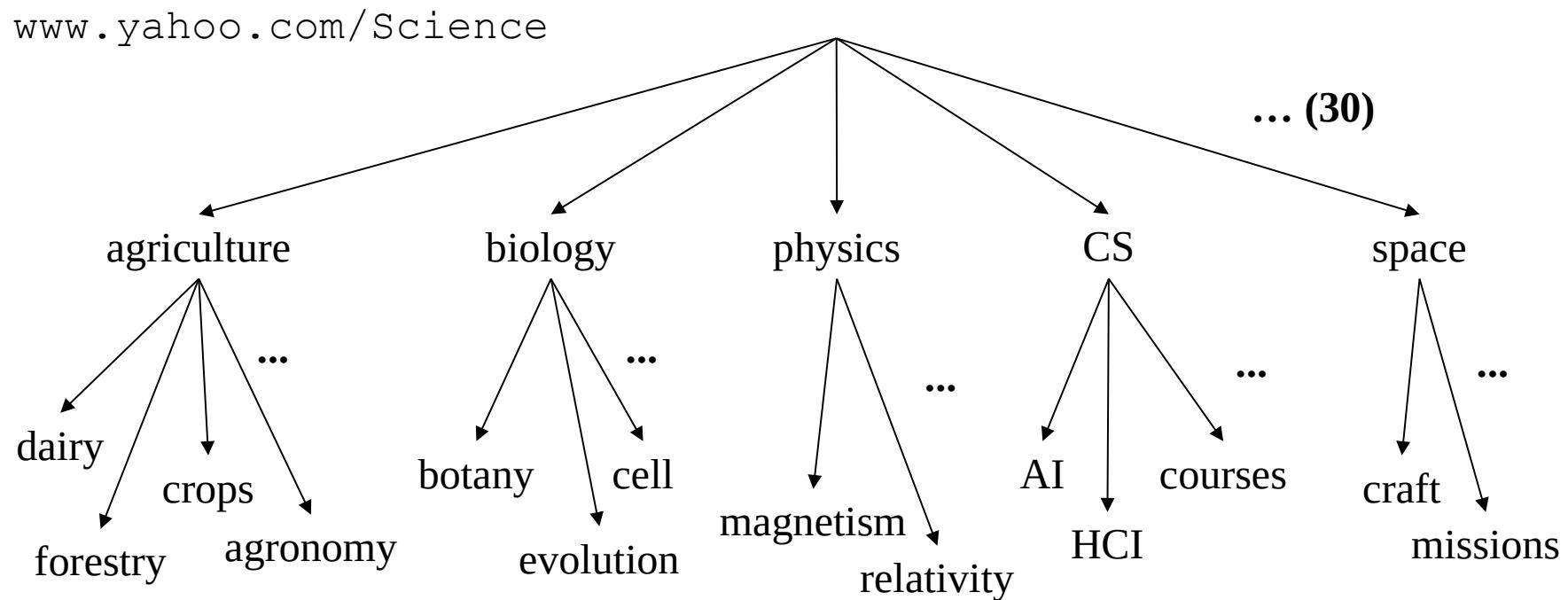
- For grouping search results thematically
- clusty.com / Vivisimo

The screenshot shows the Clusty search interface. At the top, there's a navigation bar with tabs for Web+, News, Images, Shopping, Encyclopedia, Gossip, and Customize!. Below the bar is a search input field containing "text clustering". To the right of the input is a blue "Cluster" button. Underneath the input field are links for Advanced Search, Help, Tell a Friend, Tell us what you think!, and Download the Clusty Toolbar. On the left side, there's a sidebar titled "Cluster by:" with a dropdown menu set to "Topics". Below this, a list of topics is shown, each preceded by a plus sign and a link: Mining (30), Gene (19), Clustering algorithms (13), Text Document Clustering (8), Receptor (13), Lab (6), Model, Self-Organising Hybrid (9), Ontology-based Text Clustering (6), and Text Categorization (7). A large green arrow points from the bottom left towards the sidebar. The main content area displays the search results for "text clustering". It says "Top 187 results of at least 251,221 retrieved for the query [text clustering](#) (Details)". There are two sections of results: "Sponsored Res" and "Search Res". The first result under "Sponsored Res" is "Apple 64-bit Xserve G5 - Clustering" with a snippet about it. The second result is "Custom Configured Clustering Solutions" with a snippet. The first result under "Search Res" is "Delphion: Text Clustering" with a snippet. The second result is "Vivisimo Clustering - automatic categorization and meta-search software -" with a snippet.

- Top 187 results of at least 251,221 retrieved for the query [text clustering](#) (Details)
- Sponsored Res**
- [Apple 64-bit Xserve G5 - Clustering](#)
High-density, affordable server with unparalleled functionality and usability. Featuring massive processing power, high-speed I/O and remote management tools that make it easy to deploy.
www.apple.com
- [Custom Configured Clustering Solutions](#)
Aspen Systems custom designs, manufactures and supports Beowulf Clusters. www.aspsys.com
- Search Res**
- 1. [Delphion: Text Clustering](#)
... Citation Link » Clustering « » PatentLab-II » PDF Express » Data Extract » Other Services **Text Clustering** Overview Delphion's **Text Clustering** transforms obscure, textual information into useful knowledge. ...
www.delphion.com/products/research/products-cluster - Wisenut, GigaBlast, Lycos, MSN
- 2. [Vivisimo Clustering - automatic categorization and meta-search software -](#)
...Vivisimo's document **clustering** and meta-search software....Publisher... Cluster largest text collections. ...
www.vivisimo.com - organization with document clustering, english, ...



Application: Build up a Web Catalogue



Application: Build up a Web Catalogue

dmoz open directory project

In partnership with **AOL Search.**

[about dmoz](#) | [dmoz blog](#) | [suggest URL](#) | [help](#) | [link](#) | [editor login](#)

 [advanced](#)

Arts

[Movies](#), [Television](#), [Music](#)...

Games

[Video Games](#), [RPGs](#), [Gambling](#)...

Kids and Teens

[Arts](#), [School Time](#), [Teen Life](#)...

Reference

[Maps](#), [Education](#), [Libraries](#)...

Shopping

[Clothing](#), [Food](#), [Gifts](#)...

World

[Català](#), [Dansk](#), [Deutsch](#), [Español](#), [Français](#), [Italiano](#), [日本語](#), [Nederlands](#), [Polski](#), [Русский](#), [Svenska](#)...

Business

[Jobs](#), [Real Estate](#), [Investing](#)...

Health

[Fitness](#), [Medicine](#), [Alternative](#)...

News

[Media](#), [Newspapers](#), [Weather](#)...

Regional

[US](#), [Canada](#), [UK](#), [Europe](#)...

Society

[People](#), [Religion](#), [Issues](#)...

Computers

[Internet](#), [Software](#), [Hardware](#)...

Home

[Family](#), [Consumers](#), [Cooking](#)...

Recreation

[Travel](#), [Food](#), [Outdoors](#), [Humor](#)...

Science

[Biology](#), [Psychology](#), [Physics](#)...

Sports

[Baseball](#), [Soccer](#), [Basketball](#)...

[Become an Editor](#)

Help build the largest human-edited directory of the web



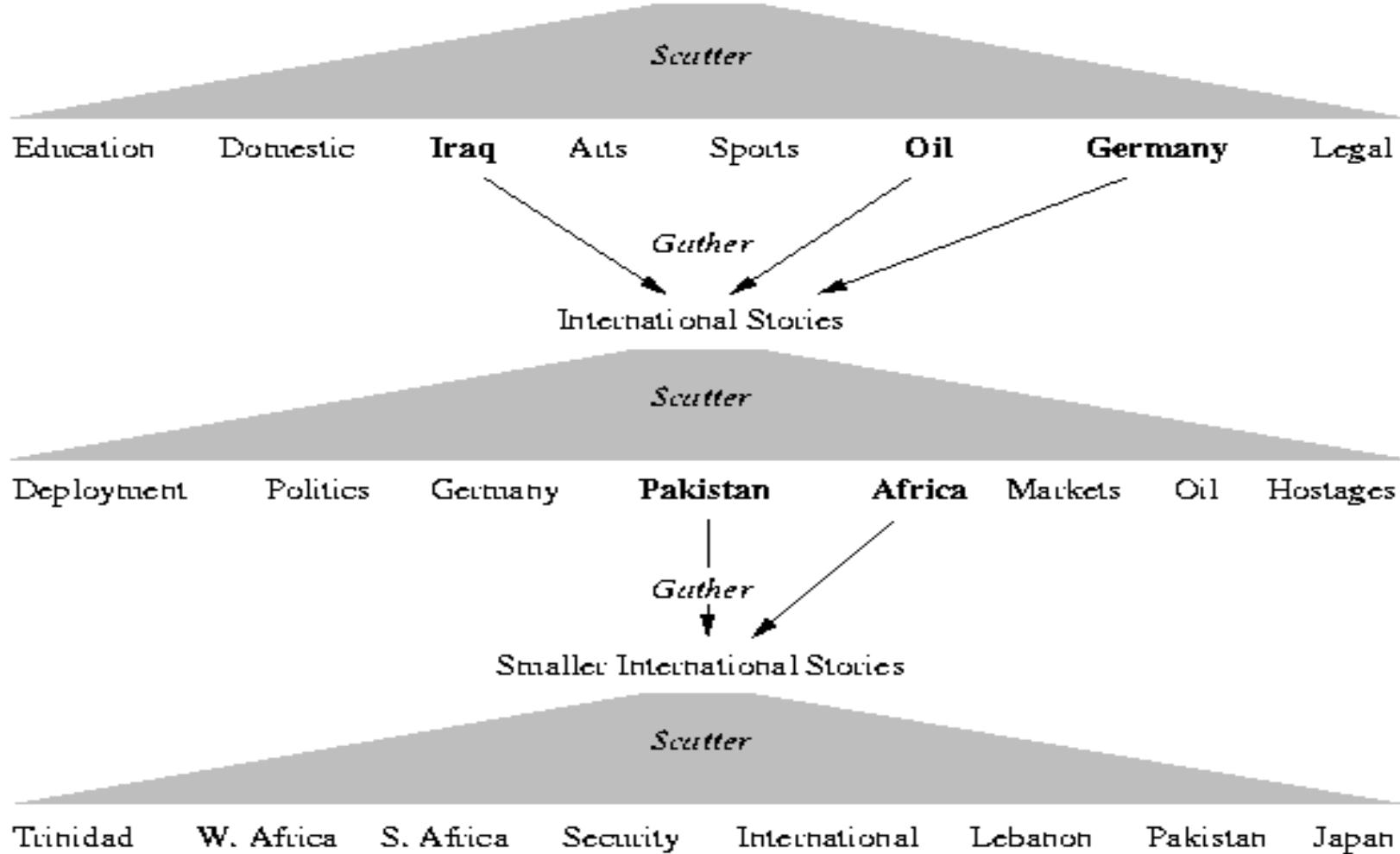
Copyright © 1998-2010 Netscape

4,529,282 sites - 85,446 editors - over 590,000 categories



Browsing Documents: Scatter/Gather (Cutting, Karger, and Pedersen)

New York Times News Service, August 1990



Now you should know...

- What is the difference between clustering and classification?
- How does k-means clustering work?
- What is the difference between divisive and agglomerative clustering?
- Why is the complexity of HAC prohibitive for big data?
- What is the key idea of BIRCH for reducing the complexity?
- What cluster features does it remember and why?
- Will it produce exactly the same result as HAC?
- How many passes through the data does it require?
- How to choose a good threshold, what are the trade-offs?
- How can it be adapted to non-convex shapes?
- How can it be adapted to non-Euclidean spaces?



Literature

- A. Rajaraman, J.D. Ullman: *Mining of Massive Datasets*, Cambridge University Press, 2012.
 - C. Aggarwal, C.K. Reddy (eds.) *Data Clustering – Algorithms and Applications*, CRC Press 2014.
-
- V. Ganti, R. Ramakrishnan, J. Gehrke, A.L. Powell, J. C. French: Clustering large datasets in arbitrary metric spaces, Proc. ICDE, pp. 502-511, 1999.
 - S. Guha, R. Rastogi, K. Shim: CURE: An efficient clustering algorithm for large databases. Information Systems 26(1):35-58, 2001.
 - T. Zhang, R. Ramakrishnan, M. Livny: BIRCH: An efficient data clustering method for very large databases. Proc. SIGMOD, pp. 103-114, 1996.

