Clustering

Danilo Croce

Web Mining & Retrieval a.a. 2019/2020

29/04/2020

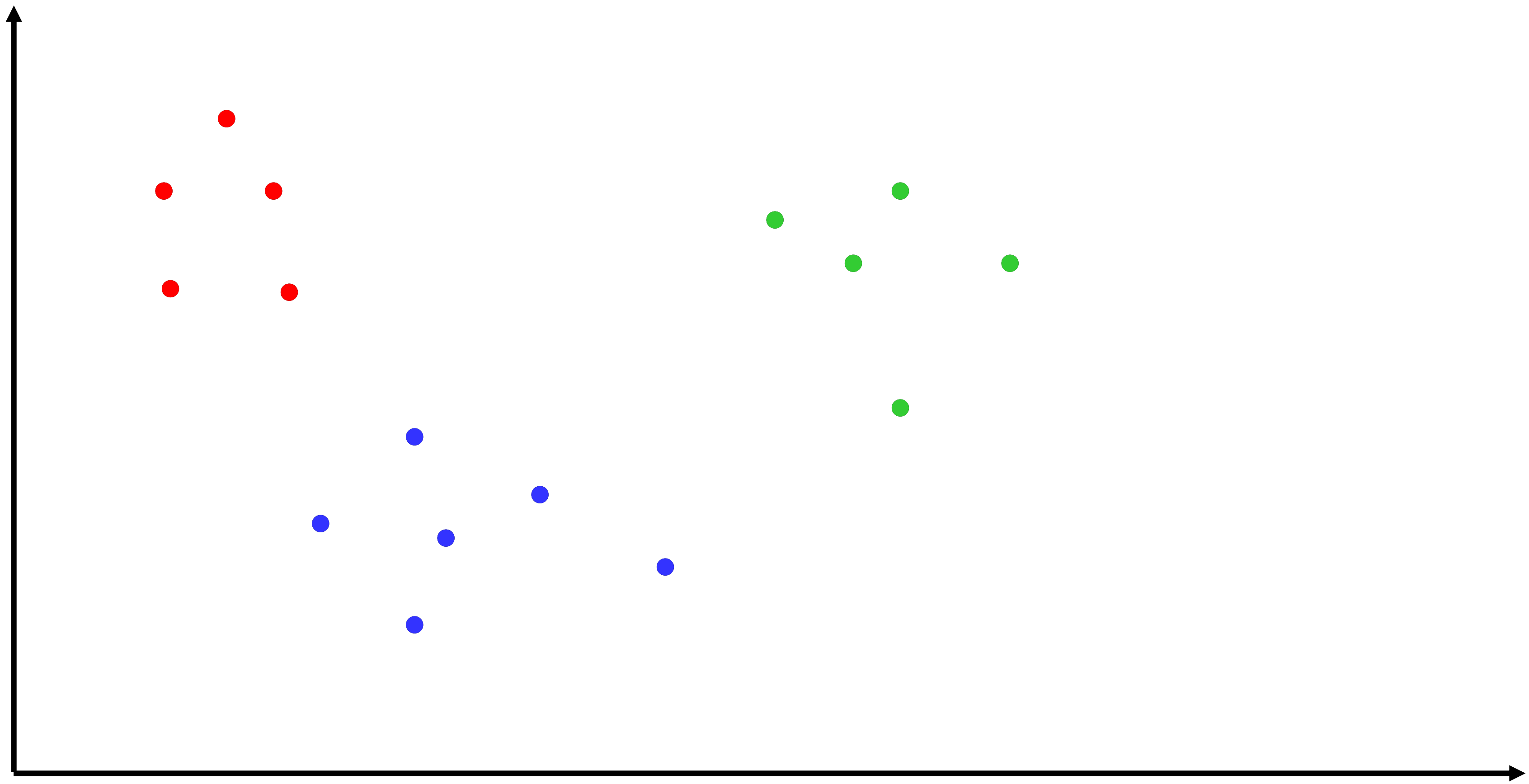
Supervised learning vs. unsupervised learning

- *Supervised* learning: discover patterns in the data that relate data attributes with target attributes
  - These patterns are then utilized to predict the values of target attributes in future data instances
- *Unsupervised* learning: The data have no target attribute
  - Find some intrinsic structures in data

Clustering

- Partition unlabeled examples into disjoint subsets of *clusters*, such that:
  - Examples within a cluster are very similar (*infra-cluster* similarity)
  - Examples in different clusters are very different (*inter-cluster* dissimilarity)
- Discover new categories in an *unsupervised* manner
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.
  - In fact, association rule mining is also unsupervised

Clustering Example

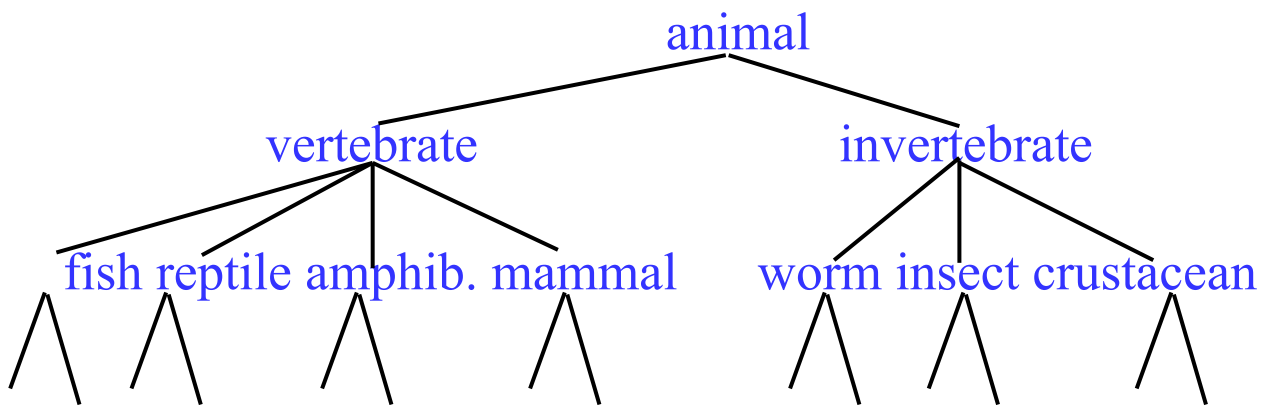


Application examples

- Example 1: In marketing, segment customers according to their similarities
  - To do targeted marketing
- Example 2: Given a collection of text documents, we want to organize them according to their content similarities
  - To produce a topic hierarchy

Hierarchical Clustering

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



- Recursive application of a standard clustering algorithm can produce a hierarchical clustering.

Direct Clustering

- *Direct clustering* methods require a specification of the number of desired clusters, k
- A *clustering evaluation function* assigns a real-value quality measure to a clustering.
- The number of clusters can be determined automatically by explicitly generating clusterings for multiple values of k and choosing the best result according to a clustering evaluation function.

Aspects of clustering

- A clustering algorithm
  - Single Link Agglomerative Clustering
  - K-Means
  - …
- A distance (similarity, or dissimilarity) function
- Clustering quality
  - Inter-clusters distance ⇒ maximized
  - Intra-clusters distance ⇒ minimized
- The quality of a clustering result depends on the algorithm, the distance function, and the application

Hierarchical Clustering: Agglomerative vs. Divisive Clustering

- *Agglomerative* (*bottom-up*) methods start with each example in its own cluster and iteratively combine them to form larger and larger clusters

- *Divisive* (*partitional, top-down*) : It starts with all data points in one cluster, the root. Splits the root into a set of child clusters. Each child cluster is recursively divided further

Hierarchical Agglomerative Clustering (HAC)

- Assumes different *similarity functions* for determining the similarity of two instances
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster
- The history of merging forms a binary tree or hierarchy

HAC Algorithm

Start with all instances in their own cluster.
Until there is only one cluster:

 Among the current clusters, determine the two clusters, c_i and c_j, that are most similar.

 Replace c_i and c_j with a single cluster $c_i \cup c_j$

HAC Algorithm: Partition based on Cluster Similarity

- How to compute similarity of two clusters each possibly containing multiple instances?
  - Single Link: Similarity of two most similar members
  - Complete Link: Similarity of two least similar members
  - Group Average: Average similarity between members

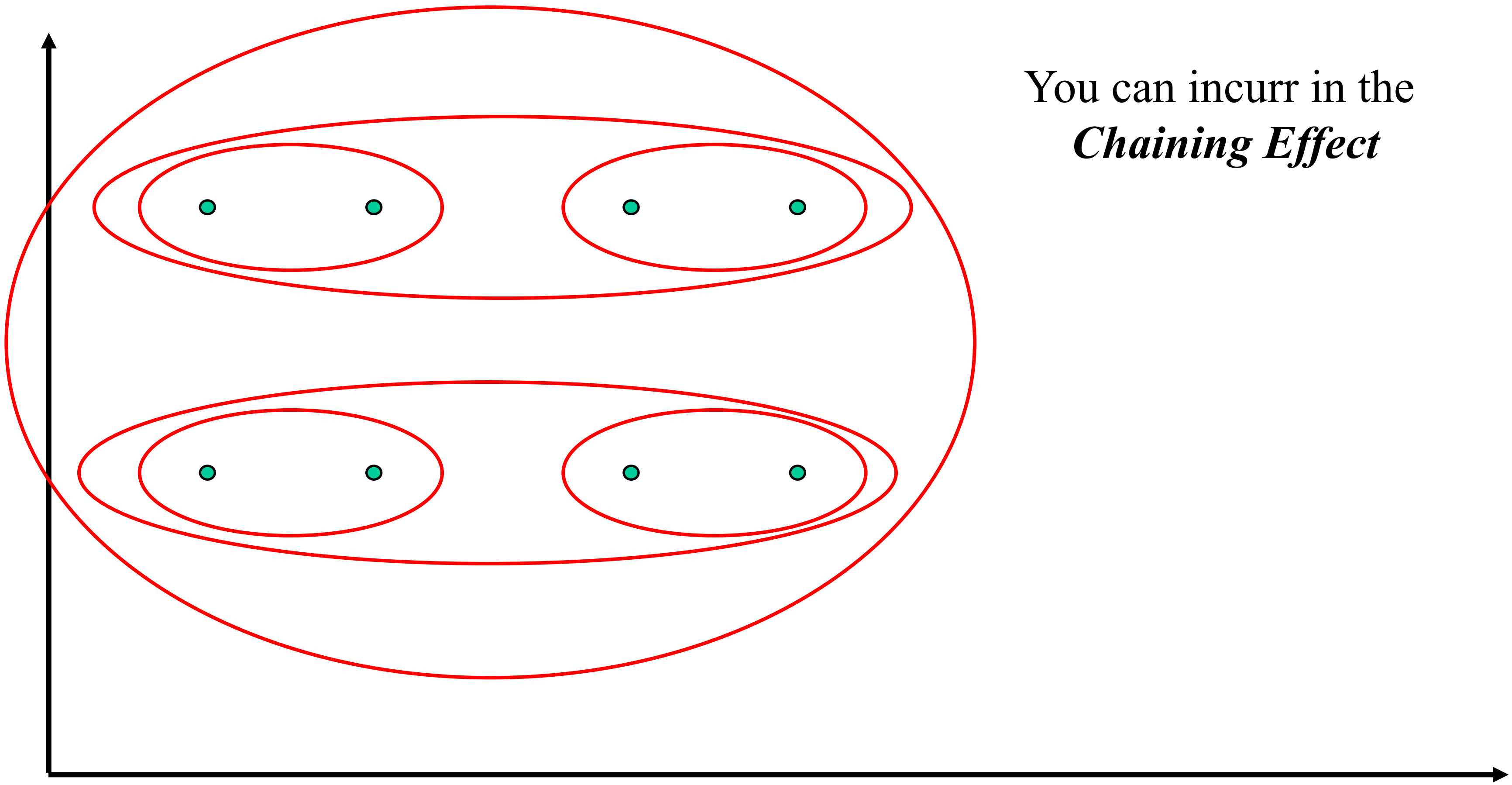
Single Link Agglomerative Clustering

- The distance between two clusters is the distance between two closest data points in the two clusters

- Use maximum similarity of pairs:

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

Single Link Example



You can incurr in the
Chaining Effect

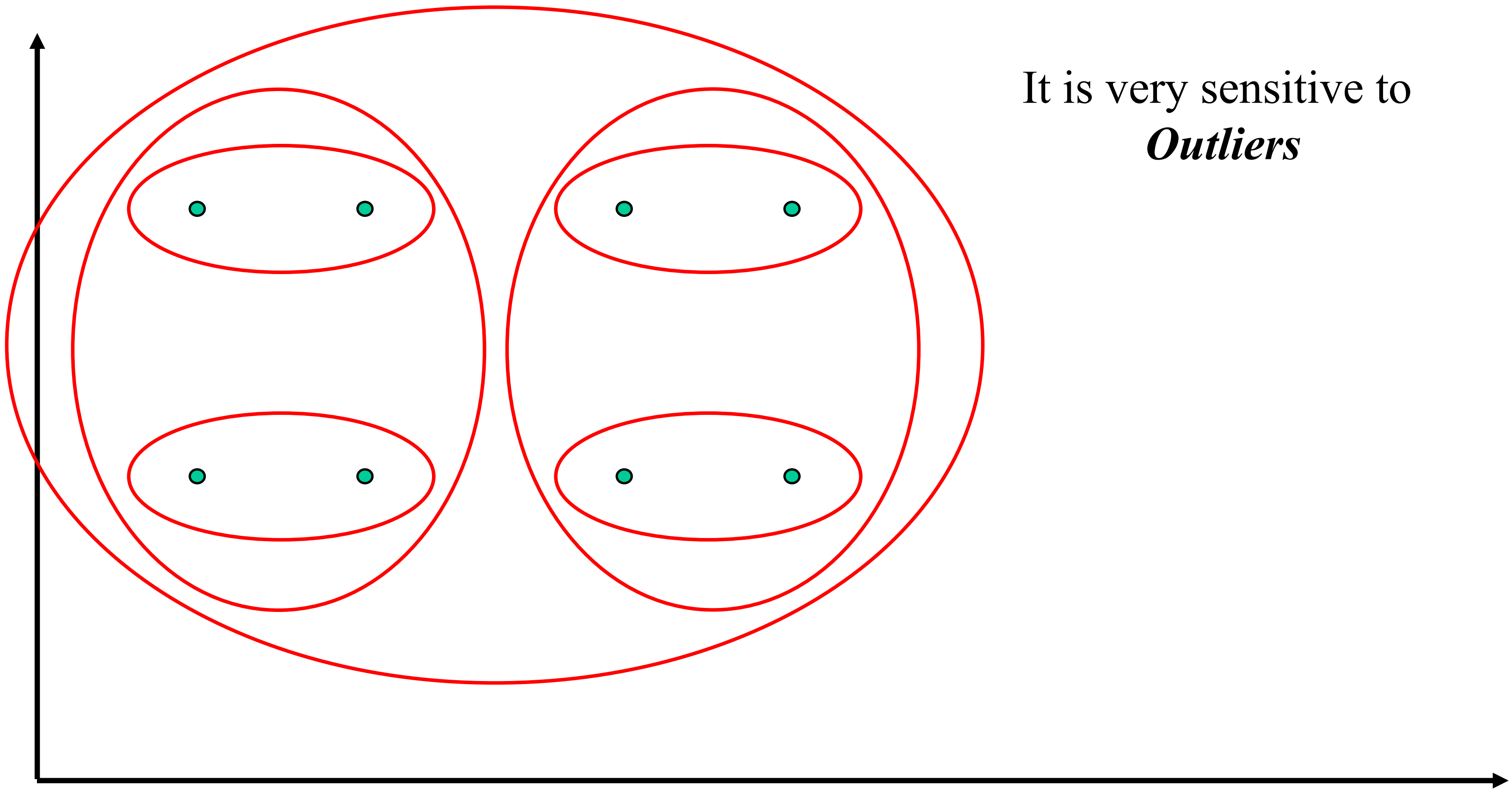
Complete Link Agglomerative Clustering

- The distance between two clusters is the distance of two furthest data points in the two clusters

- Use minimum similarity of pairs:

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

Complete Link Example



It is very sensitive to
Outliers

Group Average Agglomerative Clustering

- Use average similarity across all pairs within the merged cluster to measure the similarity of two clusters

$$sim(c_i, c_j) = \frac{1}{|c_i \cup c_j|(|c_i \cup c_j| - 1)} \sum_{\vec{x} \in (c_i \cup c_j)} \sum_{\vec{y} \in (c_i \cup c_j) : \vec{y} \neq \vec{x}} sim(\vec{x}, \vec{y})$$

- Compromise between single and complete link
- Averaged across all ordered pairs in the merged cluster instead of unordered pairs *between* the two clusters

Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances which is $O(n^2)$

- In each of the subsequent n–2 merging iterations, it must compute the distance between all existing clusters

- In order to maintain an overall $O(n^2)$ performance, computing similarity to each other cluster must be done in constant time.

Computing Group Average Similarity

- Assume cosine similarity and normalized vectors with unit length
- Always maintain sum of vectors in each cluster

$$\vec{s}(c_j) = \sum_{\vec{x} \in c_j} \vec{x}$$

- Compute similarity of clusters in constant time:

$$sim(c_i, c_j) = \frac{(\vec{s}(c_i) + \vec{s}(c_j)) \bullet (\vec{s}(c_i) + \vec{s}(c_j)) - (|c_i| + |c_i|)}{(|c_i| + |c_i|)(|c_i| + |c_i| - 1)}$$

Non-Hierarchical Clustering

- Typically must provide the number of desired clusters, *k*
- Randomly choose *k* instances as *seeds*, one per cluster
- Form initial clusters based on these seeds
- Iterate, repeatedly reallocating instances to different clusters to improve the overall clustering
- Stop when clustering converges or after a fixed number of iterations

K-Means

- Assumes instances are real-valued vectors
- Clusters based on *centroids*, *center of gravity*, or mean of points in a cluster, c:

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

- Reassignment of instances to clusters is based on distance to the current cluster centroids

K-Means Algorithm

Let d be the distance measure between instances.

Select k random instances $\{s_1, s_2, \ldots s_k\}$ as seeds.

Until clustering converges or other stopping criterion:

For each instance x_i:

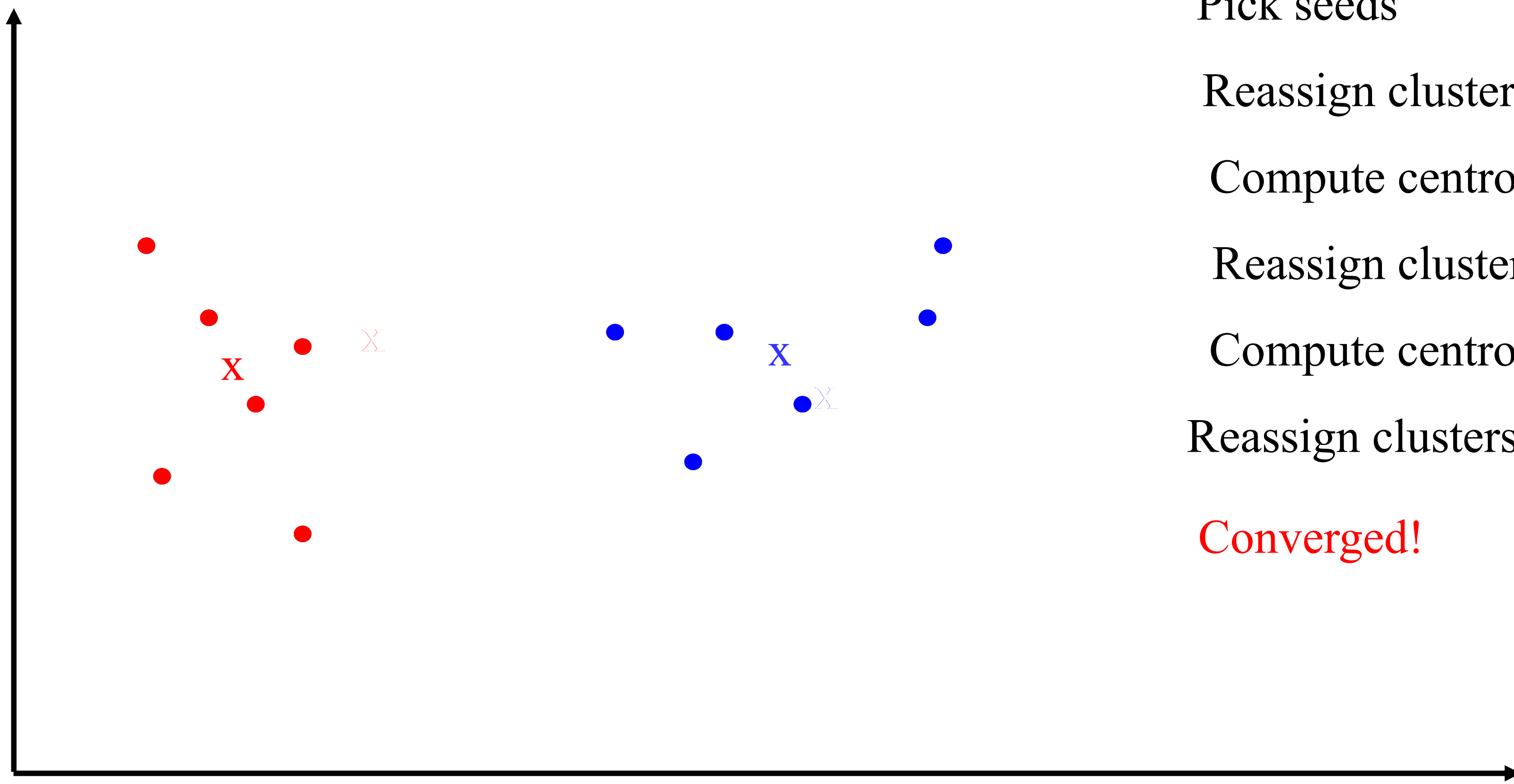
Assign x_i to the cluster c_j such that $d(x_i, s_j)$ is minimal.

(Update the seeds to the centroid of each cluster)

For each cluster c_j

$s_j = \mu(c_j)$

K Means Example
(K=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

K-Means stopping criteria

- No or minimum reassignment of data in clusters
- No or minimum change in centroids
- Minimum decrease in the sum of squared error (SSE)

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2$$

C_i is the j-th cluster, $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

K-Mean - Time Complexity

- Assume computing distance between two instances is O(m) where m is the dimensionality of the vectors
- Reassigning clusters: O(kn) distance computations, or O(knm)
- Computing centroids: Each instance vector gets added once to some centroid: O(nm)
- Assume these two steps are each done once for I iterations: O($Iknm$)
- Linear in all relevant factors, assuming a fixed number of iterations, more efficient than O(n^2) HAC

Distance Metrics

- Euclidian distance (L_2 norm):

$$L_2(\vec{x}, \vec{y}) = \sum_{i=1}^{m} (x_i - y_i)^2$$

- L_1 norm:

$$L_1(\vec{x}, \vec{y}) = \sum_{i=1}^{m} \left| x_i - y_i \right|$$

- Cosine Similarity (transform to a distance by subtracting from 1):

$$1 - \frac{\vec{x} \bullet \vec{y}}{|\vec{x}| \cdot |\vec{y}|}$$

Distance Metrics

- Chebychev distance
  - define two data points as "different" if they are different on any one of the attributes

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, \ldots, |x_{ir} - x_{jr}|)$$

Data standardization

- In the Euclidean space, standardization of attributes is recommended so that all attributes can have equal impact on the computation of distances
- Consider the following pair of data points
  - $\mathbf{x}_i$: (0.1, 20) and $\mathbf{x}_j$: (0.9, 720).

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(0.9-0.1)^2 + (720-20)^2} = 700.000457,$$

- The distance is almost completely dominated by (720-20) = 700
- Standardize attributes: to force the attributes to have a common value range

Interval-scaled attributes

- Their values are real numbers following a linear scale
  - The difference in Age between 10 and 20 is the same as that between 40 and 50
  - The key idea is that intervals keep the same importance through out the scale
- Two main approaches to standardize interval scaled attributes, **range** and **z-score**. *f* is an attribute

$$range(x_{if}) = \frac{x_{if} - \min(f)}{\max(f) - \min(f)},$$

Interval-scaled attributes (cont …)

- Z-score: transforms the attribute values so that they have a mean of zero and a **mean absolute deviation** of 1. The mean absolute deviation of attribute *f*, denoted by s_f, is computed as follows

$$s_f = \frac{1}{n}\left(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|\right),$$

$$m_f = \frac{1}{n}\left(x_{1f} + x_{2f} + \ldots + x_{nf}\right),$$

$$z(x_{if}) = \frac{x_{if} - m_f}{s_f}.$$

Clustering evaluation

- How can we evaluate produced clusters?
- We can use some *internal criteria* for the quality of a clustering
  - Typical objective functions can formalize the goal of attaining high intra-cluster similarity and low inter-cluster similarity
  - But good scores on an internal criterion do not necessarily translate into good effectiveness in an application
- It is better to adopt some *external criteria*
  - we can use a set of classes in an evaluation benchmark or gold standard
- Or we can use some *indirect evaluation criteria*
  - In some applications, clustering is not the primary task, but used to help perform another task.
  - We can use the performance on the primary task to compare clustering methods.

Cluster evaluation: External Criteria

- We use some labeled data (for classification)
- Assumption: Each class is a cluster
- After clustering, build a confusion matrix
- From the matrix, compute various measurements: Entropy, Purity, Precision, Recall and F-score
  - Let the classes in the data D be $C = (c_1, c_2, \ldots, c_k)$. The clustering method produces k clusters, which divides D into k disjoint subsets, $D_1, D_2, \ldots, D_k$.
  - We can estimate $Pr_i(c_j)$, i.e. the proportion of class c_j data points in cluster i or D_i

$$Pr_i(c_j) = |\, c_j \cap D_i \,| \,/\, |D_i|$$

Evaluation measures: purity

Purity: This again measures the extent that a cluster contains only one class of data. The purity of each cluster is computed with

$$purity(D_i) = \max_j(\Pr_i(c_j)) \quad (31)$$

The total purity of the whole clustering (considering all clusters) is

$$purity_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times purity(D_i) \quad (32)$$

If all clusters contain one instance only, the purity will be maximim, i.e. equal to 1

Evaluation measures: Entropy

Entropy: For each cluster, we can measure its entropy as follows:

$$entropy(D_i) = -\sum_{j=1}^{k} \Pr_i(c_j)\log_2 \Pr_i(c_j), \quad (29)$$

where $\Pr_i(c_j)$ is the proportion of class c_j data points in cluster i or D_i. The total entropy of the whole clustering (which considers all clusters) is

$$entropy_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times entropy(D_i) \quad (30)$$

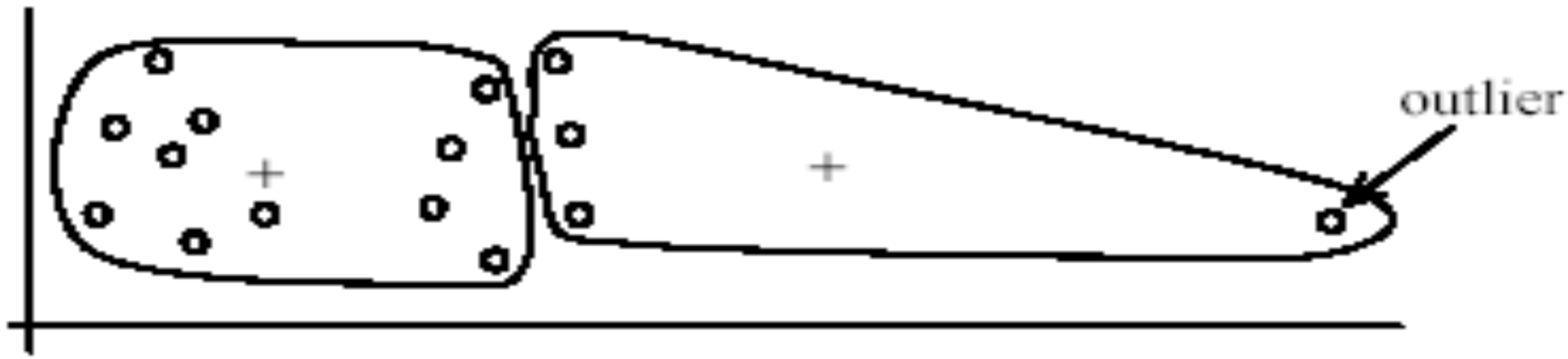
Soft Clustering

- Clustering typically assumes that each instance is (hard) assigned to exactly one cluster
  - Does not allow uncertainty in class membership or for an instance to belong to more than one cluster
- *Soft clustering* gives probabilities to instances of belonging to each clusters
  - ES: Fuzzy C-mean
- Each instance has a probability distribution across a set of discovered categories (probabilities of all categories must sum to 1)

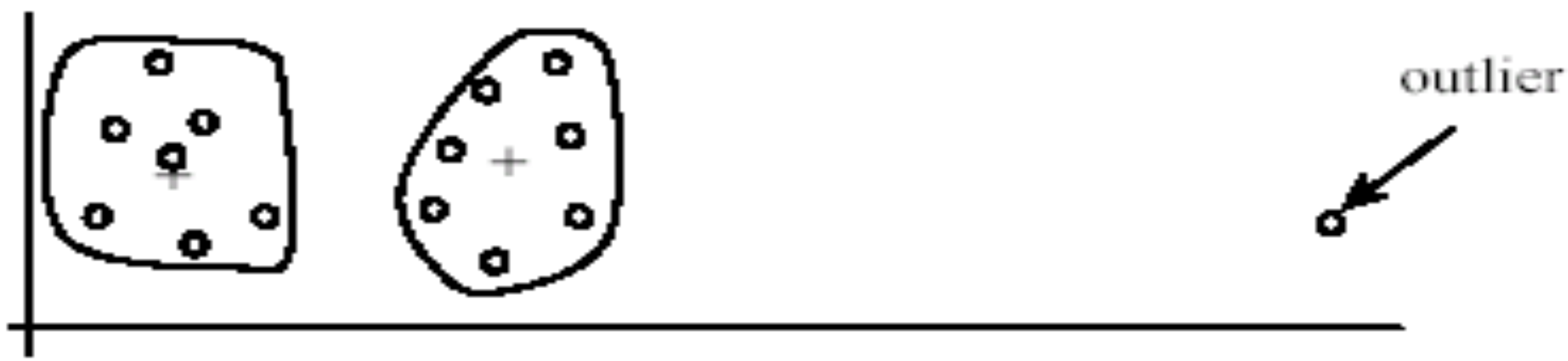
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined
  - For categorical data, *k*-mode - the centroid is represented by most frequent values
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points
  - Outliers could be errors in the data recording or some special data points with very different values
- The user needs to specify *k*
- Results can very based on random seed selection
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings
- Select good seeds using a heuristic or the results of another method

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



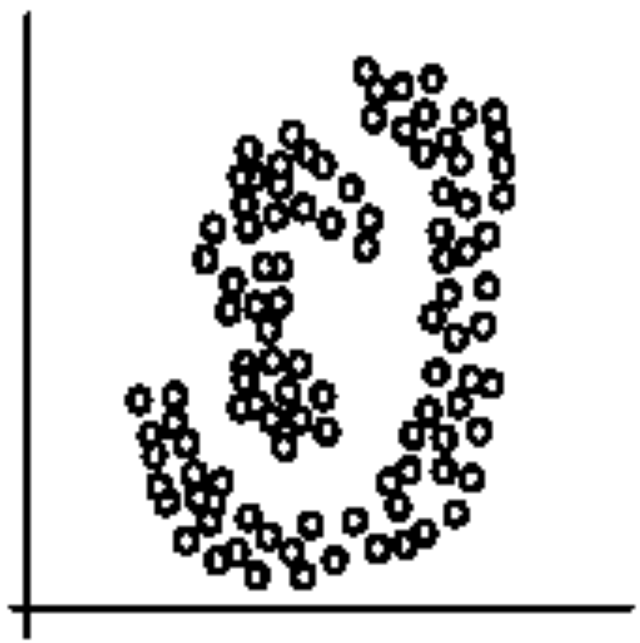
(B): Ideal clusters

Dealing with outliers

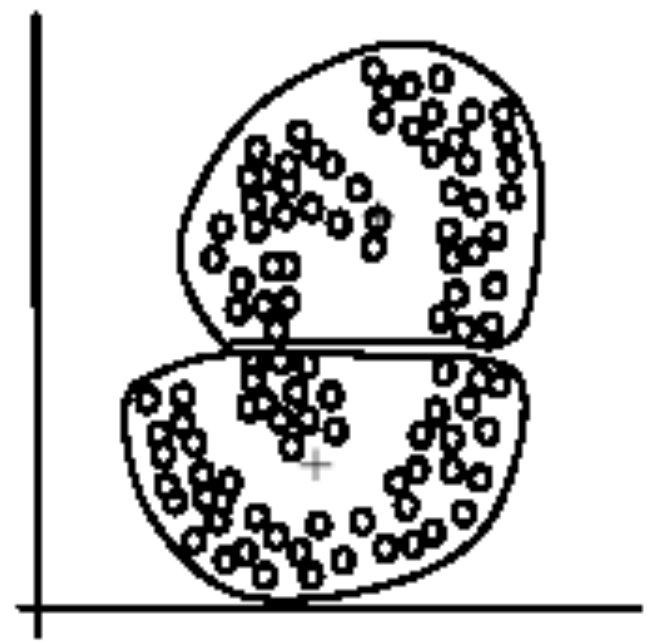
- How to deal with outliers?
- One method is to remove some data points that are far from centroids
  - However, they can be important data
  - To be safe, monitor these points over multiple iterations before removing them
- Perform random sampling
  - Choose randomly points to partition
  - Choice of outliers is unlikely

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres)



(A): Two natural clusters



(B): *k*-means clusters

Advanced Techniques QT K-Means Algorithm (1)

- *Quality Threshold (QT) K-Means* Algorithm is an evolution of basic *K-Means* that dynamically change the number of cluster *k*
- Use two threshold to consider both *inter-cluster* and *infra-cluster* similarity

Advanced Techniques
QT K-Means Algorithm (2)

Let σ and τ be two different thresholds.
Let d be the distance measure between instances.
Select k random instances $\{s_1, s_2, \ldots s_k\}$ as seeds.
Until clustering converges or other stopping criterion:

For each instance x_i:

Assign x_i to the cluster c_j such that $d(x_i, s_j)$ is minimal but less then σ.

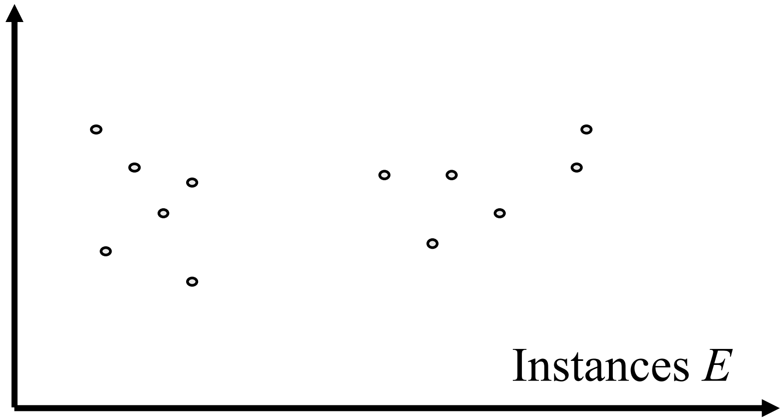
Else create new seed with instance x_i (the number k of clusters increase)

(Update the seeds to the centroid of each cluster)

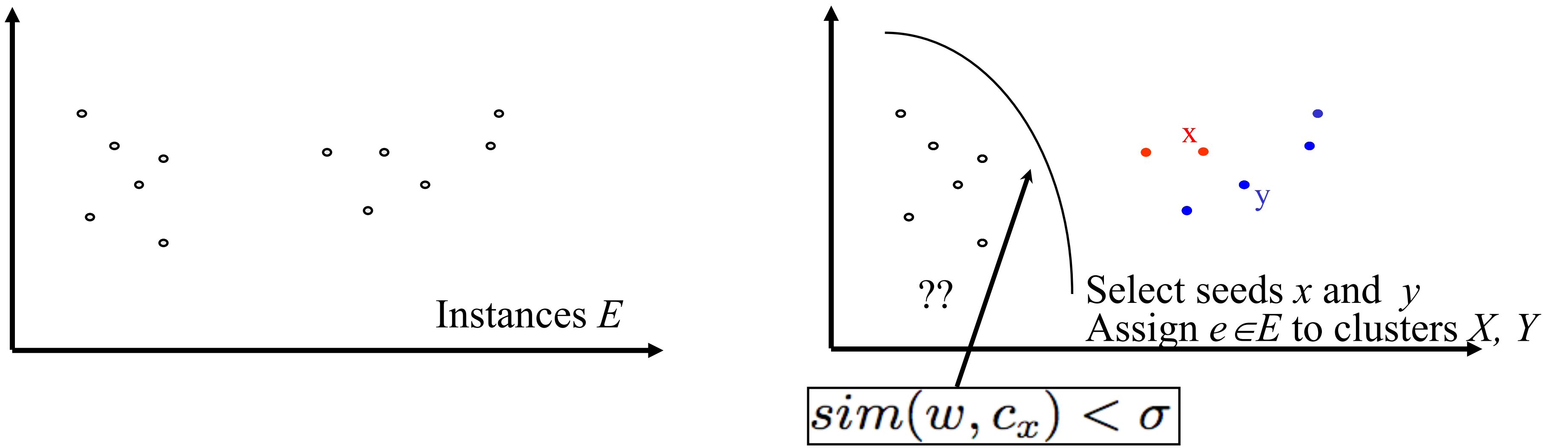
For each cluster pairs c_i, c_j tc $i \neq j$:

If $d(s_i, s_j)$ less than τ merge c_i and c_j (the number k of clusters decrease)

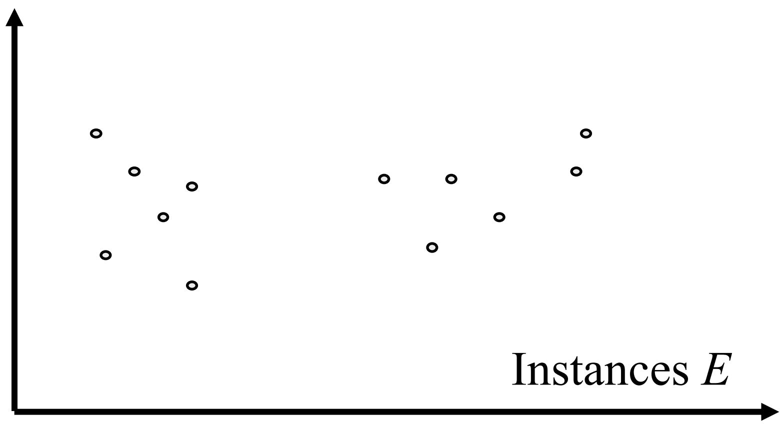
Advanced Techniques QT K-Means Algorithm (3)

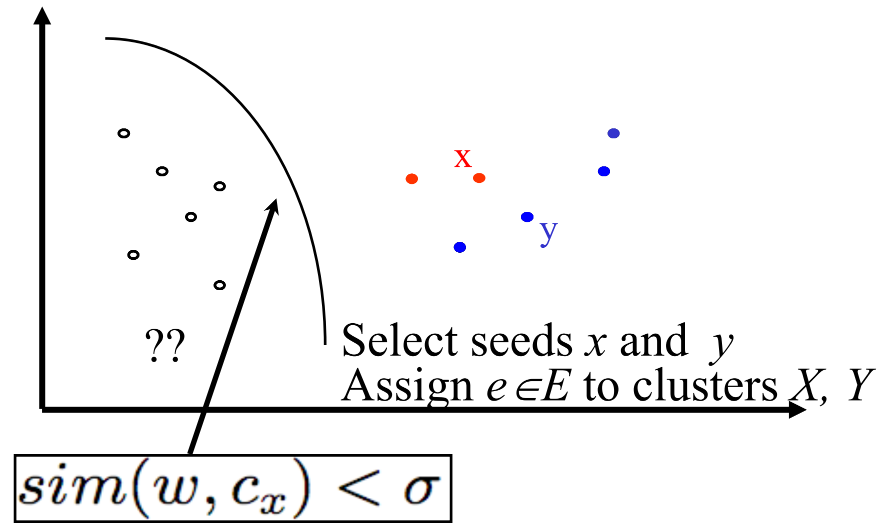


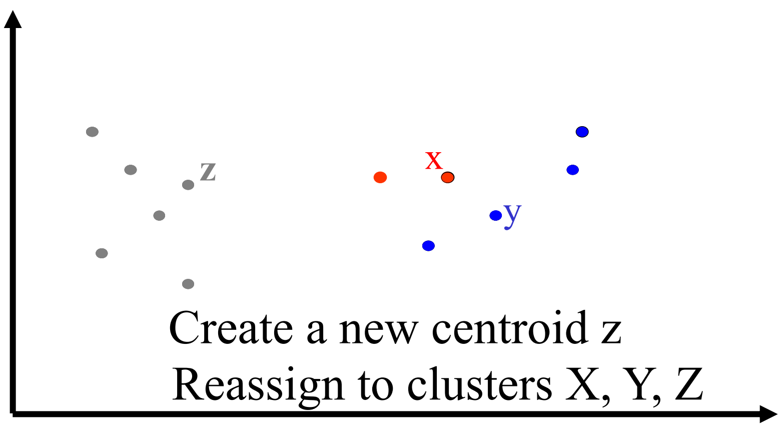
Advanced Techniques
QT K-Means Algorithm (3)



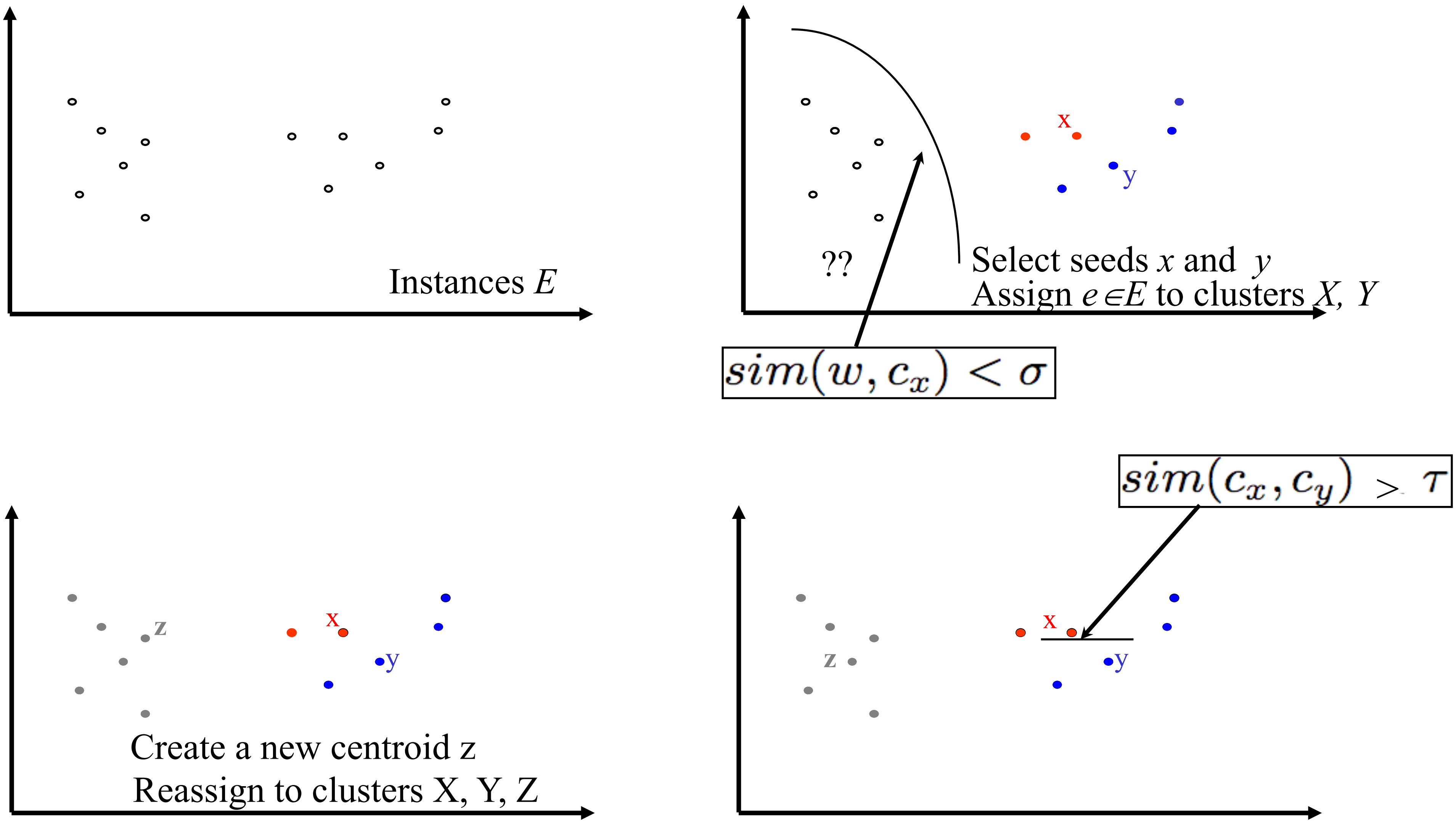
Advanced Techniques
QT K-Means Algorithm (3)



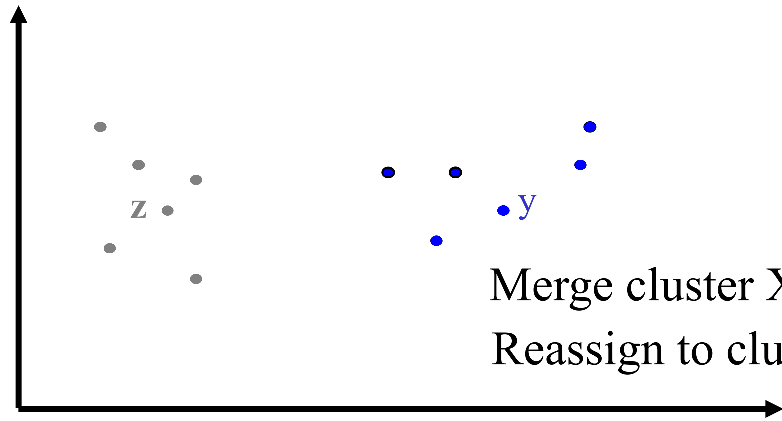




Advanced Techniques
QT K-Means Algorithm (3)



Advanced Techniques
QT K-Means Algorithm (3)



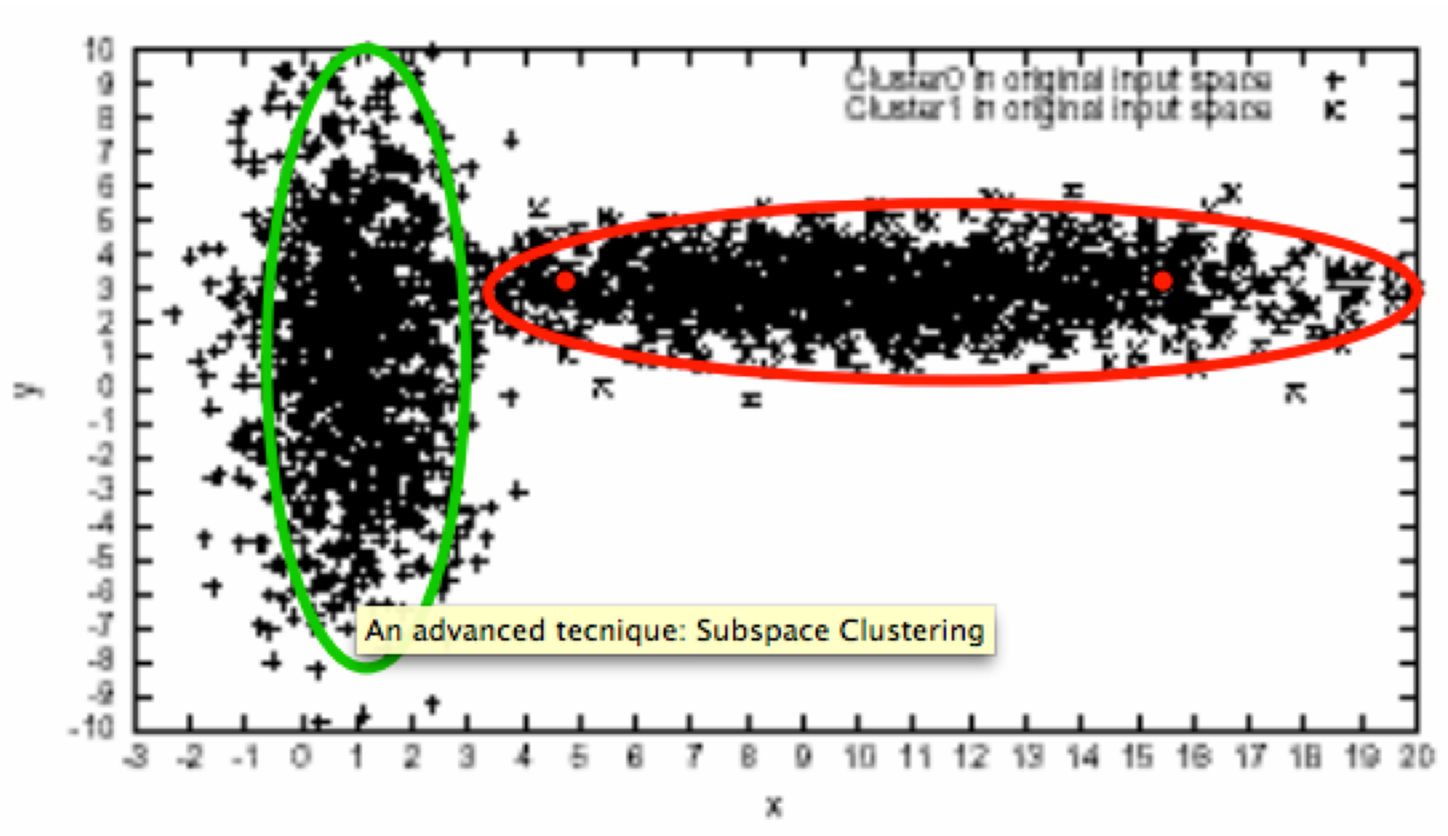
Merge cluster X and Y

Reassign to clusters Y, Z

Convergence

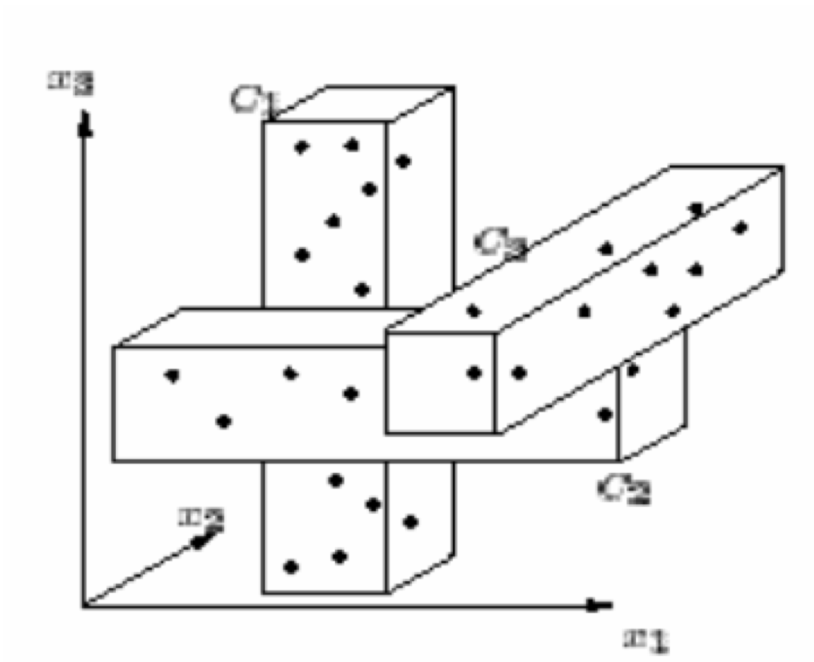
Advanced Techniques
Subspace Clustering (1)

- In high dimensional spaces, few dimensions can exist on which the points are far apart from each other



An advanced tecnique
Subspace Clustering (2)

- Subspace Clustering: seek to find clusters in a dataset by selecting the most relevant dimensions for each cluster separately



Each dimension is relevant to at least one cluster

A Subspace Clustering algorithm
Locally Adaptive Clustering

- We cannot prune off dimensions without incurring a loss of crucial information
- The data presents local structure:
  - To capture the local correlations of data a proper feature selection procedure should operate locally
  - A local operation would allow to embed different distance measures in different regions
- **IDEA:** apply a *Co-clustering* approach
  - simultaneous clustering of *both* data and dimensions

Locally adaptive metrics for clustering high dimensional data
Domeniconi et al, 2007

A Subspace Clustering algorithm
Locally **A**daptive **C**lustering

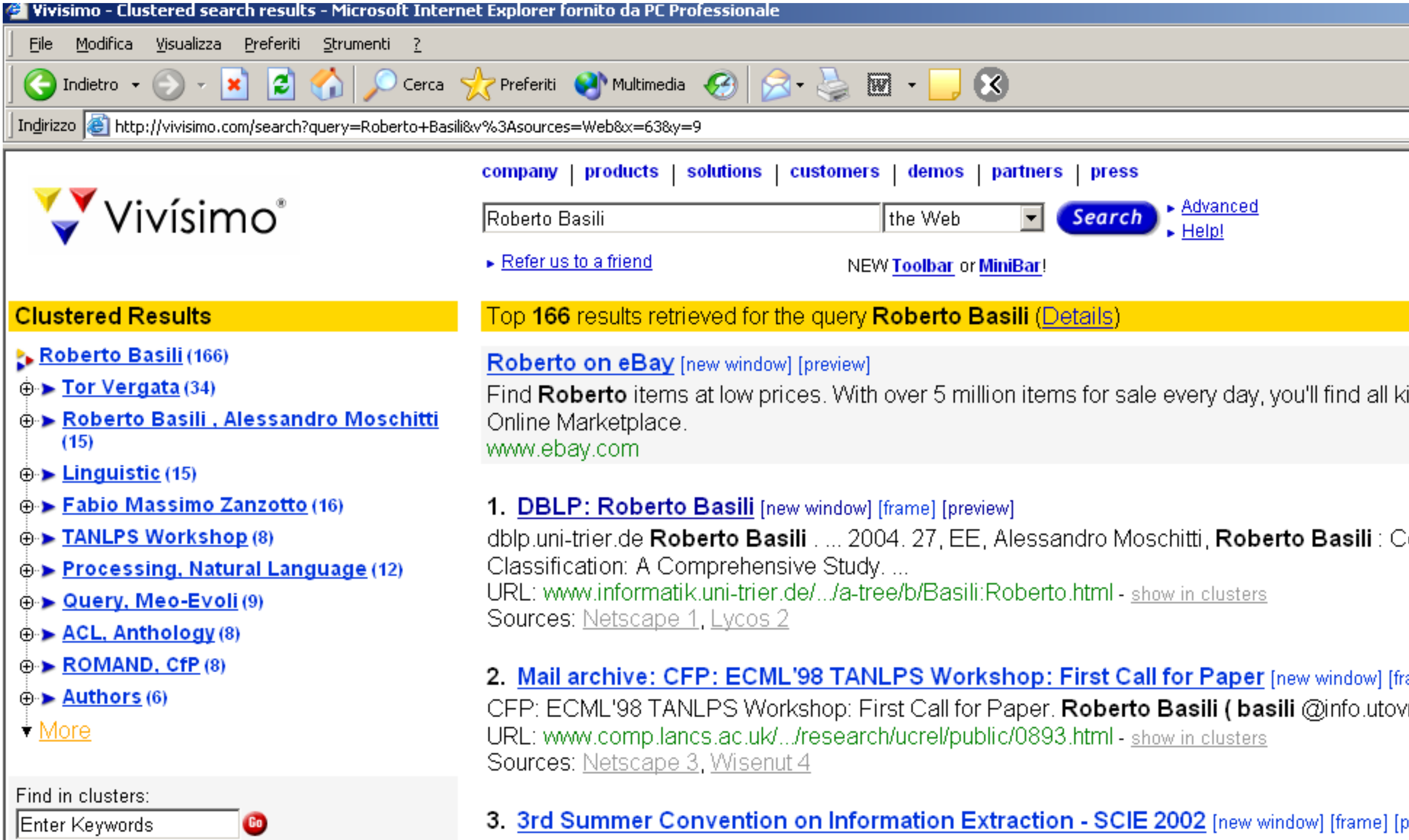
- LAC is a variant of K-Means where cluster are weighted
  - Each centroid is weighted so that only few dimensions are considered when associating data point to clusters
  - At each step the centroid weighting schema is update
  - In each cluster the weights determine the informative dimensions

Locally adaptive metrics for clustering high dimensional data
Domeniconi et al, 2007

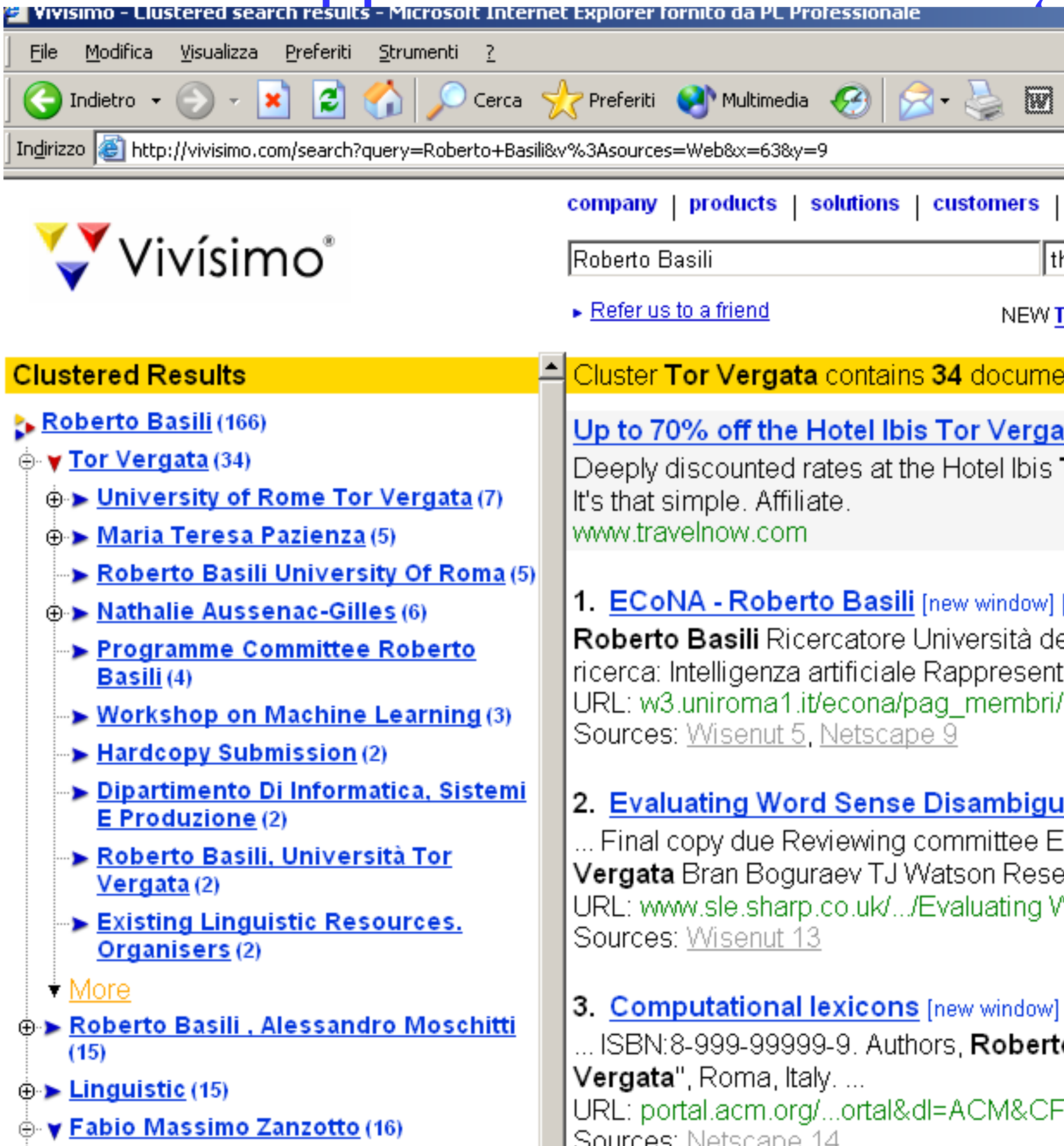
Some applications: Text Clustering

- HAC and K-Means have been applied to text in a straightforward way
- Typically use ***normalized***, TF/IDF-weighted vectors and cosine similarity
- Optimize computations for sparse vectors
- Applications:
  - During retrieval, add other documents in the same cluster as the initial retrieved documents to improve recall
  - Clustering of results of retrieval to present more organized results to the user (à la Northernlight folders)
  - Automated production of hierarchical taxonomies of documents for browsing purposes (à la Yahoo & DMOZ)

Some applications: Clustering and search (1)



Some applications: Clustering and search (2)



Current Challenges in Clustering

Many traditional clustering techniques do not perform satisfactorily in data mining scenarios due to a variety of reasons

- **Data Distribution**
  - **Large number of samples**
    - The number of samples to be processed is very high. Clustering in general is NP-hard, and practical and successful data mining algorithms usually scale linear or log-linear. Quadratic and cubic scaling may also be allowable but a linear behavior is highly desirable.
  - **High dimensionality**
    - The number of features is very high and may even exceed the number of samples. So one has to face the curse of dimensionality
  - **Sparsity**
    - Most features are zero for most samples, i.e. the object-feature matrix is sparse. This property strongly affects the measurements of similarity and the computational complexity.
  - **Significant outliers**
    - Outliers may have significant importance. Finding these outliers is highly non-trivial, and removing them is not necessarily desirable.

Current Challenges in Clustering (cont.)

- **Application context**
  - **Legacy clusterings**
    - Previous cluster analysis results are often available. This knowledge should be reused instead of starting each analysis from scratch.
  - **Distributed data**
    - Large systems often have heterogeneous distributed data sources. Local cluster analysis results have to be integrated into global models.