



Cornell University

Clustering

CS4780/5780 – Machine Learning
Fall 2011

Thorsten Joachims
Cornell University

Reading: Manning/Raghavan/Schuetze,
Chapters 16 (not 16.3) and 17
(<http://nlp.stanford.edu/IR-book/>)



- Supervised vs. Unsupervised Learning
- Hierarchical Clustering
 - Hierarchical Agglomerative Clustering (HAC)
- Non-Hierarchical Clustering
 - K-means
 - EM-Algorithm



- Supervised Learning
 - *Classification*: partition examples into groups according to pre-defined categories
 - *Regression*: assign value to feature vectors
 - Requires labeled data for training
- Unsupervised Learning
 - *Clustering*: partition examples into groups when no pre-defined categories/classes are available
 - *Novelty detection*: find changes in data
 - *Outlier detection*: find unusual events (e.g. hackers)
 - Only instances required, but no labels



- Partition unlabeled examples into disjoint subsets of *clusters*, such that:
 - Examples within a cluster are similar
 - Examples in different clusters are different
- Discover new categories in an *unsupervised* manner (no sample category labels provided).



- Cluster retrieved documents (e.g. Teoma)
 - to present more organized and understandable results to user
- Detecting near duplicates
 - Entity resolution
 - E.g. “Thorsten Joachims” == “Thorsten B Joachims”
 - Cheating detection
- Exploratory data analysis
- Automated (or semi-automated) creation of taxonomies
 - e.g. Yahoo-style
- Compression



The screenshot shows a Microsoft Internet Explorer browser window displaying the Teoma search engine results for the query "svm". The address bar shows the URL: <http://s.teoma.com/search?q=svm&qcat=1&qsrc=0&Search.x=46&Search.y=4>. The search bar contains the text "svm".

Results
Relevant web pages

Showing 1-10 of about 139,300:

- [BROADWAY DISCOUNTS](#)
Brooklyn ... is a real treasure."-WOR Radio Feel it for yourself at <http://www.brooklynthemusical.com/svm!>
www.jeremysalehouse.com/svm_guide.htm | [Cached](#)
- [Kernel Machines](#)
Shai Fine and Katya Scheinberg: Efficient SVM Training Using Low-Rank ... Probabilistic treatment of SVMs ... SVM methods for large scale...
www.kernel-machines.org/ | [Cached](#)
[\[Related Pages\]](#)
- [SVM Application List](#)
SVM Application List...
www.clopinet.com/isabelle/Projects/SVM/app... | [Cached](#)
[\[Related Pages\]](#)
- [T Speech and Image Processing Services Lab](#)
SVM overview, and software.
svm.dcs.rhbnc.ac.uk/ | [Cached](#)
- [Support Vector Machines - The Book - Support Vector](#)
Complete, simple and rigorous introduction to Support Vector Machines, learning algorithm widely used in data mining, machine vision,
www.support-vector.net/ | [Cached](#)
- [SVM-Light Support Vector Machine](#)
SVM-Light Support Vector Machine

Refine
Suggestions to narrow your search

- [Support Vector Machines](#)
- [Veterinary Medicine](#)
- [Sony Svm](#)
- [Uc Davis](#)
- [Photo Paper](#)
- [SVM Cars](#)

[\[Show All Refinements\]](#)

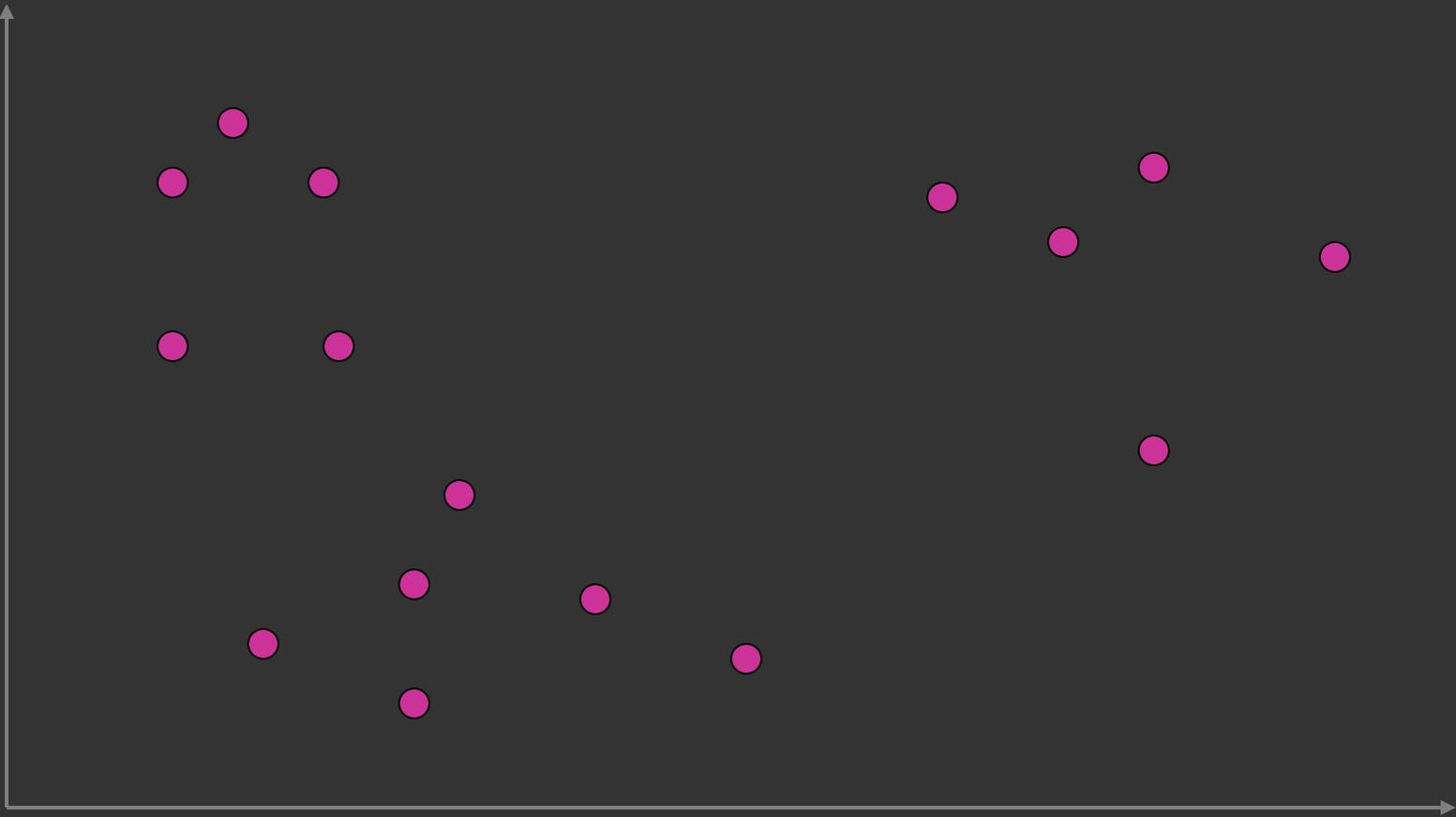
Resources
Link collections from experts and enthusiasts

We found no link collections for your search **svm**



Cornell University

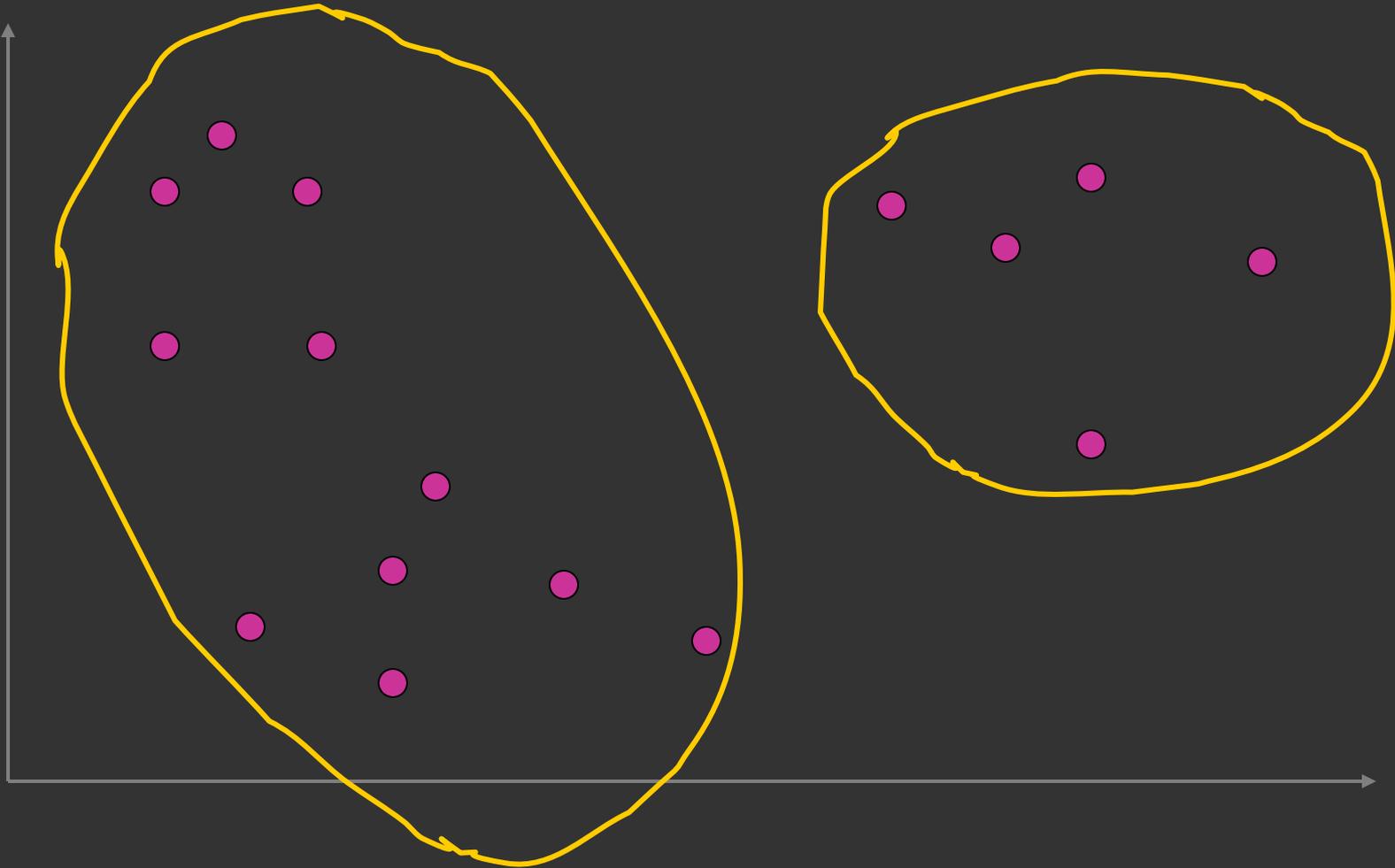
Clustering Example





Cornell University

Clustering Example





Cornell University

Clustering Example





Cornell University

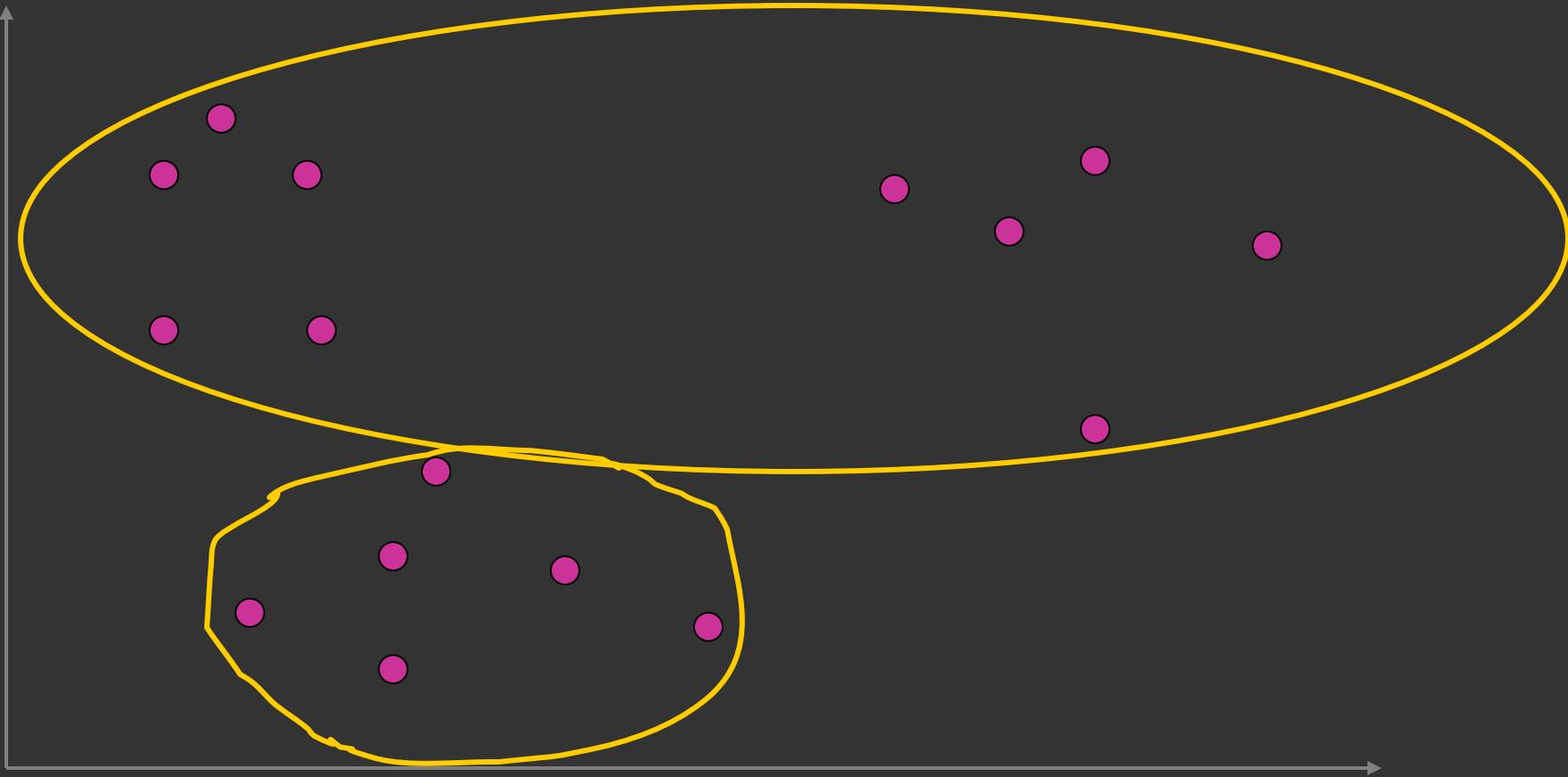
Clustering Example





Cornell University

Clustering Example





- Euclidian distance (L_2 norm):

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

- L_1 norm:

$$L_1(\vec{x}, \vec{x}') = \sum_{i=1}^m |x_i - x_i'|$$

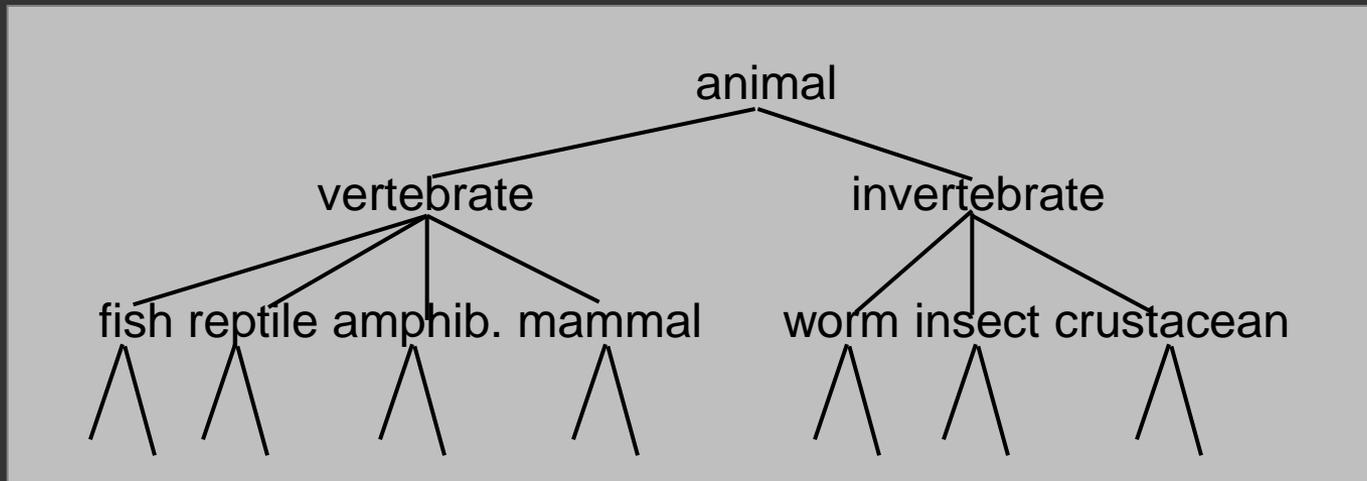
- Cosine similarity:

$$\cos(\vec{x}, \vec{x}') = \frac{\vec{x} \cdot \vec{x}'}{|\vec{x}| \cdot |\vec{x}'|}$$

- Kernels



- Build a tree-based hierarchical taxonomy from a set of unlabeled examples.

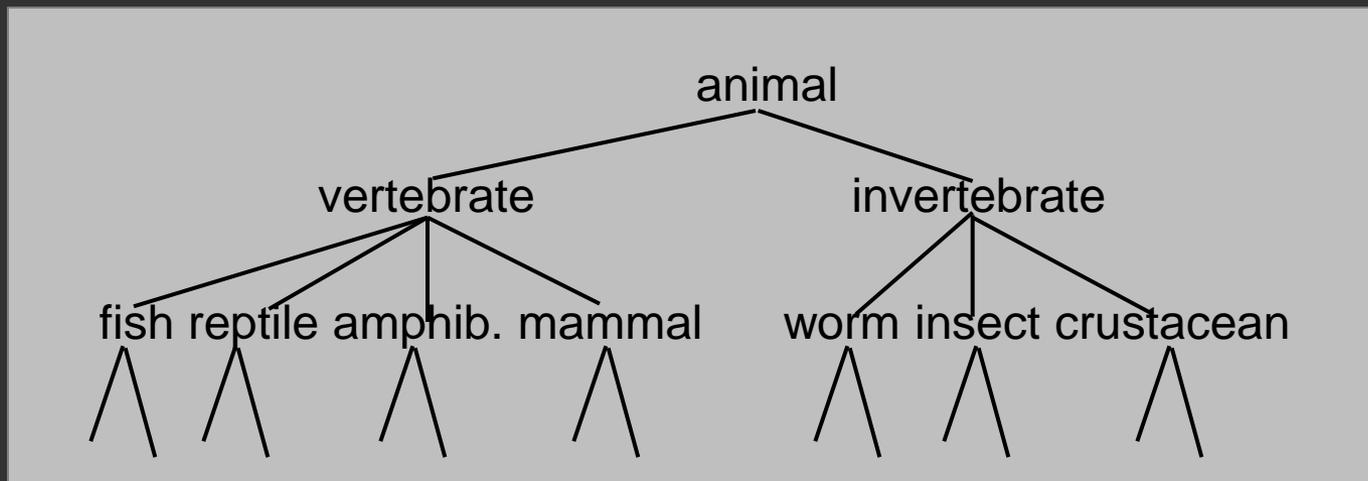


- Recursive application of a standard clustering algorithm can produce a hierarchical clustering.



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** (*top-down*) separate all examples immediately into clusters.



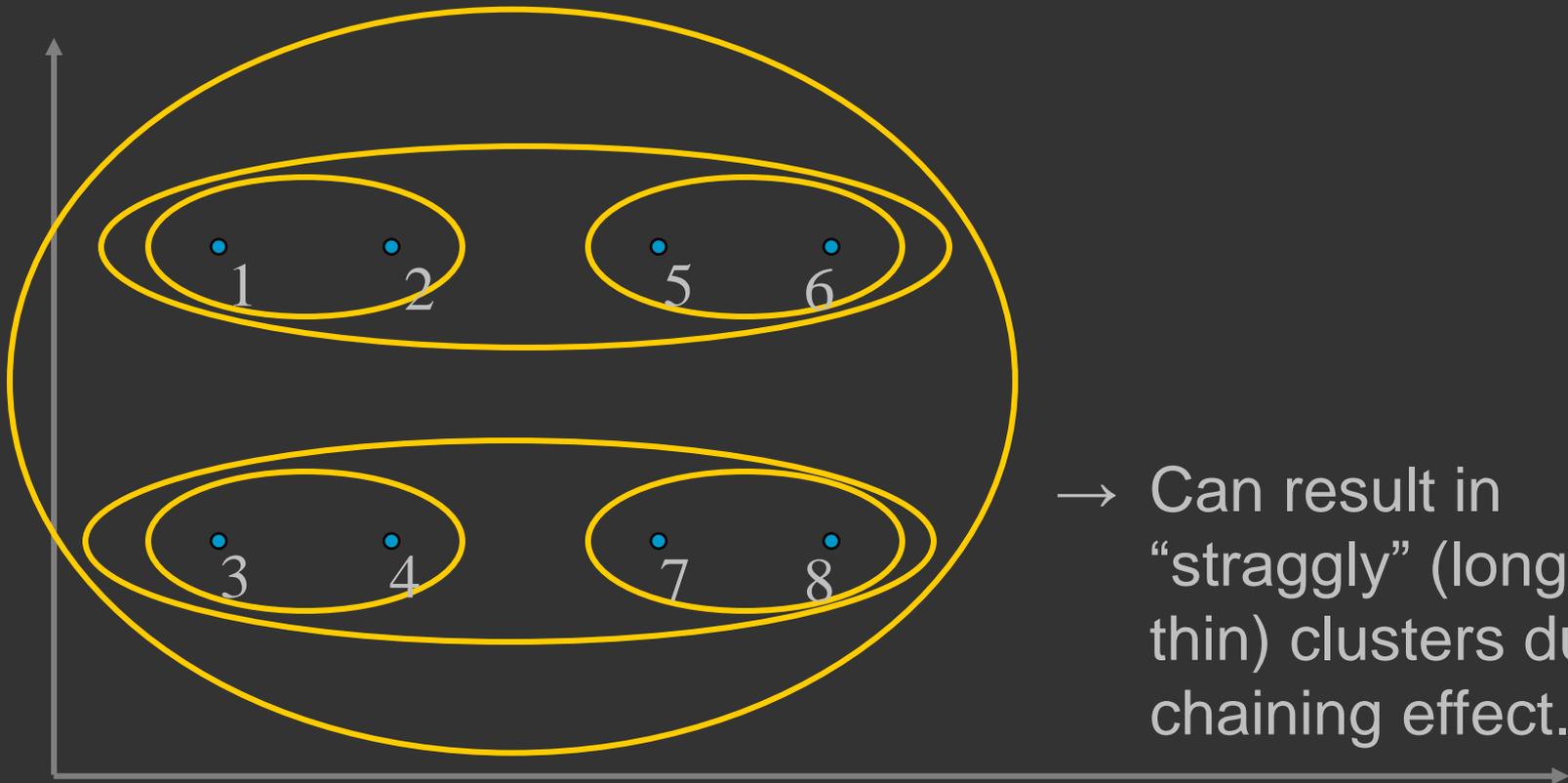


Hierarchical Agglomerative Clustering (HAC)

- Assumes a *similarity function* for determining the similarity of two clusters.
- 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.
- Basic 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 $c_i \cup c_j$



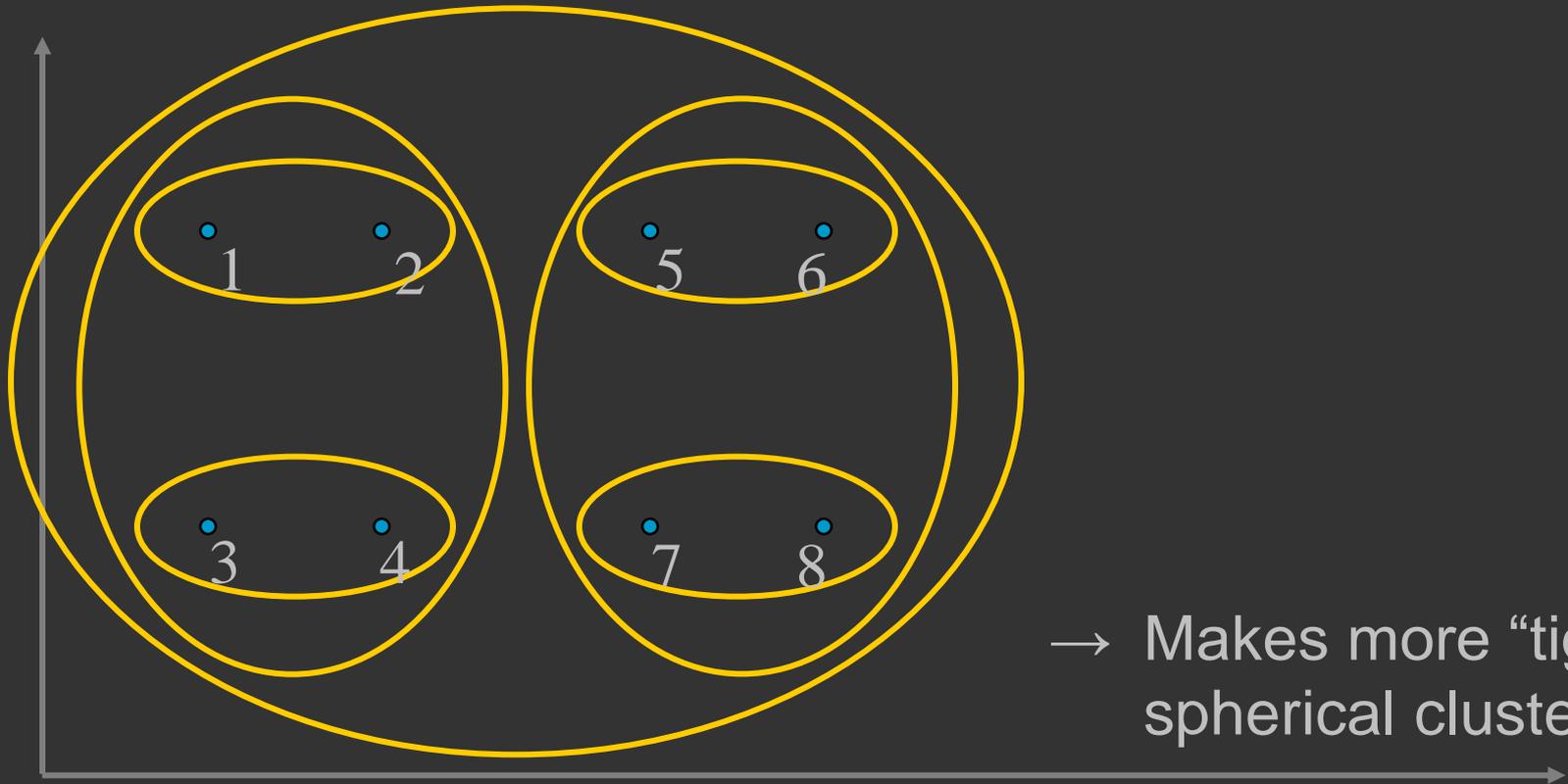
- How to compute similarity of two clusters each possibly containing multiple instances?
 - ***Single link***: Similarity of two most similar members.
 - ***Complete link***: Similarity of two least similar members.
 - ***Group average***: Average similarity between members.



→ Can result in “straggly” (long and thin) clusters due to chaining effect.

- When computing cluster similarity, use maximum similarity of pairs:

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



→ Makes more “tight,” spherical clusters.

- When computing cluster similarity, use maximum similarity of pairs:

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



- 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 $O(n)$ merging iterations, it must compute the distance between the most recently created cluster and all other existing clusters.
- In order to maintain the similarity matrix in $O(n^2)$ overall, computing the similarity to any other cluster must each be done in constant time.
- Maintain Heap to find smallest pair $\rightarrow O(n^2 \log n)$



- After merging c_i and c_j , the similarity of the resulting cluster to any other cluster, c_k , can be computed by:
 - Single Link:

$$\mathit{sim}((c_i \cup c_j), c_k) = \max(\mathit{sim}(c_i, c_k), \mathit{sim}(c_j, c_k))$$

- Complete Link:

$$\mathit{sim}((c_i \cup c_j), c_k) = \min(\mathit{sim}(c_i, c_k), \mathit{sim}(c_j, c_k))$$



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.



- 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:

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



Cornell University

Non-Hierarchical Clustering

- Single-pass clustering
- K-means clustering (“hard”)
- Expectation maximization (“soft”)



- Evaluation function that assigns a (usually real-valued) value to a clustering
 - Clustering criterion typically function of
 - within-cluster similarity and
 - between-cluster dissimilarity
- Optimization
 - Find clustering that maximizes the criterion
 - Global optimization (often intractable)
 - Greedy search
 - Approximation algorithms



- Assumes instances are real-valued vectors.
- Clusters represented via *centroids* (i.e. average of points in a cluster) c :

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

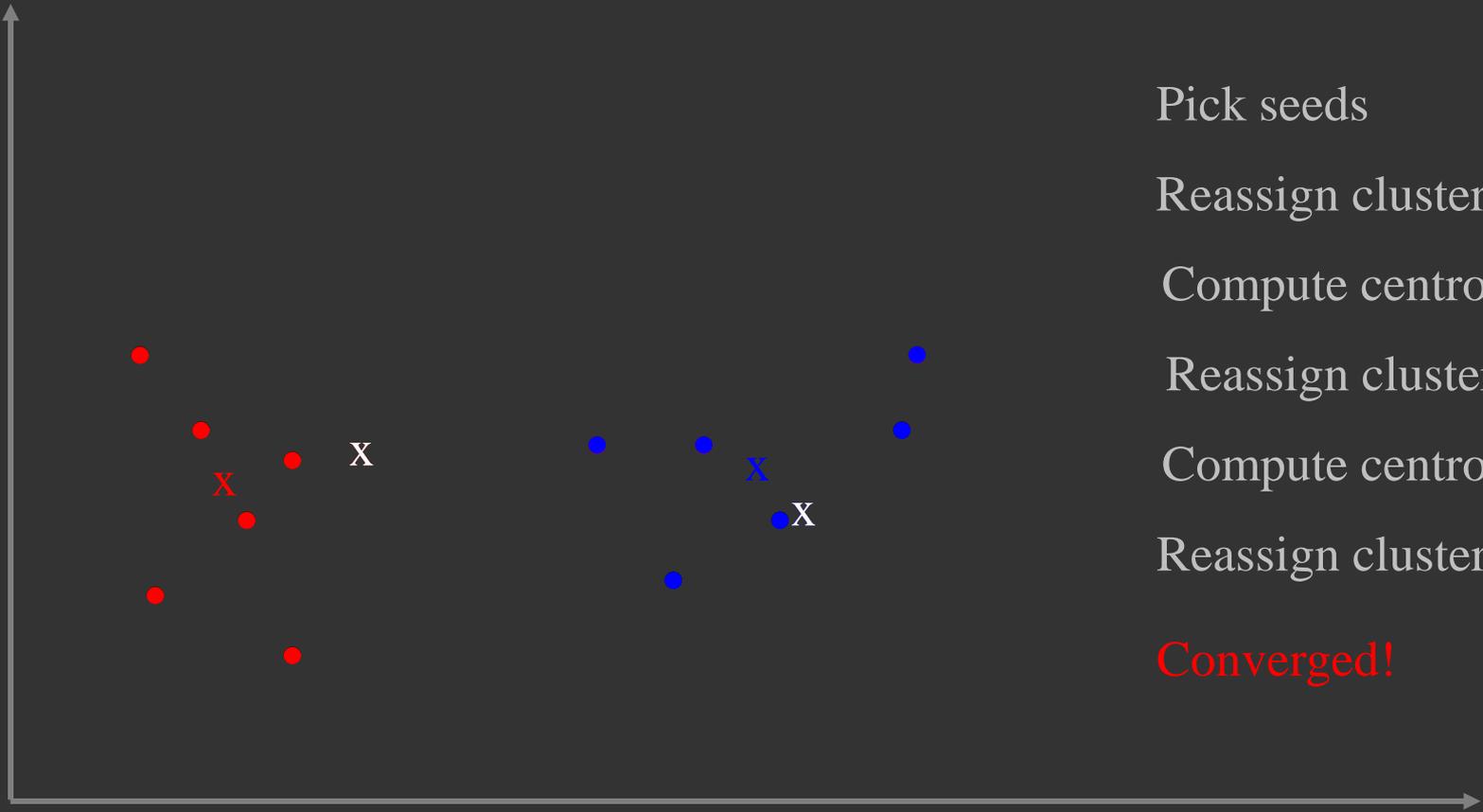
- Reassignment of instances to clusters is based on **distance** to the current cluster centroids.



- Input: k = number of clusters, distance measure d
- Select k random instances $\{s_1, s_2, \dots, s_k\}$ 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 min.
 - For each cluster c_j //update the centroid of each cluster
 - $s_j = \mu(c_j)$



K-means Example ($k=2$)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!



- Assume computing distance between two instances is $O(m)$ where m is the dimensionality of the vectors.
- Reassigning clusters for n points: $O(kn)$ distance computations, or $O(knm)$.
- Computing centroids: Each instance 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 HAC.



Problem

- Results can vary based on random seed selection, especially for high-dimensional data.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.

Idea: Combine HAC and K-means clustering.

- First randomly take a sample of instances of size $n^{1/2}$
- Run group-average HAC on this sample
- Use the results of HAC as initial seeds for K-means.
- Overall algorithm is efficient and avoids problems of bad seed selection.