Clustering: Similarity-Based Clustering

CS4780/5780 – Machine Learning
Fall 2014

Thorsten Joachims
Cornell University

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

Outline

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

Supervised Learning vs. Unsupervised Learning

- 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
  - Outlier detection: find unusual events (e.g. hackers)
  - Novelty detection: find changes in data
  - Only instances required, but no labels

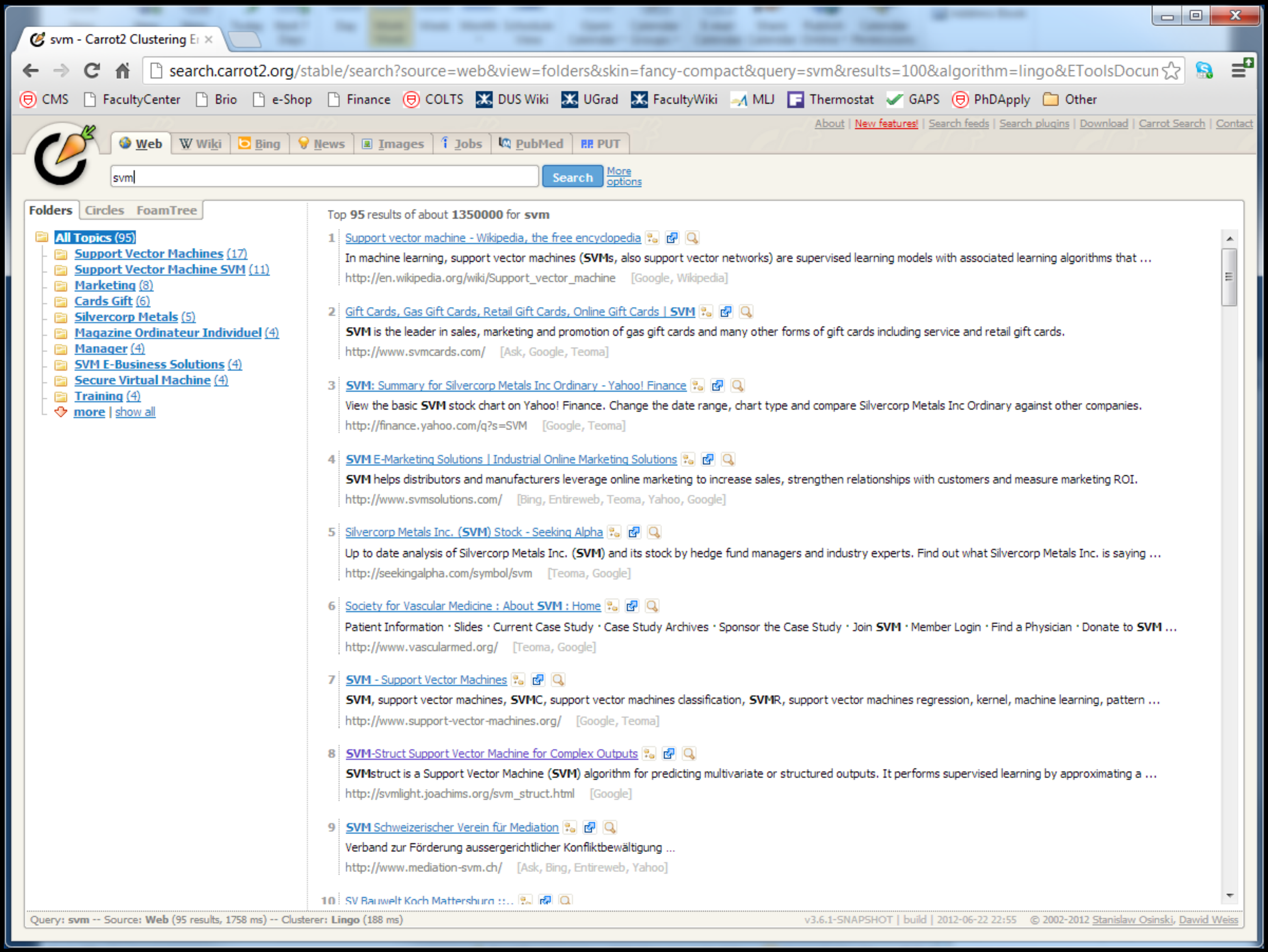
Clustering

- 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).

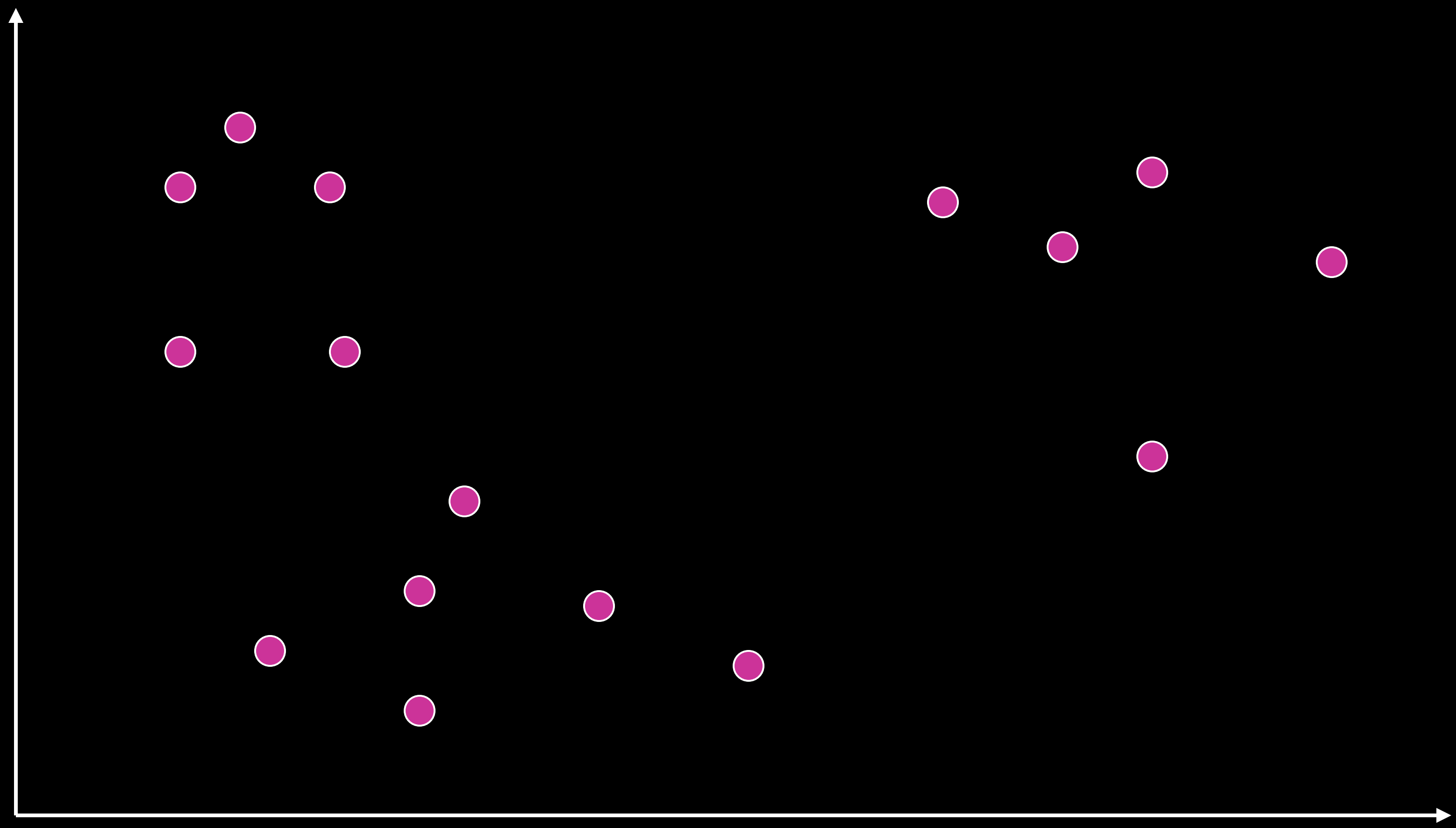
Applications of Clustering

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

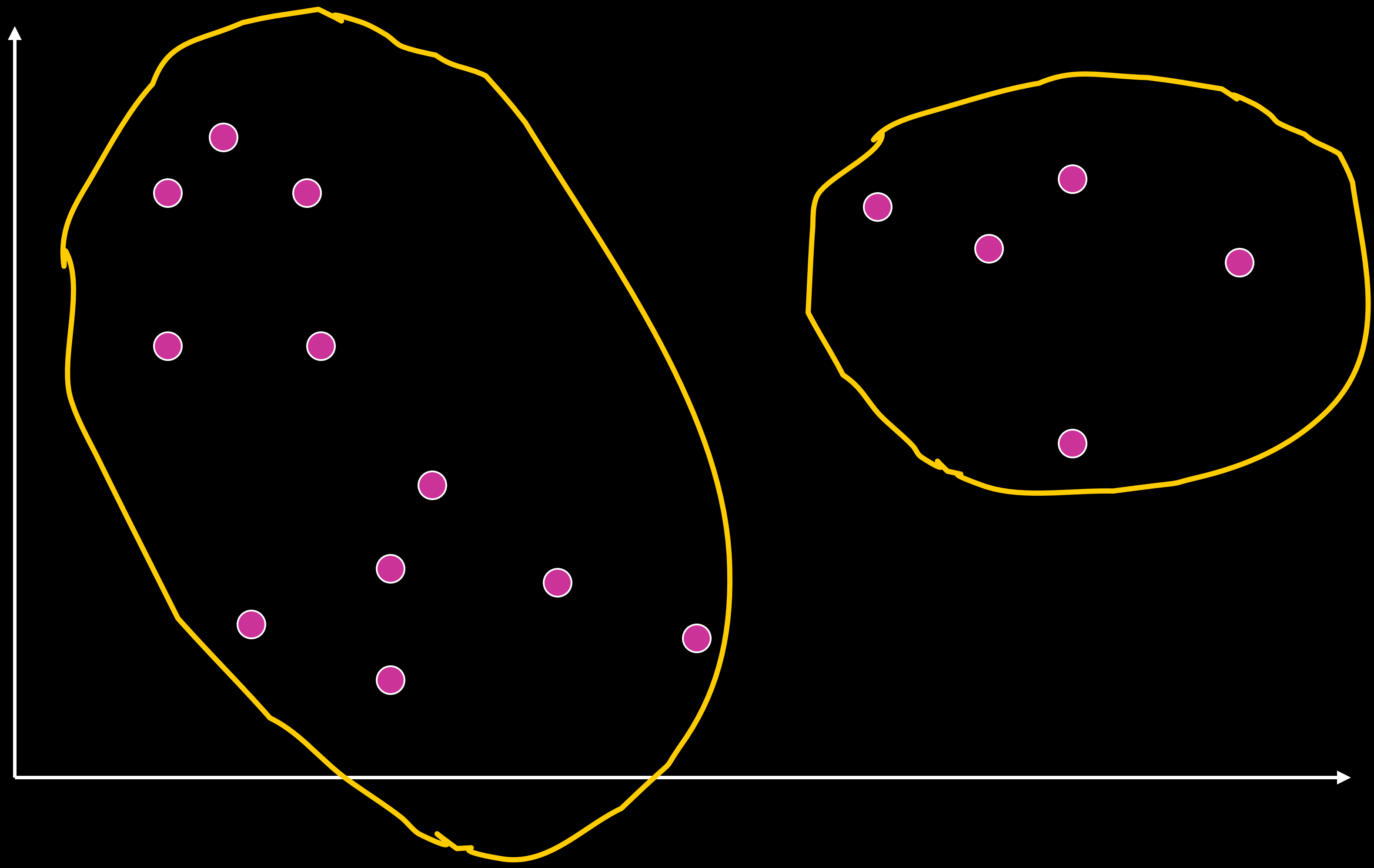
Applications of Clustering



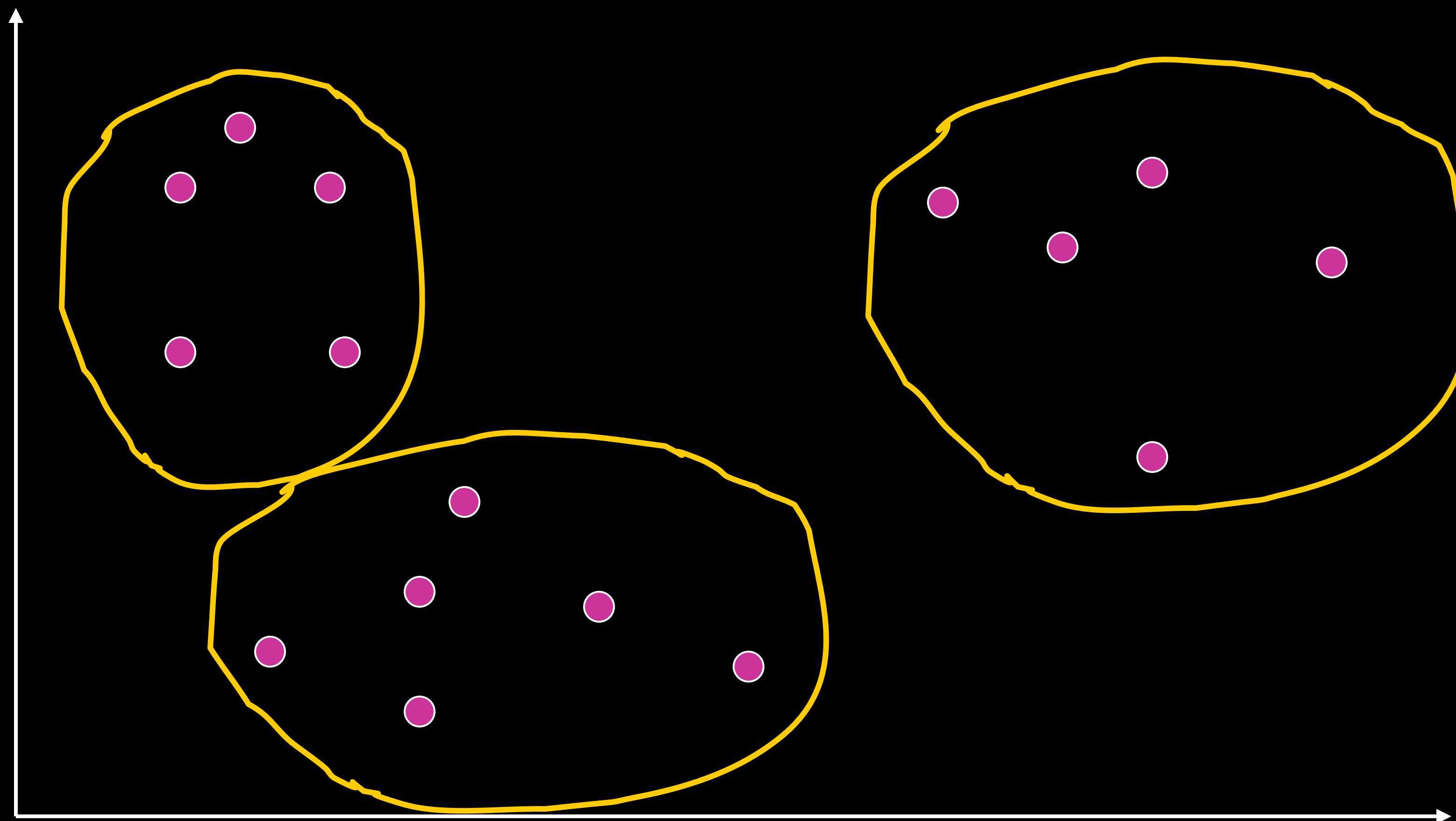
Clustering Example



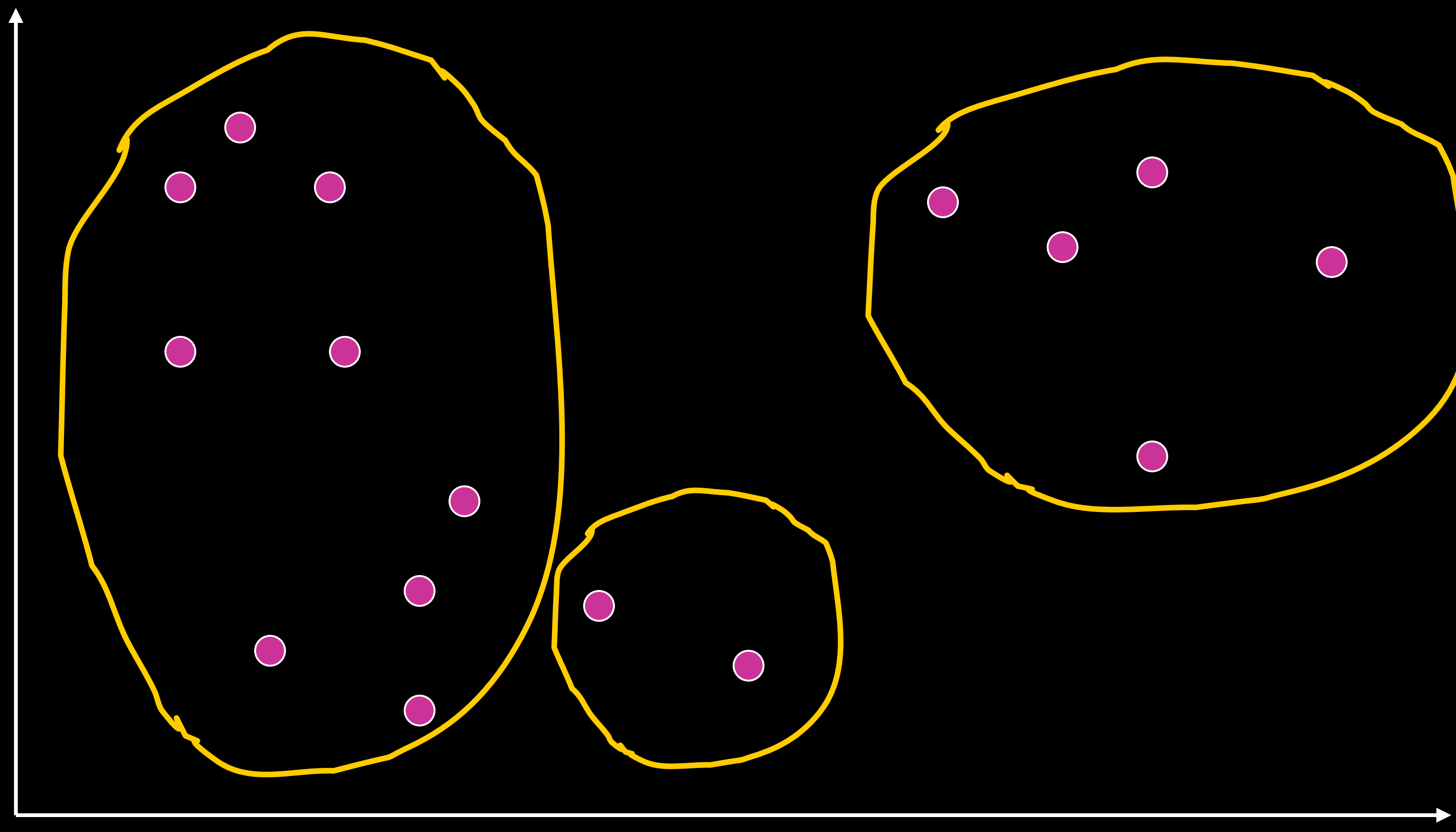
Clustering Example



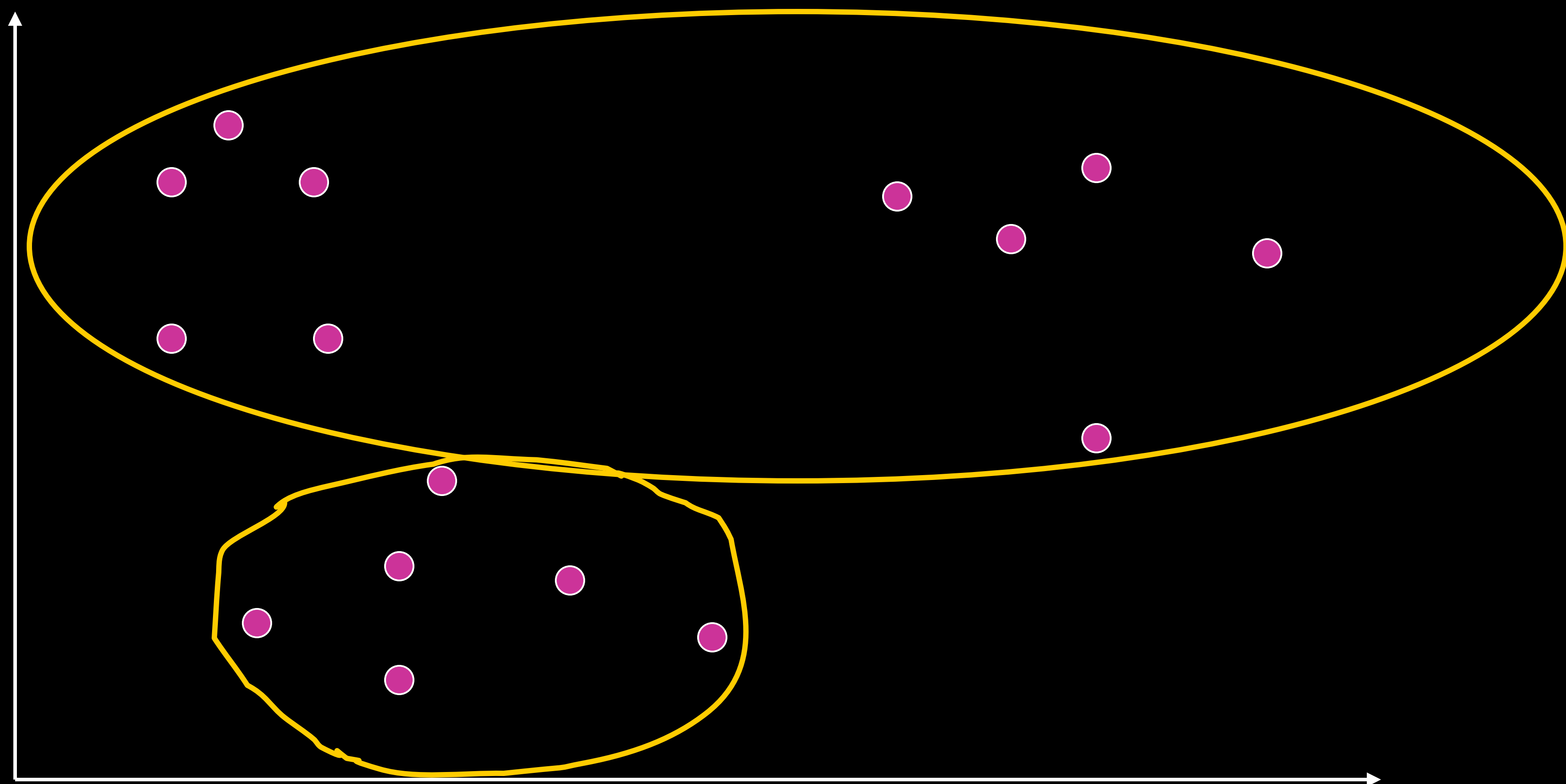
Clustering Example



Clustering Example



Clustering Example



Similarity (Distance) Measures

- Euclidian distance (L_2 norm):

$$L_2(\vec{x}, \vec{x}') = \sqrt{\sum_{i=1}^{N} \left(x_i - x_i'\right)^2}$$

- L_1 norm:

$$L_1(\vec{x}, \vec{x}') = \sqrt{\sum_{i=1}^{N} \left|x_i - x_i'\right|}$$

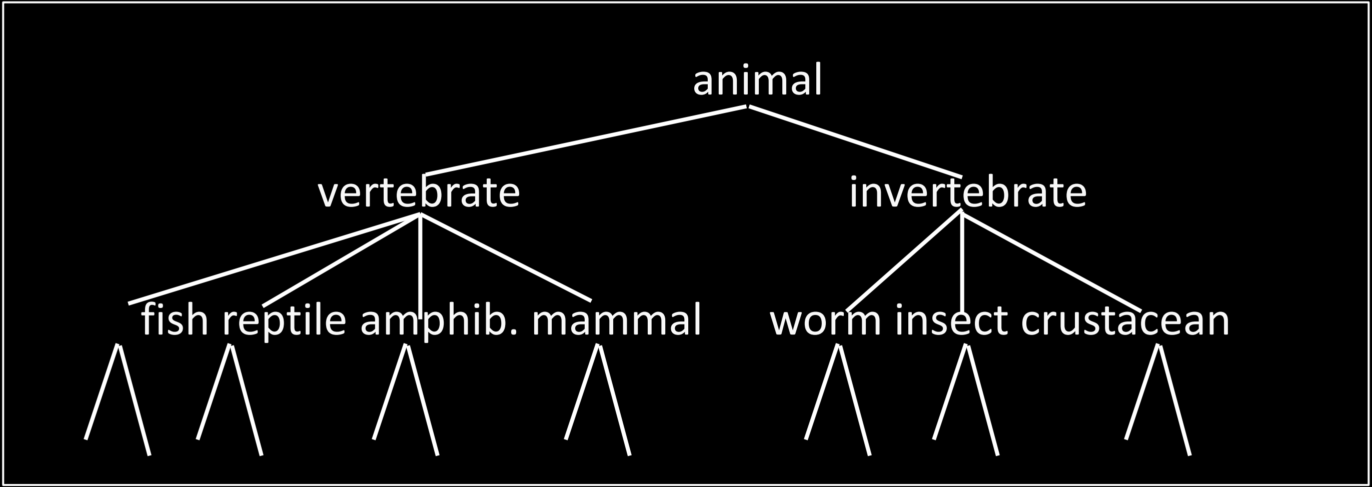
- Cosine similarity:

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

- Kernels

Hierarchical Clustering

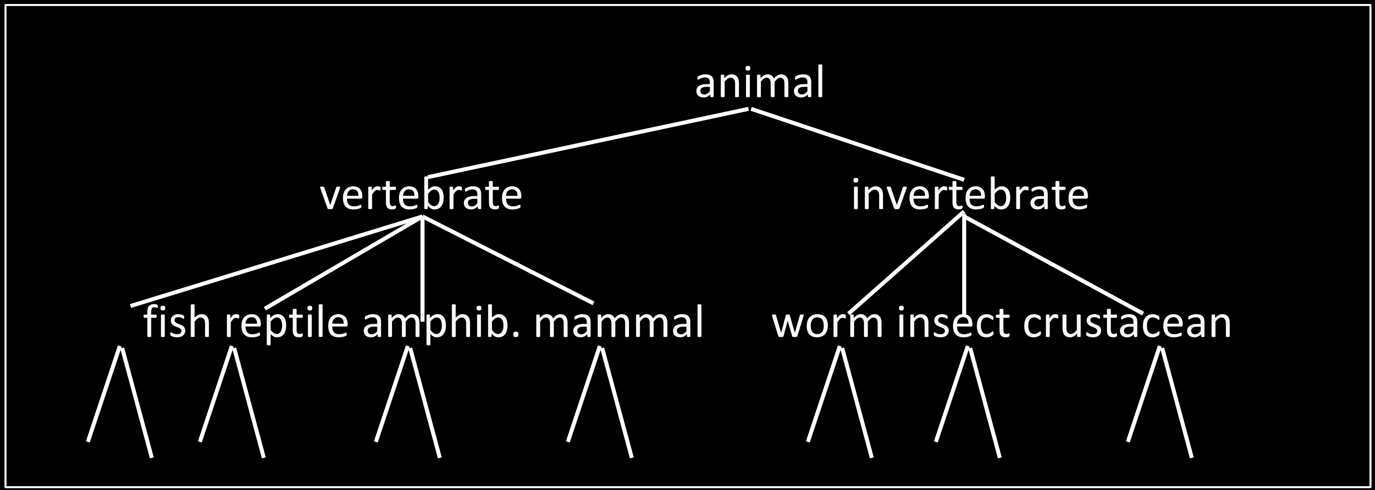
- Build a tree-based hierarchical taxonomy from a set of unlabeled examples.



- 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)

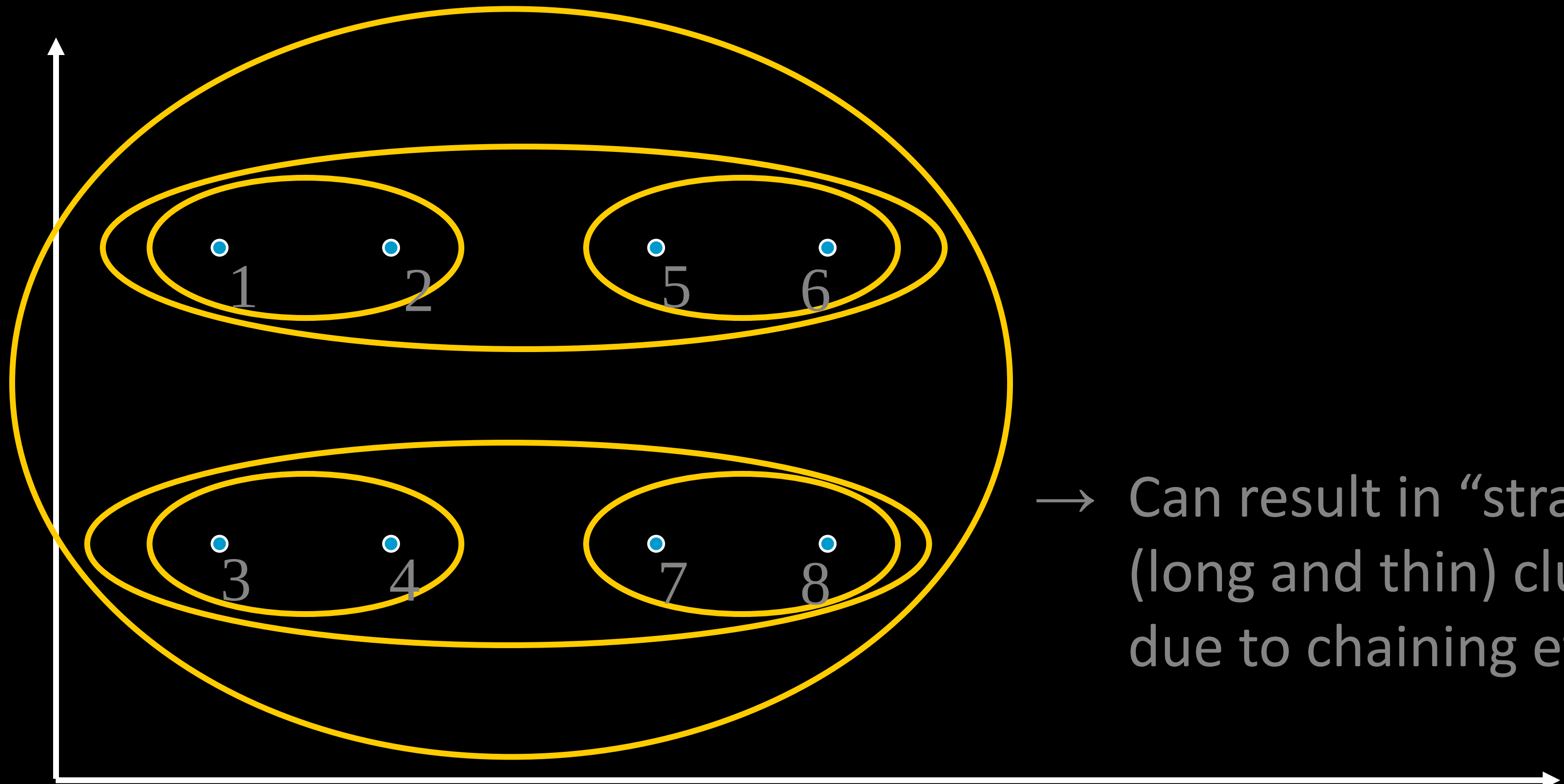
- Assumes a ***similarity function*** for determining the similarity of two clusters.
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.
- Basic 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 $c_i \cup c_j$

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.

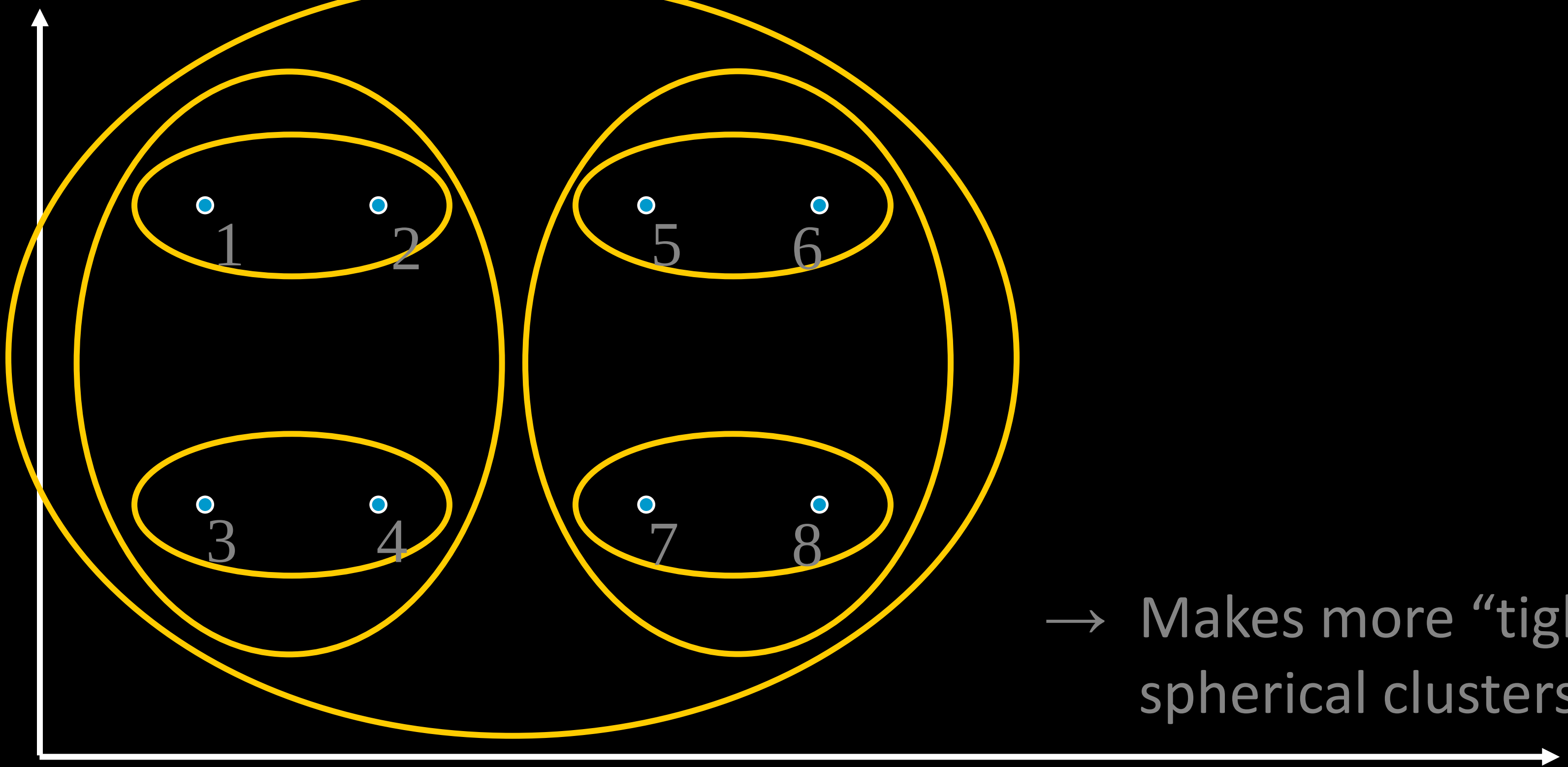
Single-Link HAC



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

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

Complete-Link HAC



→ Makes more "tight," spherical clusters.

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

Computational Complexity of HAC

- 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, must find smallest distance pair of clusters → Maintain heap O(n^2 log n)
- 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. Can this be done in constant time such that O(n^2 log n) overall?

Computing Cluster Similarity

- 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:

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

  - Complete Link:

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

Single-Link Example

	x1	x2	x3	x4	x5
x1	1	0.8	0.2	0.7	0.3
x2	0.8	1	0.1	0.5	0.2
x3	0.2	0.1	1	0.9	0.5
x4	0.7	0.5	0.9	1	0.4
x5	0.3	0.2	0.5	0.4	1

Merge x3,x4
replace with max

	x1	x2	c1	x5
x1	1	0.8	0.7	0.3
x2	0.8	1	0.5	0.2
c1	0.7	0.5	1	0.5
x5	0.3	0.2	0.5	1

Merge x1,x2
replace with max

	c2	c1	x5
c2	1	0.7	0.3
c1	0.7	1	0.5
x5	0.3	0.5	1

Merge c1,c2
replace with max

	c3	x5
c3	1	0.5
x5	0.5	1

Group Average Agglomerative Clustering

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

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

- Compromise between single and complete link.

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

- K-means clustering (“hard”)
- Mixtures of Gaussians and training via Expectation maximization Algorithm (“soft”)