Clustering:
Similarity-Based Clustering

CS4780/5780 – Machine Learning
Fall 2013

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

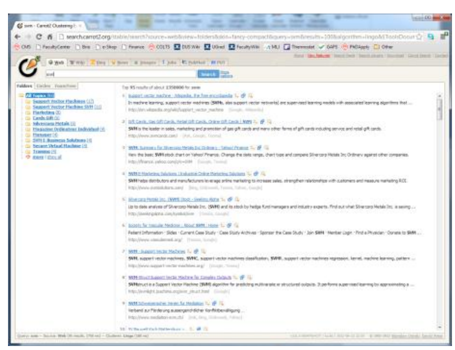
Clustering

- 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).

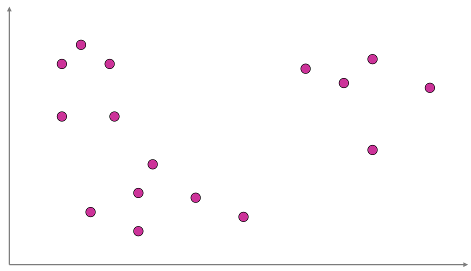
Applications of Clustering

- 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
- Exploratory data analysis
- Automated (or semi-automated) creation of taxonomies
  – e.g. Yahoo, DMOZ
- Compression

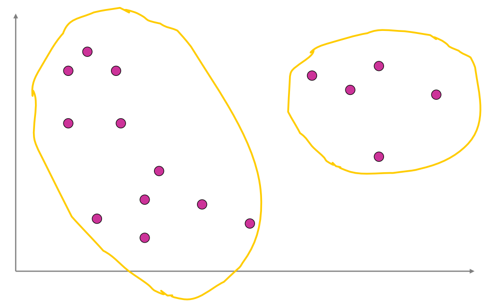
Applications of Clustering



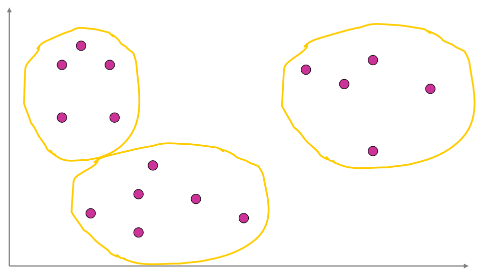
Clustering Example



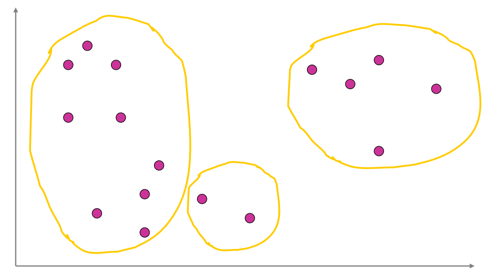
Clustering Example



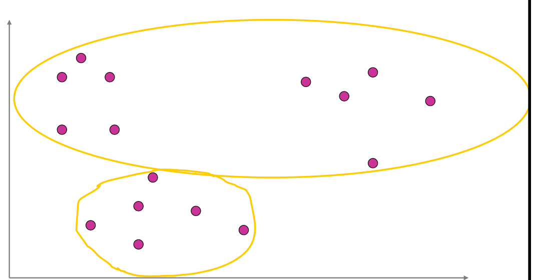
Clustering Example



Clustering Example



Clustering Example



Similarity (Distance) Measures

- Euclidian distance (L_2 norm):

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

- L_1 norm:

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

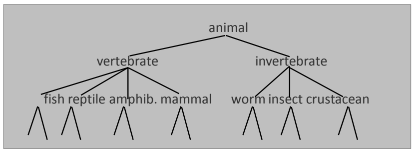
- Cosine similarity:

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

- Kernels

Hierarchical Clustering

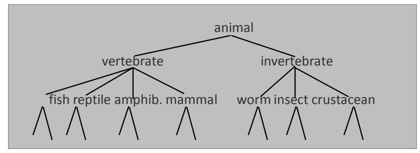
- Build a tree-based hierarchical taxonomy from a set of unlabeled examples.



- 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)

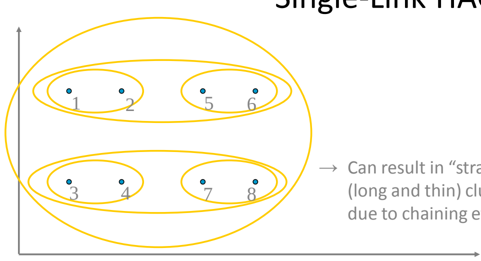
- Assumes a ***similarity function*** for determining the similarity of two clusters.
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.
- Basic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

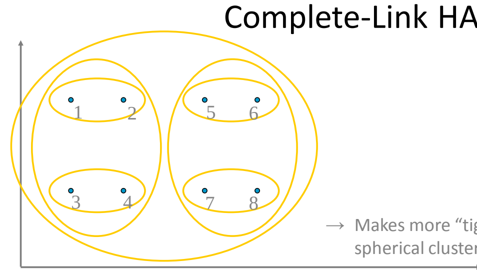
Single-Link HAC



→ 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.

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

- Compromise between single and complete link.

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")

Clustering Criterion

- 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

Centroid-Based Clustering

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

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

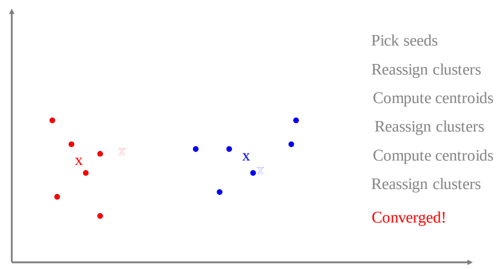
- Reassignment of instances to clusters is based on **distance** to the current cluster centroids.

K-Means Algorithm

- Input: k = number of clusters, distance measure d
- 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 min.
  - For each cluster c_j *//update the centroid of each cluster*
    - $s_j = \mu(c_j)$

K-means Example

(k=2)



Time Complexity

- Assume computing distance between two instances is O(*N*) where *N* is the dimensionality of the vectors.
- Reassigning clusters for *n* points: O(*kn*) distance computations, or O(*knN*).
- Computing centroids: Each instance gets added once to some centroid: O(*nN*).
- Assume these two steps are each done once for *i* iterations: O(*iknN*).
- Linear in all relevant factors, assuming a fixed number of iterations, more efficient than HAC.

Buckshot Algorithm

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

Clustering as Prediction

- Setup
  - Learning Task: $P(X)$
  - Training Sample: $S = (\vec{x}_1, \dots, \vec{x}_n)$
  - Hypothesis Space: $H = \{h_1, \dots, h_{|H|}\}$ each describes $P(X|h_i)$ where h_i are parameters
  - Goal: learn which $P(X|h_i)$ produces the data
- What to predict?
  - Predict where new points are going to fall

Gaussian Mixtures and EM

- Gaussian Mixture Models
  - Assume

    $$P(X = \vec{x}|h_i) = \sum_{j=1}^{k} P(X = \vec{x}|Y = j, h_i)P(Y = j)$$

    where $P(X = \vec{x}|Y = j, h) = N(X = \vec{x}|\vec{\mu}_j, \Sigma_j)$
    and $h = (\vec{\mu}_1, \dots, \vec{\mu}_k, \Sigma_1, \dots, \Sigma_k)$.
- EM Algorithm
  - Assume $P(Y)$ and k known and $\Sigma_i = 1$.
  - REPEAT
    - $\vec{\mu}_j = \frac{\sum_{i=1}^{n} P(Y=j|X=\vec{x}_i,\vec{\mu}_j)\vec{x}_i}{\sum_{i=1}^{n} P(Y=j|X=\vec{x}_i,\vec{\mu}_j)}$
    - $P(Y = j|X = \vec{x}_i, \vec{\mu}_j) = \frac{P(X=\vec{x}_i|Y=j,\vec{\mu}_j)P(Y=j)}{\sum_{l=1}^{k} P(X=\vec{x}_i|Y=l,\vec{\mu}_j)P(Y=l)} = \frac{\mathrm{e}^{-0.5(\vec{x}_i-\vec{\mu}_j)^2} P(Y=j)}{\sum_{l=1}^{k} e^{-0.5(\vec{x}_i-\vec{\mu}_l)^2} P(Y=l)}$