http://www.youtube.com/watch?v=OR_-Y-eIlQo

Machine learning: Unsupervised learning

David Kauchak
cs311
Spring 2013

adapted from:
http://www.stanford.edu/class/cs276/handouts/lecture17-clustering.ppt

Administrative

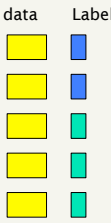
- Assignment 5 out
  - due next Friday… get started now!
  - implement k-means
  - experiment with variants
    - good chance to practice experimentation/analysis

k-means demo

Clustering evaluation

Use data with known classes

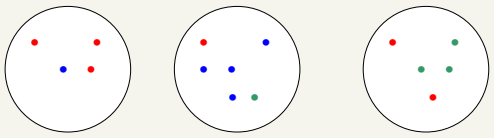
- For example, document classification data



Ideally produce clusters that mimic labels

Common approach: use labeled data

Purity, the proportion of the dominant class in the cluster



Cluster I: Purity = 1/4 (max(3, 1, 0)) = 3/4

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Overall purity?

Overall purity

Cluster I: Purity = 1/4 (max(3, 1, 0)) = 3/4

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Cluster average:

$$\frac{\frac{3}{4}+\frac{4}{6}+\frac{3}{5}}{3} = 0.672$$

Weighted average:

$$\frac{4*\frac{3}{4}+6*\frac{4}{6}+5*\frac{3}{5}}{15} = \frac{3+4+3}{15} = 0.667$$

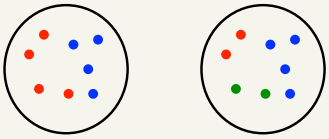
Purity issues…

Purity, the proportion of the dominant class in the cluster

Good for comparing two algorithms, but not understanding how well a single algorithm is doing, why?

- Increasing the number of clusters increases purity

Purity isn't perfect



Which is better based on purity?

Which do you think is better?

Ideas?

Common approach: use labeled data

Average entropy of classes in clusters

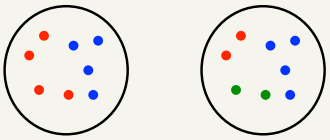
$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$

where $p(class_i)$ is proportion of class *i* in cluster

Common approach: use labeled data

Average entropy of classes in clusters

$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$

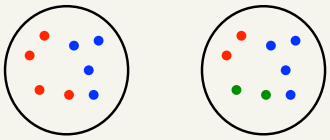


entropy?

Common approach: use labeled data

Average entropy of classes in clusters

$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$



$-0.5\log 0.5 - 0.5\log 0.5 = 1$ $-0.5\log 0.5 - 0.25\log 0.25 - 0.25\log 0.25 = 1.5$

Problems with K-means

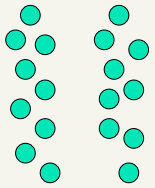
Determining K is challenging

Spherical assumption about the data (distance to cluster center)

Hard clustering isn't always right

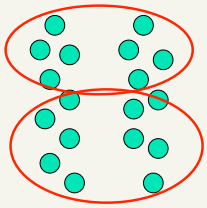
Greedy approach

Problems with K-means



What would K-means give us here?

Assumes spherical clusters

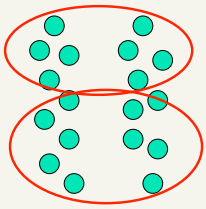


k-means assumes spherical clusters!

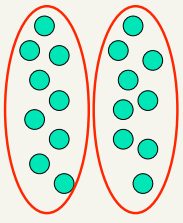
EM clustering: mixtures of Gaussians

Assume data came from a mixture of Gaussians (**elliptical data**), assign data to cluster with a certain *probability*

k-means



EM



Bayesian Classification

We represent a data item based on the features:

$$D = \langle f_1, f_2, \ldots, f_n \rangle$$

Classifying

$$label = \underset{l \in Labels}{\operatorname{argmax}} P(l \mid f_1, f_2, \ldots, f_n)$$

Given an *new* example, the Bayesian model gives us the probability of the model belonging to that class

Unsupervised

We represent a data item based on the features:

$$D = \langle f_1, f_2, \ldots, f_n \rangle$$

$$P(f_1, f_2, \ldots, f_n \mid cluster)$$

How likely is a point to be long to a cluster…

Training a Bayesian Classifier

features	Label
$f_1, f_2, f_3, \ldots, f_n$	0
$f_1, f_2, f_3, \ldots, f_n$	0
$f_1, f_2, f_3, \ldots, f_n$	1
$f_1, f_2, f_3, \ldots, f_n$	1
$f_1, f_2, f_3, \ldots, f_n$	0

train a predictive model

classifier

$p(Label \mid f_1, f_2, \ldots, f_n)$

Training

$P(Label \mid Features) =$ use Bayes rule and learn p(feature|Label)

$P(Features \mid cluster) =$ not as clear here…

EM is a general framework

Create an initial model, θ'

- Arbitrarily, randomly, or with a small set of training examples

Use the model θ' to obtain another model θ such that

$\sum_i \log P_\theta(data_i) > \sum_i \log P_{\theta'}(data_i)$ i.e. better models data (increased log likelihood)

Let θ' = θ and repeat the above step until reaching a local maximum

- Guaranteed to find a better model after each iteration

Where else have you seen EM?

EM shows up all over the place

Training HMMs (Baum-Welch algorithm)

Learning probabilities for Bayesian networks

EM-clustering

Learning word alignments for language translation

Learning Twitter friend network

Genetics

Finance

Anytime you have a model and unlabeled data!

E and M steps: creating a better model

Expectation: Given the current model, figure out the expected probabilities of the data points to each cluster

$p(x|\theta_c)$ What is the probability of each point belonging to each cluster?

Maximization: Given the probabilistic assignment of all the points, estimate a new model, θ_c

Just like NB maximum likelihood estimation, except we use fractional counts instead of whole counts

Similar to *k*-means

Iterate:

Assign/cluster each point to closest center

Expectation: Given the current model, figure out the expected probabilities of the points to each cluster $p(x|\theta_c)$

Recalculate centers as the mean of the points in a cluster

Maximization: Given the probabilistic assignment of all the points, estimate a new model, θ_c

E and M steps

Expectation: Given the current model, figure out the expected probabilities of the data points to each cluster

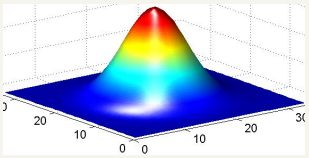
Maximization: Given the probabilistic assignment of all the points, estimate a new model, θ_c

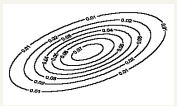
Iterate:

each iterations increases the likelihood of the data and guaranteed to converge (though to a local optimum)!

Model: mixture of Gaussians

$$N[x;\mu,\Sigma] = \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}}\exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)]$$





Covariance determines the shape of these contours

- Fit these Gaussian densities to the data, one per cluster

EM example

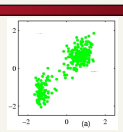


Figure from Chris Bishop

EM example

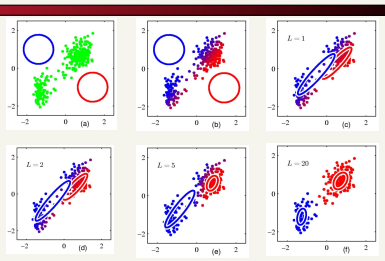


Figure from Chris Bishop

EM

EM is a general purpose approach for training a model when you don't have labels

Not just for clustering!

- K-means is just for clustering

One of the most general purpose unsupervised approaches

- can be hard to get right!

Finding Word Alignments

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

In machine translation, we train from pairs of translated sentences

Often useful to know how the words align in the sentences

Use EM!

- learn a model of P(french-word | english-word)

Finding Word Alignments

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

All word alignments are equally likely

All P(french-word | english-word) equally likely

Finding Word Alignments

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

"la" and "the" observed to co-occur frequently, so P(la | the) is increased.

Finding Word Alignments

"house" co-occurs with both "la" and "maison", but P(maison | house) can be raised without limit, to 1.0, while P(la | house) is limited because of "the"

(pigeonhole principle)

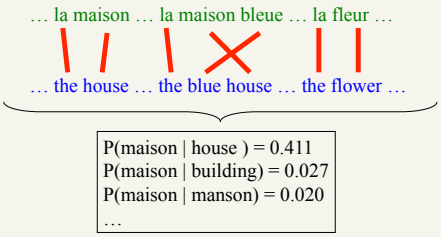
Finding Word Alignments

settling down after another iteration

Finding Word Alignments

Inherent hidden structure revealed by EM training!
For details, see
- "A Statistical MT Tutorial Workbook" (Knight, 1999).
  - 37 easy sections, final section promises a free beer.
- "The Mathematics of Statistical Machine Translation" (Brown et al, 1993)
- Software: GIZA++

Statistical Machine Translation



Estimating the model from training data

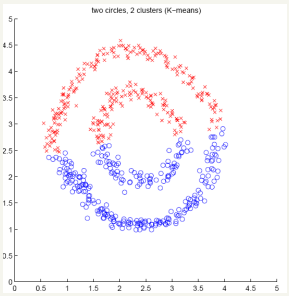
Other clustering algorithms

K-means and EM-clustering are by far the most popular for clustering

However, they can't handle all clustering tasks

What types of clustering problems can't they handle?

Non-gaussian data



What is the problem?

Similar to classification: global decision vs. local decision

Spectral clustering

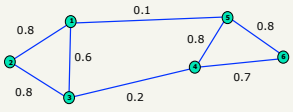
Similarity Graph

Represent dataset as a weighted graph $G(V,E)$

Data points: $\{x_1, x_2, \ldots, x_6\}$

$V=\{x_i\}$ Set of n vertices representing points

$E=\{w_{ij}\}$ Set of weighted edges indicating pair-wise similarity between points



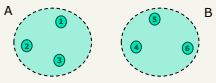
What does clustering represent?

Graph Partitioning

Clustering can be viewed as partitioning a similarity graph

Bi-partitioning task:

- Divide vertices into two disjoint groups (A,B)



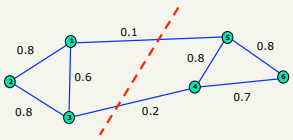
What would define a good partition?

Clustering Objectives

Traditional definition of a "good" clustering:

1. within cluster should be highly similar.
2. between different clusters should be highly dissimilar.

Apply these objectives to our graph representation



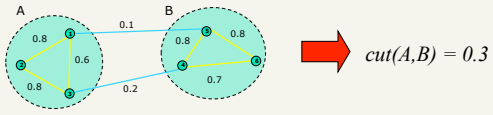
1. Maximise weight of within-group connections
2. Minimise weight of between-group connections

Graph Cuts

Express partitioning objectives as a function of the "edge cut" of the partition.

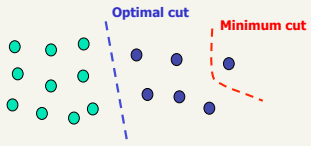
Cut: Set of edges with only one vertex in a group.

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$



Can we use the minimum cut?

Graph Cut Criteria



Problem:

- Only considers external cluster connections
- Does not consider internal cluster density

Graph Cut Criteria

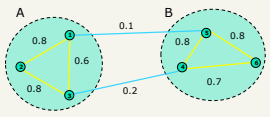
Criterion: Normalised-cut (Shi & Malik,'97)

Consider the connectivity between groups relative to the density of each group:

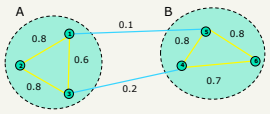
$$\min Ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

Normalize the association between groups by *volume*

- $Vol(A)$: The total weight of the edges within group A



Normalized cut



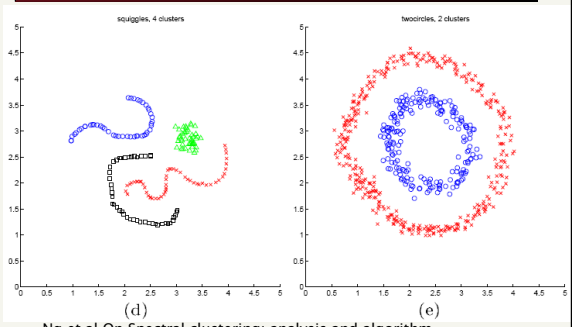
$Vol(A)$: The total weight of the edges originating from group A

minimize between group connections

$$\min Ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

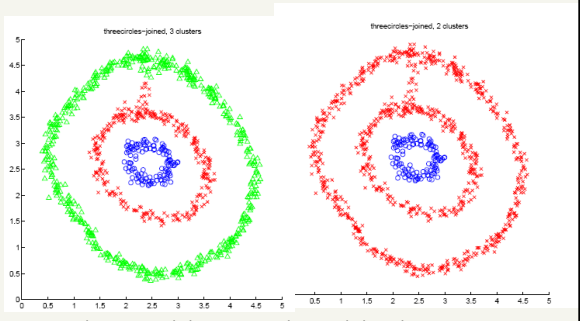
maximize within group connections

Spectral clustering examples



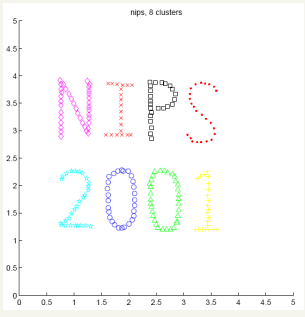
Ng et al On Spectral clustering: analysis and algorithm

Spectral clustering examples



Ng et al On Spectral clustering: analysis and algorithm

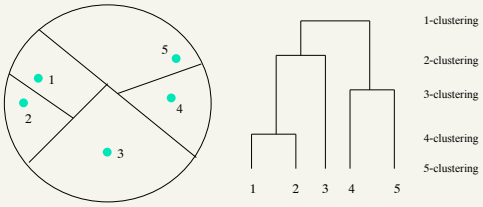
Spectral clustering examples



Ng et al On Spectral clustering: analysis and algorithm

Hierarchical Clustering

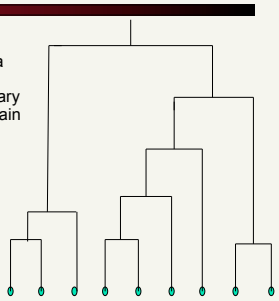
Recursive partitioning/merging of a data set



Dendogram

Represents all partitionings of the data

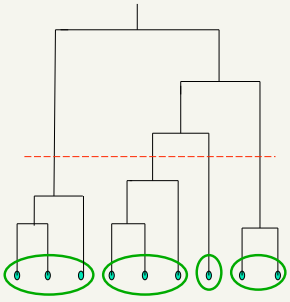
Frequently binary dendograms, but n-ary dendograms are generally easy to obtain with minor changes to the algorithms



Dendogram

Can obtain any k-clustering from the dendogram

Where is the 4-clustering?



Advantages of hierarchical clustering

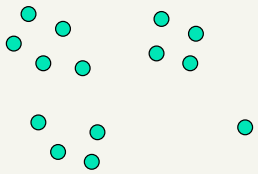
Don't need to specify the number of clusters

Good for data visualization

- See how the data points interact at many levels
- Can view the data at multiple levels of granularity
- Understand how all points interact

Specifies all of the K clusterings/partitions

Hierarchical clustering



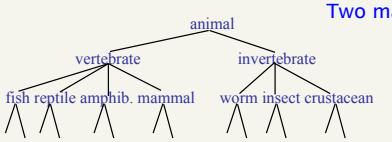
Ideas? How are we going to do this?

Hierarchical Clustering

Common in many domains

- Biologists and social scientists
- Gene expression data
- Document/web page organization
  - DMOZ
  - Yahoo directories

Two main approaches…



Divisive hierarchical clustering

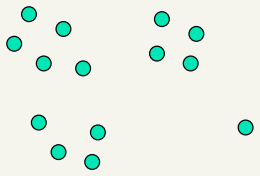
Finding the best partitioning of the data is generally exponential in time

Common approach:

- Let **C** be a set of clusters
- Initialize **C** to be the one-clustering of the data
- While there exists a cluster *c* in **C**
  - remove *c* from **C**
  - partition *c* into 2 clusters using a flat clustering algorithm, c_1 and c_2
  - Add to c_1 and c_2 **C**

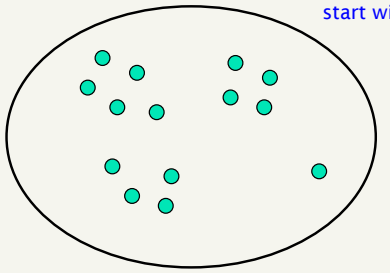
Bisecting k-means

Divisive clustering



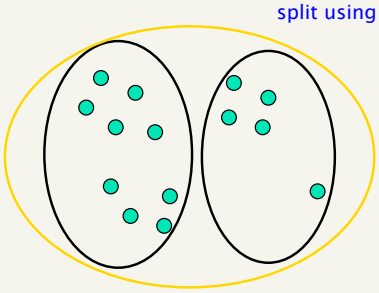
Divisive clustering

start with one cluster

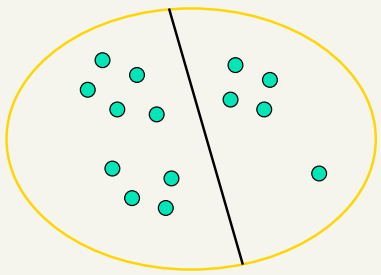


Divisive clustering

split using flat clustering

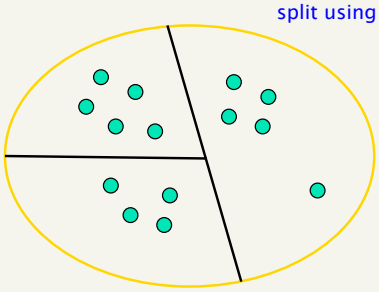


Divisive clustering



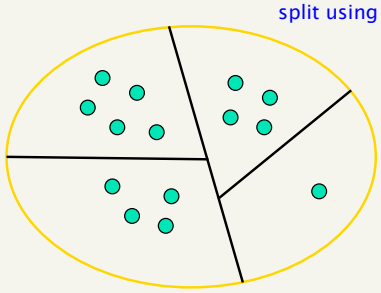
Divisive clustering

split using flat clustering

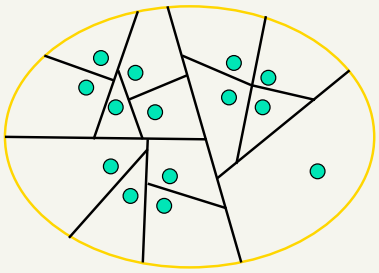


Divisive clustering

split using flat clustering



Divisive clustering



Note, there is a "temporal" component not seen here

Hierarchical Agglomerative Clustering (HAC)

Let **C** be a set of clusters
Initialize **C** to be all points/docs as separate clusters

While **C** contains more than one cluster

- find c_1 and c_2 in **C** that are closest together
- remove c_1 and c_2 from **C**
- merge c_1 and c_2 and add resulting cluster to **C**

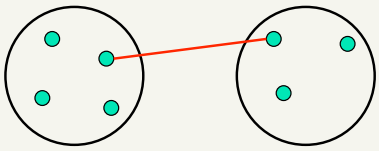
The history of merging forms a binary tree or hierarchy

How do we measure the distance between clusters?

Distance between clusters

Single-link

- Similarity of the *most* similar (single-link)



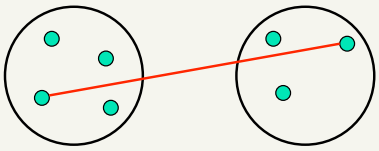
$$\max_{l \in L, r \in R} sim(l,r)$$

Distance between clusters

Complete-link

- Similarity of the "furthest" points, the *least* similar

$$\min_{l \in L, r \in R} sim(l,r)$$

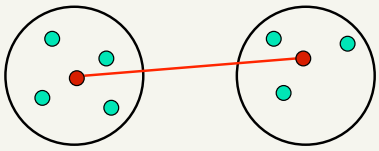


Why are these "local" methods used? efficiency

Distance between clusters

Centroid

- Clusters whose centroids (centers of gravity) are the most similar

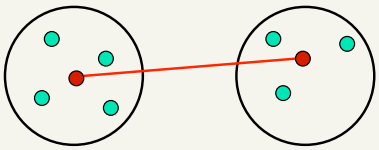


$$\|\mu(L) - \mu(R)\|^2$$

Distance between clusters

Centroid

- Clusters whose centroids (centers of gravity) are the most similar

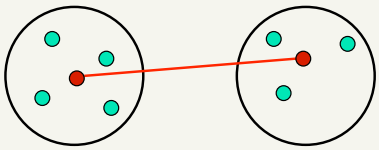


$$\frac{|L| \cdot |R|}{|L| + |R|} \|\mu(L) - \mu(R)\|^2$$ Ward's method What does this do?

Distance between clusters

Centroid

- Clusters whose centroids (centers of gravity) are the most similar

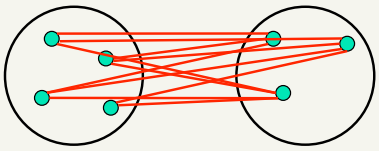


$$\frac{|L| \cdot |R|}{|L| + |R|} \|\mu(L) - \mu(R)\|^2$$ Ward's method Encourages similar sized clusters

Distance between clusters

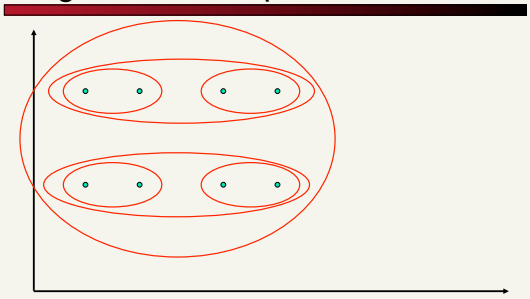
Average-link

- Average similarity between all pairs of elements

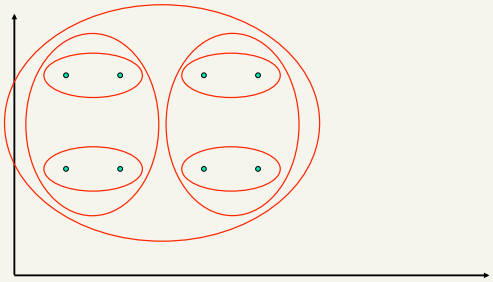


$$\frac{1}{|L| \cdot |R|} \sum_{x \in L, y \in R} \|x - y\|^2$$

Single Link Example



Complete Link Example



Computational Complexity

For

- *m* dimensions
- *n* documents/points

How many iterations?

- *n-1* iterations

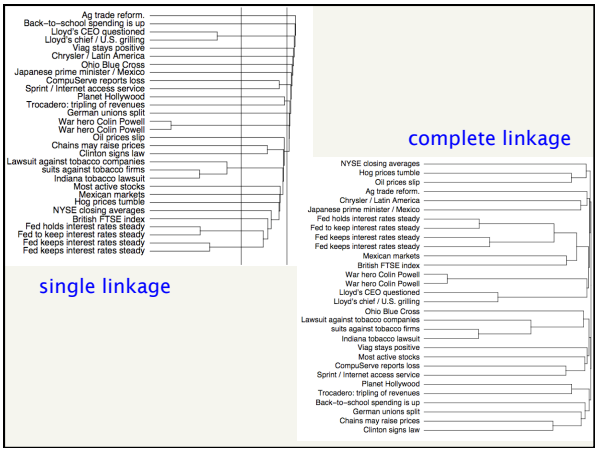
First iteration

- Need to compute similarity of all pairs of *n* points/documents: $O(n^2m)$

Remaining *n*-2 iterations

- compute the distance between the most recently created cluster and all other existing clusters: $O(nm)$
- Does depend on the cluster similarity approach

Overall run-time: $O(n^2m)$ – generally slower than flat clustering!



Problems with hierarchical clustering