17s1: COMP9417 Machine Learning and Data Mining

Unsupervised Learning

May 2, 2017

Acknowledgement: Material derived from slides for the book
Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997
`http://www-2.cs.cmu.edu/~tom/mlbook.html`

and slides by Andrew W. Moore available at
`http://www.cs.cmu.edu/~awm/tutorials`

and the book Data Mining, Ian H. Witten and Eibe Frank,
Morgan Kauffman, 2000. `http://www.cs.waikato.ac.nz/ml/weka`

and the book Pattern Classification, Richard O. Duda, Peter E. Hart,
and David G. Stork. Copyright (c) 2001 by John Wiley & Sons, Inc.

and the book "Elements of Statistical Learning", Trevor Hastie,
Robert Tibshirani and Jerome Friedman. (c) 2001, Springer.

Aims

This lecture will introduce you to statistical and graphical methods for clustering of "unlabelled" instances in machine learning. Following it you should be able to:

- describe the problem of unsupervised learning
- describe k-means clustering
- describe the role of the EM algorithm in k-means clustering
- describe hierarchical clustering
- describe conceptual clustering

Relevant WEKA programs:
weka.clusterers.EM, SimpleKMeans, Cobweb

Unsupervised vs. Supervised Learning

Informally *clustering* is assignment of objects to classes on basis of observations about objects only, i.e. not given "labels" of the categories of objects by a "teacher".

Unsupervised learning classes initially *unknown* and need to be "discovered" from the data: cluster analysis, class discovery, unsupervised pattern recognition.

Supervised learning classes *predefined* and need a "definition" in terms of the data which is used for prediction: classification, discriminant analysis, class prediction, supervised pattern recognition.

Why unsupervised learning ?

- if labelling expensive, train with small labelled sample then improve with large unlabelled sample
- if labelling expensive, train with large unlabelled sample then learn classes with small labelled sample
- tracking "concept drift" over time by unsupervised learning
- learn new "features" by clustering for later use in classification
- exploratory data analysis with visualization

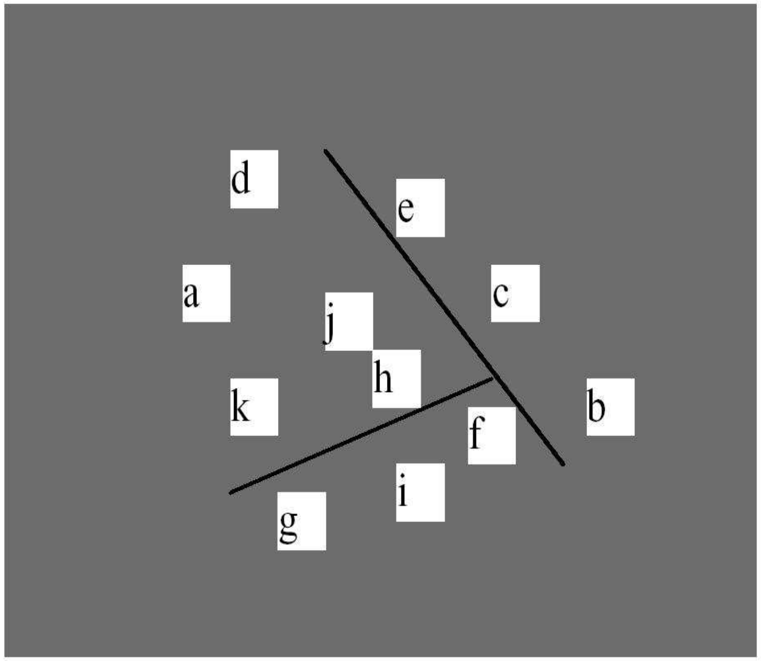
Note: sometimes the term "classification" is used to mean unsupervised discovery of classes or clusters

Clustering

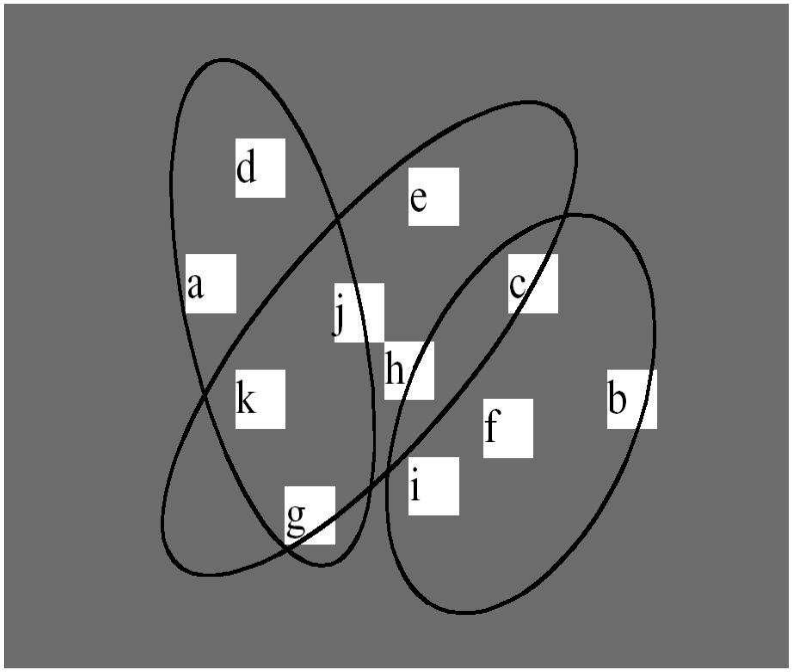
- Finding groups of items that are similar
- Clustering is unsupervised
  - The class of an example is not known
- Success of clustering often measured subjectively
  - this is problematic . . .
  - there are statistical & other approaches . . .
- A data set for clustering is just like a data set for classification, without the class

Representing clusters

Simple 2-D representation

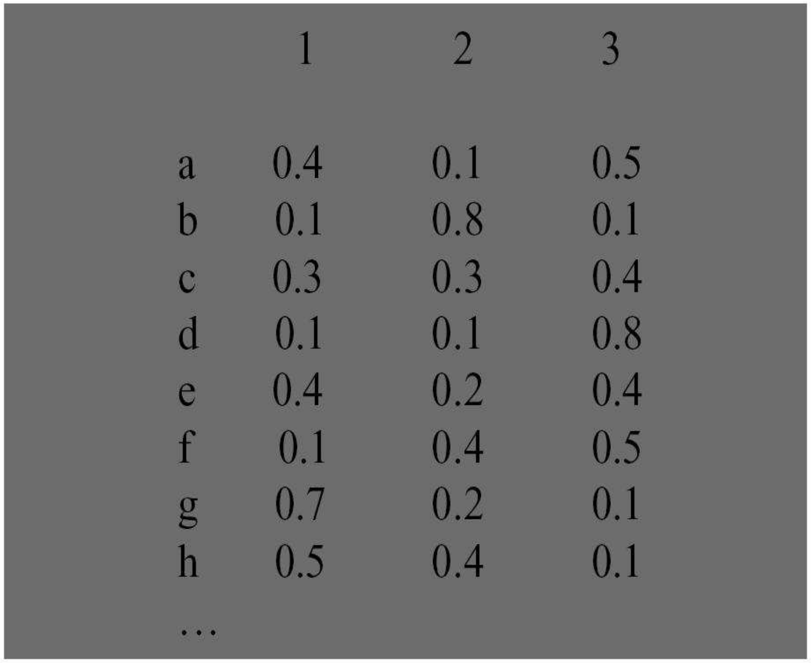


Venn diagram (Overlapping clusters)



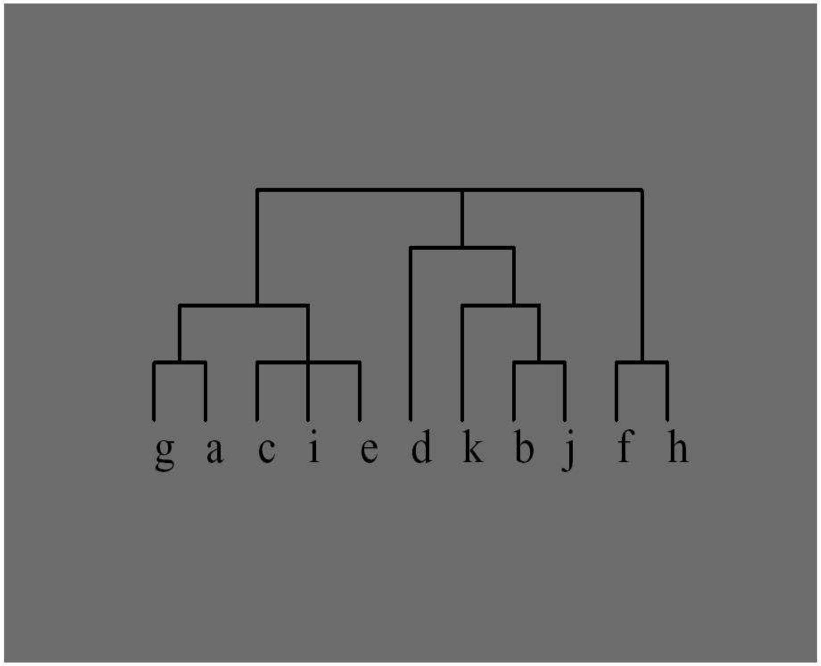
Representing clusters

Probabilistic assignment



	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1
…			

Dendrogram



Cluster analysis

Clustering algorithms form two broad categories: **hierarchical methods** and **partitioning methods**.

Hierarchical algorithms are either **agglomerative** i.e. bottom-up or **divisive** i.e. top-down.

In practice, hierarchical agglomerative methods often used - efficient exact algorithms available.

Partitioning methods usually require specification of no. of clusters, then try to construct the clusters and fit objects to them.

Representation

Let $N = \{e_1, \ldots, e_n\}$ be a set of elements, i.e. instances.

Let $\mathcal{C} = (C_1, \ldots, C_l)$ be a *partition* of N into subsets.

Each subset is called a *cluster*, and $\mathcal{C}$ is called a *clustering*.

Input data can have two forms:

1. each element is associated with a real-valued vector of p features e.g. measurement levels for different features
2. pairwise similarity data between elements, e.g. correlation, distance (dissimilarity)

Feature-vectors have more information, but similarity is generic (given the appropriate function). Feature-vector matrix: $N \times p$, similarity matrix $N \times N$. In general, often $N >> p$.

Clustering framework

The goal of clustering is to find a partition of N elements (instances) into homogeneous and well-separated clusters. Elements from same cluster should have high similarity, elements from different cluster low similarity. Note: homogeneity and separation not well-defined. In practice, depends on the problem. Also, there are typically interactions between homogeneity and separation - usually, high homogeneity is linked with low separation, and vice versa.

k-means clustering

Set value for k, the number of clusters (by prior knowledge or via search)

Initialise: choose points for centres (means) of k clusters (at random)

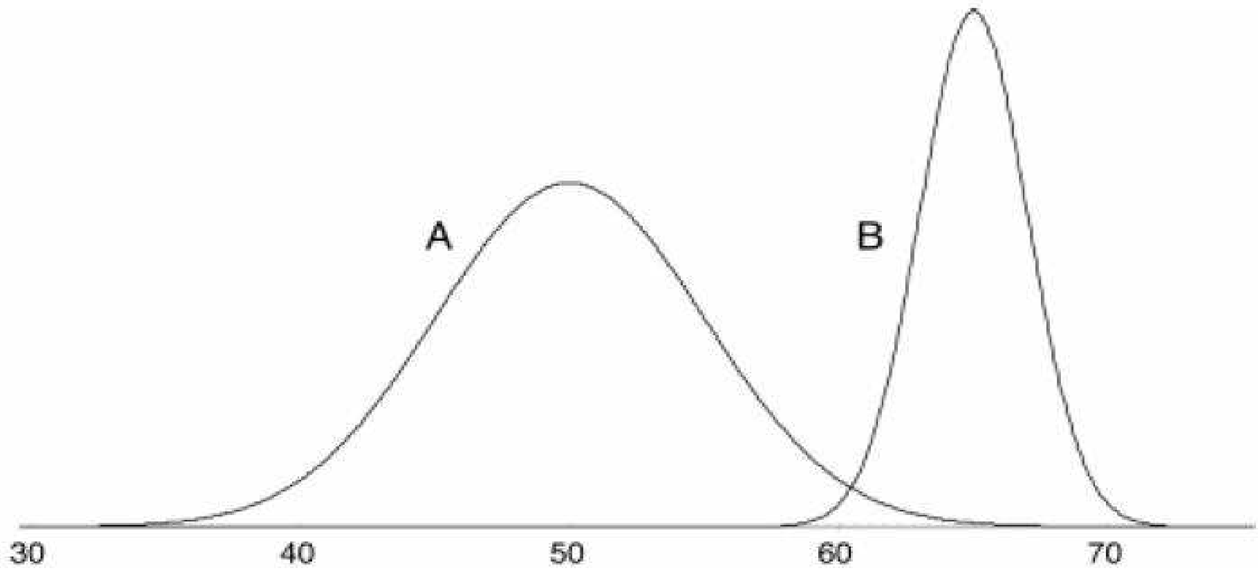
Procedure:

1. assign each instance x to the closest of the k points
2. re-assign the k points to be the means of each of the k clusters
3. repeat 1 and 2 until convergence to a reasonably stable clustering

Example: one variable 2-means (& standard deviations)

A	51	B	62	B	64	A	48	A	39	A	51
A	43	A	47	A	51	B	64	B	62	A	48
B	62	A	52	A	52	A	51	B	64	B	64
B	64	B	64	B	62	B	63	A	52	A	42
A	45	A	51	A	49	A	43	B	63	A	48
A	42	B	65	A	48	B	65	B	64	A	41
A	46	A	48	B	62	B	66	A	48		
A	45	A	49	A	43	B	65	B	64		
A	45	A	46	A	40	A	46	A	48		

model



μ_A=50, σ_A =5, p_A=0.6 μ_B=65, σ_B =2, p_B=0.4

k-means clustering

$P(i)$ is the cluster assigned to element i, $c(j)$ is the centroid of cluster j, $d(v_1, v_2)$ the Euclidean distance between feature vectors v_1 and v_2. The goal is to find a partition P for which the error (distance) function $E_P = \sum_{i=1}^{n} d(i, c(P(i))$ is minimum.

The centroid is the mean or weighted average of the points in the cluster.

k-means very popular clustering tool in many different areas.

Note: can be viewed in terms of the widely-used EM (Expectation-Maximization) algorithm.

k-means clustering algorithm

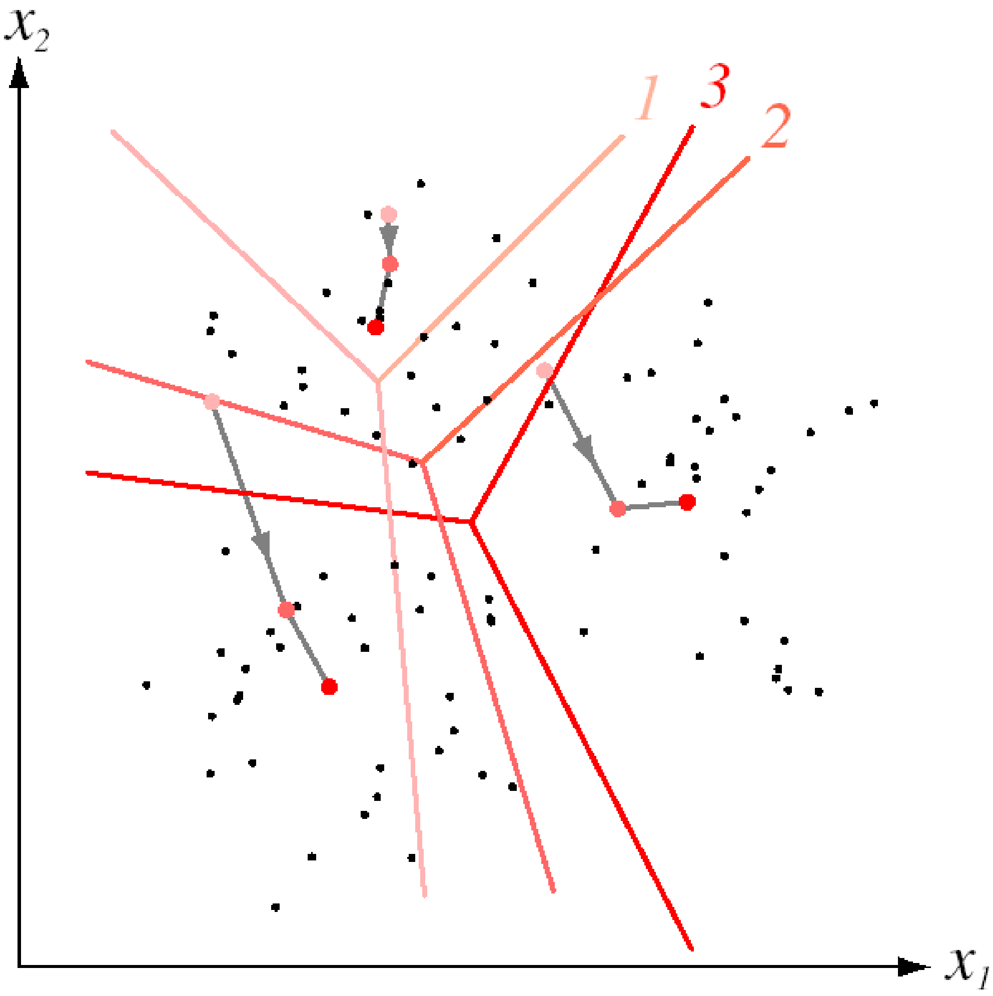
```
Algorithm      $k$-means

/* feature-vector matrix $M(ij)$ is given */


1. Start with an arbitrary partition $P$ of $N$ into $k$ clusters


2. for each element $i$ and cluster $j \neq P(i)$ let $E_P^{ij}$ be the
   cost of a solution in which $i$ is moved to $j$:
  (a) if $E_P^{i^*j^*} = min_{ij} E_P^{ij} < E_P$ then move $i^*$ to cluster $j^*$ and
      repeat step 2 else halt.
```

k-means clustering



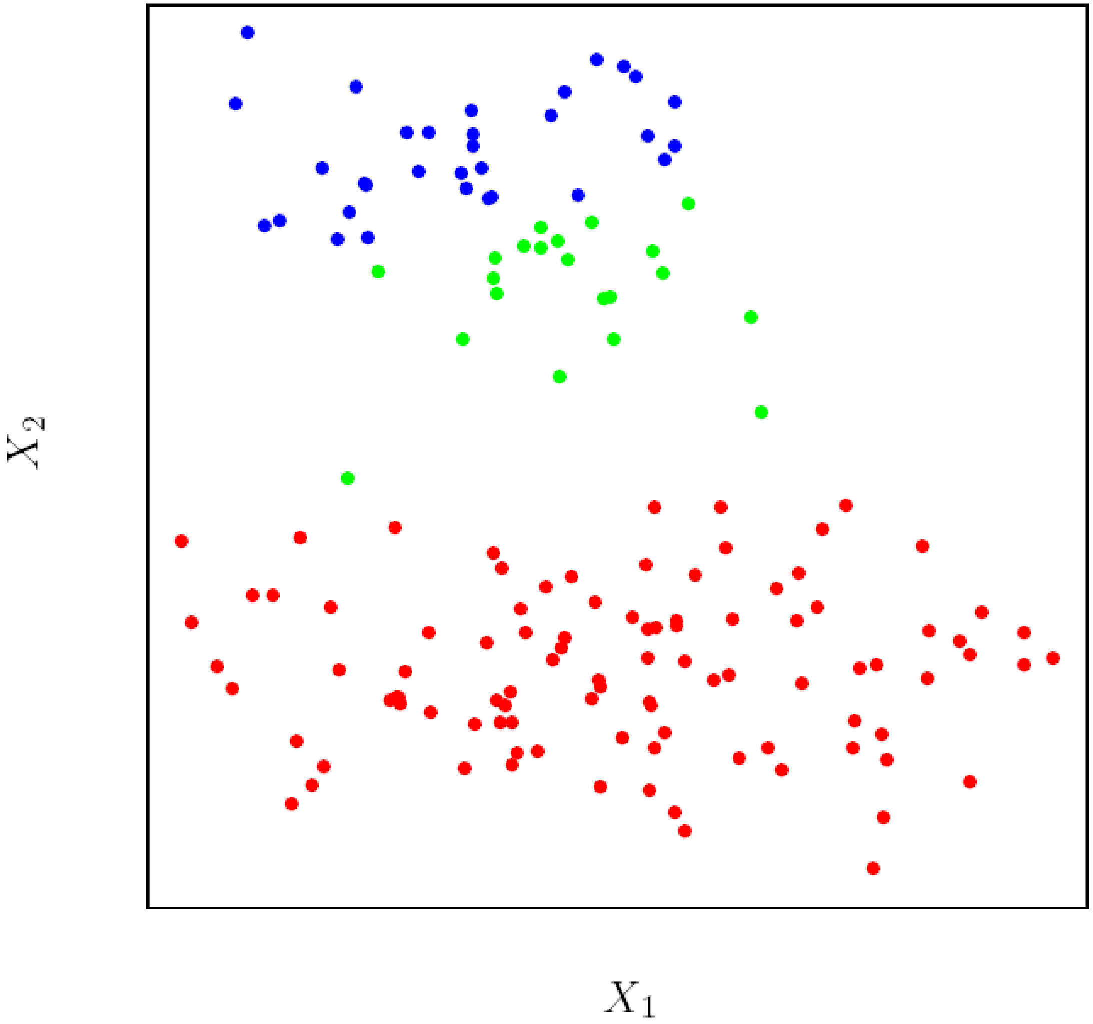
k-means clustering

Previous diagram shows three steps to convergence in k-means with $k = 3$

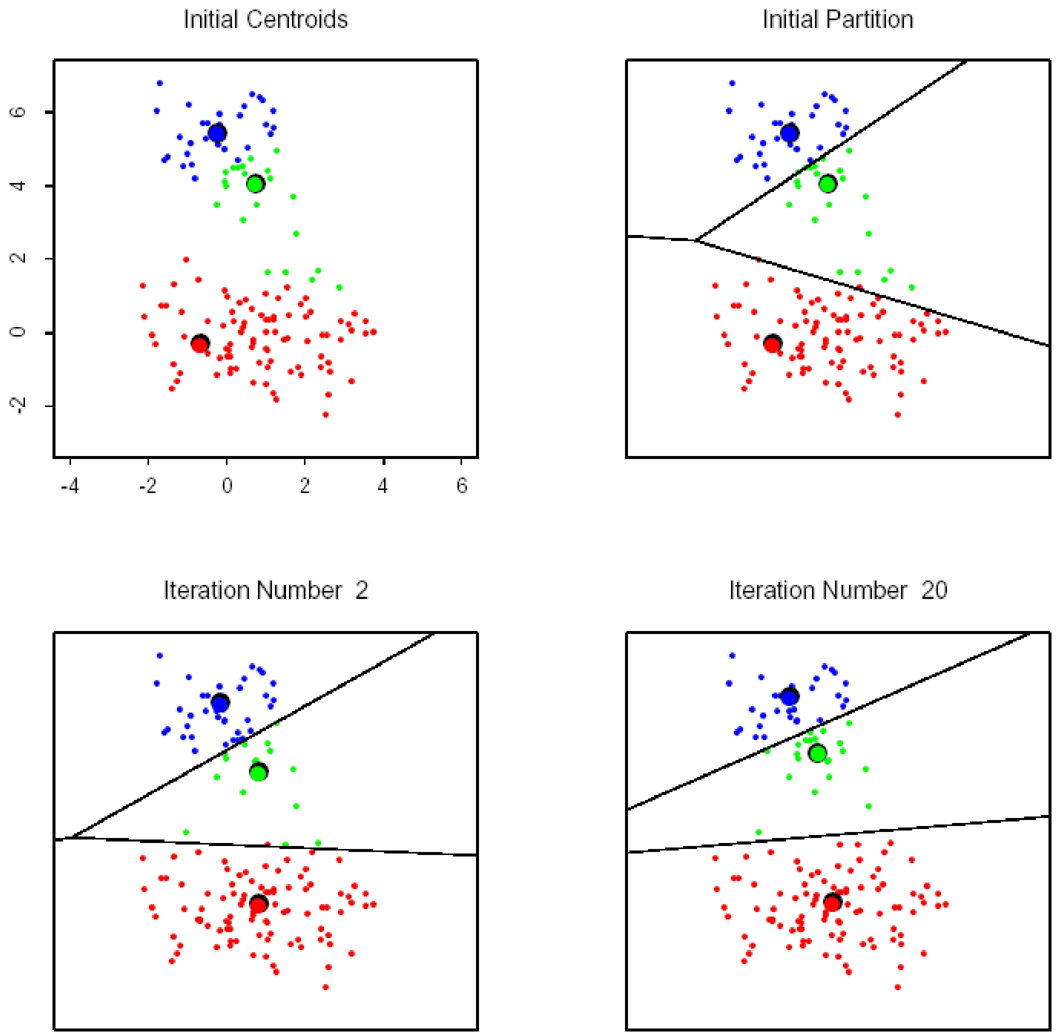
- means move to minimize squared-error criterion
- approximate method of obtaining maximum-likelihood estimates for means
- each point assumed to be in exactly one cluster
- if clusters "blend", fuzzy k-means (i.e., overlapping clusters)

Next diagrams show convergence in k-means with $k = 3$ for data with two clusters not well separated

k-means clustering



k-means clustering



Initial Centroids

Initial Partition

Iteration Number 2

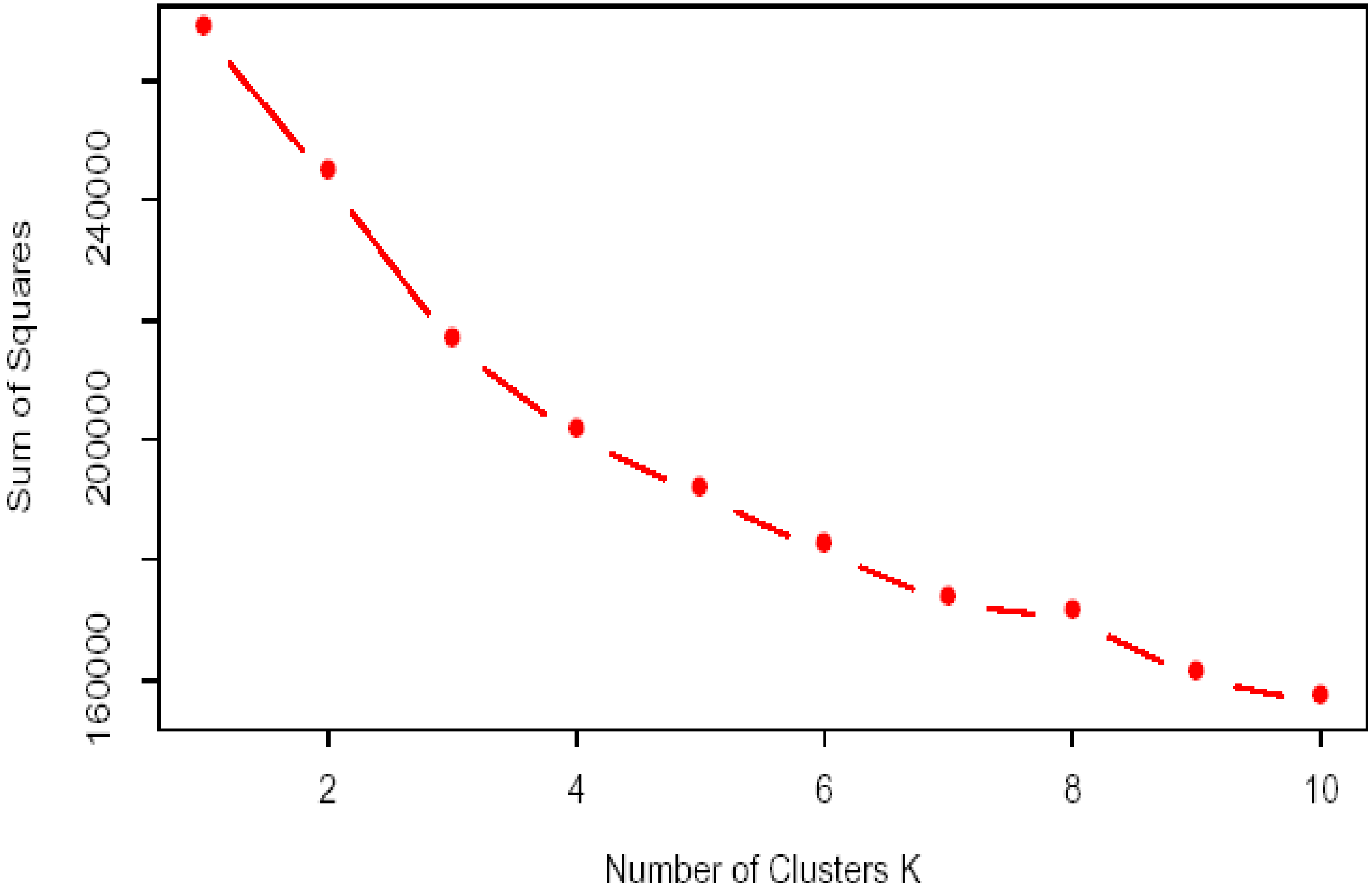
Iteration Number 20

k-means clustering

Trying to *minimize* a loss function in which the goal of clustering is *not* met

- running on microarray data of 6830×64 matrix
- total within-cluster sum-of-squares is reduced for $k = 1$ to 10
- no obvious “correct” k

k-means clustering



Practical k-means

- Result can vary significantly based on initial choice of seeds
- Algorithm can get trapped in a local minimum
  - Example: four instances at the vertices of a twodimensional rectangle
    - Local minimum: two cluster centers at the midpoints of the rectangle's long sides
- Simple way to increase chance of finding a global optimum: restart with different random seeds
  - can be time-consuming

Expectation Maximization (EM)

When to use:

- Data is only partially observable
- Unsupervised learning, e.g., clustering (class value "unobservable")
- Supervised learning (some instance attributes unobservable)

Some uses:

- Train Bayesian Belief Networks
- Unsupervised clustering (k-means, AUTOCLASS)
- Learning Hidden Markov Models (Baum-Welch algorithm)

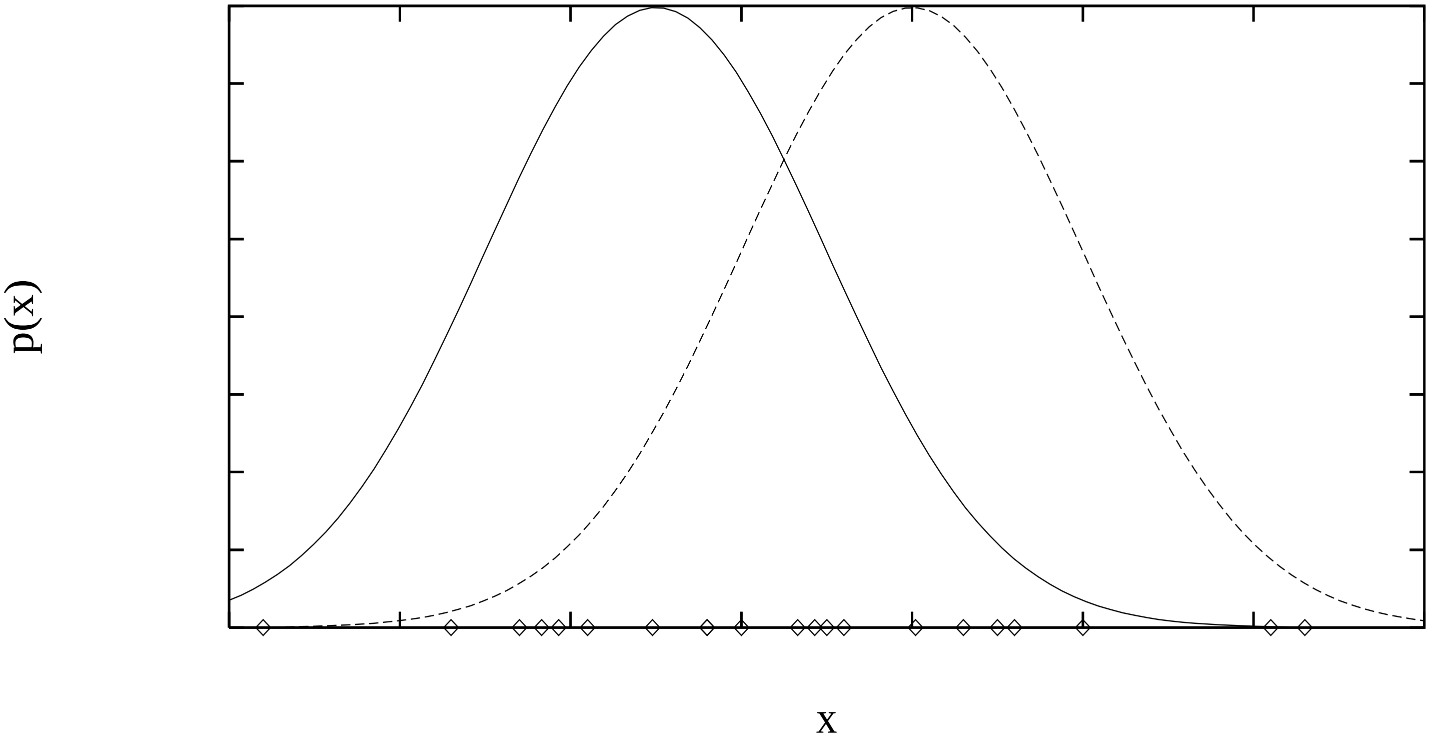
Finite mixtures

Each instance x generated by

1. Choosing one of the k Gaussians with uniform probability
2. Generating an instance at random according to that Gaussian

Called *finite mixtures* because there is only a finite number of *generating distributions* being represented.

Generating Data from Mixture of k Gaussians



EM for Estimating k Means

Given:

- Instances from X generated by mixture of k Gaussian distributions
- Unknown means $\langle \mu_1, \ldots, \mu_k \rangle$ of the k Gaussians
- Don't know which instance x_i was generated by which Gaussian

Determine:

- Maximum likelihood estimates of $\langle \mu_1, \ldots, \mu_k \rangle$

EM for Estimating k Means

Think of full description of each instance as $y_i = \langle x_i, z_{i1}, z_{i2} \rangle$, where

- z_{ij} is 1 if x_i generated by jth Gaussian, otherwise zero
- x_i observable, from instance set $x_1, x_2, \ldots, x_m$
- z_{ij} unobservable

EM for Estimating k Means

Initialise: Pick random initial $h = \langle \mu_1, \mu_2 \rangle$

Iterate:

E step: Calculate expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds:

$$\begin{aligned} E[z_{ij}] &= \frac{p(x = x_i|\mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i|\mu = \mu_n)} \\ &= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}} \end{aligned}$$

EM for Estimating k Means

M step: Calculate new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming value taken on by each hidden variable z_{ij} is expected value $E[z_{ij}]$ calculated before. Replace $h = \langle \mu_1, \mu_2 \rangle$ by $h' = \langle \mu'_1, \mu'_2 \rangle$.

$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] \quad x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

i.e.

$$\mu_j \leftarrow \frac{1}{m} \sum_{i=1}^{m} E[z_{ij}] x_i$$

EM for Estimating k Means

E step: Calculate probabilities for unknown parameters for each instance

M step: Estimate parameters based on the probabilities

In k-means the probabilities are stored as instance weights.

EM Algorithm

Converges to local maximum likelihood h

and provides estimates of hidden variables z_{ij}

In fact, local maximum in $E[\ln P(Y|h)]$

- Y is complete (observable plus unobservable variables) data
- Expected value taken over possible values of unobserved variables in Y

General EM Problem

Given:

- Observed data $X = \{x_1, \ldots, x_m\}$
- Unobserved data $Z = \{z_1, \ldots, z_m\}$
- Parameterized probability distribution $P(Y|h)$, where
  - $Y = \{y_1, \ldots, y_m\}$ is the full data $y_i = x_i \cup z_i$
  - h are the parameters

Determine:

- h that (locally) maximizes $E[\ln P(Y|h)]$

EM for Estimating k Means

Many uses:

- Train Bayesian belief networks
- Unsupervised clustering (e.g., k means)
- Hidden Markov Models

Extending the mixture model

- Using more than two distributions
- Several attributes: easy if independence assumed
- Correlated attributes: difficult
  - Modeled jointly using a bivariate normal distribution with a (symmetric) covariance matrix
  - With n attributes this requires estimating $n + n(n+1)/2$ parameters

Extending the mixture model

- Nominal attributes: easy if independence assumed
- Correlated nominal attributes: difficult
  - Two correlated attributes result in $v_1 \times v_2$ parameters
- Missing values: easy
- Distributions other than the normal distribution can be used:
  - "log-normal" if predetermined minimum is given
  - "log-odds" if bounded from above and below
  - Poisson for attributes that are integer counts
- Cross-validation can be used to estimate k - time consuming !

General EM Method

Define likelihood function $Q(h'|h)$ which calculates $Y = X \cup Z$ using observed X and current parameters h to estimate Z

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

General EM Method

EM Algorithm:

Estimation (E) step: Calculate $Q(h'|h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y.

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

Maximization (M) step: Replace hypothesis h by the hypothesis h' that maximizes this Q function.

$$h \leftarrow \underset{h'}{\operatorname{argmax}}\, Q(h'|h)$$

Hierarchical clustering

- Bottom up: at each step join the two closest clusters (starting with single-instance clusters)
  - Design decision: distance between clusters
    - E.g. two closest instances in clusters vs. distance between means
- Top down: find two clusters and then proceed recursively for the two subsets
  - Can be very fast
- Both methods produce a dendrogram (tree of "clusters")

Hierarchical clustering

```
Algorithm    Hierarchical agglomerative
/* dissimilarity matrix $D(ij)$ is given */

1. Find minimal entry $d_{ij}$ in $D$ and merge clusters $i$ and $j$

2. Update $D$ by deleting column $i$ and row $j$, and adding new row
   and column $i \cup j$

3. Revise entries using $d_{k,i \cup j} = d_{i \cup j,k} = \alpha_i d_{ki} + \alpha_j d_{kj} + \gamma |d_{ki} - d_{kj}|$

4. If there is more than one cluster then go to step 1.
```

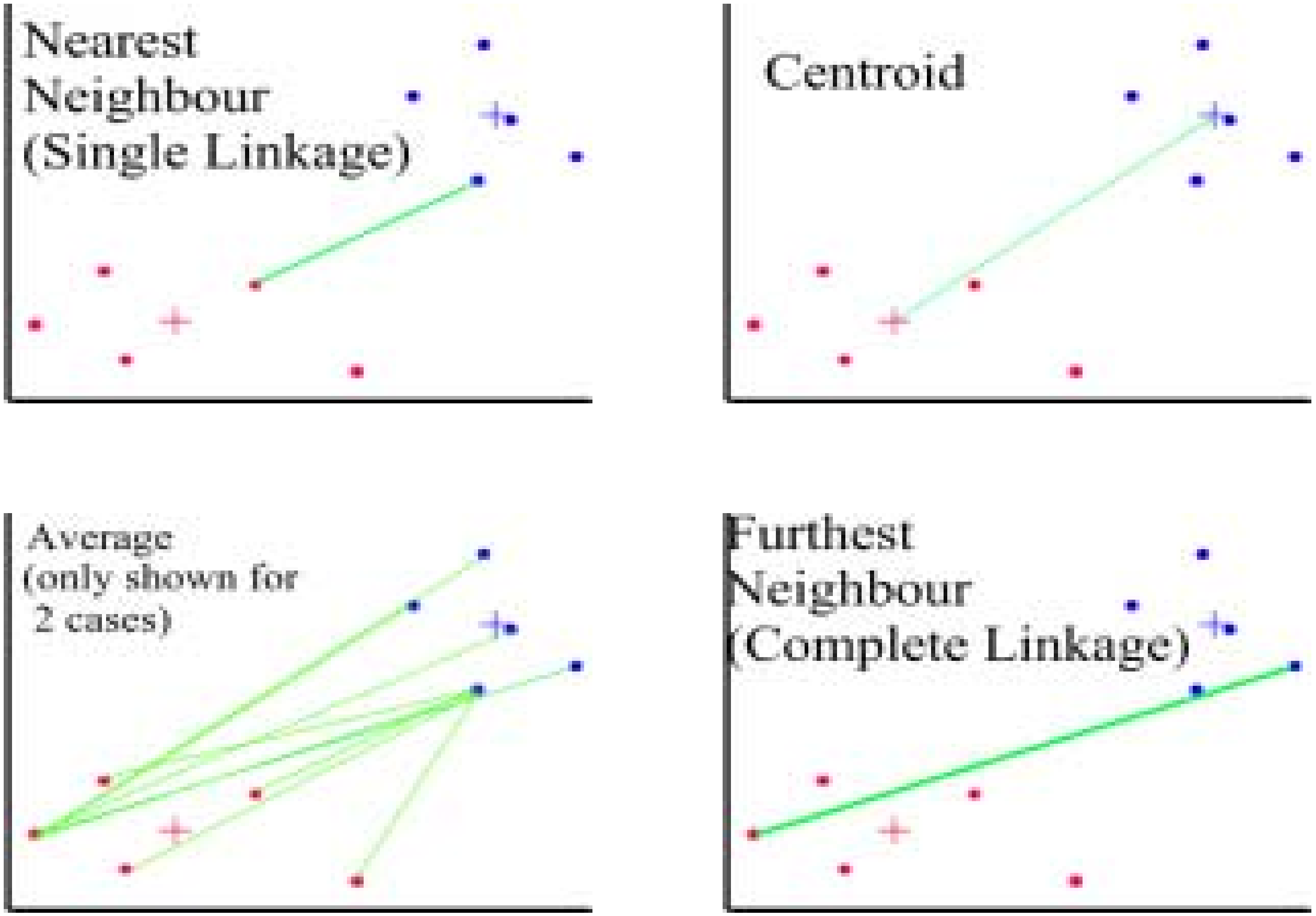
Hierarchical clustering

The algorithm relies on a general updating formula. With different operations and coefficients, many different versions of the algorithm can be used to give variant clusterings.

Single linkage $d_{k,i \cup j} = \min(d_{ki}, d_{kj})$ and $\alpha_i = \alpha_j = \frac{1}{2}$ and $\gamma = -\frac{1}{2}$.
Complete linkage $d_{k,i \cup j} = \max(d_{ki}, d_{kj})$ and $\alpha_i = \alpha_j = \frac{1}{2}$ and $\gamma = \frac{1}{2}$.
Average linkage $d_{k,i \cup j} = \frac{n_i d_{ki}}{n_i + n_j} + \frac{n_j d_{kj}}{n_i + n_j}$ and $\alpha_i = \frac{n_i}{n_i + n_j}, \alpha_j = \frac{n_j}{n_i + n_j}$ and $\gamma = 0$.

Note: dissimilarity computed for every pair of points with one point in the first cluster and the other in the second.

Hierarchical clustering



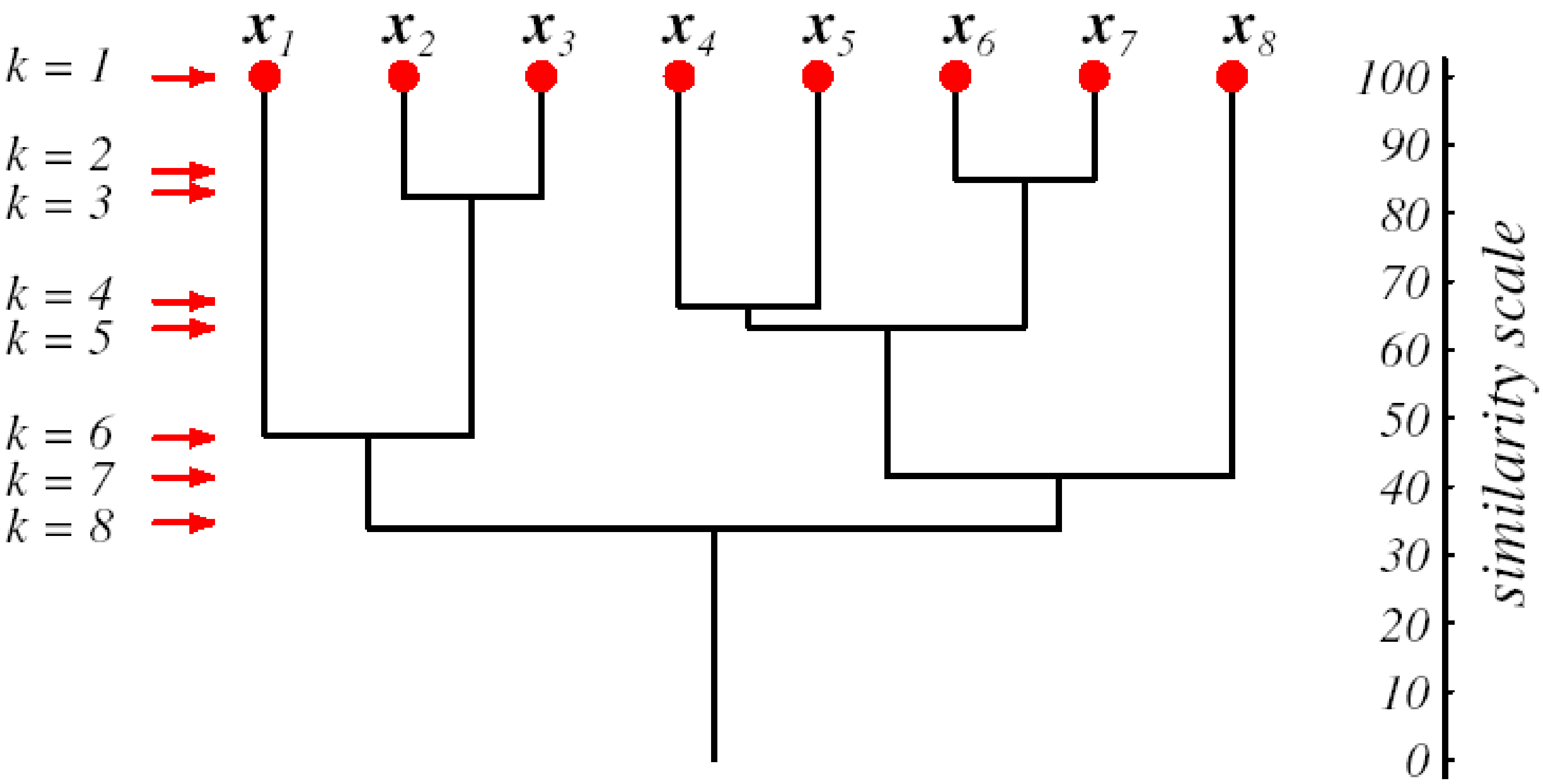
Hierarchical clustering

Represent results of hierarchical clustering with a *dendrogram*

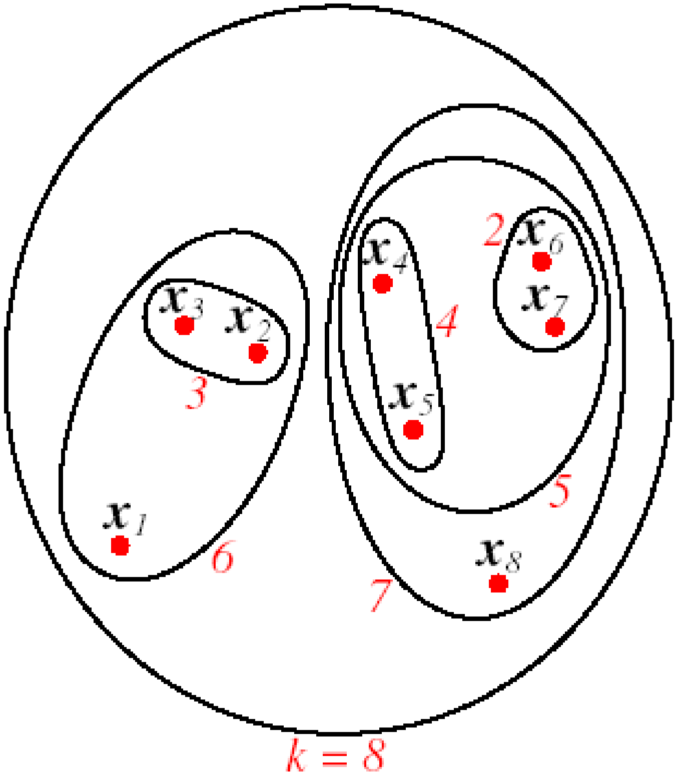
See next diagram

- at level 1 all points in individual clusters
- x_6 and x_7 are most similar and are merged at level 2
- dendrogram drawn to scale to show similarity between grouped clusters

Hierarchical clustering



Hierarchical clustering



Alternative representation of hierarchical clustering based on sets shows hierarchy but not distance

Dendrograms

Two things to beware of:

1. tree structure is not unique for given clustering - for each bottom-up merge the sub-tree to the right or left must be specified - 2^{n-1} ways to permute the n leaves in a dendrogram

2. hierarchical clustering imposes a bias - the clustering forms a dendrogram despite the possible lack of a implicit hierarchical structuring in the data

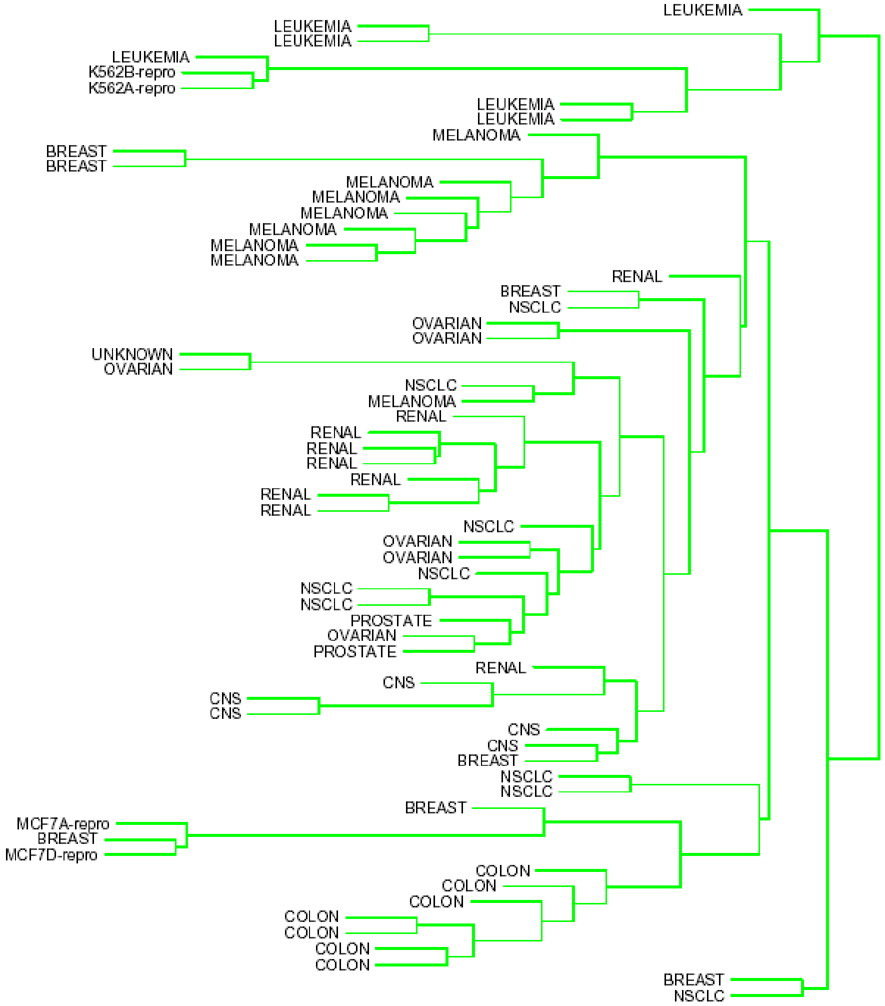
Dendrograms

Next diagram: average-linkage hierarchical clustering of microarray data

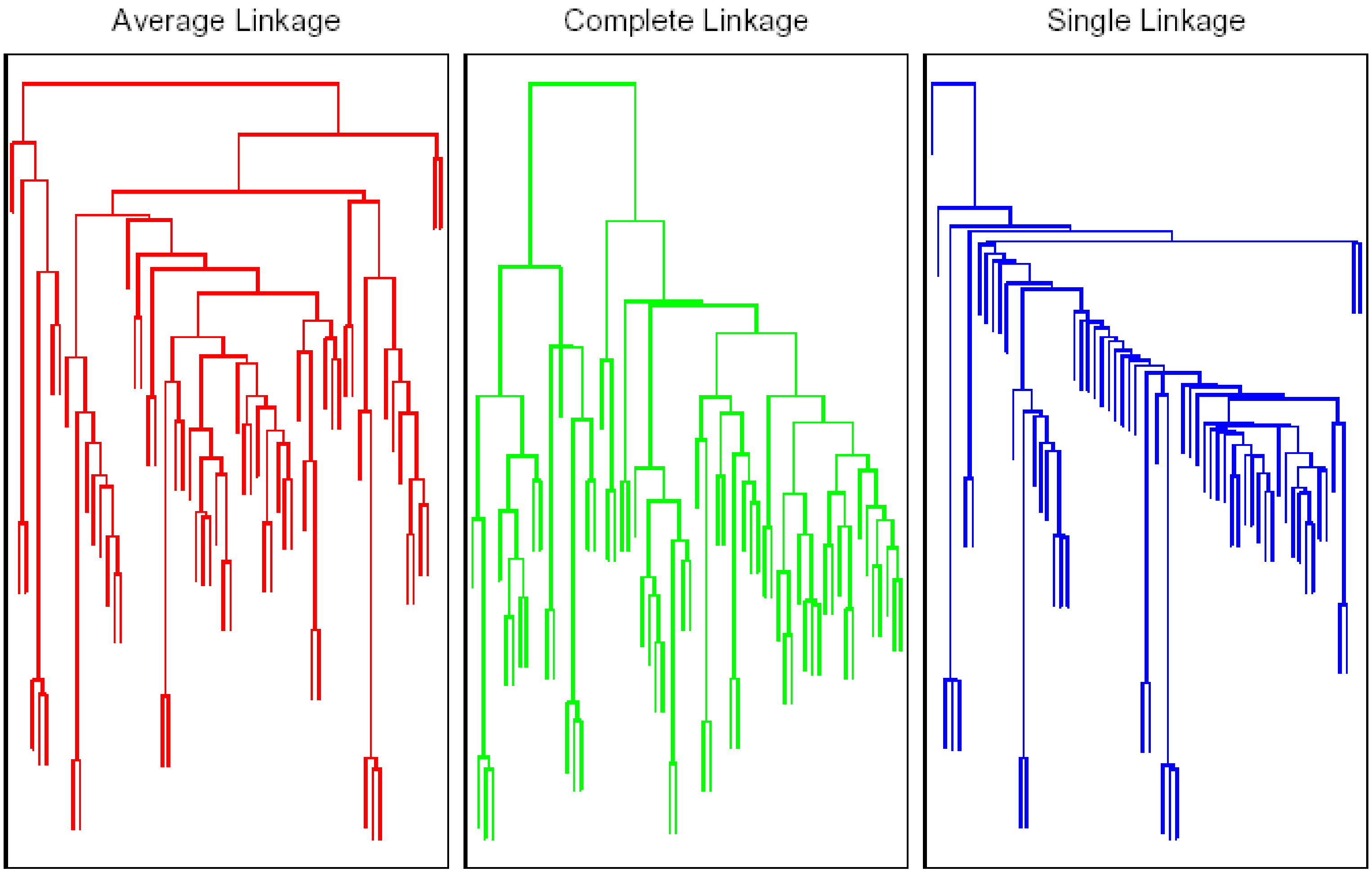
Followed by:

- average-linkage based on average dissimilarity between groups
- complete-linkage based on dissimilarity of furthest pair between groups
- single-linkage based on dissimilarity of closest pair between groups

Dendrograms



Dendrograms

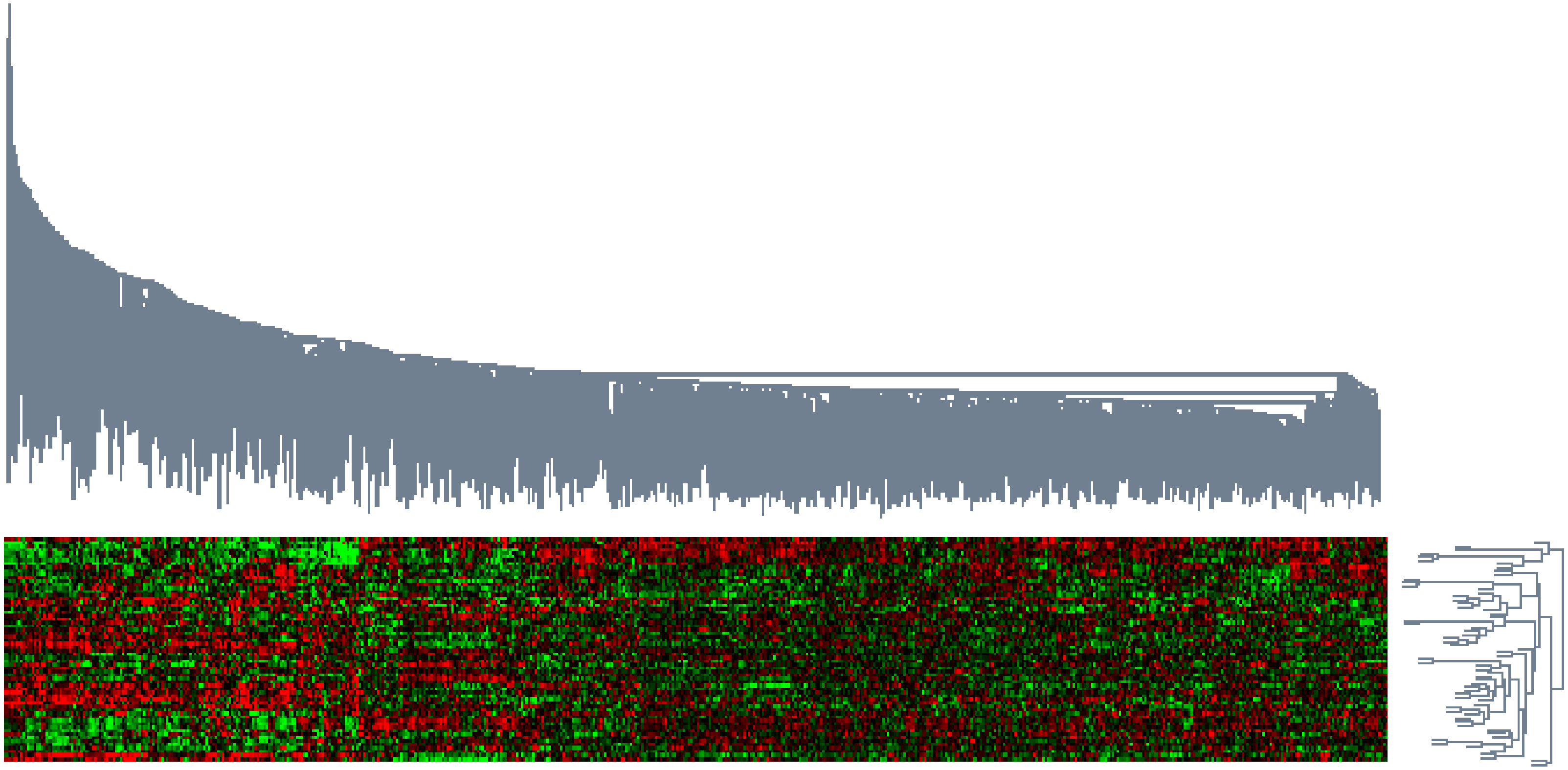


Average Linkage

Complete Linkage

Single Linkage

Dendrograms



Conceptual clustering

- COBWEB/CLASSIT: incrementally forms a hierarchy of clusters (nominal/numerical attributes)
- In the beginning tree consists of empty root node
- Instances are added one by one, and the tree is updated appropriately at each stage
- Updating involves finding the right leaf for an instance (possibly restructuring the tree)
- Updating decisions are based on *category utility*

Category utility

Category utility is a kind of quadratic loss function defined on conditional probabilities:

$$CU(C_1, C_2, \ldots C_k) = \frac{\sum_l \Pr[C_l](\sum_i \sum_j \Pr[a_i = v_{ij} \mid C_l]^2 - \Pr[a_i = v_{ij}]^2)}{k}$$

where

- $C_1, C_2, \ldots C_k$ are the k clusters
- a_i is the ith attribute with values $v_{i1}, v_{i2}, \ldots$
- intuition: knowing class C_l gives a better estimate of values of attributes than not knowing it
- measure amount by which that knowledge helps in the probability estimates

Category utility

Division by k prevents overfitting, because

- If every instance gets put into a different category $\Pr[a_i = v_{ij} \mid C_l] = 1$ for attribute-value in the instance and 0 otherwise
- the numerator becomes ($m =$ total no. of values for set of attributes):

$$m - \sum_i \sum_j \Pr[a_i = v_{ij}]^2$$

- and division by k penalizes large numbers of clusters

Category utility

Category utility can be extended to numerical attributes by assuming normal distribution on attribute values.

- estimate standard deviation of attributes and use in formula
- impose minimum variance threshold as a heuristic

Probability-based clustering

- Problems with above heuristic approach:
  - Division by k?
  - Order of examples?
  - Are restructuring operations sufficient?
  - Is result at least local minimum of category utility?
- From a probabilistic perspective, we want to find the most likely clusters given the data
- Also: instance only has certain probability of belonging to a particular cluster

MDL and clustering

- Description length (DL) needed for encoding the clusters (e.g. cluster centers)
- DL of data given theory: need to encode cluster membership and position relative to cluster (e.g. distance to cluster center)
- Works if coding scheme needs less code space for small numbers than for large ones
- With nominal attributes, we need to communicate probability distributions for each cluster

Bayesian clustering

- Problem: overfitting possible if number of parameters gets large
- Bayesian approach: every parameter has a prior probability distribution
  – Gets incorporated into the overall likelihood figure and thereby penalizes introduction of parameters
- Example: Laplace estimator for nominal attributes
- Can also have prior on number of clusters!
- Actual implementation: NASA's AUTOCLASS
  – P. Cheeseman - recently with NICTA

Semi-supervised Learning

Problem: obtaining labelled examples may be difficult, expensive

However, may have many unlabelled instances (e.g., documents)

Semi-supervised Learning

1. Learn initial classifier using labelled set
2. Apply classifier to unlabelled set
3. Learn new classifier from now-labelled data
4. Repeat until convergence

Self-training algorithm

Given: labelled data $\langle x, y \rangle$ and unlabelled data $\langle x \rangle$

Repeat:

Train classifier h from labelled data using supervised learning

Label unlabelled data using classifier h

Assumes: classifications by h will tend to be correct (especially high probability ones)

Example: use Naive Bayes algorithm

Apply self-training algorithm using Naive Bayes

A form of EM training . . .

Co-training

Blum & Mitchell (1998)

Key idea: two views of an instance, f_1 and f_2

- assume f_1 and f_2 independent and compatible
- if we have a good attribute set, leverage similarity between attribute values in each view, assuming they predict the class, to classify the unlabelled data

Co-training

Multi-view learning

Given two (or more) perspectives on data, e.g., different attribute sets

Train separate models for each perspective on small set of labelled data

Use models to label a subset of the unlabelled data

Repeat until no more unlabelled examples

Clustering summary

- many techniques available – may not be single “magic bullet” rather different techniques useful for different aspects of data

- hierarchical clustering gives a view of the complete structure found, without restricting the no. of clusters, but can be computationally expensive

- different linkage methods can produce very different dendrograms

- higher nodes can be very heterogeneous

- problem may not have a “real” hierarchical structure

Clustering summary

- k-means and SOM avoid some of these problems, but also have drawbacks
- cannot extract "intermediate features" e.g. a subset of features in which a subset of ojects is co-expressed
- for all of these methods, can cluster objects or features, but not both together (coupled two-way clustering)
- should all the points be clustered ? modify algorithms to allow points to be discarded
- visualization is important: dendrograms and SOMs are good but further improvements would help

Clustering summary

- how can the quality of clustering be estimated ?
  - if clusters known, measure proportion of disagreements to agreements
  - if unknown, measure homogeneity (average similarity between feature vectors in a cluster and the centroid) and separation (weighted average similarity between cluster centroids) with aim of increasing homogeneity and decreasing separation
  - sihouette method, etc.

- clustering is only the first step - mainly exploratory; classification, modelling, hypothesis formation, etc.