Clustering

- *Unsupervised*: no target value to be predicted
- Differences between models/algorithms:
  - Exclusive vs. overlapping
  - Deterministic vs. probabilistic
  - Hierarchical vs. flat
  - Incremental vs. batch learning
- Evaluation problematic: usually done by inspection
- But: if clustering is treated as a density estimation problem, then it can be evaluated on test data!

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

The *k*-means algorithm

- Clusters the data into *k* groups where *k* is predefined
- 1st step: cluster centers are chosen (e.g. at random)
- 2nd step: instances are assigned to clusters based on their distance to the cluster centers
- 3rd step: *centroids* of clusters are computed
- 4th step: go to 1st step until convergence

Discussion

- 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 two-dimensional 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

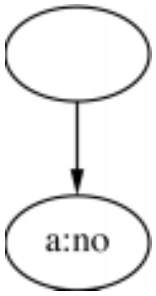
Incremental clustering

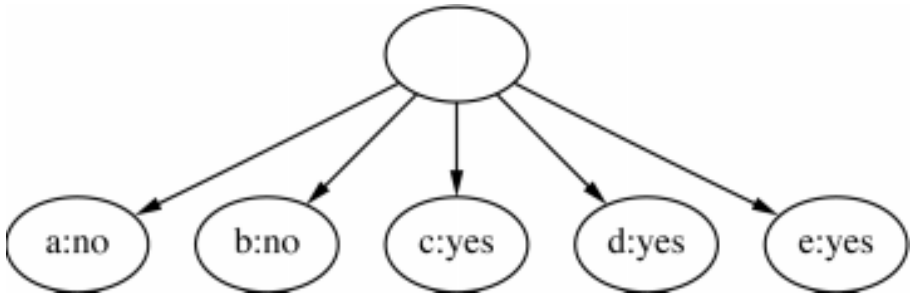
- COBWEB/CLASSIT: incrementally forms a hierarchy of clusters
- In the beginning tree consists of empty root node
- Instances are added one by one, and the tree is updated appropriately at each stage
- Updating involves finding the right leaf for an instance (possibly restructuring the tree)
- Updating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

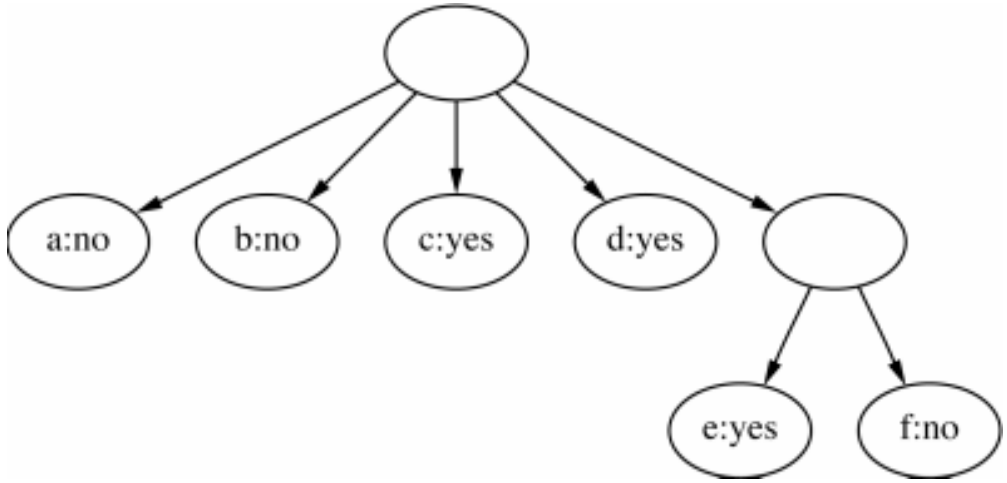
Clustering the weather data

ID code	Outlook	Temp.	Humidity	Windy
A	Sunny	Hot	High	False
B	Sunny	Hot	High	True
C	Overcast	Hot	High	False
D	Rainy	Mild	High	False
E	Rainy	Cool	Normal	False
F	Rainy	Cool	Normal	True
G	Overcast	Cool	Normal	True
H	Sunny	Mild	High	False
I	Sunny	Cool	Normal	False
J	Rainy	Mild	Normal	False
K	Sunny	Mild	Normal	True
L	Overcast	Mild	High	True
M	Overcast	Hot	Normal	False
N	Rainy	Mild	High	True

Steps 1-3

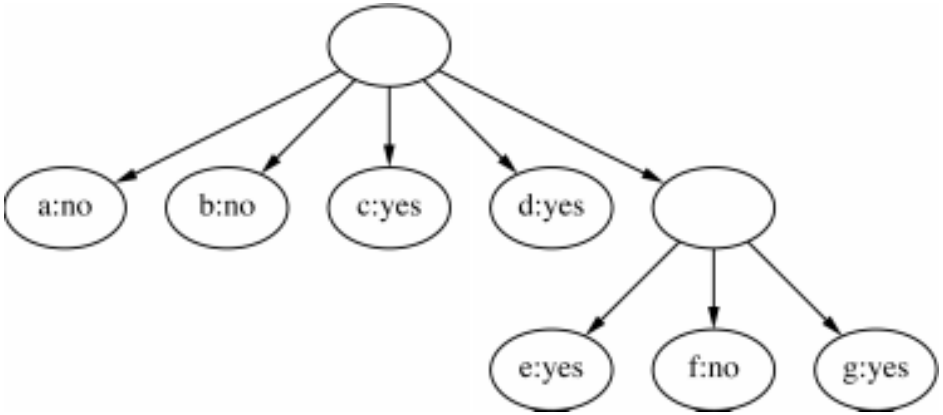


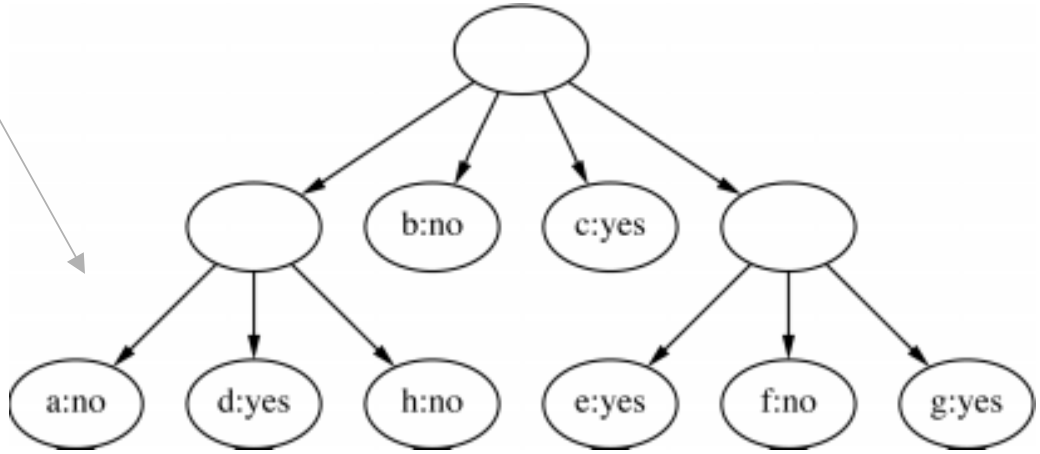




Steps 3-4

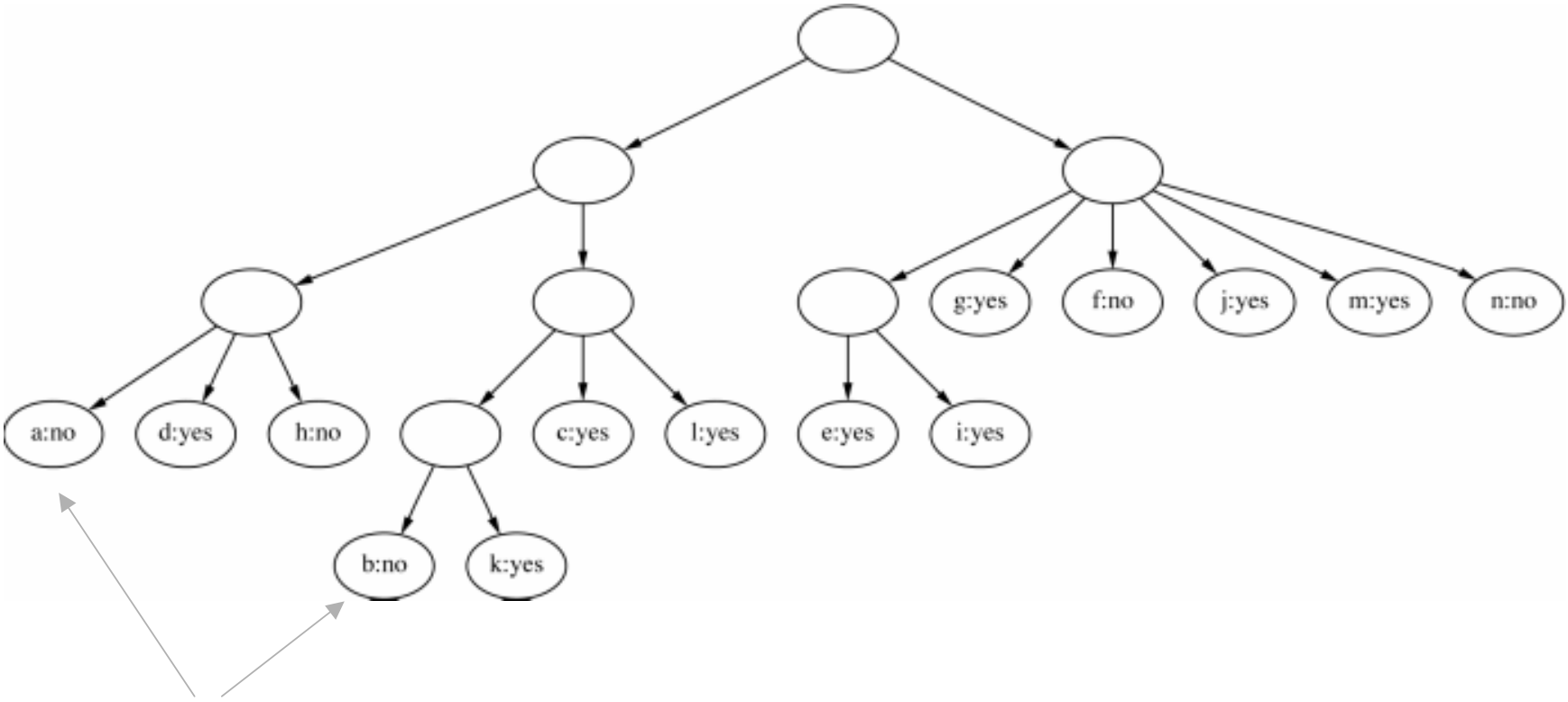
Best host and runner-up have been *merged*





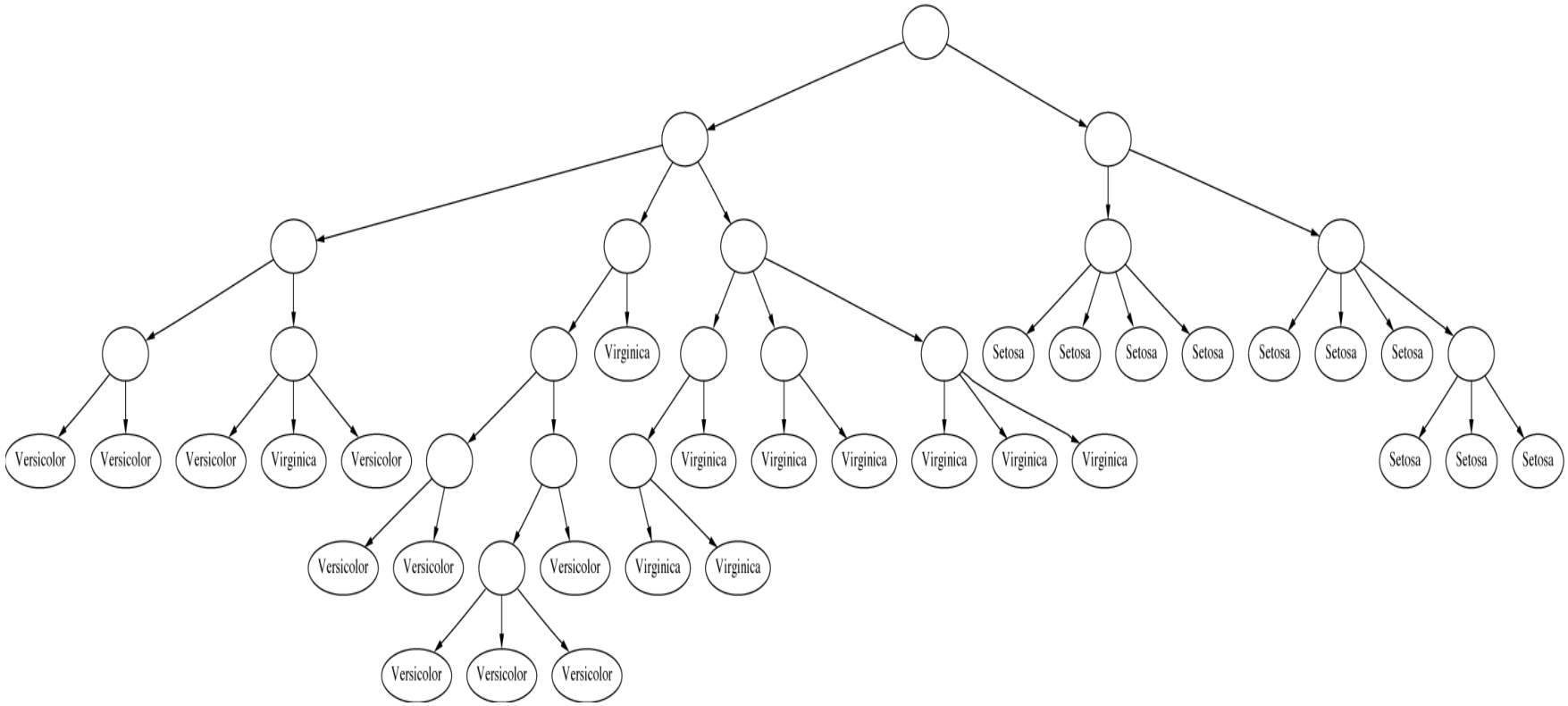
Note: *splitting* the best host is considered if merging doesn't help

The final hierarchy

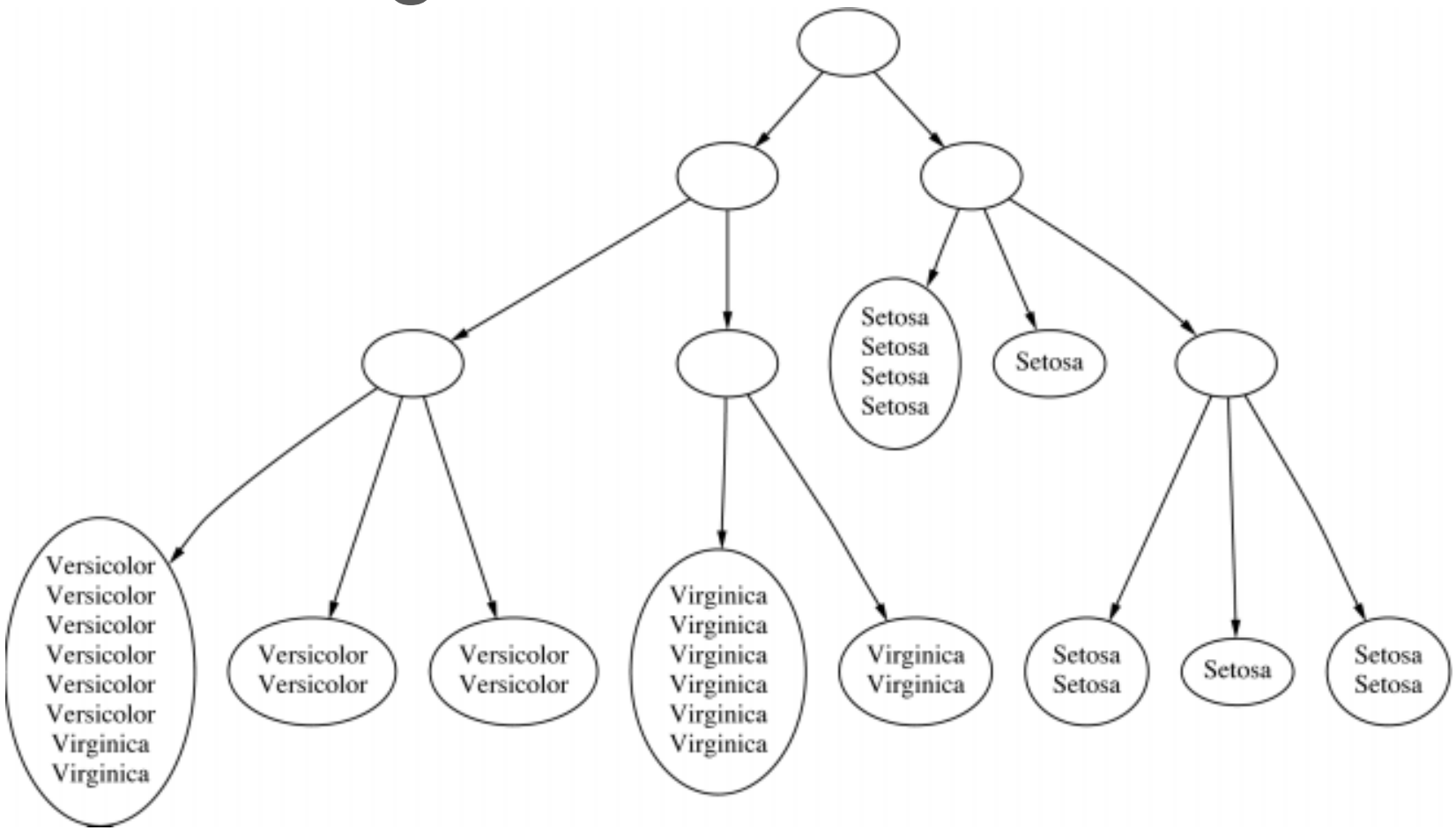


a and b are actually very similar

Clustering (parts) of the iris data



Clustering the iris data with cutoff



Category utility

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

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

- If every instance gets put into a different category the numerator becomes (m = #attributes):

$$m - \Pr[a_i = v_{ij}]^2 \quad \longleftarrow \quad \textit{maximum}$$

Numeric attributes

- We assume normal distribution:

$$f(a) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(a-\mu)^2}{2\sigma^2}}$$

- Then we get:

$$\sum_j \Pr[a_i = v_{ij}]^2 \Leftrightarrow \int f(a_i)^2 da_i = \frac{1}{2\sqrt{\pi}\sigma_i}$$

- Thus

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

is

$$CU = \frac{\sum_l \Pr[C_l] \frac{1}{2\sqrt{\pi}} \sum_i \left(\frac{1}{\sigma_{il}} - \frac{1}{\sigma_i} \right)}{k}$$

- *Acuity* parameter: prespecified minimum variance

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

Finite mixtures

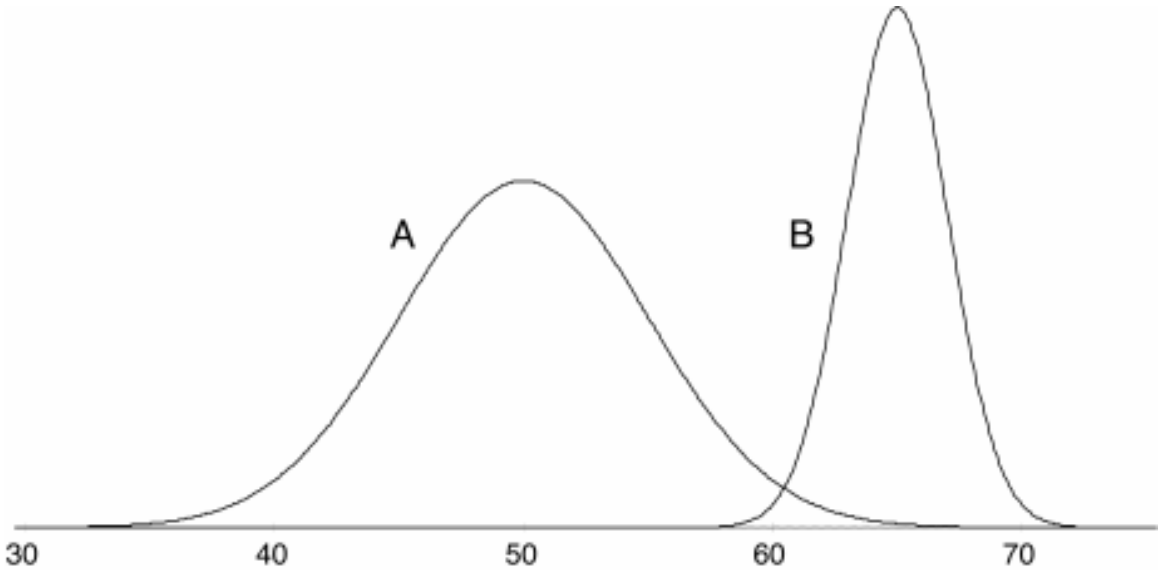
- Probabilistic clustering algorithms model the data using a *mixture* of distributions
- Each cluster is represented by one distribution
  - The distribution governs the probabilities of attributes values in the corresponding cluster
- They are called *finite mixtures* because there is only a finite number of clusters being represented
- Usually individual distributions are normal distribut.
- Distributions are combined using cluster weights

A two-class mixture model

data

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

Using the mixture model

- The probability of an instance *x* belonging to cluster *A* is:

$$\Pr[A \mid x] = \frac{\Pr[x \mid A]\Pr[A]}{\Pr[x]} = \frac{f(x; \mu_A, \sigma_A) p_A}{\Pr[x]}$$

with $$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

- The *likelihood* of an instance given the clusters is:

$$\Pr[x \mid \text{the distributions}] = \sum_i \Pr[x \mid \text{cluster}_i]\Pr[\text{cluster}_i]$$

Learning the clusters

- Assume we know that there are *k* clusters
- To learn the clusters we need to determine their parameters
  - I.e. their means and standard deviations
- We actually have a performance criterion: the likelihood of the training data given the clusters
- Fortunately, there exists an algorithm that finds a local maximum of the likelihood

The EM algorithm

- *EM algorithm*: expectation-maximization algorithm
  - Generalization of *k*-means to probabilistic setting
- Similar iterative procedure:
  1. Calculate cluster probability for each instance (expectation step)
  2. Estimate distribution parameters based on the cluster probabilities (maximization step)
- Cluster probabilities are stored as instance weights

More on EM

- Estimating parameters from weighted instances:

$$\mu_A = \frac{w_1 x_1 + w_2 x_2 + \ldots + w_n x_n}{w_1 + w_2 + \ldots + w_n}$$

$$\sigma_A{}^2 = \frac{w_1 (x_1 - \mu)^2 + w_2 (x_2 - \mu)^2 + \ldots + w_n (x_n - \mu)^2}{w_1 + w_2 + \ldots + w_n}$$

- Procedure stops when log-likelihood saturates
- Log-likelihood:

$$\sum_i \log(p_A \Pr[x_i \mid A] + p_B \Pr[x_i \mid B])$$

Extending the mixture model

- Using more then two distributions: easy
- Several attributes: easy if independence is 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
- Nominal attributes: easy if independent

More on extensions

- Correlated nominal attributes: difficult
  - Two correlated attributes result in v_1 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*!!

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

Discussion

- Clusters can be interpreted by using supervised learning in a post-processing step
- Can be used to fill in missing values
- May be advantageous to make attributes more independent in pre-processing step
  - I.e. using *principal component analysis*
- Big advantage of probabilistic clustering schemes:
  - Likelihood of data can be estimated and used to compare different clustering models