Data Mining

Practical Machine Learning Tools and Techniques

Slides for Chapter 9, Probabilistic methods

of *Data Mining* by I. H. Witten, E. Frank,
M. A. Hall, and C. J. Pal

Random variables

- In probabilistic approaches to machine learning it is common to think of data as observations arising from an underlying probability model for *random variables*
- Given a discrete random variable A, $P(A)$ is a function that encodes the probabilities for each of the categories, classes or states that A may be in
- For a continuous random variable x, $p(x)$ is a function that assigns a probability density to all possible values of x
- In contrast, $P(A=a)$ is the single probability of observing the specific event $A=a$

Notation

- The $P(A=a)$ notation is often simplified to simply $P(a)$, but one must remember if a was defined as a random variable or as an observation
- Similarly for the observation that continuous random variable x has the value x_1 it is common to write this as $p(x_1)=\mathrm{p}(x=x_1)$, a simplification of the longer but clearer notation

The product rule

- The *product rule*, sometimes referred to as the “fundamental rule of probability,” states that the joint probability of random variables *A* and *B* can be written

$$P(A,B) = P(A \mid B)P(B)$$

- The product rule also applies when *A* and *B* are groups or subsets of events or random variables.

The sum rule

- The *sum rule* states that given the joint probability of variables X_1, X_2, ..., X_N, the *marginal probability* for a given variable can be obtained by summing (or integrating) over all the other variables.
- For example, to obtain the marginal probability of X_1, sum over all the states of all the other variables:

$$P(X_1) = \sum_{x_2} \dots \sum_{x_N} P(X_1, X_2 = x_2, \dots, X_N = x_N)$$

Marginalization

- The previous notation can be simplified to

$$p(x_1) = \sum_{x_2} \dots \sum_{x_N} P(x_1, x_2, \dots, x_N)$$

- The sum rule generalizes to continuous random variables, ex. for *x₁,x₂,…,x_N* we have

$$p(x_1) = \dots \int_{x_N} p(x_1, x_2, \dots, x_N) dx_1 \dots dx_N$$

- These procedures are known as *marginalization*
- They give us *marginal distributions* of the variables not included in the sums or integrals

Bayes' Rule

- Can be obtained by swapping *A* and *B* in the product rule and observing $P(B|A)P(A)=P(A|B)P(B)$ and therefore

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}$$

- Suppose we have models for $P(A|B)$ and $P(B)$
  - We observe that $A=a$, and
  - we want to compute $P(B|A=a)$
  - $P(A=a|B)$ is referred to as the *likelihood*
  - $P(B)$ is the *prior* distribution of *B*
  - $P(B|A=a)$ is posterior distribution, obtained from:

$$P(A=a) = \sum_b P(A=a, B=b) = \sum_b P(A=a \mid B=b)P(B=b)$$

Maximum Likelihood

- Our goal is to estimate a set of parameters θ of a probabilistic model, given a set of *observations* $x_1, x_2, \ldots, x_n$.
- Maximum likelihood techniques assume that:
  1) the examples have no dependence on one another, the occurrence of one has no effect on the others, and
  2) each can be modeled in exactly the same way.
- These assumptions are often summarized by saying that events are *independent and identically distributed* (i.i.d.).

Maximum Likelihood

- The i.i.d. assumption corresponds to the use of a joint probability density function for all observations consisting of the product of the same probability model $p(x_i;\theta)$ applied to each observation independently.
- For *n* observations, this could be written as

$$p(x_1, x_2, \ldots, x_n; \theta) = p(x_1;\theta)\, p(x_2;\theta) \ldots p(x_n;\theta)$$

where each function $p(x_i;\theta)$ has the same θ

Maximum Likelihood

- The likelihood of our data can be written

$$\mathrm{L}(\theta; x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i; \theta)$$

- The data is fixed, but we can adjust θ so as to *maximize the likelihood* or *log-likelihood*

$$\theta_{ML} = \arg\max_{\theta} \sum_{i=1}^{n} \log p(x_i; \theta)$$

- We use the the log-likelihood as it is more numerically stable

Maximum a posteriori (MAP) parameter estimation

- If we treat our parameters as random variables we can compute the posterior

$$p(\theta | x_1, x_2, \ldots, x_n) = \frac{p(x_1, x_2, \ldots, x_n | \theta) p(\theta)}{p(x_1, x \ldots, x_n)}$$

- We have used | or the “given” notation in place of ; to emphasize that θ is random, but
- Conditioned on a point estimate for the posterior we have a conditionally i.i.d. model
- MAP parameter estimation seeks

$$\theta_{MAP} = \arg\max_{\theta} \left[\sum_{i=1}^{n} \log p(x_i; \theta) + p(\theta) \right]$$

The chain rule of probability

- Results from applying the product rule recursively between a single variable and the rest of the variables
- The *chain rule* states that the joint probability of *n* attributes $A_{i=1\ldots m}$ can be decomposed into the following product:

$$P(A_1, A_2, \ldots, A_n) = P(A_1) \prod_{i=1}^{n-1} P(A_{i+1} | A_i, A_{i-1}, .., A_1)$$

e.g.

$$P(A_1, A_2, A_3, A_4) = P(A_1) P(A_2 | A_1) P(A_3 | A_2, A_1) P(A_4 | A_3, A_2, A_1)$$

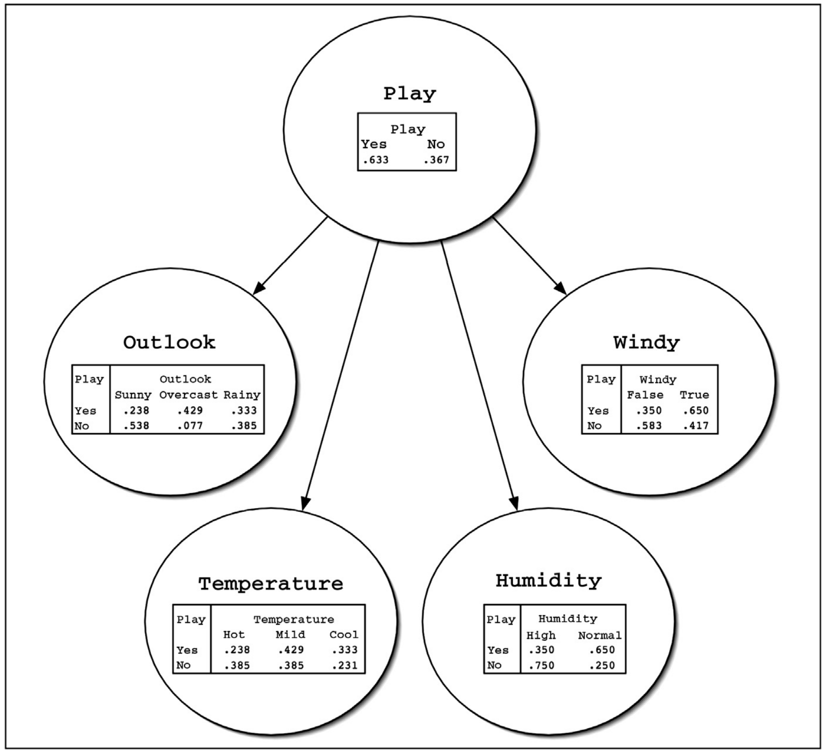
Bayesian networks

- The chain rule holds for any order for the A_is
- A Bayesian network is an acyclic graph,
- Therefore its nodes can be given an ordering where ancestors of node A_i have indices < i
- Thus a Bayesian network can be written

$$P(A_1, A_2, \ldots, A_n) = \prod_{i=1}^{n} P(A_i \mid \mathrm{Parents}(A_i))$$

- When a variable has no parents, we use the unconditional probability of that variable

Bayesian network #1 for the weather data



Play	
Yes	No
.633	.367

Play	Outlook		
	Sunny	Overcast	Rainy
Yes	.238	.429	.333
No	.538	.077	.385

Play	Windy	
	False	True
Yes	.350	.650
No	.583	.417

Play	Temperature		
	Hot	Mild	Cool
Yes	.238	.429	.333
No	.385	.385	.231

Play	Humidity	
	High	Normal
Yes	.350	.650
No	.750	.250

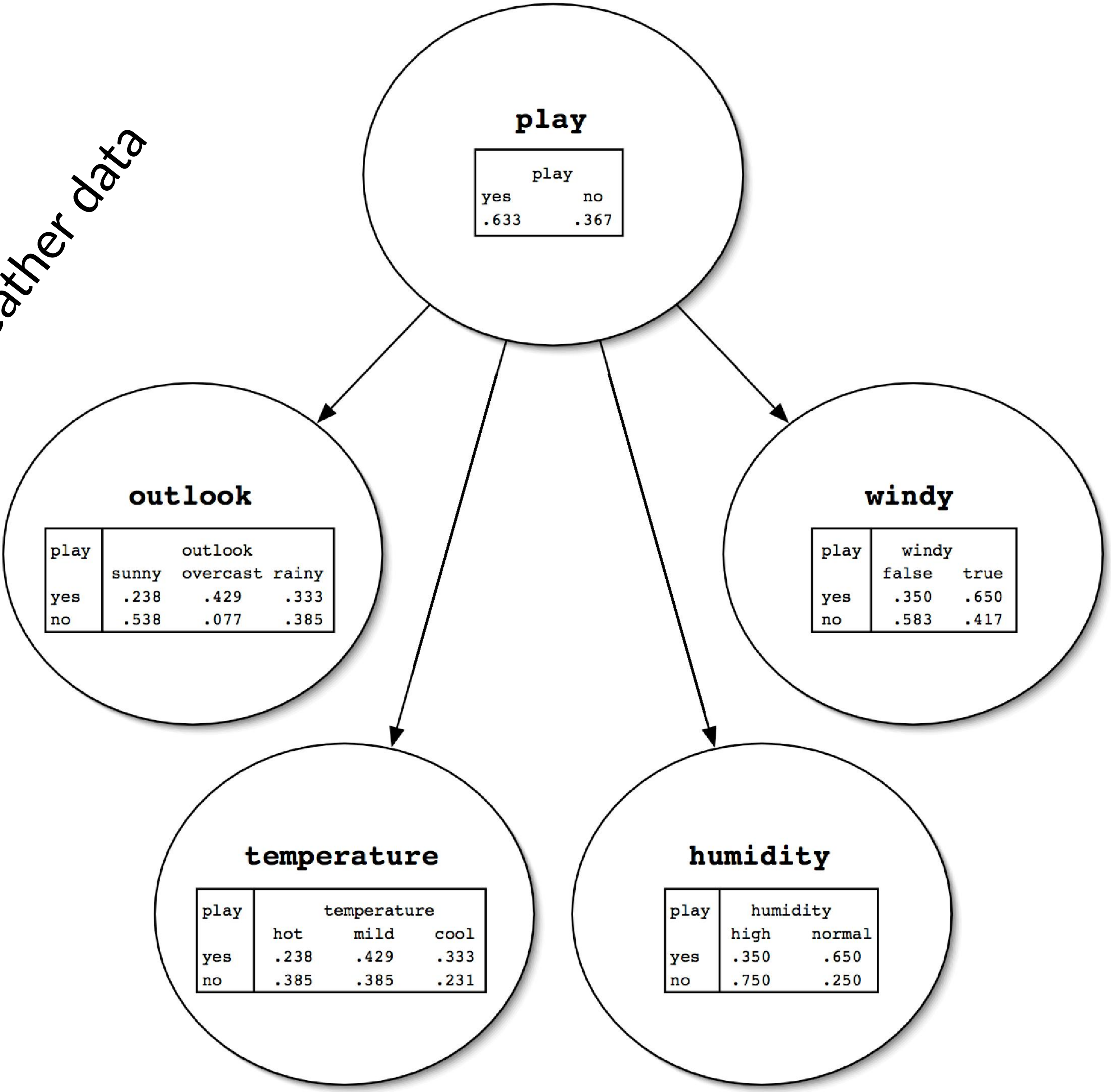
Random Variables
Y: Play
O: Outlook
T: Temperature
H: Humidity
W: Windy

$$\begin{tikzcd} & & Y \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\ W & O & T & H \end{tikzcd}$$

The graphs express the factorization below:

$$P(Y, O, T, H, W) = P(W \mid Y) P(O \mid Y) P(T \mid Y) P(H \mid Y) P(Y)$$

Network #1 for the weather data



play	
yes	no
.633	.367

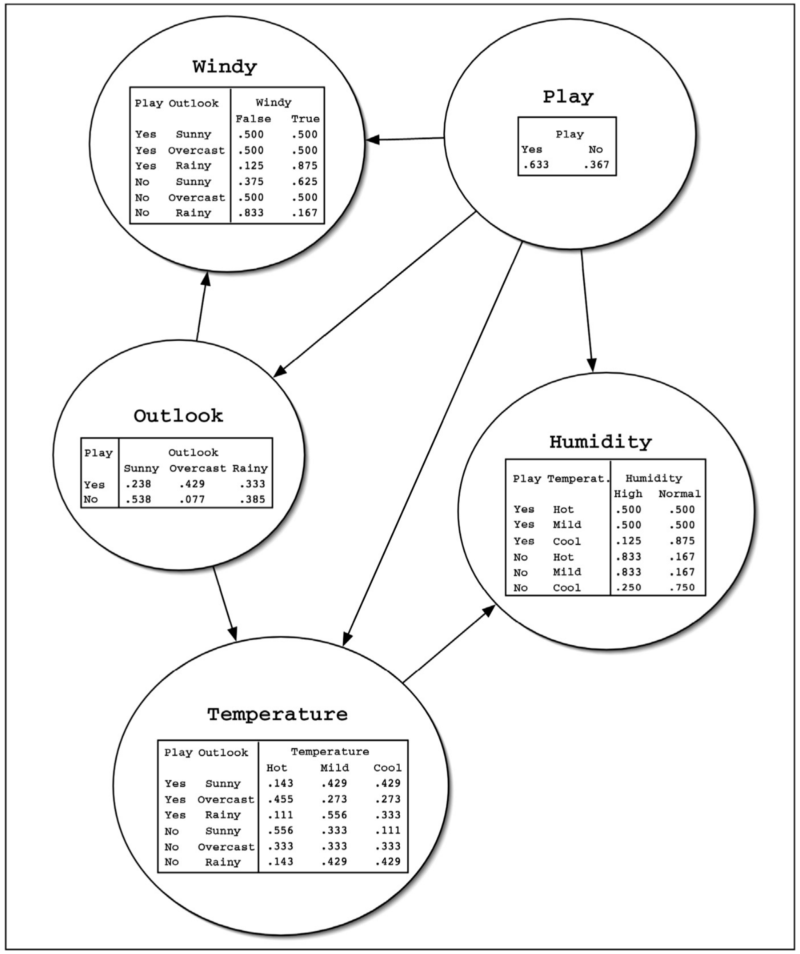
play	outlook		
	sunny	overcast	rainy
yes	.238	.429	.333
no	.538	.077	.385

play	windy	
	false	true
yes	.350	.650
no	.583	.417

play	temperature		
	hot	mild	cool
yes	.238	.429	.333
no	.385	.385	.231

play	humidity	
	high	normal
yes	.350	.650
no	.750	.250

Bayesian network #2 for the weather data



Windy

Play	Outlook	Windy False	Windy True
Yes	Sunny	.500	.500
Yes	Overcast	.500	.500
Yes	Rainy	.125	.875
No	Sunny	.375	.625
No	Overcast	.500	.500
No	Rainy	.833	.167

Play

Play Yes	Play No
.633	.367

Outlook

Play	Outlook Sunny	Outlook Overcast	Outlook Rainy
Yes	.238	.429	.333
No	.538	.077	.385

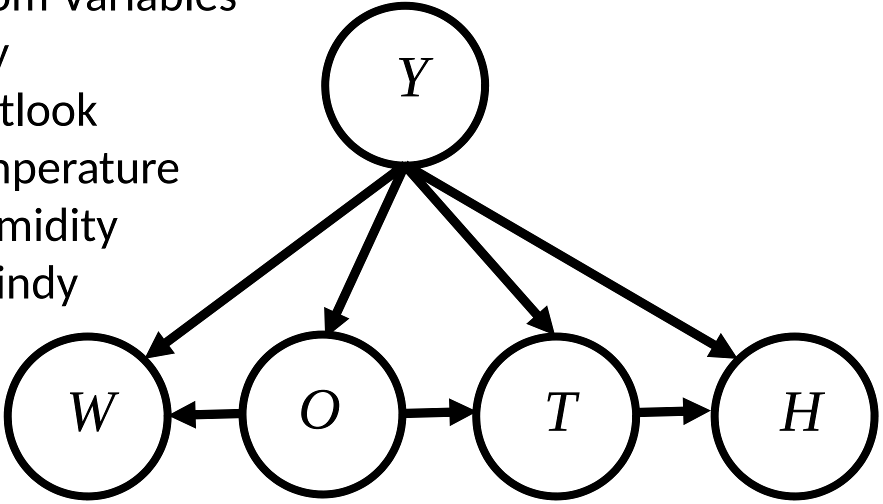
Humidity

Play	Temperat.	Humidity High	Humidity Normal
Yes	Hot	.500	.500
Yes	Mild	.500	.500
Yes	Cool	.125	.875
No	Hot	.833	.167
No	Mild	.833	.167
No	Cool	.250	.750

Temperature

Play	Outlook	Temperature Hot	Temperature Mild	Temperature Cool
Yes	Sunny	.143	.429	.429
Yes	Overcast	.455	.273	.273
Yes	Rainy	.111	.556	.333
No	Sunny	.556	.333	.111
No	Overcast	.333	.333	.333
No	Rainy	.143	.429	.429

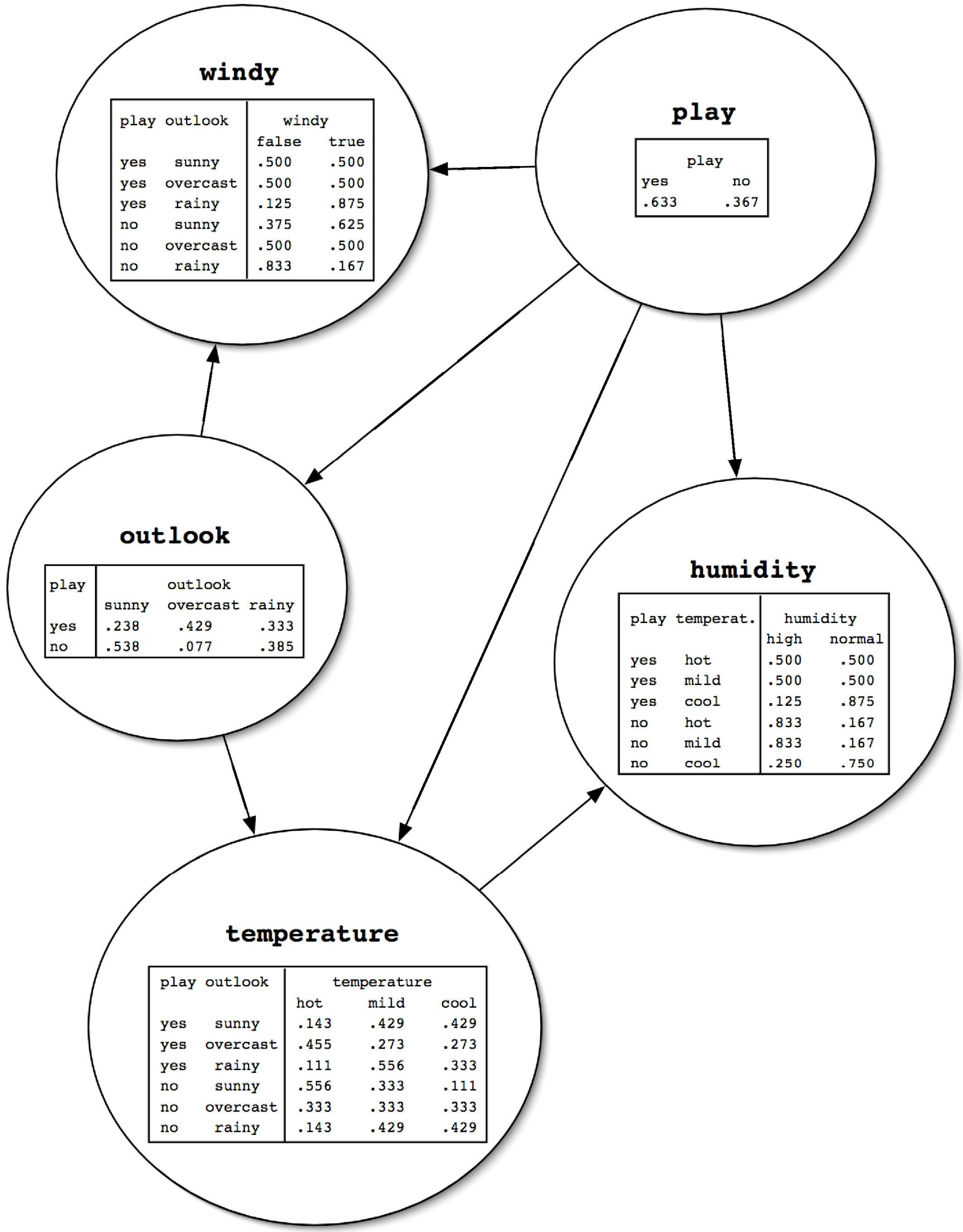
Random Variables
Y: Play
O: Outlook
T: Temperature
H: Humidity
W: Windy



The graphs express the factorization below:

$$P(Y,O,T,H,W) = P(W \mid O,Y)P(O \mid Y)P(T \mid O,Y)P(H \mid T,Y)P(Y)$$

Network #2 for the weather data



play

yes	no
.633	.367

windy

play	outlook	windy: false	windy: true
yes	sunny	.500	.500
yes	overcast	.500	.500
yes	rainy	.125	.875
no	sunny	.375	.625
no	overcast	.500	.500
no	rainy	.833	.167

outlook

play	outlook: sunny	outlook: overcast	outlook: rainy
yes	.238	.429	.333
no	.538	.077	.385

humidity

play	temperat.	humidity: high	humidity: normal
yes	hot	.500	.500
yes	mild	.500	.500
yes	cool	.125	.875
no	hot	.833	.167
no	mild	.833	.167
no	cool	.250	.750

temperature

play	outlook	temperature: hot	temperature: mild	temperature: cool
yes	sunny	.143	.429	.429
yes	overcast	.455	.273	.273
yes	rainy	.111	.556	.333
no	sunny	.556	.333	.111
no	overcast	.333	.333	.333
no	rainy	.143	.429	.429

Computing the class probabilities

- Two steps: computing a product of probabilities for each class and normalization
  1) For each class value
     - Take all attribute values and class value
     - Look up corresponding entries in conditional probability distribution tables
     - Take the product of all probabilities
  2) Divide the product for each class by the sum of the products (normalization)

Naive Bayes vs. Bayesian Network

Naive Bayes:

P(windy=true,outlook=sunny, temperature=cool, humidity=high|
play=yes)=

P(windy=true|play=yes)*P(outlook=sunny|play=yes) *
P(temperature=cool|play=yes) * P(humidity=high|play=yes)

Bayesian network:

P(windy=true,outlook=sunny, temperature=cool, humidity=high|
play=yes)=

P(windy=true|outlook=sunny,play=yes) *
P(outlook=sunny|play=yes) *
P(temperature=cool|outlook=sunny,play=yes) *
P(humidity=high|temperature=cool,play=yes)

Why can we do this? (Part I)

- Single assumption: values of a node's parents completely determine probability distribution for current node

$$Pr[\text{node}|\text{ancestors}] = Pr[\text{node}|\text{parents}]$$

- Means that node/attribute is conditionally independent of other ancestors given parents

Why can we do this? (Part II)

- Chain rule from probability theory:

$$Pr[a_{1,}a_{2,}\ldots,a_n]=\prod_{i=1}^{n}Pr[a_i|a_{i-1},\ldots,a_1]$$

- Because of our assumption from the previous slide:

$$Pr[a_1,a_2,\ldots,a_n]=\prod_{i=1}^{n}Pr[a_i|a_{i-1},\ldots,a_1]=$$
$$\prod_{i=1}^{n}Pr[a_i|a_i's\ \ parents]$$

Chain Rule vs. Bayesian Network

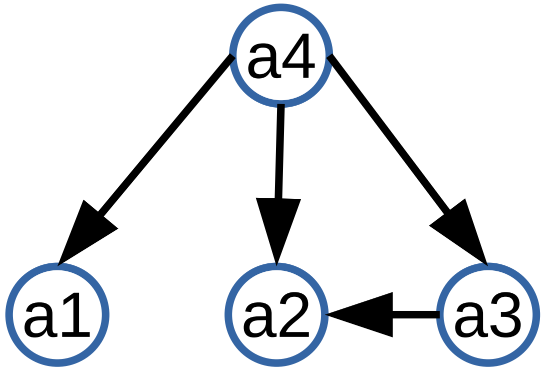
- **Chain rule:**

P(a1,a2,a3,a4)=P(a1|a2,a3,a4) * P(a2,a3,a4)
P(a2,a3,a4)= P(a2|a3,a4) * P(a3,a4)
P(a3,a4)=P(a3|a4)*P(a4)

P(a1,a2,a3,a4)=P(a1|a2,a3,a4) * P(a2|a3,a4) *
P(a3|a4) * P(a4)

- **Bayesian network**

P(a1,a2,a3,a4)=
P(a1|a4)*P(a2|a3,a4)*P(a3|a4)*P(a4)



Learning Bayes nets

- Basic components of algorithms for learning Bayes nets:

  1) Method for evaluating the goodness of a given network
     - Measure based on probability of training data given the network (or the logarithm thereof)

  2) Method for searching through space of possible networks
     - Amounts to searching through sets of edges because nodes are fixed

Estimating Bayesian network parameters

- The log-likelihood of a Bayesian network with *V* variables and *N* examples of complete variable assignments to the network is

$$\sum_{i=1}^{N} \log P(\{\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_V\}_i) = \sum_{i-1}^{N} \sum_{v=1}^{V} \log P(\tilde{A}_{v,i} \,\big|\, \text{Parents}(\tilde{A}_{v,i}); \Theta_v)$$

  where the parameters of each conditional or unconditional distribution are given by Θ_v

- We use the $\tilde{A}_{v,i}$ notation to indicate the *i*th observation of variable *v*

Estimating probabilities in Bayesian networks

- The estimation problem *decouples* into separate estimation problems for each conditional or unconditional probability
- Unconditional probabilities can be written as

$$P(A = a) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(\tilde{A}_i = a)$$

where $\mathbf{1}(\tilde{A}_i = a)$ is an indicator function returning 1 when the ith observed value for A_i=a and 0 otherwise

Estimating conditional distributions

- Estimating conditional distributions in Bayesian networks is equally easy and amounts to simply counting configurations and dividing, ex.

$$P(B=b \mid A=a) = \frac{P(B=b, A=a)}{P(A=a)} = \frac{\sum_{i=1}^{N} \mathbf{1}(\tilde{A}_i = a, \tilde{B}_i = b)}{\sum_{i=1}^{N} \mathbf{1}(\tilde{A}_i = a)}.$$

- Zero counts cause problems and this motivates the use of Bayesian priors

Estimating network structure

- One possibility is to use cross-validation to estimate the goodness of fit on held out data (so as to avoid over fitting) another is to penalize model complexity
- Let *K* be the number of parameters, *LL* the log-likelihood, and *N* the number of instances in the data.
- Two popular measures for evaluating the quality of a network are the *Akaike Information Criterion* (AIC):

$$\text{AIC score} = -LL + K$$

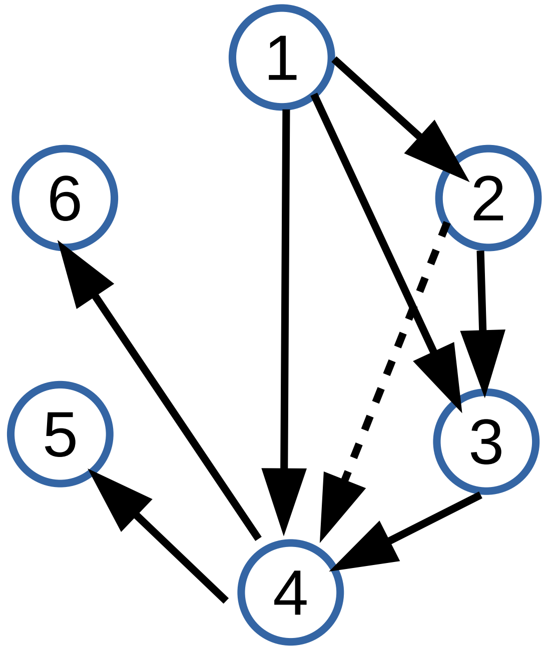
and the following *MDL metric* based on the MDL principle:

$$\text{MDL score} = -LL + \frac{K}{2}\log N$$

- In both cases the log-likelihood is negated, so the aim is to minimize these scores.
- More Bayesian approach: use prior over model structures

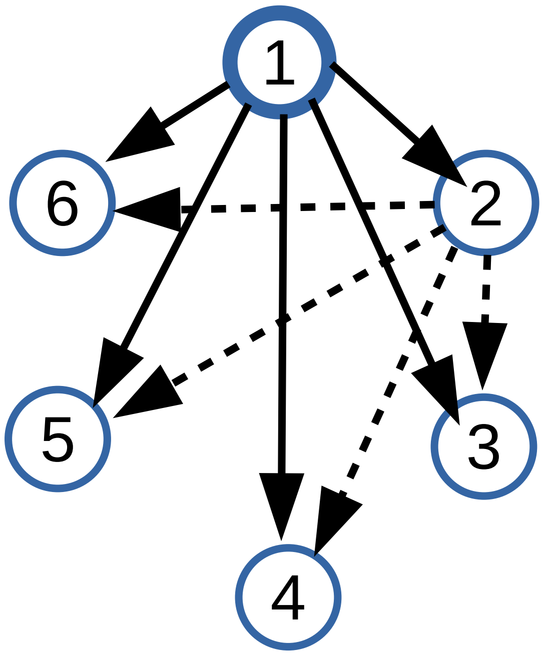
The K2 Network structure learning algoritm

- *K2:* a simple and very fast learning algorithm,
- Starts with a given ordering of the nodes
- Processes each node in turn and greedily considers adding edges from previously processed nodes to the current one
- In each step it adds the edge that maximizes the network's score
- When there is no further improvement, attention turns to the next node
- The number of parents for each node can be restricted to a predefined maximum to mitigate overfitting



Tree augmented naïve bayes (TAN)

- Another good learning algorithm for Bayesian network classifiers
- Takes the Naïve Bayes (NB) classifier and adds edges to it
- The class attribute is the sole parent of each node in a NB model: TAN considers adding a second parent to each node
- If the class node and all corresponding edges are excluded from consideration, and assuming that there is exactly one node to which a second parent is not added, the resulting classifier has a tree structure rooted at the parentless node—hence the name
- For this restricted network type there is an efficient algorithm based on computing a maximum weighted spanning tree
- Algorithm is linear in the number of instances and quadratic in the number of attributes

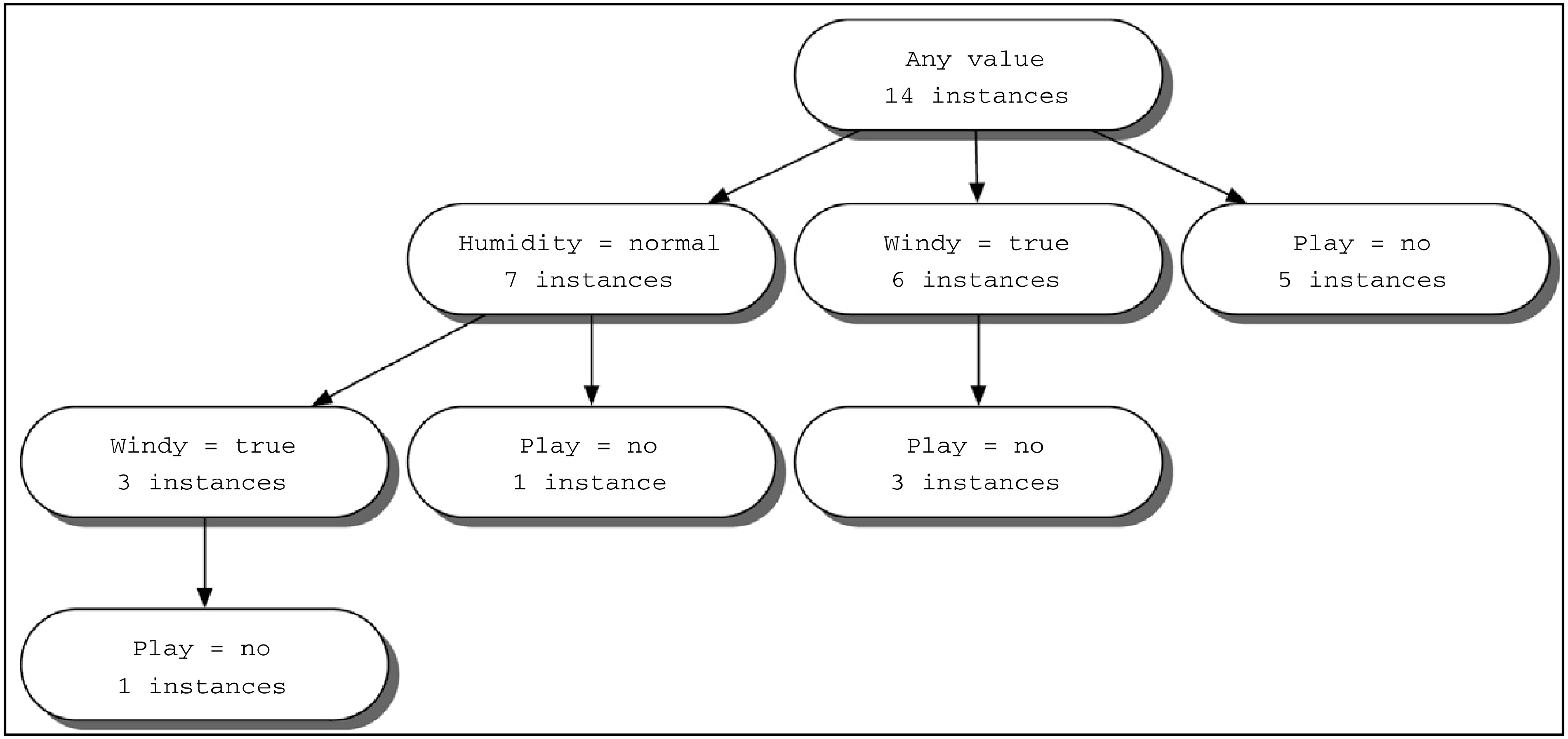


Data structures for fast Learning

- Learning Bayesian networks involves a lot of counting
- For each network structure considered in the search, the data must be scanned afresh to obtain the counts needed to fill out the conditional probability tables
- Counts can be stored effectively in a structure called an *all-dimensions (AD) tree*, which is analogous to the *k*D-trees used for nearest neighbor search
- Each node of the tree represents the occurrence of a particular combination of attribute values
- Straightforward to retrieve the count for a combination that occurs in the tree
- However, the tree does not explicitly represent many nonzero counts because the most populous expansion for each attribute is omitted

AD Tree example

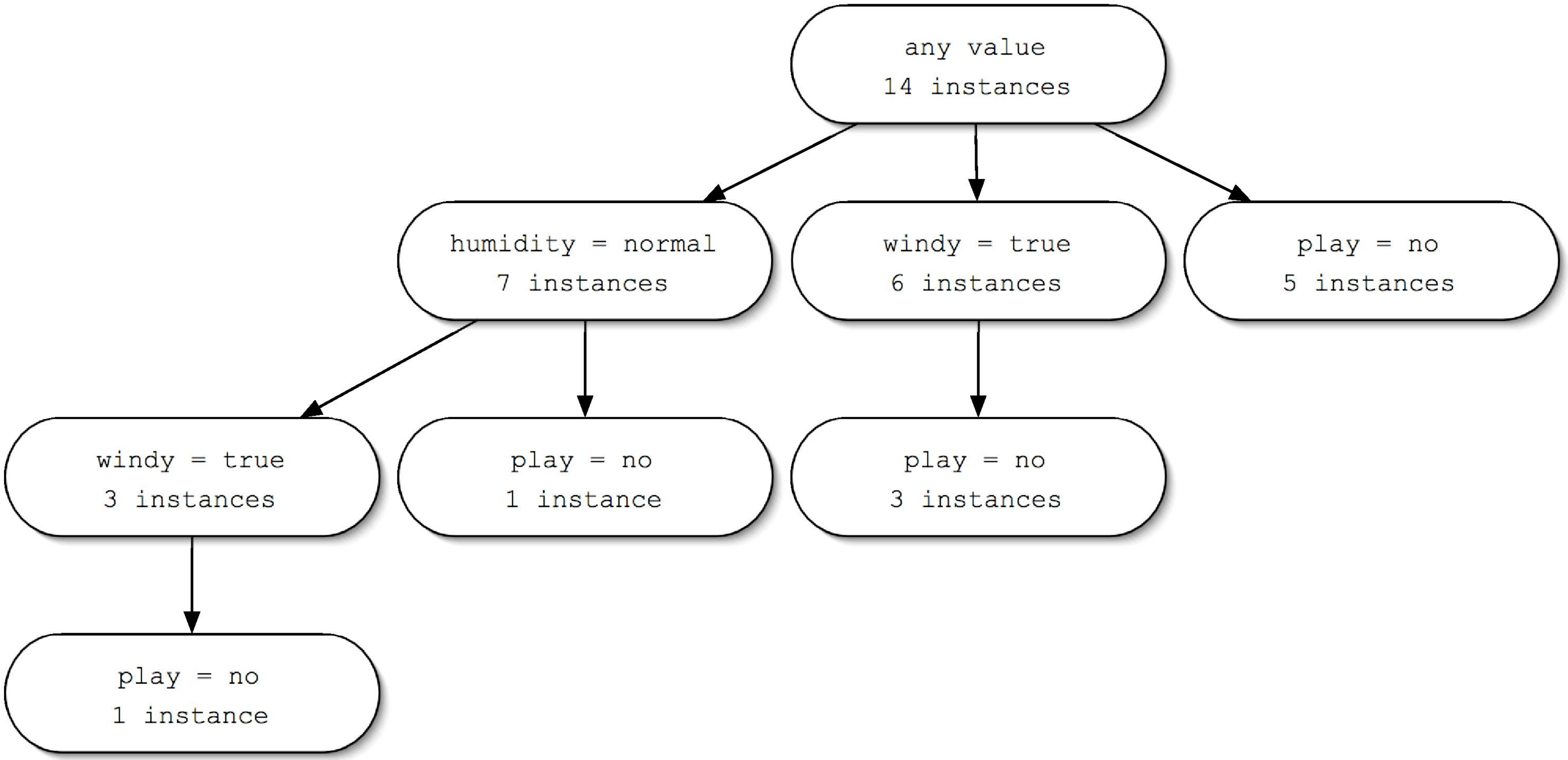
Humidity	Windy	Play	Count
High	True	Yes	1
High	True	No	2
High	False	Yes	2
High	False	No	2
Normal	True	Yes	2
Normal	True	No	1
Normal	False	Yes	4
Normal	False	No	0



Building an AD tree

- Assume each attribute in the data has been assigned an index
- Then, expand node for attribute *i* with the values of all attributes *j* > *i*
  - Two important restrictions:
    - Most populous expansion for each attribute is omitted (breaking ties arbitrarily)
    - Expansions with counts that are zero are also omitted
- The root node is given index zero

AD tree example



humidity = normal, windy = true, play = yes?

humidity = high, windy = true, play = no?

#(windy=true, play=no (3)) - #(humidity=normal, windy=true, play=no (1))

Bibliographic Notes & Further Reading

Learning and Bayesian Networks

- The K2 algorithm for learning Bayesian networks was introduced by Cooper and Herskovits (1992).
- Bayesian scoring metrics are covered by Heckerman et al. (1995).
- Friedman et al. (1997) introduced the tree augmented Naïve Bayes algorithm, and also describe multinets.
- AD trees were introduced and analyzed by Moore and Lee (1998)
- Komarek and Moore (2000) introduce AD trees for incremental learning that are also more efficient for datasets with many attributes.

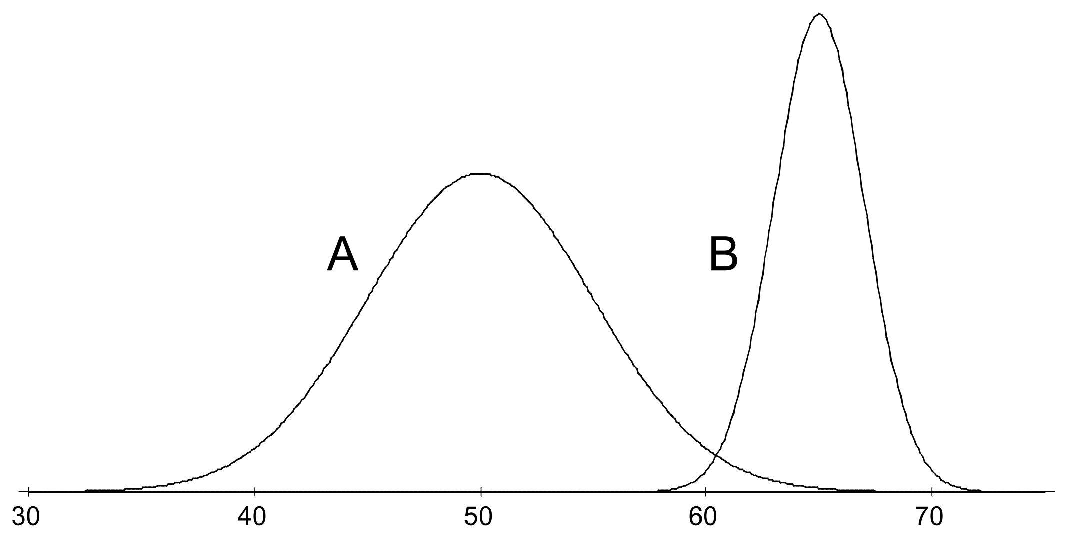
Clustering with a Gaussian Mixture

- Given the data on the left *without the labels A and B*, we wish to estimate a model for a two class Gaussian Mixture Model (GMM) on the right

Data

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

Model



$\mu_A=50,\ \sigma_A=5,\ p_A=0.6$ $\quad$ $\mu_B=65,\ \sigma_B=2,\ p_B=0.4$

Estimating Gaussian parameters

- If we knew which of the two distributions each instance came from, finding the five parameters would be easy—just estimate the mean and standard deviation for $n=n_A$ or $n=n_B$ samples x_1, x_2, …, x_n for each cluster, A and B

$$\mu = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

$$\sigma^2 = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \ldots + (x_n - \mu)^2}{n - 1}$$

- To estimate the fifth parameter p_A, just take the proportion of the instances that are in the A cluster, then $p_B=1-p_A$.

Motivating the EM algorithm

- If you knew the five parameters, finding the (posterior) probabilities that a given instance comes from each distribution would be easy
- Given an instance x_i, the probability that it belongs to cluster A is

$$P(\mathrm{A}|x_i) = \frac{P(x_i \,|\, \mathrm{A}) \cdot P(\mathrm{A})}{P(x_i)} = \frac{N(x_i; \mu_\mathrm{A}, \sigma_\mathrm{A}) p_\mathrm{A}}{N(x_i; \mu_\mathrm{A}, \sigma_\mathrm{A}) p_\mathrm{A} + N(x_i; \mu_\mathrm{B}, \sigma_\mathrm{B}) p_\mathrm{B}}$$

where $N()$ is the normal or Gaussian distribution

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

Learning the clusters

- Assume:
  - we know there are *k* clusters
- Learn the clusters ⇒
  - determine their parameters
  - I. e. means and standard deviations
- Performance criterion:
  - *probability of training data given the clusters*
- EM algorithm
  - finds a local maximum of the likelihood

EM algorithm

- EM = Expectation-Maximization
  - Generalize *k*-means to probabilistic setting
- Iterative procedure:
  - E “expectation” step:
    Calculate cluster probability for each instance
  - M “maximization” step:
    Estimate distribution parameters from cluster probabilities
- Store cluster probabilities as instance weights
- Stop when improvement is negligible

More on EM

- Estimate parameters from weighted instances

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

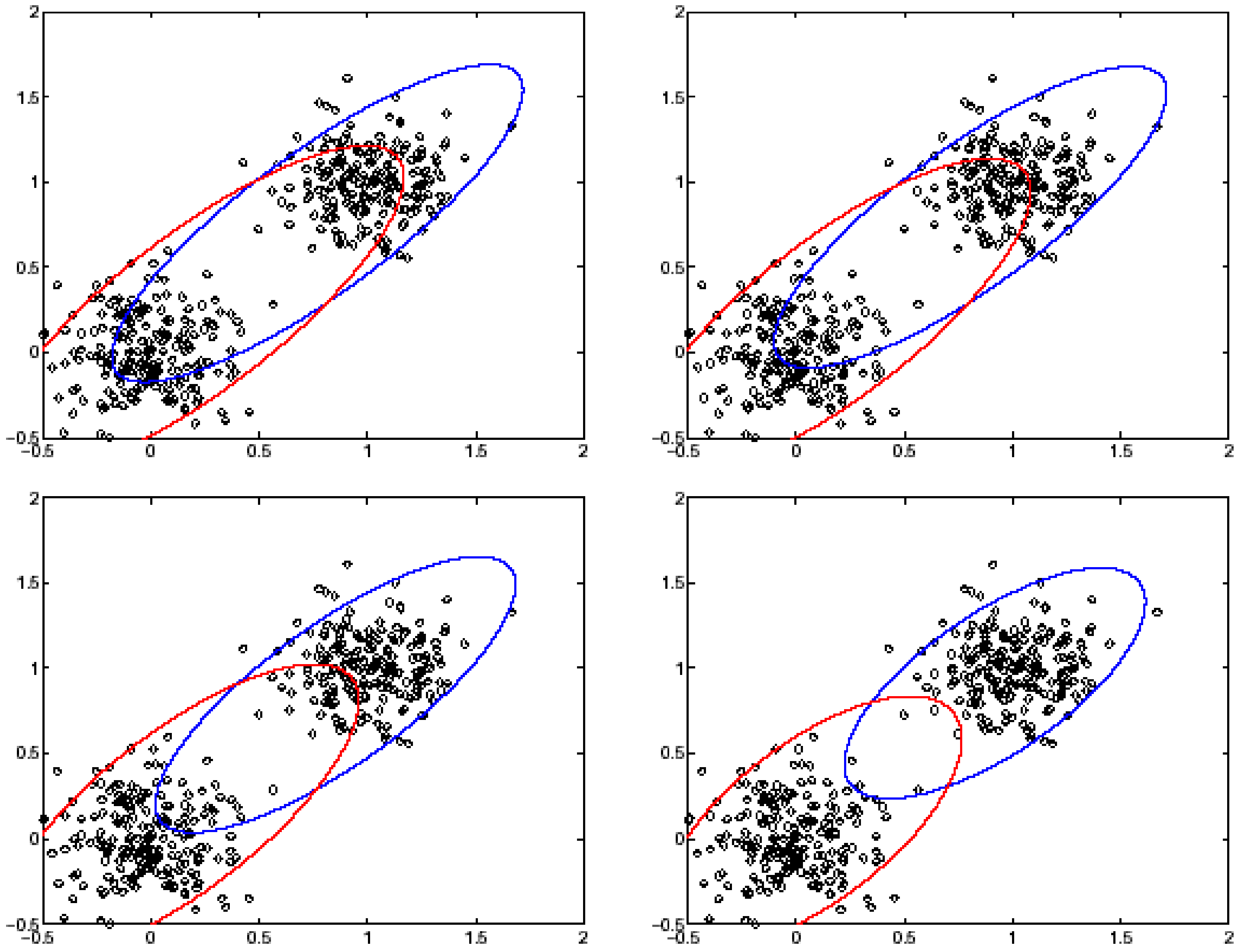
$$\sigma_A^2 = \frac{w_1 (x_1 - \mu)^2 + w_2 (x_2 - \mu)^2 + \ldots + w_n (x_n - \mu)^2}{w_1 + w_2 + \ldots + w_n}$$

- Stop when log-likelihood saturates

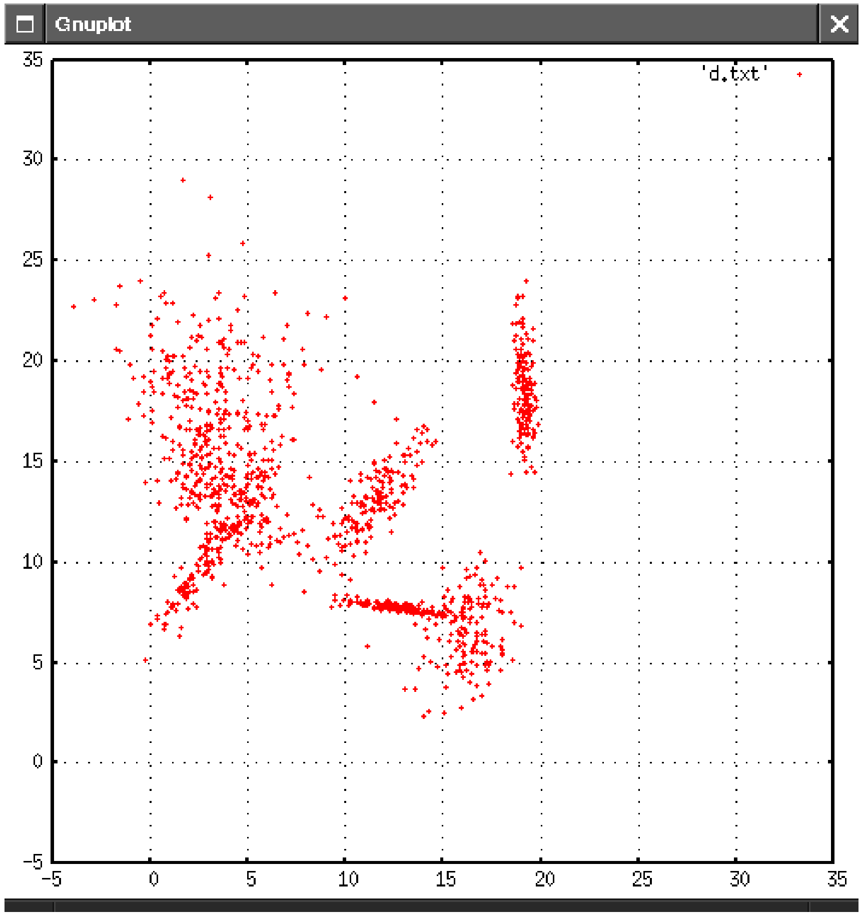
- Log-likelihood:

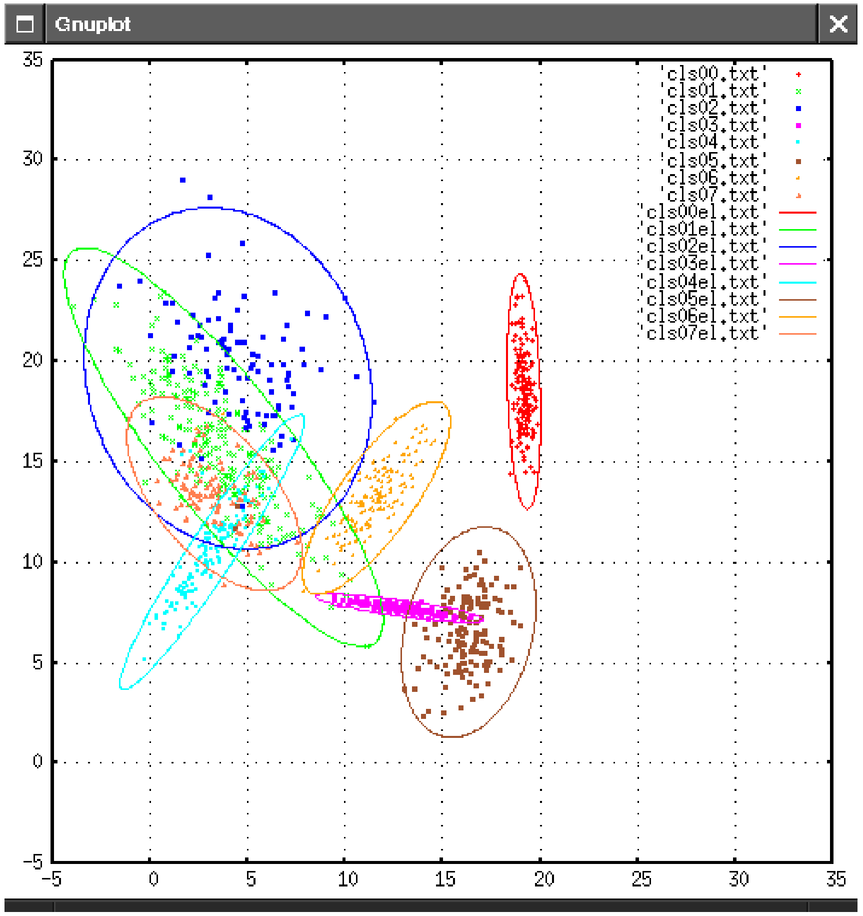
$$\sum_i \log \left(p_A Pr[x_i | A] + p_B Pr[x_i | B] \right)$$

EM Example (1)



EM Example (2)


Gnuplot
'd.txt'
35
30
25
20
15
10
5
0
-5
-5
0
5
10
15
20
25
30
35


Gnuplot
'cls00.txt'
'cls01.txt'
'cls02.txt'
'cls03.txt'
'cls04.txt'
'cls05.txt'
'cls06.txt'
'cls07.txt'
'cls00el.txt'
'cls01el.txt'
'cls02el.txt'
'cls03el.txt'
'cls04el.txt'
'cls05el.txt'
'cls06el.txt'
'cls07el.txt'
35
30
25
20
15
10
5
0
-5
-5
0
5
10
15
20
25
30
35

Extending the mixture Model

- The Gaussian distribution generalizes to n-dimensions
- Consider a two-dimensional model consisting of independent Gaussian distributions for each dimension
- We can transform from scalar to matrix notation for a two dimensional Gaussian distribution as follows:

$$P(x_1,x_2)=\frac{1}{\sqrt{2\pi}\sigma_1}\exp\left[-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}\right]\frac{1}{\sqrt{2\pi}\sigma_2}\exp\left[-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}\right]$$

$$=(2\pi)^{-1}\left(\sigma_1^2\sigma_2^2\right)^{-1/2}\exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{\mu})^T\begin{bmatrix}\sigma_1^2 & 0\\ 0 & \sigma_2^2\end{bmatrix}^{-1}(\mathbf{x}-\mathbf{\mu})\right\}$$

$$=(2\pi)^{-1}|\Sigma|^{-1/2}\exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{\mu})^T\Sigma^{-1}(\mathbf{x}-\mathbf{\mu})\right\},$$

- **Σ** is the covariance *matrix*, |**Σ**| is its determinant, the vector **x** = $[x_1\ x_2]^T$, and the mean *vector* $\boldsymbol{\mu}$ = $[\mu_{1,}\ \mu_2]^T$

The multivariate Gaussian distribution

- Can be written in the following general form

$$P(x_1, x_2, \ldots, x_d) = (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right\}.$$

- The equation for estimating the covariance matrix is

$$\mathbf{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T.$$

- The mean is simply

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i.$$

Clustering with correlated attributes

- If all attributes are continuous one can simply use a full covariance Gaussian mixture model
- But one needs to estimate $n(n + 1)/2$ parameters per mixture component for a full covariance matrix model
- As we will see later principal component analysis (PCA) can be formulated as a probabilistic model, yielding probabilistic principal component analysis (PPCA),
- Approaches known as *mixtures of principal component analyzers* or *mixtures of factor analyzers* provide ways of using a much smaller number of parameters to represent large covariance matrices