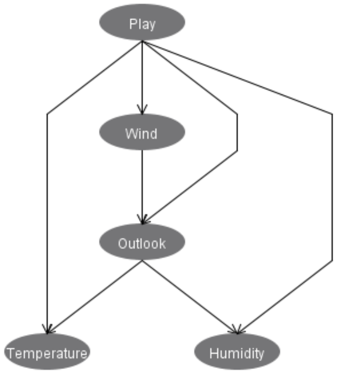


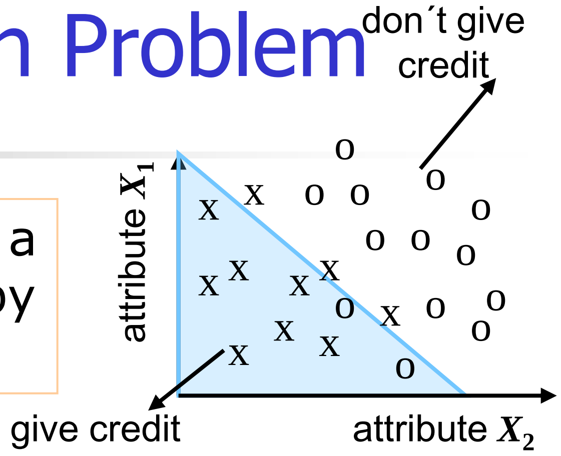
# Bayesian Networks Classifiers

Gladys Castillo
University of Aveiro

# Bayesian Networks Classifiers

## Part I – Naive Bayes

# The Supervised Classification Problem

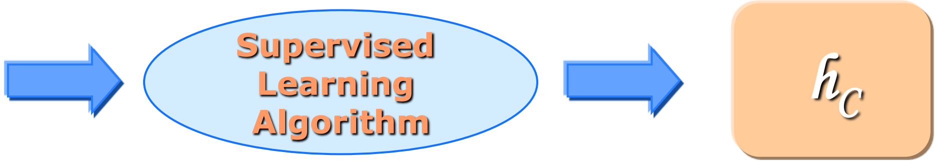


A **classifier** is a function $f: \Omega_{\mathbf{X}} \rightarrow \Omega_C$ that assigns a class label $c \in \Omega_C = \{c_1, \ldots, c_m\}$ to objects described by a set of attributes $\mathbf{X} = \{X_1, X_2, \ldots, X_n\} \in \Omega_{\mathbf{X}}$

**Learning Phase:** **Given:** a dataset $\mathcal{D}$ with $N$ labeled examples of $<\mathbf{X}, C>$

**Build:** a **classifier, a hypothesis** $h_C: \Omega_{\mathbf{X}} \rightarrow \Omega_C$ that can correctly predict the class labels of new objects

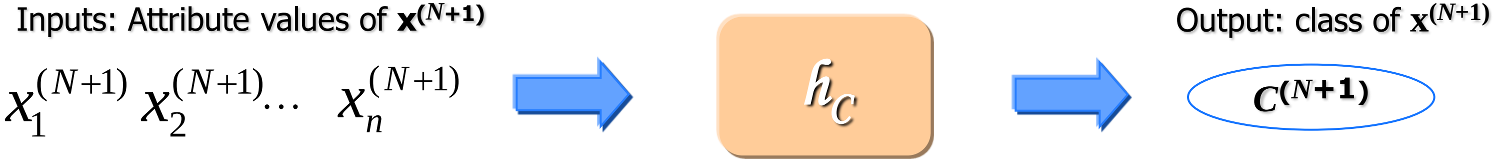
| $\mathcal{D}$ | $X_1$ | … | $X_n$ | $C$ |
|---|---|---|---|---|
| $<\mathbf{x}^{(1)}, c^{(1)}>$ | $x_1^{(1)}$ | | $x_n^{(1)}$ | $c^{(1)}$ |
| $<\mathbf{x}^{(2)}, c^{(2)}>$ | $x_1^{(2)}$ | | $x_n^{(2)}$ | $c^{(2)}$ |
| … | … | | | … |
| $<\mathbf{x}^{(N)}, c^{(N)}>$ | $x_1^{(N)}$ | | $x_n^{(N)}$ | $c^{(N)}$ |



Supervised Learning Algorithm

$h_C$

**Classification Phase:** the class attached to $\mathbf{x}^{(N+1)}$ is $c^{(N+1)} = h_C(\mathbf{x}^{(N+1)}) \in \Omega_C$



Inputs: Attribute values of $\mathbf{x}^{(N+1)}$

$x_1^{(N+1)} \; x_2^{(N+1)} \ldots \; x_n^{(N+1)}$

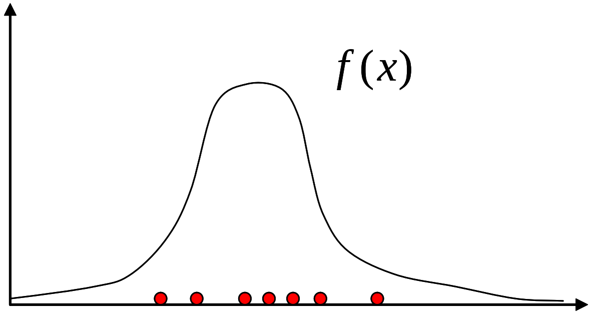
$h_C$

Output: class of $\mathbf{x}^{(N+1)}$

$c^{(N+1)}$

# Statistical Classifiers

- Treat the attributes X= $\{X_1, X_2, \ldots, X_n\}$ and the class $C$ as random variables
  - A random variable is defined by its *probability density function*
- Give the probability $P(c_j \mid \mathbf{x})$ that **x** belongs to a particular class rather than a simple classification



*Probability density function of a random variable and few observations*

⇒ instead of having the map $\mathbf{X} \rightarrow C$ , we have $\mathbf{X} \rightarrow P(C \mid \mathbf{X})$

The class $c^*$ attached to an example is the class with bigger $\mathbf{P(c_j \mid x)}$

$$\boxed{c^* = h_C(\mathbf{x}) = \arg\max_{j=1\ldots m} P(c_j \mid \mathbf{x})}$$

# Bayesian Classifiers

Bayesian because the class *c** attached to an example **x** is determined by the Bayes' Theorem

$$\left.\begin{aligned} P(X,C) &= P(X|C).P(C) \\ P(X,C) &= P(C|X).P(X) \end{aligned}\right\} \boxed{P(C|X) = \frac{P(C)P(X|C)}{P(X)}}$$

Bayes theorem is the main tool in Bayesian inference

*We can combine the prior distribution and the likelihood of the observed data in order to derive the posterior distribution.*

# Bayes Theorem

## Example

$$P(C\,|\,X)=\frac{P(C)P(X\,|\,C)}{P(X)}$$

**posterior α prior x likelihood**

Before observing the data, our prior beliefs can be expressed in a prior probability distribution that represents the knowledge we have about the unknown features.
After observing the data our revised beliefs are captured by a posterior distribution over the unknown features.

Given:

- ✓ A doctor knows that meningitis causes stiff neck 50% of the time ← likelihood
- ✓ Prior probability of any patient having meningitis is 1/50,000 ← prior
- ✓ Prior probability of any patient having stiff neck is 1/20 ← prior

If a patient has stiff neck,

**what's the probability he/she has meningitis**?

$$P(M\,|\,S)=\frac{P(S\,|\,M)P(M)}{P(S)}=\frac{0.5\times 1/50000}{1/20}=0.0002$$

# Bayesian Classifier

Classify **x** to the class which has bigger posterior probability

$$c^* = h_C(\mathbf{x}) = \arg \max_{j=1\ldots m} P(c_j \mid \mathbf{x})$$

Maximum a posteriori classification

**How to determine $P(c_j \mid \mathbf{x})$ for each class $c_j$ ?**

Bayes Theorem

$$P(c_j \mid \mathbf{x}) = \frac{P(c_j)P(\mathbf{x} \mid c_j)}{P(\mathbf{x})}$$

*P(x) can be ignored because is just a normalizing constant that ensures the posterior adds up to 1; it can be computed by summing up the numerator over all possible values of* C, i.e., $P(\mathbf{x}) = \sum_j P(c_j)\, P(\mathbf{x} \mid c_j)$

Bayesian Classification Rule

$$c^* = h_{Bayes}(\mathbf{x}) = arg \max_{j=1\ldots m} P(c_j)P(\mathbf{x} \mid c_j)$$

Duda and Hart (1973); Langley (1992)

# Naïve Bayes (NB) Classifier

- **"Bayes"** because the class c* attached to an example **x** is determined by the Bayes' Theorem

$$c^* = h_{Bayes}(\mathbf{x}) = arg \max_{j=1 \ldots m} P(c_j) \boxed{P(\mathbf{x} \mid c_j)}$$

when the attribute space is high dimensional direct estimation is hard unless we introduce some assumptions

- **"Naïve"** because of its very naïve independence assumption:

all the attributes are conditionally independent given the class

$$P(\mathbf{x} \mid c_j) = \prod_{i=1}^{n} P(X_i = x_i \mid c_j)$$

*P*(**x** | *c*$_j$) can be decomposed into a product of n terms, one term for each attribute

NB Classification Rule

$$c^* = h_{NB}(\mathbf{x}) = arg \max_{j=1 \ldots m} P(c_j) \prod_{i=1}^{n} P(X_i = x_i \mid c_j)$$

# Naïve Bayes (NB)

## Learning Phase (Statistical Parameter Estimation)

Given a training dataset $\mathcal{D}$ of $N$ labeled examples (assuming complete data)

1. Estimate $P(c_j)$ for each class $c_j$

$$\boxed{\hat{P}(c_j) = \frac{N_j}{N}}$$

$N_j$ - the number of examples of the class $\mathbf{c_j}$

2. Estimate $P(X_i{=}x_k\,|c_j)$ for each value $\mathbf{x_k}$ of the attribute $\mathbf{X_i}$ and for each class $\mathbf{c_j}$

- $X_i$ discrete

$$\boxed{\hat{P}(X_i = x_k \mid c_j) = \frac{N_{ijk}}{N_j}}$$

$N_{ijk}$ - number of examples of the class $c_j$ having the value $\mathbf{x_k}$ for the attribute $\mathbf{X_i}$

- $X_i$ continuous

Two options

- The attribute is **discretized** and then treats as a discrete attribute
- A **Normal distribution** is usually assumed

$$\boxed{P(X_i = x_k \mid c_j) = g(x_k; \mu_{ij}, \sigma_{ij})} \quad \text{onde} \quad \boxed{g(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$$

The mean $\boldsymbol{\mu_{ij}}$ and the standard deviation $\boldsymbol{\sigma_{ij}}$ are estimated from $\mathcal{D}$

# Continuous Attributes
## Normal or Gaussian Distribution

Estimate $P(X_i{=}x_k\,|c_j)$ for a value of the attribute $X_i$ and for each class $c_j$

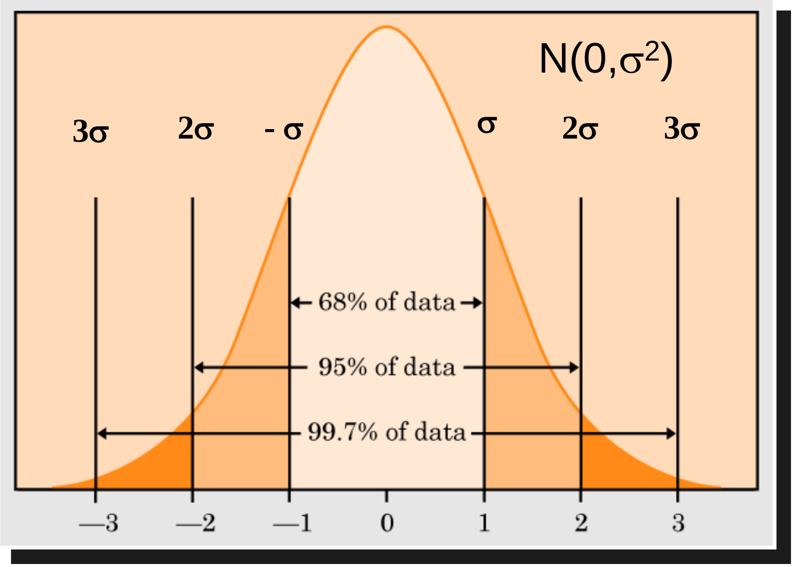
- For real attributes a Normal distribution is usually assumed

$$\boxed{P(X_i = x_k \mid c_j) = g(x_k; \mu_{ij}, \sigma_{ij})} \Rightarrow \boxed{g(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$$

$X_i \mid c_j \sim N(\mu_{ij}, \sigma^2_{ij})$ - the mean $\mu_{ij}$ and the standard deviation $\sigma_{ij}$ are estimated from $\mathcal{D}$

For a variable $X \sim N(74, 36)$, the probability of observing the value 66 is given by:

$$f(x) = g(66; 74, 6) = 0.0273$$



The density probability function

f(x) is symmetrical around its mean

# Naïve Bayes
## Probability Estimates

**Example from John & Langley (1995)**

Training dataset $\mathcal{D}$

| Class | $X_1$ | $X_2$ |
|---|---|---|
| + | *a* | 1.0 |
| + | *b* | 1.2 |
| + | *a* | 3.0 |
| - | *b* | 4.4 |
| - | *b* | 4.5 |

### Binary Classification Problem

- Two classes: **+ (positive)** , **- (negative)**
- Two attributes:
  - $X_1$ – discrete which takes values *a* e *b*
  - $X_2$ – continuous

1. Estimate $P(c_j)$ for each class $c_j$

$$\hat{P}(C=+)=\frac{3}{5} \qquad \hat{P}(C=-)=\frac{2}{5}$$

2. Estimate $P(X_i=x_k\,|c_j)$ for each value of $X_1$ and each class $c_j$

$X_1$ discrete

$$\hat{P}(X_i=a\,|\,C=+)=\frac{2}{3} \qquad \hat{P}(X_i=b\,|\,C=+)=\frac{1}{3}$$

$$\hat{P}(X_i=a\,|\,C=-)=\frac{0}{2} \qquad \hat{P}(X_i=b\,|\,C=-)=\frac{2}{2}$$

$X_2$ continuos

$$P(X_2=x\,|\,C=+)=g(x;1.73,1.10)$$

$\mu_{2+}$= 1.73, $\sigma_{2+}$ = 1.10

$$P(X_2=x\,|\,C=-)=g(x;4.45,0.07)$$

$\mu_{2-}$= 4.45, $\sigma^2_{2+}$ = 0.07

dataset from the UCI repository

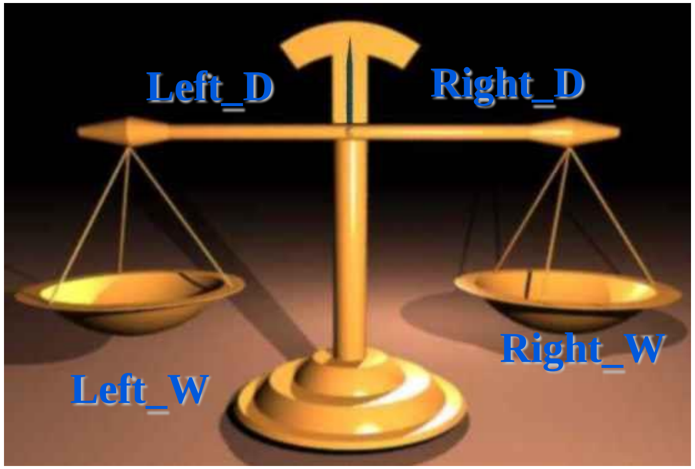
# The Balance-scale Problem

The dataset was generated to model psychological experimental results

- **Each example has 4 numerical attributes:**
  - the left weight (Left_W)
  - the left distance (Left_D)
  - the right weight (Right_W)
  - right distance (Right_D)

- **Each example is classified into 3 classes:** the balance-scale:
  - tip to the right (Right)
  - tip to the left (Left)
  - is balanced (Balanced)

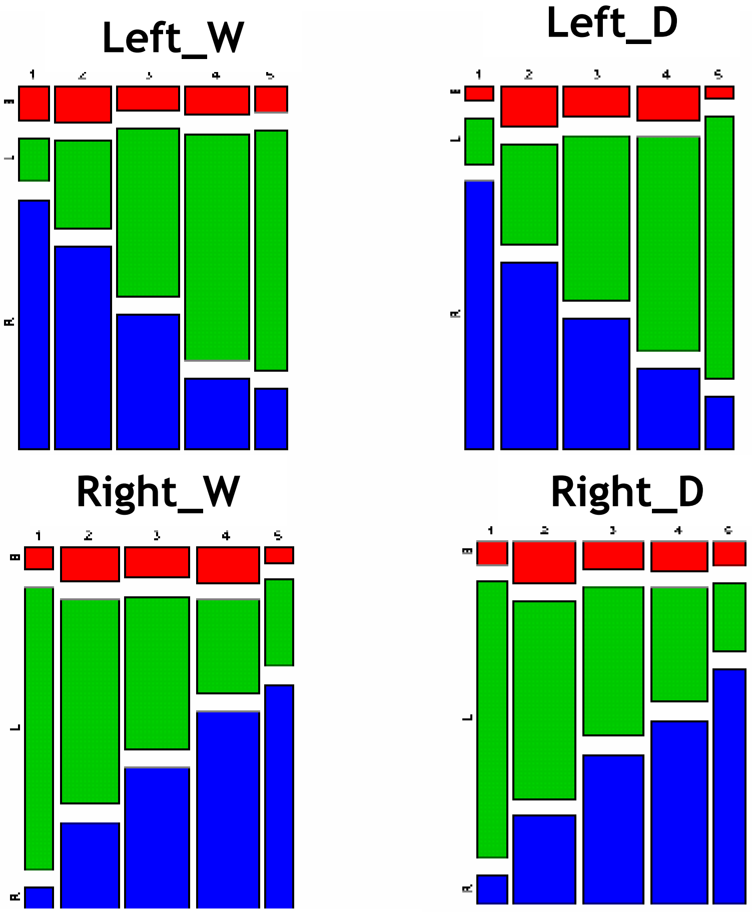
- **3 target rules:**
  - If LD x LW > RD x RW ⇒ tip to the left
  - If LD x LW < RD x RW ⇒ tip to the right
  - If LD x LW = RD x RW ⇒ it is balanced

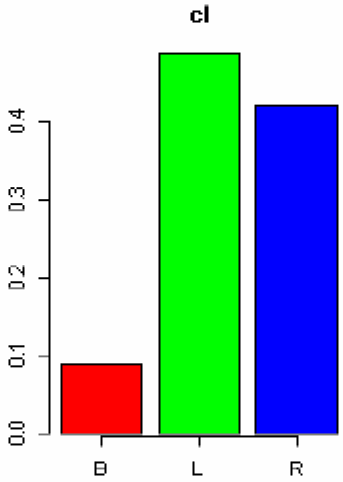


# The Balance-scale Problem

Discretization is applied: each attribute is mapped to 5  intervals

| Balance-Scale DataSet | | | | |
|---|---|---|---|---|
| Left_W | Left_D | Right_W | Right_D | Class |
| 1 | 5 | 4 | 2 | Right |
| 2 | 5 | 3 | 2 | Left |
| 3 | 4 | 6 | 2 | Balanced |
| … | … | … | … | … |





Adapted from © João Gama's slides "Aprendizagem Bayesiana"

# The Balance-scale Problem

## Learning Phase

565 examples

| Balance-Scale DataSet | | | | |
|---|---|---|---|---|
| Left_W | Left_D | Right_W | Right_D | Class |
| 1 | 5 | 4 | 2 | Right |
| 2 | 5 | 3 | 2 | Left |
| 3 | 4 | 6 | 2 | Balanced |
| … | … | … | … | … |

Build → Contingency tables

| Classes Counters | | | |
|---|---|---|---|
| Left | Balanced | Right | Total |
| 260 | 45 | 260 | 565 |

**Contingency Tables**

| | Attribute: Left_W | | | | |
|---|---|---|---|---|---|
| Class | I1 | I2 | I3 | I4 | I5 |
| **Left** | 14 | 42 | 61 | 71 | 72 |
| **Balanced** | 10 | 8 | 8 | 10 | 9 |
| **Right** | 86 | 66 | 49 | 34 | 25 |

| | Attribute: Left_D | | | | |
|---|---|---|---|---|---|
| Class | I1 | I2 | I3 | I4 | I5 |
| **Left** | 16 | 38 | 59 | 70 | 77 |
| **Balanced** | 8 | 10 | 9 | 10 | 8 |
| **Right** | 90 | 57 | 49 | 37 | 27 |

| | Attribute: Right_W | | | | |
|---|---|---|---|---|---|
| Class | I1 | I2 | I3 | I4 | I5 |
| **Left** | 87 | 63 | 49 | 33 | 28 |
| **Balanced** | 8 | 10 | 10 | 9 | 8 |
| **Right** | 16 | 37 | 58 | 70 | 79 |

| | Attribute: Right_D | | | | |
|---|---|---|---|---|---|
| Class | I1 | I2 | I3 | I4 | I5 |
| **Left** | 91 | 65 | 44 | 35 | 25 |
| **Balanced** | 8 | 10 | 8 | 10 | 9 |
| **Right** | 17 | 37 | 57 | 67 | 82 |

Assuming complete data, the computation of all the required estimates requires a simple scan through the data, an operation of time complexity $O(\mathbf{N} \times \mathbf{n})$, where $N$ is the number of training examples and *n is the number of attributes*.

# The Balance-scale Problem

## Classification Phase

$$c^* = h_{NB}(\mathbf{x}) = arg \max_{j=1...m} P(c_j)\prod_{i=1}^{n} P(X_i = x_i \mid c_j)$$

**How NB classifies this example?**

| Left_W | Left_D | Right_W | Right_D | Class |
|---|---|---|---|---|
| 1 | 5 | 4 | 2 | **?** |

The class counters and contingency tables are used to compute the posterior probabilities for each class

We need to estimate the posterior probabilities $P(c_j \mid \mathbf{x})$ for each class

$$P(c_j \mid \mathbf{x}) = P(c_j) \text{ x } P(\textbf{Left\_W=1} \mid c_j) \text{ x } P(\textbf{Left\_D=5} \mid c_j)$$
$$\text{x } P(\textbf{Right\_W=4} \mid c_j) \text{ x } P(\textbf{Right\_D=2} \mid c_j),$$
$$c_j \in \{\text{Left, Balanced, Right}\}$$

The class for this example is the class which has bigger posterior probability

| P(Left\|x) | P(Balanced\|x) | P(Right\|x) |
|---|---|---|
| 0.277796 | 0.135227 | **0.586978** |

***max***

**Class** $= Right$

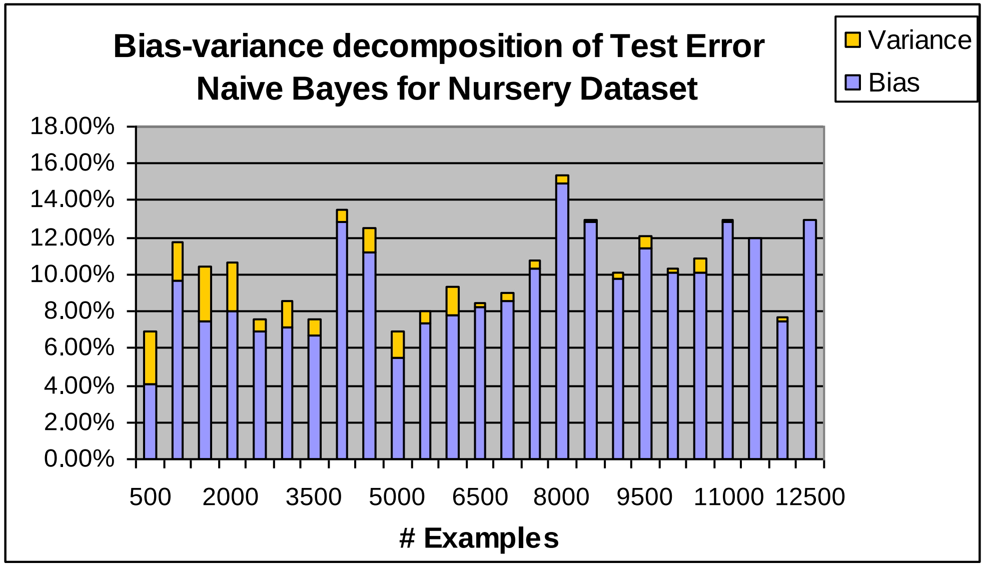
# Naïve Bayes Performance

- NB is one of more simple and effective classifiers

- NB has a very strong unrealistic independence assumption:

all the attributes are conditionally independent given the value of class

- in practice: independence assumption is violated ⇒ HIGH BIAS
  - it can lead to poor classification
- However, NB is efficient due to its high variance management
  - less parameters ⇒ LOW VARIANCE



# Improving Naïve Bayes

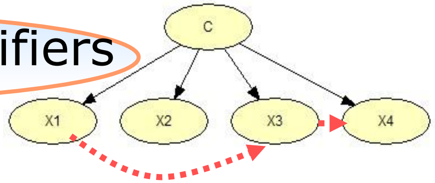
- reducing the bias of the parameter estimates
  - by improving the probability estimates computed from data

$$\hat{P}(c_j \mid \mathbf{x}) = \prod_{i=1}^{n} \hat{P}(X_i = x_i \mid c_j)\hat{P}(c_j)$$

**Relevant works:**

- Web and Pazzani (1998) - "Adjusted probability naive Bayesian induction" in LNCS v 1502
- J. Gama (2001, 2003) - "Iterative Bayes", in Theoretical Computer Science, v. 292

- reducing the bias resulting from the modeling error
  - by relaxing the attribute independence assumption
  - one natural extension: Bayesian Network Classifiers



**Relevant works:**

- Friedman, Geiger and Goldszmidt (1998) "Bayesian Network Classifiers" in Machine Learning, 29
- Pazzani (1995) - "Searching for attribute dependencies in Bayesian Network Classifiers" in Proc. of the 5th Workshop of Artificial Intelligence and Statistics
- Keogh and Pazzani (1999) - "Learning augmented Bayesian classifiers…", in Theoretical Computer Science, v. 292

# Bayesian Networks Classifiers

Part II - Introduction to Bayesian Networks

# Reasoning under Uncertainty

## Probabilistic Approach

- A problem domain is modeled by a set of random variables $X_1, X_2 \ldots, X_n$
- Knowledge about the problem domain is represented by a joint probability distribution $P(X_1, X_2, \ldots, X_n)$

Example: ALARM network (Pearl 1988)

- Story: In LA, burglary and earthquake are not uncommon. They both can cause alarm. In case of alarm, two neighbors John and Mary may call
- Problem: Estimate the probability of a burglary based who has or has not called
- Variables: Burglary (B), Earthquake (E), Alarm (A), JohnCalls (J), MaryCalls (M)
- Knowledge required by the probabilistic approach in order to solve this problem: P(B, E, A, J, M)

# Inference with Joint Probability Distribution

- To specify $P(X_1, X_2, \ldots, X_n)$
  - we need at least **$2^n - 1$** numbers probability

$\Rightarrow$

  - exponential model size
  - knowledge acquisition difficult (complex, unnatural)
  - exponential storage and inference

Solution:

- by exploiting **conditional independence** we can obtain an appropriate factorization of the **joint probability distribution**

  $\Rightarrow$ the number of parameters could be substantially reduced

- A good factorization is provided with **Bayesian Networks**

Pearl (1988, 2000); Jensen (1996); Lauritzen (1996)

# Bayesian Networks

Bayesian Networks (BNs) graphically represent the joint probability distribution of a set $\mathbf{X}$ of random variables in a problem domain

A BN=($S$, $\Theta_S$) consist in two components:

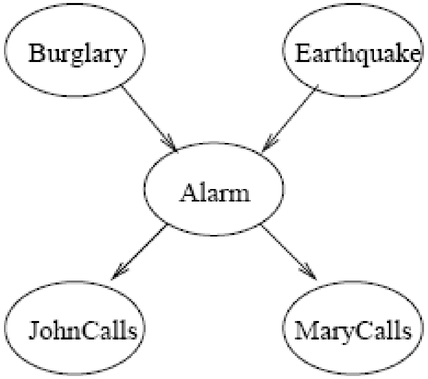
Graph theory

Probability theory

## Qualitative part

the structure $S$ - a Directed Acyclic Graph (DAG)

- nodes – random variables
- arcs - direct dependence between variables



## Quantitative part

the set of parameters $\Theta_S$ = set of conditional probability distributions (CPDs)

- discrete variables: $\Theta_S$ - multinomial (CPDs are CPTs)
- continuos variables: $\Theta_S$ - Normal ou Gaussiana

**Markov condition :** *each node is independent of its non descendants given its parents* in $S$

⇒ The joint distribution is factorized

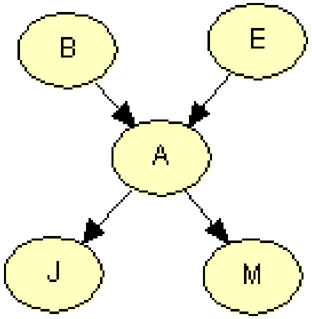
$$P(X_1,\dots,X_n) = \prod_i P(X_i \mid pa(X_i))$$

# Example: Alarm Network (Pearl) Bayesian Network

**X** = { **B** - Burglary, **E**- Earthquake, **A** - Alarm, **J** - JohnCalls, **M** – MaryCalls }

The Bayesian Network has two components:

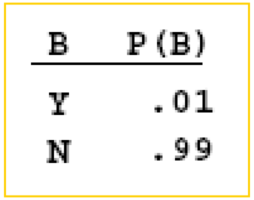
- the structure – a Directed Acyclic Graph



- the parameters – a set of conditional probability tables (CPTs)

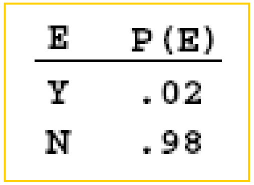
⇒ pa(**B**) = { }, pa(**E**) = { }, pa(**A**) = {B, E}, pa(**J**) = {A}, pa(**M**) = {A}

P(**B**)



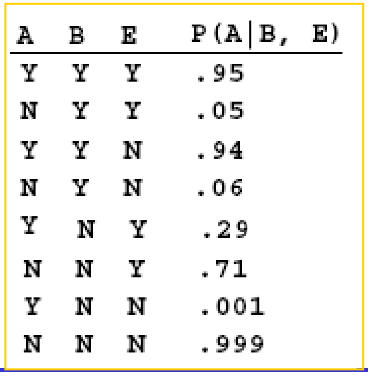
| B | P(B) |
|---|---|
| Y | .01 |
| N | .99 |

P(**E**)



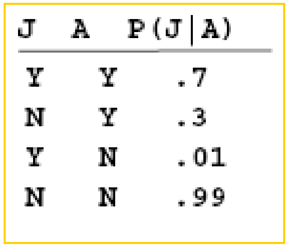
| E | P(E) |
|---|---|
| Y | .02 |
| N | .98 |

P(**A** | B, E)



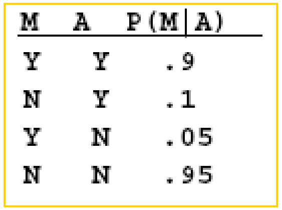
| A | B | E | P(A\|B, E) |
|---|---|---|---|
| Y | Y | Y | .95 |
| N | Y | Y | .05 |
| Y | Y | N | .94 |
| N | Y | N | .06 |
| Y | N | Y | .29 |
| N | N | Y | .71 |
| Y | N | N | .001 |
| N | N | N | .999 |

P(**J** | A)



| J | A | P(J\|A) |
|---|---|---|
| Y | Y | .7 |
| N | Y | .3 |
| Y | N | .01 |
| N | N | .99 |

P( **M** | A}

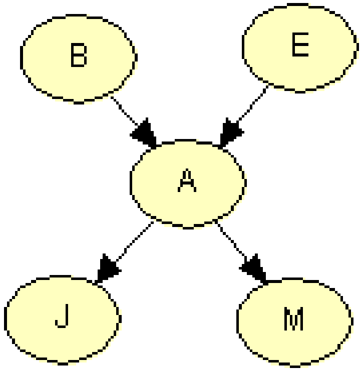


| M | A | P(M\|A) |
|---|---|---|
| Y | Y | .9 |
| N | Y | .1 |
| Y | N | .05 |
| N | N | .95 |

**Model size reduced from 31 to 1+1+4+2+2=10**

# Example: Alarm Network (Pearl)
## Factored Joint Probability Distribution

**X** = {B (Burglary), (E) Earthquake, (A) Alarm, (J) JohnCalls, (M) MaryCalls}

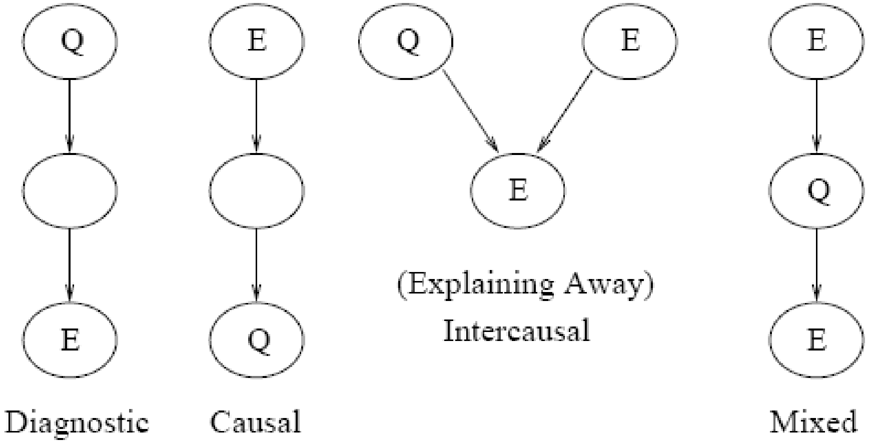


$$\boxed{P(B, E, A, J, M) = P(B)\, P(E)\, P(A|B, E)\, P(J|A)\, P(M|A)}$$

# Inference in Bayesian Networks

Compute the posterior probability distribution for a set of **query variables**, given values for some **evidence variables**

P(Q | E ) = ?



**Diagnostic inferences:** from effect to causes

P(Burglary|JohnCalls)

**Causal Inferences**: from causes to effects

P(JohnCalls|Burglary)

P(MaryCalls|Burglary)

**Intercausal Inferences:**

between causes of a common effect

P(Burglary|Alarm)

P(Burglary|Alarm  Earthquake)

**Mixed Inference**:

combining two or more of above.

P(Alarm|JohnCalls  EarthQuake)

P(Burglary|JohnCalls  EarthQuake)

Picture taken from Ann Nicolson Lecture 2: Introduction to Bayesian Networks

# Bayesian Network Resources

- Repository: www.cs.huji.ac.il/labs/compbio/Repository/
- Softwares:

  for a updated list visit http://www.kdnuggets.com/software/bayesian.html
  - GeNIe: genie.sis.pitt.edu
  - Hugin: www.hugin.com
  - Analytica: www.lumina.com
  - JavaBayes: www.cs.cmu.edu/ javabayes/Home/
  - Bayesware: www.bayesware.com
- Others
  - Bayesian Belief Nets by Russell Greiner http://webdocs.cs.ualberta.ca/~greiner/bn.html
  - List of Bayesian Network software by Kevin Murphy http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html

# Bayesian Networks: Summary

Bayesian Networks: an **efficient** and **effective** representation of the joint probability distribution of a set of random variables

- Efficient:
  - Local models
  - Independence (d-separation)
- Effective:
  Algorithms take advantage of structure to
  - Compute posterior probabilities
  - Compute most probable instantiation
  - Decision making
- But there is more: statistical induction ⮝ LEARNING

adapted from © 1998, Nir Friedman, U.C. Berkeley, and Moises Goldszmidt, SRI International. All rights reserved.

# Bayesian Network Classifiers

## Part III - Learning Bayesian Networks

# Learning Bayesian Networks

Let $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables for a domain under study

**Given:**

✓ a training dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$ of i.i.d examples of $\mathbf{X}$

✓ some prior information $\xi$ (background knowledge)

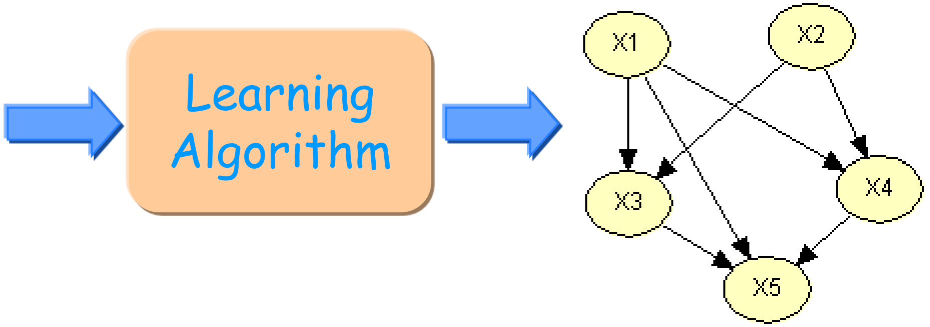
**a BN or fragment of it, prior probabilities, etc.**

**Induce:** a Bayesian Network BN=$(S, \Theta_S)$ that **best matches** $\mathcal{D}$

Data

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| . | . |  |  | . |
| 0 | 1 | 0 | 1 | 0 |

Prior information



Learning Algorithm

# Learning Bayesian Networks

## How many Learning Problems?

We can distinguish a variety of learning problems, depending on whether the structure is *known* or *unknown*, the data is *complete* or *incomplete*, and there are *hidden variables* or not.

| | **Known Structure** | **Unknown Structure** |
|---|---|---|
| **Complete Data** | Statistical parametric estimation (closed-form eq.) | Discrete optimization over structures (discrete search) |
| **Incomplete Data** | Parametric optimization (EM, gradient descent...) | Combined (Structural EM, mixture models…) |

Adapted from Nir Friedman and Moises Goldszmidt slides

# Learning Problem

## Complete Data, Known Structure

| | **Known Structure** | **Unknown Structure** |
|---|---|---|
| **Complete** | Statistical parametric estimation<br>(closed-form eq.) | Discrete optimization over structures<br>(discrete search) |
| **Incomplete** | Parametric optimization<br>(EM, gradient descent...) | Combined<br>(Structural EM, mixture models…) |

↓

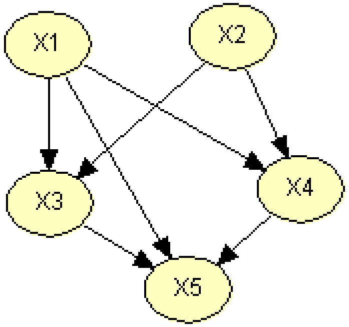
Parameter Learning

# Parameter Learning
## Complete Data

| | Known Structure | Unknown Structure |
|---|---|---|
| Complete data | ● | |
| Incomplete data | | |

Given:

- A Bayesian Network Structure



- Complete Data (no missing values)

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| . | . | | | . |
| 0 | 1 | 0 | 1 | 0 |

Estimate:

- conditional probabilities:

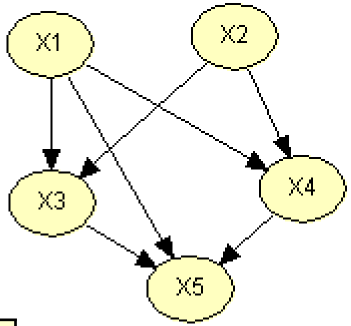
$$P(X_1),\ P(X_2), P(X_3 \mid X_1, X_2), P(X_4 \mid X_1, X_2), P(X_5 \mid X_1, X_3, X_4)$$

# Parameter Learning

## Complete Data

### Conditional Probabilities Tables:

$$P(X_1),\ \ P(X_2), P(X_3 \mid X_1, X_2), P(X_4 \mid X_1, X_2), P(X_5 \mid X_1, X_3, X_4)$$



| $P(X_1)$ | |
|---|---|
| $X_1=1$ | $X_1=0$ |
| ? | ? |

| $P(X_2)$ | |
|---|---|
| $X_2=1$ | $X_2=0$ |
| ? | ? |

| $pa(X_3)$ | | $P(X_3 \mid pa(X_3))$ | |
|---|---|---|---|
| $X_1$ | $X_2$ | $X_3=1$ | $X_3=0$ |
| 1 | 1 | ? | ? |
| 1 | 0 | ? | ? |
| 0 | 1 | ? | ? |
| .0 | 0 | ? | ? |

| $pa(X_4)$ | $P(X_4 \mid pa(X_4))$ | |
|---|---|---|
| $X_1$ | $X_4=1$ | $X_4=0$ |
| 1 | ? | ? |
| 0 | ? | ? |

| $pa(X_5)$ | | | $P(X_5 \mid pa(X_5))$ | |
|---|---|---|---|---|
| $X_1$ | $X_3$ | $X_4$ | $X_5=1$ | $X_5=0$ |
| 1 | 1 | 1 | ? | ? |
| 1 | 1 | 0 | ? | ? |
| 1 | 0 | 1 | ? | ? |
| 1 | 0 | 0 | ? | ? |
| 0 | 1 | 1 | ? | ? |
| 0 | 1 | 0 | ? | ? |
| 0 | 0 | 1 | ? | ? |
| 0 | 0 | 0 | ? | ? |

For each parent configuration we have an independent estimation problem for each local binomial model

For binary variables is enough to fill only one probability

# Parameter Learning
## Complete Data

$\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ - set of random discrete variables

### Statistical Parameter Estimation Problem

How to obtain from data an estimate of each conditional probability?

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_k \mid Pa_i = pa_j) = ?$$

- Estimation relies on **sufficient statistics**
  - count the number of cases in $\mathcal{D}$ such that $X_i = x_k$ and $pa(X_i) = pa_j$

$$N_{ijk} \equiv N(X_i = x_k \mid pa(X_i) = pa_j)$$

  - count the number of cases in $\mathcal{D}$ such that $pa\ (X_i) = pa_j$

$$N_{ij} \equiv N(pa(X_i) = pa_j)$$

# Parameter Learning
## Complete Data

$\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ - set of random discrete variables

$$\boxed{\hat{\theta}_{ijk} = \hat{P}(X_i = x_k \mid Pa_i = pa_j) = ?}$$

- There are two main approaches for parameter estimation:

**frequentist** vs. **Bayesian**

**MLE estimator** **Bayesian estimator**

$$\boxed{\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}}$$

$$\boxed{\hat{\theta}_{ijk} = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}}}$$

Theoretically they can be though of as "*imaginary*" counts from our past experience (priors on counters). In practice they can be though are initial counters

$$N_{ij} = \sum_k N_{ijk} \, ; \, \alpha_{ij} = \sum_k \alpha_{ijk}$$

# Parameter Learning

## Complete Data. Example

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |

| | $pa(X_3)$ | | $P(X_3 \mid pa(X_3))$ | |
|---|---|---|---|---|
| | $X_1$ | $X_2$ | $X_3=1$ | $X_3=0$ |
| $pa_1$ → | 1 | 1 | ? | ? |
| $pa_2$ → | 1 | 0 | ? | ? |
| $pa_3$ → | 0 | 1 | ? | ? |
| $pa_4$ → | 0 | 0 | ? | ? |

**MLE for $P(X_3 = 1 \mid X_1 = 0, X_2 = 1)$**

$$\hat{\theta}_{331} = \frac{4}{6} = 0.66$$

**MLE for $P(X_3 = 0 \mid X_1 = 0, X_2 = 1)$**

$$\hat{\theta}_{330} = 1 - 0.66 = 0.34$$

# Learning Problem
## Incomplete Data, Known Structure

| | Known Structure | Unknown Structure |
|---|---|---|
| Complete | Statistical parametric estimation<br>(closed-form eq.) | Discrete optimization over structures<br>(discrete search) |
| Incomplete | Parametric optimization<br>(EM, gradient descent...) | Combined<br>(Structural EM, mixture models…) |

- Methods for learning: EM and Gradient Ascent

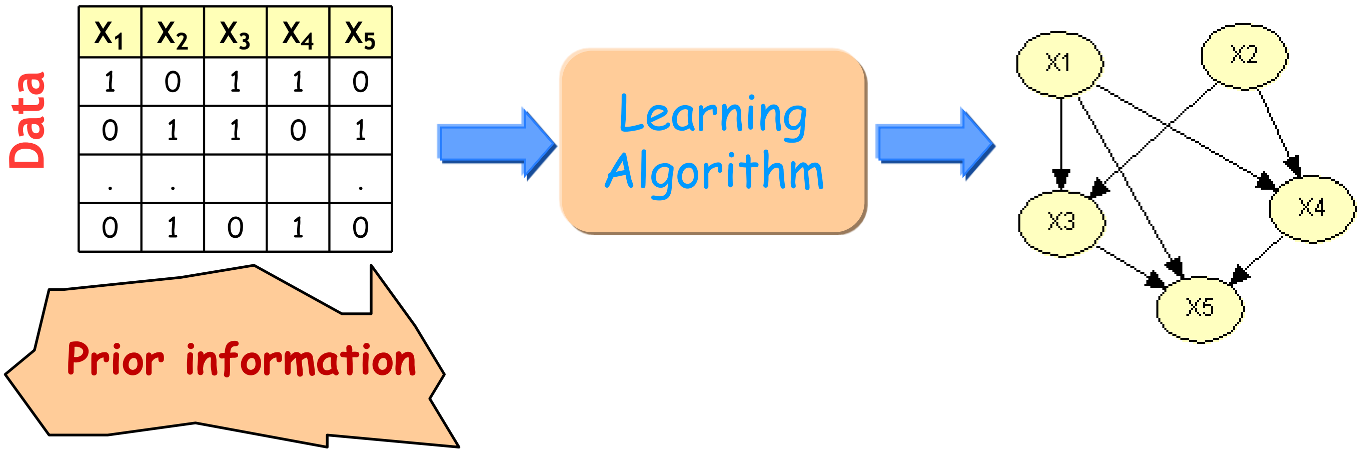
**Difficulties:**

- Exploration of a complex likelihood/posterior
  - More missing data ⇒ many more local maxima
  - Cannot represent posterior ⇒ must resort to approximations

# Learning Problem

## Complete Data, Unknown Structure

| | Known Structure | Unknown Structure |
|---|---|---|
| **Complete** | Statistical parametric estimation<br>(closed-form eq.) | Discrete optimization over structures<br>(discrete search) |
| **Incomplete** | Parametric optimization<br>(EM, gradient descent...) | Combined<br>(Structural EM, mixture models…) |

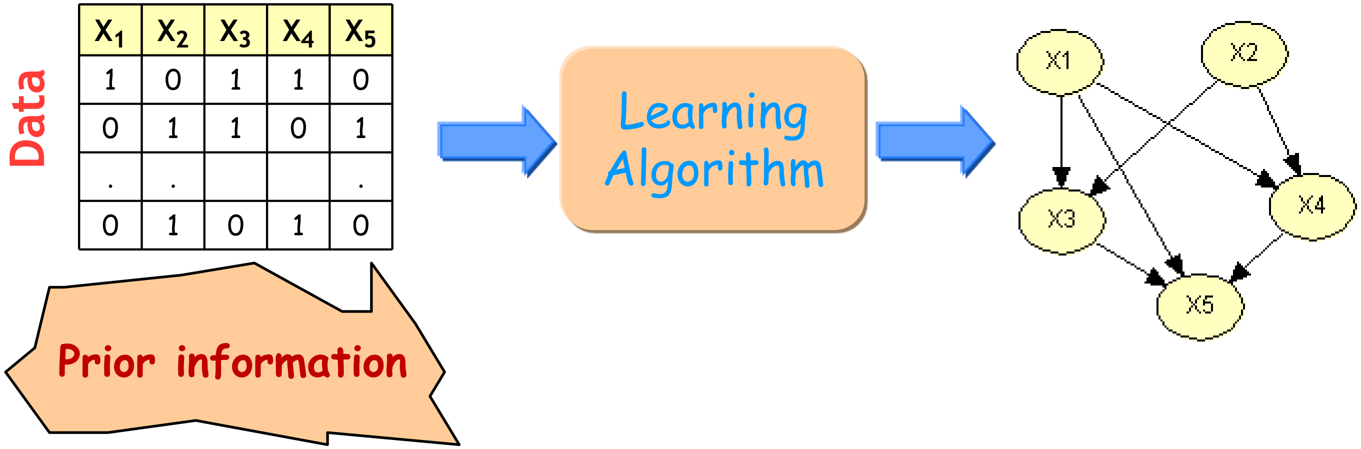


| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| . | . | | | . |
| 0 | 1 | 0 | 1 | 0 |

# Learning Problem

## Incomplete Data, Unknown Structure

| | Known Structure | Unknown Structure |
|---|---|---|
| **Complete** | Statistical parametric estimation<br>(closed-form eq.) | Discrete optimization over structures<br>(discrete search) |
| **Incomplete** | Parametric optimization<br>(EM, gradient descent...) | Combined<br>(Structural EM, mixture models…) |



Data

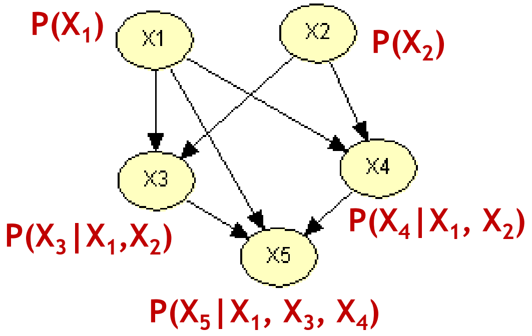
| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| . | . | | | . |
| 0 | 1 | 0 | 1 | 0 |

# Learning Bayesian Networks

- Known structure – learn parameters

➢ Complete data:
parameter estimation (ML, MAP)

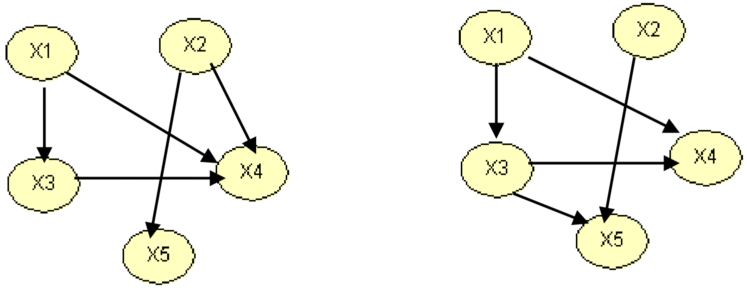
➢ Incomplete data:
non-linear parametric optimization (gradient descent, EM)



- Unknown structure – learn graph and parameters

➢ Complete data:
optimization (search in space of graphs)

➢ Incomplete data:
structural EM, mixture models



$$S = \underset{S}{\textbf{arg max}}\ \textbf{Score}(S)$$

Adapted from Nir Friedman and Moises Goldszmidt slides

# Structure Learning
## Complete Data

### Statistical Model Selection Problem

Given data, to find the structure that best fits the data

**Two approaches:**

- **Constraint based (dependency analysis & search)**
  - Perform tests of conditional independence
    - $\chi^2$ test or mutual information test locally measure the dependency relationships between the variables
  - Search for a network that is consistent with the observed dependencies and independencies
- **Score based (scoring & search)**
  - Define a score that evaluates how well the (in)dependencies in a structure match the observations
  - Search for a structure that maximizes the score

# Structure Learning

## Score based Approaches

- Must choose:

  a ***score(S)*** to measure how well a model fits the data

**Discrete Optimization Problem**: Find the structure that maximizes the **score** in the space S of possible networks

$$\hat{\mathbf{S}} = \arg\max_{S} \mathbf{score(S)}$$

NP-hard problem proved by Chickering et al. (1994)

- **NP-hard optimization problem**:

  we can solve it by using ***heuristic search algorithms***

  - greedy hill climbing, best first search, simulated annealing, etc.

# Structure Learning

## Scores

From a practical point of view, not philosophical we can classify scores into four categories:

1. Log-likelihood-based scores
   - MLC - Maximum Likelihood criterion
2. Penalized log-likelihood scores
   - BIC - Bayesian Information Criterion (Schwarz, 1978)
   - MDL - Minimum Description Length (Rissanen, 1978)
   - AIC - Akaike´s Information Criterion
3. Bayesian scores:  Bayes, BDe, K2
4. Predictive scores
   - Cross-validation (**k-Fold-CV**)
   - Prequential (**Preq**)

# Structure Learning

## Log-likelihood based Scores

$\mathcal{L}(M : \mathcal{D}) \equiv P(\mathcal{D} \mid M)$

- MLC score: derived from the Fisher's *likelihood principle*:

*"a hypothesis is more plausible or likely than another, in the light only of the observed data if it makes those data more probable"*

$$Score_{\mathsf{MLC}}(S, \mathcal{D}) \equiv l(\hat{\Theta}_{ML} : \mathcal{D}, S) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}$$

- Penalized log-likelihood scores: BIC/MDL, AIC

MDL e AIC are derived from information-theoretic arguments

$$Score_{\mathsf{BIC}}(S, \mathcal{D}) \equiv Score_{\mathsf{MLC}}(S, \mathcal{D}) - \frac{1}{2}\, logN \parallel S \parallel$$

*maximize BIC is equivalent to minimize MDL*

$$Score_{\mathsf{AIC}}(S, \mathcal{D}) \equiv Score_{\mathsf{MLC}}(S, \mathcal{D}) - \parallel S \parallel$$

$\parallel S \parallel$ is the dimension of the BN defined as the number of its paramters

# Structure Learning

## Bayesian Scores

Bayes (BD) score: the log of the relative posterior probability

$$\mathit{Score}_{\mathrm{BD}}(S^h, \mathcal{D}) \equiv \log\ P(S^h) + \log\ P(\mathcal{D} \mid S^h)$$

To obtain the Bayesian score we need to asses the **prior distribution** *P*(*S*) for each candidate structure and to compute **the marginal likelihood** *P*(*D* | *S*)

*Cooper et al (1992) and Heckerman et al.(1995) proved that under some nice assumptions the marginal likelihood can be derived in a closed-form solution and it decomposes into a product of terms, one for each local distribution*

$$P(\mathcal{D} \mid S^h) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

*where* $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ *and* $\Gamma(.)$ *is the gamma-function.*

Priors on counters = initial counters

# Structure Learning

## Bayesian Scores

$$\boxed{Score_{\mathrm{BD}}(S^h, \mathcal{D}) \equiv log\ P(S^h) + log\ P(\mathcal{D} \mid S^h)}$$

To compute the Bayes(BD) score we need to set:

1. Priors over structures
   - Assuming uniform priors for all the candidate we obtain the ***log marginal likelihood***
     (one of the most commonly used scores in learning BNs)
2. Priors on parameters:
   different priors leads to special cases of the BD score:
   - **K2** (Cooper and Herskovits, 1992): *log marginal likelihood* with the simple *non-informative* prior $\alpha_{ijk} = 1$
   - **BDe** (Buntine, 1991; Heckerman et al., 1995):
     BD with the additional assumption of *likelihood equivalence*
     - $\alpha_{ijk} = 1/(r_i \cdot q_i)$ , $r_i$ – the number of possible values of $X_i$
       $q_i$ – the number of possible parent configurations of $X_i$

# Structure Learning

## Bayesian Scores

### Bayes (BD) score:

$$\boxed{Score_{\text{BD}}(S^h, \mathcal{D}) \equiv log\ P(S^h) + log\ P(\mathcal{D} \mid S^h)}$$

To obtain the Bayesian score we need to asses the **prior distribution** *P*(*S*) for each candidate structure and to compute **the marginal likelihood** *P*(*D* | *S*)

*Cooper et al.(1992) and Heckerman et al (1995) proved that under some nice assumptions P(D | Sh) can be derived in a closed-form solution and it decomposes into a product of terms, one for each local distribution*

$$\boxed{P(\mathcal{D} \mid S^h) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}}$$

*where* $\alpha_{ij} = \sum_{k=1}^{r_i} \underset{\downarrow}{\alpha_{ijk}}$, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ *and* $\Gamma(.)$ *is the gamma-function.*

Priors on counters = initial counters

# Structure Learning

## Predictive Scores

**Given**: a set $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of candidate BN structures

**Choose**: $S \in$ S so that the predictive distribution yields the most accurate predictions for future data

Predictive scores require a loss function for measuring the predictive accuracy

**unsupervised learning** (BN is used for general purposes)

The training dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$ has $N$ i.i.d examples of **X**:

$$\log Loss(S,\mathcal{D}) = -\sum_{l=1}^{N} \log P(\mathbf{x}^{(l)} \mid S, \theta_S)$$

*The log-loss of S is based on the joint predictive distribution P(**X**)*

$\Rightarrow$ *generative model*

**supervised learning** (BN is used as a classifier)

The training dataset $\mathcal{D} = \{<\mathbf{x}^{(1)}, c^{(1)}>, \ldots, <\mathbf{x}^{(N)}, c^{(N)}>\}$ has $N$ *i.i.d labelled* examples

✓ the *zero-one loss* ← *the most used*

✓ the conditional log-loss

$$C\log Loss(S,\mathcal{D}) = -\sum_{l=1}^{N} \log P(c^{(l)} \mid \mathbf{x}^{(l)}, S, \theta_S)$$

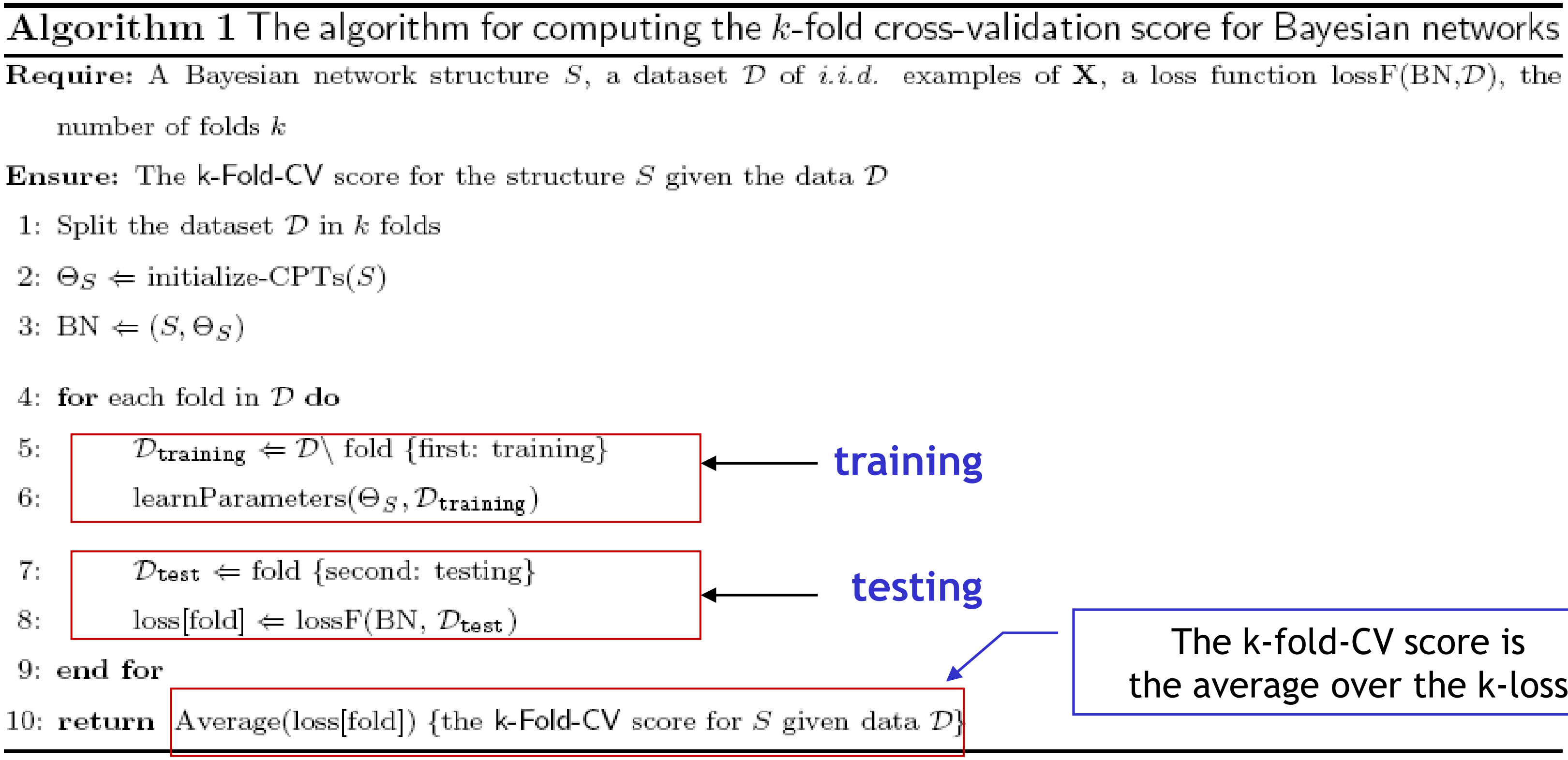
*The log-loss of S is based on the conditional predictive distribution P(**X**|C)*

$\Rightarrow$ *discriminative model*

# Structure Learning

## k-fold Cross-Validation Score

One natural way to measure the predictive performance is provided by **cross-validation** (Stone -1974)



**Algorithm 1** The algorithm for computing the $k$-fold cross-validation score for Bayesian networks

```
Require: A Bayesian network structure $S$, a dataset $\mathcal{D}$ of $i.i.d.$ examples of $\mathbf{X}$, a loss function lossF(BN,$\mathcal{D}$), the
    number of folds $k$
Ensure: The k-Fold-CV score for the structure $S$ given the data $\mathcal{D}$
Split the dataset $\mathcal{D}$ in $k$ folds
$\Theta_S \Leftarrow$ initialize-CPTs($S$)
BN $\Leftarrow (S, \Theta_S)$
for each fold in $\mathcal{D}$ do
    $\mathcal{D}_{\mathtt{training}} \Leftarrow \mathcal{D} \backslash$ fold {first: training}
    learnParameters($\Theta_S, \mathcal{D}_{\mathtt{training}}$)
    $\mathcal{D}_{\mathtt{test}} \Leftarrow$ fold {second: testing}
    loss[fold] $\Leftarrow$ lossF(BN, $\mathcal{D}_{\mathtt{test}}$)
end for
return Average(loss[fold]) {the k-Fold-CV score for $S$ given data $\mathcal{D}$}
```

**training**

**testing**

The k-fold-CV score is the average over the k-loss

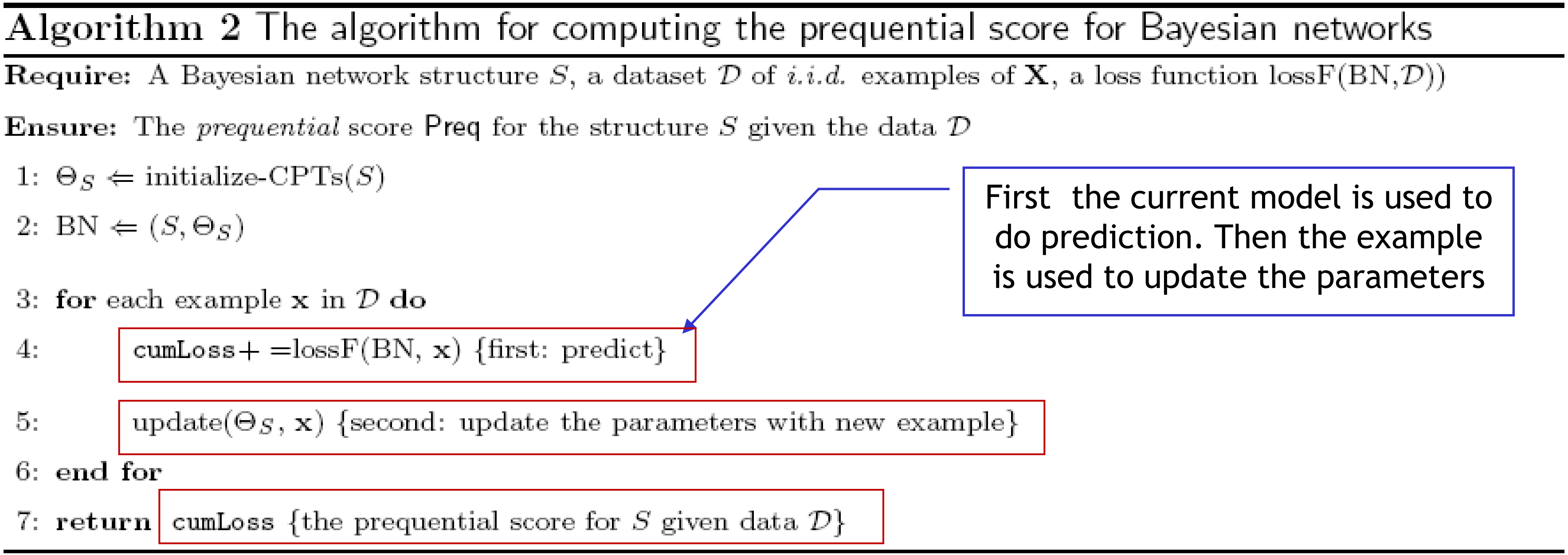
The ***leave-one-out cross-validation score*** *(LOO-CV) is a particular case* when ***k = N***

# Structure Learning

A. P. Dawid, 1984

## Prequential (Cumulative) Score

The prequential score is based on the Dawid's prequential approach



**Algorithm 2** The algorithm for computing the prequential score for Bayesian networks

**Require:** A Bayesian network structure $S$, a dataset $\mathcal{D}$ of *i.i.d.* examples of $\mathbf{X}$, a loss function lossF(BN,$\mathcal{D}$))

**Ensure:** The *prequential* score Preq for the structure $S$ given the data $\mathcal{D}$

```
$\Theta_S \Leftarrow$ initialize-CPTs($S$)
BN $\Leftarrow$ ($S, \Theta_S$)
for each example $\mathbf{x}$ in $\mathcal{D}$ do
    cumLoss+ =lossF(BN, $\mathbf{x}$) {first: predict}
    update($\Theta_S$, $\mathbf{x}$) {second: update the parameters with new example}
end for
return cumLoss {the prequential score for $S$ given data $\mathcal{D}$}
```

First the current model is used to do prediction. Then the example is used to update the parameters

The prequential score is the cumulative loss

# Structure Learning
## Scores. Conclusions

- MDL and Bayesian scores are the most popular in learning BNs

  Asymptotically: MDL is equivalent to MLC and BDe

- Evaluation of prior distributions over structures and parameters
  - Log-likelihood based scores (frequentist approach): do not require
  - Bayesian scores (bayesian approach to model selection): require
- Model Complexity (number of parameters)
  - Likelihood based scores prefer more complex models

    ⇒ can **overfit** the data, specially for small training dataset

  - Penalized likelihood scores (BIC/MDL, AIC) are based on the *Occam's Razor:*

    *"given two equally predictive theories, choose the simpler "*

    ⇒ a more optimal *complexity-fitness* trade-off but can **underfit** for small data

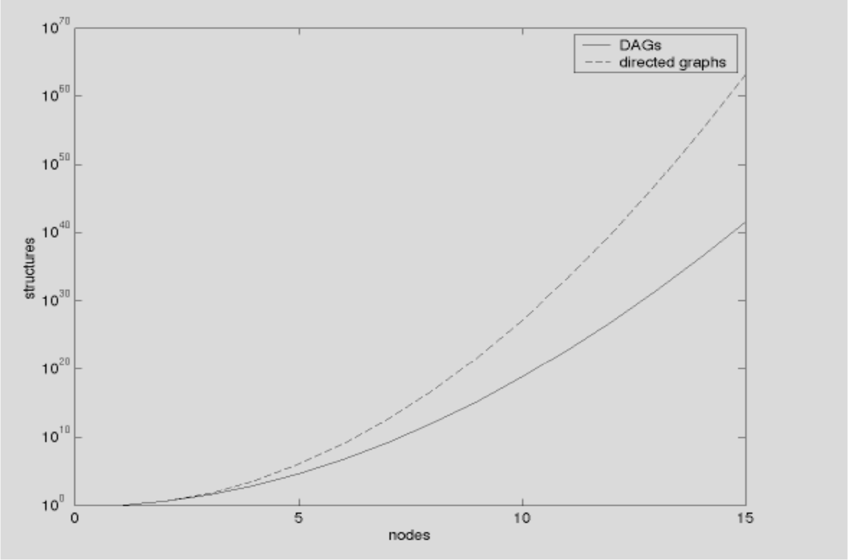
- Arranged according their bias toward simplicity:

  MDL, Bayes, AIC, Preq, CV, MLC, BDe

# Structure Learning
## Search Problem

*a discrete optimization problem*

Problem Optimization:
How to find the structure that maximizes a scoring function?



Naïve Method: brute-force
1. compute the score of every structure
2. pick the one with the highest score

The number of possible structures grows super-exponentially with the number of variables ⇒ exhaustive search is infeasible

The problem of finding the best structure is known to be **NP-hard**
⇒ use **heuristic search algorithms** to traverse the solution space in searching for optimal/near optimal solutions

# Structure Learning
## Heuristic Search Algorithms

We need to define:

- Search space:
  - solution space
  - set of operators
- Initial solution in the solution space
- Search strategy
- Objective Function: a scoring function
- Goal State: usually a stopping criterion is used
  - Ex: stop when the new solution cannot improve the current solution

# Structure Learning

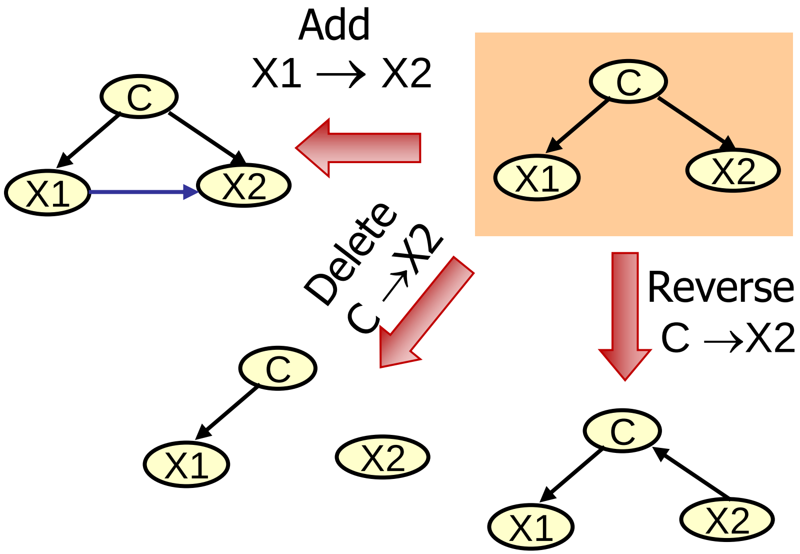
## Search Space

### B-space

the simplest formulation of the search space - (Chickering – 2002)

- Solution Space: the set of all possible solutions
  - individual DAGs
- Set of Operators:

  used by the search algorithm to transform one state to another
  - arc addition: **addArc**
  - arc deletion: **deleteArc**
  - arc reversion: **reverseArc**

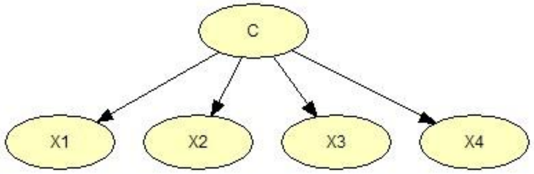


# Structure Learning
## B-space: Initial Solution

- a DAG with no edges
  then iteratively add arcs that most increase
  the score subject to never introducing a cycle

- a complete DAG
  then iteratively delete arcs that most increase
  the score subject to never introducing a cycle

- Some DAG in the middle of the B-space
  ex: a Naive Bayes structure



# Structure Learning
## Search Strategies

Search strategies define how to organize the search in the search space:

- deterministics: all the runs always obtain the same solution
  - Hill-climbing (greedy search) (one of the most used in learning BNs)
  - Tabu search
  - Branch and bound

As a rule, they tend to get stuck in local maximums

- non-deterministics:
  use randomness for escaping from local maximums
  - simulated annealing
  - Genetic algorithms
  - GRASP – Greedy Randomize Adaptive Search procedure

# Structure Learning

## Local score based search

- Local search methods:

  use scores that are decomposed into local scores, one for each node

$$Score(S,D)=\sum_{i=1}^{n} Score_{local}(X_i \mid Pa(X_i), N_{X_i|Pa(X_i)})$$

- change in score that results from the application of an operator can be computed locally
- the global optimization problem is decomposed into local optimization problems
- Local score metrics: **MLC**, **AIC**, **BIC/MDL**, **BD**, **K2**, **BDe** (all are decomposable)
  - They are based on the log likelihood or log marginal likelihood

    both decompose into a sum of terms, one for each node

# Structure Learning
## Global score based search

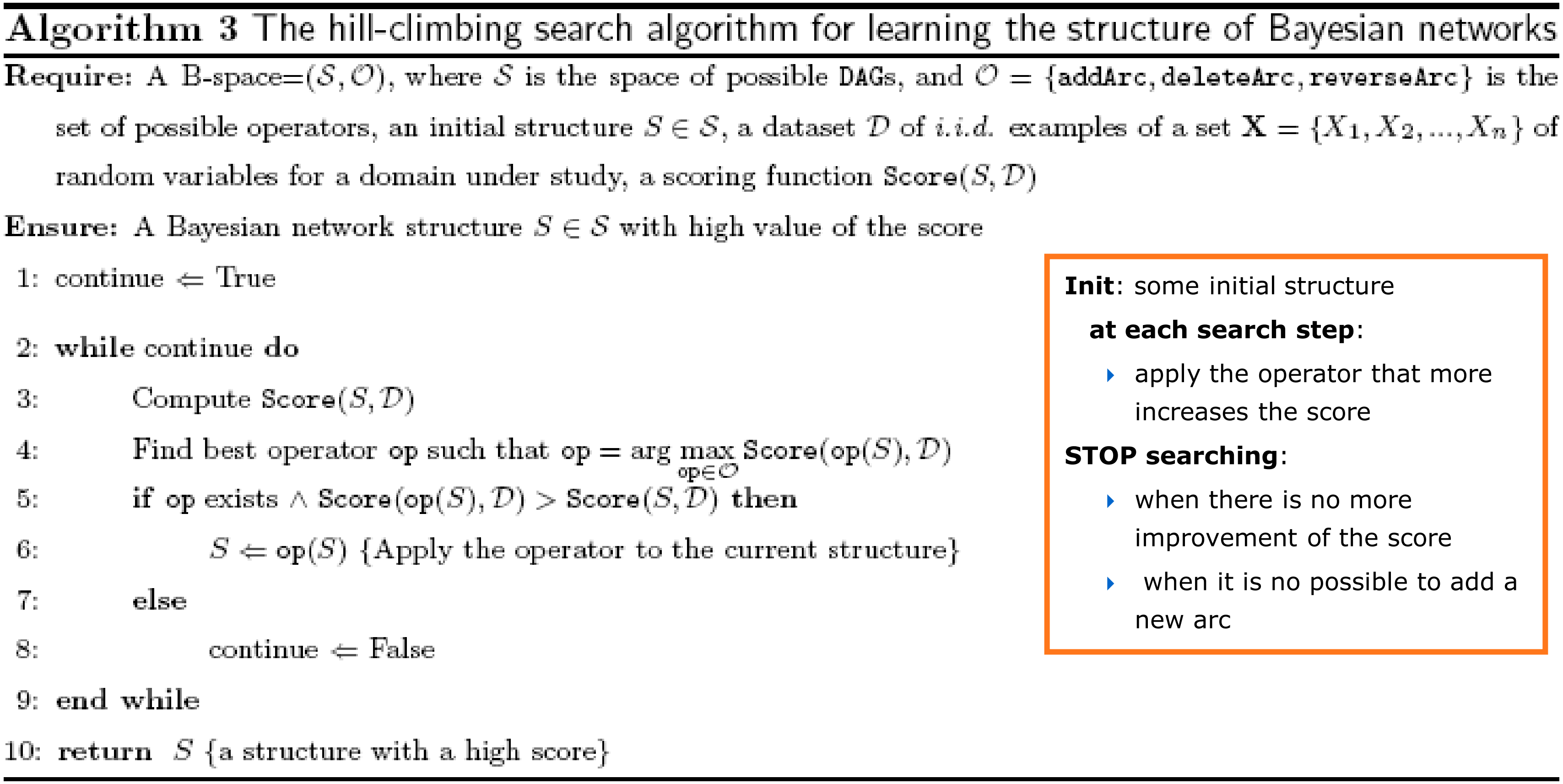
- **Global search methods**:

  use scores that cannot be decomposed into local scores for each node

    - Predictive scores - **k-Fold-CV**, **Prequential**, etc.

  - the whole network needs to be considered to determine the change in the score that results from the application of an operator

# Structure Learning
## Greedy Hill Climbing



**Algorithm 3** The hill-climbing search algorithm for learning the structure of Bayesian networks

**Require:** A B-space=$(\mathcal{S}, \mathcal{O})$, where $\mathcal{S}$ is the space of possible DAGs, and $\mathcal{O} = \{\texttt{addArc}, \texttt{deleteArc}, \texttt{reverseArc}\}$ is the set of possible operators, an initial structure $S \in \mathcal{S}$, a dataset $\mathcal{D}$ of *i.i.d.* examples of a set $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ of random variables for a domain under study, a scoring function $\texttt{Score}(S, \mathcal{D})$

**Ensure:** A Bayesian network structure $S \in \mathcal{S}$ with high value of the score

```
continue ⇐ True
while continue do
    Compute $\texttt{Score}(S, \mathcal{D})$
    Find best operator op such that $\texttt{op} = \arg\max_{\texttt{op} \in \mathcal{O}} \texttt{Score}(\texttt{op}(S), \mathcal{D})$
    if op exists $\wedge$ $\texttt{Score}(\texttt{op}(S), \mathcal{D}) > \texttt{Score}(S, \mathcal{D})$ then
        $S \Leftarrow \texttt{op}(S)$ {Apply the operator to the current structure}
    else
        continue ⇐ False
end while
return $S$ {a structure with a high score}
```

**Init**: some initial structure

**at each search step**:

- apply the operator that more increases the score

**STOP searching**:

- when there is no more improvement of the score
- when it is no possible to add a new arc

# Structure Learning
## Problems with hill-climbing

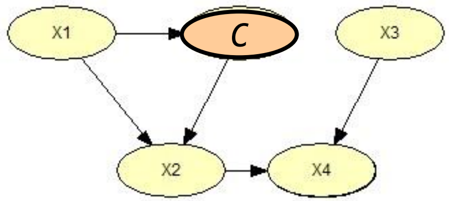
- Local maxima:
  - All one-edge changes reduced the score, but not optimal yet.
- Plateaus:
  - Neighbors have the same score.
- Solutions:
  - Random restart.
  - TABU-search:
    - Keep a list of K most recently visited structures and avoid them.
    - Avoid plateau.
  - Simulated annealing.

# Bayesian Network Classifiers

Part IV - Learning Bayesian Network Classifiers

# Bayesian Network Classifiers

- In classification problems the domain variables are partitioned into:
  - the set of attributes, **X**= $\{X_1,\ldots, X_n\}$
  - the class variable $C$



- A BN can be used as a classifier that gives the posterior probability distribution of the class node $C$ given the values of the attributes X
  - Given an example **x** we can compute the predictive distribution $P(C\,|\mathbf{x}, S)$ by marginalizing the joint distribution $P(C, \mathbf{x} \mid S)$

$$\boxed{P(C\,|\,\mathbf{x},S) = \frac{P(C,\mathbf{x}\,|\,S)}{P(\mathbf{x}\,|\,S)} \alpha\ P(C,\mathbf{x}\,|\,S)}$$

# Bayesian Network Classifiers

## Classification Rule

BNC Classification Rule

$$c^* = h_{BNC}(\mathbf{x}) = arg \max_{j=1...m} P(\mathbf{x}, c_j \mid S, \Theta_s)$$

How to compute $P(\mathbf{x}, c_j)$ for each class $c_j$ ?

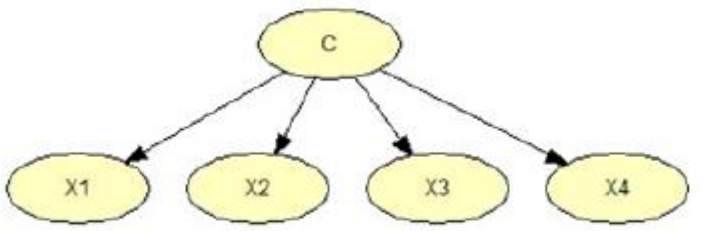
*Assuming complete data (all the attributes values for* **x** *are known)*
*we do not need complicated inference algorithms*
$\Rightarrow$ *just calculate the joint probability distribution for each class*

Given a complete example **x** = $(x_1, x_2, \ldots, x_n)$

$$P(\mathbf{x}, c_j \mid S, \Theta_S) = \prod_{i=1}^{n} P(X_i = x_i \mid pa(X_i)).P(C = c_j \mid pa(C))$$

# Naïve Bayes Bayesian Network

- A NB classifier can be viewed as a Bayesian network with a simple structure that has the class node as the parent node of all other attribute nodes.



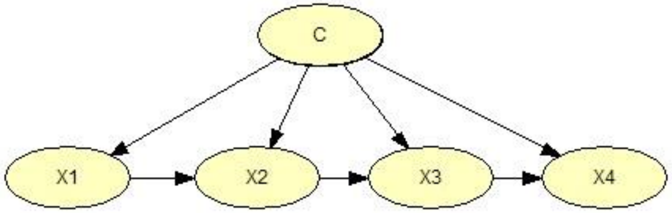
- Priors P($C$) and conditionals P($X_i$|$C$) provide CPTs for the network.

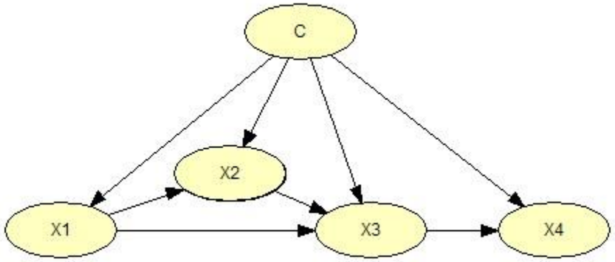
$$P(\mathbf{x}, c_j \mid S, \Theta_S) = \prod_{i=1}^{n} P(X_i = x_i \mid C = c_j).P(C = c_j)$$

*because the structure of the Bayesian Network:*

# Bayesian Network Classifiers
## Restristed vs. Unrestricted Approaches

- **Restricted approach**: contains the NB structure



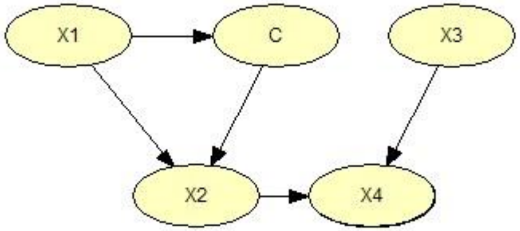


A **T**ree **A**ugmented **N**B (TAN)

A **B**N **A**ugmented **N**B (BAN)

(adding the best set of augmented arcs is an intractable problem )

- **Unrestricted approach:** the class node is treated as an ordinary node



A General **B**N
(GBN) structure

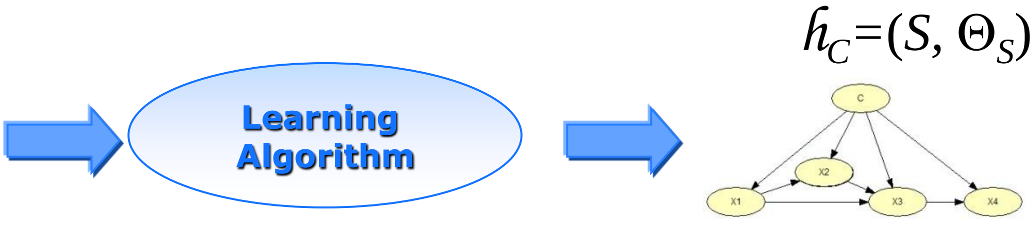
# Bayesian Network Classifiers

## Learning Problem

**Given**: a dataset $\mathcal{D}$ of i.i.d labelled examples
**Induce**: the BNC that best fit the data in some sense

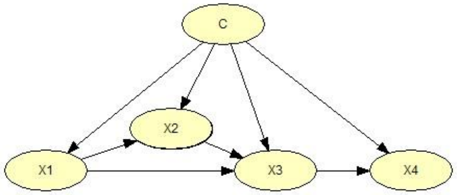
$\mathcal{D}$

| $X_1$ | $X_2$ | ... | $X_n$ | $C$ |
|---|---|---|---|---|
| $x_1^1$ | $x_2^1$ | ... | $x_n^1$ | $c_1$ |
| $x_1^2$ | $x_2^2$ | ... | $x_n^2$ | $c_2$ |
| $x_1^3$ | $x_2^3$ | ... | $x_n^3$ | $c_3$ |
| $x_1^4$ | $x_2^4$ | ... | $x_n^4$ | $c_4$ |



**How to solve?**

1. Choose a suitable class (class-model) of BNCs
   - For example a BAN can be chosen ⇒ the solution space is restricted
2. Choose a structure within this class-model that best fit the given data

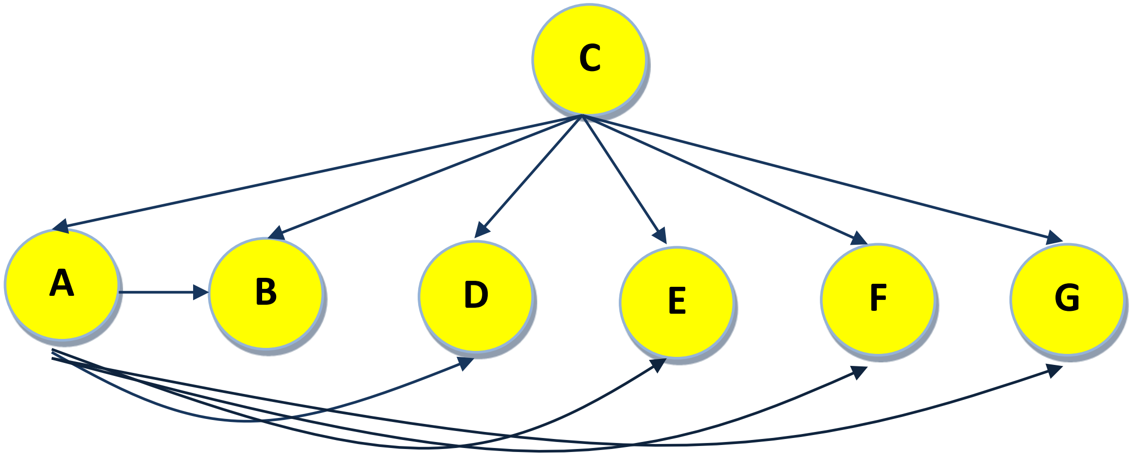


3. Estimate the parameters for the chosen structure from data

# A Tree Augmented NB (TAN)

Friedman, Geiger and Goldszmith, 1997

- The attribute independence assumption is relaxed
  ⇒ less restrictive than Naive Bayes
- impose acceptable restriction: each attribute has as parents the class variable and at most one other attribute



polynomial time bound on constructing network
O((# attributes)$^2$ * |training set|)

Friedman, Geiger and Goldszmith, 1997

# TAN Learning Algorithm
## (dependency analysis & search)

1. Compute the conditional mutual information, $I(X_i, X_j \mid C)$, between each pair of attributes, $i \neq j$
2. Build a complete undirected graph between all the attributes (excluding the class variable). Arc weight is the conditional mutual information between the vertices
3. Find maximum weight spanning tree over the graph
    1. Mark the two arcs with biggest weights and add to the tree
    2. Repeat until n-1 arc were added to the tree:
        - find the arc with biggest weight and add to the tree if not lead to a cycle
4. Transform the undirected tree to directed by picking a root in tree and making all arcs directed (to be outward from the root)
5. Construct a TAN model by adding a node labeled C and adding an arc from each attribute node to C

# Mutual Information

*Mutual Information:* measures the mutual dependence of two variables. From the information theory point of view it measures the amount of information that can be obtained about one random variable by observing another.

$$I\left(X_i, X_j\right) = \sum_{i=1}^{r_x} \sum_{j=1}^{r_y} p\left(x_i, x_j\right) \log \frac{p\left(x_i, x_j\right)}{p\left(x_i\right) p\left(x_j\right)}$$

*Conditional mutual information:* measures the mutual information of two random variables conditioned on a third.

$$I\left(X_i, X_j \mid C\right) = \sum_{C} p(c) I\left(X_i, X_j \mid C = c\right) = \sum_{i=1}^{r_x} \sum_{j=1}^{r_y} \sum_{k=1}^{r_c} p\left(x_i, x_j, c_k\right) \log \frac{p\left(x_i, x_j \mid c_k\right)}{p\left(x_i \mid c_k\right) p\left(x_j \mid c_k\right)}$$

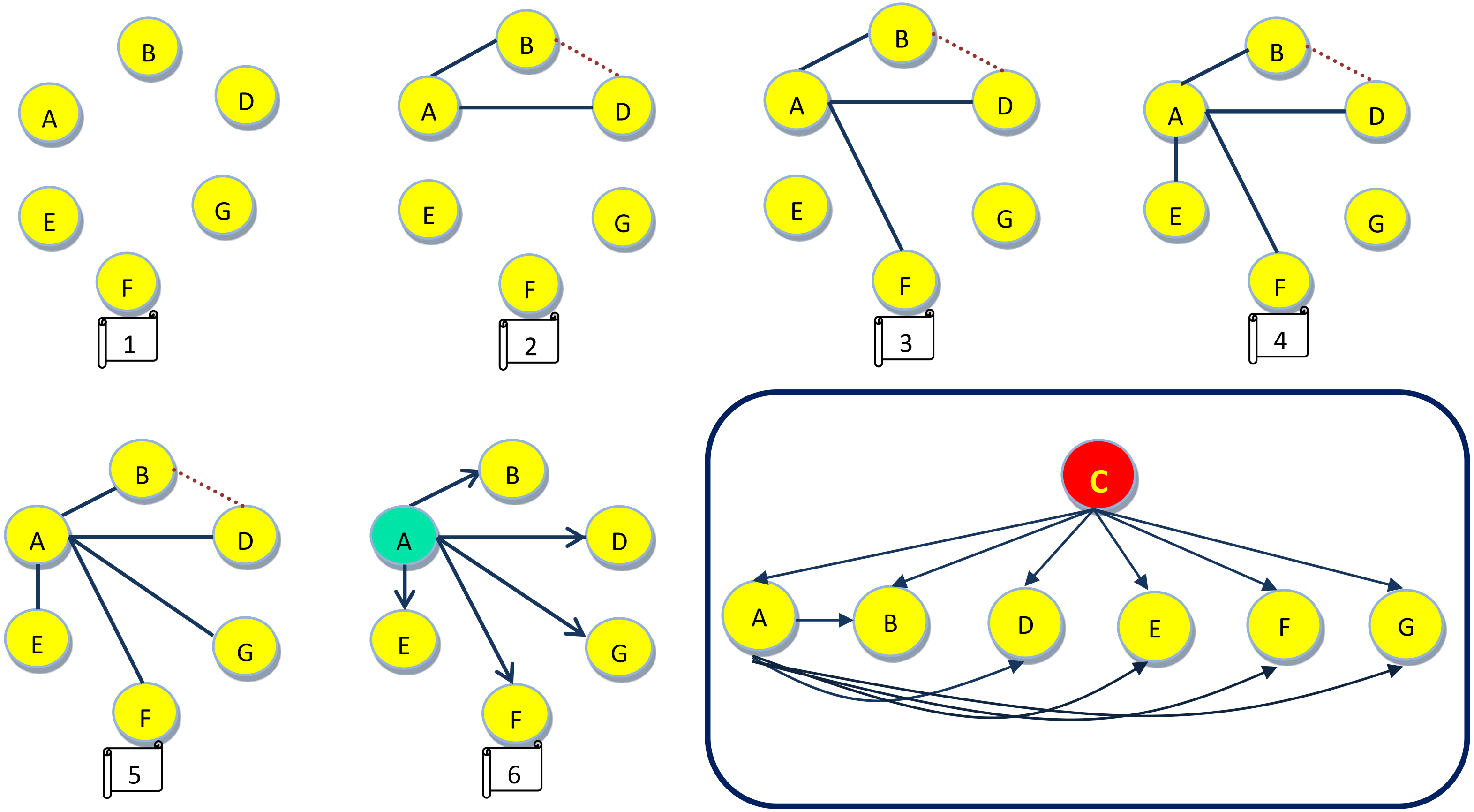
# TAN Learning Algorithm

## An example

6 attributes: A,B,D,E,F,G

$$I(A,B|C)>I(A,D|C)>I(B,D|C)>I(A,F|C)>I(A,E|C)>I(A,G|C)>I(B,E|C)>I(B,F|C)>I(B,G|C)>$$

$$I(D,E|C)>I(D,F|C)>I(D,G|C)>I(E,F|C)>I(E,G|C)>I(F,G|C)$$

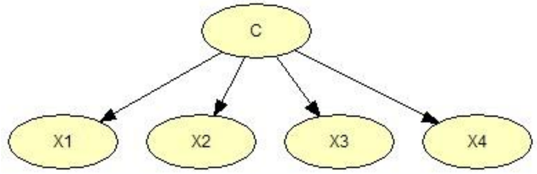


# k-Dependence Bayesian Classifiers (k-DBCs)

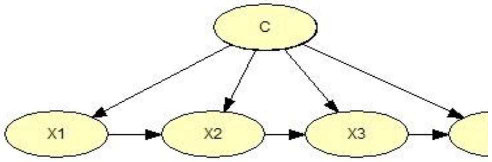
Sahami, M., 1996

k-DBCs represent an unified framework for all the BNCs class-models containing the structure of the Naïve Bayes

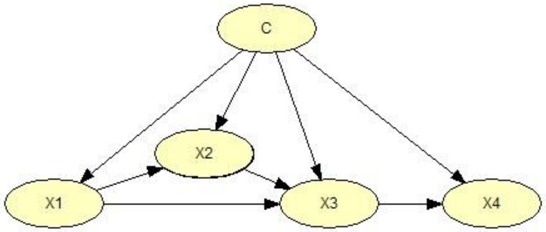
- Definition: A k-DBC is a Bayesian network which:
  - contains the structure of NB
  - allows each attribute $X_i$ to have a maximum of $k$ attribute nodes as parents



NB is a 0-DBC



TAN is a 1-DBC



This BAN is a 2-DBC

Model’s complexity increases

*By varying the value of k we can obtain classifiers that smoothly move along the spectrum of attribute dependences*

We can control the complexity of k-DBCs by choosing an appropriate ***k*** value

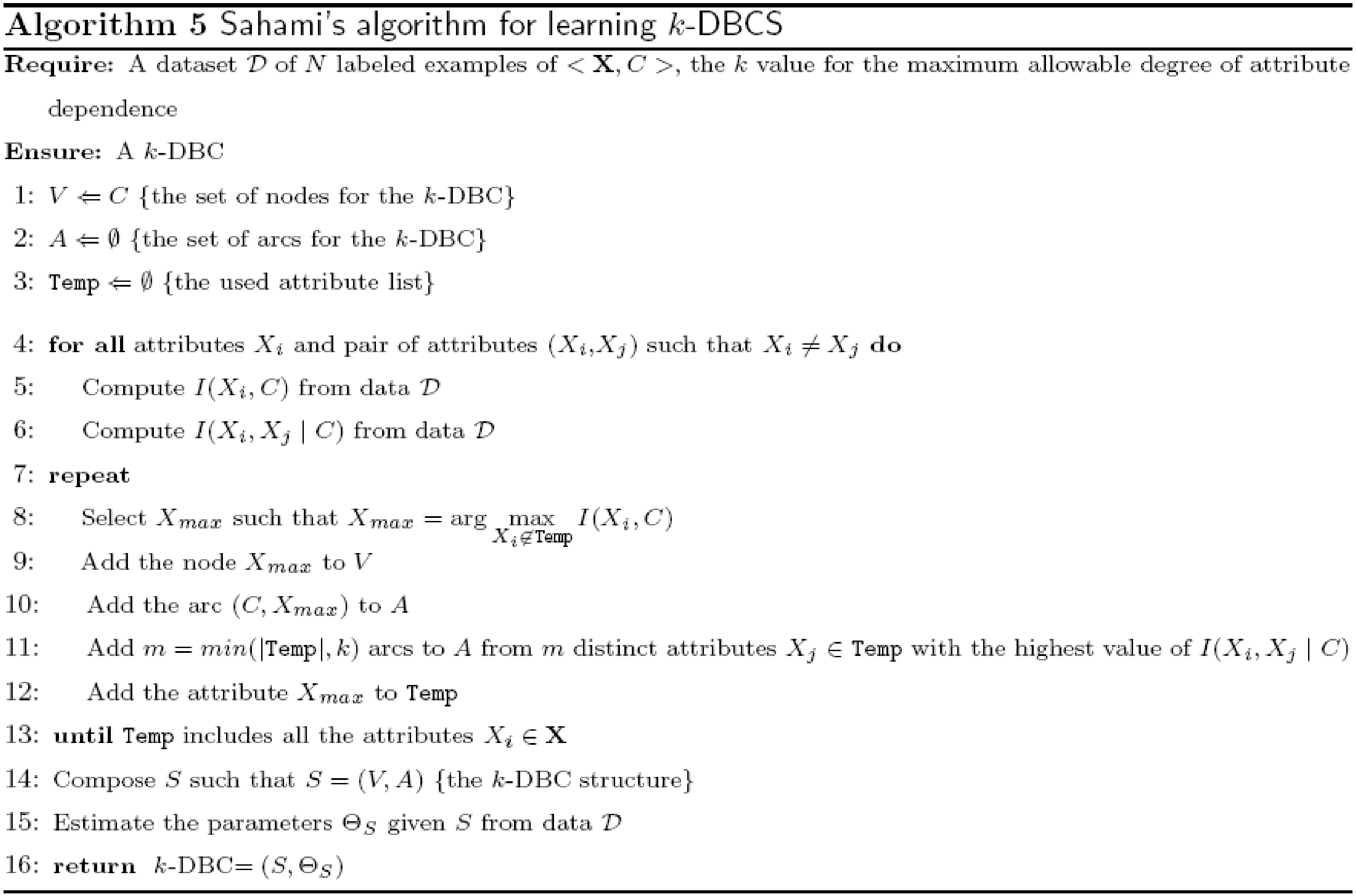
# Sahami's Learning Algorithm
## (dependency analysis & search)

- The algorithm can be viewed as a generalization of the TAN algorithm
- Mutual information is used as a measure of degree of attribute dependence

- Rationale:
  1. Starting with a *k*-DBC's structure *S* with a single class node *C*, the algorithm iteratively add *m* = *min*(|*S*|, *k*) parents to each new attribute added to *S* with largest dependence with the class *C*.
  2. The *m* parents for each new attribute are selected among those with higher degree of dependence given the class.
  3. The process finishes when all the attributes have been added to the structure *S*.

# Sahami's Learning Algorithm

## (dependency analysis & search)



**Algorithm 5** Sahami's algorithm for learning $k$-DBCS

```
Require: A dataset $\mathcal{D}$ of $N$ labeled examples of $<\mathbf{X}, C>$, the $k$ value for the maximum allowable degree of attribute dependence
Ensure: A $k$-DBC
$V \Leftarrow C$ {the set of nodes for the $k$-DBC}
$A \Leftarrow \emptyset$ {the set of arcs for the $k$-DBC}
Temp $\Leftarrow \emptyset$ {the used attribute list}

for all attributes $X_i$ and pair of attributes ($X_i$,$X_j$) such that $X_i \neq X_j$ do
    Compute $I(X_i, C)$ from data $\mathcal{D}$
    Compute $I(X_i, X_j \mid C)$ from data $\mathcal{D}$
repeat
    Select $X_{max}$ such that $X_{max} = \arg \max_{X_i \notin \texttt{Temp}} I(X_i, C)$
    Add the node $X_{max}$ to $V$
    Add the arc $(C, X_{max})$ to $A$
    Add $m = min(|\texttt{Temp}|, k)$ arcs to $A$ from $m$ distinct attributes $X_j \in$ Temp with the highest value of $I(X_i, X_j \mid C)$
    Add the attribute $X_{max}$ to Temp
until Temp includes all the attributes $X_i \in \mathbf{X}$
Compose $S$ such that $S = (V, A)$ {the $k$-DBC structure}
Estimate the parameters $\Theta_S$ given $S$ from data $\mathcal{D}$
return $k$-DBC$= (S, \Theta_S)$
```

# Learning BNCs augmented NB
# Hill Climbing using only arc addition

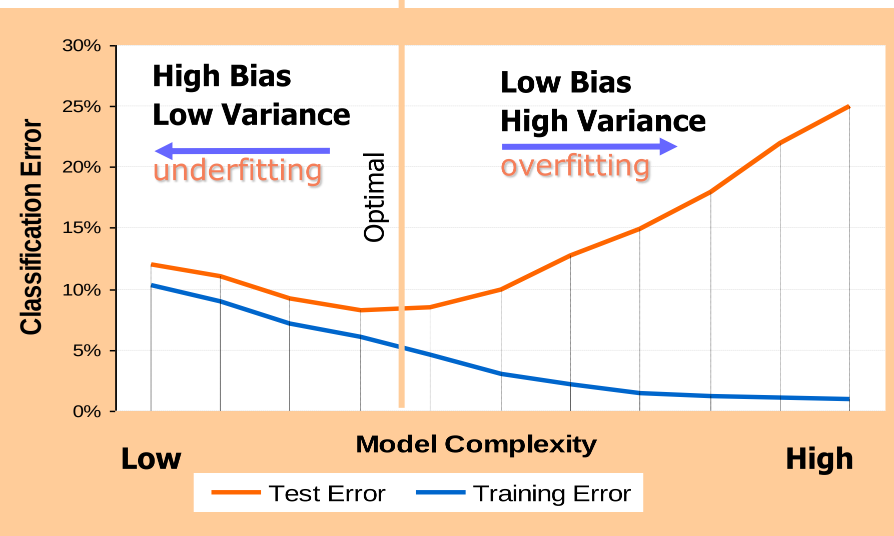
**(scoring & search)**

```
procedure Learn_BNC(data, maxNrOfParents, score){
Init:   BNC ← Learn_NaiveBayes(data)
        bestScore ← calcScore(BNC, score)
repeat   // go do the search
    bestArc ← findBestArcToAdd(BNC, data, maxNrOfParents)
    newScore ← calcScore(BNC,score);
    if (bestArc != null) && (newScore > bestScore)
       BNC ← AddArc (BNC, bestArc)
       bestScore ←  newScore
      else
        return(BNC)
}
```

# Choosing the Class-Model

Learning algorithms should be able to select a model with the appropriate complexity for the available data

Behavior of test error and training error varying model complexity



there is an optimal model that gives minimum test error

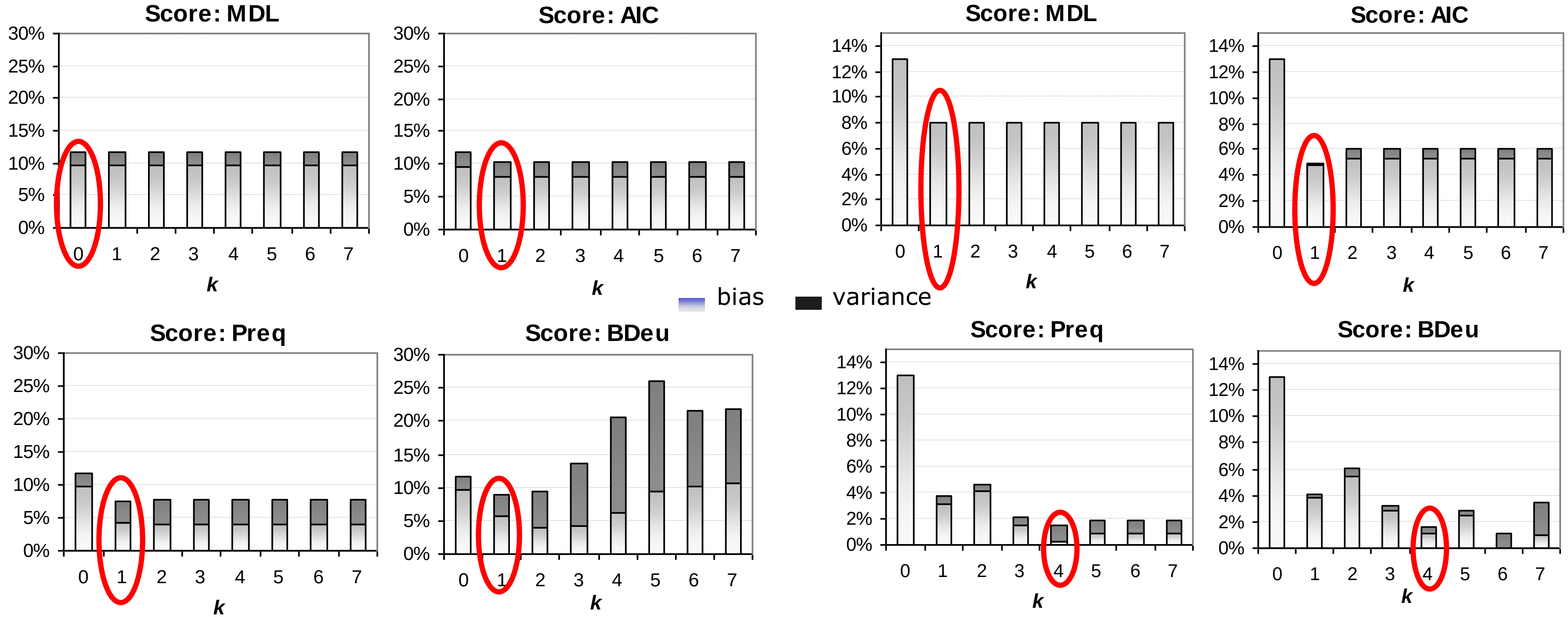
- Classifier too simple (e.g. NB)
  - underfit the data, too much bias
- Classifier too complex
  - overfit the data, too much variance
- Controling the BNCs complexity
  - BNC class-models: NB, TAN, BAN, etc., differ by **the number of parents allowed** for attribute
  - **choosing of the appropriate class-model** of BNCs ⇒ **not trivial**: depend on the chosen score and available data

Results from an empirical study conducted with the class of k-DBCs and the **nursery dataset from the UCI repository**

# What k to choose?

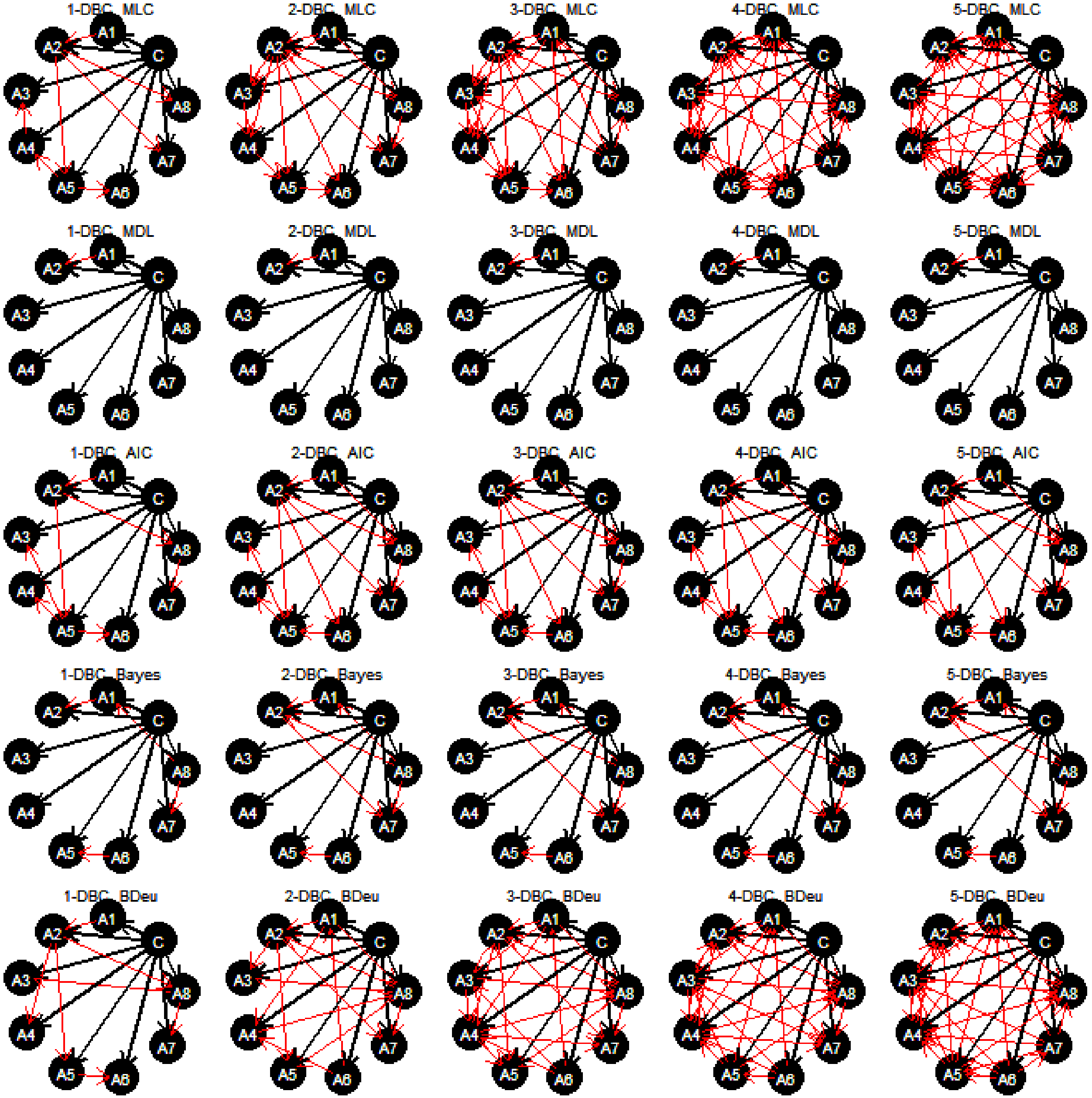
Varying *k*, the **score** and the **training set size** can have different effects on bias and variance and consequently in the test error

**1000 training examples** **12500 training examples**



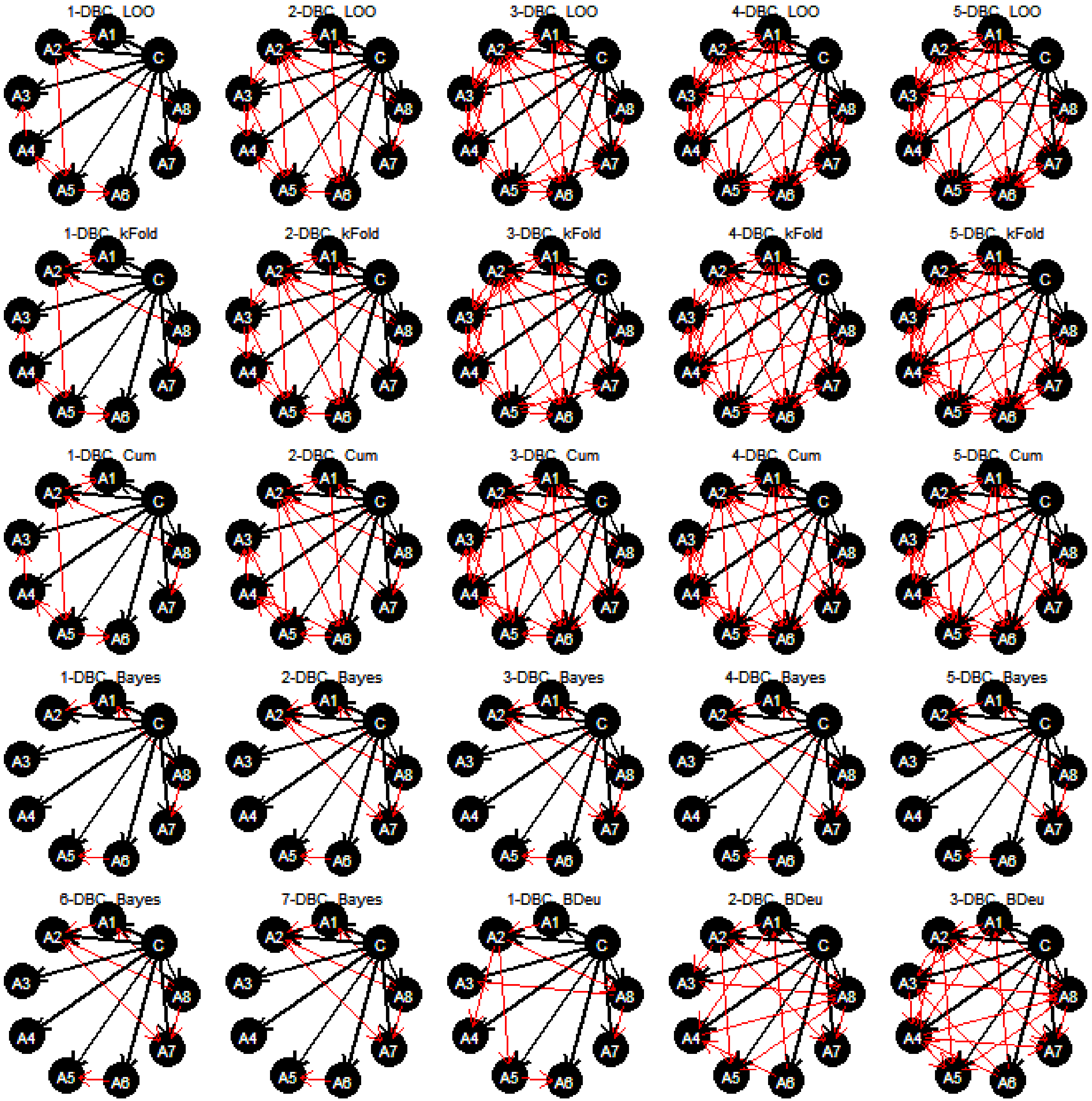
If the data increases, it makes sense to **gradually increase** the *k* value to adjust the complexity of the class-model, and hence, the complexity of BNCs to the available data k-DBCs learned from the Nursery dataset using the hill-climbing learning algorithm with five unsupervised scores:

- MLC
- MDL
- AIC
- Bayes
- BDe

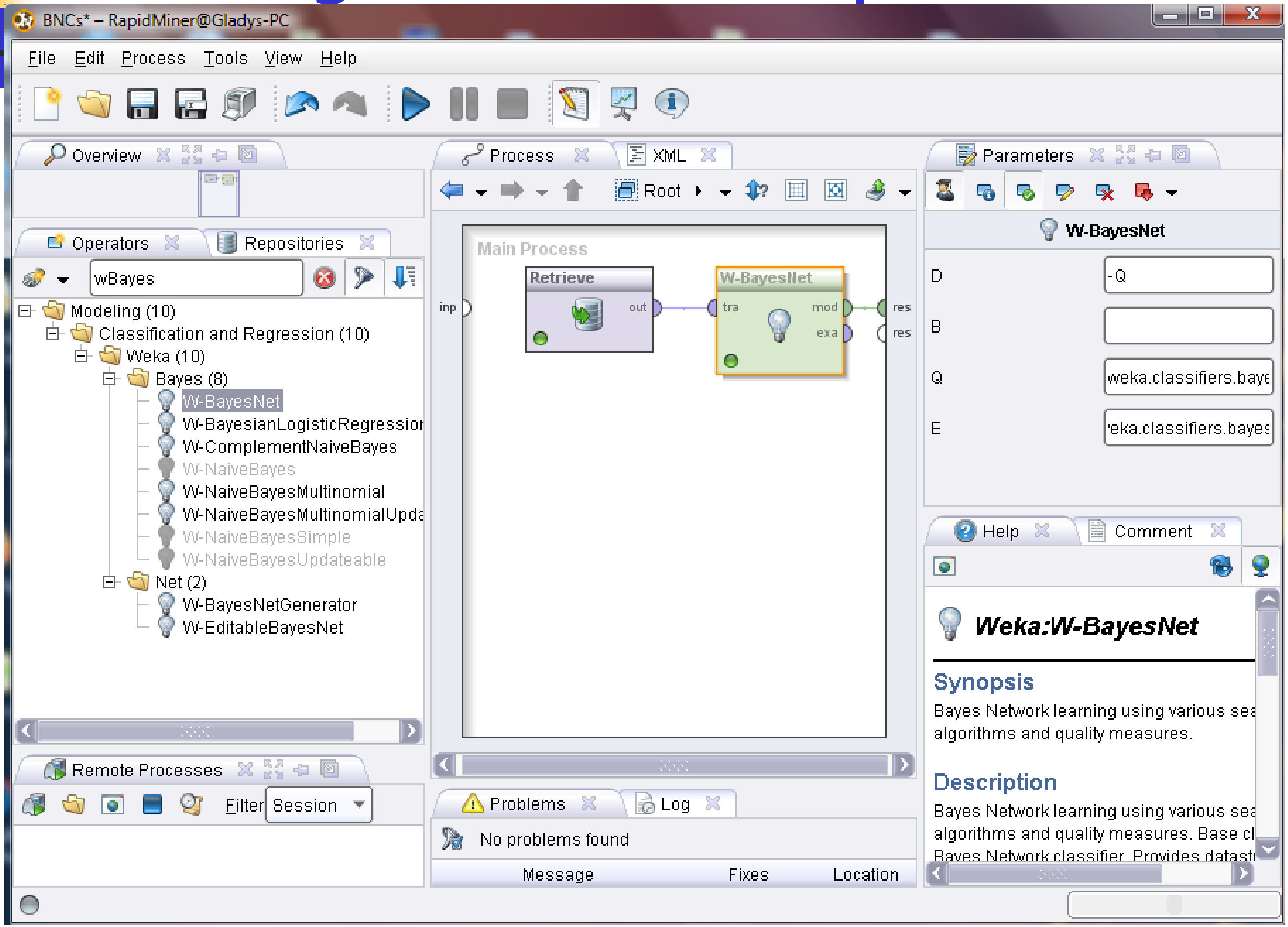


k-DBCs learned from the Nursery dataset using the hill-climbing learning algorithm with three supervised scores:

- LOO-CV
- k-Fold-CV
- Cumulative



# Learning BNC with RapidMiner



wBayes

- Modeling (10)
  - Classification and Regression (10)
    - Weka (10)
      - Bayes (8)
        - W-BayesNet
        - W-BayesianLogisticRegressior
        - W-ComplementNaiveBayes
        - W-NaiveBayes
        - W-NaiveBayesMultinomial
        - W-NaiveBayesMultinomialUpda
        - W-NaiveBayesSimple
        - W-NaiveBayesUpdateable
      - Net (2)
        - W-BayesNetGenerator
        - W-EditableBayesNet

W-BayesNet

| | |
|---|---|
| D | -Q |
| B | |
| Q | weka.classifiers.baye |
| E | 'eka.classifiers.bayes |

***Weka:W-BayesNet***

**Synopsis**

Bayes Network learning using various sea algorithms and quality measures.

**Description**

Bayes Network learning using various sea algorithms and quality measures. Base cl Bayes Network classifier. Provides datastı

No problems found

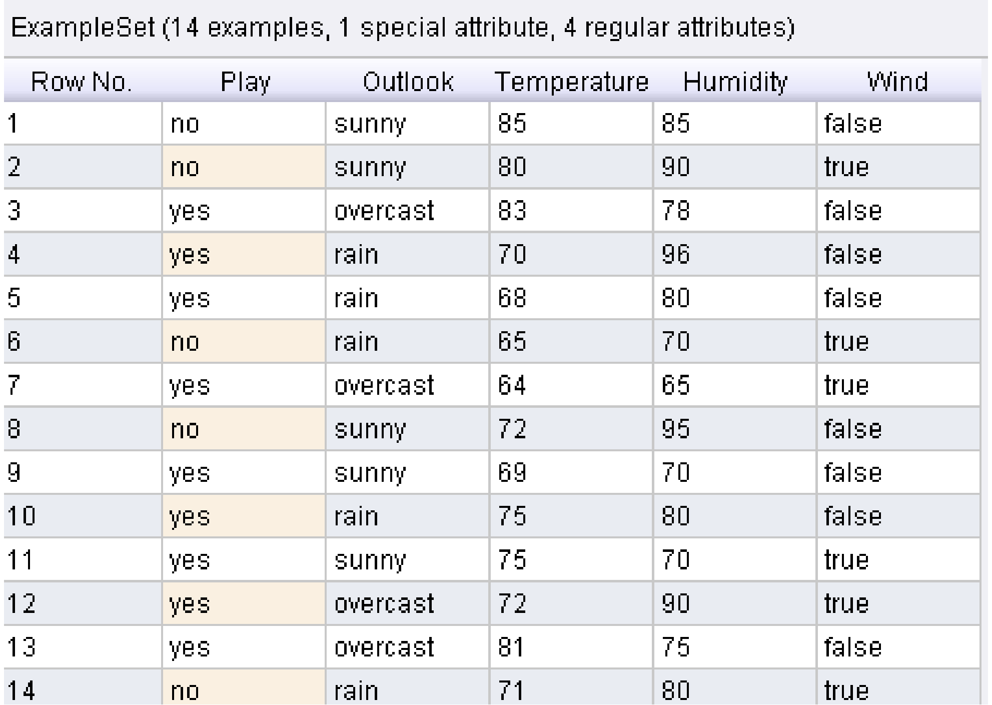
# Building a TAN classifier using the Weka class W-BayesNet

## Parameter Settings:

Q = weka.classifiers.bayes.net.search.local.TAN (structure learning algorithm)

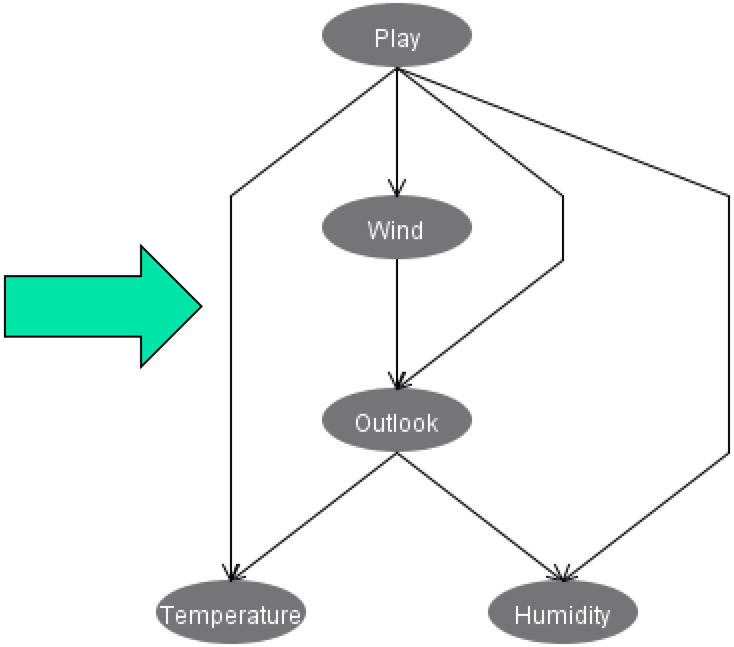
E = weka.classifiers.bayes.net.estimate.SimpleEstimator (parameter learning algorithm)

## Golf Dataset



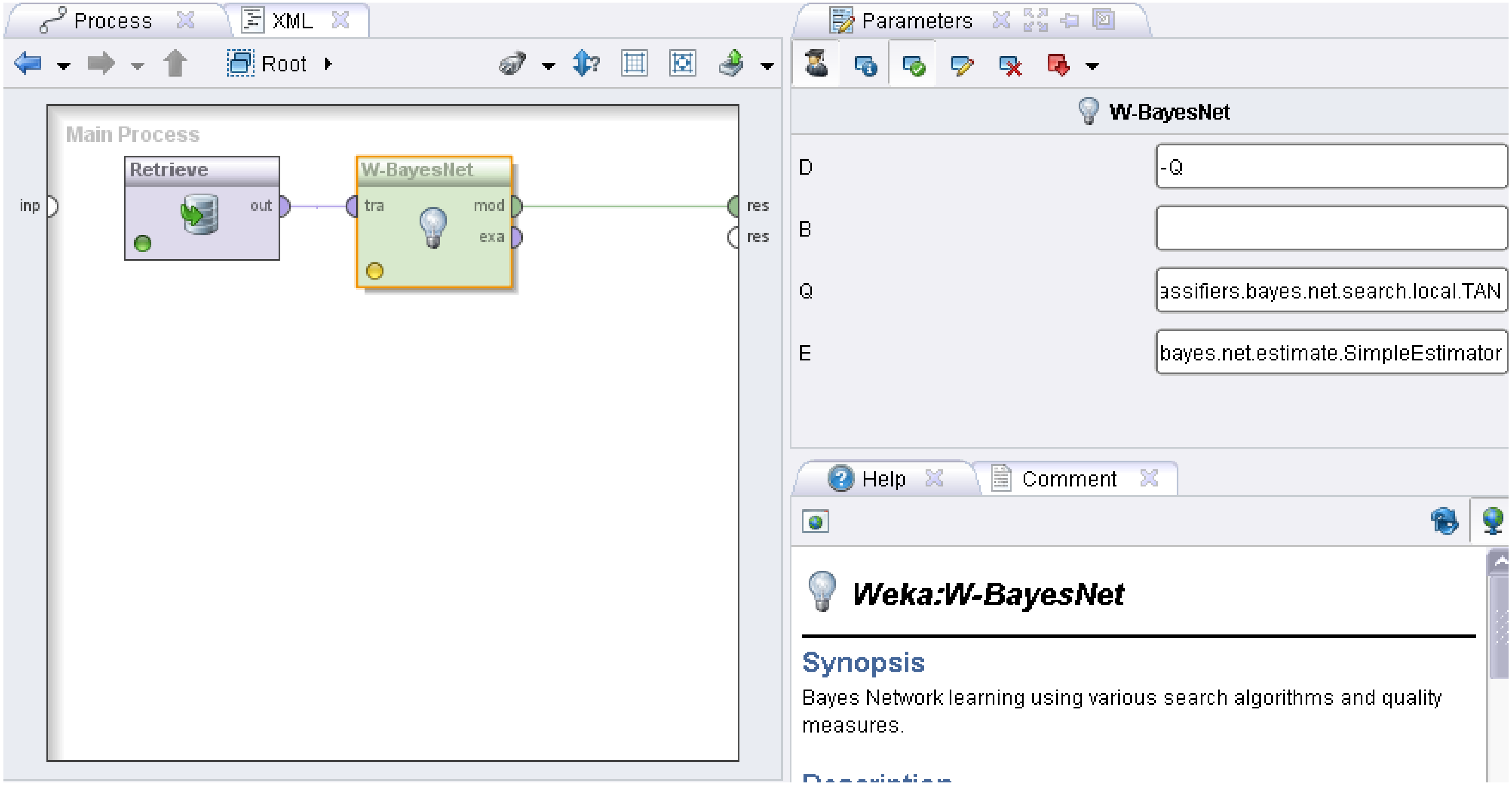
ExampleSet (14 examples, 1 special attribute, 4 regular attributes)

| Row No. | Play | Outlook | Temperature | Humidity | Wind |
|---|---|---|---|---|---|
| 1 | no | sunny | 85 | 85 | false |
| 2 | no | sunny | 80 | 90 | true |
| 3 | yes | overcast | 83 | 78 | false |
| 4 | yes | rain | 70 | 96 | false |
| 5 | yes | rain | 68 | 80 | false |
| 6 | no | rain | 65 | 70 | true |
| 7 | yes | overcast | 64 | 65 | true |
| 8 | no | sunny | 72 | 95 | false |
| 9 | yes | sunny | 69 | 70 | false |
| 10 | yes | rain | 75 | 80 | false |
| 11 | yes | sunny | 75 | 70 | true |
| 12 | yes | overcast | 72 | 90 | true |
| 13 | yes | overcast | 81 | 75 | false |
| 14 | no | rain | 71 | 80 | true |



# Learning BNCs using the Weka class weka/classifiers/bayes/BayesNet

Several learning algorithms for BNCs are implemented in Weka
(see full documentation at http://www.cs.waikato.ac.nz/~remco/weka.bn.pdf



W-BayesNet

D: -Q

B:

Q: assifiers.bayes.net.search.local.TAN

E: bayes.net.estimate.SimpleEstimator

*Weka:W-BayesNet*

Synopsis

Bayes Network learning using various search algorithms and quality measures.

# References

1. Buntine, W. (1996) A guide to the literature on learning probabilistic networks from data, IEEE Transactions on Knowledge and Data Engineering 8, 195–210
2. Bouckaert, R. (2007) Bayesian Network Classifiers in Weka, Technical Report
3. Castillo, G. (2006) Adaptive Learning Algorithms for Bayesian Network Classifiers, PhD Thesis, Chapter 3.
4. Cheng, J. and Greiner, R. (1999) Comparing Bayesian Network Classifiers, Proceedings of UAI
5. Chickering, D.M. (2002) Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*
6. Friedman, N., Geiger, D. and Goldszmidt M. (1997) Bayesian network classifiers, Machine Learning 29, 139–164.
7. Heckerman D. (1995) A tutorial on learning with Bayesian Networks. Technical Report. Microsoft Research
8. Mitchell T. (1997) Machine Learning, Chapter 6. Bayesian Learning, Tom Mitchell, McGrau Hill

## Presentation Slides

- **Learning Bayesian Networks from Data** by Nir Friedman and Daphne Koller
  - Powerpoint Presentation
  - Document in PDF (1.72MB)
- **Learning Bayesian Networks from Data by** Nir Friedman and Moises Goldszmidt
  - Powerpoint Presentation Show (1.4MB), PDF (15MB) (6 x pages)
- **Learning Bayesian Networks** by Andrew Moore
- **Aprendizagem Bayesiana** by João Gama, Universidade do Porto
- **Clasificadores Bayesianos** by Abdelmalik Moujahid, Iñaki Inza e Pedro Larrañaga. Universidad del País Vasco