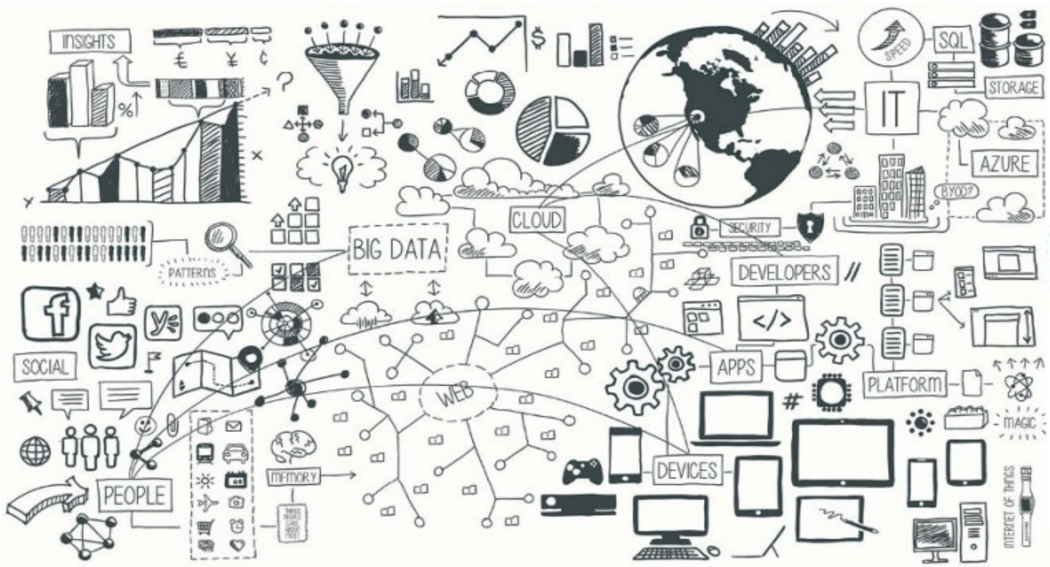
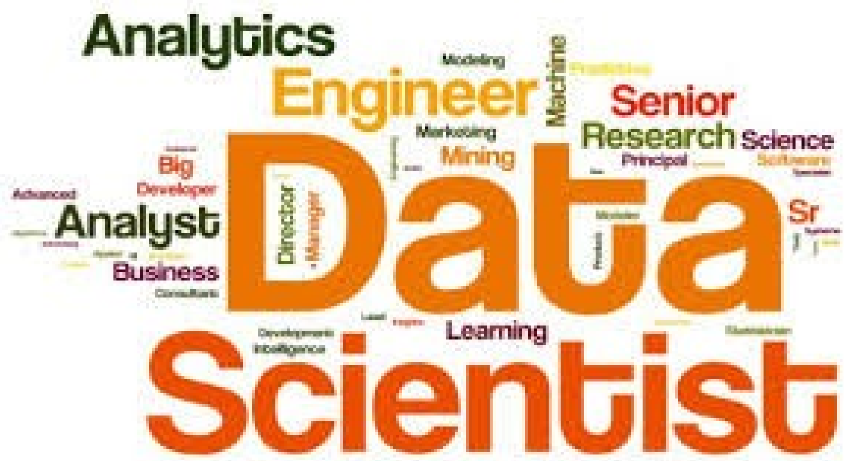
M1970 – Machine Learning II
Redes Probabilísticas Discretas (Inferencia)





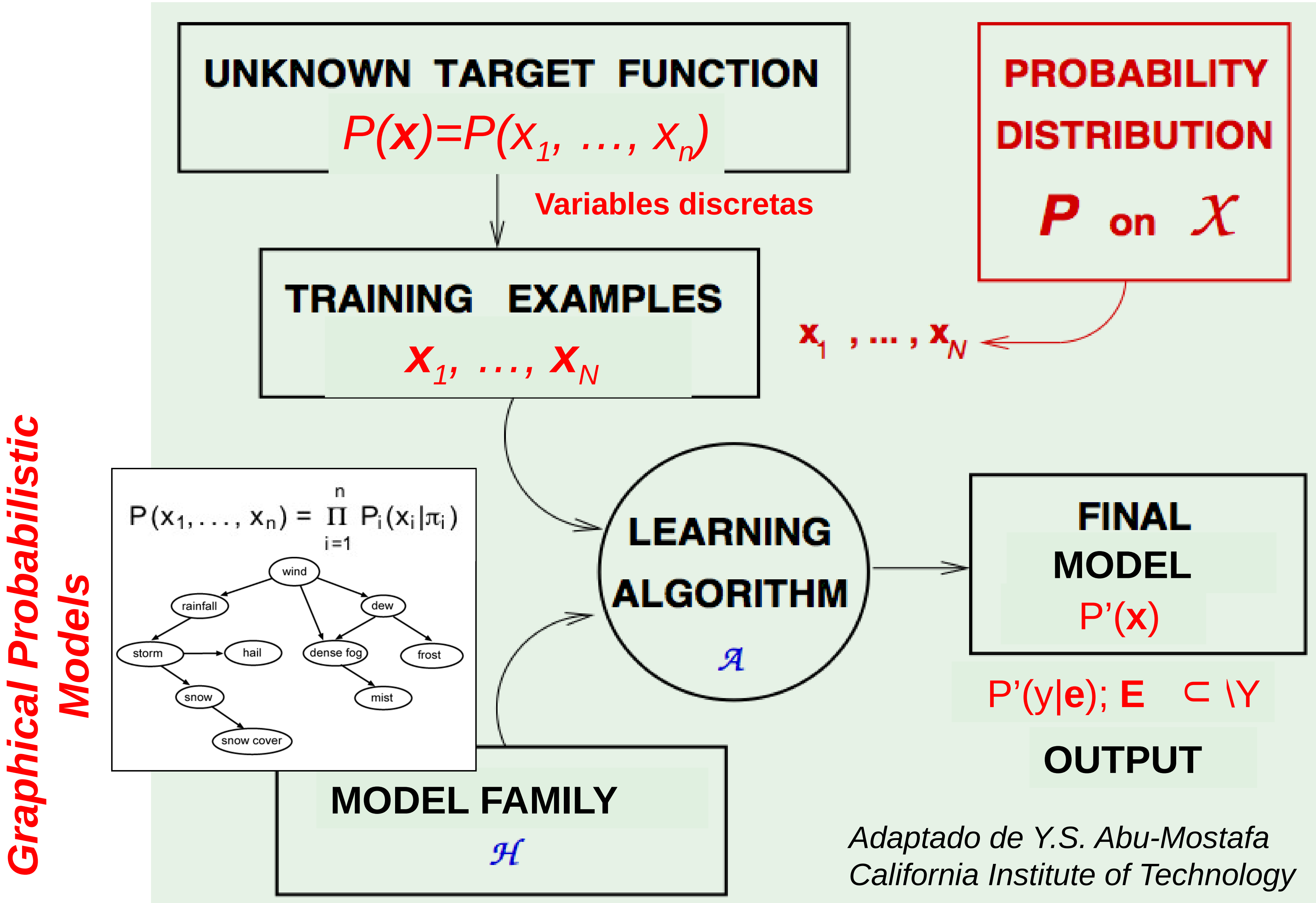
Sixto Herrera (sixto.herrera@unican.es)
José M. Gutiérrez, Mikel Legasa

Grupo de Meteorología

Univ. de Cantabria – CSIC
MACC / IFCA

M1970 – Machine Learning (L 16:00-18:00; X 16:00-18:00)			
Mar	2	L	Introducción - Redes Probabilísticas Discretas (2h-T)
	4	X	Redes Bayesianas: Creación e Inferencia (2h-L)
	9	L	Clasificacidores Bayesianos. Naive Bayes (2h-L)
	11	X	Redes Bayesianas: Aprendizaje Estructural (2h-T)
	16	L	Redes Bayesianas: Aprendizaje Paramétrico – R. Gaussianas/Mixtas (2h-TL)
	18	X	Redes Bayesianas: Aprendizaje (2h-L)
	23	L	Evaluación (2h)

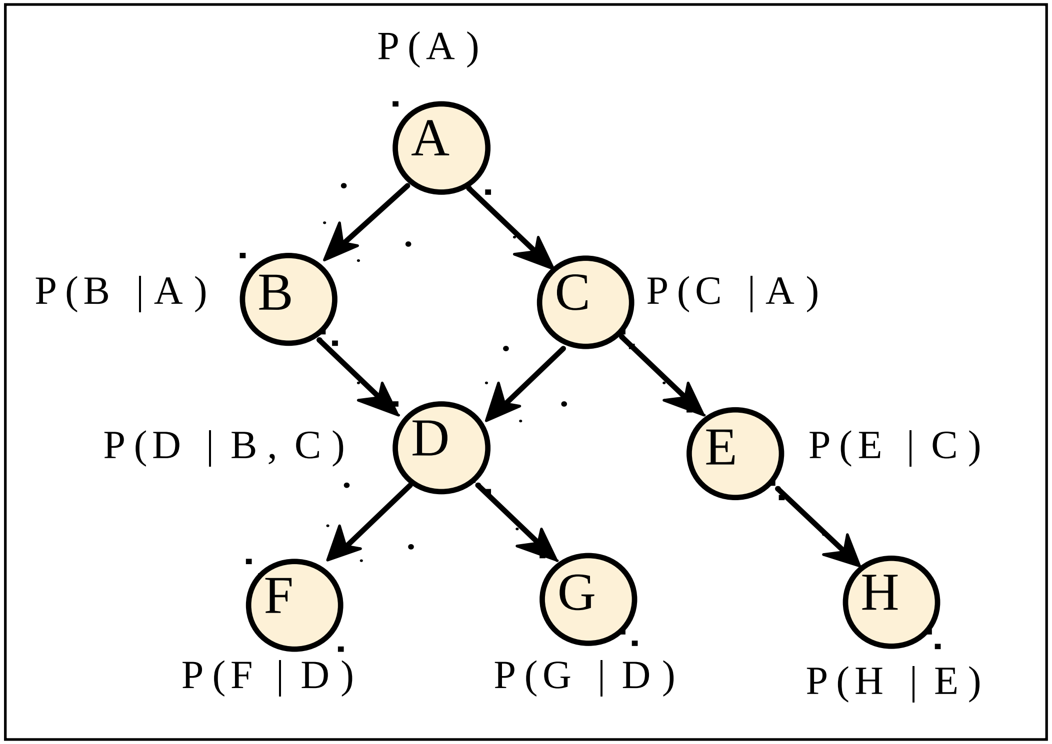
NOTA: Las líneas de código de R en esta presentación se muestran sobre un fondo gris.



UNKNOWN TARGET FUNCTION
P(x)=P(x1, …, xn)
Variables discretas
PROBABILITY DISTRIBUTION
P on X
TRAINING EXAMPLES
x1, …, xN
x1 , … , xN
LEARNING ALGORITHM
A
FINAL MODEL
P'(x)
P'(y|e); E ⊂ \Y
OUTPUT
P(x1, . . . , xn) = Π Pi(xi|πi)
wind
rainfall
dew
storm
hail
dense fog
frost
snow
mist
snow cover
MODEL FAMILY
H
Graphical Probabilistic Models
Adaptado de Y.S. Abu-Mostafa
California Institute of Technology

Directed graphs lead to a probabilistic model directly obtained from the graph, defining the factorization of the joint probability function as product of conditional probabilities of each node x_i given his parents π_i.

$$P(X)=\prod_{i=1}^{n} P(X_i|\pi_i)$$



$$P(A,B,C,D,E,F,G,H)=P(A)P(B|A)P(C|A)P(D|B,C)x\ldots$$
$$xP(E|C)P(F|D)P(G|D)P(H|E)$$

Bayesian Networks obtain a compact representation of the joint probability function through the conditional independences.

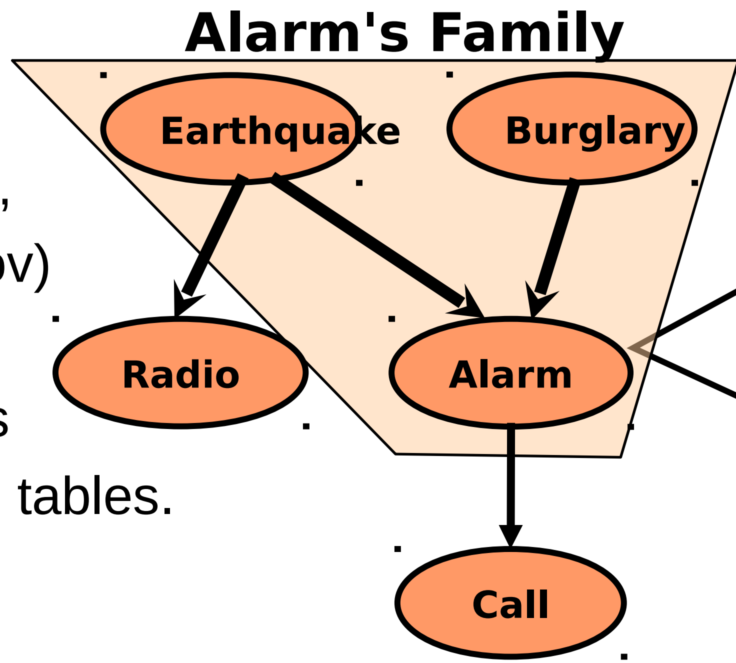
Structure:

Acyclic Directed Graph (DAG),

or non-directed graphs (Markov)

- Nodes – variables
- Links – direct dependences

Parameters: Probabilities and tables.



E	B	P(A | E,B)	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

Once the Bayesian Network (**DAG**+**CPT**) is defined, some questions grow. In particular, given a **new evidence**

- Which is the probability of an event? ← **CPT-Inference**

- There are new (in)dependences between variables? ← **DAG-Inference**

...

Bayesian Networks obtain a compact representation of the joint probability function through the conditional independences.

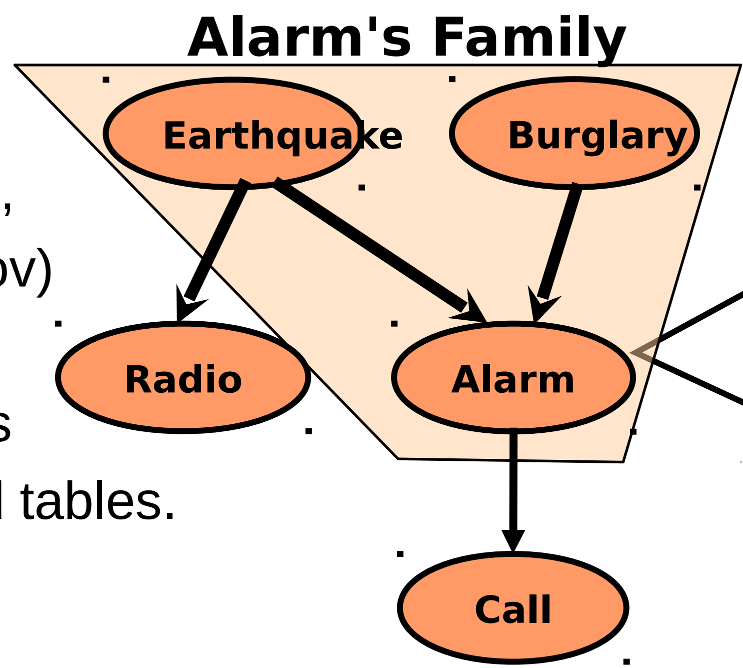
Structure:

Acyclic Directed Graph (DAG),

or non-directed graphs (Markov)

- Nodes – variables
- Links – direct dependences

Parameters: Probabilities and tables.



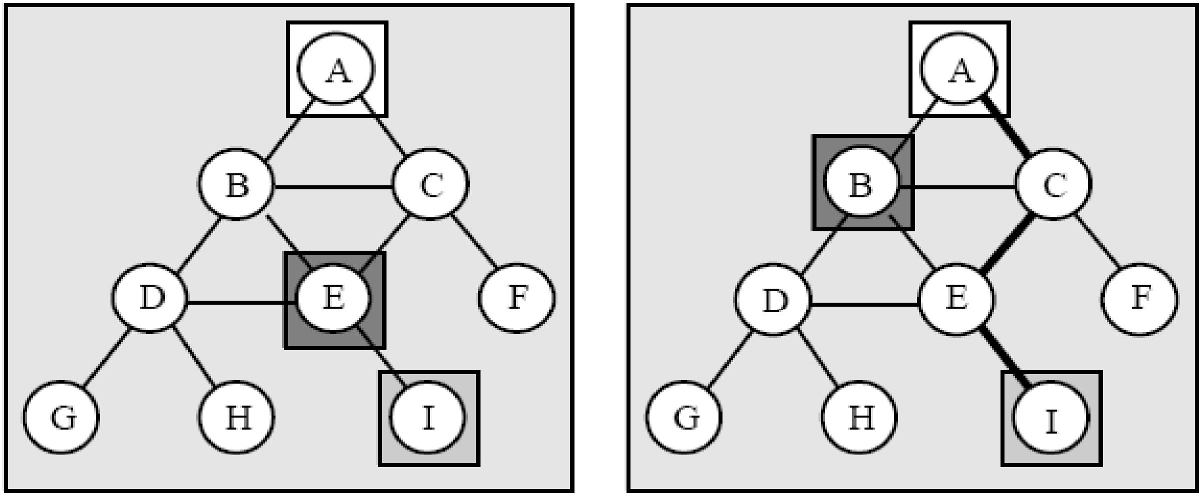
E	B	P(A | E,B)	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

Once the Bayesian Network (**DAG**+**CPT**) is defined, some questions grow. In particular, given a **new evidence**

- Which is the probability of an event? ← **CPT-Inference**

- There are new (in)dependences between variables? ← **DAG-Inference** → **11/03**

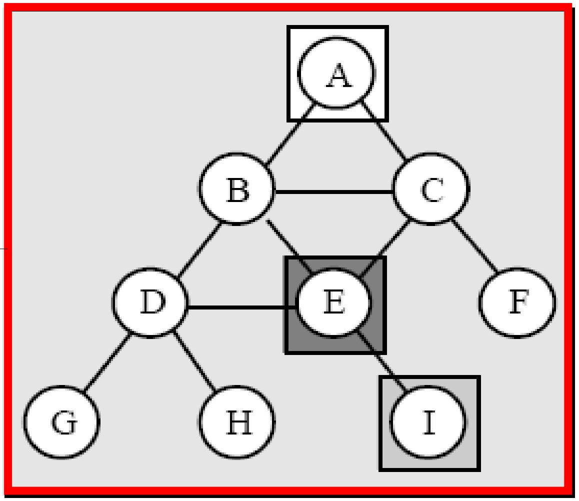
...



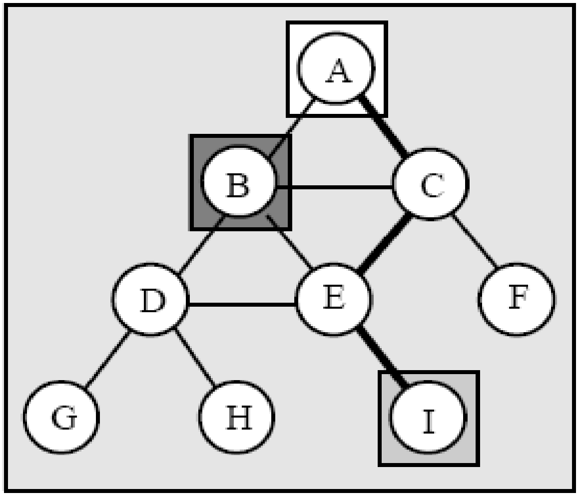
(a) $I(A, I \mid E)$ (b) $D(A, I \mid B)$

Links of the graph reflect **dependences** between the linked variables.

Non directed graphs define the conditional dependence through the **d-separation** concept.



(a) $I(A, I \mid E)$

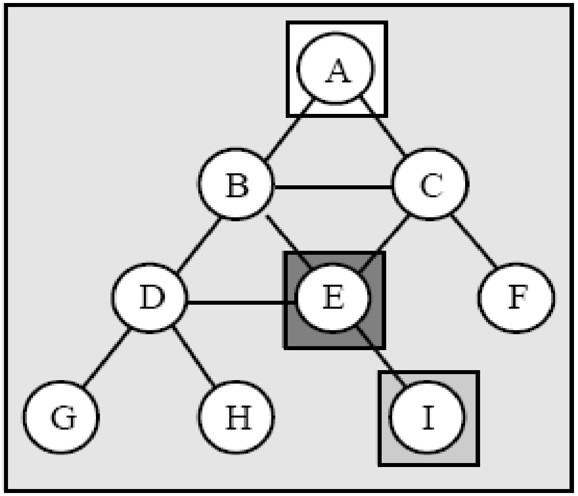


(b) $D(A, I \mid B)$

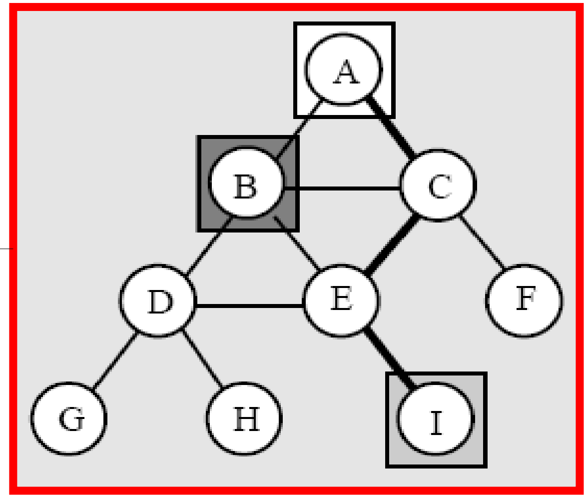
Links of the graph reflect **dependences** between the linked variables.

Non directed graphs define the conditional dependence through the **d-separation** concept.

There is not a path linking A and I not passing for E.
Thus A and I are dependent but conditional independent given E.



(a) $I(A, I \mid E)$

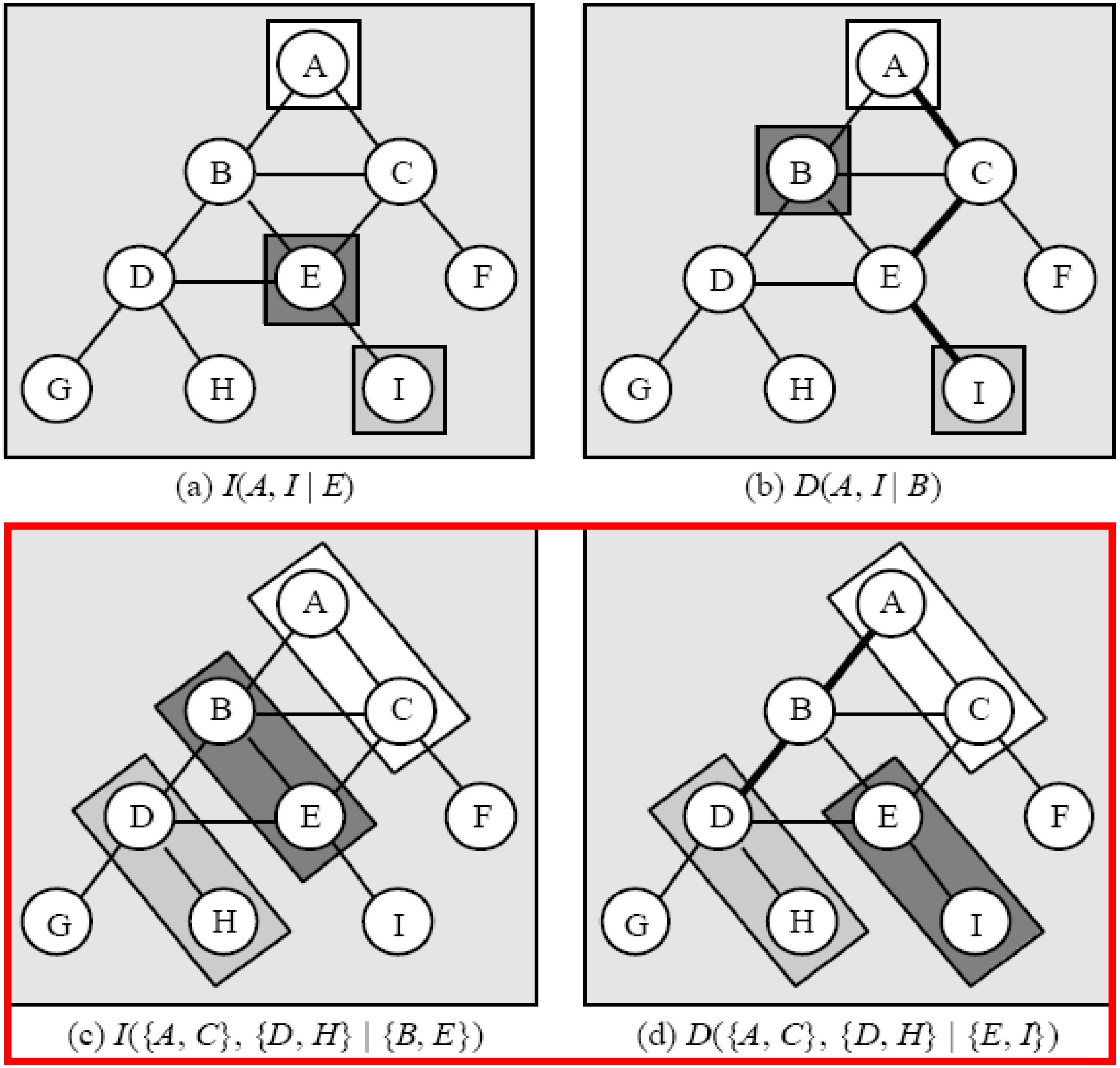


(b) $D(A, I \mid B)$

Links of the graph reflect **dependences** between the linked variables.

Non directed graphs define the conditional dependence through the **d-separation** concept.

There is a path linking A and I not passing for B (A->C->E->I).
Thus A and I are dependent given B and B doesn't d-separate A and I.



(a) $I(A, I \mid E)$ (b) $D(A, I \mid B)$

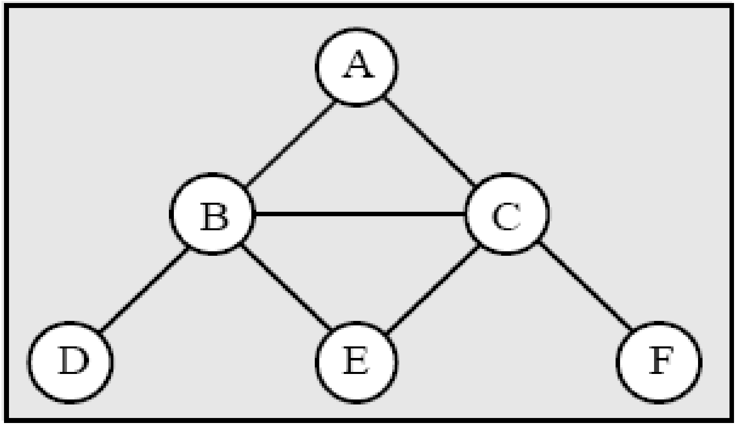
(c) $I(\{A, C\}, \{D, H\} \mid \{B, E\})$ (d) $D(\{A, C\}, \{D, H\} \mid \{E, I\})$

Links of the graph reflect **dependences** between the linked variables.

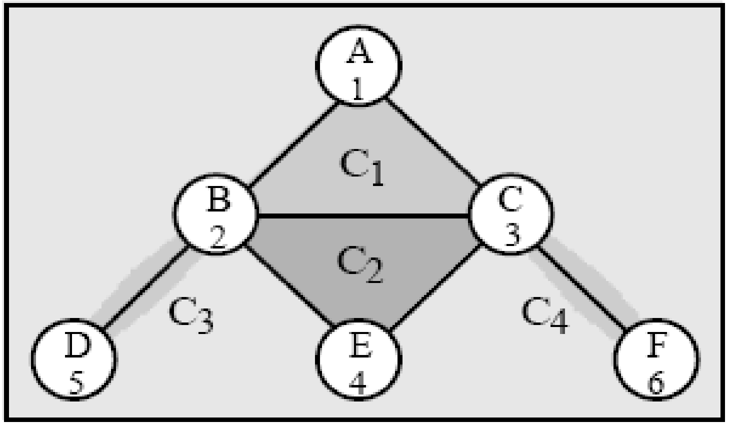
Non directed graphs define the conditional dependence through the **d-separation** concept.

D-separation is extended to set of variables.

Non-directed graphs define a graphical probabilistic model family based on the **cliques** of the graph and the factorization of the joint probability function given by them.



(a)



(b)

$$C_1 = \{A, B, C\}, \quad C_2 = \{B, C, E\},$$
$$C_3 = \{B, D\}, \qquad C_4 = \{C, F\}.$$

i	Clique C_i	Separator S_i	Residual R_i
1	A, B, C	ϕ	A, B, C
2	B, C, E	B, C	E
3	B, D	B	D
4	C, F	C	F

$$\begin{aligned} p(a,b,c,d,e,f) &= \psi_1(c_1)\psi_2(c_2)\psi_3(c_3)\psi_4(c_4) \\ &= \psi_1(a,b,c)\psi_2(b,c,e)\psi_3(b,d)\psi_4(c,f). \end{aligned}$$

$$p(a,b,c,d,e,f) = \prod_{i=1}^{4} p(r_i|s_i) = p(a,b,c)p(e|b,c)p(d|b)p(f|c).$$

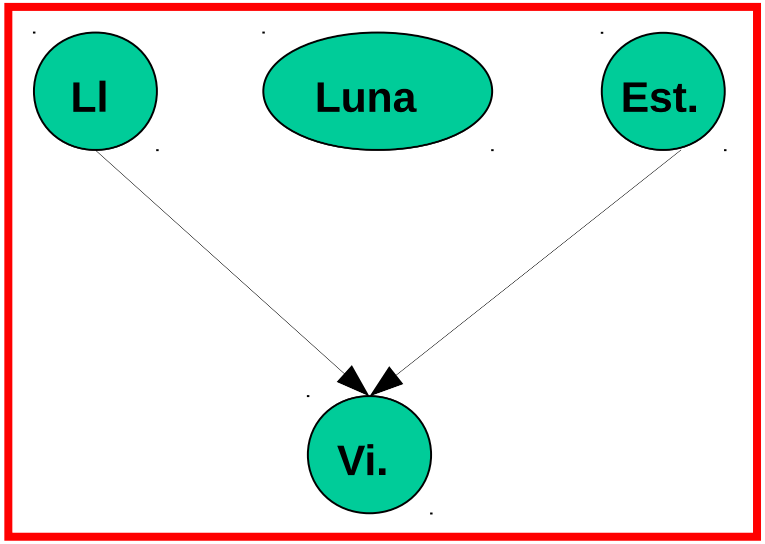
$$P_Z(Y|X) = P(Y|X,Z) = P(Y|Z) = P_Z(Y) \Rightarrow I(X,Y|Z)$$

	Anual		Invierno		Primavera		Verano		Otoño	
	S	Ll	S	Ll	S	Ll	S	Ll	S	Ll
NE	1014	516	190	99	287	166	360	162	177	89
SE	64	57	24	18	6	4	1	9	33	26
SW	225	661	98	223	18	119	15	71	94	248
NW	288	825	49	150	95	277	108	251	36	147
Total	1591	2059	361	490	406	566	484	493	340	510

P (Ll / Primavera) = 0.576
P (Ll / Invierno) = 0.582

Direct independence variables → Involve only two variables

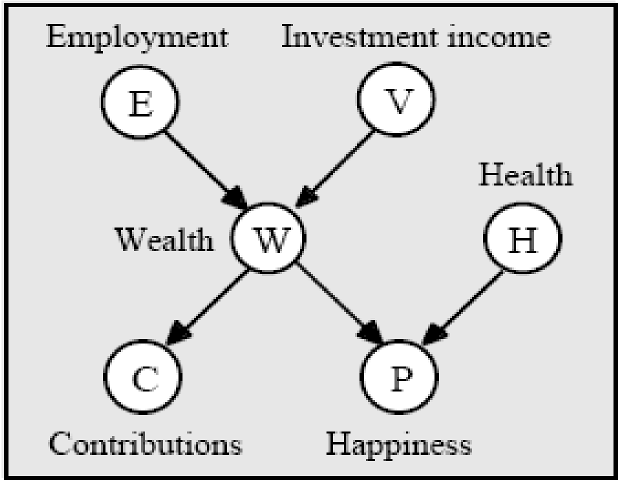
P (Ll) = 0.564

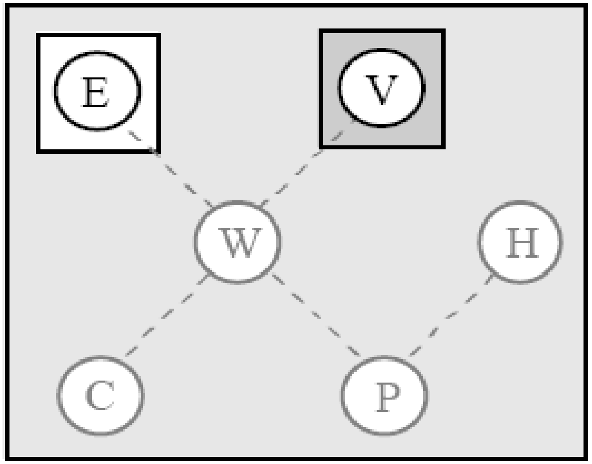


Non-directed graphs are not able to represent this kind of dependence!!!

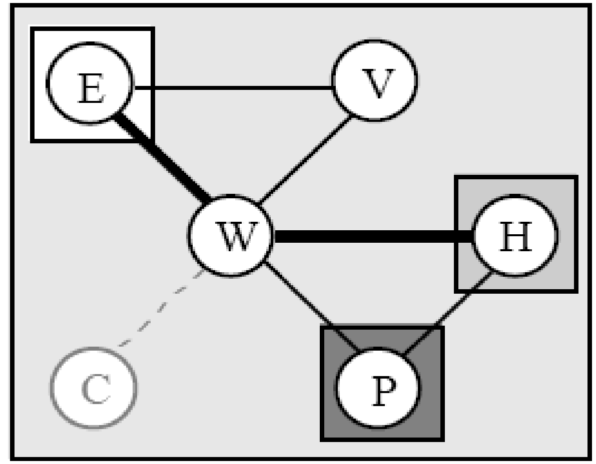
Conditional dependence between rainfall and season, given the wind

D-separation concept for directed graphs enrich the representativity of the model → **Moral graph.**





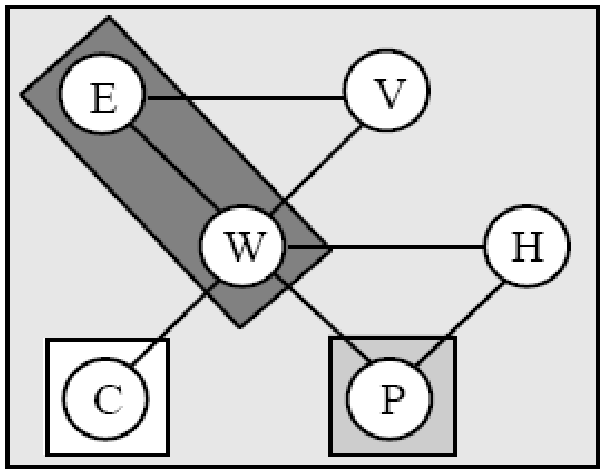
(a) $I(E, V \mid \emptyset)$



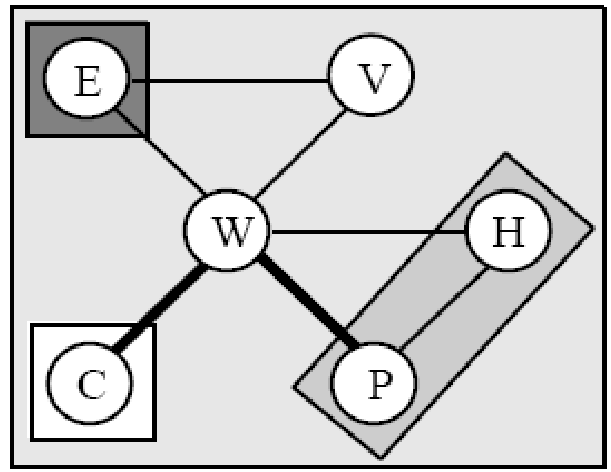
(b) $D(E, H \mid P)$

Links between variables imply probabilistic dependence **NOT CAUSALITY !!!!!!**

Causal Networks (not seen)

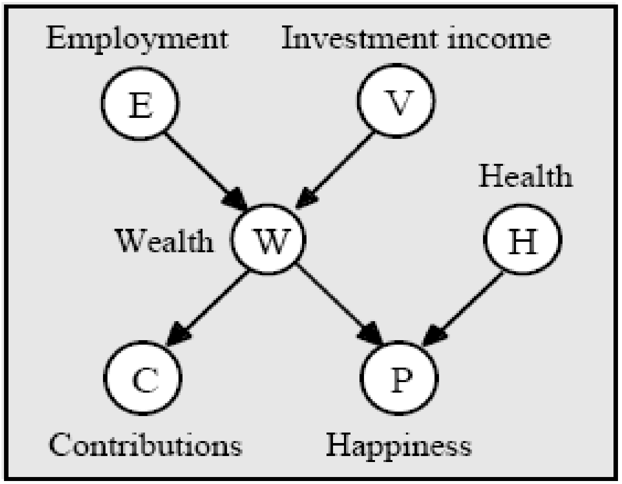


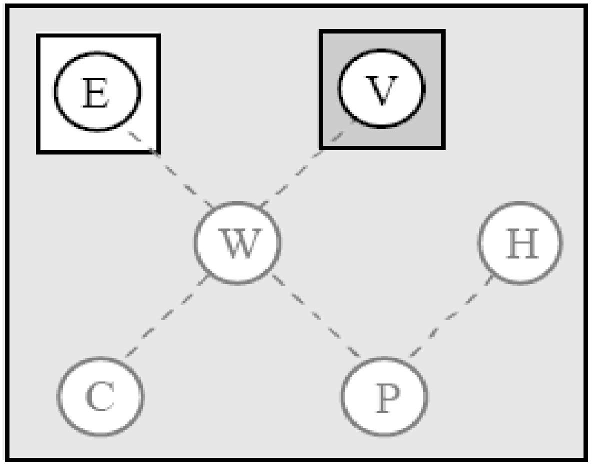
(c) $I(C, P \mid \{E, W\})$



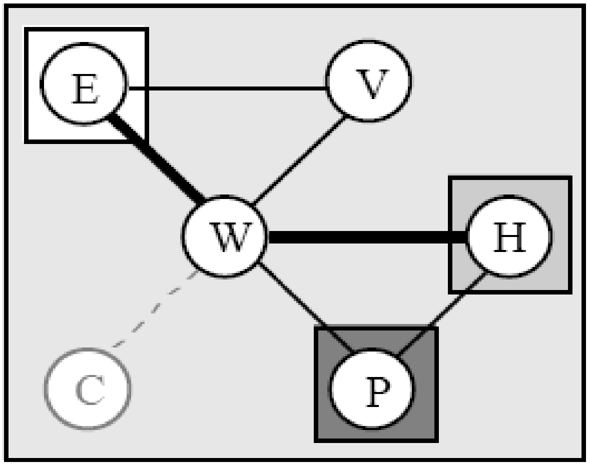
(d) $D(C, \{H, P\} \mid E)$

D-separation concept for directed graphs enrich the representativity of the model → **Moral graph.**





(a) $I(E, V \mid \emptyset)$



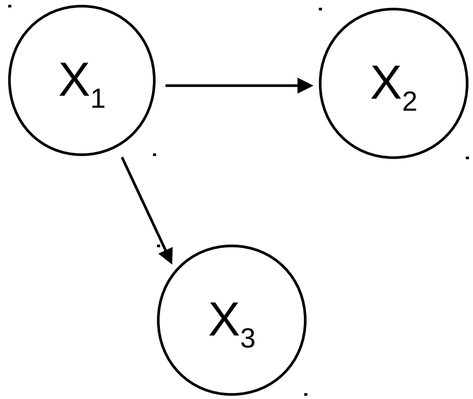
(b) $D(E, H \mid P)$

```
## Load bnlearn:
library(bnlearn)
## Defining an empty graph:
dag<-empty.graph(nodes=c("E","V","W","H","C","P"))
class(dag)
print(dag)
plot(dag)
## Adding link between nodes:
dag<-set.arc(dag,from="E",to="W")
dag<-set.arc(dag,from="V",to="W")
## Complete and plot the graph:
## Evaluate the separation included in the previous slide (See ? dsep and ?path):
```

D-separation concept for directed graphs enrich the representativity of the model → **Moral graph.**

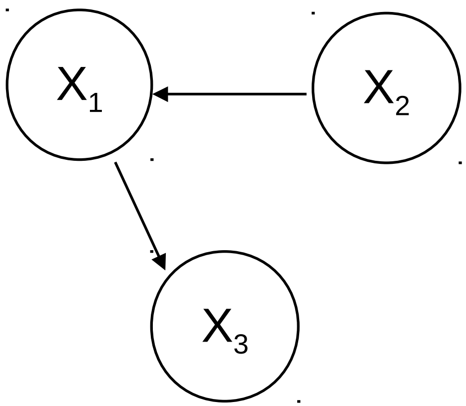
Two directed graph are **equivalents** when they lead to the same probabilistic model:

$$P(X_1, X_2, X_3) = P(X_1) P(X_2|X_1) P(X_3|X_1) = P(X_1, X_2) P(X_3|X_1)$$



Equivalents

$$P(X_1, X_2, X_3) = P(X_2) P(X_1|X_2) P(X_3|X_1) = P(X_1, X_2) P(X_3|X_1)$$

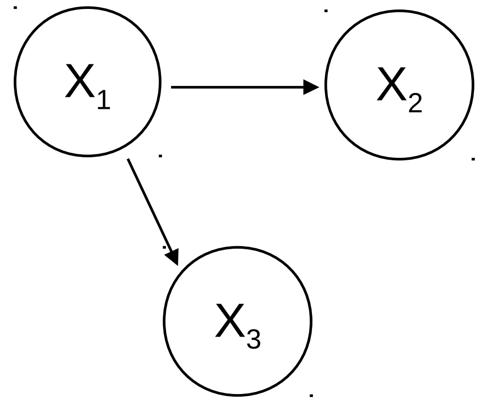


D-separation concept for directed graphs enrich the representativity of the model → **Moral graph.**

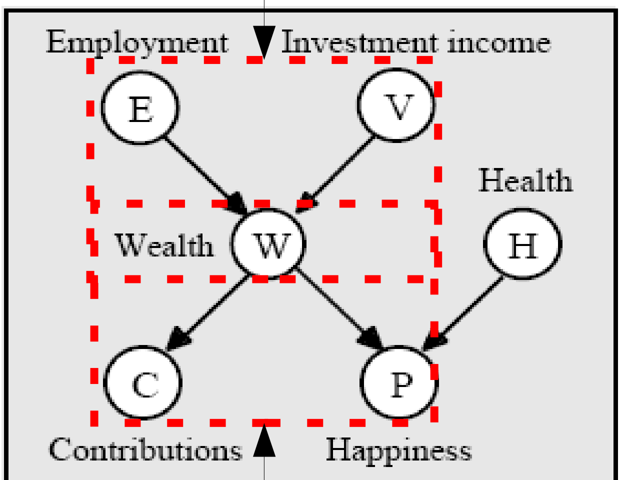
Two directed graph are **equivalents** when they lead to the same probabilistic model.

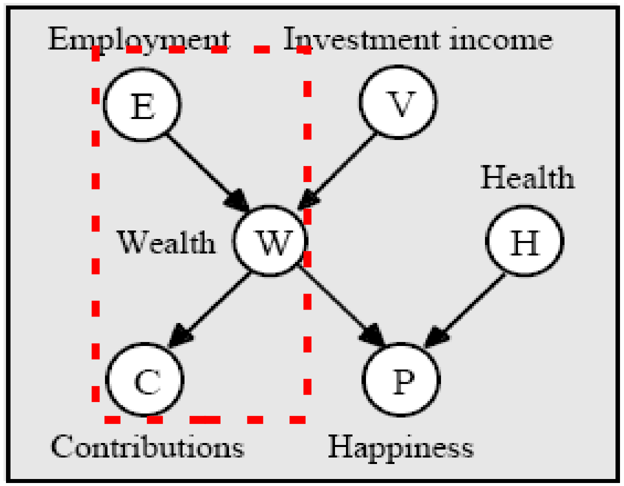
This occurs when the **subyacent non-directed graph** is the same and include the same **V-structures**.

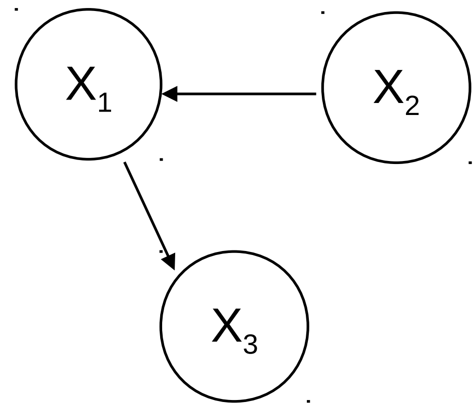
$$P(X_1, X_2, X_3) = P(X_1, X_2) P(X_3 | X_1)$$



Common effect

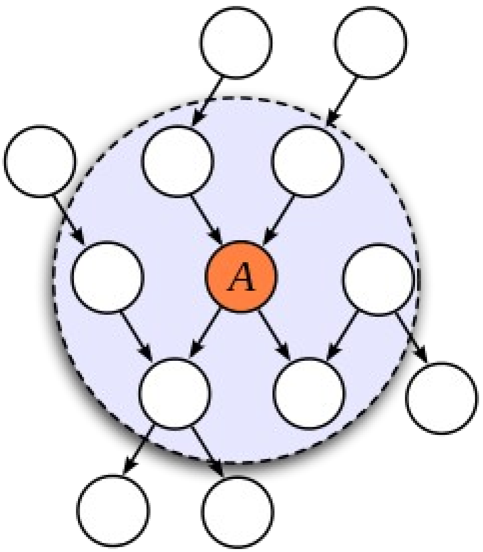






Common cause **Indirect evidential/causal effect**

D-separation concept for directed graphs enrich the representativity of the model → **Moral graph.**
Two directed graph are **equivalents** when they lead to the same probabilistic model.
This occurs when the **subyacent non-directed graph** is the same and include the same **V-structures**.
The **Skeleton** of the graph is the undirected graph underlying.
The **Markov Blanket** of a node ***A*** is the set of nodes that completely separates ***A*** from the rest of the graph. In particular, it includes the parents and childrens of the node ***A***, and those children's other parents.



Source: Image from https://en.wikipedia.org/wiki/Markov_blanket

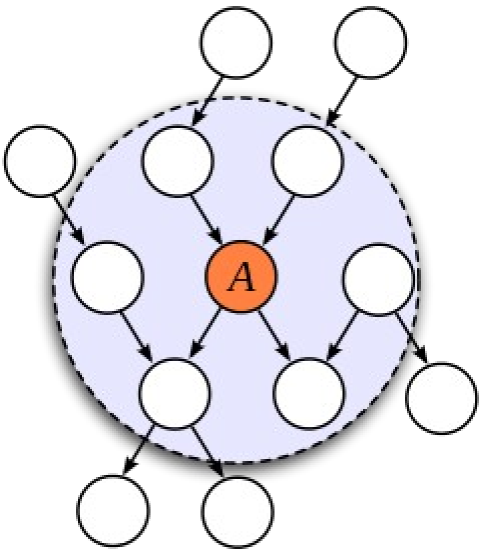
D-separation concept for directed graphs enrich the representativity of the model → **Moral graph.**

Two directed graph are **equivalents** when they lead to the same probabilistic model.

This occurs when the **subyacent non-directed graph** is the same and include the same **V-structures**.

The **Skeleton** of the graph is the undirected graph underlying.

The **Markov Blanket** of a node ***A*** is the set of nodes that completely separates ***A*** from the rest of the graph. In particular, it includes the parents and childrens of the node ***A***, and those children's other parents.



The **Markov Blanket** of is the set of nodes that includes all the knowledge needed to do inference on the node ***A***, from estimation to hypothesis testing to prediction.

Source: Image from https://en.wikipedia.org/wiki/Markov_blanket

Bayesian Networks obtain a compact representation of the joint probability function through the conditional independences.

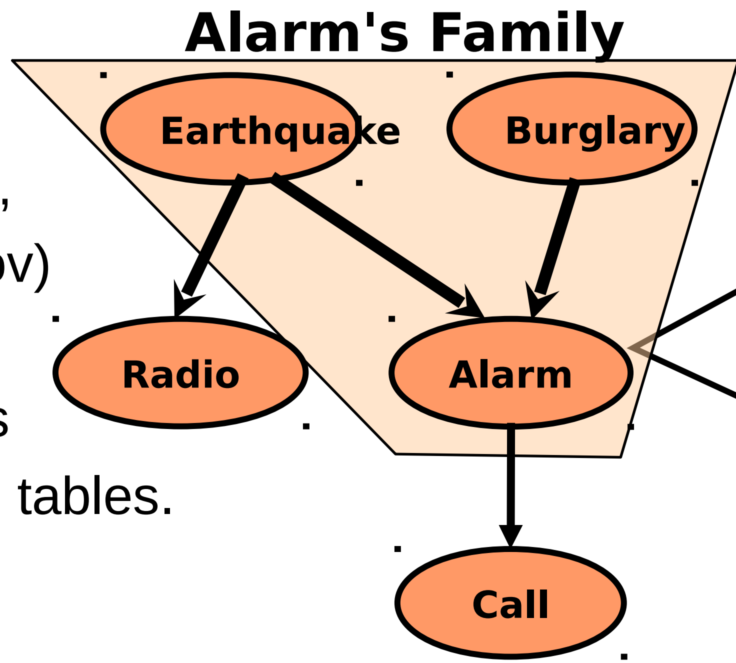
Structure:

Acyclic Directed Graph (DAG),

or non-directed graphs (Markov)

- Nodes – variables
- Links – direct dependences

Parameters: Probabilities and tables.



E	B	P(A | E,B)	
e	b	**0.9**	**0.1**
e	$\bar{b}$	**0.2**	**0.8**
$\bar{e}$	b	**0.9**	**0.1**
$\bar{e}$	$\bar{b}$	**0.01**	**0.99**

Once the Bayesian Network (**DAG**+**CPT**) is defined, some questions grow. In particular, given a **new evidence**

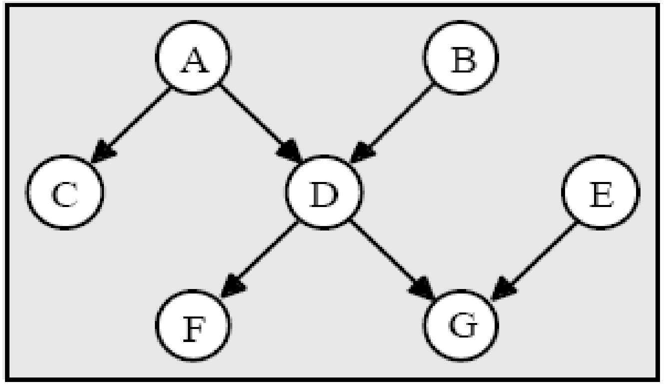
- Which is the probability of an event? ← **CPT-Inference**

- There are new (in)dependences between variables? ← **DAG-Inference**

...

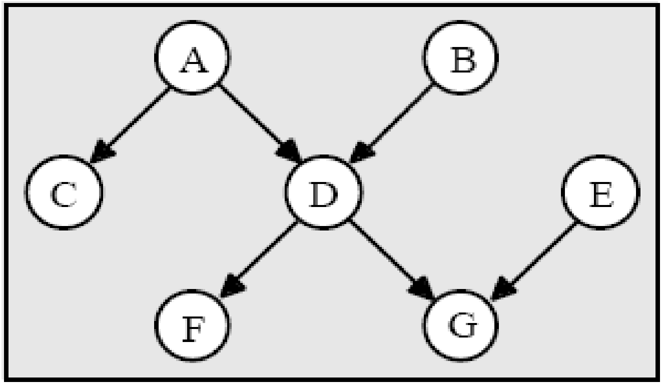
$$\boxed{p(x_i|e)} \quad E \subset X \quad X_i = e_i \quad X_i \in E$$

$$p(x) = p(a)p(b)p(c|a)p(d|a,b)p(e)p(f|d)p(g|d,e),$$



$$\boxed{p(x_i|e)} \quad E \subset X \quad X_i = e_i \quad X_i \in E$$

$$p(x) = p(a)p(b)p(c|a)p(d|a,b)p(e)p(f|d)p(g|d,e),$$



$$p(d) = \sum_{x \setminus d} p(x) = \sum_{a,b,c,e,f,g} p(a,b,c,d,e,f,g).$$

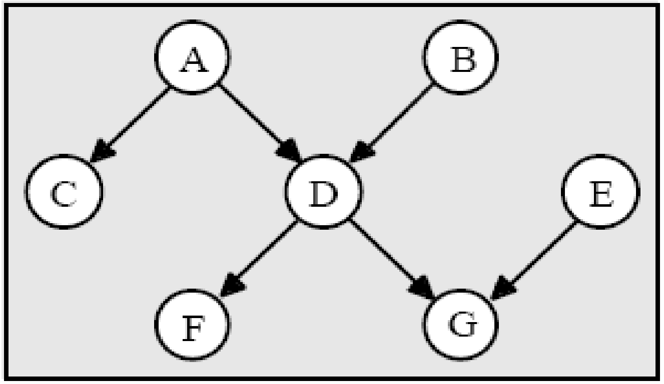
$$p(d) = \sum_{a,b,c,e,f,g} p(a)p(b)p(c|a)p(d|a,b)p(e)p(f|d)p(g|d,e)$$

$$= \left(\sum_{a,b,c} p(a)p(b)p(c|a)p(d|a,b)\right)\left(\sum_{e,f,g} p(e)p(g|d,e)p(f|d)\right),$$

$$\sum_a \left[p(a)\sum_c \left[p(c|a)\sum_b p(b)p(d|a,b)\right]\right]\sum_e \left[p(e)\sum_f \left[p(f|d)\sum_g p(g|d,e)\right]\right]$$

$$\boxed{p(x_i|e)} \quad E \subset X \quad X_i = e_i \quad X_i \in E$$

$$p(x) = p(a)p(b)p(c|a)p(d|a,b)p(e)p(f|d)p(g|d,e),$$



$$p(d) = \sum_{a,b,c,e,f,g} p(a)p(b)p(c|a)p(d|a,b)p(e)p(f|d)p(g|d,e)$$

$$= \left(\sum_{a,b,c} p(a)p(b)p(c|a)p(d|a,b)\right)\left(\sum_{e,f,g} p(e)p(g|d,e)p(f|d)\right),$$

$$\sum_a \left[p(a)\sum_c \left[p(c|a)\sum_b p(b)p(d|a,b)\right]\right]\sum_e \left[p(e)\sum_f \left[p(f|d)\sum_g p(g|d,e)\right]\right]$$

Moralized non-directed graph is obtained and efficient graphs algorithms are applied to obtain the new probabilities. → Exact Inference

Exact inference suffers when the graph is dense (hyper-conected) or there are many variables in the model, losing most of their efficiency and making more adequate the use of aproximated algorithms based on simulation.
Herer we include a brief description of the general approach used by this algorithms:

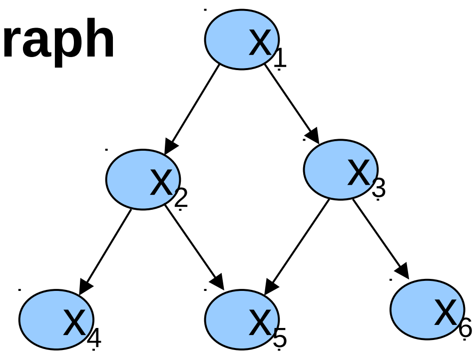
Input: Real probability function P(X) and distribution considered for the simulation h(X) (e.g. uniform), sample size N and a subset Y$\subset$ X.
Output: Approximated value for P(y) for y in Y.

1. For j=1 .. N
   - Generate $x^j = (x^j_1,...,x^j_n)$ from h(x).
   - Estimate $s(x^j) = p(x^j)/ h(x^j)$.

2. For each ***y***, estimate $P(y) \approx \Sigma_y s(x^j) / \Sigma_j s(x^j)$

1. For j=1 .. N
   - Generate $x^j = (x^j_1,...,x^j_n)$ from h(x).
   - Estimate $s(x^j) = p(x^j)/ h(x^j)$.

Graph



2. For each ***y***, estimate $P(y) \approx \Sigma_y\, s(x^j) / \Sigma_j\, s(x^j)$

Joint Probability Function

$$P(X_1,\ldots,X_6)=P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2)P(X_5|X_2,X_3)P(X_6|X_3)$$

x_1	$p(x_1)$
0	0.3
1	0.7

x_1	x_2	$p(x_2\|x_1)$
0	0	0.4
0	1	0.6
1	0	0.1
1	1	0.9

x_1	x_3	$p(x_3\|x_1)$
0	0	0.2
0	1	0.8
1	0	0.5
1	1	0.5

x_2	x_4	$p(x_4\|x_2)$
0	0	0.3
0	1	0.7
1	0	0.2
1	1	0.8

x_3	x_6	$p(x_6\|x_3)$
0	0	0.1
0	1	0.9
1	0	0.4
1	1	0.6

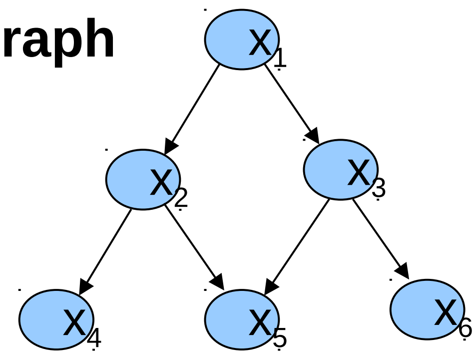
x_2	x_3	x_5	$p(x_5\|x_2,x_3)$
0	0	0	0.4
0	0	1	0.6
0	1	0	0.5
0	1	1	0.5
1	0	0	0.7
1	0	1	0.3
1	1	0	0.2
1	1	1	0.8

For example, for the event (0,1,1,1,0,0) this is the probability:
$p(0,1,1,1,0,0) = p(x_1=0)p(x_2=1|x_1=0)p(x_3=1|x_1=0)p(x_4=1|x_2=1)$
$p(x_5=0|x_2=1,x_3=1)p(x_6=0|x_3=1) = 0.3 \times 0.6 \times 0.8 \times 0.8 \times 0.2 \times 0.4=0.009216$

1. For j=1 .. N
   - Generate $x^j = (x^j_1,\ldots,x^j_n)$ from h(x).
   - Estimate $s(x^j) = p(x^j)/ h(x^j)$.

2. For each ***y***, estimate $P(y) \approx \Sigma_y\, s(x^j) / \Sigma_j\, s(x^j)$

Graph



Joint Probability Function

$$P(X_1,\ldots,X_6)=P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2)P(X_5|X_2,X_3)P(X_6|X_3)$$

x_1	$p(x_1)$
0	0.3
1	0.7

x_1	x_2	$p(x_2\|x_1)$
0	0	0.4
0	1	0.6
1	0	0.1
1	1	0.9

x_1	x_3	$p(x_3\|x_1)$
0	0	0.2
0	1	0.8
1	0	0.5
1	1	0.5

x_2	x_4	$p(x_4\|x_2)$
0	0	0.3
0	1	0.7
1	0	0.2
1	1	0.8

x_3	x_6	$p(x_6\|x_3)$
0	0	0.1
0	1	0.9
1	0	0.4
1	1	0.6

x_2	x_3	x_5	$p(x_5\|x_2,x_3)$
0	0	0	0.4
0	0	1	0.6
0	1	0	0.5
0	1	1	0.5
1	0	0	0.7
1	0	1	0.3
1	1	0	0.2
1	1	1	0.8

Six binary variables → 2^6=64 posibilities → Suppose ***h*** uniform → h(x)=1/64

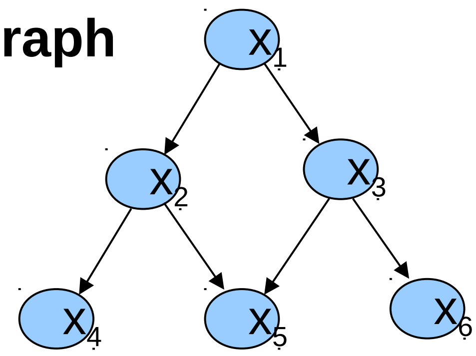
Step 1

Realization x^j	$p(x^j)$	$h(x^j)$	$s(x^j)= p(x^j)/h(x^j)$
x^1=(0,1,1,1,0,0)	0.0092	1/64	0.5898
x^2=(1,1,0,1,1,0)	0.0076	1/64	0.4838
x^3=(0,0,1,0,0,1)	0.0086	1/64	0.5529
x^4=(1,0,0,1,1,0)	0.0015	1/64	0.0941
x^5=(1,0,0,0,1,1)	0.0057	1/64	0.3629

1. For j=1 .. N
   - Generate $x^j = (x^j_1,\dots,x^j_n)$ from $h(x)$.
   - Estimate $s(x^j) = p(x^j)/ h(x^j)$.

2. For each ***y***, estimate $P(y) \approx \Sigma_y\, s(x^j) / \Sigma_j\, s(x^j)$

Graph



Joint Probability Function

$$P(X_1,\dots,X_6)=P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2)P(X_5|X_2,X_3)P(X_6|X_3)$$

x_1	$p(x_1)$
0 1	0.3 0.7

x_1	x_2	$p(x_2\|x_1)$
0	0	0.4
0	1	0.6
1	0	0.1
1	1	0.9

x_1	x_3	$p(x_3\|x_1)$
0	0	0.2
0	1	0.8
1	0	0.5
1	1	0.5

x_2	x_4	$p(x_4\|x_2)$
0	0	0.3
0	1	0.7
1	0	0.2
1	1	0.8

x_3	x_6	$p(x_6\|x_3)$
0	0	0.1
0	1	0.9
1	0	0.4
1	1	0.6

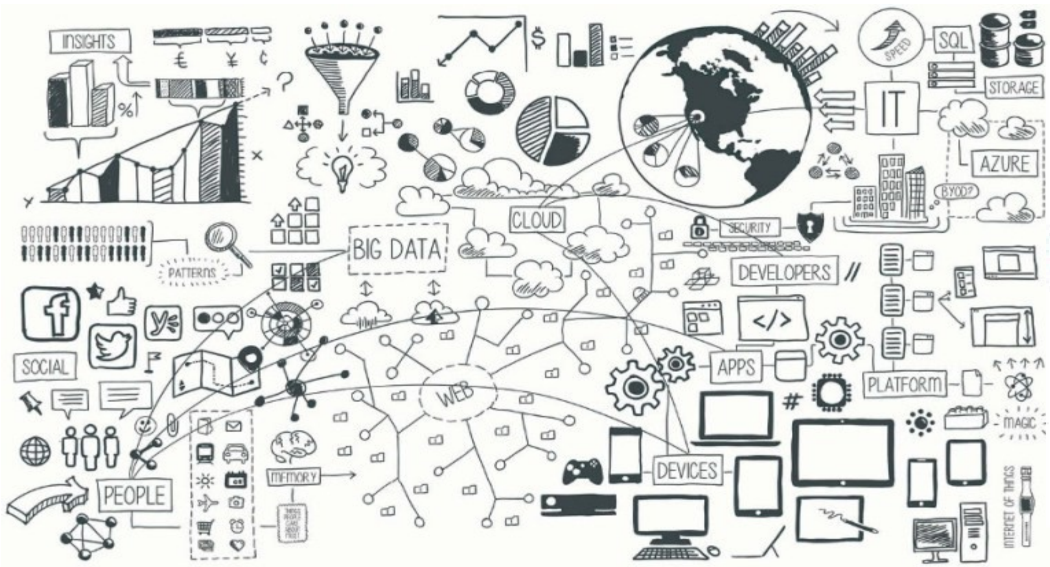
Step 2

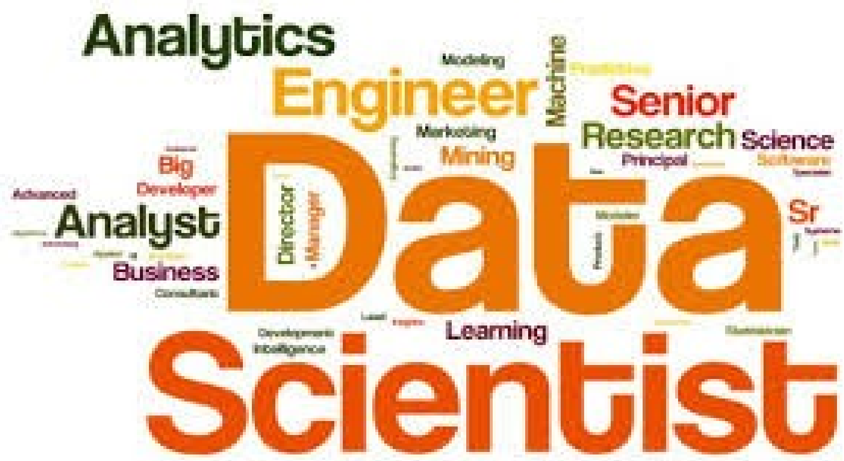
Realization x^j	$p(x^j)$	$h(x^j)$	$s(x^j)= p(x^j)/h(x^j)$
$x^1=(0,1,1,1,0,0)$	0.0092	1/64	0.5898
$x^2=(1,1,0,1,1,0)$	0.0076	1/64	0.4838
$x^3=(0,0,1,0,0,1)$	0.0086	1/64	0.5529
$x^4=(1,0,0,1,1,0)$	0.0015	1/64	0.0941
$x^5=(1,0,0,0,1,1)$	0.0057	1/64	0.3629

Poor estimation due to the number of simulations (5)

$p(X_1=0)\approx[s(x^1)+s(x^3)]/\Sigma_j s(x^j)=[0.5898+0.5529]/2.0835=$ **0.5485**

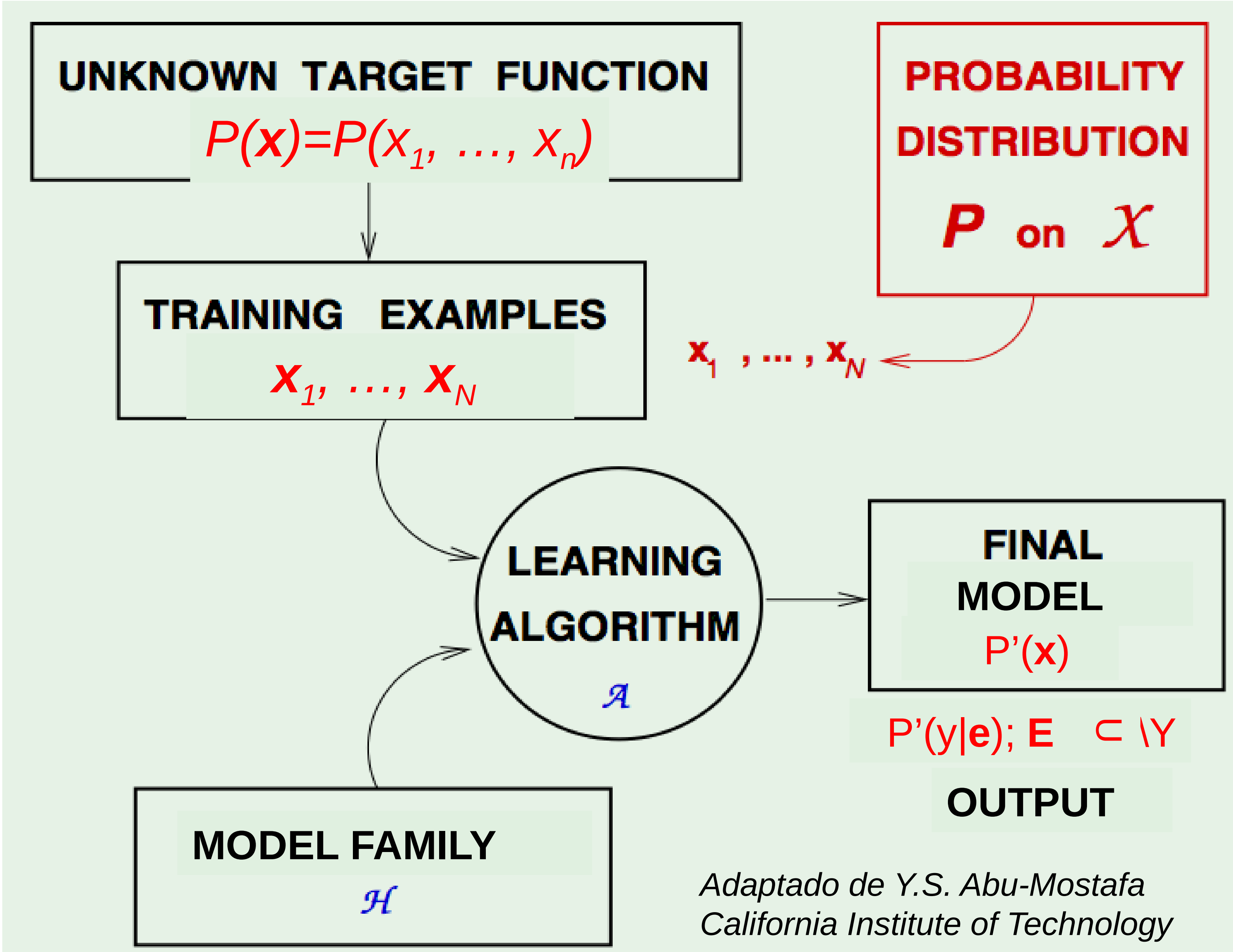
M1970 – Machine Learning II
Redes Probabilísticas Discretas
(Clasificadores Bayesianos)



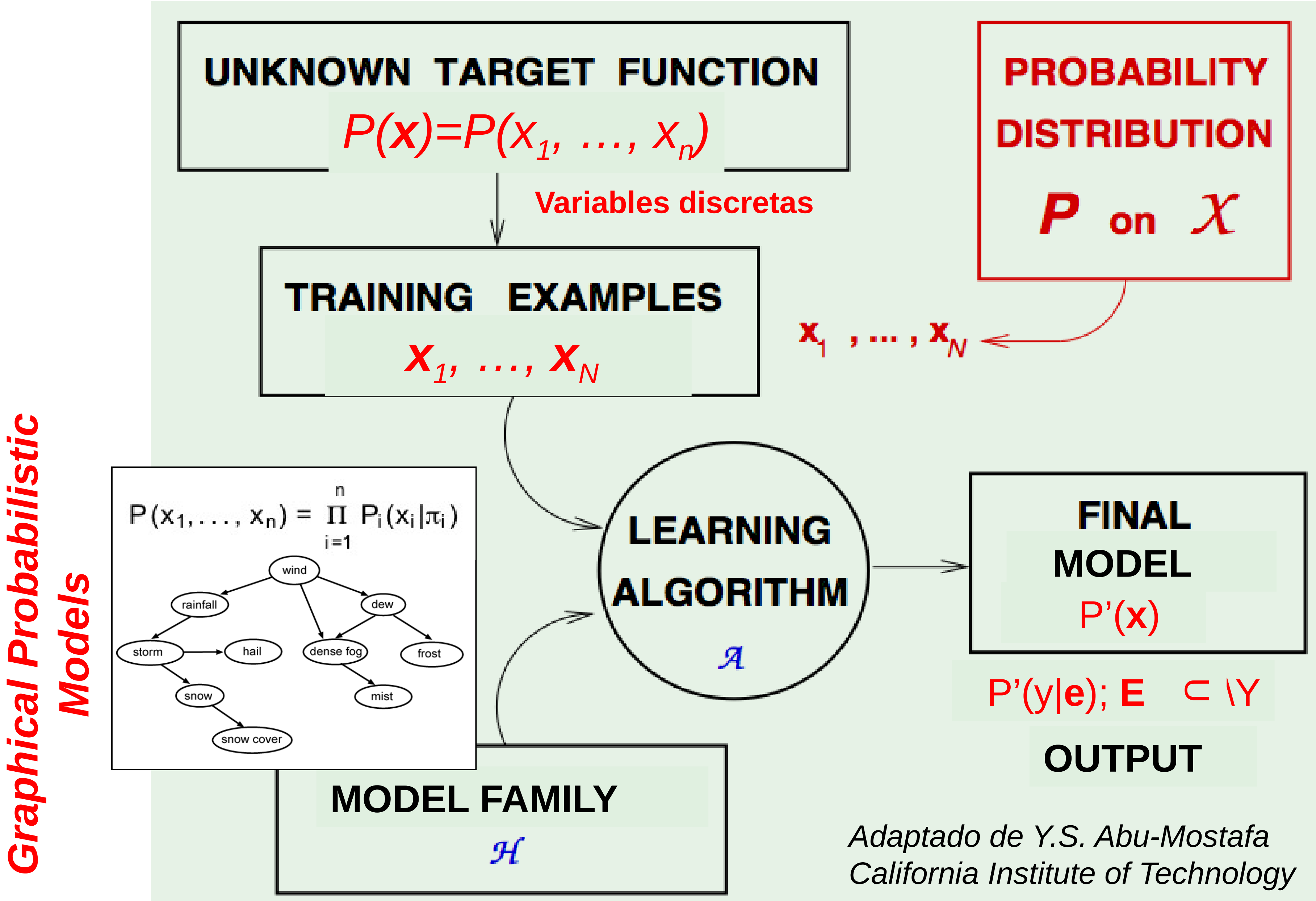


Sixto Herrera (sixto.herrera@unican.es)
José M. Gutiérrez, Mikel Legasa

Grupo de Meteorología
Univ. de Cantabria – CSIC
MACC / IFCA



Adaptado de Y.S. Abu-Mostafa
California Institute of Technology


UNKNOWN TARGET FUNCTION
P(x)=P(x1, …, xn)
Variables discretas
PROBABILITY DISTRIBUTION
P on X
TRAINING EXAMPLES
x1, …, xN
x1 , … , xN
P(x1, . . . , xn) = Π Pi(xi|πi)
wind
rainfall
dew
storm
hail
dense fog
frost
snow
mist
snow cover
LEARNING ALGORITHM
A
FINAL MODEL
P'(x)
P'(y|e); E ⊂ \Y
OUTPUT
MODEL FAMILY
H
Adaptado de Y.S. Abu-Mostafa
California Institute of Technology
Graphical Probabilistic Models

x	y	z	$p(x,y,z)$
0	0	0	0.12
0	0	1	0.18
0	1	0	0.04
0	1	1	0.16
1	0	0	0.09
1	0	1	0.21
1	1	0	0.02
1	1	1	0.18

To explicitly define the joint probability function is not possible in most of the real cases due to the large amount of parameters (e.g. 10^{25} parameters for 100 variables).

x	y	z	$p(x,y,z)$
0	0	0	0.12
0	0	1	0.18
0	1	0	0.04
0	1	1	0.16
1	0	0	0.09
1	0	1	0.21
1	1	0	0.02
1	1	1	0.18

To explicitly define the joint probability function is not possible in most of the real cases due to the large amount of parameters (e.g. 10^{25} parameters for 100 variables).

Bayes's Theorem → Factorization

$$P(X_i|B)=\frac{P(B|X_i)\,P(X_i)}{\sum_{j=1}^{n} P(B|X_j)\,P(X_j)}$$

$$\underbrace{B \wedge X_i\ independent \Rightarrow P(X_i|B)=P(X_i) \wedge P(B|X_i)=P(B)}$$

Reduction of parameters

x	y	z	$p(x,y,z)$
0	0	0	0.12
0	0	1	0.18
0	1	0	0.04
0	1	1	0.16
1	0	0	0.09
1	0	1	0.21
1	1	0	0.02
1	1	1	0.18

To explicitly define the joint probability function is not possible in most of the real cases due to the large amount of parameters (e.g. 10^{25} parameters for 100 variables).

Bayes's Theorem → Factorization

$$P(X_i|B)=\frac{P(B|X_i)\,P(X_i)}{\sum_{j=1}^{n} P(B|X_j)\,P(X_j)}$$

$$\underbrace{B \wedge X_i \; independent \Rightarrow P(X_i|B)=P(X_i) \wedge P(B|X_i)=P(B)}$$

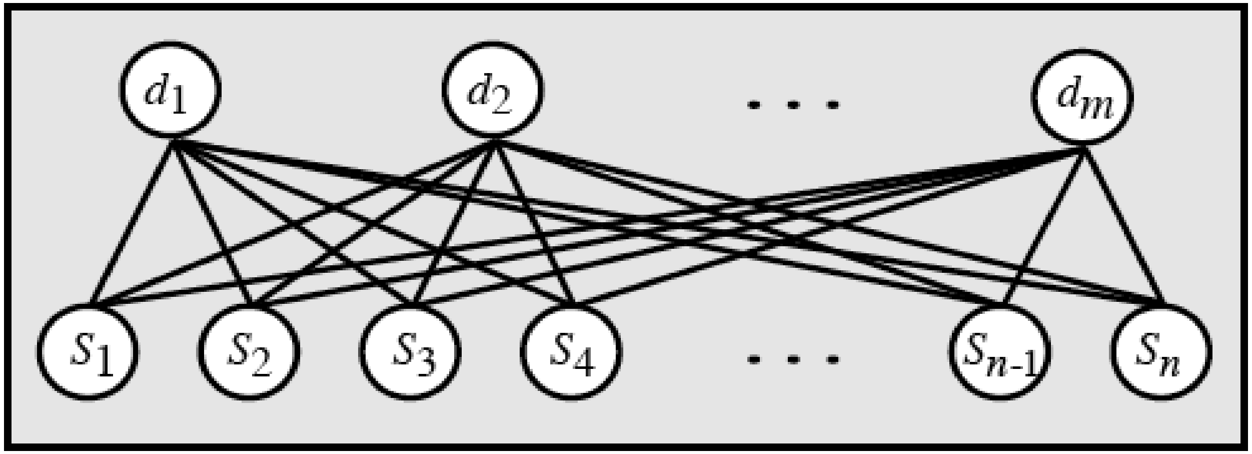
.

Reduction of parameters

Can we build a model including some pre-defined independences?

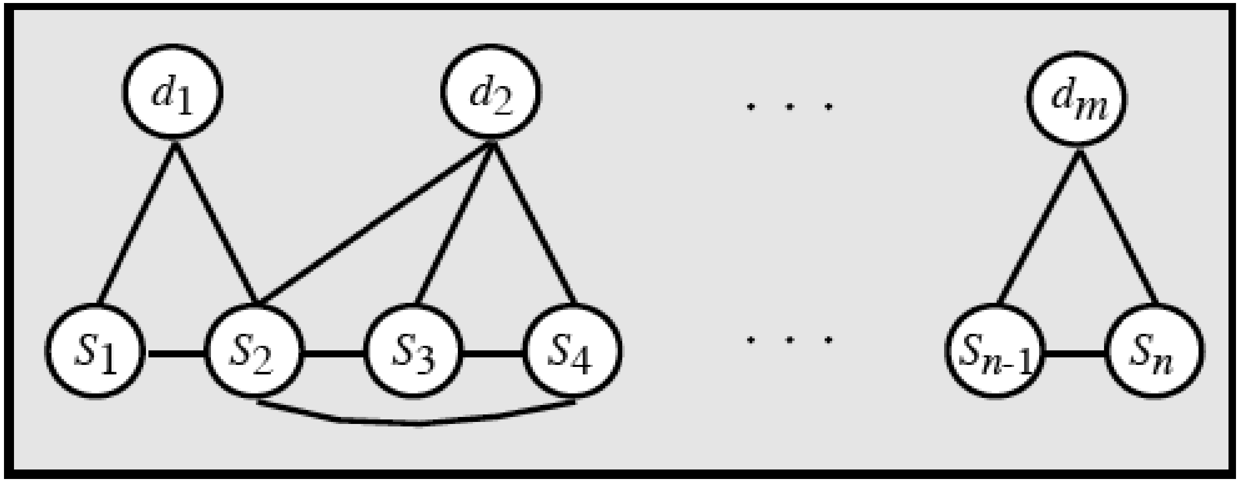
Firstly, unrealistic models "*ad-hoc*" were proposed.

$$P(s_1, \ldots, s_n, d_1, \ldots, d_m) = P(s_1, \ldots, s_n | d_1, \ldots, d_m)\, P(d_1, \ldots, d_m)$$



Independent symptoms model → Independent symptoms given a disease

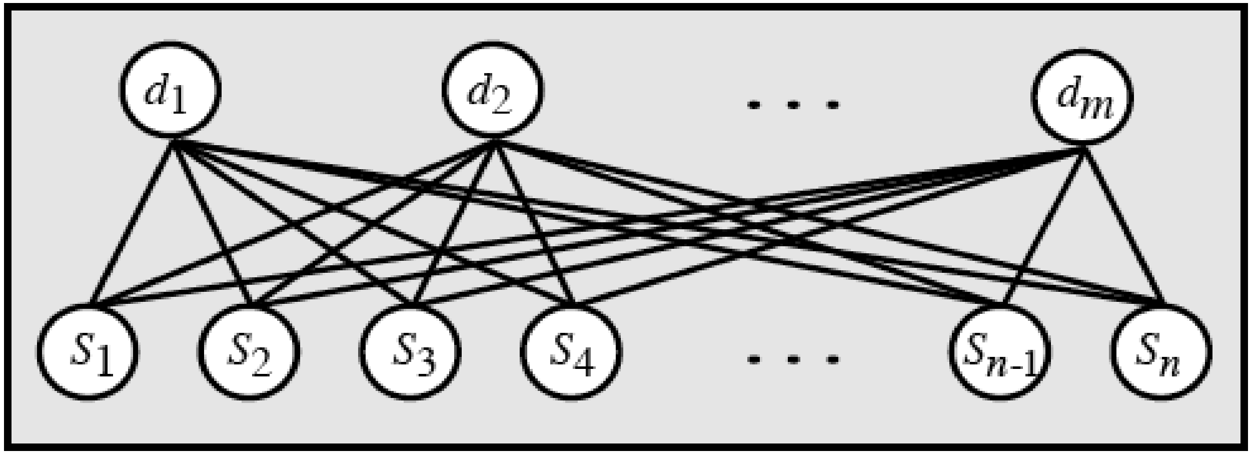
$$p(s_1, \ldots, s_n | d_i) = \prod_{i=1}^{n} p(s_j | d_i).$$



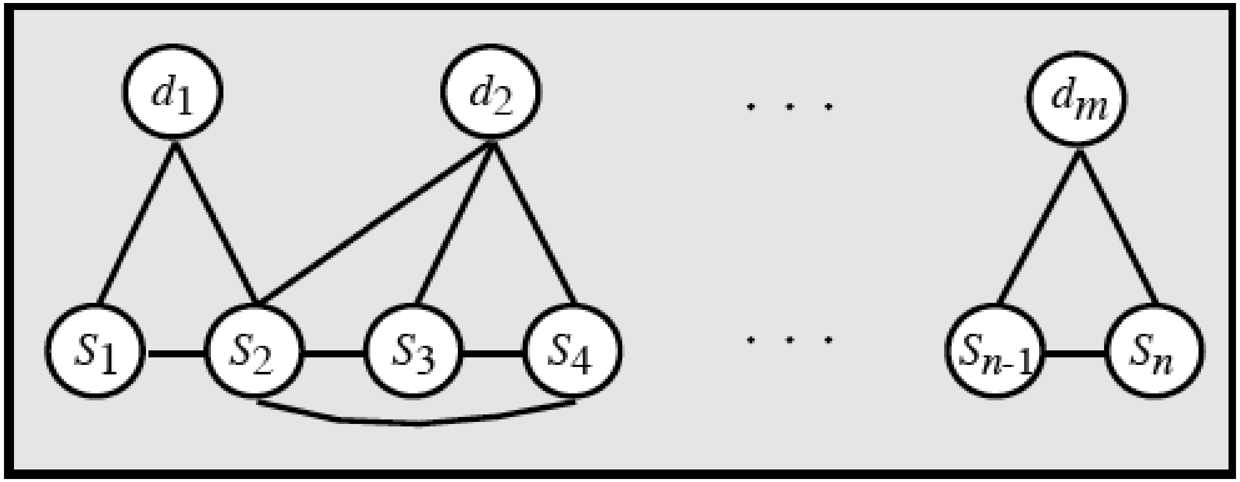
Syndrome model → for each disease there are a relevant subset of dependent symptoms.

Firstly, unrealistic models "*ad-hoc*" were proposed.

$$P(s_1, \ldots, s_n, d_1, \ldots, d_m) = P(s_1, \ldots, s_n | d_1, \ldots, d_m)\, P(d_1, \ldots, d_m)$$



$$p(s_1, \ldots, s_n | d_i) = \prod_{j=1}^{n} p(s_j | d_i).$$



Is there any method to objectively define dependences between the variables and reduce the number of parameters?

Subjective/ad-hoc approach
Large amount of parameters

	Número de parámetros	
Modelo	Fórmula	Valor
DSM	$m2^n - 1$	$> 10^{62}$
ISM	$m(n+1) - 1$	20,099
IRSM	$m(r+1) + n - 1$	1,299
DRSM	$m2^r + n - 1$	102,599

$$C \in \{c_1, \ldots, c_m\}$$ Target variable with ***m*** states/classes

$$X = \{X_1, \ldots, X_n\}$$ Predictors in a ***n-dimensional*** space

.

[Bayes' Theorem (Predictands vs. Predictors)]()

$$P(C|\{X_1, \ldots, X_n\}) = \frac{P(\{X_1, \ldots, X_n\}|C)\, P(C)}{P(\{X_1, \ldots, X_n\})}$$

Bayesian Classifier

$$Arg_C[Max(P(C|\{X_1, \ldots, X_n\}))]$$

$$C \in \{c_1, \dots, c_m\}$$ Target variable with ***m*** states/classes

$$X = \{X_1, \dots, X_n\}$$ Predictors in a ***n-dimensional*** space

.

[Bayes' Theorem (Predictands vs. Predictors)]()

$$P(C|\{X_1, \dots, X_n\}) = \frac{P(\{X_1, \dots, X_n\}|C)\,P(C)}{P(\{X_1, \dots, X_n\})}$$

Bayesian Classifier

$$Arg_C[Max(P(C|\{X_1, \dots, X_n\}))]$$

$P(\{X_1, \dots, X_n\})$ ← **Constant**

$$Arg_C[Max(P(\{X_1, \dots, X_n\}|C)\,P(C))]$$

$$C \in \{c_1, \ldots, c_m\}$$ Target variable with ***m*** states/classes

$$X = \{X_1, \ldots, X_n\}$$ Predictors in a ***n-dimensional*** space

Bayes' Theorem (Predictands vs. Predictors)

$$P(C|\{X_1,\ldots,X_n\}) = \frac{P(\{X_1,\ldots,X_n\}|C)\,P(C)}{P(\{X_1,\ldots,X_n\})}$$

Bayesian Classifier

$$Arg_C[Max(P(C|\{X_1,\ldots,X_n\}))]$$

$P(\{X_1,\ldots,X_n\})$ **Constant**

Probability "***a priori***"

$$Arg_C[Max(P(\{X_1,\ldots,X_n\}|C)\boxed{P(C))}]$$

$C \in \{c_1, \dots, c_m\}$ Target variable with ***m*** states/classes

$X = \{X_1, \dots, X_n\}$ Predictors in a ***n-dimensional*** space

[Bayes' Theorem (Predictands vs. Predictors)]()

$$P(C|\{X_1,\dots,X_n\}) = \frac{P(\{X_1,\dots,X_n\}|C)P(C)}{P(\{X_1,\dots,X_n\})}$$

Bayesian Classifier

$$Arg_C[Max(P(C|\{X_1,\dots,X_n\}))]$$

$P(\{X_1,\dots,X_n\})$ **Constant**

Probability "***a priori***"

$$Arg_C[Max(\boxed{P(\{X_1,\dots,X_n\}|C)}P(C))]$$

Verisimilitude

$C \in \{c_1, \ldots, c_m\}$ ⟵ Target variable with ***m*** states/classes

$X = \{X_1, \ldots, X_n\}$ ⟵ Predictors in a ***n-dimensional*** space

.

Bayes' Theorem (Predictands vs. Predictors)

$$P(C|\{X_1,\ldots,X_n\}) = \frac{P(\{X_1,\ldots,X_n\}|C)\,P(C)}{P(\{X_1,\ldots,X_n\})}$$

Bayesian Classifier

$$Arg_C[Max(P(C|\{X_1,\ldots,X_n\}))]$$

m	n		parámetros
3	10	$\simeq$	$8 \cdot 10^3$
5	20	$\simeq$	$33 \cdot 10^6$
10	50	$\simeq$	$11 \cdot 10^{17}$

$P(\{X_1,\ldots,X_n\})$ ⟵ **Constant**

Probability "***a priori***"

$$Arg_C[Max(\boxed{P(\{X_1,\ldots,X_n\}|C)}\,P(C))]$$

Verisimilitude

$$C \in \{c_1, \ldots, c_m\}$$ Target variable with ***m*** states/classes

$$X = \{X_1, \ldots, X_n\}$$ Predictors in a ***n-dimensional*** space

Bayes' Theorem (Predictands vs. Predictors)

$$P(C|\{X_1,\ldots,X_n\}) = \frac{P(\{X_1,\ldots,X_n\}|C)\,P(C)}{P(\{X_1,\ldots,X_n\})}$$

\+ $P(X_i|\{X_j, C\}) = P(X_i|C)\ \forall\ j \neq i$

Naive Bayesian Classifier

Exclusive states/classes
Predictors conditionally independent given the state.

$$C \in \{c_1, \ldots, c_m\}$$

Target variable with ***m*** states/classes

$$X = \{X_1, \ldots, X_n\}$$

Predictors in a ***n-dimensional*** space

[Bayes' Theorem (Predictands vs. Predictors)]

$$P(C|\{X_1, \ldots, X_n\}) = \frac{P(\{X_1, \ldots, X_n\}|C)\,P(C)}{P(\{X_1, \ldots, X_n\})}$$

\+ $P(X_i|\{X_j, C\}) = P(X_i|C)\ \forall\ j \neq i$

Naive Bayesian Classifier

Exclusive states/classes
Predictors conditionally independent given the state.

$$Arg_C[Max(P(\{X_1, \ldots, X_n\}|C)\,P(C))] = Arg_C[Max(P(X_1|C) \ldots P(X_n|C)\,P(C))]$$

$$C \in \{c_1, \dots, c_m\}$$

Target variable with ***m*** states/classes

$$X = \{X_1, \dots, X_n\}$$

Predictors in a ***n-dimensional*** space

.

Bayes' Theorem (Predictands vs. Predictors)

$$P(C|\{X_1,\dots,X_n\}) = \frac{P(\{X_1,\dots,X_n\}|C)\,P(C)}{P(\{X_1,\dots,X_n\})}$$

+ $P(X_i|\{X_j, C\}) = P(X_i|C)\ \forall\ j \neq i$

Naive Bayesian Classifier

Exclusive states/classes
Predictors conditionally independent given the state.

$$Arg_C[Max(P(\{X_1,\dots,X_n\}|C)P(C))] = Arg_C[Max(P(X_1|C)\dots P(X_n|C)P(C))]$$

Bayesian Classifier

m	n		parámetros
3	10	$\simeq$	$8 \cdot 10^3$
5	20	$\simeq$	$33 \cdot 10^6$
10	50	$\simeq$	$11 \cdot 10^{17}$

Naive Bayesian Classifier

m	n	parámetros
3	10	32
5	20	104
10	50	509

$$C \in \{c_1, \dots, c_m\}$$ Target variable with ***m*** states/classes

$$X = \{X_1, \dots, X_n\}$$ Predictors in a ***n-dimensional*** space

.

[Bayes' Theorem (Predictands vs. Predictors)]

$$P(C|\{X_1,\dots,X_n\}) = \frac{P(\{X_1,\dots,X_n\}|C)\,P(C)}{P(\{X_1,\dots,X_n\})}$$

Naive Bayesian Classifier

[+] $P(X_i|\{X_j, C\}) = P(X_i|C)\;\forall\; j \neq i$

$$Arg_C[Max(P(\{X_1,\dots,X_n\}|C)\,P(C))] = Arg_C[Max(P(X_1|C)\dots P(X_n|C)\,P(C))]$$

How should be the graph for a Naive Bayesian Classifier?

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

```
## Defining the states of the variables:
variables.Names <- c("Outlook","Temperature","Humidity","Windy","Play Golf")
sample.Number <- c(1:14)
estados.O <- c("Rainy", "Overcast", "Sunny")
estados.T <- c("Hot", "Mild", "Cool")
estados.H <- c("Normal", "High")
estados.W <- c("True", "False")
estados.G <- c("Yes", "No")
Nclass <- c(3, 3, 2, 2, 2)
```

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

```
## Defining the table:
variables.Names <- c("Outlook","Temperature","Humidity","Windy","Play Golf")
sample.Number <- c(1:14)
data.table <- array(c("Rainy", "Rainy", "Overcast", "Sunny", "Sunny", "Sunny", "Overcast", "Rainy", "Rainy",
"Sunny", "Rainy", "Overcast", "Overcast", "Sunny",
"Hot", "Hot", "Hot", "Mild", "Cool", "Cool", "Cool", "Mild", "Cool", "Mild", "Mild", "Mild", "Hot", "Mild",
"High", "High", "High", "High", "Normal", "Normal", "Normal", "High", "Normal", "Normal", "Normal", "High",
"Normal", "High",
"False", "True", "False", "False", "False", "True", "True", "False", "False", "False", "True", "True", "False", "True",
"No", "No", "Yes", "Yes", "Yes", "No", "Yes", "No", "Yes", "Yes", "Yes", "Yes", "Yes", "No"), dim = c(14,5),
                  dimnames = list(event = sample.Number, variable = variables.Names))
```

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

- Define the corresponding graph

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

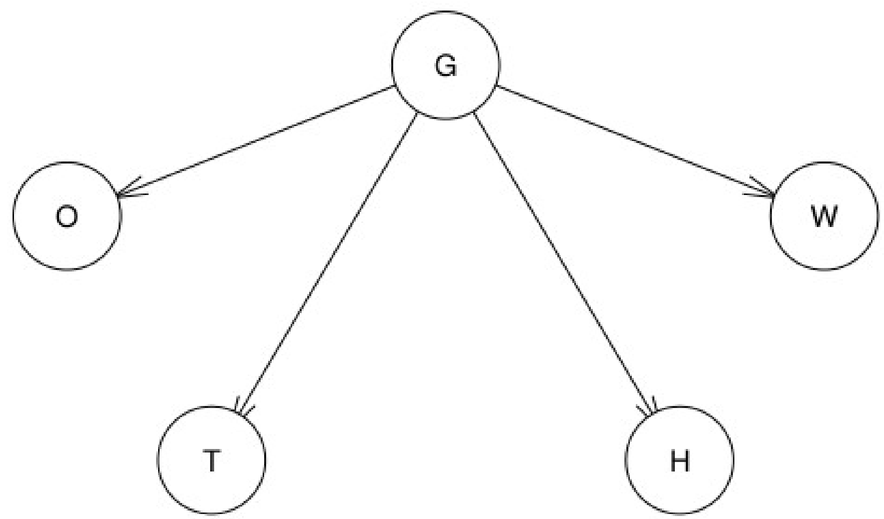
- Define the corresponding graph

```
## Defining the Graph:
dag <- empty.graph(nodes = c("O","T","H","W","G"))
dag <- set.arc(dag, from = "G", to = "O")
dag <- set.arc(dag, from = "G", to = "T")
dag <- set.arc(dag, from = "G", to = "H")
dag <- set.arc(dag, from = "G", to = "W")
modelstring(dag)
```

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

- Define the corresponding graph

```
## Defining the Graph:
dag <- empty.graph(nodes = c("O","T","H","W","G"))
dag <- set.arc(dag, from = "G", to = "O")
dag <- set.arc(dag, from = "G", to = "T")
dag <- set.arc(dag, from = "G", to = "H")
dag <- set.arc(dag, from = "G", to = "W")
plot(dag)
```

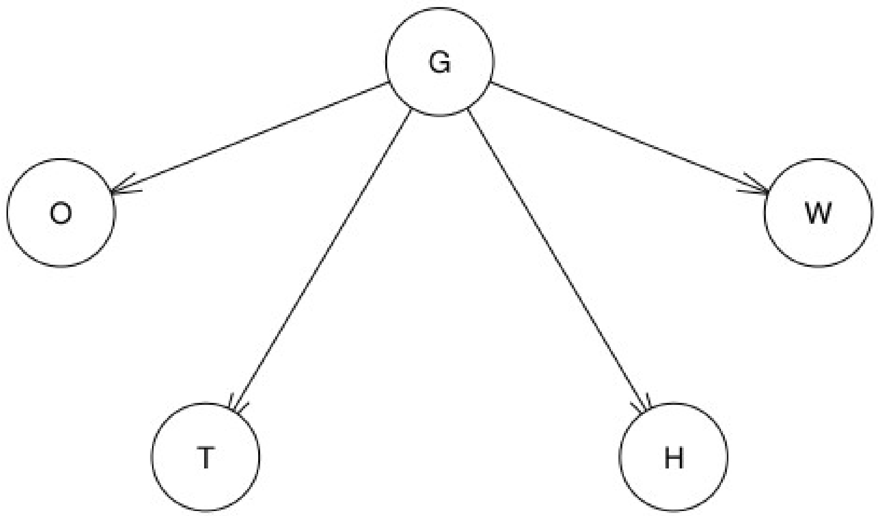


Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

- Define the probabilities, which probability functions should be defined?

- **Define the probabilities, which probability functions should be defined?**

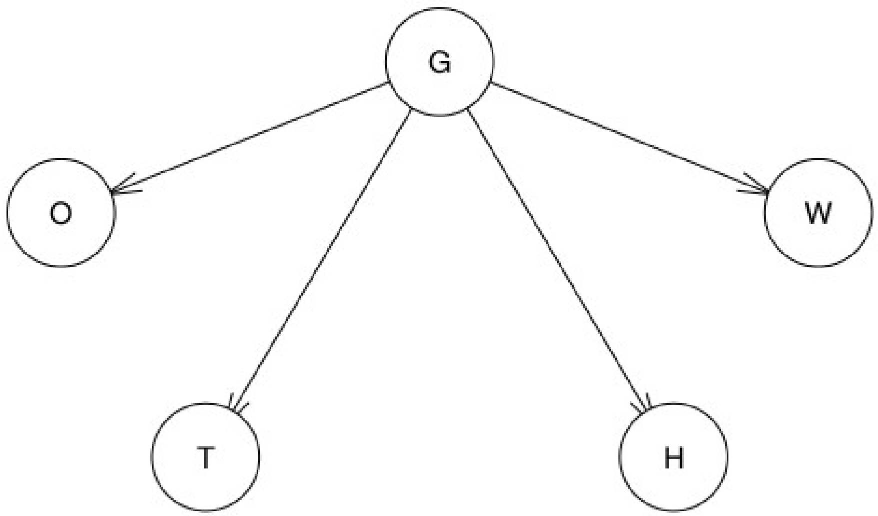
```
## [1] "[G][O|G][T|G][H|G][W|G]"
```



P(G) ← Play Golf
P(O|G) ← Outlook
P(T|G) ← Temperature
P(H|G) ← Humidity
P(W|G) ← Windy

- Define the probabilities, which probability functions should be defined?

```
## [1] "[G][O|G][T|G][H|G][W|G]"
```

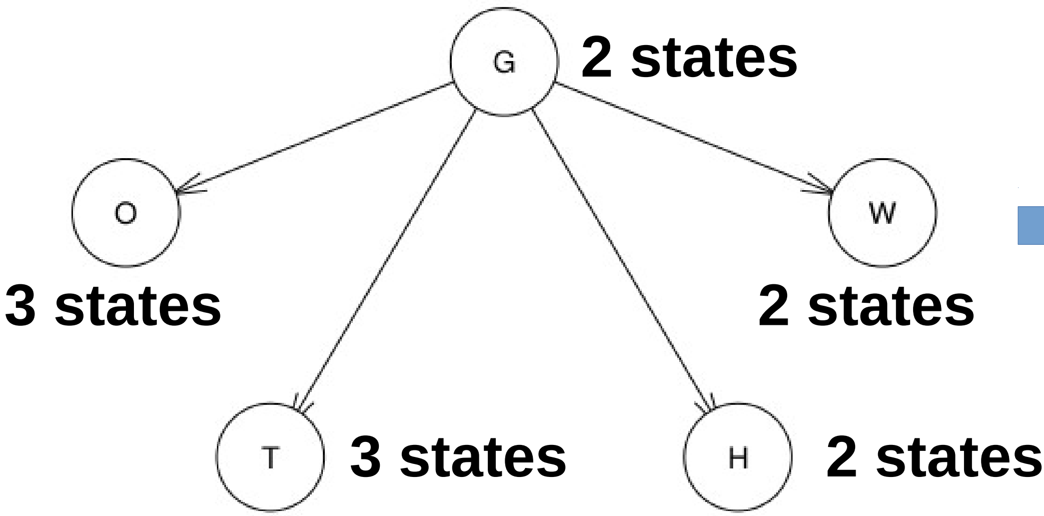


P(G) ← Play Golf
P(O|G) ← Outlook
P(T|G) ← Temperature
P(H|G) ← Humidity
P(W|G) ← Windy

P(G,O,T,H,W) ← 2*3*2*3*2 = 72 parameters

- Define the probabilities, which probability functions should be defined?

```
## [1] "[G][O|G][T|G][H|G][W|G]"
```



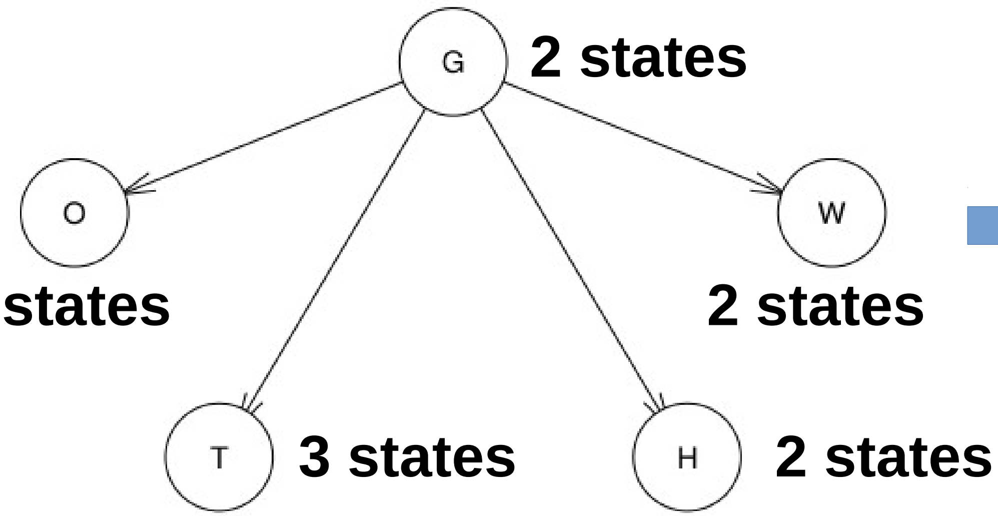
P(G) ← 2 parameters
P(O|G) ← 6 parameters
P(T|G) ← 6 parameters
P(H|G) ← 4 parameters
P(W|G) ← 4 parameters

P(G,O,T,H,W) ← 22 parameters

P(G,O,T,H,W) ← 2*3*2*3*2 = 72 parameters

- **Define the probabilities, which probability functions should be defined?**

```
## [1] "[G][O|G][T|G][H|G][W|G]"
```



P(G) ← 2 parameters
P(O|G) ← 6 parameters
P(T|G) ← 6 parameters
P(H|G) ← 4 parameters
P(W|G) ← 4 parameters

P(G,O,T,H,W) ← 22 parameters

Normalization

P(G,O,T,H,W) ← 13 parameters

P(G,O,T,H,W) ← 2*3*2*3*2 = 72 parameters

Normalization

P(G,O,T,H,W) ← 71 parameters

- Define the probabilities, which probability functions should be defined?

```
## [1] "[G][O|G][T|G][H|G][W|G]"
```

```
## Defining the probabilities:
G.prob <- array(c(length(which(data.table[,"Play Golf"] == "Yes")), length(which(data.table[,"Play Golf"] ==
"No")))/length(data.table[,"Play Golf"]), dim = 2, dimnames = list(G = estados.G))
O.prob <- array(data = 0, dim = c(Nclass[1],Nclass[5]), dimnames = list(O = estados.O, G = estados.G))
T.prob <- array(data = 0, dim = c(Nclass[2],Nclass[5]), dimnames = list(T = estados.T, G = estados.G))
H.prob <- array(data = 0, dim = c(Nclass[3],Nclass[5]), dimnames = list(H = estados.H, G = estados.G))
W.prob <- array(data = 0, dim = c(Nclass[4],Nclass[5]), dimnames = list(W = estados.W, G = estados.G))
for (g in 1:Nclass[5]){
  for (o in 1:Nclass[1]){
    O.prob[o,g] <- length(which(data.table[,"Play Golf"] == estados.G[g] & data.table[,"Outlook"] ==
estados.O[o]))/length(which(data.table[,"Play Golf"] == estados.G[g]))
  }
}
```

```
> O.prob
          G
O                Yes  No
  Rainy    0.2222222 0.6
  Overcast 0.4444444 0.0
  Sunny    0.3333333 0.4
```

- Define the probabilities, which probability functions should be defined?

```
## [1] "[G][O|G][T|G][H|G][W|G]"
```

```
## Defining the probabilities:
G.prob <- array(c(length(which(data.table[,"Play Golf"] == "Yes")), length(which(data.table[,"Play Golf"] ==
"No")))/length(data.table[,"Play Golf"]), dim = 2, dimnames = list(G = estados.G))
O.prob <- array(data = 0, dim = c(Nclass[1],Nclass[5]), dimnames = list(O = estados.O, G = estados.G))
T.prob <- array(data = 0, dim = c(Nclass[2],Nclass[5]), dimnames = list(T = estados.T, G = estados.G))
H.prob <- array(data = 0, dim = c(Nclass[3],Nclass[5]), dimnames = list(H = estados.H, G = estados.G))
W.prob <- array(data = 0, dim = c(Nclass[4],Nclass[5]), dimnames = list(W = estados.W, G = estados.G))
for (g in 1:Nclass[5]){
  for (o in 1:Nclass[1]){
    O.prob[o,g] <- length(which(data.table[,"Play Golf"] == estados.G[g] & data.table[,"Outlook"] ==
estados.O[o]))/length(which(data.table[,"Play Golf"] == estados.G[g]))
  }
}
```

Obtain the probabilities for the rest of variables

```
> O.prob
          G
O                 Yes  No
  Rainy     0.2222222 0.6
  Overcast  0.4444444 0.0
  Sunny     0.3333333 0.4
```

- Define the probabilities, which probability functions should be defined?

```
## [1] "[G][O|G][T|G][H|G][W|G]"
```

```
## Defining the probabilities:
G.prob <- array(c(length(which(data.table[,"Play Golf"] == "Yes")), length(which(data.table[,"Play Golf"] ==
"No")))/length(data.table[,"Play Golf"]), dim = 2, dimnames = list(G = estados.G))
O.prob <- array(data = 0, dim = c(Nclass[1],Nclass[5]), dimnames = list(O = estados.O, G = estados.G))
T.prob <- array(data = 0, dim = c(Nclass[2],Nclass[5]), dimnames = list(T = estados.T, G = estados.G))
H.prob <- array(data = 0, dim = c(Nclass[3],Nclass[5]), dimnames = list(H = estados.H, G = estados.G))
W.prob <- array(data = 0, dim = c(Nclass[4],Nclass[5]), dimnames = list(W = estados.W, G = estados.G))
for (g in 1:Nclass[5]){
  for (o in 1:Nclass[1]){
    O.prob[o,g] <- length(which(data.table[,"Play Golf"] == estados.G[g] & data.table[,"Outlook"] ==
estados.O[o]))/length(which(data.table[,"Play Golf"] == estados.G[g]))
  }
}
```

Obtain the probabilities for the rest of variables

```
> O.prob
          G
O                Yes No
  Rainy    0.2222222 0.6
  Overcast 0.4444444 0.0
  Sunny    0.3333333 0.4
```

```
> T.prob
      G
T             Yes No
  Hot  0.2222222 0.4
  Mild 0.4444444 0.4
  Cool 0.3333333 0.2
```

```
> H.prob
        G
H              Yes No
  Normal 0.6666667 0.2
  High   0.3333333 0.8
```

```
> W.prob
       G
W             Yes No
  True  0.3333333 0.6
  False 0.6666667 0.4
```

- Define the probabilities, which probability functions should be defined?

```
## [1] "[G][O|G][T|G][H|G][W|G]"
```

```
## Defining the probabilities:
G.prob <- array(c(length(which(data.table[,"Play Golf"] == "Yes")), length(which(data.table[,"Play Golf"] ==
"No")))/length(data.table[,"Play Golf"]), dim = 2, dimnames = list(G = estados.G))
O.prob <- array(data = 0, dim = c(Nclass[1],Nclass[5]), dimnames = list(O = estados.O, G = estados.G))
T.prob <- array(data = 0, dim = c(Nclass[2],Nclass[5]), dimnames = list(T = estados.T, G = estados.G))
H.prob <- array(data = 0, dim = c(Nclass[3],Nclass[5]), dimnames = list(H = estados.H, G = estados.G))
W.prob <- array(data = 0, dim = c(Nclass[4],Nclass[5]), dimnames = list(W = estados.W, G = estados.G))
for (g in 1:Nclass[5]){
  for (o in 1:Nclass[1]){
    O.prob[o,g] <- length(which(data.table[,"Play Golf"] == estados.G[g] & data.table[,"Outlook"] ==
estados.O[o]))/length(which(data.table[,"Play Golf"] == estados.G[g]))
  }
}
```

Obtain the Naive Bayesian Network

- Define the probabilities, which probability functions should be defined?

```
## [1] "[G][O|G][T|G][H|G][W|G]"
```

```
## Defining the probabilities:
G.prob <- array(c(length(which(data.table[,"Play Golf"] == "Yes")), length(which(data.table[,"Play Golf"] ==
"No")))/length(data.table[,"Play Golf"]), dim = 2, dimnames = list(G = estados.G))
O.prob <- array(data = 0, dim = c(Nclass[1],Nclass[5]), dimnames = list(O = estados.O, G = estados.G))
T.prob <- array(data = 0, dim = c(Nclass[2],Nclass[5]), dimnames = list(T = estados.T, G = estados.G))
H.prob <- array(data = 0, dim = c(Nclass[3],Nclass[5]), dimnames = list(H = estados.H, G = estados.G))
W.prob <- array(data = 0, dim = c(Nclass[4],Nclass[5]), dimnames = list(W = estados.W, G = estados.G))
for (g in 1:Nclass[5]){
  for (o in 1:Nclass[1]){
    O.prob[o,g] <- length(which(data.table[,"Play Golf"] == estados.G[g] & data.table[,"Outlook"] ==
estados.O[o]))/length(which(data.table[,"Play Golf"] == estados.G[g]))
  }
}
```

Obtain the Naive Bayesian Network

```
## Defining the Bayesian Network:
cpt <- list(G = G.prob, O = O.prob, T = T.prob, H = H.prob, W = W.prob)
bn <- custom.fit(dag, cpt)
str(bn)
```

```
> str(bn)
List of 5
 $ O:List of 4
  ..$ node    : chr "O"
  ..$ parents : chr "G"
  ..$ children: chr(0)
  ..$ prob    : 'table' num [1:3, 1:2] 0.222 0.444 0.333 0.6 0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ O: chr [1:3] "Rainy" "Overcast" "Sunny"
  .. .. ..$ G: chr [1:2] "Yes" "No"
  ..- attr(*, "class")= chr "bn.fit.dnode"
```

Naive Bayes

```
> dag

  Random/Generated Bayesian network

  model:
   [G][O|G][T|G][H|G][W|G]
  nodes:                                 5
  arcs:                                  4
    undirected arcs:                     0
    directed arcs:                       4
  average markov blanket size:           1.60
  average neighbourhood size:            1.60
  average branching factor:              0.80

  generation algorithm:                  Empty
```

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

- According to the weather, could we play golf today?

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

- According to the weather, could we play golf today?

```
## Exact inference:
jsex <- setEvidence(junction, nodes = c("O","T","H","W"), states = c("Overcast","Cool","Normal","True"))
querygrain(jsex, nodes = "G")$G
```

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

- According to the weather, could we play golf today?

```
## Exact inference:
jsex <- setEvidence(junction, nodes = c("O","T","H","W"), states = c("Overcast","Cool","Normal","True"))
querygrain(jsex, nodes = "G")$G
## Simulated inference:
set.seed(1)
cpquery(bn,event=(G=="Yes"),
        evidence=((O=="Overcast") & (T=="Cool") & (H=="Normal") & (W=="True")))
```

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

- Considering the whole sample, which is the accuracy of the classifier?

Outlook	Temperature	Humidity	Windy	Play Golf	Prediction
Rainy	Hot	High	False	No	No (0.17, 0.72)
Rainy	Hot	High	True	No	No (0.10, 0.93)
Overcast	Hot	High	False	Yes	Yes (1.00, 0.00)
Sunny	Mild	High	False	Yes	Yes (0.52, 0.50)
Sunny	Cool	Normal	False	Yes	Yes (0.95, 0.07)
Sunny	Cool	Normal	True	No	Yes (0.80, 0.19)
Overcast	Cool	Normal	True	Yes	Yes (1.00, 0.00)
Rainy	Mild	High	False	No	No (0.31, 0.71)
Rainy	Cool	Normal	False	Yes	Yes (0.79, 0.09)
Sunny	Mild	Normal	False	Yes	Yes (0.91, 0.12)
Rainy	Mild	Normal	True	Yes	Yes (0.48, 0.47)
Overcast	Mild	High	True	Yes	Yes (1.00, 0.00)
Overcast	Hot	Normal	False	Yes	Yes (1.00, 0.00)
Sunny	Mild	High	True	No	No (0.25, 0.66)

- Considering the whole sample, which is the accuracy of the classifier?

```
> confusionMatrix(as.factor(golf.predicted), as.factor(data.table[,5]))
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No   4   0
       Yes  1   9

               Accuracy : 0.9286
                 95% CI : (0.6613, 0.9982)
    No Information Rate : 0.6429
    P-Value [Acc > NIR] : 0.01807

                  Kappa : 0.8372

            Sensitivity : 0.8000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9000
             Prevalence : 0.3571
         Detection Rate : 0.2857
   Detection Prevalence : 0.2857
      Balanced Accuracy : 0.9000
```

The Naive Bayesian Classifier is included in the R-package ***e1071*** (see function ***naiveBayes***). Reproduce the results obtained with bnlearn considering the function ***naiveBayes***. Are there any significant difference?

Pros:

It is easy and fast to predict class of test data set. It also perform well in multi class prediction

When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.

It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction.

Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

$$C \in \{c_1, \ldots, c_m\}$$ Target variable with ***m*** states/classes

$$X = \{X_1, \ldots, X_n\}$$ Predictors in a ***n-dimensional*** space

[Bayes' Theorem (Predictands vs. Predictors)]

$$P(C|\{X_1,\ldots,X_n\}) = \frac{P(\{X_1,\ldots,X_n\}|C)\,P(C)}{P(\{X_1,\ldots,X_n\})}$$

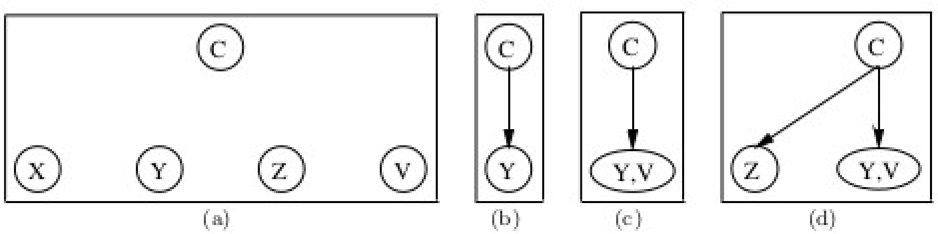
$+ \quad P(X_i|\{X_j, C\}) = P(X_i|C)\ \forall\ j \neq i$

Naive Bayesian Classifier

Exclusive states/classes
Predictors conditionally independent given the state.

Very restrictive hypothesis

Semi-Naive Bayesian Classifier



$$C \in \{c_1, \dots, c_m\}$$ Target variable with ***m*** states/classes

$$X = \{X_1, \dots, X_n\}$$ Predictors in a ***n-dimensional*** space

[Bayes' Theorem (Predictands vs. Predictors)]

$$P(C|\{X_1,\dots,X_n\}) = \frac{P(\{X_1,\dots,X_n\}|C)\,P(C)}{P(\{X_1,\dots,X_n\})}$$

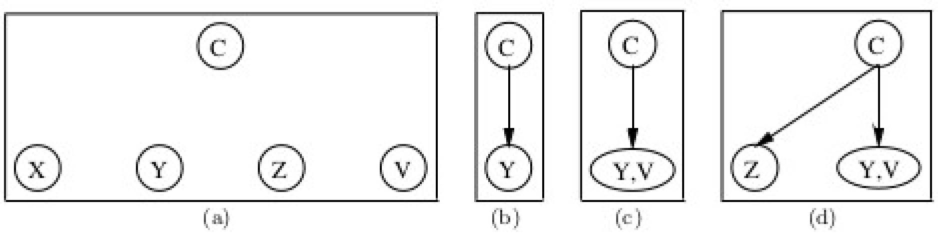
+ $$P(X_i|\{X_j, C\}) = P(X_i|C)\ \forall\ j \neq i$$

Naive Bayesian Classifier

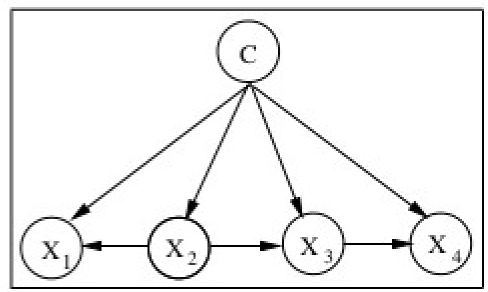
Exclusive states/classes
Predictors conditionally independent given the state.

Very restrictive hypothesis

Semi-Naive Bayesian Classifier



Tree Augmented-Naive (TAN)



$$C \in \{c_1, \ldots, c_m\}$$ Target variable with ***m*** states/classes

$$X = \{X_1, \ldots, X_n\}$$ Predictors in a ***n-dimensional*** space

Bayes' Theorem (Predictands vs. Predictors)

$$P(C|\{X_1,\ldots,X_n\}) = \frac{P(\{X_1,\ldots,X_n\}|C)\,P(C)}{P(\{X_1,\ldots,X_n\})}$$

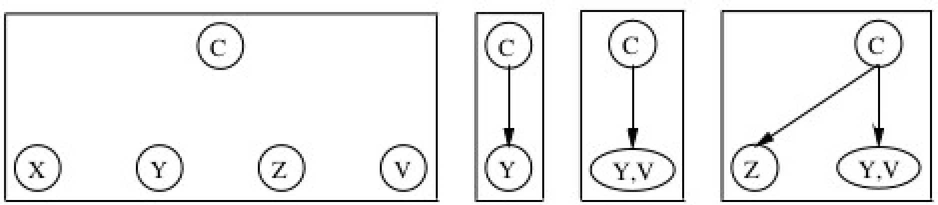
+ $P(X_i|\{X_j, C\}) = P(X_i|C)\ \forall\ j \neq i$

Naive Bayesian Classifier

Exclusive states/classes
Predictors conditionally independent given the state.

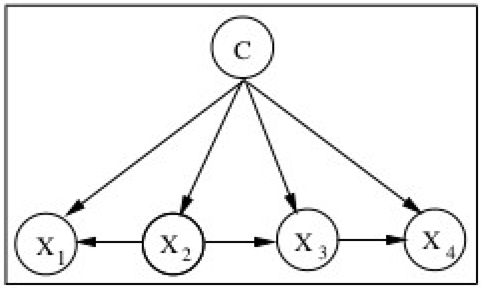
Very restrictive hypothesis

Semi-Naive Bayesian Classifier



Structural Improvement

Tree Augmented-Naive (TAN)



Extensions

$C \in \{c_1, \ldots, c_m\}$ Target variable with ***m*** states/classes

$X = \{X_1, \ldots, X_n\}$ Predictors in a ***n-dimensional*** space

.

[Bayes' Theorem (Predictands vs. Predictors)]

$$P(C|\{X_1,\ldots,X_n\}) = \frac{P(\{X_1,\ldots,X_n\}|C)P(C)}{P(\{X_1,\ldots,X_n\})}$$

$+$ $P(X_i|\{X_j, C\}) = P(X_i|C) \forall j \neq i$

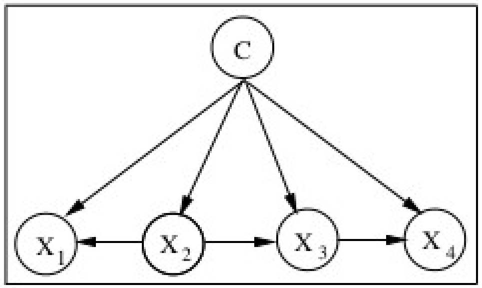
Naive Bayesian Classifier

Exclusive states/classes
Predictors conditionally independent given the state.

Very restrictive hypothesis

Particular case of Bayesian Networks

Tree Augmented-Naive (TAN)



Extensions

CATEGÓRICO

Overview Data Kernels Discussion Activity Download (30 KB) **New Kernel**

http://archive.ics.uci.edu/ml/datasets/Mushroom

Data Set Characteristics:	Multivariate	**Number of Instances:**	8124	**Area:**	Life
Attribute Characteristics:	Categorical	**Number of Attributes:**	22	**Date Donated**	1987-04-27
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	298439

Attribute Information: (classes: edible=e, poisonous=p)

cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,...
bruises: bruises=t,no=f
odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,...
...

```
mush <- read.csv("Data_mining/datasets/mushrooms.csv")
str(mush)
```

```
'data.frame':   8124 obs. of  23 variables:
 $ class                    : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
 $ cap.shape                : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
 $ cap.surface              : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
 $ cap.color                : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...
 $ bruises                  : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
 $ odor                     : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
 ...
```


Exercise (~1h)

CATEGÓRICO

https://www.kaggle.com/uciml/mushroom-classification/data

Overview Data Kernels Discussion Activity Download (30 KB) New Kernel

http://archive.ics.uci.edu/ml/datasets/Mushroom

Data Set Characteristics:	Multivariate	**Number of Instances:**	8124	**Area:**	Life
Attribute Characteristics:	Categorical	**Number of Attributes:**	22	**Date Donated**	1987-04-27
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	298439

Attribute Information: (classes: edible=e, poisonous=p)

cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,...
bruises: bruises=t,no=f
odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,...
...

Consider the Mushroom dataset
- How much parameters would be needed?
- How much parameters are obtained for the Naive Bayes?
- Train the model and evaluate the Bayesian Classifier obtained