Science Journal of Mathematics & Statistics **Published By Science Journal Publication**
ISSN: 2276-6324
http://www.sjpub.org/sjms.html

Research Article **Volume 2013 (2013), Article ID sjms-105, 22 Pages, doi: 10.7237/sjms/105**

Overview of Bayesian Network

Loc Nguyen

University of Technology, Ho Chi Minh city, Vietnam

Accepted 4th June, 2013

Abstract

Bayesian network is applied widely in machine learning, data mining, diagnosis, etc; it has a solid evidence-based inference which is familiar to human intuition. However Bayesian network causes a little confusion because there are many complicated concepts, formulas and diagrams relating to it. Such concepts should be organized and presented in clear manner so as to be easy to understand it. This is the goal of this report.
This report includes 4 main parts that cover principles of Bayesian network:
Part 1: Introduction to Bayesian network giving some basic concepts.
Part 2: Bayesian network inference discussing inference mechanism inside Bayesian network.
Part 3: Parameter learning tells us how to update parameters of Bayesian network.
Part 4: Structure learning surveys some main techniques to build up Bayesian network.

Keywords: Bayesian network, parameter learning, structure learning

1.0 Introduction

1.1. Bayesian rule

Bayesian network theory starts with the concept of Bayesian inference, a form of statistical method, which is responsible for collecting evidences to change the current belief in given hypothesis. The more evidences are observed, the higher degree of belief in hypothesis is. First, this belief was assigned an initial probability. When evidences were gathered enough, the hypothesis is considered trustworthy.

Bayesian inference was based on Bayesian rule with some special aspects:

$$P(H \mid E) = \frac{P(E \mid H) * P(H)}{P(E)} \quad (1.1)$$

Where H is probability variable denoting a hypothesis existing before evidence and E is also probability variable notating an observed evidence.

$P(H)$ is *prior probability* of hypothesis and $P(H \mid E)$ which is the conditional probability of H with given E, is called *posterior probability*. It tells us the changed belief in hypothesis when occurring evidence.

$P(E)$ is the probability of occurring evidence E together all mutually exclusive cases of hypothesis. If H and E are discrete, $P(E) = \sum_H P(E \mid H) * P(H)$ otherwise $f(E) = \int f(E \mid H) f(H) dH$ with H and E being continuous, f denoting probability density function.

When $P(E)$ is constant value, $P(E \mid H)$ is the *likelihood function* of H with fixed E. Likelihood function is often used to estimate parameters of probability distribution.

1.2. Bayesian network

Bayesian network (BN) is the directed acyclic graph (DAG) [1] in which the nodes (vertices) are linked together by directed edges (arcs); each edge expresses the dependence relationships between nodes. If there is the edge from node A to B, we call "A causes B" or "A is parent of B", in other words, B depends conditionally on A. So the edge $A \rightarrow B$ denotes parent-child, prerequisite or cause-effect relationship. Otherwise there is no edge between A and B, it asserts the conditional independence. Let $V=\{X_1, X_2, X_3, \ldots, X_n\}$ and E be a set of nodes and a set of edges, the BN is denoted as below:
$G=(V, E)$ where G is the DAG, V is a set of nodes and E is a set of edges

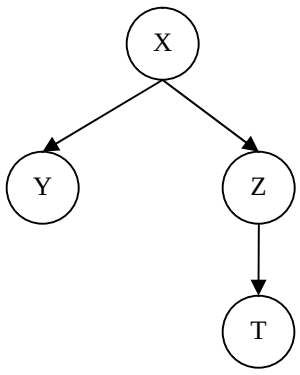


Figure 1.1: Bayesian network.

Note that node X_i is also random variable. In this paper the uppercase letter (for example X, Y, Z, etc.) denotes random variables or set of random variables; the lowercase letter (for example x, y, z, etc.) denote its instantiation. We should glance over other popular concepts.

- If there is an edge between X and Y ($X \rightarrow Y$ or $X \leftarrow Y$) then X and Y are called *adjacent* each other (or *incident* to the edge).
- Given k nodes $\{X_1, X_2, X_3, \ldots, X_k\}$ in such a way that every pair of node (X_i, X_{i+1}) are incident to the edge $X_i \rightarrow X_{i+1}$ where $1 \leq i \leq k-1$, all edges that connects such k nodes compose a *path* from X_1 to X_k denoted as $[X_1, X_2, X_3, \ldots, X_k]$ or $X_1 \rightarrow X_2 \rightarrow \ldots \rightarrow X_k$. The nodes $X_2, X_3, \ldots, X_{k-1}$ are called *interior* nodes of the path. The *sub-path* $X_m \rightarrow \ldots X_n$ is a path from X_m to X_n: $X_m \rightarrow X_{m+1} \rightarrow \ldots \rightarrow X_n$ where $1 \leq m < n \leq k$. The *directed cycle* is a path from a node to itself. The *simple path* is a path that has no directed cycle. The DAG is the graph that has no directed cycle.
- If there is a path from X to Y then X is called *ancestor* of Y and Y is called *descendant* of X. If Y isn't a descendant of X, Y is called *non-descendent* of X.
- If the direction isn't considered then edge and path are called *link* and *chain*, respectively. Link is denoted $A - B$. Chain is denoted $A - B - C$, for example.
- Graph G is a tree if every node except root has only one parent. G is called single-connected if there is only one chain (if exists) between two nodes. Almost BN (s) surveyed here are single-connected DAG (s).

The strength of dependence between two nodes is quantified by conditional probability table (CPT). In continuous case, CPT becomes conditional probability density function (CPD). So each node has its own local CPT. In case that a node has no parent, its CPT degenerates into prior probabilities. For example, suppose X_k is binary node and it has two parents X_i and X_j, the CPT (or CPD) of X_k which is the conditional probability $P(X_k \mid X_i, X_j)$ has eight entries:

$P(X_k=1 \mid X_i=1, X_j=1)$	$P(X_k=0 \mid X_i=1, X_j=1)$
$P(X_k=1 \mid X_i=1, X_j=0)$	$P(X_k=0 \mid X_i=1, X_j=0)$
$P(X_k=1 \mid X_i=0, X_j=1)$	$P(X_k=0 \mid X_i=0, X_j=1)$
$P(X_k=1 \mid X_i=0, X_j=0)$	$P(X_k=0 \mid X_i=0, X_j=0)$

It is asserted that if X_i is binary node and has n parents then its CPT has 2^{n+1} entries. However only 2^n entries are specified in practice due to $P(X_i=0 \mid \ldots) = 1 - P(X_i=1 \mid \ldots)$ when X_i is binary. In case that X_i has k possible values, each CPT has k^n entries.

Example 1.1: Suppose event "*cloudy*" is cause of event "*rain*". Events "*rain*" and "*sprinkler*" which in turn is cause of "*grass is wet*" [5] [7]. So we have three causal-effect relationships of: *1*-cloudy to rain, *2*- rain to wet grass, *3*-sprinkler to wet grass. This model is expressed below by BN with four nodes and three arcs corresponding to four events and three relationships. Every node has two possible values *True* (*1*) and *False* (*0*) together its CPT.

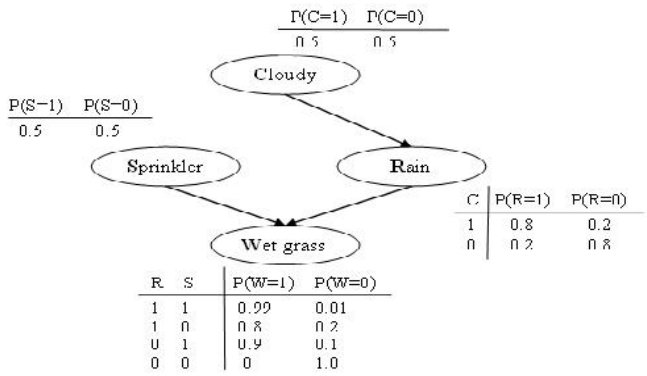


Figure 1.2: Bayesian network with CPT (s) in example 1.1.

Let PA_i be the set of parents of node X_i, the *joint probability distribution* of whole BN is defined as product of CPT(s) or CPD(s) in continuous case of all nodes.

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid PA_i) \quad (1.2)$$

So BN is represented by its joint probability distribution P and its DAG.
(G, P) where $G=(V, E)$ is a DAG and P is joint probability distribution.

Suppose Ω_i is the subset of PA_i such that X_i must depend conditionally and directly on every variable in Ω_i. In other words, there is always an edge from each node in Ω_i to X_i and no intermediate node between them. This criterion is called as Markov condition which will be discussed later. The joint probability P is re-written as below:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \Omega_i) \quad (1.3)$$

Back the "wet grass" BN in example 1.1, the joint probability is:
$P(C, R, S, W) = P(C)*P(R)*P(R|C)*P(S|C)*P(W|C,R,S)$

We have $P(S \mid C) = P(S)$ due to the conditional independence assertion about variables S and C. Furthermore, because S is intermediate node between C and W, we should remove C from $P(W \mid C, R, S)$, hence, $P(W \mid C, R, S) = P(W \mid R, S)$. In short, the joint probability is shown below:

$P(C, R, S, W) = P(C)*P(S)*P(R|C)*P(W|R,S)$

1.3. Bayesian network reference

Using Bayesian reference, we need to compute the posterior probability of each hypothesis node in network. In general,

the computation based on Bayesian rule is known as the inference in Bayesian network.

Reviewing example 1.1, suppose W becomes evidence variable which is observed the fact that the grass is wet, so, W has value *1*. There is request for answering the question: how to determine which cause (sprinkler or rain) is more possible for wet grass. Hence, we will calculate two posterior probabilities of S (*=1*) and R (*=1*) in condition W (*=1*). These probabilities are also called *explanations* for W.

$$P(R=1|W=1)=\frac{\sum_{C,S}P(C,R=1,S,W=1)}{\sum_{C,R,S}P(C,R,S,W=1)}=0.581$$

$$P(S=1|W=1)=\frac{\sum_{C,R}P(C,R,S=1,W=1)}{\sum_{C,R,S}P(C,R,S,W=1)}=0.614$$

Because of $P(R=1|W=1) < P(S=1|W=1)$, it is concluded that sprinkler is the most likely cause of wet grass. Note that two above formulas which are also variants of Bayesian rule (see formula 1.1) will be surveyed more carefully in the "Bayesian network inference" section.

1.4. Markov condition and Markov equivalence

The inference in BN becomes complex and ineffective when the size of BN is large. Suppose BN has n binary nodes. In the worst case, each node has *n-1* parents, thus, the joint probability has $n*2^n$ entries. There is a boom of CPT (s). There is a restrictive criterion so-called Markov condition that makes the relationships (also CPT) among nodes simpler.

Given Bayesian network (G, P) and three sets of nodes:
$A=\{X_i,...,X_j\}$, $B=\{X_k,...,X_l\}$ and $C=\{X_m,...,X_n\}$:

- The denotation $I_P(A,B)$ or $I_G(A,B)$ indicates that A and B are independent.
- The denotation $I_P(A,B|C)$ or $I_G(A,B|C)$ indicates that A and B conditional independent given C.

Let (G, P) be Bayesian network, Markov condition is stated that every node X is conditional independent from its non-descendants given its parent. In other word node X is only Dependent on its previous nodes (directed parents).

$\forall X \in E$, $I_P(X, N_X | PA_X)$

Where E is the set of edges in G, N_X and PA_X are set of non-descendants of X and parents of X, respectively.

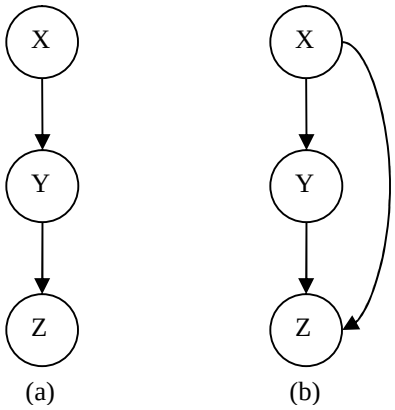


Figure 1.3: Example about Markov condition: (a) satisfy, (b) not satisfy

Because inference and structure learning algorithms are based on Markov condition, please pay attention to it.

Suppose Bayesian (G, P) satisfies Markov condition, it is necessary to find out or check whether a node (or a set of nodes) Z that separates a node (or a set of nodes) X from another node (or a set of nodes) Y. It means that whether there is $I_P(X, Y | Z)$. In this case, X and Y are called *d-separated* by Z.

There are some important concepts that constitute the d-separation concept:

The chain $X \to Z \to Y$ or $X \leftarrow Z \leftarrow Y$ is called serial path.
The chain $X \to Z \leftarrow Y$ is called convergent.
The chain $X \leftarrow Z \to Y$ is called divergent.
The chain $X–Z–Y$ is called uncoupled chain if X and Y aren't adjacent.

Of course, serial path, convergent path and divergent path are uncoupled chain.

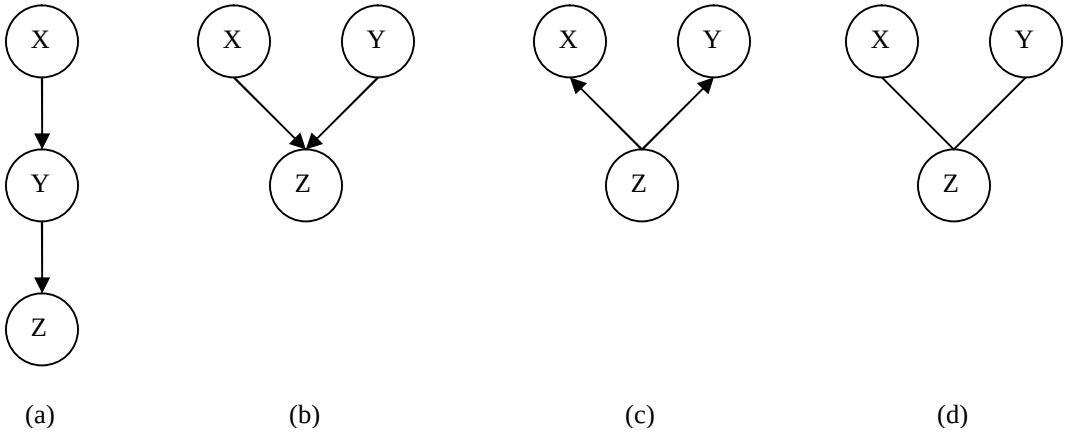


Figure 1.4: Serial path (a), convergent path (b), divergent path (c), and uncoupled chain (d).

Let *X, Y* and *Z* be sets of nodes where *X, Y, Z* $\subseteq$ *V*. Given the chain *p* between *X* and *Y*, *p* is blocked by *Z* if and only if one of two conditions is satisfied:

There is an intermediate node *M* $\in$ *Z* on *p* so that all edges on *p* incident to *M* are serial or divergent at *M*.

There is an intermediate node *M* on *p* so that:

M $\notin$ *Z* and all descendants of M $\notin$ *Z*

All edges op *p* incident to *M* are convergent.

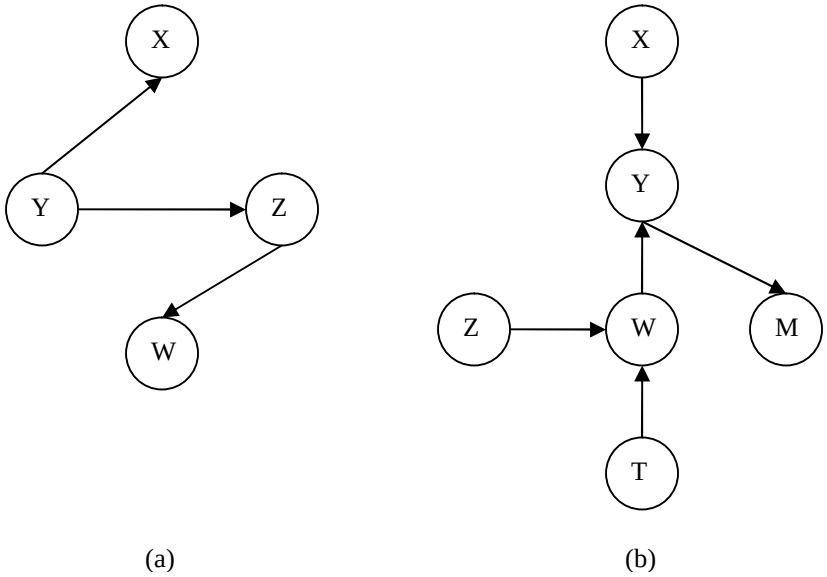


Figure 1.5: The chain *X*–*Y*–*Z*–*W* in (a) is blocked by {*Y, Z*} because edges incident to *Y* are divergent at *Y*.

The chain *X*–*Y*–*Z*–*W*–*T* in (b) is blocked by {*Z, W*} because there is such a node *Y* on chain that *Y* $\notin$ {*Z, W*}, its descendant *M* $\notin$ {*Z, W*}, and edges incident to *Y* are convergent at *Y*.

X and *Y* are d-separated by *Z* if all chains between *X* and *Y* are blocked by *Z*. *Z* is also called a *d*-separation of *G*.

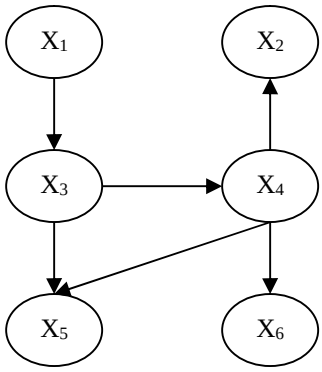


Figure 1.6: $\{X_1, X_2\}$ is *d*-separated from $\{X_5, X_6\}$ by $\{X_3, X_4\}$.

BN (s) which have the same set of nodes are Markov equivalent if and only if they have same *d*-separations. In other words, BN (s) that are Markov equivalent have the same independences. Given $G_1=(V, E_1)$ and $G_2=(V, E_2)$, we

Have:

$$\forall A, B, C \subseteq V, I_{G_1}(A, B \mid C) \Rightarrow I_{G_2}(A, B \mid C)$$

Where *A, B, C* are mutually disjoint sub-set of *V*. Note that G_1 and G_2 must be DAG and satisfy Markov condition.

The goal of giving "Markov equivalent" concept is to represent BN (s) that have the same structure and joint probability. So the representation of such BN (s) is called *Markov equivalent class* which is also a Bayesian network. In conclusion, Markov equivalence divides all DAG (or BN) into disjoint *Markov equivalent classes*. In practice, Markov equivalent class is often find out or surveyed instead of considering many BN (s).

2.0 Bayesian network inference

2.1. Simple inference

The essence of Bayesian reference is to compute the posterior probabilities of nodes given evidences. Note that evidences or conditions are also nodes which are observed and have concrete values. Back example 1.1 "wet grass". The posterior probability of $R = 1$ (rain) given $W = 1$ (wet grass) is the ratio of the marginal probability of *R, W* over *C, S* to the marginal probability of *W* over *C, R, S*.

$$P(R=1 \mid W=1) = \frac{P(R=1, W=1)}{P(W=1)} = \frac{\sum_{C,S} P(C, R=1, S, W=1)}{\sum_{C,R,S} P(C, R, S, W=1)}$$

Let $V=\{X_1, X_2, \ldots X_n\}$ be a whole set of nodes. Let $D=\{X_m, X_u, \ldots, X_n\}$ be a set of evidences, $D \subset V$. Let $d=(x_m, x_u, \ldots, x_n)$ be the instantiation of *D*. In general case, the marginal probability of $X_k=x_k$ is:

$$P(X_k = x_k, D = d) = \sum_{V-\{X_k, D\}} P(X_1, X_2, \ldots, x_k, \ldots, d, \ldots, X_n)$$

Where $P(X_1, X_2, \ldots, X_n)$ is the global joint probability.
The marginal probability of $D = d$ is:

$$P(D = d) = \sum_{V-D} P(X_1, X_2, \ldots, d, \ldots, X_n)$$

The probability of $X_k = k$ given $D = d$ is:

$$P(X_k = x_k \mid D=d) = \frac{P(X_k = x_k, D=d)}{P(D=d)} = \frac{\sum_{V-\{X_k, D\}} P(X_1, X_2, \ldots, x_k, \ldots d, \ldots, X_n)}{\sum_{V-D} P(X_1, X_2, \ldots d, \ldots, X_n)} \quad (2.1)$$

The above formula is the basic idea of simple inference. Note that it is also a variant of Bayesian rule (see formula 1.1). But the cost of computing it based on marginal probabilities is very high because there are a huge number of numeric operations such as additions and multiplications in computation expression. If the joint probability has many terms, brute force method for determining combinations of such operations is impossible. There are three main approaches that improve this computation:

Taking advantage of Markov condition: Pearl's message propagation [1] [4] is well-known algorithm.

OR-gate model inference [1] which simulates OR-gate electronic circuit.

Reducing the amount of numeric operations computed in marginal probability. Optimal factoring [1] is the well-known technique.

2.2. Pearl's message propagation algorithm

Suppose Bayesian network is DAG $G=(E, V)$ which is a tree having only one root. Given a set of evidence nodes $D \subseteq V$; every node in *D* has concrete value. Let D_X is the sub-set of *D* including *X* and descendants of *X* and let N_X be the sub-set of *D* including *X* and non-descendant of *X*. Let C_X and PA_X be children and parents of *X*, respectively. Let *R* be root node. Let *O* be evidence node, $O \in D$.

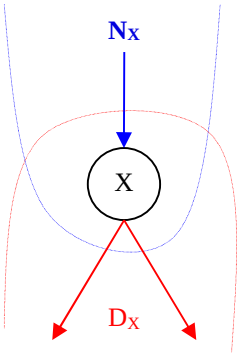


Figure 2.1: X, DX and NX. Note that NX is green and DX is red.

The essence of inference is to compute the posterior probability $P(X/D)$ for every X. We have:

$$P(X \mid D) = P(X \mid D_X, N_X)$$
$$= \frac{P(D_X, N_X \mid X)P(X)}{P(D_X, N_X)} \text{ (due to Bayes' rule)}$$
$$= \frac{P(D_X \mid X)P(N_X \mid X)P(X)}{P(D_X, N_X)} \text{ (D_X and N_X are conditionally independent given X)}$$
$$= P(D_X \mid X)\frac{P(N_X \mid X)P(X)}{P(N_X)}\frac{P(N_X)}{P(D_X, N_X)}$$
$$= \sim P(D_X \mid X)P(X \mid N_X)$$

Where $\sim = \frac{P(N_X)}{P(D_X, N_X)}$ is the constant independent from X.

Let $\lambda(X)$ and $\pi(X)$ be $P(D_X/X)$ and $P(X/N_X)$, respectively.

$P(X/D) = \alpha\lambda(X)\,\pi(X)$ (2.2)

The $\lambda(X)$ and $\pi(X)$ are called λ value and π value of X, respectively.
For each child Y of X, let $\lambda_Y(X)$ be λ message that connects X and Y. Note that $\lambda_Y(X)$ is conditional probability of D_Y given X.

$$\lambda_Y(X) = P(D_Y \mid X) = \sum_Y P(D_Y \mid Y)P(Y \mid X) = \sum_Y \lambda(Y)P(Y \mid X) \quad (2.3)$$

For each parent Z of X, let $\pi_X(Z)$ be π message that connects Z and X. Note that $\pi_X(Z)$ is conditional probability of X given N_X.

$$\pi_X(Z) = P(Z \mid N_X)$$
$$= P(Z \mid N_Z, \bigcap_{K \in C_Z - \{X\}} D_K) \text{ (where } C_Z - \{X\} \text{ is the set of Z's children except X)}$$
$$= \frac{P(N_Z, \bigcap_{K \in C_Z - \{X\}} D_K \mid Z)P(Z)}{P(N_Z, \bigcap_{K \in C_Z - \{X\}} D_K)} \text{ (Bayes' rule)}$$
$$= \frac{P(N_Z \mid Z)P(\bigcap_{K \in C_Z - \{X\}} D_K \mid Z)P(Z)}{P(N_Z, \bigcap_{K \in C_Z - \{X\}} D_K)} \text{ (because Z and } C_Z - \{X\} \text{ are conditional independent given Z)}$$
$$= \frac{P(Z \mid N_Z)P(N_Z)P(\bigcap_{K \in C_Z - \{X\}} D_K \mid Z)P(Z)}{P(Z)P(N_Z, \bigcap_{K \in C_Z - \{X\}} D_K)}$$
$$= P(Z \mid N_Z)P(\bigcap_{K \in C_Z - \{X\}} D_K \mid Z)\frac{P(N_Z)}{P(N_Z, \bigcap_{K \in C_Z - \{X\}} D_K)}$$
$$= kP(Z \mid N_Z)P(\bigcap_{K \in C_Z - \{X\}} D_K \mid Z) \text{ (where } k = \frac{P(N_Z)}{P(N_Z, \bigcap_{K \in C_Z - \{X\}} D_K)} \text{ is the constant independent from X, Z)}$$
$$= k\pi(Z)\prod_{K \in C_Z - \{X\}} P(D_K \mid Z) \text{ (because X's children are mutually independent)}$$
$$= k\pi(Z)\prod_{K \in C_Z - \{X\}} \lambda_K(Z)$$
$$\approx \pi(Z)\prod_{K \in C_Z - \{X\}} \lambda_K(Z) \quad (2.4)$$

Don't worry about $\pi_X(Z)$ is proportioned to $\pi(Z)\prod_{K \in C_Z - \{X\}} \lambda_K(Z)$ by removing constant k because the posterior probability $P(X/D)$ itself is also proportioned to $\lambda(X)$ and $\pi(X)$ via constant α. These constants will be eliminated when $P(X/D)$ is normalized. Now we have:

Value $\lambda(X) = P(D_X/X)$

Message $\lambda_Y(X) = P(D_Y \mid X) = \sum_Y \lambda(Y)P(Y \mid X)$ for each $Y \in C_X$

Value $\pi(X) = P(X/N_X)$

Message $\pi_X(Z) = P(Z \mid N_X) = \pi(Z)\prod_{K \in C_Z - \{X\}} \lambda_K(Z)$ for each $Z \in PA_X$.

The λ and π values are updated according to λ and π messages. Whenever evidence $O \in D$ occurs, Pearl's algorithm propagates downwards π message and propagates upwards λ message in order to update λ value and π value of each variable X so that the posterior probability $P(X/D)$ can be computed. The process of upwards-downwards propagation spreads over all variables of network.

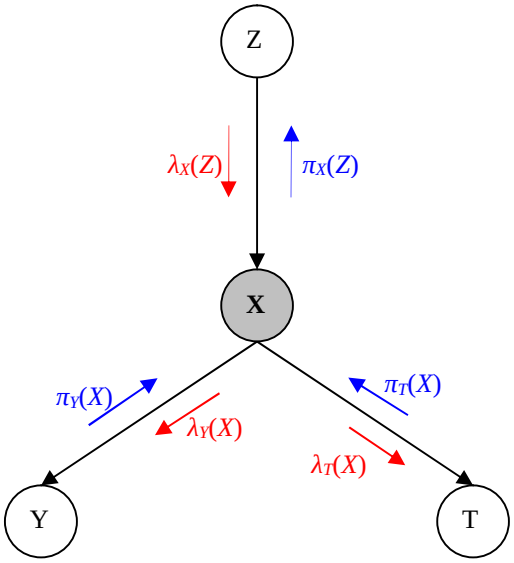


Figure 2.2: Pearl propagation algorithm (*X* is focused node).

Please pay attention to following notices when updating λ value and π value at certain variable X:
If $X \in D$ and suppose *X*'s instantiation (value) is *x* then:

$\lambda(X=x) = P(x|x) = 1$ due to $X \in D_X$ and Markov condition. So $\lambda(X \neq x) = 0$
$\pi(X=x) = P(x|x) = 1$ due to $X \in N_X$ and Markov condition. So $\pi(X \neq x) = 0$
$P(X=x|D) = 1$ and $P(X \neq x|D) = 0$.

If $X \notin D$ and *X* is leaf then:

$\lambda(X) = P(\emptyset|X) = 1$ due to $D_X = \emptyset$
$\pi(X)$ is computed as if *X* were intermediate variable.
$P(X|D) = \alpha\pi(X)$

If $X \notin D$ and *X* is root then:

$\lambda(X)$ is computed as if *X* were intermediate variable.
$\pi(X) = P(X|\emptyset) = P(X)$
$P(X|D) = \alpha\lambda(X)P(X)$

If $X \notin D$ and *X* is intermediate variable then:

$$\lambda(X) = P(D_X \mid X) = P\left(\bigcap_{Y \in C_X} D_Y \mid X\right) = \prod_{Y \in C_X} P(D_Y \mid X) = \prod_{Y \in C_X} \lambda_Y(X)$$

(Because X's children are mutually independent)

$$\pi(X) = P(X \mid N_X) = \sum_Z P(X \mid Z) P(Z \mid N_X) = \sum_Z P(X \mid Z) \pi_X(Z)$$

Where *Z* is parent of *X*.
$P(X|D) = \alpha\lambda(X)\pi(X)$

Pseudo-code for Pearl's algorithm shown below includes three functions:
Function "*void init*" initialize π value for every node. At that time the set of evidence nodes *D* is empty.
Function "*void update*" is executed whenever evidence node *O* occurs. This function adds *O* to set *D*, propagates upwards λ message over all parents of *O* by calling function "*void propagate_up*", and propagates down π message over all children of *O* by calling function "*void propagate_down*".
Function "*void propagate_up_λ_message*" computes λ value and posterior probability of current node, and continues to propagate upwards and downwards λ, π messages by calling itself and function "*void propagate_down_π_message*". Process of propagation stops when there is no node to be propagated.
Function "*void propagate_down_π_message*" computes π value and posterior probability of current node, and continues to propagate downwards π message by calling itself. Process of propagation stops when there is no node to be propagated.

```
void init(G, D)
{
D=Ø;
for each X ∈ V
{
λ(X) = 1;
//due to D = Ø
for each parent Z of X
 //propagate up λ message
λX(Z) = 1;
// due to D = Ø
}
P(R|D) = P(R);
//posterior probability of root node
π(R) = P(R);
 // π value

for each child K of R
 //browse root's children
propagate_up_π_message(R, K);
}

void update(O, o)
{
D = D ∪ O
λ(O=o) = π(O=o) = P(O=o|D) = 1;
//due to O ∈ D
λ(O ≠ o) = π(O ≠ o) = P(O ≠ o|D) = 1;
//due to O ∉ D

if O ≠ R and O's parent Z ∉ D
 // O isn't root and parent of O doesn't belong to D
propagate_up_ λ_message(O, Z);

for each child K of O such that K ∉ D
```

//browse O's children propagate_up_π_message(O, K);}

void propagate_up_λ_message(*Y, X*)
{
$\lambda_Y(X) = \sum_Y \lambda(Y)P(Y \mid X)$;
//Y propagate upwards λ message
$\lambda(X) = \prod_{Y \in C_X} \lambda_Y(X)$;
//update λ value
P(X|D)= αλ(X)π(X);
//compute posterior probability of X
normalize P(X|D);
//eliminate constant α

if X ≠ R and X's parent Z ∉ D
propagate_up_ λ_message(X, Z);

for each child K of X such that K ≠ Y and K ∉ D
//browse O's children propagate_up_π_message(X, K);
}

void propagate_down_π_message(Z, X)
{
$\Re_X(Z) = \Re(Z) \prod_{K \in C_Z - \{X\}} \lambda_K(Z)$;
//Y propagate downwards π

message
$\Re(X) = \sum_Z P(X \mid Z)\Re_X(Z)$; *//update π value*
P(X|D)= αλ(X)π(X);
//compute posterior probability of X normalize P(X|D);
//eliminate constant α

for each child K of X such that K ∉ D //browse O's children
propagate_up_π_message(X, K);
}

Example 2.1: Given Bayesian network shown in figure 2.3, suppose evidence *X* has value *1*. Hence, we need to compute posterior probabilities of *T, Y, Z* in condition *X=1*. Firstly, function "*void init*" is called to initialize network.

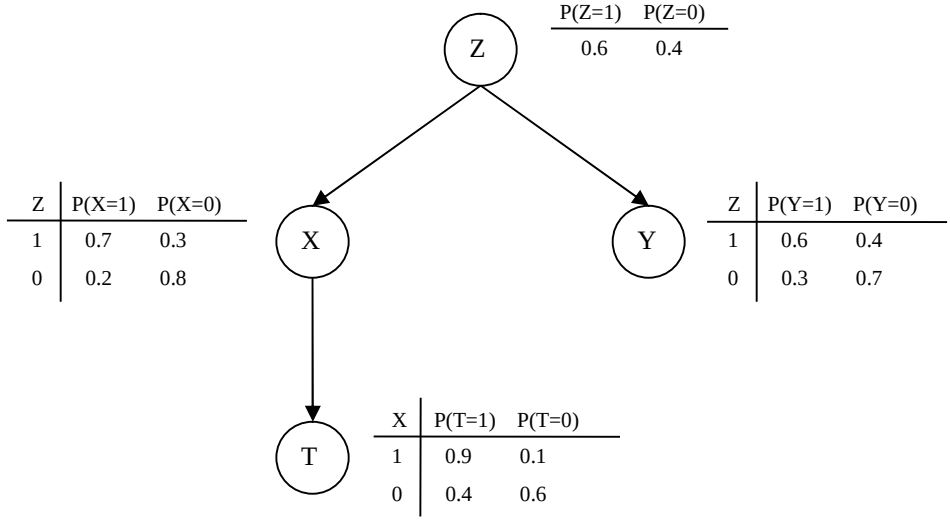


Figure 2.3: Bayesian network with CPT (s)

Function *init*(*G,D*) is executed:
D = Ø
λ(Z= 1) = λ(Z = 0) = 1
λ(X = 1) = λ(X = 0) = 1
λ(Y = 1) = λ(Y = 0) = 1
λ(T = 1) = λ(T = 0) = 1

λ_X(Z=1) = λ_X(Z=0) = 1
λ_Y(Z=1) = λ_Y(Z=0) = 1
λ_T(X=1) = λ_T(X=0) = 1

P(Z=1|d) = P(Z=1) = 0.6. Note that let *d* be instantiation of *D*
P(Z=0|d) = P(Z=0) = 0.4
π(Z=1) = P(Z=1) = 0.6
π(Z=0) = P(Z=0) = 0.4

Calling *propagate_down_π_message*(*Z, X*)
Calling *propagate_down_π_message*(*Z, Y*)

Then, function *propagate_down_π_message*(*Z, X*) is executed:
$\pi_X(Z=1) = \pi(Z=1)\ \lambda_X(Z=1) = 1*0.6 = 0.6$
$\pi_X(Z=0) = \pi(Z=0)\ \lambda_X(Z=0) = 1*0.4 = 0.4$

$\pi(X=1) = P(X=1|Z=1)\ \pi_X(Z=1) + P(X=1|Z=0)\ \pi_X(Z=0) = 0.7*0.6 + 0.2*0.4 = 0.5$
$\pi(X=0) = P(X=0|Z=1)\ \pi_X(Z=1) + P(X=0|Z=0)\ \pi_X(Z=0) = 0.3*0.6 + 0.8*0.4 = 0.5$

$P(X=1) = \alpha\,\lambda(X=1)\,\pi(X=1) = \alpha 1*0.5 = \alpha 0.5$
$P(X=0) = \alpha\,\lambda(X=0)\,\pi(X=0) = \alpha 1*0.5 = \alpha 0.5$

$$P(X=1) = \frac{\sim 0.5}{\sim 0.5 + \sim 0.5} = 0.5$$
$$P(X=0) = \frac{\sim 0.5}{\sim 0.5 + \sim 0.5} = 0.5$$

Calling *propagate_down_π_message*(*X, T*)
Then, function *propagate_down_π_message*(*X, T*) is executed:
$\pi_T(X=1) = \pi(X=1) = 0.5$
$\pi_T(X=0) = \pi(X=0) = 0.5$

$\pi(T=1) = P(T=1|X=1)\ \pi_T(X=1) + P(T=1|X=0)\pi_T(X=0) = 0.9*0.5 + 0.4*0.5 = 0.65$
$\pi(T=0) = P(T=0|X=1)\ \pi_T(X=1) + P(T=0|X=0)\pi_T(X=0) = 0.1*0.5 + 0.6*0.5 = 0.4$

$P(T=1) = \alpha\,\lambda(T=1)\pi(T=1) = \alpha 1*0.65 = \alpha 0.65$
$P(T=0) = \alpha\,\lambda(T=0)\pi(T=0) = \alpha 1*0.4 = \alpha 0.4$

$$P(T=1) = \frac{\sim 0.65}{\sim 0.65 + \sim 0.4} = 0.62$$
$$P(T=0) = \frac{\sim 0.4}{\sim 0.65 + \sim 0.4} = 0.38$$

Then function *propagate_down_π_message*(*Z, Y*) is executed:
$\pi_Y(Z=1) = \pi(Z=1)\lambda_Y(Z=1) = 1*0.6 = 0.6$
$\pi_Y(Z=0) = \pi(Z=0)\lambda_Y(Z=0) = 1*0.4 = 0.4$

$\pi(Y=1) = P(Y=1|Z=1)\pi_X(Z=1) + P(Y=1|Z=0)\pi_X(Z=0) = 0.6*0.6 + 0.3*0.3 = 0.45$
$\pi(Y=0) = P(Y=0|Z=1)\pi_X(Z=1) + P(Y=0|Z=0)\pi_X(Z=0) = 0.3*0.4 + 0.8*0.7 = 0.68$

$P(Y=1) = \alpha\lambda(Y=1)\pi(Y=1) = \alpha 1*0.45 = \alpha 0.45$
$P(Y=0) = \alpha\lambda(Y=0)\pi(Y=0) = \alpha 1*0.68 = \alpha 0.68$

$$P(Y=1) = \frac{\sim 0.45}{\sim 0.45 + \sim 0.68} = 0.4$$
$$P(Y=0) = \frac{\sim 0.68}{\sim 0.45 + \sim 0.68} = 0.6$$

The initialized Bayesian network is shown below:

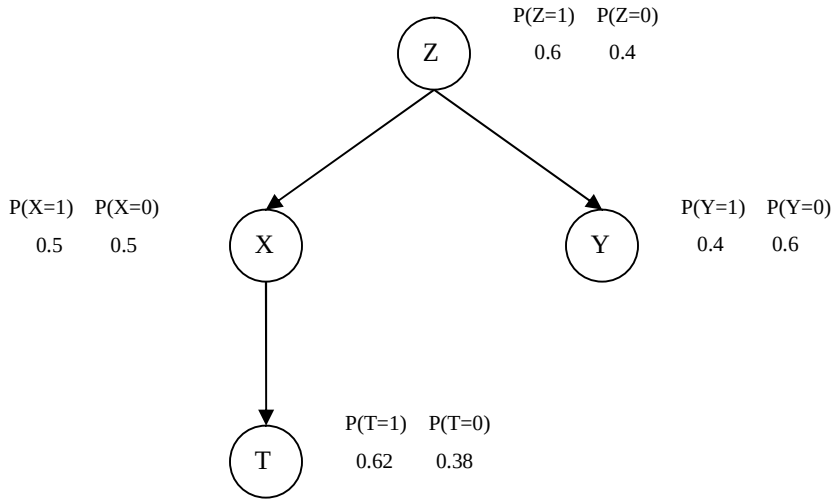


Figure 2.4: Initialized Bayesian network

When *X* becomes evidence and gains value *1*, the function *update*(*X, 1*) is called:
$D = D \cup X = \{X\}$
Because *d* is instantiation of *D*, we have $d = \{X=1\}$
$\lambda(X=1) = \pi(X=1) = P(X=1|d) = 1$
$\lambda(X=0) = \pi(X=0) = P(X=0|d) = 0$

Calling *propagate_up_λ_message*(*X, Z*)

Calling *propagate_down_π_message* (*X, T*)

Then, function *propagate_up_λ_message*(*X, Z*) is executed:
$\lambda_X(Z=1) = \lambda(X=1)P(X=1|Z=1) + \lambda(X=0)P(X=0|Z=1) = 1*0.7 + 0*0.3 = 0.7$
$\lambda(Z=1) = \lambda_X(Z=1)\lambda_Y(Z=1) = 0.7*1 = 0.7$
$P(Z=1|d) = \alpha\lambda(Z=1)\pi(Z=1) = \alpha 0.7*0.6 = \alpha 0.42$

$\lambda_X(Z=0) = \lambda(X=1)P(X=1|Z=0) + \lambda(X=0)P(X=0|Z=0) = 1*0.2 + 0*0.8 = 0.2$
$\lambda(Z=0) = \lambda_X(Z=0)\lambda_Y(Z=0) = 0.2*1 = 0.2$
$P(Z=0|d) = \alpha\lambda(Z=0)\,\pi(Z=0) = \alpha 0.2*0.4 = \alpha 0.08$

$$P(Z=1|d) = \frac{\sim 0.42}{\sim 0.42 + \sim 0.08} = 0.84$$
$$P(Z=0|d) = \frac{\sim 0.08}{\sim 0.42 + \sim 0.08} = 0.16$$

Calling *propagate_down_π_message*(*Z, Y*)

Then, function *propagate_down_π_message* (*Z, Y*) is executed:

$\pi_Y(Z=1) = \pi(Z=1)\,\lambda_Y(Z=1) = 1*0.6 = 0.6$
$\pi_Y(Z=0) = \pi(Z=0)\,\lambda_Y(Z=0) = 1*0.4 = 0.4$

$\pi(Y=1) = P(Y=1|Z=1)\,\pi_X(Z=1) + P(Y=1|Z=0)\,\pi_X(Z=0) = 0.6*0.6 + 0.3*0.4 = 0.48$
$\pi(Y=0) = P(Y=0|Z=1)\,\pi_X(Z=1) + P(Y=0|Z=0)\,\pi_X(Z=0) = 0.3*0.6 + 0.8*0.4 = 0.5$

$P(Y=1) = \alpha\,\lambda(Y=1)\,\pi(Y=1) = \alpha 1*0.48 = \alpha 0.48$
$P(Y=0) = \alpha\,\lambda(Y=0)\,\pi(Y=0) = \alpha 1*0.5 = \alpha 0.5$

$$P(Y=1) = \frac{\sim 0.48}{\sim 0.48 + \sim 0.5} = 0.49$$
$$P(Y=0) = \frac{\sim 0.5}{\sim 0.48 + \sim 0.5} = 0.51$$

Then function *propagate_down_π_message*(*X, T*) is executed
$\pi_T(X=1) = \pi(X=1) = 1$
$\pi_T(X=0) = \pi(X=0) = 0$

$\pi(T=1) = P(T=1|X=1)\,\pi_T(X=1) + P(T=1|X=0)\,\pi_T(X=0) = 0.9*1 + 0.4*0 = 0.9$
$\pi(T=0) = P(T=0|X=1)\,\pi_T(X=1) + P(T=0|X=0)\,\pi_T(X=0) = 0.1*1 + 0.6*0 = 0.1$

$P(T=1) = \alpha\,\lambda(T=1)\,\pi(T=1) = \alpha 1*0.9 = \alpha 0.9$
$P(T=0) = \alpha\,\lambda(T=0)\,\pi(T=0) = \alpha 1*0.1 = \alpha 0.1$

$$P(T=1) = \frac{\sim 0.9}{\sim 0.9 + \sim 0.1} = 0.9$$
$$P(T=0) = \frac{\sim 0.1}{\sim 0.9 + \sim 0.1} = 0.1$$

Finally, all posterior probabilities are computed as in following figure

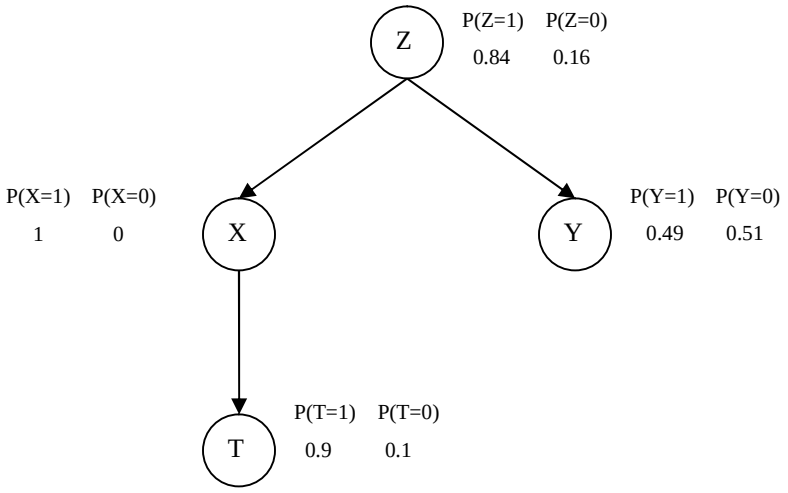


Figure 2.4: All posterior probabilities are computed after running Pearl algorithm (*X* is evidence)

2.3. OR-gate inference

In OR-gate electric circuit, the output value becomes *TRUE* if there is at least one of inputs being *TRUE*. Suppose every node is binary, OR-gate inference [1] in Bayesian network simulates such circuit based on three assumptions:

Cause inhibition: Given a cause-effect relationship denoted by edge $X \to Y$, there is a factor *I* that inhibits *X* from causing *Y*. Factor *I* is called inhibition of *X*. That the inhibition *I* is turned off is the prerequisite of *X* causing *Y*.

$I = 0 \Leftrightarrow I$ turned OFF
$I = 1 \Leftrightarrow I$ turned ON

Inhibition independence: Inhibitions are mutually independent. For example inhibition I_1 of X_1 is independent from inhibition I_2 of X_2.

OR condition: Suppose we have a set of cause-effect relationships in which Y is the effect of many causes $X_1, X_2,..., X_n$ (see following figure). Let I_i be the inhibition of X_i. The effect *Y can not happen* ($Y=0$) if *at least* one of X_i is equal *0* or one of inhibitions is *ON*:

$$\exists i : X_i = 0 \vee I_i = 1 \Rightarrow Y = 0$$

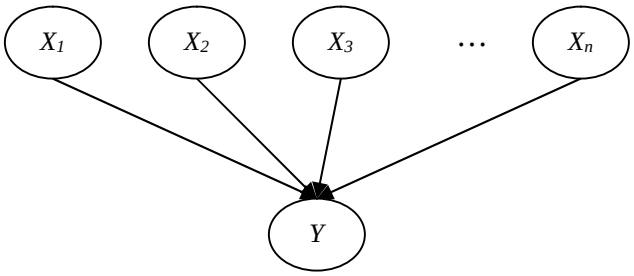


Figure 2.4: Cause-effect relationships.

Suppose we have n causes $X_1, X_2,..., X_n$ and one result Y. According to "*cause inhibition*" and "*inhibition independence*" assumptions, let I_i be the inhibition of X_i. Let A_i be dummy variable so that A_i is *ON* (=*1*) if X_i is equal to *1* and I_i is *OFF* (=0).

$P(A_i = ON \mid X_i=1, I_i=OFF) = 1$
$P(A_i = ON \mid X_i=1, I_i=ON) = 0$
$P(A_i = ON \mid X_i=0, I_i=OFF) = 0$
$P(A_i = ON \mid X_i=0, I_i=ON) = 0$

$P(A_i = OFF \mid X_i=1, I_i=OFF) = 0$
$P(A_i = OFF \mid X_i=1, I_i=ON) = 1$
$P(A_i = OFF \mid X_i=0, I_i=OFF) = 1$
$P(A_i = OFF \mid X_i=0, I_i=ON) = 1$

Applying "*OR condition*", the condition probability of Y is equal *0* (Y never happens) if at least one A_i is *ON*. It means that Y happens ($Y=1$) if all A_i (s) are *ON*.

$P(Y=0 \mid \exists A_i=ON) = 0$
$P(Y=0 \mid \forall A_i=OFF) = 1$
$P(Y=1 \mid \forall A_i=ON) = 1$
$P(Y=1 \mid \exists A_i=OFF) = 0$

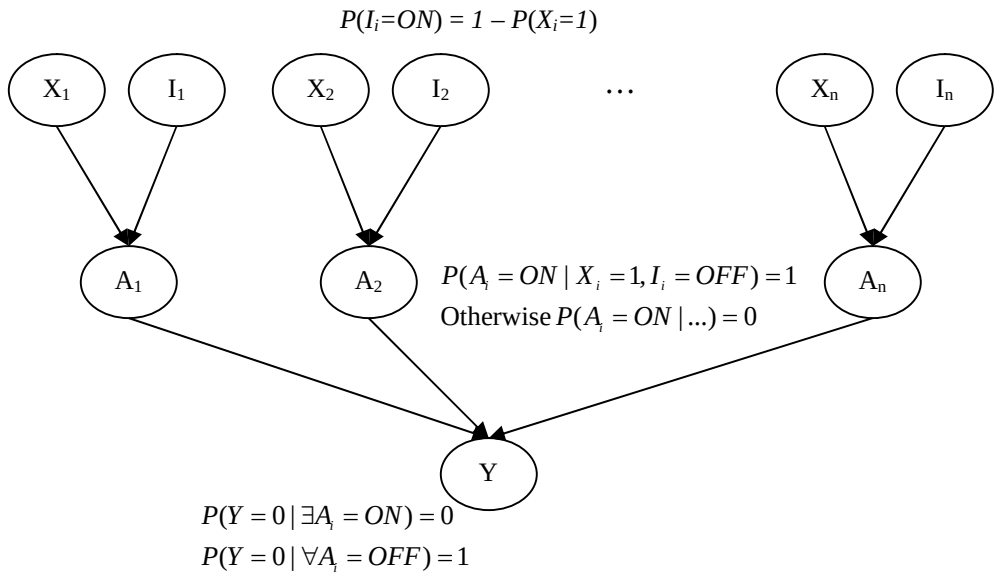


Figure 2.5: OR-gate model.

Now the strength of each cause-effect relationship $X_i \rightarrow Y$ is quantified by the CPT $P(Y|X_i)$. Suppose causes ($X_1, X_2,..., X_i,...,$

X_n) become evidences having values ($x_1, x_2,..., x_i,..., x_n$). Let $P(X_i=1) = p_i$ be the probability of $X_i = 1$. The probability of X_i 's inhibition is the inverse:
$P(I_i=ON) = 1 - P(X_i=1)$
Let O be the set of such i that $X_i = 1$.
$\forall i \in O, X_i = 1$

The goal of inference is to determine the posterior probability $P(Y| X_1, X_2,...,X_i,..., X_n)$. We have:

$$P(Y=0 \mid X_1=x_1,..., X_n=x_n)$$
$$= \sum_{a_1,...,a_n} P(Y=0 \mid A_1=a_1,..., A_n=a_n) P(A_1=a_1,..., A_n=a_n \mid X_1=x_1,..., X_n=x_n)$$
(Due to the law of total probability)
$$= \sum_{a_1,...,a_n} P(Y=0 \mid A_1=a_1,..., A_n=a_n) \prod_i P(A_i \mid X_1=x_1,..., X_n=x_n)$$
(Due to A_i (s) are mutually independent)
$$= \sum_{a_1,...,a_n} P(Y=0 \mid A_1=a_1,..., A_n=a_n) \prod_i P(A_i \mid X_i=x_i)$$
(Because A_i is only dependent on X_i)
$$= \prod_i P(A_i = OFF \mid X_i = x_i)$$
(Because that any A_i is equal ON causes the conditional probability $P(Y = 0 \mid A_i = ON) = 0$, we just focus on A_i = OFF)
$$= \prod_i (P(A_i = OFF \mid X_i = x_i, I_i = ON) P(I_i = ON) + P(A_i = OFF \mid X_i = x_i, I_i = OFF) P(I_i = OFF))$$
$$= \prod_{i \in O} (P(A_i = OFF \mid X_i = 1, I_i = ON) P(I_i = ON) + P(A_i = OFF \mid X_i = 1, I_i = OFF) P(I_i = OFF))$$
$$+ \prod_{i \notin O} (P(A_i = OFF \mid X_i = 1, I_i = ON) P(I_i = ON) + P(A_i = OFF \mid X_i = 1, I_i = OFF) P(I_i = OFF))$$
$$= \prod_{i \in O} (1(1 - P(X_i)) + 0 P(X_i)) \prod_{i \notin O} (1(1 - P(X_i)) + 1 P(X_i))$$
$$= \prod_{i \in O} (1 - P(X_i))$$

In conclusion, we have

$$P(Y=0/X_1,X_2,..., X_n) = \prod_{i \in O} (1 - P(X_i)) \quad (2.5)$$

$$P(Y=1/X_1,X_2,..., X_n) = 1 - \prod_{i \in O} (1 - P(X_i)) \quad (2.6)$$

Where O is the set of such i that $X_i = 1$.

Example 2.2: Given cause-effect relationship shown in following figure. Given prior probabilities of causes X_1, X_2, X_3 are *0.2, 0.5, 0.3*, respectively. We need to compute the conditional probability of effect $P(Y=1/X_1=1, X_2=0, X_3=1)$.

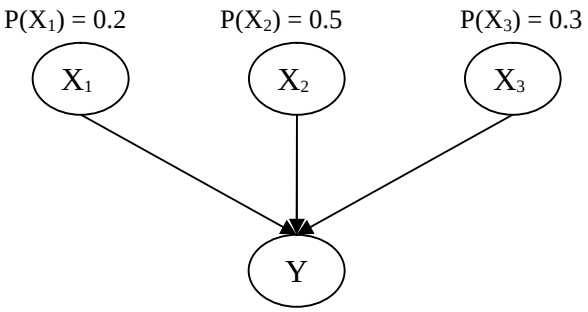


Figure 2.6: OR-gate inference example.

Applying formula 2.6, we have:
$P(Y=1/ X_1=1, X_2=0, X_3=1) = 1 - (1 - P(X_1=1))(1 - P(X_3=1)) = 1 - 0.8*0.7 = 0.44$

2.4. Optimal factoring

The basic idea of this technique is to reduce the amount of numeric operations by changing the order of combinations of such operations. Back example 1.1, given joint probability $P(C, R, S, W)=P(C)*P(S)*P(R/C)*P(W/R,S)$, the marginal probability of $R = 1$ is factorized as below:

$$P(R=1,W=1) = \sum_{C,S} P(C)P(S)P(R=1 \mid C)P(W=1 \mid R=1,S)$$

Because each binary variable has *2* values, there are 2^2 combinations of C and S. Each product has *3* multiplications. So the total number of required multiplications is $3*2^2 = 12$. Now the ordering of expression is changed by the factorization as below:

$$P(R=1,W=1) = \sum_C P(C)P(R=1 \mid C) \sum_S P(S)P(W=1 \mid R=1,S)$$

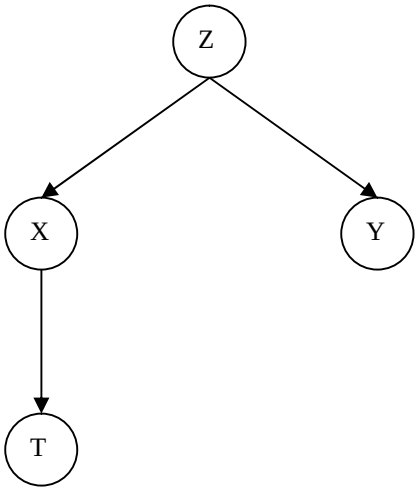
The inner sum of products $\sum_S P(S)P(W=1 \mid R=1,S)$ has $1*2^1=2$ multiplications. Although the outer sum of products $\sum_C P(C)P(R=1 \mid C) \sum_S (...)$ contains *4* variables, it has $2*2^1= 4$ multiplications because expressions which don't relate to variable S such as $P(C)$ and $P(R=1/C)$ are taken out the inner sum of products. So the total number of required multiplications is *4+2=6.* Six multiplications are saved.

It is easy to recognize the best ordering of expressions which produces the minimal required multiplications if the number of variables is small. How we can do that in case of many variables. The answer relates to the optimal factoring problem.

Given $F = (V, S, Q)$ is defined as the triple consisting of [1, pp. 163]:

A set of n nodes (or variables) $V= \{X_1, X_2,..., X_n\}$
A set of m sub-sets $S = \{S_{\{1\}}, S_{\{2\}},..., S_{\{m\}}\}$ where $S_{\{I\}} \subset V$
A *target set* $Q \subset V$
The factoring α of S is a binary tree satisfying three following condition [1, pp.164]:
All and only member $S_{\{I\}}$ of S are leaves.
The parent of nodes $S_{\{I\}}$ and $S_{\{J\}}$ are denoted $S_{\{I \cup J\}}$
The root of tree is $S_{\{1,2,..,m\}}$
Note that S corresponds to operands of marginal probability and α corresponds with the factorization of marginal probability.

Example 2.3: Like example 2.1, let *Z, X, Y, T* be nodes of Bayesian network shown in following figure.



The joint probability is $P(Z,X,Y,T) = P(Z)P(X|Z)P(Y|Z)P(T|X)$. Suppose X is evidence, we need to compute the posterior conditional probability $P(Z=1|X=1)$. The marginal probability over Z, X shown below is the sum of products which will be optimized:

$$P(Z=1, X=1) = \sum_{Y,T} P(Z=1)P(X=1|Z=1)P(Y|Z=1)P(T|X=1)$$

The factoring instance $F(V, S, Q)$ is defined as below:

$V = \{Z, X, Y, T\}$

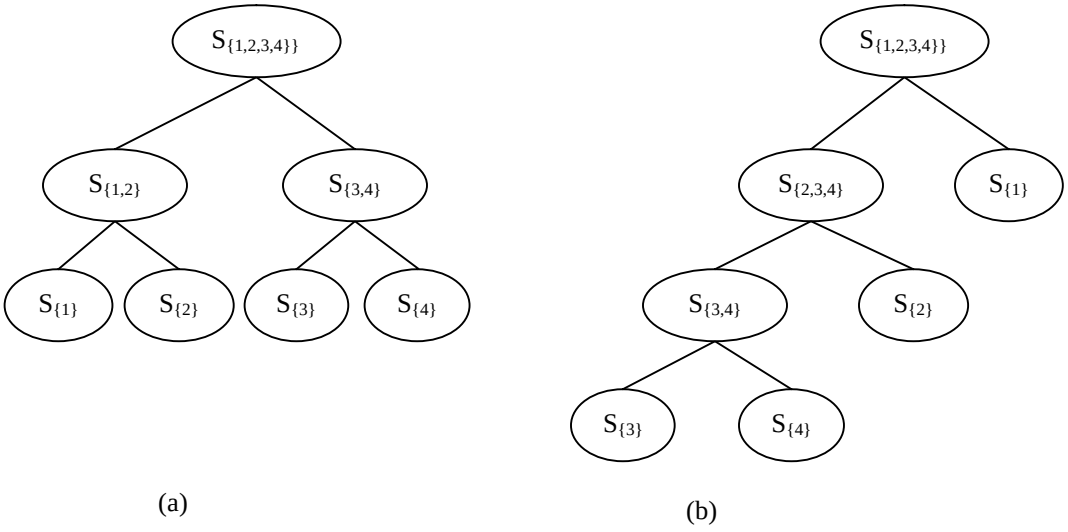
$S = \{S_{\{1\}}=\{Z\}, S_{\{2\}}=\{X, Z\}, S_{\{3\}}=\{Y, Z\}, S_{\{4\}}=\{T,X\}\}$

$Q = \{Z, X\}$

Suppose factoring α_1, α_2 correspond to two factorizations of marginal probability $P(Z=1,X=1)$.

$$\sim_1 \approx P(Z=1)P(X=1|Z=1)\sum_{Y}(P(Y|Z=1)\sum_{T} P(T|X=1)$$

$$\sim_2 \approx \sum_{Y,T} P(Z=1)P(X=1|Z=1)P(Y|Z=1)P(T|X=1)$$



(a) (b)

Figure 2.7: (a) Factoring α_1 and (b) Factoring α_2

Given F, the cost of factoring α denoted $cost_\alpha(F)$ is two following steps:

All non-leave nodes are determined according to formula:

$S_{\{I \cup J\}} = S_{\{I\}} \cup S_{\{J\}} - W_{\{I \cup J\}}$ where $W_{\{I \cup J\}} = \{w \notin Q$ and $w \notin S_{\{k\}}$ for all $k \notin I \cup J\}$

The cost of each node is computed according to formula:

For leaf nodes: $cost_\alpha(S_{\{j\}}) = 0, j = \overline{1,m}$

For non-leaf nodes: $cost_\alpha(S_{\{I \cup J\}}) = cost_\alpha(S_I) + cost_\alpha(S_J) + 2^{|S_I \cup S_J|}$

Where /./ denotes the cardinality of the set.

The cost of factoring α: $cost_\alpha(F) = cost_\alpha(S_{\{1,\ldots,m\}})$. The less this cost is, the better binary tree is.

Applying optimal factoring problem into Bayesian inference, the set of nodes V in F corresponds with variables in BN and the tree α corresponds with the ordering of multiplications in marginal probability. The cost of factoring instance $cost_\alpha(F)$ is equal to the number of multiplications. The problem becomes easy when we find out the best binary tree α having the least $cost_\alpha(F)$ and compute the marginal probability with the same ordering of multiplications to this tree.

Back example 2.3, the cost of factoring α_1 is computed as below:

$cost_{\alpha 1}(S\{1,2,3,4\}) = cost_{\alpha 1}(S_{\{1,2\}}) + cost_{\alpha 1}(S_{\{3,4\}}) = (0+0+2^0) + (0+0+2^2) = 5$

$cost_{\alpha 2}(S\{1,2,3,4\}) = cost_{\alpha 1}(S_{\{2,3,4\}}) + cost_{\alpha 1}(S_{\{1\}}) + 2^2 = cost_{\alpha 1}(S_{\{2,3,4\}}) + 0 + 2^2$

$= cost_{\alpha 1}(S_{\{3,4\}}) + cost_{\alpha 1}(S_{\{2\}}) + 2^2 + 2^2$

$= (0+0+2^1) + 0 + 2^2 + 2^2 = 10$

Because $cost_{\alpha 1}(S\{1,2,3,4\})$ is lesser than $cost_{\alpha 2}(S\{1,2,3,4\})$, the following ordering of multiplications is chosen:

$$P(Z=1, X=1) = P(Z=1)P(X=1 \mid Z=1)\sum_Y (P(Y \mid Z=1)\sum_T P(T \mid X=1)$$

3. Parameter learning

3.1. Beta function and augmented Bayesian network

There is a family of PDF which quantifies and updates the strength of conditional dependencies among nodes by natural way is called beta density function, denoted as $\beta(f; a, b)$ or $Beta(f; a, b)$ with parameters $a, b, N=a+b$ where a, b should be integer number > 0

$$(f) = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} f^{a-1}(1-f)^{b-1}$$

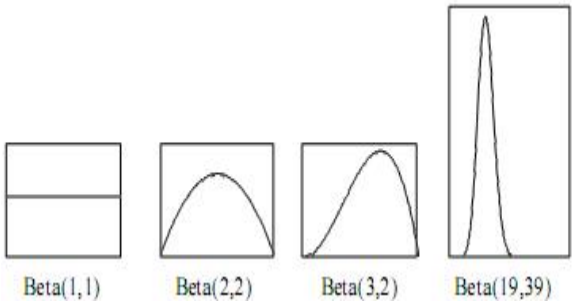


Figure 3.1: Beta functions

It means that, there are "a" successful outcomes (for example, $f=1$) in "$a+b$" trials. Higher value of "a" is, higher ratio of success is, so, the graph leans forward right. Higher value of "$a+b$" is, the more the mass is concentrate around $a/(a+b)$ and the more narrow the graph is. Definition of beta function is based on gamma function described below:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt \qquad (3.1)$$

The integral will converges if $x>0$, at that time,

$\Gamma(x) = (x-1)!$. Of course, we have $\frac{\Gamma(x+1)}{\Gamma(x)} = x$ (3.2)

From formula 3.1, $\int_0^1 f^a (1-f)^b\, df = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)}$ (3.3)

Suppose there is one binary variable X in network and the probability distribution of X is considered as relative frequency having values in $[0, 1]$ which is the range of variable F. We add a dummy variable F (whose space consists of numbers in $[0, 1]$, of course) which acts as the parent of X and has a beta density function $\beta(f; a, b)$, so as to:

$P(X=1|f) = f$, where f denotes values of F

X and F constitute a simple network which is referred as augmented BN. So X is referred as real variable (hypothesis) opposite to dummy variable.

Figure 3.2: The simple augmented BN with only one hypothesis node X

Obviously, $P(X=1) = E(F)$ where $E(F)$ is the expectation of F

Proof, owing to the law of total probability

$$P(X=1)=\int_0^1 P(X=1\mid f)\ \ (f)df=\int_0^1 f\ \ (f)df=E(F)$$

Due to F is beta function,

$$E(F)=\frac{a}{N}\text{, so, }\Pr(X=1)=\frac{a}{N} \quad (3.4)$$

Proof,

$$E(F)=\int_0^1 f\ (f)df=\int_0^1 f\frac{\Gamma(N)}{\Gamma(a)\Gamma(b)}f^{a-1}(1-f)^{b-1}df$$

$$=\frac{\Gamma(N)}{\Gamma(a)\Gamma(b)}\int_0^1 f^{a}(1-f)^{b-1}=\frac{\Gamma(N)}{\Gamma(a)\Gamma(b)}\frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(N+1)}\ \text{(due to formula 3.3)}$$

$$=\frac{a}{N}\quad\text{(applying formula 3.2)}$$

The ultimate purpose of Bayesian inference is to consolidate a hypothesis (namely, variable) by collecting evidences. Suppose we perform M trials of a random process, the

outcome of u^{th} trial is denoted $X^{(u)}$ considered as evidence variable whose probability $P(X^{(u)} = 1 \mid f) = f$. So, all $X^{(u)}$ are conditionally dependent on F. The probability of variable X, $P(X=1)$ is learned by these evidences.

We denote the vector of all evidences as $D = (X^{(1)}, X^{(2)},..., X^{(M)})$ which is also called the sample of size M. Given this sample, $\beta(f)$ is called the prior density function, and $P(X^{(u)} = 1) = a/N$ (due to formula 3.1) is called prior probability of $X^{(u)}$. *It is necessary to determine the posterior density function $\beta(f|D)$ and the posterior probability of X, namely $P(X|D)$. The nature of this process is the parameter learning*. Note that $P(X|D)$ is referred as $P(X^{(M+1)} \mid D)$.

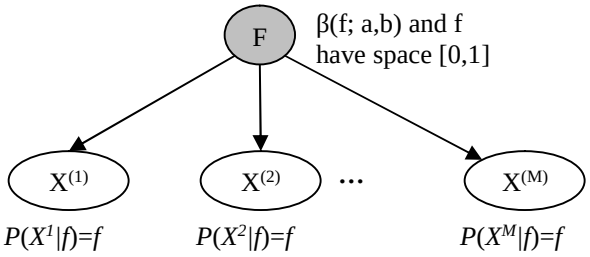


Figure 3.3: The sample $D=(X^{(1)}, X^{(2)},..., X^{(M)})$ size of M

We only surveyed in the case of binomial sample, in other words, D having binomial distribution is called binomial sample and the network in figure 3 becomes a binomial augmented BN. Then, suppose s is the number of all evidences $X^{(i)}$ which have value *1* (success), otherwise, t is the number of all evidences $X^{(j)}$ which have value *0* (failed). Of course, $s + t = M$.

Owing the law of total probability, we have

$$E(f^s(1-f)^t)=\int_0^1 f^s(1-f)^t\ \ (f)df$$

$$=\int_0^1 f^s(1-f)^t\frac{\Gamma(N)}{\Gamma(a)\Gamma(b)}f^{a-1}(1-f)^{b-1}df\quad\text{(applying formula 3.1)}$$

$$=\frac{\Gamma(N)}{\Gamma(a)\Gamma(b)}\int_0^1 f^{a+s-1}(1-f)^{b+t-1}df \quad (3.5)$$

$$=\frac{\Gamma(N)}{\Gamma(a)\Gamma(b)}\frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a+b+s+t)}\quad\text{(due to formula 3.3)}$$

$$=\frac{\Gamma(\mathrm{N})}{\Gamma(\mathrm{N+M})}\frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)}\quad(\text{due to s+t}=M)$$

And,

$$P(D)=\int_0^1 \Pr(D|f)\ \ (f)df=\int_0^1\prod_{i=1}^{M}P(X^i\mid f)\ \ (f)df$$

$$=\int_0^1 f^s(1-f)^s\ \ (f)df=E(f^s(1-f)^s),\ \text{due to}\prod_{i=1}^{M}P(X^i\mid f)=f^s(1-f)^t \quad (3.6)$$

3.2. Parameter learning

The essence of parameter learning is to compute the posterior density function [1]. Now, we need to compute the posterior density function $\beta(f|D)$ and the posterior probability $P(X=1|D)$. It is essential to determine the probability distribution of X.

$$(f\mid D)=\frac{P(D\mid f)\ \ (f)}{P(D)}\quad\text{(Bayes' law)}$$

$$=\frac{f^s(1-f)^t\ \ (f)}{E(f^s(1-f)^t)}\ (\text{due to P(D}\mid\text{f)}=\prod_{i=1}^{M}P(X^i\mid f)=f^s(1-f)^t\ \text{ and apply formula 3.6}) \quad (3.7)$$

$$=\frac{f^s(1-f)^t\frac{\Gamma(N)}{\Gamma(a)\Gamma(b)}f^{a-1}(1-f)^{b-1}}{\frac{\Gamma(N)}{\Gamma(N+M)}\frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)}}\quad\text{(apply formula 3.1, 3.5)}$$

$$=\frac{\Gamma(\mathrm{N+M})}{\Gamma(\mathrm{a+s})\Gamma(\mathrm{b+t})}f^{a+s-1}(1-f)^{b+t-1}=\ \ (f;a+s,b+t)$$

Then the posterior density function is $\beta(f; a+s, b+t)$ where the prior density function is $\beta(f; a, b)$. According to formula 3.4, the posterior probability:

$$P(X=1|D)=E(\beta(f|D))=\frac{a+s}{a+s+b+t}=\frac{a+s}{N+M} \quad (3.8)$$

In general, you should merely engrave the formula 3.1, 3.4, 3.7, 3.8 and the way to recognize prior density function, prior probability of X and posterior density function, posterior probability of X, respectively on your memory.

3.3. Expanding augmented BN with more than one hypothesis node

Suppose we have a BN with two binary random variables and there is conditional dependence assertion between these nodes. See the network and CPT (s) in the figure below

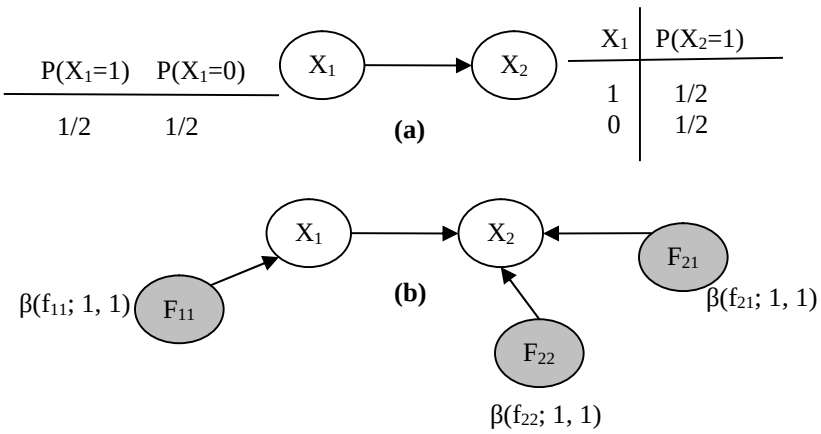


Figure 3.4: BN (a) and expended augmented BN (b)

For every node (variable) X_i, we add dummy parent nodes to X_i, obeying two ways below:

If X_i has no parent (not conditionally dependent on any others), we add only one dummy variable denoted F_{i1} having the probability density function $\beta(f_{i1};\ a_{i1},\ b_{i1})$ so as to: $P(X_i=1|f_{i1})=f_{i1}$

If X_i has a set of k_i parents and each parent pa_{il} ($l=\overline{1,k_i}$) is binary, we add a set of $c_i=2k_i$ dummy variables $F_i=\{f_{i1}, f_{i2},..., f_{ic_i}\}$, in turn, instantiations of parents $PA_i=\{pa_{i1}, pa_{i2}, pa_{i3},..., pa_{ic_i}\}$. In other words, c_i denotes the number of instantiations of the parents PA_i. We have $P(X_i=1|pa_{ij}, f_{i1},..., f_{ij},..., f_{ic_i})=f_{ij}$. where $(f_{ij})=\dfrac{\Gamma(N_{ij})}{\Gamma(a_{ij})\Gamma(b_{ij})} f^{a_{ij}-1}(1-f_{ij})^{b_{ij}-1}$

All f_{ij} have no parent and are mutually independent, so, $\beta(f_{i1}, f_{i2},..., f_{ic_i}) = \beta(f_{i1})\ \beta(f_{i2})...\ \beta(f_{ic_i})$. Besides this local parameter independence, we have the global parameter independence if reviewing all variables X_i s, such below:

$\beta(F_1, F_2,..., F_n)=\beta(f_{11}, f_{12},..., f_{ic_{in}})=\beta(f_{i1})\ \beta(f_{i2})...\ \beta(f_{ic_{in}})$

All variables X_i and their dummy variables form the expended augmented BN representing the trust BN in figure 4. In the trust BN, the conditional probability of variable X_i with the instantiation of its parent pa_{ij}, in other words, the ij^{th} conditional distribution is given by $P(X_i=1|\ pa_{ij}=1) = E(F_{ij})=\dfrac{a_{ij}}{N_{ij}}$ (3.8), that's to say the expected value of F_{ij}.

Proof, $P(X_i=1|\ pa_{ij}=1)=\int_0^1...\int_0^1 P(X_i=1|\ pa_{ij}=1, f_{i1},..., f_{ic_i})\ (f_{i1})...\ (f_{ic_i})df_{i1}...df_{ic_i}$

$=\int_0^1...\int_0^1 f_{ij}\ (f_{i1})...\ (f_{ic_i})df_{i1}...df_{ic_i} = E(F_{ij})$

(due to F_{ij} (s) are mutually independen t, $P(X_i=1|\ pa_{ib_i}=1, f_{i1},..., f_{ic_i}) = P(X_i=1|\ pa_{ij}=1, f_{ij}) = f_{ij}$)

Suppose we perform M trials of random process, the outcome of i^{th} trial which is BN like figure 4 is represented as a random vector $X^{(u)}=\begin{pmatrix} X_1^{(u)} \\ ... \\ X_n^{(u)} \end{pmatrix}$ containing all hypothesis variables in network. $X^{(u)}$ is also called evidence vector (or evidence, briefly). M trials constitute the sample of size M which is the set of random vectors denoted as $D=\{X^{(1)}, X^{(2)},..., X^{(M)}\}$. D is also called evidence matrix. We review only in case of binomial sample; it means that D is the binomial BN sample of size M. For example, this sample corresponding to the network in figure 4 is showed below:

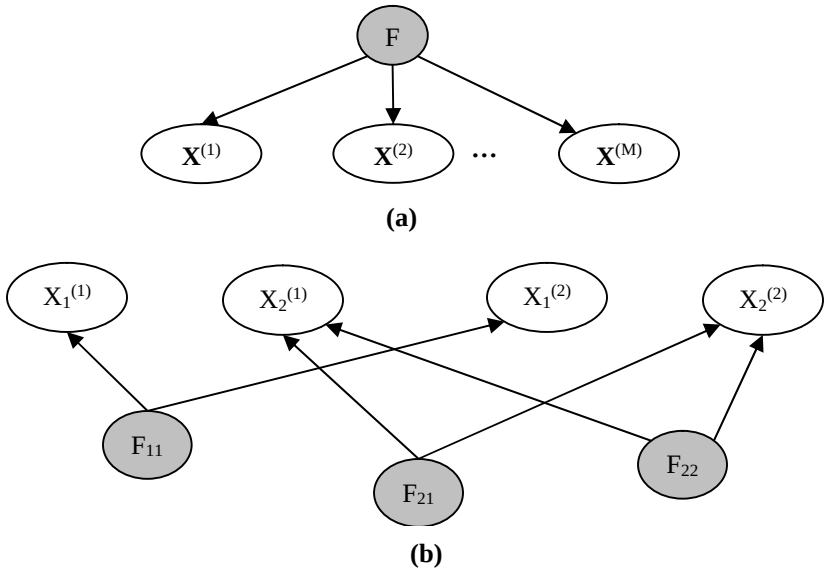


Figure 3.5: Expanded binomial BN sample of size *M*

After occurring *M* trial, the augmented BN was updated and dummy variables' density functions and hypothesis variables' conditional probabilities changed. We need to compute the posterior density function $\beta(f_{ij}|D)$ of each dummy variable F_{ij} and the posterior condition probability $P(X_i=1|\ pa_{ij}=1,\ D)$ of each variable X_i. Note that the samples $X^{(u)}$ s are mutually independent with all given F_{ij}. We have,

$$\prod_{u=1}^{M} P(X_i^{(u)} \mid pa_i, F_i) = \prod_{j=1}^{c_i} (f_{ij})^{s_{ij}} (1-f_{ij})^{t_{ij}}$$

Where

c_i is the number of instances of $X_i^{(u)}$'s parents. In binary case, each $X_i^{(u)}$ (s) parent has two instances/values, namely, *0* and *1*.

s_{ij}, respective to f_{ij}, is the number of all evidences that variable $X_i = 1$ and $pa_{ij} = 1$

t_{ij}, respective to f_{ij}, is the number of all evidences that variable $X_i = 1$ and $pa_{ij} = 0$.

We have,

$$P(D \mid F_1,...,F_n) = \prod_{i=1}^{n}\prod_{u=1}^{M} \Pr(X_i^{(u)} \mid pa_i, F_i) = \prod_{i=1}^{n}\prod_{j=1}^{c_i} (f_{ij})^{s_{ij}} (1-f_{ij})^{t_{ij}} \quad (3.9)$$

$$P(D) = \prod_{i=1}^{n}\prod_{j=1}^{c_i} E(f_{ij}^{\ s_{ij}} (1-f_{ij})^{t_{ij}}) \quad (2.10)$$

Proof, $P(D) = \prod_{i=1}^{n} \left(\int_{F_i} \prod_{u=1}^{M} P(X_i^{(u)} \mid pa_i, F_i)\ (F_i)\, dF_i\right)$

(due to the law of total probabilit y and the joint probabilit y distributi on)

$= \prod_{i=1}^{n} \left(\int_{F_i} \prod_{j=1}^{c_i} (f_{ij})^{s_{ij}} (1-f_{ij})^{t_{ij}}\ (F_i)\, dF_i\right)$

(applying formula $\prod_{u=1}^{M} P(X_i^{(u)} \mid pa_i, F_i) = \prod_{j=1}^{c_i} (f_{ij})^{s_{ij}} (1-f_{ij})^{t_{ij}}$)

$= \prod_{i=1}^{n} \prod_{j=1}^{c_i} \int_0^1 (f_{ij})^{s_{ij}} (1-f)^{t_{ij}}\ (f_{ij})\, df_{ij}$

$= \prod_{i=1}^{n} \prod_{j=1}^{c_i} E(f_{ij}^{\ s_{ij}} (1-f_{ij})^{t_{ij}})$

There is the question "how to determine $E(f_{ij}^{\ s_{ij}} (1-f_{ij})^{t_{ij}})$". Applying formula 3.5, we have:

$$E(f_{ij}^{\ s_{ij}} (1-f_{ij}^{\ t_{ij}})) = \frac{\Gamma(N_{ij})}{\Gamma(N_{ij}+M_{ij})} \frac{\Gamma(a_{ij}+s_{ij})\Gamma(b_{ij}+t_{ij})}{\Gamma(a_{ij})\Gamma(b_{ij})} \quad (3.11)$$

Where $N_{ij}=a_{ij}+b_{ij}$ and $M_{ij}=s_{ij}+t_{ij}$

3.4. Updating posterior density function with multi-node Bayesian network

$$(f_{ij} \mid D) = \frac{(f_{ij})^{s_{ij}}(1-f_{ij})^{t_{ij}} \quad (f_{ij})}{E(f_{ij}^{\ s_{ij}}(1-f_{ij})^{t_{ij}})} = beta(f_{ij}; a_{ij}+s_{ij}, b_{ij}+t_{ij}) \quad \textbf{(3.12)}$$

Proof,

$$(f_{mn} \mid D) = \frac{P(D \mid f_{mn}) \quad (f_{mn})}{P(E)} \quad \text{(Bayes'law)}$$

$$= \frac{\left(\int_0^1 \ldots \int_0^1 \mathrm{P}(\mathrm{D} \mid F_1, F_2, \ldots, F_n) \prod_{ij \neq mn} (f_{ij}) df_{ij}\right) \quad (f_{mn})}{P(E)} \quad \text{(law of total probability)}$$

$$= \frac{(f_{mn})^{s_{mn}}(1-f_{mn})^{t_{mn}} \left(\prod_{ij \neq mn} \int_0^1 (f_{ij})^{s_{ij}}(1-f_{ij})^{t_{ij}} \quad (f_{ij}) df_{ij}\right) \quad (f_{mn})}{\prod_{i=1}^{n} \prod_{j=1}^{c_i} E(f_{ij}^{\ s_{ij}}(1-f_{ij})^{t_{ij}})}$$

(apply formula 3.9, 3.10)

$$= \frac{(f_{mn})^{s_{mn}}(1-f_{mn})^{t_{mn}} \quad (f_{mn})}{E(f_{mn}^{\ s_{mn}}(1-f_{mn})^{t_{mn}})}$$

$$= \frac{(f_{mn})^{s_{mn}}(1-f_{mn})^{t_{mn}} \frac{\Gamma(N_{mn})}{\Gamma(a_{mn})\Gamma(b_{mn})}(f_{mn})^{a_{mn}-1}(1-f_{mn})^{b_{mn}-1}}{\frac{\Gamma(N_{mn})}{\Gamma(N_{mn}+M_{mn})} \frac{\Gamma(a_{mn}+s_{mn})\Gamma(b_{mn}+t_{mn})}{\Gamma(a_{mn})\Gamma(b_{mn})}}$$

(expansion of (f_{mn}) and applying formula 3.11 to $E(f_{mn}^{\ s_{mn}}(1-f_{mn})^{t_{mn}})$)

$$= \frac{\Gamma(N_{mn}+M_{mn})}{\Gamma(a_{mn}+s_{mn})\Gamma(b_{mn}+t_{mn})}(f_{mn})^{a_{mn}+s_{mn}-1}(1-f_{mn})^{b_{mn}+t_{mn}-1}$$

$$= beta(f_{mn}; a_{mn}+s_{mn}, b_{mn}+t_{mn})$$

According to formula 3.8 and 3.12,

$$P(X_i=1 \mid pa_{ij}=1, D) = E(F_{ij}) = E(\beta(f_{ij}/D)) = \frac{a_{ij}+s_{ij}}{a_{ij}+s_{ij}+b_{ij}+t_{ij}} = \frac{a_{ij}+s_{ij}}{N_{ij}+M_{ij}} \quad \textbf{(3.13)}$$

In short, in case of binomial distribution, if we have the real/trust BN embedded in the expanded augmented network such as figure 3.4 and each dummy node F_{ij} has a prior beta distribution $\beta(f_{ij}; a_{ij}, b_{ij})$ and each hypothesis node X_i has the prior conditional probability $P(X_i=1 \mid pa_{ij}=1) = E(\beta(f_{ij})) = \frac{a_{ij}}{N_{ij}}$, the parameter learning process based on a set of evidences is to update the posterior density function $\beta(f_{ij}/D)$ and the posterior conditional probability $P(X_i=1 \mid pa_{ij}=1, D)$. Indeed,

$$(f_{ij} \mid D) = beta(f_{ij}; a_{ij}+s_{ij}, b_{ij}+t_{ij}) \quad \text{and} \quad P(X_i=1 \mid pa_{ih_i}=1, E) = E(\beta(f_{ij}/D)) = \frac{a_{ij}+s_{ij}}{N_{ij}+M_{ij}}$$

Example 3.1: Suppose we have the set of 5 evidences $D=\{X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}\}$ owing to network in figure 3.4

	X_1	X_2
$X^{(1)}$	$X_1^{(1)}=1$	$X_2^{(1)}=1$
$X^{(2)}$	$X_1^{(2)}=1$	$X_2^{(2)}=1$
$X^{(3)}$	$X_1^{(3)}=1$	$X_2^{(3)}=1$
$X^{(4)}$	$X_1^{(4)}=1$	$X_2^{(4)}=0$
$X^{(5)}$	$X_1^{(5)}=0$	$X_2^{(5)}=0$

Table 3.1: Set of evidences D corresponding to 5 trials (sample of size 5)

Note that the first evidence $X^{(1)} = \begin{pmatrix} X_1^{(1)}=1 \\ X_2^{(1)}=1 \end{pmatrix}$ implies that variable $X_2=1$ given $X_1=1$ occurs in the first trial. We need to compute all posterior density functions $\beta(f_{11}/D)$, $\beta(f_{21}/D)$, $\beta(f_{22}/D)$ and all conditional probabilities $P(X_1=1)$, $P(X_2=1/X_1=1)$, $P(X_2=1/X_1=0)$ from prior density functions $\beta(f_{11}; 1,1)$, $\beta(f_{21}; 1,1)$, $\beta(f_{22}; 1,1)$. In fact,

$s_{11}=1+1+1+1+0=4 \quad t_{11}=0+0+0+0+1=1$
$s_{21}=1+1+1+0+0=3 \quad t_{21}=0+0+0+0+1=1$
$s_{22}=0+0+0+0+0=0 \quad t_{21}=0+0+0+0+1=1$

$\beta(f_{11}/D) = \beta(f_{11}; a_{11}+s_{11}, b_{11}+t_{11}) = \beta(f_{11}; 1+4, 1+1) = \beta(f_{11}; 5, 2)$
$\beta(f_{21}/D) = \beta(f_{21}; a_{21}+s_{21}, b_{21}+t_{21}) = \beta(f_{21}; 1+3, 1+1) = \beta(f_{11}; 4, 2)$
$\beta(f_{22}/D) = \beta(f_{22}; a_{22}+s_{22}, b_{22}+t_{22}) = \beta(f_{22}; 1+0, 1+1) = \beta(f_{11}; 1, 2)$

and $P(X_1=1)$, $P(X_2=1/X_1=1)$, $P(X_2=1/X_1=0)$ are expectations of $\beta(f_{11}/D)$, $\beta(f_{21}/D)$, $\beta(f_{22}/D)$. Then,

$P(X_1=1) = \frac{5}{5+2} = \frac{5}{7}$ $\quad P(X_2=1/X_1=1) = \frac{4}{4+2} = \frac{2}{3}$ $\quad P(X_2=1/X_1=0) = \frac{1}{1+2} = \frac{1}{3}$

Network in figure 3.4 changed as follows:

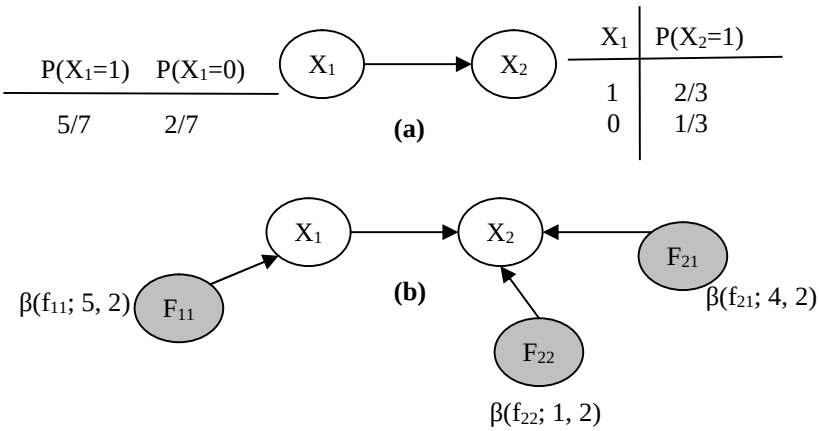


Figure 3.6: Updated version of BN (a) and augmented BN (b) in figure 3.4

3.5. Parameter learning in case of data missing

In practice there are some evidences in D such as $X^{(u)}$ (s) which lack information and thus, it stimulates the question "How to update network from data missing". We must address this problem by artificial intelligence techniques, namely, expectation maximization (EM) algorithm – a famous technique solving estimation of data missing.

Example 3.2: Like above example, we have the set of 5 evidences $D=\{X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}\}$ along with network in figure 4 but the evidences $X^{(2)}$ and $X^{(5)}$ have not data yet.

	X_1	X_2
$X^{(1)}$	$X_1^{(1)} = 1$	$X_2^{(1)} = 1$
$X^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = \boldsymbol{v_1?}$
$X^{(3)}$	$X_1^{(3)} = 1$	$X_2^{(3)} = 1$
$X^{(4)}$	$X_1^{(4)} = 1$	$X_2^{(4)} = 0$
$X^{(5)}$	$X_1^{(5)} = 0$	$X_2^{(5)} = \boldsymbol{v_2?}$

Table 3.2: Set of evidences D (for network in figure 4) with data missing

As known, s_{21} , t_{21} and s_{22} , t_{22} can't be computed directly, it means that it is not able to compute directly the posterior density functions $\beta(f_{21}|D)$ and $\beta(f_{22}|D)$. In evidence $X^{(2)}$, v_1 must be determined. Obviously, v_1 obtains one of two values which is respective to two situations:

$X_1^{(2)} = 1$ and $X_2^{(2)}=1$, it is easy to infer that $v_1=P(X_2^{(2)}=1|X_1^{(2)} = 1)=E(\beta_{21})=\frac{a_{21}}{a_{21}+b_{21}} = ½$

$X_1^{(2)} = 1$ and $X_2^{(2)}=0$, it is easy to infer that $v_1=P(X_2^{(2)}=1|X_1^{(2)} = 0)=E(\beta_{22})=\frac{a_{22}}{a_{22}+b_{22}} = ½$

We split $X^{(2)}$ into two $X'^{(2)}$ (s) corresponding to two above situations in which the probability of occurrence of $X_2=1$ given $X_1=1$ is estimated as ½ and the probability of occurrence of $X_2=0$ given $X_1=1$ is also considered as ½. We perform similarly this task for $X^{(5)}$.

	X_1	X_2
$X^{(1)}$	$X_1^{(1)} = 1$	$X_2^{(1)} = 1$
$X'^{(2)}$	$X_1'^{(2)} = 1$	$X_2'^{(2)} = \boldsymbol{1/2}$
$X'^{(2)}$	$X_1'^{(2)} = 1$	$X_2'^{(2)} = \boldsymbol{1/2}$
$X^{(3)}$	$X_1^{(3)} = 1$	$X_2^{(3)} = 1$
$X^{(4)}$	$X_1^{(4)} = 1$	$X_2^{(4)} = 0$
$X'^{(5)}$	$X_1'^{(5)} = 0$	$X_2'^{(5)} = \boldsymbol{1/2}$
$X'^{(5)}$	$X_1'^{(5)} = 0$	$X_2'^{(5)} = \boldsymbol{1/2}$

Table 3.3: New split evidences D' for network in figure 3.4

So, we have $\begin{pmatrix} s'_{21}=1+\frac{1}{2}+1=\frac{5}{2} \\ t'_{21}=\frac{1}{2}+1=\frac{3}{2} \end{pmatrix}$ and $\begin{pmatrix} s'_{22}=\frac{1}{2} \\ t'_{22}=\frac{1}{2} \end{pmatrix}$ where s'_{21}, t'_{21}, s'_{22}, t'_{22} are the counts in D'. Then

$\beta(f_{21}|D)= \beta(f_{21};\ a_{21}+s'_{21},\ b_{21}+t'_{21})= \beta(f_{21};1+5/2,\ 1+3/2)= \beta(f_{21};7/2,\ 5/2)$
$\beta(f_{22}|D)= \beta(f_{22};\ a_{22}+s'_{22},\ b_{22}+t'_{22})= \beta(f_{22};1+1/2,\ 1+1/2)= \beta(f_{22};3/2,\ 3/2)$

$P(X_2=1|\ X_1=1)=E(\beta(f_{21}|D))= \frac{7/2}{7/2+5/2}=\frac{7}{12}$ $\quad$ $P(X_2=0|\ X_1=1)=E(\beta(f_{22}|D))= \frac{3/2}{3/2+3/2}=\frac{1}{2}$

If there are more evidences, this task repeated more and more brings out the EM algorithm [1] [6] having two steps.

X_1	X_2
1	0
1	0
1	0
1	1
0	1
0	0

Step1. We compute s'_{ij} and t'_{ij} based on the expected value of given $\beta(f_{ij})$, $s'_{ij}=E(\beta(f_{ij}))$ and $t'_{ij}=1- E(\beta(f_{ij}))$. Next, replacing missing data by s'_{ij} and t'_{ij} . This step is called **E**xpectation step.

Step 2. We determine the posterior density function f_{ij} by computing its parameters $a_{ij}=a_{ij}+s_{ij}$ and $b_{ij}=b_{ij}+t_{ij}$. Note that s_{ij} and t_{ij} are recomputed absolutely together on occurrence of s'_{ij} and t'_{ij}. Terminating algorithm if the stop condition (for example, the number of iterations approaches k *times*) becomes true, otherwise, reiterating step 1. This step is called the **M**aximization step.

After k^{th} iteration, we have $\lim_{k\to\infty} Expectation_{ij} = \lim_{k\to\infty} \frac{a_{ij} + s_{ij}^{(k)}}{a_{ij} + s_{ij}^{(k)} + b_{ij} + t_{ij}^{(k)}}$ which will approach a certain limit. Don't worry about the case of infinite iterations, we will obtain approximate s'_{ij}, t'_{ij}, *posterior* f_{ij} if k is large enough due to certain value of $\lim_{k\to\infty} Expectation_{ij}$

4. Structure learning

As discussed, DAG (s) that contain the same given nodes V are Markov equivalent if they satisfy Markov condition and have the same d-separations. In other words, they entail the same conditional independences and their joint conditional probabilities are identical. Let the pattern gp represent these Markov equivalent DAG (s). Such pattern gp is called Markov *equivalent class*. Of course given a set of nodes V, there are a lot of equivalent classes. Let GP be random variable whose values are pattern gp. The basic idea of structure learning approaches is to find out the pattern gp that satisfy some condition best. Instead of searching many individual DAG According to given condition, there are two main learning approaches:

Score-based approach [1]: For each pattern $gp \in GP$, the gp which gains the maximal scoring criterion $score(D,gp)$ given training data set D is the best gp. Because the essence of score-based approach is find out the most likely structure, it is also called *model selection* [1] approach.

Constraint-based approach [1]: Given a set of conditional independences (a set of d-separations), the best gp is the DAG which satisfy Markov condition over all and only these conditional independences. Such independences play the role of the "door latch" for learning algorithm.

Note that in structure learning context, Bayesian network or pattern gp is mentioned as a DAG.

4.1. Score-based approach

Given a set of random variables (nodes) $V = \{X_1, X_2,..., X_n\}$, let (G, P) be possible Bayesian network where P is joint conditional probability density and $G=(V, E)$ is the DAG. Let $(G, F^{(G)}, \ ^{(G)})$ be the augmented BN with equivalent sample size N where $F(G)$ is augmented variables (nodes) attached to every nodes in V and (G) represents beta distributions for augmented (see section about parameter learning). Pattern gp also represents Markov equivalent augmented BN. Scored-based approach has three following steps:

Suppose all augmented BN (s) has the same equivalent sample size N.

Let r_i be the number of possible values of variable X_i. If X_i is binary then $r_i = 2$. Let q_i be the number of distinct instantiations of parents of X_i. For example, if X_i and its parents are binary and X_i have *1* parents then $q_i = 2$. All augmented variables F_{ij} representing the conditional probability of X_i given instantiation j of its parent are assigned to uniform distribution according to equivalent sample size N:

$$a_{ijk} = \frac{N}{r_i q_i}$$

Given $D=\{X^{(1)}, X^{(2)},..., X^{(M)}\}$ is the training data set size M, where $X^{(h)}$ is a trial. Note that $X^{(h)}=(X^{(h)}_1, X^{(h)}_2,..., X^{(h)}_n)$ is a n-dimension vector which is a outcome (instantiation) of variable X_i. $X^{(h)}_i$ has the same space to X_i. Each DAG gp which is connected by variables in V is assigned a value so-called scoring criterion $score(D,gp)$. This score is the posterior probability of gp given training data set D.

$$score(D, gp) = P(gp \mid D) = \frac{P(gp)P(D \mid gp)}{P(D)}$$

Where $P(D \mid gp) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \left(\frac{\Gamma(N_{ij})}{\Gamma(N_{ij} + M_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(a_{ijk} + s_{ijk})}{\Gamma(a_{ijk})}\right)$

$P(gp)$ is the prior probability of gp. $P(D)$ is constant.

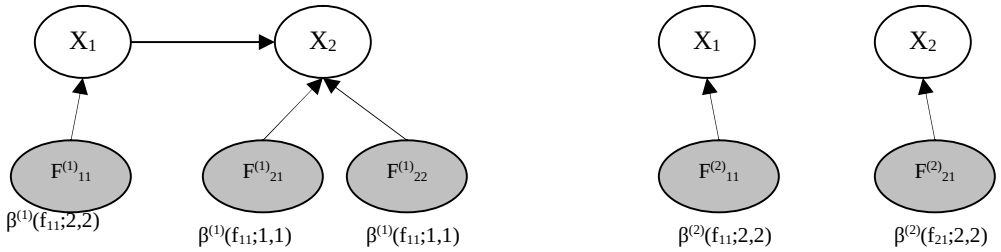
In practice, $score(D,gp)$ is only dependent on $P(D|gp)$ when $P(D)$ is ignored and $P(gp)$ is initialized subjectively.

$$score(D, gp) \approx \prod_{i=1}^{n} \prod_{j=1}^{q_i} \left(\frac{\Gamma(N_{ij})}{\Gamma(N_{ij} + M_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(a_{ijk} + s_{ijk})}{\Gamma(a_{ijk})}\right) \quad (4.1)$$

Which *gp* gaining maximal *score*(*D*,*gp*) is chosen.

Example 4.1: Suppose there are two variables X_1, X_2, we don't know exactly their relationship but the training data *D* is observed as below:

Let gp_1 be the DAG in which X_1 is parent of X_2; otherwise let gp_2 be the DAG in which X_1 and X_2 are mutually independent. Given the sample size is $N = 4$



(a) (b)

Figure 4.1: Augmented Bayesian networks of gp_1 (a) and gp_2 (b)

We have:

$$score(D, gp_1) \approx P(D \mid gp_1) =$$
$$= \left(\frac{\Gamma(N_{11}^{(1)})}{\Gamma(N_{11}^{(1)}+M_{11}^{(1)})}\left(\frac{\Gamma(a_{11}^{(1)}+s_{11}^{(1)})}{\Gamma(a_{11}^{(1)})}\frac{\Gamma(b_{11}^{(1)}+t_{11}^{(1)})}{\Gamma(b_{11}^{(1)})}\right)\right)\times$$
$$\left(\frac{\Gamma(N_{21}^{(1)})}{\Gamma(N_{21}^{(1)}+M_{21}^{(1)})}\left(\frac{\Gamma(a_{21}^{(1)}+s_{21}^{(1)})}{\Gamma(a_{21}^{(1)})}\frac{\Gamma(b_{21}^{(1)}+t_{21}^{(1)})}{\Gamma(b_{21}^{(1)})}\right)\right)\times$$
$$\left(\frac{\Gamma(N_{22}^{(1)})}{\Gamma(N_{22}^{(1)}+M_{22}^{(1)})}\left(\frac{\Gamma(a_{22}^{(1)}+s_{22}^{(1)})}{\Gamma(a_{22}^{(1)})}\frac{\Gamma(b_{22}^{(1)}+t_{22}^{(1)})}{\Gamma(b_{22}^{(1)})}\right)\right)$$
$$= \left(\frac{\Gamma(4)}{\Gamma(4+6)}\frac{\Gamma(2+4)}{\Gamma(2)}\frac{\Gamma(2+2)}{\Gamma(2)}\right)\times\left(\frac{\Gamma(2)}{\Gamma(2+4)}\frac{\Gamma(1+1)}{\Gamma(1)}\frac{\Gamma(1+3)}{\Gamma(1)}\right)\times\left(\frac{\Gamma(2)}{\Gamma(2+2)}\frac{\Gamma(1+1)}{\Gamma(1)}\frac{\Gamma(1+1)}{\Gamma(1)}\right)$$
$$= \frac{\Gamma(4)\Gamma(4)}{\Gamma(10)} = 9.9\times10^{-5}$$

$$score(D, gp_1) \approx P(D \mid gp_2) =$$
$$= \left(\frac{\Gamma(N_{11}^{(1)})}{\Gamma(N_{11}^{(1)}+M_{11}^{(1)})}\left(\frac{\Gamma(a_{11}^{(1)}+s_{11}^{(1)})}{\Gamma(a_{11}^{(1)})}\frac{\Gamma(b_{11}^{(1)}+t_{11}^{(1)})}{\Gamma(b_{11}^{(1)})}\right)\right)\times$$
$$\left(\frac{\Gamma(N_{21}^{(1)})}{\Gamma(N_{21}^{(1)}+M_{21}^{(1)})}\left(\frac{\Gamma(a_{21}^{(1)}+s_{21}^{(1)})}{\Gamma(a_{21}^{(1)})}\frac{\Gamma(b_{21}^{(1)}+t_{21}^{(1)})}{\Gamma(b_{21}^{(1)})}\right)\right)\times$$
$$= \left(\frac{\Gamma(4)}{\Gamma(4+6)}\frac{\Gamma(2+4)}{\Gamma(2)}\frac{\Gamma(2+2)}{\Gamma(2)}\right)\times\left(\frac{\Gamma(4)}{\Gamma(4+6)}\frac{\Gamma(2+2)}{\Gamma(2)}\frac{\Gamma(2+4)}{\Gamma(2)}\right)$$
$$= \left(\frac{\Gamma(6)\Gamma(4)\Gamma(4)}{\Gamma(10)}\right)^2 = 1190\times10^{-5}$$

Because *score*(*D*,gp_2) is larger than *score*(*D*,gp_2), the equivalent pattern gp_1 is chosen as Bayesian network appropriate to training data set.

In above example we recognize that it is difficult to determine all DAG (s). So the score-based approach becomes ineffective in case of many variables. The number of DAG (s) which is surveyed to compute scoring criterion gets huge. It is impossible to do brute-force searching over DAG (s) space. There are some heuristic algorithms to reduce whole DAG (s) space to smaller space so-called candidate set of DAG (s) obeying some restriction, for example, the prior ordering of variables. Such heuristic algorithms are classified into approximate learning. The global score can defined as a product of local scores:

$$score(D, gp) = \prod_{i=1}^{n} score(D, X_i, PA_i)$$

Where *score*(*D*,X_i,PA_i) is the local score of X_i given its parents PA_i.

$$score(D, X_i, PA_i) = P(X_i \mid PA_i, D) = \prod_{j=1}^{q^{(PA_i)}}\left(\frac{\Gamma(\sum_{k=1}^{r_i} a_{ijk})}{\Gamma(\sum_{k=1}^{r_i} a_{ijk}+\sum_{k=1}^{r_i} s_{ijk})}\prod_{k=1}^{r_i}\frac{\Gamma(a_{ijk}+s_{ijk})}{\Gamma(a_{ijk})}\right) \quad (4.2)$$

Let $q^{(PA_i)}$ be the number of distinct instantiations of parents of X_i

The K2 algorithm tries to find out the pattern DAG *gp* whose each variable X_i maximizes local score *score*(*D*,X_i,PA_i) instead of discovering all DAG (s). It means that K2 algorithm finds out optimal parents PA_i of each X_i. Note that it expects that

the global score will be approached by maximizing each partial local score. K2 algorithm has following steps:

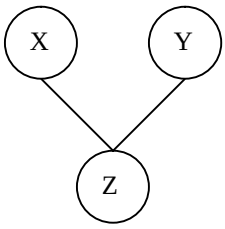
Suppose there is an ordering ($X_1, X_2,..., X_n$). There is no backward edge, for example, the edge $X_i \leftarrow X_j$ (if exist) where $i < j$ is invalid. Let $Pre(X_i)$ be the set of previous nodes of X_i in ordering. Let PA_i is parents of X_i. K2's mission is to find out PA_i for every X_i. Firstly, each PA_i (s) is set to be empty and each local $score(D,X_i,PA_i)$ is initialized with such empty PA_i.

Each X_i is visited according to the ordering. When X_i is visited, which node in $Pre(X_i)$ that maximizes the local $score(D,X_i,PA_i)$ is added to PA_i.
Algorithm terminates when no node is added to PA_i.

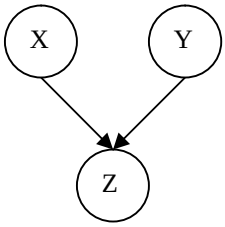
4.2. Constraint-based approach

Given (G, P) let IND_P be a set of conditional independences. IND_P is considered as the set of constraints. Constraint-based approach tries to find out the DAG that satisfies IND_P based on theory of *d*-separation. In other words the set of *d*-separations of the best DAG pattern are the same as IND_P.

Example 4.2: Suppose we have $V = \{X, Y, Z\}$ and $IND_P = \{I(X,Y)\}$. Because X and Z isn't *d*-separated from any set, there must be a link between X and Z. In similar, there is must be a link between Y and Z. We have:



Because $X–Z–Y$ is uncoupled chain and there is a *d*-separation $I(X,Y)$, the chain $X–Z–Y$ should be converged.



If the number of variables is large we need effective algorithms. The simple algorithm includes two steps:

Firstly, the structure of DAG is drafted as "skeleton". If there is no conditional independence relating to X_i and X_j then the link between them is created. So skeleton is the undirected graph which contains variables (nodes) and links.

The second step is to determine direction of links by applying four following rules in sequence rule 1, rule 2, rule 3, rule 4:

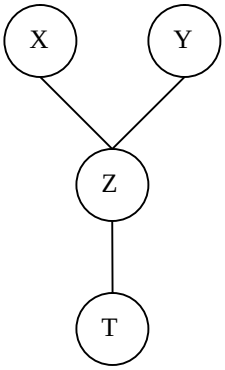
Rule 1: If the uncoupled chain $X–Z–Y$ exists and Z isn't in any set that d-separate X from Y then this chain is assumed convergent: $X \rightarrow Z \leftarrow Y$

Rule 2: If the uncoupled chain $X \rightarrow Z–Y$ exists (having an edge $X \rightarrow Z$) then this chain is assumed serial path: $X \rightarrow Z \rightarrow Y$.

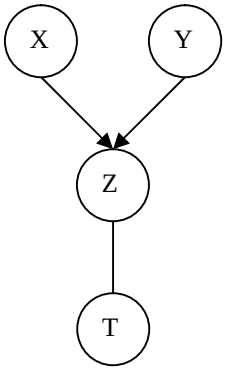
Rule 3: If the edge $X \rightarrow Y$ caused a directed cycle at a position in network then it is reversed: $X \leftarrow Y$. This rule is applied to remove directed cycles so that the expected BN is a DAG.

Rule 4: If all rules 1, 2, 3 are consumed the all remaining links have arbitrary direction.

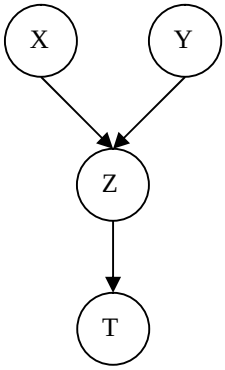
Example 4.3: Suppose we have $V = \{X, Y, Z, T\}$ and $IND_P = \{I(X,Y), I(X,T), I(Y,T)\}$. Because there is no conditional independence between X and Y, between Z and T, the "skeleton" is drafted as below:



Applying *rule 1*: Because the uncoupled chain $X–Z–Y$ exists and Z isn't in any set that d-separate X from Y, this chain is assumed convergent: $X \rightarrow Z \leftarrow Y$



Applying *rule 2*: Because the uncoupled chain $X \rightarrow Z - T$ exists, we have the assumed serial path: $X \rightarrow Z \rightarrow T$.



5. Conclusion

Three significant domains of Bayesian network (BN) are inference mechanism, parameter learning and structure learning. The first domain tells the usability of BN and the others indicates how to build up BN. The ideology of BN is to apply a mathematical inference tool (namely Bayesian rule) into a graph with expectation of extending and enhancing the ability of such tool so as to solve realistic problems,

especially diagnosis domain.

However in the process of developing BN, there are many problems involving in real number (continuous case) and nodes dependency. This report focuses on discrete case when the probability of each node is discrete CPT, not continuous PDF. The first-order Markov condition has important role in BN study when there is an assumption "nodes are dependent on only their direct parents". If the first-order Markov condition isn't satisfied, many inference and learning algorithms go wrong. I think that BN will get more potential and enjoyable if first-order (Markov) condition is replaced by *n*-order condition.

Moreover the parameter & structure learning becomes difficult when training data is missing (not complete). Missing data problem is introduced in section 3 but its detail goes beyond this report. I hope that we have a chance to discuss about it.

Finally, BN discussed here is "static" BN because the temporal relationships among nodes aren't concerned. The "static" BN is represented at only one time point. Otherwise dynamic Bayesian network (DBN) aims to model the temporal relationships among nodes. The process of inference is concerned in time series; in some realistic case this is necessary. However the cost of inference and learning in DBN is much higher than BN because the size of DBN gets huge for long-time process. Because of the limitation of this report, the algorithm that keeps the size of DBN intact (not changed) isn't introduced here. In general, the essence of such algorithm is to take advantage of both Markov condition and knowledge (inference) accumulation. Due to the complexity of DBN, we should consider to choose which one (BN or DBN) to apply into concrete domain. It depends on what your domain is and what your purpose is.

Reference

1. Richard E. Neapolitan (2003). Learning Bayesian Networks. Northeastern Illinois University Chicago, Illinois 2003.

2. David Heckerman (1995). A Tutorial on Learning With Bayesian Networks. Technical Report MSR-TR-95-06. Microsoft Research Advanced Technology Division, Microsoft Corporation.

3. Man Nguyen (2009). Man Nguyen documents and publications, http://www.cse.hcmut.edu.vn/~mnguyen/research_preprints.html. University of Technology, Ho Chi Minh city, Vietnam.

4. Judea Pearl (1986). Fusion, Propagation, and Structuring in Belief Networks. Artificial Intelligence, Vol. 29, 1986.

5. Judea Pearl (1988). Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann Publish, San Mateo, California, 1988. ISBN-13: 978-1558604797

6. Sean Borman (2009). The Expectation Maximization Algorithm - A short tutorial, last updated January 09, 2009. Department of Electrical Engineering at the University of Notre Dame, in South Bend, Indiana.

7. Kevin P Murphy (1998). A Brief Introduction to Graphical Models and Bayesian Networks at http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html. Department of Computer Science, University of British Columbia.