**Overview of Bayesian Network**

Loc Nguyen

International Engineering and Technology Institute (IETI), Hong Kong

Anum Shafiq

Preston University, Islamabad, Pakistan

# 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 may cause confusions 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. The report includes 5 main sections that cover principles of Bayesian network. The section 1 is an introduction to Bayesian network giving some basic concepts. Advanced concepts are mentioned in section 2. Inference mechanism of Bayesian network is described in section 3. Parameter learning which tells us how to update parameters of Bayesian network is described in section 4. Section 5 focuses on structure learning which mentions how to build up Bayesian network. In general, three main subjects of BN are inference, parameter learning, and structure learning which are mentioned in successive sections 3, 4, and 5. Section 6 is the optional one mentioning applications of Bayesian network. Section 7 is the conclusions. Main contents of this reported are extracted from the book “Learning Bayesian Networks” by Richard E. Neapolitan (2003) and the PhD dissertation “A User Modeling for Adaptive Learning” by Loc Nguyen (2014).

|  |  |
| --- | --- |
|  | (9.99) |

# 1. Introduction

This introduction section starts with a little bit discussion of Bayesian inference which is the base of both Bayesian network and inference in Bayesian network described later. Note, main content of this reported are extracted from the book “Learning Bayesian Networks” by Richard E. Neapolitan (Neapolitan, 2003) and the PhD dissertation “A User Modeling for Adaptive Learning” by Loc Nguyen (Nguyen, 2014).

**Bayesian inference** (Wikipedia, Bayesian inference, 2006), a form of statistical method, 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 by an initial probability or prior probability. Note, in classical statistical theory, the random variable’s probability is objective (physical) through trials. But, in Bayesian method, the probability of hypothesis is “personal” because its initial value is set subjectively by expert. When evidences were gathered enough, the hypothesis is considered trustworthy.

Bayesian inference is based on so-called Bayes’ rule or Bayes’ theorem (Wikipedia, Bayesian inference, 2006) specified in equation 1.1 as follows:

|  |  |
| --- | --- |
|  | (1.1) |

Where,

* *H* is probability variable denoting a hypothesis existing before evidence.
* *D* is also probabilistic variable denoting an observed evidence. It is conventional that notations *d*, *D* and are used to denote evidence, evidences, evidence sample, data sample, sample, training data and corpus (another term for data sample). Data sample or evidence sample is defined as a set of data or a set of observations which is collected by an individual, a group of persons, a computer software or a business process, which focuses on a particular analysis purpose (Wikipedia, Sample (statistics), 2014). The term “data sample” is derived from statistics; please read the book “Applied Statistics and Probability for Engineers” by Montgomery and Runger (Montgomery & Runger, 2003, p. 4) for more details about sample and statistics.
* *P*(*H*) is *prior probability* of hypothesis *H*. It reflects the degree of subjective belief in hypothesis *H*.
* *P*(*H|D*), conditional probability of *H* with given *D*, is called *posterior probability*. It tells us the changed belief in hypothesis when occurring evidence. Whether or not the hypothesis in Bayesian inference is considered trustworthy is determined based on the posterior probability. In general, posterior probability is cornerstone of Bayesian inference.
* *P*(*D|H*) is conditional probability of occurring evidence *D* when hypothesis *H* was given. In fact, likelihood ratio is *P*(*D|H*)/ *P*(*D*) but *P*(*D*) is constant value. So we can consider *P*(*D*|*H*) as *likelihood function* of *H* with fixed *D*. Please pay attention to the conditional probability because it is mentioned over the whole research.
* *P*(*D*) is probability of occurring evidence *D* together all mutually exclusive cases of hypothesis. If *H* and *D* are discrete, then , otherwise with *H* and *D* being continuous, *f* denoting probability density function (Montgomery & Runger, 2003, p. 99). Because of being sum of products of prior probability and likelihood function, *P*(*D*) is called *marginal probability*.

Note: *H*, *D* must be random variables (Montgomery & Runger, 2003, p. 53) according to theory of probability and statistics and *P*(.) denotes *random probability*.

Beside Bayes’ rule, there are three other rules such as additional rule, multiplication rule and total probability rule which are relevant to conditional probability. Given two random events (or random variables) *X* and *Y*, the additional rule (Montgomery & Runger, 2003, p. 33) and multiplication rule (Montgomery & Runger, 2003, p. 44) are expressed in equations 1.2 and 1.3, respectively as follows:

|  |  |
| --- | --- |
|  | (1.2) |

|  |  |
| --- | --- |
|  | (1.3) |

Where notations and denote union operator and intersection operator in set theory (Wikipedia, Set (mathematics), 2014). Your attention please, when *X* and *Y* are numerical variables, notations and also denote operators “or” and “and” in theory logic (Rosen, 2012, pp. 1-12). If *X* and *Y* are mutually independent (mutually exclusive) then, and are often denoted as *X*+*Y* and *XY*, respectively and so, we have:

The probability *P*(*XY*)=*P*(*X*,*Y*) is often known as joint probability.

Given a complete set of mutually exclusive events *X*1, *X*2,…, *Xn* such that

The total probability rule (Montgomery & Runger, 2003, p. 44) is specified in equation 1.4 as follows:

|  |  |
| --- | --- |
|  | (1.4) |

If *X* and *Y* are continuous variables, the total probability rule is re-written in integral form as follows:

|  |  |
| --- | --- |
|  | (1.5) |

Note, *P*(*Y|X*) and *P*(*X*) are continuous functions known as probability density functions mentioned right after. Please pay attention to Bayes’ rule (equation 1.1) and total probability rule (equations 1.4 and 1.5) because they are used frequently over the whole research.

**Bayesian network (BN)** (Neapolitan, 2003, p. 40) is combination of graph theory and Bayesian inference. It is a directed acyclic graph (DAG) which has a set of nodes and a set of directed arcs; please pay attention to the terms “DAG” and “BN” because they are used over the whole research. By default, directed graphs in this report are DAGs if there no additional explanation. Each node represents a random variable which can be an evidence or hypothesis in Bayesian inference. Each arc reveals the relationship among two nodes. If there is the arc from node *A* to *B*, we call “*A* causes *B*” or “*A* is parent of *B*”, in other words, *A* depends conditionally on *B*. Otherwise there is no arc between *A* and *B*, it asserts the conditional independence. Note, in BN context, terms: *node and variable are the same*.

A node has a local Conditional Probability Distribution (CPD) with attention that conditional probability distribution is often called shortly *probability distribution* or *distribution*. If variables are discrete, CPD is simplified as Conditional Probability Table (CPT). If variables are continuous, CPD is often called conditional Probability Density Function (PDF) which will be mentioned in section 4 – how to learn CPT from beta density function. PDF can be called *density function*, in brief. CPD is the general term for both CPT and PDF; there is convention that CPD, CPT and PDF indicate both probability and conditional probability. In general, each CPD, CPT or PDF specifies a random variable and is known as the *probability distribution* or *distribution* of such random variable.

Another representation of CPD is cumulative distribution function (CDF) (Montgomery & Runger, 2003, p. 64) (Montgomery & Runger, 2003, p. 102) but CDF and PDF have the same meaning and they share interchangeable property when PDF is derivative of CDF; in other words, CDF is integral of PDF. In practical statistics, PDF is used more commonly than CDF is used and so, PDF is mentioned over the whole research. Note, notation *P*(.) often denotes probability and it can be used to denote PDF but we prefer to use lower case letters such as *f* and *g* to denote PDF. Given a variable having PDF *f*, we often state that “such variable has distribution *f* or such variable has density function *f*”. Let *F*(*X*) and *f*(*X*) be CDF and PDF, respectively, equation 1.6 is the definition of CDF and PDF.

|  |  |
| --- | --- |
|  | (1.6) |

Because this introduction section focuses on BN, please read (Montgomery & Runger, 2003, pp. 98-103) for more details about CDF and PDF.

Now please pay attention to the concept CPT because it occurs very frequently in the research; you can understand simply that CPT is essentially collection of discrete conditional probabilities of each node (variable). It is easy to infer that CPT is discrete form of PDF. When one node is conditionally dependent on another, there is a corresponding probability (in CPT or CPD) measuring the influence of causal node on this node. In case that node has no parent, its CPT *degenerates into prior probabilities*. This is the reason CPT is often identified with probabilities and conditional probabilities. This report focuses on discrete BN and so CPT is an important concept.

E.g., in figure 1.1, event “cloudy” is cause of event “rain” which in turn is cause of “grass is wet” (Murphy, 1998). So we have three causal 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 1.1.** Bayesian network (a classic example about wet grass)

Note that random variables *C*, *S*, *R*, and *W* denote phenomena or events such as cloudy, sprinkler, rain, and wet grass, respectively and the table next to each node expresses the CPT of such node. For instance, focusing on the CPT attached to node “Wet grass”, if it is rainy (*R*=1) and garden is sprinkled (*S*=1), it is almost certain that grass is wet (*W*=1). Such assertion can be represented mathematically by the condition probability of event “grass is wet” (*W*=1) given evident events “rain” (*R*=1) and “sprinkler” (*S*=1) is 0.99 as in the attached table, *P*(*W*=1|*R*=1, *S*=1) = 0.99. As seen, the conditional probability *P*(*W*=1|*R*=1,*S*=1) is an entry of the CPT attached to node “Wet grass”. In general, BN consists of two models such as qualitative model and quantitative model. Qualitative model is the structure as the graph shown in figure 1.1. Quantitative model includes parameters which are CPTs attached to nodes in BN. Thus, CPTs as well as conditional probabilities are known as parameters of BN. Parameter learning and structure learning will be mentioned in sections 4 and 5.

Beside important subjects of BN such as parameter learning and structure learning, there is a more essential subject which is inference mechanism inside BN when the inference mechanism is a very powerful mathematical tool that BN provides us. Before studying inference mechanism in this wet grass example, we should know some advanced concepts of Bayesian network.

Let {*X*1, *X*2,…, *Xn*} be the set of nodes in BN, the global joint probability distribution (GJPD) is defined as the probability function of event {*X*1=*x*1, *X*2=*x*2,…, *Xn=xn*} (Neapolitan, 2003, p. 24). Such joint probability function satisfies two conditions specified by equation 1.7:

|  |  |
| --- | --- |
|  | (1.7) |

Suppose *PAi* is the set of direct parent nodes such that *Xi* only depends on variables in *PAi.* In other words, the DAG satisfies Markov condition in which there is always an arc from each variable in *PAi* to *Xi* and no intermediate node between them. Markov condition will be mentioned in section 2 again. At that time, the joint probability *P*(*X*1, *X*2,…, *Xn*) is specified by equation 1.8.

|  |  |
| --- | --- |
|  | (1.8) |

As a convention, BN satisfies Markov condition and so its joint probability specified by equation 1.8 is product of all local CPTs with note that *P*(*Xi*|*PAi*) in equation 1.8 is CPT of *Xi*. According to Bayesian rule, given evidence (random variables) , the posterior probability *P*(*Xi*|) of variable *Xi* is computed in equation 1.9 as below:

|  |  |
| --- | --- |
|  | (1.9) |

Where *P*(*Xi*) is prior probability of random variable *Xi* and *P*(|*Xi*) is conditional probability of occurring given *Xi* and *P*() is probability of occurring together all mutually exclusive cases of *X*. From equations 1.8 and 1.9, we gain equation 1.10 as follows:

|  |  |
| --- | --- |
|  | (1.10) |

Where and are all possible values *X* = (*X*1, *X*2,…, *Xn*) with fixing (excluding) and fixing (excluding) , respectively. Note that evidence including at least one random variable *Xi* is a subset of *X* and the sign “\” denotes the subtraction (excluding) in set theory (Wikipedia, Set (mathematics), 2014). Please pay attention that the equation 1.10 is the base for inference inside Bayesian network, which is used over the whole research. Equations 1.9 and 1.10 are extensions of Bayes’ rule specified by equation 1.1. It is not easy to understand equation 1.10 and so, please see equations 1.12 and 1.13 which are advanced posterior probabilities applied into wet grass example in order to comprehend equation 1.10.

From figure 1.1 of wet grass example, we have:

Applying equation 1.8, *P*(*S*|*C*)=*P*(*S*) due to no 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 | C*, *R*, *S*), hence *P*(*W* | *C*, *R*, *S*)= *P*(*W* | *R*, *S*). In short, applying equation 1.8, we have equation 1.11 for determining global joint probability distribution of “wet grass” Bayesian network as follows:

|  |  |
| --- | --- |
|  | (1.11) |

Using **Bayesian inference**, 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 BN.

Reviewing figure 1.1, suppose *W* becomes evidence variable which is observed as 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 *R* (=1) and *S* (=1) in condition *W* (=1). Such probabilities called *explanations* for *W* are simple forms of equation 1.10, expended by equations 1.12 and 1.13 as follows:

|  |  |
| --- | --- |
|  | (1.12) |

|  |  |
| --- | --- |
|  | (1.13) |

Note that the numerator in the right side of equation 1.12 is the sum of possible probabilities over possible values of *C* and *S*. Concretely, we have an interpretation for the numerator as follows:

Applying equation 1.11 for global joint probability distribution of “wet grass” Bayesian network, we have:

It is easy to infer that there is the same interpretation for numerators and denominators in right sides of equations 1.12 and 1.13 and the previous equation 1.10 is also understood simply by this way when {*C*, *S*} = {*C*, *R*, *S*, *W*}\{*R*, *W*} and fixing {*R*, *W*}. In similar, we have:

In fact, equations 1.12 and 1.13 are expansions of equation 1.10. As a result, we have:

Obviously, the posterior probability of event “sprinkler” (*S*=1) is larger than the posterior probability of event “rain” (*R*=1) given evidence “wet grass” (*W*=1), which leads to conclusion that sprinkler is the most likely cause of wet grass. Now a short description of Bayesian is introduced. Next section will concern advanced concepts of Bayesian network.

# 2. Advanced concepts

Recall that Bayesian network (BN) is the directed acyclic graph (DAG) (Neapolitan, 2003, p. 40) 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→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,…, *Xn*} and *E* be a set of nodes and a set of edges, Let *G* = (*V*, *E*) denote a DAG where *V* is a set of nodes, *E* is a set of edges, and there is no directed cycle in *G*. The “wet grass” graph shown in figure 1.1 is a DAG. Figure 2.1 (Neapolitan, 2003, p. 72) is another example of a DAG which is structure of a BN.



**Figure 2.1.** An example of a DAG

Note that node *Xi* is also random variable. In this report, uppercase letters, for example *X*, *Y*, *Z*, often denote random variables or set of random variables whereas lowercase letters, for example *x*, *y*, *z*, often denote their instantiations. We should glance over other popular concepts (Neapolitan, 2003, p. 31), (Neapolitan, 2003, p. 71).

* If there is an edge between *X* and *Y* (*X*→*Y* or *X*←*Y*) then, *X* and *Y* are called *adjacent* each other(or *incident* to the edge). Given the edge *X*→*Y*, the tail is at *X* and the head is at *Y*.
* Given *k* nodes {*X*1, *X*2, *X*3,…, *Xk*} in such a way that every pair of node (*Xi*, *Xi*+1) are incident to the edge *Xi→Xi*+1 where 1 *i  k*–1, all edges that connects such *k* nodes compose a *path* from *X*1 to *Xk* denoted as [*X*1, *X*2, *X*3,…, *Xk*] or *X*1*→X*2*→…→Xk*. The nodes *X*2, *X*3,…, *Xk*–1 are called *interior* nodes of the path. A *sub-path* *Xm→…Xn* is the path from *Xm* to *Xn*: *Xm→Xm*+1*→…→Xn* where 1 *≤ m < n ≤ k*. A *directed cycle* is the path from a node to itself. A *simple path* is the path that has no directed cycle. A DAG is the graph that has no directed cycle. By default, directed graphs in this report are DAGs if there no additional explanation. Figures 1.1 and 2.1 are examples of DAG.
* If there is an edge from *X* to *Y* then, *X* is called *parent* of *Y*. 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* then, *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. A *cycle* is the chain from a node to itself. A *simple chain* is the chain that has no cycle. The concepts “adjacent”and “incident” are kept intact with link.
* A DAG *G* is a (directed) *tree* if every node except root has only one parent. A DAG *G* is called *single-connected* if there is only one chain (if exists) between two nodes. Of course, (directed) tree is single-connected DAG.

The strength of dependence between two nodes is quantified by conditional probability table (CPT) in discrete case. 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 *Xk* is binary node and it has two parents *Xi* and *Xj*, the CPT (or CPD) of *Xk* which is the conditional probability *P*(*Xk | Xi, Xj*) has eight entries:

|  |  |
| --- | --- |
| *P*(*Xk*=1*|Xi*=1, *Xj*=1) | *P*(*Xk*=0*|Xi*=1, *Xj*=1) |
| *P*(*Xk*=1*|Xi*=1, *Xj*=0) | *P*(*Xk*=0*|Xi*=1, *Xj*=0) |
| *P*(*Xk*=1|*Xi*=0, *Xj*=1) | *P*(*Xk*=0*|Xi*=0, *Xj*=1) |
| *P*(*Xk*=1|*Xi*=0, *Xj*=0) | *P*(*Xk*=0*|Xi*=0, *Xj*=0) |

It is asserted that if *Xi* is binary node and has *n* parents then its CPT has *2n*+1 entries. However only *2n* entries are specified in practice due to *P*(*Xi=*0 | …) = 1–*P*(*Xi*=1|…) when *Xi* is binary. In case that *Xi* has *k* possible values, each CPT has *kn* entries. Figure 1.1 shows an example of BN. Let *PAi* be the set of direct parents of node *Xi*, the (global) *joint probability distribution* of whole BN is defined as product of CPT(s) or CPD(s) of all nodes in case that Markov condition mentioned later is satisfied, according to equation 1.8.

BN is represented by its joint probability distribution *P* and its DAG. Let (*G*, *P*) denote a BN where *G* = (*V*, *E*) is the DAG and *P* is the joint probability distribution. Hence, BN is a combination of probabilistic model and graph model. Note, by default, *G* is a DAG.

Suppose BN has *n* binary nodes. In the worst case, each node has *n*–1 parents, thus, the joint probability can have *n\*2n* entries at most. There is a boom of CPTs. There is a restrictive criterion called Markov condition that makes the relationships (also CPT) among nodes simpler. Given Bayesian network (*G*, *P*) and three subsets of nodes: *A =* {*Xi*,…, *Xj*}, *B =* {*Xk*,…, *Xl*} and *C* = {*Xm*,…, *Xn*}:

* The denotation *IP*(*A*, *B*) indicates that *A* and *B* are independent (Neapolitan, 2003, p. 18), which means that *P*(*A*, *B*) = *P*(*A*)*P*(*B*). Note, the independence *IP*(*A*, *B*) here is defined based on the joint probability.
* The denotation *IP*(*A*, *B | C*) indicates that *A* and *B* are conditionally independent given *C* (Neapolitan, 2003, p. 19), which means that *P*(*A*, *B* | *C*) = *P*(*A* | *C*)*P*(*B* | *C*). Note, the independence *IP*(*A*, *B | C*) here is defined based on the joint probability.

According to definition 2.5 in (Neapolitan, 2003, p. 75), two conditional dependences *IP*(*A*1, *B*1 *| C*1) and *IP*(*A*2, *B*2 *| C*2) are *equivalent* if for every joint probability *P* of *V*, *IP*(*A*1, *B*1 *| C*1) holds if and only if *IP*(*A*2, *B*2 *| C*2) holds.

Let (*G*, *P*) be a BN, **Markov condition** (Neapolitan, 2003, p. 31) is stated that every node *X* is conditional independent from its non-descendants given its parents. In other word node *X* is only dependent on its directed parents. Equation 2.1 defines Markov condition (Neapolitan, 2003, p. 31).

|  |  |
| --- | --- |
|  | (2.1) |

Where *E* is the set of edges in *G*, *NDX* and *PAX* are the set of non-descendants of *X* and the set of parents of *X*, respectively. As a convention, *NDX* excludes *X* and *PAX* excludes *X* too. *NDX* is not empty but *PAX* can be empty. When *PAX* is empty, equation 2.1 becomes:

Note, Markov condition is defined based on the joint probability distribution *P* and so, equation 2.1 is interpreted as follows:

When *PAX* is empty,

Given two DAGs *G*1 and *G*2 shown in figure 2.2 and suppose the joint probability *P*(*X*, *Y*, *Z*) = *P*(*X*)*P*(*Y* | *X*)*P*(*Z* | *X*), we will test whether (*G*1, *P*) and (*G*2, *P*) satisfy Markov condition.



**Figure 2.2.** An example of two DAGs

For (*G*1, *P*), we only test whether *IP*(*Y*, *Z* | *X*) holds because there is only one *IP*(*Y*, *Z* | *X*) in *G*1. According equation 1.10, we have:

Due to:

It implies *IP*(*Y*, *Z* | *X*) holds in *G*1 and so (*G*1, *P*) satisfies Markov condition. For (*G*2, *P*), we only test whether *IP*(*Y*, *Z*) holds because there is only one *IP*(*Y*, *Z*) in *G*2. According to the total probability rule, we have:

Whereas,

Obviously, *P*(*Y* = *y*, *Z* = *z*) is different from the product *P*(*Y* = *y*)*P*(*Z* = *z*) for some values of *P*(*X*, *Y*, *Z*) at least, which means that *IP*(*Y*, *Z*) in *G*2 does not hold. In other words, (*G*2, *P*) does not satisfy Markov condition. However, if the joint probability is re-defined as *P*(*X*, *Y*, *Z*) = *P*(*Y*)*P*(*Z*)*P*(*X* | *Y*, *Z*) then, (*G*2, *P*) satisfies Markov condition due to:

According to theorem 1.4 in (Neapolitan, 2003, p. 34), if Markov condition is satisfied, the join probability is product of conditions probabilities of nodes given their parents (Neapolitan, 2003, p. 34) whenever these conditional probabilities exist, according to aforementioned equation 1.8.

In practice, most of BNs satisfy Markov condition. Because inference and structure learning algorithms are based on Markov condition, please pay attention to it. Conversely, according to theorem 1.5 in (Neapolitan, 2003, p. 37), given a DAG *G* and every node *Xi* in *G* has a condition probability *P*(*Xi* | *PAi*) on its parents then, the BN (*G*, *P*) composed of *G* and the joint probability will satisfies Markov condition. BN is constructed in practice with theorem 1.5 in (Neapolitan, 2003, p. 37).

Every joint probability *P* owns “inherent” conditional independences. When a BN (*G*, *P*) satisfies Markov condition, each “Markov” conditional independence of each node from its non-descendants given its parents belongs to “inherent” conditional independences of *P* via equation 2.1. In other words, that (*G*, *P*) satisfies Markov conditions means *G* entails only a subset or whole of “inherent” conditional independences of *P*. Given *G*1 in figure 2.2 and *P*(*X*, *Y*, *Z*) = *P*(*X*)*P*(*Y* | *X*)*P*(*Z* | *X*), the *IP*(*Y*, *Z* | *X*) is a “Markov” conditional independence of *Y* (and *Z*) given parent node *X* and it is also a “inherent” conditional independence derived from *P*. There is a question: whether the Markov condition entails other conditional independences different from “Markov” conditional independences of nodes? Neapolitan (Neapolitan, 2003, p. 66) said yes. According to definition 2.1 in (Neapolitan, 2003, p. 66), let *G* = (*V*, *E*) be a DAG, where *V* is a set of random variables. We say that, based on the Markov condition, *G* entails conditional independency *IP*(*A*, *B* | *C*) for *A*, *B*, *C* ⊆ *V* if *IP*(*A*, *B* | *C*) holds for very where is the set of all joint probabilities that (*G*, *P*) satisfies Markov condition. We (Neapolitan, 2003, p. 66) also say the Markov condition entails the conditional independency for *G* and that the conditional independency is in *G*. According to lemma 2.2 in (Neapolitan, 2003, p. 75), any conditional independency entailed by a DAG, based on the Markov condition, is equivalent to a conditional independency among disjoint sets of random variables.

Independence in a BN (*G*, *P*) until now is defined based on the joint probability. For instance,

However, independence in BN can be defined by topology of the DAG *G* = (*V*, *E*). The concept of d-separation is used to determine the topological independence. There are some important concepts that constitute the d-separation concept (Neapolitan, 2003, p. 71):

* The chain *X→Z→Y* or *X←Z←Y* is called serial path. It is also called a *head-to-tail meeting*, in which the edges meet head-to-tail at *Z* and *Z* is a head-to-tail node on the chain.
* The chain *X←Z→Y* is called divergent path. It is also called a *tail-to-tail meeting*, in which the edges meet tail-to-tail at *Z* and *Z* is a tail-to-tail node on the chain.
* The chain *X→Z←Y* is called convergent path. It is also called a *head-to-head* *meeting*, in which the edges meet head-to-head at *Z*, and *Z* is a head-to-head node on the chain.
* The chain *X–Z–Y* is called uncoupled chain if *X* and *Y* aren’t adjacent. It is also called an *uncoupled meeting*.

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

Let *X*, *Y* be two nodes and let *C* be a subset of nodes such that *C V*, , , and *X* ≠ *Y*. Given the chain *p* between *X* and *Y*, *p* is *blocked* by *C* if and only if one of two following conditions is satisfied (Neapolitan, 2003, pp. 71-72):

1. There is an intermediate node on *p* so that all edges on *p* incident to *Z* are serial (head-to-tail meeting) or divergent (tail-to-tail meeting) at *Z*.
2. There is an intermediate node *Z* on *p* so that:

* *Z* and all descendants of *Z* are not in *C* ().
* All edges op *p* incident to *Z* are convergent (head-to-head meeting).

The DAG shown in figure 2.1 is now used for illustrating d-separation.



In figure 2.1, the chain [*Y* – *X* – *Z* – *S*] is blocked by {*X*} because the edges on the chain incident to *X* meet tail-to-tail at *X*, according to the condition 1 (Neapolitan, 2003, p. 72). That chain is also blocked by {*Z*} because the edges on the chain incident to *Z* meet head-to-tail at *Z*, according to the condition 1 (Neapolitan, 2003, p. 72). The chain [*W* – *Y* – *R* – *Z* – *S*] is blocked by because , , and the edges on the chain incident to *R* meet head-to-head at *R*, according to the condition 2 (Neapolitan, 2003, p. 72).

According to definition 2.3 in (Neapolitan, 2003, p. 72), *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*. In figure 2.1, *X* and *R* are d-separated by {*Y*, *Z*} because the chain [*X* – *Y* – *R*] is blocked at *Y*, and the chain [*X* – *Z* – *R*] is blocked at *Z* (Neapolitan, 2003, p. 72). Similarly, *X* and *T* are d-separated by {*Y*, *Z*} because the chain [*X* – *Y* – *R* – *T*] is blocked at *Y*, the chain [*X* – *Z* – *R* – *T*] is blocked at *Z*, and the chain [*X* – *Z* – *S* – *R* – *T*] is blocked at *Z* and at *S* (Neapolitan, 2003, p. 72).

According to definition 2.3 in (Neapolitan, 2003, p. 73), given DAG *G* = (*V*, *E*) and given *A*, *B*, and *C* are mutually disjoint subsets of nodes, if for every node and , *X* and *Y* are d-separated by *C* then, we have a topological independence denoted as follows:

If *C* is empty, we write:

Lemma 2.1 in (Neapolitan, 2003, p. 74) is used to link the probabilistic independence (conditional independence as a convention) and the topological independence (d-separation). According to this lemma, a BN (*G*, *P*) satisfies Markov condition if and only if

|  |  |
| --- | --- |
|  | (2.2) |

Where *A*, *B*, and *C* are mutually disjoint subsets of nodes. From lemma 2.1 in (Neapolitan, 2003, p. 74), when the BN (*G*, *P*) satisfies Markov condition, *G* is called an *independence map* of *P*.

According to lemma 2.3 in (Neapolitan, 2003, p. 75), let *G* = (*V*, *E*) be a DAG, and be the set of all probability distributions *P* such that the BN (*G*, *P*) satisfies the Markov condition. Then for every three mutually disjoint subsets *A*, *B*, *C* ⊆ *V*,

|  |  |
| --- | --- |
|  | (2.3) |

According to definition 2.6 in (Neapolitan, 2003, p. 76), a conditional independence *IP*(*A*, *B* | *C*) is *identified by d-separation* in *G* if one of two following conditions is satisfied:

1. *IG*(*A*, *B* | *C*) holds.
2. *A*, *B*, and *C* are not mutually disjoint; *A*’, *B*’, and *C*’ are mutually disjoint, *IP*(*A*, *B* | *C*) and *IP*(*A*’, *B*’ | *C*’) are equivalent, and we have *IG*(*A*’, *B*’ | *C*’).

Recall that two conditional dependences *IP*(*A*1, *B*1 *| C*1) and *IP*(*A*2, *B*2 *| C*2) are equivalent if for every joint probability *P* of *V*, *IP*(*A*1, *B*1 *| C*1) holds if and only if *IP*(*A*2, *B*2 *| C*2) holds. As a result, according theorem 2.1 in (Neapolitan, 2003, p. 76), based on the Markov condition, a DAG *G* entails all and only conditional independencies that are identified by d-separation in *G*.

DAGs which have the same set of nodes are **Markov equivalent** if and only if they have same *d*-separations. In other words, DAGs that are Markov equivalent have the same topological independences. Equation 2.4 (Neapolitan, 2003, pp. 84-85) defines Markov equivalence given two DAGs such as *G*1 = (*V*, *E*1) and *G*2 = (*V*, *E*2).

|  |  |
| --- | --- |
|  | (2.4) |

Where *A*, *B*, and *C* are mutually disjoint subsets of *V*. Shortly, Markov condition is defined based on joint probability distribution whereas Markov equivalence is defined based on topology of DAG (d-separation). Hence, theorem 2.3 and corollary 2.1 in (Neapolitan, 2003, p. 85) are used to connect Markov condition and Markov equivalence. According to theorem 2.3 (Neapolitan, 2003, p. 85), two DAGs are Markov equivalent if and only if, based on the Markov condition, they entail the same conditional independencies. According to corollary 2.1 (Neapolitan, 2003, p. 85), let *G*1 = (*V*, *E*1) and *G*2 = (*V*, *E*2) be two DAGs containing the same set of variables *V* then, *G*1 and *G*2 are Markov equivalent if and only if for every probability distribution *P* of *V*, (*G*1, *P*) satisfies the Markov condition if and only if (*G*2, *P*) satisfies the Markov condition.

According to lemma 2.6 in (Neapolitan, 2003, p. 86), If two DAGs *G*1 and *G*2 are Markov equivalent, then arbitrary nodes *X* and *Y* are adjacent in *G*1 if and only if they are adjacent in *G*2. So, Markov equivalent DAGs have the same links (edges without regard for direction). According to theorem 2.4 in (Neapolitan, 2003, p. 87), two DAGs *G*1 and *G*2 are Markov equivalent if and only if they have the same links (edges without regard for direction) and the same set of uncoupled head-to-head meetings (*X→Z←Y*). From lemma 2.6 and theorem 2.4 in (Neapolitan, 2003, pp. 86-87), Neapolitan (Neapolitan, 2003, p. 91) stated that *Markov equivalent class* can be represented with a graph that has the same links and the same uncoupled head-to-head meeting as the DAGs in the class. Markov equivalence divides all DAGs into disjoint Markov equivalent classes. Figure 2.3 (Neapolitan, 2003, p. 85) shows three DAGs of the same Markov equivalent class.



**Figure 2.3.** Three DAGs of the same Markov equivalent class

If we assign a direction to a link and such assignment does not produce a head-to-head meeting then, we create a new member of the existing equivalent class but we do not create a new equivalent class. For instance (Neapolitan, 2003, p. 91), if a Markov equivalent class has the edge *X→Z* and the uncoupled meeting *X→Z−W* is not head-to-head then, all the DAGs in the equivalent class must have *Z*−*W* oriented as *Z→W*. According to (Neapolitan, 2003, p. 91), a **DAG pattern** is defined for a Markov equivalence class to be the graph that has the same links as the DAGs in the equivalence class and has oriented all and only the edges common to all DAGs in the equivalent class. Edges (directed links) in DAG pattern are called *compelled edges*. In general, DAG pattern is the representation of Markov equivalent class. Figure 2.4 is a DAG pattern of the Markov equivalent class in figure 2.3.



**Figure 2.4.** DAG pattern of the Markov equivalent class in figure 2.3

DAG pattern is the core of Bayesian structure learning. Note, DAG pattern can have both edges and links; so, DAG pattern is not a DAG and it is only a graph. Therefore, we should survey properties of DAG pattern.

According to definition 2.8 in (Neapolitan, 2003, p. 91), let *gp* be a DAG pattern whose nodes are the elements of *V*, and *A*, *B*, and *C* be mutually disjoint subsets of *V*. Then *A* and *B* are d-separated by *C* in *gp* if *A* and *B* are d-separated by *C* in every DAG *G* in the Markov equivalent class represented by *gp*. Two lemmas 2.7 and 2.8 in (Neapolitan, 2003, p. 91) are derived from this definition. According to lemma 2.7 (Neapolitan, 2003, p. 91), let *gp* be DAG pattern and *X* and *Y* be nodes in *gp* then, *X* and *Y* are adjacent in *gp* if and only if they are not d-separated by some set in *gp*. According to lemma 2.8 (Neapolitan, 2003, p. 91), suppose we have a DAG pattern *gp* and an uncoupled meeting *X*−*Z*−*Y* then, the followings are equivalent:

1. *X*−*Z*−*Y* is a head-to-head meeting.
2. There exists a set not containing *Z* that d-separates *X* and *Y*.
3. All sets containing *Z* do not d-separate *X* and *Y*.

Recall that when a BN (*G*, *P*) satisfies Markov condition, *G* is called an independence map of *P* according to lemma 2.1 in (Neapolitan, 2003, p. 74), which causes that then every DAG which is Markov equivalent to *G* is also an independence map of *P*. As a result (Neapolitan, 2003, p. 92), based on Markov condition, DAG pattern *gp* representing the equivalence class is an independence map of *P*.

From theorem 1.4 (Neapolitan, 2003, p. 34) and theorem 1.5 (Neapolitan, 2003, p. 37), Markov condition entail the independence but it does not entail the dependence. Concretely (Neapolitan, 2003, p. 65), given a BN (*G*, *P*) satisfies Markov condition, the absence of an edge from node *X* to node *Y* implies independence of *Y* from *X* but the presence of an edge from node *X* to node *Y* does not implies dependence of *X* and *Y*. Another condition called faithfulness condition (Neapolitan, 2003, p. 65) will entail both independence and dependence between two nodes based on both absence and presence of their edge. Faithfulness condition is essential to constraint-based structure learning (Neapolitan, 2003, p. 542). Before defining faithfulness condition, we need to survey some relevant concepts. A DAG is called *complete DAG* (Neapolitan, 2003, p. 94) if there always exits an edge between two arbitrary nodes. Given two nodes *X* and *Y*, there is a *direct dependency* between *X* and *Y* if {*X*} and {*Y*} are not conditionally dependent (Neapolitan, 2003, p. 94) given any subset of *V*. According to definition 2.9, given a joint probability *P* and a DAG *G* = (*V*, *E*), the (*G*, *P*) satisfies **faithfulness condition** if two following conditions are satisfied:

1. (*G*, *P*) satisfies Markov condition, which means that *G* entails only conditional independences in *P*.
2. All conditional independences in *P* are entailed by *G*, based on Markov condition.

In other words, a (*G*, *P*) satisfies faithfulness condition if *G* entails only and all conditional independences in *P*, based on Markov condition. So, faithfulness condition is stronger than Markov condition. When (*G*, *P*) satisfies the faithfulness condition, we say *P* and *G* are *faithful to each other*, and we say *G* is a *perfect map* of *P* (Neapolitan, 2003, p. 95).

Theorem 2.5 in (Neapolitan, 2003, p. 96) and theorem (Neapolitan, 2003, p. 97) connect faithfulness condition and topological independences. According to theorem 2.5 in (Neapolitan, 2003, p. 96), a (*G*, *P*) satisfies faithfulness condition if and only if all and only conditional independencies in *P* are identified by d-separation in the DAG *G*. According to theorem 2.6 in (Neapolitan, 2003, p. 97), if (*G*, *P*) satisfies faithfulness condition, then *P* satisfies this condition with all and only DAGs that are Markov equivalent to *G*. Furthermore, if we let *gp* be the DAG pattern corresponding to this Markov equivalence class, the d-separations in *gp* identify all and only conditional independencies in *P*. We say that *gp* and *P* are faithful to each other, and *gp* is a perfect map of *P*.

According to theorem 2.7 in (Neapolitan, 2003, p. 99), suppose a joint probability distribution *P* admits some faithful DAG representation then, *gp* is the DAG pattern faithful to *P* if and only if the two following conditions are satisfied:

1. *X* and *Y* are adjacent in *gp* if and only if there is no subset such that *IP*({*X*}, {*Y*} | *S*) holds. That is, *X* and *Y* are adjacent if and only if there is a direct dependency between *X* and *Y*.
2. Any chain [*X* − *Z* − *Y*] is a head-to-head meeting in *gp* if and only if implies *IP*({*X*}, {*Y*} | *S*) does not hold.

In general, if faithfulness condition is satisfied, independence and dependence in DAG *G* are as same as independence and dependence in joint probability *P*. In other words, absence and presence of an edge in *G* implies independence and dependence in *P*. Faithfulness condition makes the pair (*G*, *P*) are matched totally, which causes that the BN (*G*, *P*) is perfect.

Now advanced concepts relevant to BN were introduced in sections 1 and 2. Three main subjects of BN are inference, parameter learning, and structure learning which are mentioned in successive sections 3, 4, and 5. Recall that this report focuses on discrete BN. In other words, nodes are random discrete variables attached to CPTs. Especially, random binary variables are preferred.

# 3. Inference

The essence of Bayesian reference is to compute the posterior probabilities of nodes given evidences. Equation 1.10 is the base of simple inference, which is an extension of Bayes’ rule specified equation 1.1. Note that evidences or conditions are also nodes which are observed and have concrete values. Going back example “wet grass” in section 1, 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*, according to equation 1.12 with note that equation 1.12 is an interpretation of equation 1.10.

Here we make clear equation 1.0 again. Let *V* = {*X*1, *X*2,…, *Xn*} be a whole set of nodes. Let *D =* {*Xm*+1, *Xm*+2,…, *Xm*+*u*} be a set of evidences, . Let *d =* {*xm*+1, *xm*+2,…, *xm*+*u*} be the instantiation of *D*. In general case, the marginal probability of *Xi* = *xi* is:

Where *P*(*X*1, *X*2,…, *Xn*) is the global joint probability. The marginal probability of *D* = *d* is:

The posterior probability of *Xi* = *xi* given *D = d* is:

|  |  |
| --- | --- |
|  | (3.1) |

The equation 3.1 is the basic idea of simple inference, which is an interpretation of equation 1.10. 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 advantages of Markov condition: Pearl’s message propagation (Pearl, 1986), (Neapolitan, 2003, pp. 126-156) is well-known algorithm.
* Taking advantages of the structure of DAG: Noisy OR-gate model (Neapolitan, 2003, pp. 156-160) and Junction Tree (Neapolitan, 2003, p. 161) are well-known algorithms.
* Reducing the amount of numeric operations computed in marginal probability: Symbolic probabilistic inference (SPI) algorithm (Neapolitan, 2003, pp. 162-170) is the well-known algorithm which finds optimal factoring for marginal probability computation.

## 3.1. Markov condition based inference

When a BN (*G*, *P*) satisfies Markov condition, each node of *G* is associated with a CPT. The well-known algorithm that takes advantages of conditional independences entailed by Markov condition is Pearl’s message propagation algorithm (Pearl, 1986). Pearl’s algorithm starts with a BN (*G*, *P*) where the DAG *G* is a tree.

Suppose Bayesian network is DAG *G* = (*E*, *V*) which is a tree having only one root. Given a set of evidence nodes *D V*; every node in *D* has concrete value. Let *DX* is the sub-set of *D* including *X* and descendants of *X* and let *NX* be the sub-set of *D* including *X* and non-descendant of *X*. Let *CX* and *PAX* be children and parents of *X*, respectively. Note, both *CX* and *PAX* exclude *X*. Let *R* be root node. Let *O* be evidence node, *O* *D*. In figure 3.1.1, *NX* is green and *DX* is red.



**Figure 3.1.1.** *X*, *DX*, and *NX*

The essence of inference is to compute the posterior probability *P*(*X|D*) for every *X*. We have (Neapolitan, 2003, p. 128):

(Due to Bayes’ rule)

(Because *DX* and *NX* are conditionally independent given *X*)

(Due to Bayes’ rule)

Where is the constant independent from *X*. Let *λ*(*X*) = *P*(*DX|X*) and *π*(*X*) = *P*(*X|NX*), equation 3.1.1 is used to calculate the posterior probability *P*(*X|D*), which is the base of Pearl’s message propagation algorithm (Neapolitan, 2003, p. 128).

|  |  |
| --- | --- |
|  | (3.1.1) |

The *λ*(*X*) and *π*(*X*) are called *λ* value and *π* value of *X*, respectively. For each child *Y* of *X*, let *λY*(*X*) be *λ* message that is propagated up from *Y* to *X*. Note that *λY*(*X*) is conditional probability of *DY* given *X*. Equation 3.1.2 specifies the *λ* message *λY*(*X*).

|  |  |
| --- | --- |
|  | (3.1.2) |

Following is the proof of equation 3.1.2.

For each parent *Z* of *X*, let *πX*(*Z*) be *π* message that is propagated down from *Z* to *X*. Note that *πX*(*Z*) is conditional probability of *X* given *NX*. Equation 3.1.3 specifies the *π* message *πX*(*Z*).

|  |  |
| --- | --- |
|  | (3.1.3) |

Where the notation “” denote proportion and *CZ*\{*X*} is the set of *Z*’s children except *X*. Following is the proof of equation 3.1.3.

(Due to Bayes’ rule)

(Because *NZ* and are conditionally independent give *Z*)

(Where is the constant independent from *X* and *Z*)

(Because *Z*’s children are mutually independent)

Don’t worry about *πX*(*Z*) which is proportioned to and the posterior probability *P*(*X|D*) itself is also proportioned to *λ*(*X*)and *π*(*X*) via constant *α*. These constants will be eliminated when *P*(*X|D*) is normalized. For example, given binary random variable *X*, if *P*(*X*=1 | *D*) = *αp*1 and *P*(*X*=0 | *D*) = *αp*2, they are normalized as follows.

Now we have:

* Value *λ*(*X*) = *P*(*DX|X*).
* Message *λY*(*X*) is calculated according to equation 3.1.2 for each .
* Value *π* (*X*) = *P*(*X|NX*).
* Message *πX*(*Z*) is calculated according to equation 3.1.3 for each .

The *λ* and *π* values will be updated according to *λ* and *π* messages, mentioned later. Whenever evidence 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, as seen in figure 3.1.2.



**Figure 3.1.2.** Pearl propagation algorithm (*X* is focused node)

Please pay attention to four following cases when updating *λ* value and *π* value at certain variable *X* (Neapolitan, 2003, pp. 127-128):

1. Ifand suppose *X*’s instantiation (value) is *x* then:

*λ*(*X=x*) = *P*(*x|x*) = 1 due to and Markov condition. So *λ*(*X*≠*x*) = 0.

*π*(*X=x*) = *P*(*x|x*) = 1 due to and Markov condition. So *π*(*X*≠*x*) = 0.

*P*(*X=x|D*) = 1 and *P*(*X*≠*x|D*) = 0.

1. If *X D* and *X* is leaf then:

*λ*(*X*) = *P*(Ø|*X*) = 1 due to *DX* = Ø.

*π*(*X*) is computed as if *X* were intermediate variable according to case 4.

*P*(*X|D*) = *απ*(*X*).

1. If *X D* and *X* is root then:

*λ*(*X*) is computed as if *X* were intermediate variable according case 4.

*π*(*X*) = *P*(*X|*Ø) = *P*(*X*).

*P*(*X|D*) = *αλ*(*X*)*P*(*X*).

1. If *X D* and *X* is intermediate variable then, *λ*(*X*) and *π*(*X*) are computed according equations 3.1.4 and 3.1.5. Later on *P*(*X|D*) is calculated according to equation 3.1.1, *P*(*X|D*)*= αλ*(*X)π*(*X*).

Hence, equation 3.1.4 is used to update and *λ* value based on *λ* message.

|  |  |
| --- | --- |
|  | (3.1.4) |

Following is the proof of equation 3.1.4.

(Because *X*’s children are mutually independent).

Equation 3.1.5is used to update *π* value according to *π* message.

|  |  |
| --- | --- |
|  | (3.1.5) |

Following is the proof of equation 3.1.5.

Where *Z* is parent of *X*. The C-like pseudo-code for Pearl’s algorithm shown below includes four functions:

* Function “*init”* initialize *π* value for every node. At that time the set of evidence nodes *D* is empty.
* Function “*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 “*propagate*\_*up*\_*λ*\_*message*”, and propagates down *π* message over all children of *O* by calling function “*propagate*\_*down*\_*π*\_*message*”.
* Function “*propagate*\_*up*\_*λ*\_*message*” computes *λ* value, posterior probability of current node, and continues to propagate upwards and downwards *λ* and *π* messages by calling itself and function “*propagate*\_*down*\_*π*\_*message*”. Process of propagation stops when there is no node to be propagated.
* Function “*propagate*\_*down*\_*π*\_*message*” computes *π* value, 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.

Followings are descriptions of these functions.

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*\_*down*\_*π*\_*message*(*R*, *K*);

}

void *update*(*O*, *o*)

{

*D = D O*;

*λ*(*O*=*o*) = *π*(*O*=*o*) = *P*(*O*=*o|D*) = 1; //due to *OD* and *O*=*o*

*λ*(*O*≠*o*) = *π*(*O*≠*o*) = *P*(*O*≠*o|D*) = 0; //due to *OD* and *O*≠*o*

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*\_*down*\_*π*\_*message*(*O*, *K*);

}

void *propagate*\_*up*\_*λ*\_*message*(*Y*, *X*)

{

; //*Y* propagate upwards *λ* message

; //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*\_*down*\_*π*\_*message*(*X*, *K*);

}

void *propagate*\_*down*\_*π*\_*message*(*Z*, *X*)

{

; //*Y* propagate downwards *π* message

; //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*\_*down*\_*π*\_*message*(*X*, *K*);

}

Given BN (*G*, *P*) shown in figure 3.1.3 where DAG *G* is a tree satisfying Markov condition and each binary node has a CPT, suppose evidence *X* has value 1. Hence, we need to compute posterior probabilities of *T*, *Y*, *Z* in condition *X*=1.



**Figure 3.1.3.** Bayesian network with CPTs

Firstly, function “*init*” is called to initialize network.

*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

*λX*(*Z*=1) = *λX*(*Z*=0) = 1

*λY*(*Z*=1) = *λY*(*Z*=0) = 1

*λT*(*X*=1) = *λT*(*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:

*πX*(*Z*=1) = *π*(*Z*=1)*λX*(*Z*=1) = 1\*0.6 = 0.6

*πX*(*Z*=0) = *π*(*Z*=0)*λX*(*Z*=0) = 1\*0.4 = 0.4

*π*(*X*=1) = *P*(*X*=1|*Z*=1)*πX*(*Z*=1) + *P*(*X*=1|*Z*=0)*πX*(*Z*=0) = 0.7\*0.6 + 0.2\*0.4 = 0.5

*π*(*X*=0) = *P*(*X*=0|*Z*=1)*πX*(*Z*=1) + *P*(*X*=0|*Z*=0)*πX*(*Z*=0) = 0.3\*0.6 + 0.8\*0.4 = 0.5

*P*(*X*=1) = *αλ*(*X*=1)*π*(*X*=1) = *α*\*1\*0.5 *= α*0.5

*P*(*X*=0) = *αλ*(*X*=0)*π*(*X*=0) = *α*\*1\*0.5 = *α*0.5

Normalizing *P*(*X*)

*P*(*X*=1) = (*α*0.5) / (*α*0.5 + *α*0.5)= 0.5

*P*(*X*=0) = (*α*0.5) / (*α*0.5 + *α*0.5)= 0.5

Calling *propagate*\_*down*\_*π*\_*message*(*X*, *T*)

Then, function *propagate*\_*down*\_*π*\_*message*(*X*, *T*) is executed:

*πT*(*X*=1) = *π*(*X*=1) = 0.5

*πT*(*X*=0) = *π*(*X*=0) = 0.5

*π*(*T*=1) = *P*(*T*=1|*X*=1)*πT*(*X*=1) + *P*(*T*=1|*X*=0)*πT*(*X*=0) = 0.9\*0.5 + 0.4\*0.5 = 0.65

*π*(*T*=0) = *P*(*T*=0|*X*=1)*πT*(*X*=1) + *P*(*T*=0|*X*=0)*πT*(*X*=0) = 0.1\*0.5 + 0.6\*0.5 = 0.40

*P*(*T*=1) = *αλ*(*T*=1)*π*(*T*=1) = *α*\*1\*0.65 = *α*0.65

*P*(*T*=0) = *αλ*(*T*=0)*π*(*T*=0) = *α*\*1\*0.40 = *α*0.40

Normalizing *P*(*T*)

*P*(*T*=1) =(*α*0.65) / (*α*0.65 + *α*0.40) = 0.62

*P*(*T*=0) = (*α*0.40) / (*α*0.65 + *α*0.40) = 0.38

Then function *propagate*\_*down*\_*π*\_*message*(*Z*, *Y*) is executed:

*πY*(*Z*=1) = *π*(*Z*=1)*λY*(*Z*=1) = 1\*0.6 = 0.6

*πY*(*Z*=0) = *π*(*Z*=0)*λY*(*Z*=0) = 1\*0.4 = 0.4

*π*(*Y*=1) = *P*(*Y*=1|*Z*=1)*πX*(*Z*=1) + *P*(*Y*=1|*Z*=0)*πX*(*Z*=0) = 0.6\*0.6 + 0.3\*0.3 = 0.45

*π*(*Y*=0) = *P*(*Y*=0|*Z*=1)*πX*(*Z*=1) + *P*(*Y*=0|*Z*=0)*πX*(*Z*=0) = 0.3\*0.4 + 0.8\*0.7 = 0.68

*P*(*Y*=1) = *αλ*(*Y*=1)*π*(*Y*=1) = *α*\*1\*0.45 = *α*0.45

*P*(*Y*=0) = *αλ*(*Y*=0)*π*(*Y*=0) = *α*\*1\*0.68 = *α*0.68

Normalizing *P*(*Y*)

*P*(*Y*=1) = (*α*0.45) / (*α*0.45 + *α*0.68) = 0.4

*P*(*Y*=0) = (*α*0.68) / (*α*0.45 + *α*0.68) = 0.6

The initialized Bayesian network is shown in figure 3.1.4.



**Figure 3.1.4.** Initialized Bayesian network

When *X* becomes evidence and gains value 1, the function *update*(*X*, 1) is called:

*D* = *D* {*X*} =Ø {*X*} = {*X*}

Because *d* is instantiation of *D*, we have *d =* {*X*=1}

*λ*(*X*=1) = *π*(*X*=1) = *P*(*X*=1|*d*) = 1

*λ*(*X*=0) = *π*(*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:

*λX*(*Z*=1) = *λ*(*X*=1)*P*(*X*=1|*Z*=1) + *λ*(*X*=0)*P*(*X*=0|*Z*=1) = 1\*0.7 + 0\*0.3 = 0.7

*λ*(*Z*=1) = *λX*(*Z*=1)*λY*(*Z*=1) = 0.7\*1 = 0.7

*P*(*Z*=1|*d*) = *αλ*(*Z*=1)*π*(*Z*=1) = *α*0.7\*0.6 = *α*0.42

*λX*(*Z*=0) = *λ*(*X*=1)*P*(*X*=1|*Z*=0) + *λ*(*X*=0)*P*(*X*=0|*Z*=0) = 1\*0.2 + 0\*0.8 = 0.2

*λ*(*Z*=0) = *λX*(*Z*=0)*λY*(*Z*=0) = 0.2\*1 = 0.2

*P*(*Z*=0|*d*) = *αλ*(*Z*=0)*π*(*Z*=0) = *α*0.2\*0.4 = *α*0.08

Normalizing *P*(*Z*)

*P*(*Z*=1|*d*) = (*α*0.42) / (*α*0.42 + *α*0.08) = 0.84

*P*(*Z*=0|*d*) = (*α*0.08) / (*α*0.42 + *α*0.08) = 0.16

Calling *propagate*\_*down*\_*π*\_*message* (*Z*, *Y*)

Then, function *propagate*\_*down*\_*π*\_*message* (*Z*, *Y*) is executed:

*πY*(*Z*=1) = *π(Z*=1)*λY*(*Z*=1) = 1\*0.6=0.6

*πY*(*Z*=0) = *π*(*Z*=0)*λY*(*Z*=0) = 1\*0.4=0.4

*π*(*Y*=1) = *P*(*Y*=1|*Z*=1)*πX*(*Z*=1) + *P*(*Y*=1|*Z*=0)*πX*(*Z*=0) = 0.6\*0.6 + 0.3\*0.4 = 0.48

*π*(*Y*=0) = *P*(*Y*=0|*Z*=1)*πX*(*Z*=1) + *P*(*Y*=0|*Z*=0)*πX*(*Z*=0) = 0.3\*0.6 + 0.8\*0.4 = 0.50

*P*(*Y*=1) = *αλ*(*Y*=1)*π*(*Y*=1) = *α*\*1\*0.48 = *α*0.48

*P*(*Y*=0) = *αλ*(*Y*=0)*π*(*Y*=0) = *α*\*1\*0.5 = *α*0.50

Normalizing *P*(*Y*)

*P*(*Y*=1) = (*α*0.48) / (*α*0.48 + *α*0.50) = 0.49

*P*(*Y*=0) = (*α*0.50) / (*α*0.48 + *α*0.50) = 0.51

Then function *propagate*\_*down*\_*π*\_*message*(*X*, *T*) is executed

*πT*(*X*=1) = *π*(*X*=1) = 1

*πT*(*X*=0) = *π*(*X*=0) = 0

*π*(*T*=1) = *P*(*T*=1|*X*=1)*πT*(*X*=1) + *P*(*T*=1|*X*=0)*πT*(*X*=0) = 0.9\*1 + 0.4\*0 = 0.9

*π*(*T*=0) = *P*(*T*=0|*X*=1)*πT*(*X*=1) + *P*(*T*=0|*X*=0)*πT*(*X*=0) = 0.1\*1 + 0.6\*0= 0.1

*P*(*T*=1) = *αλ*(*T*=1)*π*(*T*=1) = *α*\*1\*0.9 = *α*0.9

*P*(*T*=0) = *αλ*(*T*=0)*π*(*T*=0) = *α*\*1\*0.1 = *α*0.1

Normalizing *P*(*T*)

*P*(*T*=1) = (*α*0.9) / (*α*0.9 + *α*0.1) = 0.9

*P*(*T*=0) = (*α*0.1) / (*α*0.9 + *α*0.1) = 0.1

Finally, all posterior probabilities are computed as in figure 3.1.5.



**Figure 3.1.5.** All posterior probabilities are computed after running Pearl algorithm (*X* is evidence)

## 3.2. DAG based inference

DAG based inference algorithms take advantages of straightforward structure of DAG without cycle. This approach starts with as simplest DAG which has many parents (input nodes) and one child (output node) following the *noisy OR-gate* model in which the output value becomes *true* (1) if there is at least one of inputs being *true* (1). Figure 3.2.1 is an example of noisy OR-gate network with cause-effect relationships.



**Figure 3.2.1.** Noisy OR-gate network with cause-effect relationships

Noisy OR-gate model can be used to build a simple BN when there are too many parent nodes or each parent node has too many discrete values. For example, if there are *n* binary parent nodes, given binary child node need a CPT having 2*n*+1 entries. In this case, noisy OR-gate algorithm establishes one equation to determine the CPT of child node, which is an interesting result.

Suppose every node is binary, noisy OR-gate inference in Bayesian network simulates electronic circuit based on three assumptions:

* *Cause inhibition*: Given a cause-effect relationship denoted by edge *X*→*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*.
* *Inhibition independence* (exception independence): Inhibitions are mutually independent. For example, inhibition *I*1 of *X*1 is independent from inhibition *I*2 of *X*2.
* *Noisy OR-gate condition* (accountability): Suppose we have a set of cause-effect relationships in which *Y* is the effect of many causes *X*1, *X*2,…, *Xn* (see figure 3.2.1). Let *Ii* be the inhibition of *Xi*. The effect *Y* cannot happen (*Y*=0) if at least one of *Xi* is equal 0 or one of inhibitions is *ON*:

Suppose we have *n* causes *X*1, *X*2,…, *Xn* and one result *Y*. According to “cause inhibition” and “inhibition independence” assumptions, let *Ii* be the inhibition of *Xi*. Let *Ai* be accountability variable so that *Ai* is *ON* (=1) if *Xi* is equal to 1 and *Ii* is *OFF* (=0).

*P*(*Ai* = *ON | Xi*=1, *Ii*=*OFF*) = 1

*P*(*Ai* = *ON | Xi*=1, *Ii*=*ON*) = 0

*P*(*Ai* = *ON | Xi*=0, *Ii*=*OFF*) = 0

*P*(*Ai* = *ON | Xi*=0, *Ii*=*ON*) = 0

*P*(*Ai* = *OFF | Xi*=1, *Ii*=*OFF*) = 0

*P*(*Ai* = *OFF | Xi*=1, *Ii*=*ON*) = 1

*P*(*Ai* = *OFF | Xi*=0, *Ii*=*OFF*) = 1

*P*(*Ai* = *OFF | Xi*=0, *Ii*=*ON*) = 1

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

*P*(*Y*=0*| Ai*=*ON*) = 0

*P*(*Y*=0| *Ai*=*OFF*) = 1

*P*(*Y*=1*|* *Ai*=*ON*) = 1

*P*(*Y*=1| *Ai*=*OFF*) = 0

Figure 3.2.2 shows the noisy OR-gate model of the network shown in figure 3.2.1



**Figure 3.2.2.** OR-gate model.

Now the strength of each cause-effect relationship *Xi→Y* is quantified by the CPT *P*(*Y|Xi*). Suppose causes (*X*1, *X*2,…, *Xi*,…, *Xn*) become evidences having values (*x*1, *x*2,…, *xi,…, xn*). Let *P*(*Xi*=1) = *pi* be the probability of *Xi* = 1. The probability of *Xi* ‘s inhibition is the inverse of *P*(*Xi*=1) according to equation 3.2.1.

|  |  |
| --- | --- |
| *P*(*Ii*=*ON*) = 1 – *P*(*Xi*=1) | (3.2.1) |

Let *O* be the set of such *i* that *Xi* = 1,

The goal of inference is to determine the posterior probability *P*(*Y*| *X*1, *X*2,…, *Xn*). We have:

(Due to total probability rule)

(Because *Ai* (s) are mutually independent)

(Because each *Ai* is only dependent on *Xi*)

(Due to *P*(*Y*=0*| Ai*=*ON*) = 0 and *P*(*Y*=0| *Ai*=*OFF*) = 1)

In general, we have equation 3.2.2 to specify noisy OR-gate inference.

|  |  |
| --- | --- |
|  | (3.2.2) |

Where *O* is the set of *i* such that *Xi* = 1. Given cause-effect relationship shown in figure 3.2.3. Given prior probabilities of causes *X*1=1, *X*2=1, *X*3=1 are 0.2, 0.5, 0.3, respectively. For example, we need to compute the conditional probability of effect *P*(*Y*=1*|X*1=1, *X*2=0, *X*3=1).



**Figure 3.2.3.** Noisy OR-gate inference example.

Applying equation 3.2.2, 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.

## 3.3. Optimal factoring based inference

Given a BN (*G*, *P*) (Neapolitan, 2003, p. 162) where *G* is the DAG shown in figure 3.3.1 and *P* is the joint probability *P*(*X*, *Y*, *Z*, *W*, *T*) = *P*(*T* | *Z*)*P*(*W* | *Y*, Z)*P*(*Y* | *X*)*P*(*Z* | *X*)*P*(*X*). Note, all nodes are binary variables.



**Figure 3.3.1.** A DAG used for illustrating optimal factoring based inference

Suppose *W* becomes evidence and we need to make an inference on *T* which is to compute the posterior probability *P*(*T* | *W*) according to equation 1.10 and 3.1 as follows (Neapolitan, 2003, p. 162):

We survey the numerator of the equation above as an example of optimal factoring based inference.

|  |  |
| --- | --- |
|  | (3.3.1) |

Because the sum is over 3 binary variables (*X*, *Y*, *Z*) and there are 4 multiplications in *P*(*T*, *W*), it requires 23 \* 4 = 32 multiplications to calculate one *P*(*T*, *W*). Because *T* and *W* has 4 possible values, it requires totally 32\*4 = 128 multiplications to calculate all values of *P*(*T*, *W*). The computation cost will be save if each product is not re-calculated when it is needed. For example, we factorize *P*(*T*, *W*) into 4 products as follows (Neapolitan, 2003, p. 163):

For illustration, suppose we create 4 buckets for such 4 products. Of course, such buckets are pseudo.

So, we have *bucket*1 = {*P*(*T* | *Z*)*P*(*W* | *Y*, Z)} for the first product, *bucket*2 = {*bucket*1\**P*(*Y*|*X*)} for the second product, *bucket*3 = {*bucket*2\**P*(*Z*|*X*)} for the third product, and *bucket*4 = {*bucket*3\**P*(*X*)} for the fourth product. After these products are calculated, they are stored in buckets. *Bucket*4 contains all possible values of *P*(*T*, *W*). Now we determine how many multiplications used for these buckets. The *bucket*1 as the first product *P*(*T* | *Z*)*P*(*W* | *Y*, Z) requires 24 = 16 multiplications (combinations) because it involves 4 binary variables. The *bucket*2 as the second product *bucket*1\**P*(*Y*|*X*) = *P*(*T* | *Z*)*P*(*W* | *Y*, Z)*P*(*Y* | *X*) requires 25 = 32 multiplications (combinations) because it involves 5 binary variables. The *bucket*3 as the third product *bucket*2\**P*(*Z*|*X*) = *P*(*T* | *Z*)*P*(*W* | *Y*, Z)*P*(*Y* | *X*)*P*(*Z* | *X*) requires 25 = 32 multiplications (combinations) because it involves 5 binary variables. The *bucket*4 as the fourth product *bucket*3\**P*(*X*) = *P*(*T* | *Z*)*P*(*W* | *Y*, Z)*P*(*Y* | *X*)*P*(*Z* | *X*)*P*(*X*) requires 25 = 32 multiplications (combinations) because it involves 5 binary variables. In general, *P*(*T*, *W*) requires |*bucket*1| + |*bucket*2| + |*bucket*3| + |*bucket*4| = 16 + 32 + 32 + 32 = 112 multiplications. We save 16 multiplications when *P*(*T*, *W*) needs 128 multiplications as usual.

We can save more multiplications by summing over a variable when such variable no longer appears in remaining terms as follows (Neapolitan, 2003, p. 163):

The *bucket*1 requires 24 = 16 multiplications because it involves 4 binary variables. The *bucket*2 requires 25 = 32 multiplications because it involves 5 binary variables. The *bucket*3 requires 24 = 16 multiplications because it only involves 4 binary variables when we sum *Y* out before taking *bucket*3. The *bucket*4 require 23 = 8 multiplications because it only involves 3 binary variables when we sum *Z* out before taking *bucket*4. In general, *P*(*T*, *W*) requires |*bucket*1| + |*bucket*2| + |*bucket*3| + |*bucket*4| = 16 + 32 + 16 + 8 = 72 multiplications.

The other factorization of *P*(*T*, *W*) is optimal as follows:

|  |  |
| --- | --- |
|  | (3.3.2) |

Now the *bucket*1 requires 22 = 4 multiplications because it involves 2 binary variables. The *bucket*2 requires 23 = 8 multiplications because it involves 3 binary variables. The *bucket*3 requires 23 = 8 multiplications because it only involves 3 binary variables when we sum *X* out before taking *bucket*3. The *bucket*4 require 23 = 8 multiplications because it only involves 3 binary variables when we sum *Y* out before taking *bucket*4. In general, *P*(*T*, *W*) requires |*bucket*1| + |*bucket*2| + |*bucket*3| + |*bucket*4| = 4 + 8 + 8 + 8 = 28 multiplications. Such the number of multiplications is now minimum for the aforementioned *P*(*W*, *T*). In general, we need to find out a way to factorize the product *P*(*T* | *W*) into a minimum number of multiplications as equation 3.3.2. This is the *Optimal Factoring Problem* given by Shachter, D’Ambrosio, and Del Favero (Shachter, D'Ambrosio, & Del Favero, 1990).

According to definition 3.1 in (Neapolitan, 2003, p. 163), a *factoring instance* *F* = {*V*, *S*, *Q*} is defined as a triple consisting of:

1. A set of *n* variables *V=* {*X*1, *X*2,…, *Xn*}
2. A set of *m* sub-sets *S =* {*S*{1}*, S*{2}*,…, S*{*m*}} where *S*{*i*} *V*
3. A *target set* *Q* *V*

According to definition 3.2 in (Neapolitan, 2003, p. 164), the factoring *α* of *S* is a binary tree satisfying three following properties (Neapolitan, 2003, p. 164):

* All and only members *S*{*i*} of *S* are leaves.
* The parent of nodes *SI* and *SJ* is denoted .
* The root of tree is *S*{1, 2,.., *m*}.

Given *F*, the cost of factoring *α* denoted *μα*(*F*) is three following steps (Neapolitan, 2003, p. 164):

1. All non-leave nodes are determined according to equation 3.3.3.

|  |  |
| --- | --- |
|  | (3.3.3) |

Note, the sign “\” denotes the subtraction (excluding) in set theory (Wikipedia, Set (mathematics), 2014).

1. The cost of each node is computed according to equations 3.3.4.

|  |  |
| --- | --- |
|  | (3.3.4) |

Where *|.|* denotes the cardinality of the set.

1. The cost of factoring *α* is *μα*(*F*) = *μα*(*S*{1,…, *m*})).

The less the cost *μα*(*F*) is, the better factoring *α* is. Hence, the optimal factoring problem is to find the optimal factoring *α* for the factoring instance *F* such that *µα*(*F*) is minimal.

When applying optimal factoring problem into Bayesian inference, the set of variables *V* in *F* corresponds with nodes in DAG, *S* corresponds with operands of the marginal probability, and the factoring *α* corresponds with the factorization of such probability. The cost of factoring instance *μα*(*F*) is equal to the number of multiplications. The problem becomes easy when we find out the best tree *α* having least *μα*(*F*) and compute the marginal probability with the same ordering of multiplications to this tree. According to definition 3.1 in (Neapolitan, 2003, p. 163), let the following factoring instance model the joint probability *P*(*T*, *W*) specified by equation 3.3.1 for the DAG shown in figure 3.3.1 as follows (Neapolitan, 2003, p. 164):

* Let *n* = 5 and *V* = {*X*, *Y*, *Z*, *W*, *T*}.
* Let *m* = 5 and *S*{1} = {*X*}, *S*{2} = {*X*, *Z*}, *S*{3} = {*X*, *Y*}, *S*{4} = {*Y*, *Z*, *W*}, and *S*{5} = {*Z*, *T*}.
* Let *Q* = {*W*, *T*}.

It is easy to recognize that *S*{1}, *S*{2}, *S*{3}, *S*{4}, and *S*{5} correspond with *P*(*X*), *P*(*Z* | *X*), *P*(*Y* | *X*), *P*(*W* | *Y*, Z), and *P*(*T* | *Z*), respectively. Suppose the optimal factorizing *α* shown in figure 3.3.2 (Neapolitan, 2003, p. 165) corresponds with the factorization of the marginal probability *P*(*W*, *T*) shown in equation 3.3.2 with note that Shachter, D’Ambrosio, and Del Favero (Shachter, D'Ambrosio, & Del Favero, 1990) proposed a linear time algorithm to find out such *α*.



**Figure 3.3.2.** An optimal factorizing

We will know the cost *μα*(*F*) of the factorizing *α* shown in figure 3.3.2 is 28 as aforementioned. In fact, we have (Neapolitan, 2003, p. 166):

The costs are computed as follows:

So, the cost of the factoring *α* is *μα*(*F*) = *μα*(*S*{1, 2, 3, 4, 5})) = 28.

Shortly, after giving the optimal factoring problem, Shachter, D’Ambrosio, and Del Favero (Shachter, D'Ambrosio, & Del Favero, 1990) proposed a linear time algorithm which solves the optimal factoring problem when the DAG is singly-connected. Because their algorithm combines both the symbolic reasoning and the numeric computation for doing probabilistic inference, it is called *Symbolic Probabilistic Inference* (*SPI*) algorithm.

# 4. Parameter learning

We turn back Bayesian inference introduced in equation 1.1 here. As a convention, uppercase letter such as *X*, *Y*, and *Z* often denote random variable whereas lowercase letters such as *x*, *y*, and *z* often denote instances or values of random variables. According to Bayesian approach, parameters such as mean *μ* and variance *σ*2 of normal distribution and probability *p* of binominal distribution are random variables. These random variables are commonly denoted Θ, which are hypotheses according to equation 1.1. The prior distribution (prior probability) is denoted *P*(Θ | *ξ*) where *ξ* denotes background knowledge about Θ. Note that *ξ* is often parameter of the prior distribution and so it can be called hyper-parameter of prior distribution. For example, if Θ follows beta distribution, its prior distribution is:

Where *ξ* = (*a*, *b*) are two parameters of such prior (beta) distribution. Note that Γ(.) is gamma function:

For another example, if Θ is mean *μ* and it follows normal distribution, its prior distribution is:

Where *ξ* = (*μ*0, *σ*02) are mean and variance of such prior (normal) distribution. Given sample *D* = {*X*1, *X*2,…, *Xm*) consisting *m* observations (evidences) *Xi* with note that *m* can be 1. Equation 4.1 specifies posterior distribution (posterior probability) of Θ, according to Bayes’ rule (Heckerman, 1995, p. 6) (Wikipedia, Bayesian inference, 2006).

|  |  |
| --- | --- |
|  | (4.1) |

Where *P*(*D* | Θ) is the likelihood function of Θ and *P*(*D*) is the marginal probability of sample. Equation 4.1 is an extension of equation 1.1. Note, the equation 4.1 is written fully as follows:

However *P*(*D* | Θ, *ξ*) = *P*(*D* | Θ) and *P*(*D* | *ξ*) = *P*(*D*) because *D* is only dependent on Θ. The marginal probability *P*(*D*) is expectation of the likelihood function *P*(*D* | Θ) given prior probability *P*(Θ | *ξ*).

Equation 4.1 which defines posterior probability of parameter Θ is used to assess hypothesis Θ after surveying sample *D*. This is a so-called *Bayesian inference*.

Suppose there is a requirement of predicting possibility of a new observation *Xm*+1 given previous sample *D* with note that *Xm*+1 is independent from *D*. In other words, we need to calculate the probability *P*(*Xm*+1 | *D*) called posterior predictive probability. Equation 4.2 specifies the *posterior predictive probability* (Heckerman, 1995, p. 6) (Wikipedia, Bayesian inference, 2006).

|  |  |
| --- | --- |
|  | (4.2) |

According to equation 4.2, the posterior predictive probability *P*(*Xm*+1 | *D*) is expectation of the probability *P*(*Xm*+1 | Θ) given posterior distribution *P*(Θ | *D*). Equation 4.2 establishes a so-called *Bayesian prediction*. The probability *P*(*Xm*+1 | Θ, *ξ*) is always determined because it is probability of observation.

Given a sample *D* = {*X*1, *X*2,…, *Xm*), there is a requirement of estimating parameter (hypothesis) Θ according to Bayesian inference. Let denote a Bayesian estimate of Θ. There are some methods to determine . The most popular method is Maximum A Posteriori (MAP) estimation (Wikipedia, Maximum a posteriori estimation, 2017). According to MAP, is a maximizer of posterior distribution given sample *D*.

Because *P*(*D*) is constant regarding Θ, we have:

For convenience, we take natural logarithm of *P*(*D*|Θ)*P*(Θ|*ξ*), which produce equation 4.3 to determine according to MAP.

|  |  |
| --- | --- |
|  | (4.3) |

Hence, MAP is extension of maximum likelihood estimation (MLE) method. If equation 4.3 is complicated, some approximate methods such as Newton-Raphson, gradient descent, and Lagrange duality to solve equation 4.3.

Other estimation methods use so-called loss functions. The squared-error loss function is defined according to equation 4.4 (Walpole, Myers, Myers, & Ye, 2012, p. 717):

|  |  |
| --- | --- |
|  | (4.4) |

The mean of posterior distribution *P*(Θ | *D*, *ξ*) is a Bayesian estimate of Θ under squared-error loss function, according to equation 4.5 (Walpole, Myers, Myers, & Ye, 2012, p. 717). In other words, such mean minimizes the squared-error loss function.

|  |  |
| --- | --- |
|  | (4.5) |

The absolute loss function is defined according to equation 4.6 (Walpole, Myers, Myers, & Ye, 2012, p. 718):

|  |  |
| --- | --- |
|  | (4.6) |

The median of posterior distribution *P*(Θ | *D*, *ξ*) is a Bayesian estimate of Θ under absolute loss function, according to equation 4.7 (Walpole, Myers, Myers, & Ye, 2012, p. 718). In other words, such median minimizes the absolute loss function.

|  |  |
| --- | --- |
|  | (4.7) |

Therefore, equations 4.3, 4.5 and 4.7 are popular equations for *Bayesian parameter estimation*. If the posterior distribution *P*(Θ | *D*, *ξ*) is symmetric, equations 4.5 and 4.7 produces the same estimate .

Now we survey a common case of Bayesian inference in which *D* is binominal sample. At that time which every *Xi* is binary random variable and the likelihood function *P*(*D* | Θ) becomes (Heckerman, 1995, p. 6):

Where *s* and *t* are the numbers of *Xi* = 1 and *Xi* = 0, respectively. The *s* and *t* are sufficient statistics of binomial sampling. Hence, Θ = *P*(*Xi* = 1) is the probability of *Xi* = 1. Equation 4.8 (Heckerman, 1995, p. 6), which is a special case of equation 4.1, specifies Bayesian inference (posterior probability of Θ) in case of binominal sampling.

|  |  |
| --- | --- |
|  | (4.8) |

Derived from equation 4.2, the posterior predictive probability *P*(*Xm*+1 = 1 | *D*) in case of binomial sampling becomes (Heckerman, 1995, p. 6):

Due to:

Equation 4.9, which is a variant of equation 4.2, specifies the posterior predictive probability *P*(*Xm*+1 = 1 | *D*) in case of binomial sampling (Heckerman, 1995, p. 6).

|  |  |
| --- | --- |
|  | (4.9) |

Where *E*(Θ | *D*, *ξ*) denotes the expectation of Θ given the posterior probability *P*(Θ | *D*, *ξ*) specified by equation 4.8. Comparing equation 4.9 and equation 4.5, we recognize that the posterior predictive probability is the estimate of Θ under squared-error loss function in case of binomial sampling.

Suppose Θ is distributed according to beta distribution, its prior probability is specified by equation 4.10.

|  |  |
| --- | --- |
|  | (4.10) |

Where,

So *ξ* = (*a*, *b*) are two parameters of such prior beta distribution. Note, we use two notations “beta” and “*β*” to denote beta distribution. Note, Γ(.) denotes gamma function (Neapolitan, 2003, p. 298) which is essentially an integral approximated to factorial function according to equation 4.11.

|  |  |
| --- | --- |
|  | (4.11) |

It is conventional that *e*(.) and *exp*(.) denote exponent function and *e*2.71828 is Euler’s number. If *x* is positive integer, gamma function in equation 4.11 is equivalent to factorial function,

There is an important property of gamma function which is expressed as follows (Neapolitan, 2003, p. 298):

Figure 4.1 shows beta density function with various parameters *a* and *b*. Beta functions *β*(*x*;2,2), *β*(*x*;4,2), and *β*(*x*;2,4) are drawn as black line, green line, and red line, respectively.



**Figure 4.1.** Beta density functions with various parameters *a* and *b*

In beta density function, there are “*a*” successful outcomes (for example, *x* =1) in “*a+b*” trials. The higher value of “*a*” is, the higher ratio of success is, so, the graph leans forward right. The higher value of “*a+b*” is, the more the mass concentrates around *a*/(*a+b*) and the narrower the graph is.

The theoretical mean and variance of beta distribution is specified by equation 4.12.

|  |  |
| --- | --- |
|  | (4.12) |

Where *N* = *a* + *b*. The marginal probability *P*(*D*) is calculated as follows:

Where *M* = *s* + *t*. Shortly, equation 4.13 specifies the marginal probability *P*(*D*).

|  |  |
| --- | --- |
|  | (4.13) |

The posterior probability of Θ is re-calculated as follows:

Therefore, in case of binomial sampling, if the prior probability of Θ conforms beta distribution beta(Θ | *a*, *b*) then, the posterior probability of Θ conforms beta distribution beta(Θ | *a* + *s*, *b* + *t*), which is a beautiful result according to equation 4.14 (Heckerman, 1995, p. 7).

|  |  |
| --- | --- |
|  | (4.14) |

Equation 4.14 is the special case of equation 4.8 in case of beta prior distribution. Equation 4.15 (Heckerman, 1995, p. 7) specifies the posterior predictive probability *P*(*Xm*+1 = 1 | *D*) in case of binomial sampling and prior beta distribution.

|  |  |
| --- | --- |
|  | (4.15) |

Essentially, equation 4.15 is the special case of equation 4.5 in case of binomial sampling and beta prior distribution, which is used to estimate Θ under squared-error loss function.

Basic concepts of Bayesian learning were introduced. Sub-sections 4.1 and 4.2 mentions how to learn CPTs of BN in which sub-section 4.1 focuses on parameter learning in complete data whereas sub-section 4.2 focuses on parameter learning in incomplete data. In sub-sections 4.1 and 4.2, nodes in BN are binary random variables and data sample is binomial sample. Recall that the report focuses on discrete BN.

## 4.1. Parameter learning in complete data

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

|  |  |
| --- | --- |
| *P*(*X*=1 | *F*) = *F*, where *F* has beta density function *β*(*F*; *a*, *b*) | (4.1.1) |

Please pay attention to equation 4.1.1, *P*(*X=*1 *| F*)= *F* implicating that *F* representsrelative frequency of *X* (Neapolitan, 2003, p. 301) because it is the key of learning CPT based on beta density function. Variables *X* and *F* constitute a simple network which is referred as augmented BN (Neapolitan, 2003, p. 324). So, *X* is referred as real variable (hypothesis) opposite to dummy variable *F*. When hypothesis variable *X* is attached by variable *F* then, variable *F*, the probability *P*(*X=*1|*F*)= *F*, and beta function *β*(*F* | *a*, *b*) share the same purpose and all of them represent CPT of *X*. Figure 4.1.1 shows the simplest augmented BN. Therefore, we use binomial sample to learn BN and variable *F* is the parameter Θ when Θ is considered as random variable in Bayesian approach, *F* = Θ.



**Figure 4.1.1.** The simple (binomial) augmented BN with only one hypothesis node *X*

The augmented BN is often denoted as a triple (*G*, *F*(*G*), *β*(*G*)) whereas the BN is denoted as a pair (*G*, *P*). The probability *P*(*X* = 1) which is parameter of BN is really the prior predictive probability and so we have a simple but effective equation 4.1.2 to compute *P*(*X* = 1) as follows:

|  |  |
| --- | --- |
|  | (4.1.2) |

Following is the proof of equation 4.1.2.

*Proof.*

Please refer to equation 4.12 to know how to calculate the mean of beta distribution. Please pay attention to equation 4.1.2, it is the most essential equation used in parameter learning. The equation 4.1.2 is corollary 6.1 in (Neapolitan, 2003, p. 302).

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 *uth* trial is denoted *X*(*u*) considered as evidence variable whose probability *P*(*X*(*u*)= 1 *| F*)= *F*. So, all *X*(*u*) are conditionally dependent on *F*. The probability of variable *X*, *P*(*X=*1) is learned by these evidences. Note that evidence *X*(*u*) is considered as random variable like *X*.

We denote the vector of all evidences as  *=* (*X*(1), *X*(2),…, *X*(*m*)) which is also called the sample of size *m*. Hence, is known as a sample or an evidence vector and we often implicate as a collection of evidences. Given this sample, *β*(*F*) is called prior density function, and *P*(*X*(*u*) = 1) = *a*/*N* (due to equation 4.1.2) is called prior probability of *X*(*u*). It is necessary to determine the posterior density function *β*(*F|*)and the posterior probability of *X*, namely *P*(*X|*)*. The nature of this process is the parameter learning* which aims to determine CPTs that are parameters of discrete BN with note that such CPTs essentially are posterior probabilities *P*(*X|*). Note, *P*(*X|*) can be referred as *P*(*X*(*m+*1) *|* ). Figure 4.1.2 depicts this sample  *=* (*X*(1), *X*(2),…, *X*(*m*)).



**Figure 4.1.2.** The binomial sample *=*(*X*(1), *X*(2),…, *X*(*m*)) of size *m*

We only survey the case of binomial sample. Thus, having binomial distribution is called binomial sample and the network in figure 4.1.1 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*. Note that *s* and *t* are often called counters or count numbers.

**Computing posterior density function and posterior probability**

Now, we need to compute the posterior density function *β*(*F|*) and the posterior probability *P*(*X=*1|). It is essential to determine the probability distribution of *X*. Fortunately, *β*(*F|*) and *P*(*X=*1|) are already determined by equations 4.14 and 4.15 when *F* = Θ and *P*(*X=*1|) = *P*(*Xn*+1*=*1|). For convenience, we replicate equations 4.14 and 4.15 as equations 4.1.3 and 4.1.4, respectively.

|  |  |
| --- | --- |
|  | (4.1.3) |
|  | (4.1.4) |

Equation 4.1.4 is theorem 6.4 in (Neapolitan, 2003, p. 309). In general, you should merely remember the equations 4.1.2 and 4.1.4 to calculate prior probability of *X* and posterior probability of *X*, respectively. Essentially, equations 4.15 or 4.1.4 is the special case of equation 4.5 in case of binomial sampling and beta prior distribution, which is used to estimate *F* under squared-error loss function.

**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. Note, a BN having more than one hypothesis variable is known as multi-node BN. See the networks and CPTs in following figure 4.1.3 (Neapolitan, 2003, p. 329):



**Figure 4.1.3.** BN (a) and complex augmented BN (b)

In figure 4.1.3, the BN (a) having no attached dummy variable is also called original BN or trust BN, from which augmented BN (b) is derived by the way: for every node (variable) *Xi*, we add dummy parent nodes to *Xi*, obeying two principles below:

1. If *Xi* has no parent (not conditionally dependent on any others, *Xi* is a root), we add only one dummy variable denoted *Fi*1having the probability density function *β*(*Fi*1; *ai*1, *bi*1) so as to *P*(*Xi=*1*|Fi*1) *= Fi*1.
2. If *Xi* has a set of *ki* parent nodes and each parent node is binary, we add a set of *qi=*2*ki* dummy variables {*Fi*1, *Fi*2,…, } which, in turn, correspond to instances of parentnodes of *Xi*,namely{*PAi*1, *PAi*2, *PAi*3,…,} where each *PAij* is an instance of a parent node of *Xi* with note that each binary parent node has two instances (0 and 1, for example). For convenience, each *PAij* is called a parent instance of *Xi* and we let *PAi=*{*PAi*1, *PAi*2, *PAi*3,…,} be the vector or collection of parent instances of *Xi*. We also let *Fi*={*Fi*1, *Fi*2,…, } be the respective vector or collection of dummy variables *Fi*1 (s) attached to *Xi*. It is conventional that each *Xi* has *qi* parent instances ; in other words, *qi* denotes the size of *PAi* and the size of *Fi*. For example, in figure 4.1.3, node *X*2 has one parent node *X*1, which causes that *X*2 has two parent instances represented by two dummy variables *F*21 and *F*22. Additionally, *F*21 (*F*22) and its beta density function specify conditional probabilities of *X*2 given *X*1 = 1 (*X*1 = 0) because parent node *X*1 is binary. We have equation 4.1.5 for connecting CPT of variable *Xi* with beta density function of dummy variable *Fi*.

|  |  |
| --- | --- |
|  | (4.1.5) |

Equation 4.1.5 is an extension of equation 4.1.1 in multi-node BN and equation 4.1.5 degenerates to equation 4.1.1 if *Xi* has no parent. Note that the beta density function of *Fij* is *β*(*Fij*; *aij*, *bij*) and of course, in figure 4.1.3, we have *a*11=1, *b*11=1, *a*21=1, *b*21=1, *a*22=1, *b*22=1.

The beta density function for each *Fij* is specified in equation 4.1.6 as follows:

|  |  |
| --- | --- |
|  | (4.1.6) |

Where *Nij* = *aij* + *bij*. Note that equations 4.10 and 4.1.6 have the same meaning for representing beta function except that equation 4.1.6 is used in multi-node BN. Variables *Fij* (s) attached to the same *Xi* have no parent and are mutually independent, so, it is very easy to compute the joint beta density function *β*(*Fi*1, *Fi*2,…, ) with regard to node *Xi* as follows:

|  |  |
| --- | --- |
|  | (4.1.7) |

Besides the local parameter independence expressed in equation 4.1.7, we have global parameter independence if reviewing all variables *Xi* (s) with note that all respective *Fij* (s) over entire augmented BN are mutually independent. Equation 4.1.8 expresses the global parameter independence of all *Fij* (s).

|  |  |
| --- | --- |
|  | (4.1.8) |

Concepts “local parameter independence” and “global parameter independence” are defined in (Neapolitan, 2003, p. 333).

All variables *Xi* and their dummy variables form the complex augmented BN representing the trust BN in figure 4.1.3. In the trust BN, the conditional probability of variable *Xi* with respect to its parent instance *PAij*, in other words, the *ijth* conditional distribution is the expected value of *Fij* as below:

|  |  |
| --- | --- |
|  | (4.1.9) |

The equation 4.1.9 is extension of equation 4.1.2 when variable *Xi* has parent and both equations express prior probability of variable *Xi*. Following is proof of equation 4.1.9.

(due to local parameter independence specified in equation 4.1.7 when *Fij* (s) are mutually independent)

The equation 4.1.9 is theorem 6.7 proved by the similar way in (Neapolitan, 2003, pp. 334-335) to which I referred. For illustrating equations 4.1.5 and 4.1.9, recall that variables *Fij* (s) and their beta density functions *β*(*Fij*) (s) specify conditional probabilities of *Xi* (s) as in figure 4.1.3, and so, the CPTs in figure 4.1.3 is interpreted in detailed as follows:

Note that inverted probabilities in CPTs such as *P*(*X*1=0), *P*(*X*2=0*|X*1=1) and *P*(*X*2=0*|X*1=0) are not mentioned because *Xi* (s) are binary variables and so, *P*(*X*1=0) = 1 – *P*(*X*1=1) = 1/2, *P*(*X*2=0*|X*1=1) = 1 – *P*(*X*2=1*|X*1=1) = 1/2 and *P*(*X*2=0*|X*1=0) = 1 – *P*(*X*2=1*|X*1=0) = 1/2.

Suppose we perform *m* trials of random process, the outcome of *uth* trial which is BN like figure 4.1.3 is represented as a random vector *X*(*u*) containing all evidence variables in network. Vector *X*(*u*) is also called the *uth* *evidence* (vector) of entire BN. Suppose *X*(*u*) has *n* components or partial evidences *Xi*(*u*) when BN has *n* nodes; in figure 4.1.3, *n* = 2. Note that evidence *Xi*(*u*) is considered as random variable like *Xi*.

It is easy to recognize that each component *Xi*(*u*) represents the *uth* evidence of node *Xi* in the BN. The *m* trials constitute the sample of size *m* which is the set of random vectors denoted as *=*{*X*(1), *X*(2),…, *X*(*m*)}. is also called evidence matrix,evidence sample,training data,orevidences, in brief. We only review the case of binomial sample; it means that is the binomial BN sample of size *m*. For example, this sample corresponding to the network in figure 4.1.3 is depicted by figure 4.1.4 as below (Neapolitan, 2003, p. 337):



**Figure 4.1.4.** Expanded binomial BN sample of size *m*

After *m* trials are performed, the augmented BN are updated and so, dummy variables’ density functions and hypothesis variables’ conditional probabilities are changed. We need to compute the posterior density function *β*(*Fij|*) of each dummy variable *Fij* and the posterior condition probability *P*(*Xi=*1*| PAij*,) of each variable *Xi*. Note that evidence vectors *X*(*u*) (s) are mutually independent given all *Fij* (s). It is easy to infer that given fixed *i*, all evidences *Xi*(*u*) corresponding to variable *Xi* are mutually independent. Based on binomial trials and mentioned mutual independence, equation 4.1.10 is used for calculating probability of evidences corresponding to variable *Xi* over *m* trials as follows:

|  |  |
| --- | --- |
|  | (4.1.10) |

Where,

* Number *qi* is the number of parent instances of *Xi*. In binary case, each *Xi*(*u*) ‘s parent node has two instances/values, namely, 0 and 1.
* Counter *sij*, respective to *Fij*, is the number of all evidences among *m* trials such that variable *Xi* = 1 and *PAij* = 1. Counter *tij*, respective to *Fij*, is the number of all evidences among *m* trials such that variable *Xi* = 1 and *PAij* = 0. Note that *sij* and *tij* are often called *counters* or count numbers.
* *PAi=*{*PAi*1, *PAi*2, *PAi*3,…,} is the vector of parent instances of *Xi* and *Fi* = {*Fi*1, *Fi*2,…, } is the respective vector of variables *Fi*1 (s) attached to *Xi*.

From equation 4.1.10, it is easy to compute conditional probability *P*(*|F*1, *F*2,…, *Fn*) of evidence sample given *n* vectors *Fi* (s) with assumption that BN has *n* variables *Xi* (s) as follows:

(because evidence vectors *X*(*u*) (s) are mutually independent)

(due to Bayes’ rule specified in equation 1.1)

(applying multiplication rule specified by equation 1.3 into the numerator)

(because *Xi*(*u*) (s) are mutually independent given *Fi* (s) and each *Xi* depends only on *PAi* and *Fi*)

In brief, we have equation 4.1.11 for calculating conditional probability *P*(*|F*1, *F*2,…, *Fn*) of evidence sample given *n* vectors *Fi* (s).

|  |  |
| --- | --- |
|  | (4.1.11) |

The equation 4.1.11 is lemma 6.8 proved by similar way in (Neapolitan, 2003, pp. 338-339) to which I referred. It is necessary to calculate the whole probability *P*() of evidence sample , we have:

(due evidence vectors *X*(*u*) (s) are independent)

(due to total probability rule in continuous case, please see equation 1.5)

(Because *Xi*(*u*) (s) are mutually independent given *Fi* (s) and each *Xi* depends only on *PAi* and *Fi*. Moreover, all *Fi* (s) are mutually independent)

In brief, we have equation 4.1.12 for determining the whole probability *P*() of evidence sample as product of expectations of binomial trials.

|  |  |
| --- | --- |
|  | (4.1.12) |

Equation 4.1.12 is theorem 6.11 in (Neapolitan, 2003, p. 343). There is the question “how to determine in equation 4.1.12” and so we have equation 4.1.13 for calculating this expectation by extending equation 4.13, as follows:

|  |  |
| --- | --- |
|  | (4.1.13) |

Where *Nij=aij+bij* and *Mij=sij+tij*. When both condition probability *P*(*|F*1, *F*2,…, *Fn*) and whole probability *P*() for evidences are determined, it is easy to update the posterior density function and posterior probability which are main subjects of learning parameters or CPT evolution.

**Updating posterior density function and posterior probability in multi-node BN**

Now, we need to compute the posterior density function *β*(*Fij|*) and the posterior probability *P*(*Xi=*1*|PAij*, ) for each variable *Xi* in BN. In fact, we have:

(due to Bayes’ rule specified in equation 1.1)

(Due to total probability rule in continuous case, specified by equation 1.5. Note that *Fi* = {*Fi*1, *Fi*2,…, })

(due to equation 4.1.11)

(applying equation 4.1.12 into denominator)

(applying definition of beta density function specified by equation 4.10 into numerator and applying equation 4.1.13 into denominator, note that *Nij* = *aij* + *bij* and *Mij* = *sij* + *tij*)

(due to definition of beta density function specified in equation 4.10)

In brief, we have equation 4.1.14 for calculating posterior beta density function *β*(*Fij|*).

|  |  |
| --- | --- |
|  | (4.1.14) |

Note that equation 4.1.14 is an extension of equation 4.1.3 in case of multi-node BN. Equation 4.1.14 is corollary 6.7 proved by similar way in (Neapolitan, 2003, p. 347) to which I referred. Applying equations 4.1.9 and 4.1.14, it is easy to specify the posterior probability *P*(*Xi=*1*|PAij*, ) of variable *Xi* given its parent instance *PAij* as follows:

|  |  |
| --- | --- |
|  | (4.1.15) |

Where *Nij=aij+bij* and *Mij=sij+tij*. It is easy to recognize that equation 4.1.15 is an extension of equation 4.1.4 in case of multi-node BN. Hence, *Fij* is estimated by equation 4.1.15 under squared-error loss function with binomial sampling and prior beta distribution. In general, in case of binomial distribution, if we have the real/trust BN embedded in the expanded augmented network like figure 4.1.3 and each dummy node *Fij* has a prior beta distribution *β*(*Fij*; *aij*, *bij*) and each hypothesis node *Xi* has the prior conditional probability *P*(*Xi=*1*|PAij*) = *E*(*Fij*) = , the parameter learning process based on a set of evidences is to update the posterior density function *β*(*Fij|*) and the posterior conditional probability *P*(*Xi=*1*|PAij*,). Indeed, we have *β*(*Fij|*) = *beta*(*Fij*; *aij+sij*, *bij+tij*) and *P*(*Xi=*1*|PAij*,) = *E*(*Fij|*) = .

**Example of parameter learning based on beta density function**

Suppose we have the set of 5 evidences *=*{*X*(1)*, X*(2)*, X*(3)*, X*(4)*, X*(5)} owing to network in figure 4.1.3. Evidence sample (evidence matrix) is shown in table 4.1.1 (Neapolitan, 2003, p. 358).

|  |  |  |
| --- | --- | --- |
|  | *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 4.1.1.** Evidence sample corresponding to 5 trials (sample of size 5)

In order to interpret evidence sample in table 4.1.1, for instance, the first evidence (vector) implies that variable *X*2=1 given *X*1=1 occurs in the first trial. We need to compute all posterior density functions *β*(*F*11|), *β*(*F*21|), *β*(*F*22|) and all posterior conditional probabilities *P*(*X*1=1*|*), *P*(*X*2=1*|X*1=1,), *P*(*X*2=1*|X*1=0,) from prior density functions *β*(*F*11; 1,1), *β*(*F*21; 1,1), *β*(*F*22; 1,1). As usual, letcounter *sij* (*tij*) be the number of evidences among 5 trials such that variable *Xi* = 1 and *PAij* = 1 (*PAij* = 0), the following table 4.1.2 shows counters *sij*, *tij* (s) and posterior density functions calculated based on these counters; please see equation 4.1.14 for more details about updating posterior density functions. For instance, the number of rows (evidences) in table 4.1.1 such that *X*2=1 given *X*1=1 is 3, which causes *s*21 = 3 in table 4.1.2.

|  |  |
| --- | --- |
| *s*11=1+1+1+1+0=4 | *t*11=0+0+0+0+1=1 |
| *s*21=1+1+1+0+0=3 | *t*21=0+0+0+0+1=1 |
| *s*22=0+0+0+0+0=0 | *t*21=0+0+0+0+1=1 |
| *β*(*F*11|) = *β*(*F*11; *a*11+*s*11, *b*11+*t*11)*= β*(*F*11; 1+4, 1+1)*= β*(*F*11; 5, 2)  *β*(*F*21|) = *β*(*F*21; *a*21+*s*21, *b*21+*t*21)*= β*(*F*21; 1+3, 1+1)*= β*(*F*11; 4, 2)  *β*(*F*22|) = *β*(*F*22; *a*22+*s*22, *b*22+*t*22)*= β*(*F*22; 1+0, 1+1)*= β*(*F*11; 1, 2) | |

**Table 4.1.2.** Posterior density functions calculated based on count numbers *sij* and *tij*

When posterior density functions are determined, it is easy to compute posterior conditional probabilities *P*(*X*1=1*|*), *P*(*X*2=1|*X*1=1,),and *P*(*X*2=1|*X*1=0,) as conditional expectations of *F*11, *F*21, and *F*22, respectively according to equation 4.1.15. Table 4.1.3 expresses such posterior conditional probabilities as evolutional CPTs of *X*1 and *X*2.

|  |
| --- |
|  |

**Table 4.1.3.** Updated CPTs of *X*1 and *X*2

Note that inverted probabilities in CPTs such as *P*(*X*1=0*|*), *P*(*X*2=0*|X*1=1,) and *P*(*X*2=0*|X*1=0,) are not mentioned because *Xi* (s) are binary variables and so, *P*(*X*1=0*|*) = 1 – *P*(*X*1=1*|*) = 2/7, *P*(*X*2=0*|X*1=1,) = 1 – *P*(*X*2=1*|X*1=1,) = 1/3 and *P*(*X*2=0*|X*1=0,) = 1 – *P*(*X*2=1*|X*1=0,) = 2/3.

Now BN in figure 4.1.3 is updated based on evidence sample and it is converted into the evolved BN with full of CPTs shown in figure 4.1.5 as follows:



**Figure 4.1.5.** Updated version of BN (a) and augmented BN (b)

It is easy to perform parameter learning by counting numbers *sij* and *tij* among sample according to expectation of beta density function as in equation 4.1.4 and 4.1.15 but a problem occurs when data in sample is missing. This problem is solved by expectation maximization (EM) algorithm mentioned in next sub-section 4.2.

## 4.2. Parameter learning in incomplete data

In practice there are some evidences in such as *X*(*u*) (s) which lack information and thus, it stimulates the question “How to update network from missing data”. We must address this problem by artificial intelligence techniques, namely, Expectation Maximization (EM) algorithm – a famous technique solving estimation of missing data. EM algorithm has two steps such as Expectation step (E-step) and Maximization step (M-step), which aims to improve parameters after a number of iterations; please read (Borman, 2004) for more details about EM algorithm. We will know thoroughly these steps by reviewing above example shown in table 4.1.1, in which there is the set of 5 evidences *=*{*X*(1)*, X*(2)*, X*(3)*, X*(4)*, X*(5)} along with network in figure 4.1.3 but the evidences *X*(2) and *X*(5) have not data yet. Table 4.2.1 shows such missing data (Neapolitan, 2003, p. 359).

|  |  |  |
| --- | --- | --- |
|  | *X*1 | *X*2 |
| ***X*(1)** | *X*1(1) = 1 | *X*2(1) = 1 |
| ***X*(2)** | *X*1(2) = 1 | *X*2(2) =***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) =***v*2?** |

**Table 4.2.1.** Evidence sample with missing data

As known,count numbers *s*21, *t*21and *s*22, *t*22can’t be computed directly, it means that it is not able to compute directly the posterior density functions *β*(*F*11|), *β*(*F*21|), and *β*(*F*22|). It is necessary to determine missing values *v*1 and *v*2. Because *v*1 and *v*2 are binary values (1 and 0), we calculate their occurrences. So, evidence *X*(2) is split into two *X*‘(2*)* (s) corresponding to two values 1 and 0 of *v*1. Similarly, evidence *X*(5) is split into two *X*‘(5*)* (s) corresponding to two values 1 and 0 of *v*2. Table 4.2.2 shows new split evidences for missing data.

|  |  |  |  |
| --- | --- | --- | --- |
|  | *X*1 | *X*2 | #Occurrences |
| ***X*(1)** | *X*1(1) = 1 | *X*2(1) = 1 | 1 |
| ***X*‘(2)** | *X*1’(2) = 1 | *X*2’(2) = 1 | #*n*11 |
| ***X*‘(2)** | *X*1’(2) = 1 | *X*2’(2) = 0 | #*n*10 |
| ***X*(3)** | *X*1(3) = 1 | *X*2(3) = 1 | 1 |
| ***X*(4)** | *X*1(4) = 1 | *X*2(4) = 0 | 1 |
| ***X*‘(5)** | *X*1’(5) = 0 | *X*2’(5) = 1 | #*n*21 |
| ***X*‘(5)** | *X*1’(5) = 0 | *X*2’(5) = 0 | #*n*20 |

**Table 4.2.2.** New split evidences for missing data

The number #*n*11 (#*n*10) of occurrences of *v*1=1(*v*1=0)is estimated by the probability of *X*2 = 1 given *X*1 = 1 (*X*2 = 0 given *X*1 = 1) with assumption that *a*21 = 1 and *b*21 = 1 as in figure 4.1.3.

Similarly, the number #*n*21 (#*n*20) of occurrences of *v*2=1(*v*2=0)is estimated by the probability of *X*2 = 1 given *X*1 = 0 (*X*2 = 0 given *X*1 = 0) with assumption that *a*22 = 1 and *b*22 = 1 as in figure 4.1.3.

When #*n*11, #*n*10, #*n*21, and #*n*20 are determined, missing data is filled fully and evidence sample is completed as in table 4.2.3.

|  |  |  |  |
| --- | --- | --- | --- |
|  | *X*1 | *X*2 | #Occurrences |
| ***X*(1)** | *X*1(1) = 1 | *X*2(1) = 1 | 1 |
| ***X*‘(2)** | *X*1’(2) = 1 | *X*2’(2) = 1 | 1/2 |
| ***X*‘(2)** | *X*1’(2) = 1 | *X*2’(2) = 0 | 1/2 |
| ***X*(3)** | *X*1(3) = 1 | *X*2(3) = 1 | 1 |
| ***X*(4)** | *X*1(4) = 1 | *X*2(4) = 0 | 1 |
| ***X*‘(5)** | *X*1’(5) = 0 | *X*2’(5) = 1 | 1/2 |
| ***X*‘(5)** | *X*1’(5) = 0 | *X*2’(5) = 0 | 1/2 |

**Table 4.2.3.** Complete evidence sample in E-step of EM algorithm

In general, the essence of this task – estimating missing values by *expectations* of *F*21 and *F*22 based on previous parameters *a*21, *b*21, *a*22, and *b*22 of beta density functions is E-step in EM algorithm. Of course, in E-step, when missing values are estimated, it is easy to determine counters *s*11, *t*11, *s*21, *t*21, *s*22, and *t*22. Recall that counters *s*11 and *t*11 are numbers of evidences such that *X*1 = 1 and *X*1 = 0, respectively. Counters *s*21 and *t*21 (*s*22 and *t*22) are numbers of evidences such that *X*2 = 1 and *X*2 = 0 given *X*1 = 1 (*X*2 = 1 and *X*2 = 0 given *X*1 = 0), respectively. In fact, these counters are ultimate results of E-step. From complete sample in table 4.2.3, we have table 4.2.4 showing such ultimate results of E-step:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Table 4.2.4.** Counters *s*11, *t*11, *s*21, *t*21, *s*22, and *t*22 from estimated values (of missing values)

The next step of EM algorithm, M-step is responsible for updating posterior density functions *β*(*F*11|), *β*(*F*21|), and *β*(*F*22|), which leads to update posterior probabilities *P*(*X*1=1*|*), *P*(*X*2=1*|X*1=1,),and *P*(*X*2=1*|X*1=0,), based on current counters *s*11, *t*11, *s*21, *t*21, *s*22, and *t*22 from complete evidence sample (table 4.2.3). Table 4.2.5 shows results of M-step which are posterior density functions *β*(*F*11|), *β*(*F*21|), and *β*(*F*22|) along with posterior probabilities (updated CPT) such as *P*(*X*1=1*|*), *P*(*X*2=1*|X*1=1,),and *P*(*X*2=1*|X*1=0,).

|  |
| --- |
|  |

**Table 4.2.5.** Posterior density functions and posterior probabilities are updated in M-step of EM algorithm

Note that origin parameters such as *a*11=1, *b*11=1, *a*21=1, *b*21=1, *a*22=1, and *b*22=1 (see figure 4.1.3) are kept intact in the task of updating posterior density functions *β*(*F*11|), *β*(*F*21|), and *β*(*F*22|). For example, *β*(*F*11|) = *β*(*F*11; *a*11+*s*11,*b*11+*t*11) = *β*(*F*11; 1+4,1+1) = *β*(*F*11; 5,2). After the updating task, these parameters are changed into new values; concretely, *a*11=5, *b*11=2, *a*21=7/2, *b*21=5/2, *a*22=3/2, and *b*22=3/2. These parameters updated with new values, which are called posterior parameters, are in turn used for the new iteration of EM algorithm.

The process of such two steps (E-step and M-step) repeated more and more brings out the EM algorithm. In general, EM algorithm is the iterative algorithm having many iterations and each iteration has two steps: E-step and M-step. Given the *kth* iteration in EM algorithm whose two steps such as E-step and M-step are summarized as follows:

1. *E-step*. Missing values are estimated based on expectations of *Fij* with regard to previous ((*k–*1)*th*) parameters *aij* and *bij*. Current (*kth*) counters *sij* and *tij* are calculated with estimated values (of such missing values). Table 4.2.4 shows such current counters which are ultimate results of E-step.
2. *M-step*. Posterior density functions andposterior probabilities (CPT) are updated based on current (*kth*) counters *sij* and *tij*. Of course, *aij* and *bij* are updated because they are parameters of (beta) density functions. Table 4.2.5 shows results of M-step. Terminating algorithm if stop condition becomes true, otherwise, reiterating step 1. The stop condition may be “posterior density functions andposterior probabilities are not changed significantly”, “the number of iterations approaches *k times*”or “there is no missing value”.

After *kth* iteration, the limit

will approach a certain limit. Note, the upper script (*k*) denotes the *kth* iteration. Don’t worry about the case of infinite iterations, we will obtain optimal probability *P*(*Xi=*1*|PAij*,) = if *k* is large enough. This limit is noted similarly as equation 6.17 in (Neapolitan, 2003, p. 361). EM algorithm for learning parameters in BN is also mentioned particularly in (Neapolitan, 2003, pp. 359-363).

Go backing the example of missing data, the results of EM algorithm at the first iteration are summarized from table 4.2.5, as follows:

When compared with the origin probabilities

There is significant change in these probabilities if the maximum deviation is pre-defined 0.05. It is easy for us to verify this assertion, concretely, |0.71 – 0.5| = 0.21 > 0.05. So it is necessary to run the EM algorithm at the second iteration.

At the second iteration, the E-step starts calculating the number #*n*11 (#*n*10) of occurrences of *v*1=1(*v*1=0)andthe number #*n*21 (#*n*20) of occurrences of *v*2=1(*v*2=0) again:

When #*n*11, #*n*10, #*n*21, and #*n*20 are determined, missing data is filled fully and evidence sample is completed as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | *X*1 | *X*2 | #Occurrences |
| ***X*(1)** | *X*1(1) = 1 | *X*2(1) = 1 | 1 |
| ***X*‘(2)** | *X*1’(2) = 1 | *X*2’(2) = 1 | 7/12 |
| ***X*‘(2)** | *X*1’(2) = 1 | *X*2’(2) = 0 | 5/12 |
| ***X*(3)** | *X*1(3) = 1 | *X*2(3) = 1 | 1 |
| ***X*(4)** | *X*1(4) = 1 | *X*2(4) = 0 | 1 |
| ***X*‘(5)** | *X*1’(5) = 0 | *X*2’(5) = 1 | 1/2 |
| ***X*‘(5)** | *X*1’(5) = 0 | *X*2’(5) = 0 | 1/2 |

Recall that counters *s*11 and *t*11 are numbers of evidences such that *X*1 = 1 and *X*1 = 0, respectively. Counters *s*21 and *t*21 (*s*22 and *t*22) are numbers of evidences such that *X*2 = 1 and *X*2 = 0 given *X*1 = 1 (*X*2 = 1 and *X*2 = 0 given *X*1 = 0), respectively. These counters which are ultimate results of E-step are calculated as follows:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Posterior density functions *β*(*F*11|), *β*(*F*21|), and *β*(*F*22|), posterior probabilities *P*(*X*1=1*|*), *P*(*X*2=1*|X*1=1,),and *P*(*X*2=1*|X*1=0,) are updated at M-step as follows:

When compared with the previous probabilities

There is no significant change in these probabilities if the maximum deviation is pre-defined 0.05. It is easy for us to verify this assertion, concretely, |0.75 – 0.71| = 0.04 < 0.05, |0.61 – 0.58| = 0.03 < 0.05, and |0.5 – 0.5| = 0 < 0.05. So the EM algorithm is stopped with note that we can execute more iterations for EM algorithm in order to receive more precise results that posterior probabilities are stable . Consequently, the Bayesian network in figure 4.1.3 is converted into the evolutional version specified in figure 4.2.1.



**Figure 4.2.1.** Updated version of BN (a) and augmented BN (b) in case of missing data

In general, parameter learning is described thoroughly in this section. The next section mentions structure learning.

# 5. Structure learning

As discussed in section 2, DAGs which have the same set of nodes are Markov equivalent if and only if they have same d-separations. From lemma 2.6 and theorem 2.4 in (Neapolitan, 2003, pp. 86-87), Neapolitan (Neapolitan, 2003, p. 91) stated that Markov equivalent class can be represented with a graph that has the same links and the same uncoupled head-to-head meeting as the DAGs in the class. Markov equivalence divides all DAGs into disjoint Markov equivalent classes. According to (Neapolitan, 2003, p. 91), a DAG pattern is defined for a Markov equivalence class to be the graph that has the same links as the DAGs in the equivalence class and has oriented all and only the edges common to all DAGs in the equivalent class. Let the pattern *gp* be a DAG pattern. Let *GP* be random variable whose values are patterns *gp*. The basic idea of structure learning approaches is to find out the pattern *gp* that satisfies some condition best. Instead of searching many individual DAG according to given condition, there are two main learning approaches:

* *Score-based approach* (Neapolitan, 2003, pp. 441-476): For each pattern *gp GP*, the *gp* which gains the maximal scoring criterion *score*(*D*, *gp*) given training data set *D* is the best one. Because the essence of score-based approach is select the most likely structure, it is also called *model selection* approach (Neapolitan, 2003, p. 445).
* *Constraint-based approach* (Neapolitan, 2003, pp. 541-603): Given a set of conditional independences in the joint probability, the best *gp* is the one for which Markov condition entails all and only these conditional independences. Such independences play the role of the “door latch” for learning algorithm. In other words, we try to find out the DAG that satisfies faithfulness condition (Neapolitan, 2003, p. 541).

## 5.1. Score-based approach

Given a set of random variables (nodes) *V* = {*X*1, *X*2,…, *Xn*}, let (*G*, *P*) be possible Bayesian network where *P* is joint conditional probability density and *G =* (*V*, *E*) is a DAG. Let (*G*, *F(G)*,*β*(*G*)) be the augmented BN with equivalent sample size *N* where *F*(*G*) is augmented variables attached to every nodes in *V* and *β*(*G*) represents beta distributions for augmented variables (see section 4). Pattern *gp* also represents Markov equivalent augmented BN. Given multinomial sampling with note that each *Xi* is multinomial variable in general, scored-based approach has three following steps (Neapolitan, 2003, p. 445):

1. Suppose all augmented BNs has the same equivalent sample size *N*.
2. Let *ri* be the number of possible values of variable *Xi*. Note, in simpler case that *Xi* is binary then, *ri* = 2. Let *qi* be the number of distinct instantiations of parents of *Xi*. For example, if *Xi* and its parents are binary and *Xi* have 1 parents then *qi* = 2. All augmented variables *Fij* representing the conditional probability of *Xi* given instantiation *j* of its parent are assigned to uniform distribution according to equivalent sample size *N*:

|  |  |
| --- | --- |
|  | (5.1.1) |

1. Given *D=*{*X*(1)*, X*(2)*,…, X*(*M*)} is the training data set size *M*, where *X*(*u*) is a trial. Note that *X*(*u*)*=*(*X*1(*u*), *X*2(*u*),…, *Xn*(*u*)) is a *n*-dimension vector which is a outcome (instantiation) of *n* variables *Xi*, *X*2,…, *Xn*. Each *Xi*(*u*) has the same space to *Xi*. Each DAG pattern *gp* is assigned a so-called scoring criterion *score*(*D*, *gp*). This score is the likelihood function of *gp* given training data set *D* (Neapolitan, 2003, p. 449).

|  |  |
| --- | --- |
|  | (5.1.2) |

Where Γ(.) is gamma function specified by equation 4.11.

1. Which *gp* gaining maximal *score*(*D*, *gp*) is selected as the final DAG *G*. The final joint probability *P* is determined by values *Mij* and *sijk* based on Markov condition; please see section 4 for more details about parameter learning. As a result, the learned BN (*G*, *P*) is determined. The score-based approach is also called selection model approach.

Following is explanation of equation 5.1.2. The posterior probability of *gp* given data sample *D* is:

Where *P*(*gp*) is the prior probability and *P*(*D*|*gp*) is the likelihood function.

The best is the one that maximizes such posterior probability *P*(*gp* | *D*).

Because the marginal probability *P*(*D*) is constant with regard to *gp*, the best is now a maximizer of *P*(*D* | *gp*)*P*(*gp*).

In practice, the prior probability *P*(*gp*) is initialized subjectively or all DAG patterns are assigned the same prior probability; so *P*(*gp*) it can be ignored. Hence, the best can be a maximizer of the likelihood function *P*(*D* | *gp*).

The scoring criterion *score*(*D*, gp) is defined as such likelihood function *P*(*D* | *gp*).

**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:

|  |  |
| --- | --- |
| *X*1 | *X*2 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |
| 1 | 1 |
| 0 | 1 |
| 0 | 0 |

Let *gp*1 be the DAG pattern in which *X*1 is parent of *X*2; otherwise let *gp*2 be the DAG pattern in which *X*1 and *X*2 are mutually independent. Given the equivalent sample size is *N* = 4, figure 5.1.1 shows such *gp*1 and *gp*2.



**Figure 5.1.1.** Augmented Bayesian networks of *gp*1 in (a) and *gp*2 in (b)

We have:





Because the *score*(*D*, *gp*1) is larger than the *score*(*D*, *gp*2), the DAG pattern *gp*1 is chosen as the DAG *G* of Bayesian network appropriate to training data set.

In above example we recognize that it is difficult to determine all DAGs. So the score-based approach becomes ineffective in case of many variables. The number of DAGs which is surveyed to compute scoring criterion gets huge. It is impossible to do brute-force searching over DAG space. There are some heuristic algorithms to reduce whole DAG space to smaller space called candidate set of DAGs obeying some restriction, for example, the prior ordering of variables. Such heuristic algorithms are classified into *approximate learning* or approximate selection (Neapolitan, 2003, pp. 511-538). The global score can defined as a product of local scores:



Where *score*(*D,Xi,PAi*) is the local score of *Xi* given its parents *PAi* (Neapolitan, 2003, p. 512).

 (**4.2**)

Let be the number of distinct instantiations of parents of *Xi*.

A well-known heuristic algorithm belonging to approximate learning is K2 algorithm. The K2 algorithm tries to find out the pattern DAG *gp* whose each variable *Xi* maximizes local score *score*(*D,Xi,PAi*) instead of discovering all DAGs. It means that K2 algorithm finds out optimal parents *PAi* of each *Xi*. Note that it expects that the global score will be approached by maximizing each partial local score. K2 algorithm has following steps:

1. Suppose there is an ordering (*X1, X2,…, Xn*). There is no backward edge, for example, the edge *Xi←Xj* (if exist) where *i < j* is invalid. Let *Pre*(*Xi*) be the set of previous nodes of *Xi* in ordering. Let *PAi* is parents of *Xi*. K2’s mission is to find out *PAi* for every *Xi*. Firstly, each *PAi* (s) is set to be empty and each local *score*(*D,Xi,PAi*) is initialized with such empty *PA*i.
2. Each *Xi* is visited according to the ordering. When *Xi* is visited, which node in *Pre*(*Xi*) that maximizes the local *score*(*D,Xi,PAi*) is added to *PAi*.
3. Algorithm terminates when no node is added to *PAi*.

## 5.2. Constraint-based approach

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

**Example 4.2**: Suppose we have *V* = {*X, Y, Z*} and *INDP* = {*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:

1. Firstly, the structure of DAG is drafted as “skeleton”. If there is no conditional independence relating to *Xi* and *Xj* then the link between them is created. So skeleton is the undirected graph which contains variables (nodes) and links.
2. 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→Z←Y*
* *Rule 2*: If the uncoupled chain *X→Z–Y* exists (having an edge *X→Z*) then this chain is assumed serial path: *X→Z→Y*.
* *Rule 3*: If the edge *X→Y* caused a directed cycle at a position in network then it is reversed: *X←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 *INDP* = {*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→Z←Y*

Applying *rule 2*: Because the uncoupled chain *X→Z–T* exists, we have the assumed serial path: *X→Z→T.*

|  |  |
| --- | --- |
|  | (9.99) |

# 6. Applications

# 7. Conclusions

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.

# References

Borman, S. (2004). *The Expectation Maximization Algorithm - A short tutorial.* University of Notre Dame, Department of Electrical Engineering. South Bend, Indiana: Sean Borman's Home Page.

Heckerman, D. (1995). *A Tutorial on Learning With Bayesian Networks.* Microsoft Corporation, Microsoft Research. Redmond: Microsoft Research. Retrieved from ftp://ftp.research.microsoft.com/pub/dtg/david/tutorial.ps

Montgomery, D. C., & Runger, G. C. (2003). *Applied Statistics and Probability for Engineers* (3rd Edition ed.). New York, NY, USA: John Wiley & Sons, Inc.

Murphy, K. P. (1998). *A Brief Introduction to Graphical Models and Bayesian Networks.* Retrieved 2008, from Kevin P. Murphy's home page: http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html

Neapolitan, R. E. (2003). *Learning Bayesian Networks.* Upper Saddle River, New Jersey, USA: Prentice Hall.

Rosen, K. H. (2012). *Discrete Mathematics and Its Applications* (7nd Edition ed.). (M. Lange, Ed.) McGraw-Hill Companies.

Walpole, R. E., Myers, R. H., Myers, S. L., & Ye, K. (2012). *Probability & Statistics for Engineers & Scientists* (9th ed.). (D. Lynch, Ed.) Boston, Massachusetts, USA: Pearson Education, Inc.

Wikipedia. (2006). *Bayesian inference*. (Wikimedia Foundation) Retrieved 2007, from Wikipedia website: http://en.wikipedia.org/wiki/Bayesian\_inference

Wikipedia. (2014, October 29). *Sample (statistics)*. (Wikimedia Foundation) Retrieved October 31, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Sample\_(statistics)

Wikipedia. (2014, October 10). *Set (mathematics)*. (A. Rubin, Editor, & Wikimedia Foundation) Retrieved October 11, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Set\_(mathematics)