**Evolution of Parameters in Bayesian Overlay Model**

Loc Nguyen

University of Science, Ho Chi Minh, Vietnam

Email: ng\_phloc@yahoo.com

Phung Do

University of Information Technology, Ho Chi Minh, Vietnam

Email: dtminhphung@yahoo.com

## Abstract

Adaptive learning systems require well-organized user model along with solid inference mechanism. Overlay modeling is the method in which the domain is decomposed into a set of elements and the user model is simply a set of masteries over those elements. The combination between overlay model and Bayesian network (BN) will make use of the flexibility and simplification of overlay modeling and the power inference of BN. Thus it is compulsory to pre-define parameters, namely, Conditional Probability Tables (CPT (s)) in BN but no one ensured absolutely the correctness of these CPT (s). This research focuses on how to enhance parameters’ quality in Bayesian overlay model, in other words, this is the evolution of CPT (s).

**Keywords**: adaptive learning, user modeling, user model, learner model, overlay model, Bayesian network, parameter learning.

## 1. Introduction

Adaptive learning systems require well-organized user model along with solid inference mechanism. Overlay modeling is the method in which the domain is decomposed into a set of elements and the user model is simply a set of masteries over those elements. The combination between overlay model and Bayesian network (BN) will make use of the flexibility and simplification of overlay modeling and the power inference of BN. This combination is described and evaluated in (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009). Thus it is compulsory to pre-define parameters, namely, Conditional Probability Tables (CPT (s)) in BN but no one ensured absolutely the correctness of these CPT (s). This research concerns about how to enhance parameters’ quality in Bayesian overlay model, in other words, this is the evolution of CPT (s).

As known, user model is the core of most adaptive learning systems. There are some effective modeling methods such as stereotype, overlay, plan recognition but overlay model is proven soundness due to two its properties: flexible graphic structure and reflecting comprehensibly the domain knowledge in education course. The basic ideology of overlay model is to represent user knowledge as subset of domain model. The combination between overlay model and BN (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) will make use of each method’s strong points and restrain drawbacks.

* The structure of overlay model is translated into BN, each user knowledge element becomes an node in BN.
* Aggregation relationships between domain elements in overlay model become conditional dependence assertions signified by CPT (s) of nodes in BN.
* Domain elements are defined as hidden nodes and other learning objects which are used to assess user’s performance are considered as evidence nodes in BN.

Such model is called Bayesian overlay model, Bayesian model, or Bayesian network in brief with note that knowledge sub-model inside Triangular Learner Model (TLM) is represented as such Bayesian model (Nguyen, ZEBRA: A new User Modeling System for Triangular Model of Learners' Characteristics, 2009). In process of parameter specification by weighting arcs, the gained CPT (s) are confident but it is necessary to improve them after inference tasks from collected evidences. This trend relates to learning parameters, that’s to say, the evolution of CPT (s) with note that CPT (s) are parameters in BN. Section [2](#_2._Bayesian_network) is a short introduction of BN. Some works related to Bayesian network for modeling user are discussed in section [3](#_I.3._Bayesian_network) and most of them do not have mechanism for the evolution of BN like this research. Section [4](#_III.1.3.1._Learning_parameters) discusses about main subject “learning parameters or the evolution of parameters”. Section [5](#_III.1.3.2._Learning_parameters) mentions how to learn parameters in case of missing data. Section [6](#_III.3.3._An_example) gives an example of learning parameters. Section [7](#_7._Conclusion) is the conclusion.

## 2. Bayesian network

This section [2](#_2._Bayesian_network) starts with a little bit discussion of Bayesian inference which is the base of both Bayesian network and inference in Bayesian network described later.

**Bayesian inference**

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 formula [2.1](#_Formula_III.1.1a._Bayes’) as follows:

**Formula 2.1.** Bayes’ rule

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 person, 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 authors (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 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 formulas [2.2](#_Formula_III.1.1b._Additional) and [2.3](#_Formula_III.1.1c._Multiplication), respectively as follows:

**Formula 2.2.** Additional rule

**Formula 2.3.** Multiplication rule

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 formula [2.4](#_Formula_III.1.1d._Total) as follows:

**Formula 2.4.** Total probability rule

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

**Formula 2.5.** Total probability rule in continuous case

Note, *P*(*Y|X*) and *P*(*X*) are continuous functions known as probability density functions mentioned right after.

Please pay attention to Bayes’ rule (formula [2.1](#_Formula_III.1.1a._Bayes’)) and total probability rule (formulas [2.4](#_Formula_III.1.1d._Total) and [2.5](#_Formula_III.1.1d’._Total)) because they are used frequently over the whole research.

**Bayesian network (BN)**

Bayesian network (BN) (Neapolitan, 2003, p. 40) (Nguyen L. , Overview of Bayesian Network, 2013, p. 1) is combination of graph theory and Bayesian inference. It having a set of nodes and a set of directed arcs is the directed acyclic graph (DAG); please pay attention to the terms “DAG” and “BN” because they are used over the whole research. 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](#_III.1.3.1._Learning_parameters) – 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, formula [2.6](#_Formula_III.1.1e._Definition) is the definition of CDF and PDF.

**Formula 2.6.** Definition of cumulative distribution function (CDF) and probability density function (PDF)

Because this section [2](#_2._Bayesian_network) 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.

E.g., in figure [2.1](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Figure_III.1.1._Bayesian), 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 2.1.** 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 [2.1](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Figure_III.1.1._Bayesian). Quantitative model includes parameters which are CPT (s) attached nodes in BN. Thus, CPT (s) as well as conditional probabilities are known as parameters of BN. Parameter learning is mentioned in sections [4](#_III.1.3.1._Learning_parameters) and [5](#_III.1.3.2._Learning_parameters). Please refer to (Neapolitan, 2003, pp. 441-505) for more details about structure learning.

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.

Suppose we use two notations *Xi* and *PA*(*Xi*) to indicate a node and a set of its parent, respectively. Let *X* be vector which was constituted of all *Xi*, *X* = (*X*1, *X*2,…, *Xn*). The **G**lobal **J**oint **P**robability **D**istribution (GJPD) *P*(*X*) being product of all local CPD (s) or CPT (s) is formulated as:

**Formula 2.7.** Global joint probability distribution of random vector

Suppose is the subset of *PA(Xi*) such that *Xi* must depend conditionally and directly on every variable in *.* In other words, there is always an arc from each variable in to *Xi* and no intermediate node between them. Thus, formula [2.7](#_Formula_III.1.2._Global) becomes:

**Formula 2.8.** Reduced global joint probability distribution of random vector

Note that *P*(*Xi*|) in formula [2.8](#_Formula_III.1.2’._Reduced) is the CPT of *Xi*. According to Bayesian rule, given evidence (random variables) , the posterior probability *P*(*Xi*|) of variable *Xi* is computed in formula [2.9](#_Formula_III.1.3._Posterior) as below:

**Formula 2.9.** Posterior probability of variable *Xi* given evidence

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 formulas [2.8](#_Formula_III.1.2’._Reduced) and [2.9](#_Formula_III.1.3._Posterior), we gain formula [2.10](#_Formula_III.1.3’._Advanced) as follows:

**Formula 2.10.** Advanced posterior probability of variable *Xi* given evidence

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 formula [2.10](#_Formula_III.1.3’._Advanced) is the base for inference inside Bayesian network, which is used over the whole research. Formulas [2.9](#_Formula_III.1.3._Posterior) and [2.10](#_Formula_III.1.3’._Advanced) are extensions of Bayes’ rule specified by formula [2.1](#_Formula_III.1.1a._Bayes’). It is not easy to understand formula [2.10](#_Formula_III.1.3’._Advanced) and so, please see formulas [2.12](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.12._Posterior) and [2.13](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.13._Posterior) which are advanced posterior probabilities applied into wet grass example in order to comprehend formula [2.10](#_Formula_III.1.3’._Advanced).

From figure [2.1](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Figure_III.1.1._Bayesian) of wet grass example and according to formula [2.7](#_Formula_III.1.2._Global), we have:

Applying formula [2.8](#_Formula_III.1.2’._Reduced), *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 formula [2.8](#_Formula_III.1.2’._Reduced), we have formula [2.11](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.11._Global) for determining global joint probability distribution of “wet grass” Bayesian network as follows:

**Formula 2.11.** Global joint probability distribution of wet grass Bayesian network

**Inference in Bayesian network**

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 [2.1](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Figure_III.1.1._Bayesian), 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 formula [2.10](#_Formula_III.1.3’._Advanced), expended by formulas [2.12](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.12._Posterior) and [2.13](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.13._Posterior) as follows:

**Formula 2.12.** Posterior probability of rain given wet grass evidence

**Formula 2.13.** Posterior probability of sprinkler given wet grass evidence

Note that the numerator in the right side of formula [2.12](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.12._Posterior) is the sum of possible probabilities over possible values of *C* and *S*. Concretely, we have an interpretation for the numerator as follows:

Applying formula [2.11](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.11._Global) 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 formulas [2.12](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.12._Posterior) and [2.13](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.13._Posterior) and the previous formula [2.10](#_Formula_III.1.3’._Advanced) 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, formulas [2.12](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.12._Posterior) and [2.13](file:///C:\Users\Admin\AppData\Roaming\Microsoft\Word\wet#_Formula_III.1.1.13._Posterior) are expansions of formula [2.10](#_Formula_III.1.3’._Advanced). 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 basic concepts of Bayesian network were introduced in this section [2](#_2._Bayesian_network). The next section [3](#_I.3._Bayesian_network) will describes some works related to Bayesian network for modeling user.

## 3. Bayesian network user model

Bayesian network approach is very important in the design of user modeling system. So I reserve this section [3](#_I.3._Bayesian_network) for glancing over the state of the art of Bayesian network model. My proposal (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) of Bayesian network model is described in detail in (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009).

There are three methods of building up Bayesian network user model: expert centric, efficiency centric and data centric (Mayo, 2001, p. 74).

* *Expert-centric method*: The structure and conditional probabilities are defined totally by experts. KBS hyperbook system (Henze, 2000), Andes (Conati, Gertner, & Vanlehn, 2002) are significant systems that apply this method.
* *Efficiency-centric method*: The structure and conditional probabilities are specified and restricted based on some restrictions. SQL-Tutor (Mitrovic, 1998) system applies this method.
* *Data-centric method*: The structure and conditional probabilities are learned directly from real-world data by machine learning algorithms such as information theory based approach (Cheng, Bell, & Liu, 1997).

However KBS hyperbook, Andes and SQL-Tutor are hybrid systems when they take advantage of both approaches expert-centric and efficiency method. I will introduce such significant systems. All user models in this section [3](#_I.3._Bayesian_network) are based on Bayesian network and so, you can read the report (Nguyen L. , Overview of Bayesian Network, 2013) which is a good introduction to generic Bayesian network together with basic concepts, inference and learning techniques. Recall that overlay model is essential graph model whose nodes are knowledge elements, which leads to many approaches to build up overlay model from statistics to machine learning and one of them is Bayesian network method. Thus, it is possible to say that Bayesian network is an advanced variant of overlay model. Please read the excellent article “Bayesian network for student model engineering” by authors (Millán, Loboda, & Pérez-de-la-Cruz, Bayesian networks for student model engineering, 2010) for comprehending applying Bayesian network user model in learning context.

### 3.1. KBS hyperbook system

KBS hyperbook system is developed by author (Henze, 2000) in her/his PhD thesis. In KBS hyperbook system, the domain is composed of a set of knowledge items (*KI*). Each *KI* can be the concept, topic, etc. that student must master. There is a partial order on *KI* (s) to express the prerequisite relationships among them. Suppose *KI*1 is prerequisite for *KI*2, the partial order is denoted as *KI*1 *< KI*2. It means that student must master *KI*1 before learning *KI*2. Given user *U*, the user knowledge *KV*(*U*) is represented as a knowledge vector in which the *ith* component of this vector is the conditional probability expresing how user masters the *KIi* (Henze, 2000, p. 46).

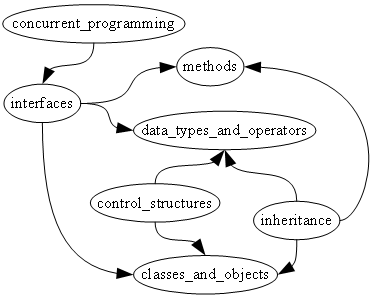
*KV*(*U*) = { *P*(*KI*1*|*), *P*(*KI*2*|*),..., *P*(*KIn|*) }

Where *KI*1*, KI*2*,..., KIn* denotes knowledge items and is evidence that system observes about user in learning process. Note that *notation P*(*.*) *denotes the probability in this research*.

The author (Henze, 2000, p. 52) defines the dependency graph as the neighbouring graph in which the nodes are *KI*(s) and the arcs represent partial order among *KI* (s). Namely, that the arc from node *B* to node *A* and there is no node *Z* interveneing between *A* and *B* (*A<Z<B*) tells us the order *A < B*. If a *KI* has no prerequisite, it is called top-most *KI*. All *KI* (s) are classified into three levels.

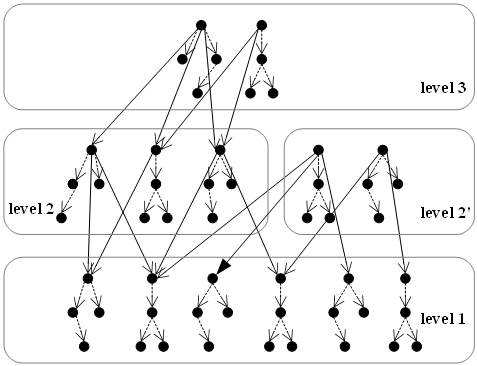
* The first level includes top-most *KI* (s).
* The second level includes *KI* (s) that require top-most *KI* (s). This level is further divided into two parts: one that is prerequisite for some third-level *KI* (s) and one that is not required by any third-level *KI*.
* The third level includes *KI* (s) that require second-level *KI* (s).

In each level, there are always *KI* (s) so-called main *KI* (s) that have no parent. Main *KI* (s) in the same level are assumed to be mutually independent. Figure [3.1.1](#_Figure_I.3.1._An) (Henze, 2000, p. 52) depicts an example of dependency graph mentioning basic concepts of Java programming language (Oracle, Java language).



**Figure 3.1.1.** An example of dependency graph

Figure [3.1.2](#_Figure_I.3.2._The) (Henze, 2000, p. 53) expresses levels of *KI* (s).



**Figure 3.1.2.** The levels of *KI* (s)

Each top-most *KI* has a set of its children describing it in more detail. So the root tree is defined as the sub-graph having a top-most *KI* and all remaining nodes are children of *KI*. If there are *n* top-most *KI*, we have *n* root trees. Figure [3.1.3](#_Figure_I.3.3._Root) is an example of 1 root tree.

**Figure 3.1.3.** Root tree

The dependency graph and a set of root trees found the Bayesian network in which nodes are represented as random variables and arcs are expressed by conditional probability tables. Each variable (*KI*) has four discrete grades {*E, F, A, N*} as follows (Henze, 2000, p. 46):

* *E* is an abbreviation of expert, which refers that user has expert’s knowledge on a *KI*.
* *F* refers that user has advanced knowledge on a *KI* with some difficulties but mainly excellent.
* *A* refers that user has beginner’s knowledge on a *KI.*
* *N* is an abbreviation of novice, which refers that user does not know anything about a *KI*.

The computation expense of inference tasks increases exponentially when continuously directed cycles exists in network. There are three approaches to eliminate cycles from Bayesian network: clustering, conditioning and stochastic simulation.

* *Clustering approach*: The nodes that cause directed cycle are clustered to single node, as seen in figure [3.1.4](#_Figure_I.3.4._Clustering) (Henze, 2000, p. 54).

**Figure 3.1.4.** Clustering approach

* *Conditional approach*: The whole network having directed cycles is transformed into some simpler sub-networks. Each sub-network includes variables instantiated to one of their values. For example, if nodes have two values: 0,1 then whole network is transformed into two sub-networks: one for instances of variables having value 0 and one for instances of variables having value 1.
* *Stochastic approach*: The simulation of network is run repeatedly for calculating approximations of the exact evaluation (Henze, 2000, p. 55).

#### 3.1.1. Yet Another Clustering Formalism (YACF)

The author (Henze, 2000, pp. 55-62) developed a new clustering approach so-called Yet Another Clustering Formalism (YACF) enabling to generate a directed graph without cycles in the underlying undirected graph. YACF gives an additional cluster node while other the nodes (normal nodes) in clusters are not changed, only the conditional probability tables of the child vertices of the cluster must be changed. Note that there are two kinds of nodes: the (normal) node that represents *KI* and the (additional) cluster node.

The additional cluster node (Henze, 2000, p. 55) is the node that owns income nodes (so-called parent nodes) and outcome nodes (so-called child nodes). The cluster node receives information (maybe evidence) from parent nodes and distributes it to child nodes. This is the information propagation from parents to children. The cluster node is realized as the random cluster variable whose range is the sum of the ranges of all child nodes. One part of the range of cluster variable holds for a particular child node. Thus each child has to listen only to the part of cluster’s variable which holds information about it. For example, there are three variables *KI*1, *KI*2, *KI*3 and both *KI*1 and *KI*2 are parents of *KI*3. If *KI*1 has value *E* (expert) and *KI*2 has value *A* (beginner) then the value of *KI*3 is the best grade among values of *KI*1 and *KI*2; so *KI*3 has value *E* (expert). It means that the information is passed from *KI*1 to *KI*3.

The excellence of YACF method is only to specify the conditional probability tables (CPT) of cluster node and child nodes. It isn’t necessary to re-construct whole network; the structure of network and the CPT (s) of parent nodes are keep in origin. Figure [3.1.1.1](#_Figure_I.3.5._YACF) (Henze, 2000, p. 56) describes YACF method.

Nodes of level 3

Nodes of level 2

Nodes of level 2’

Nodes of level 1

**…..**

**…..**

**…..**

**…..**

**Figure 3.1.1.1.** YACF method

#### 3.1.2. How to define CPT (s) of cluster node and child nodes

Suppose *X*1, *X*2,…, *XN* are parent nodes and *Y*1, *Y*2,…, *YM* are child nodes and *H* is cluster node (see in figure [3.1.2.1](#_Figure_I.3.6._Cluster)) (Henze, 2000, p. 57). Each *Yi* where is depedent on at least one *Xk* where . Let 1*Yi*,…, *LYi* denote the part of the range of *H* which carries information for node *Yi*. Note that if *KI* has for values as discussed {*E, F, A, N*} then the set {1,…, *L*} becomes {*E, F, A, N*} and values 1*Yi*,…, *LYi* are often denoted *E*\_*Yi, F*\_*Yi, A*\_*Yi* and *N*\_*Yi*.

*X*1

*X*2

*X*3

*X*4

*X*1

*X*2

*X*3

*X*4

*Y*1

*Y*2

*Y*3

*Y*1

*Y*2

*Y*3

**Figure 3.1.2.1.** Cluster node

The CPT of cluster node *H* is defined as the matrix where *R*(*.*) denotes the range of given variable. The author (Henze, 2000, p. 56) proposed formula [3.1.2.1](#_Formula_I.3.1.1._Condition) for calculating conditional probability of cluster node.

**Formula 3.1.2.1.** Condition probability of cluster node

Where *best\_grade* returns the maximum value of parent variables *Xl* (s) on which *Yi* is dependent. Table [3.1.2.1](#_Table_I.3.1._CPT) (Henze, 2000, p. 57) shows the conditional probability table (CPT) of cluster node *H*, based on formula [3.1.2.1](#_Formula_I.3.1.1._Condition).

|  |  |  |  |
| --- | --- | --- | --- |
| ***X*1,……...*Xk*,……..,*Xn*** | *range of H holding*  *evidence for node Y*1  ***P*(*H*=*E\_Y*1*|*…)…*P*(*H*=*N\_Y*1*|*…)** | **…** | *range of H holding*  *evidence for node YM*  ***P*(*H*=*E\_YM|*…)…*P*(*H*=*N\_YM|*…)** |
| *X1*=*E*,…, *Xk*=*E*,…, *XN*=*E*  **.**  **.** | 1/*M* 0 0 0  **.**  **.** | **…** | 1/*M* 0 0 0  **.**  **.** |
| *X1*=*F*,…, *Xk*=*N*,…, *XN*=*E*  **.**  **.** | 01/*M* 0 0  **.**  **.** | **…**  **.**  **.** | 1/*M* 0 0 0  ***.***  ***.*** |
| *X1*=*N*,…, *Xk*=*E*,…, *XN*=*N*  **.**  **.** | 1/*M* 0 0 0  **.**  **.** | **…**  **.**  **.** | 0 0 0 1/*M*  **.**  **.** |

###### **Table 3.1.2.1.** CPT of a YACF cluster node

The conditional probability of child node *Yi* where given cluster node *H* is defined in the following (Henze, 2000, p. 57):

**Formula 3.1.2.2.** Conditional probability of child node *Yi*

Table [3.1.2.2](#_Table_I.3.2._CPT) (Henze, 2000, p. 58) shows conditional probability of an arbitrary child node *Y* given cluster node *H*, based on formula [3.1.2.2](#_Formula_I.3.1.2._Conditional).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster node(*H*) | *P*(*Y*=*E|H*=…) | *P*(*Y*=*F|H*=…) | *P*(*Y*=*A|H*=…) | *P*(*Y*=*N|H*=…) |
| *H*1*=E*  **.**  **.**  *Hi*–1*=N* | 0.25  **.**  **.**  0.25 | 0.25  **.**  **.**  0.25 | 0.25  **.**  **.**  0.25 | 0.25  **.**  **.**  0.25 |
| ***Hi=E***  ***Hi=A***  ***Hi=F***  ***Hi=N***  (range of *H*  holding evidence  for *Y*) | **0.8**  **0.2**  **0.0**  **0.0** | **0.2**  **0.6**  **0.2**  **0.0** | **0.0**  **0.2**  **0.6**  **0.2** | **0.0**  **0.0**  **0.2**  **0.8** |
| *Hi*+1*=E*  **.**  **.**  *Hn=N* | 0.25  **.**  **.**  0.25 | 0.25  **.**  **.**  0.25 | 0.25  **.**  **.**  0.25 | 0.25  **.**  **.**  0.25 |

###### **Table 3.1.2.2.** CPT of a child node Y given cluster node H

The conditional probability of child node *Yi* given parent nodes *X*1,…, *XN* is defined by (Henze, 2000, p. 59) as below:

**Formula 3.1.2.3.** Conditional probability of child node *Yi* given parent nodes *X*1,…, *XN*

When conditional probability of each child node *Yi* is calculated according to formula [3.1.2.3](#_Formula_I.3.1.3._Conditional) which is core of YACF method, the Bayesian network without cycles is totally determined.

### 3.2. Andes

Andes developed by authors (Conati, Gertner, & Vanlehn, 2002) is the intelligent tutoring system (ITS) that helps students to solve physical problems or exercises. Please refer to (Mayo, 2001, pp. 2-3) for more details about ITS. The special thing in Andes is that the Bayesian network is not built up directly from training data or by experts like other systems; thus, for each problem or exercise, the rule-based problem solver generates the data structure called *solution graph* which is then converted into Bayesian network. The solution graph is initilized right before student solves a problem. Figure [3.2.1](#_Figure_I.3.7._Student) depicts student modeling in intelligent tutoring system OLAE (Martin & VanLehn, 1995, p. 580) which is applied into Andes.

Cognitive Model

Physics Problem (rules)

**Problem**

**Solution Graph**

Student Model

(Prior probabilities for all rules)

**Student**

**Bayesian Network**

Student Data

Updated

Bayesian network

Output Student Model

(Posterior probabilities for all rules)

**Figure 3.2.1.** Student modeling in Andes

The description of Andes is mainly extracted from the article “Using Bayesian Networks to Manage Uncertanty in Student Modeling” (Conati, Gertner, & Vanlehn, 2002). Readers are recommended to read this article for comprehending Andes.

#### 3.2.1. Solution graph

Andes also constructs physics knowledge base including physics rules which are used to encode solution graph. The following are some sample physics rules (Conati, Gertner, & Vanlehn, 2002, p. 10).

* *R-try-Newton-2law*: If the problem’s goal is to find a force then set the goal to try Newton’s second law to solve the problem.
* *R-goal-choose-body*: If there is a goal to try Newton’s second law to solve a problem then set the goal to select a body to which to apply the law.
* *R-body-by-force*: If there is a goal to select a body to apply Newton’s second law and the problem goal is to find a force on an object then select as body the object to which the force is applied.
* *R-normal-exists*: If there is a goal to find all forces on a body and the body rests on a surface then there is a normal force exerted on the body by the surface.

For example, there is a sample problem (Conati, Gertner, & Vanlehn, 2002, p. 11) shown in figure [3.2.1.1](#_Figure_I.3.8._Sample) below:

|  |  |
| --- | --- |
|  | Block A of mass 50kg rests on top of a table. Another block B of mass 10kg rests on top of block A.  What is the normal force exerted by the table on block A?  (Conati, Gertner, & Vanlehn, 2002, p. 11) |

**Figure 3.2.1.1.** Sample problem to find normal force

The goal of problem is to compute the normal force. Firstly, the problem solver generates the top-level goal of finding normal force. Secondly it determines the sub-goal of using Newton’s second law to find normal force. Finally, it generates three sub-goals corresponding to necessary steps so as to apply Newton’s second law: “choosing a body to which to apply the law”, “identifying all the forces on the body” and “writing the component equation *F* = *m*” (Conati, Gertner, & Vanlehn, 2002, p. 11). The solution graph is shown in figure [3.2.1.2](#_Figure_I.3.9._Solution) (Conati, Gertner, & Vanlehn, 2002, p. 11).

G-find-Nat

RA-try-Newton-2law

G-try-Newton-2law

RA-choose-bodies

G-choose-bodies

RA-choose-separate-bodies

G-define-bodies-A-B

F-A-is-body

G-define-forces-on-A

R-try-Newton-2law

R-choose-bodies

R-choose-separate-bodies

F-B-is-body

S-choose-body-

strategy

RA-choose-compound-body

G-define-compound-AB

R-choose-compound-body

**Figure 3.2.1.2.** Solution graph in Andes

Each node in the solution graph (Conati, Gertner, & Vanlehn, 2002, p. 11) denotes a particular type of information (goal, rule, rule application, strategy). For example, the nodes: *G-find-Nat*, *G-try-Newton-2law*, *G-choose-bodies*, *G-define-bodies-A-B*, *G-define-forces-on-A* denote goals: top-level goal of finding the value of normal force, sub-goal of using Newton’s second law to find normal force, sub-goal of choosing a body to which to apply the law, sub-goal of identifying all the forces on the body and sub-goal of writing the component equation *F* = *m*, respectively. These nodes and relationships among them are used to construct the task-specific part of Bayesian network.

#### 3.2.2. Bayesian network in Andes

The Bayesian network in Andes includes two parts (Conati, Gertner, & Vanlehn, 2002, p. 12): one part so-called *domain-general part* that encodes the domain-general knowledge and another part so-called *task-specific part* that encodes the task-specific knowledge. While domain-general knowledge base includes general concepts and procedures which define the proficiency in domain, task-specific knowledge base represents knowledge related to students’ performance on problems and exercises.

Domain-general part is stable when it is based on domain-general knowledge base specified by experts. Its structure is maintained across problems and examples. The marginal probability of each node in this part is always computed when students finish their exercise, which expresses students’ mastery of such node. On the contrary, the task-specific part is temporal when it is automatically generated from the solution graph of each problem or exercises on which students work. When students finish their problem or exercise, the task-specific part is discarded (but it will be re-constructed in the next time) and the posterior marginal probabilities of domain-general part is computed and used as the priors for next time.

**Domain-general part in Bayesian network**

This part models student knowledge, whose nodes are classified into two types: *rule* and *context-rule* (Conati, Gertner, & Vanlehn, 2002, p. 12). Each node has two values: 0 denoting not mastered and 1 denoting mastered. A rule node represents a piece of knowledge in its fully general form while a context-rule node represents the mastery of a rule in concrete problem solving context. There is always a conditional relationship between a rule node and a context-rule node, in which rule node is the parent of context-rule node, as seen in figure [3.2.2.1](#_Figure_I.3.10._Relationship) (Conati, Gertner, & Vanlehn, 2002, p. 13). It means that the parent (rule node) represents the general knowledge and the child (context-node) tells us how the student masters such general knowledge in specific context.

rule

context-rule 1

context-rule 2

context-rule *n*

**.....**

**Figure 3.2.2.1.** Relationship between rule and context-rule nodes

The conditional probability *P*(*context-rule*=*mastered* | *rule*=*mastered*) equals 1 because if the student masters the general knowledge then she/he can apply it to solve any problems or exercise. The conditional probability *P*(*context-rule*=*mastered* | *rule*=*not-mastered*) expresses the probability that student solves successfully a problem or exercise even if she/he doesn’t master the general knowledge. How to specify the conditional probability of context-rule is the role of experts.

**Task-specific part in Bayesian network**

The task-specific part is temporal because it is discarded right after students finish their work and it is re-constructed in the next time. The task-specific part includes *context-rule* nodes and four other nodes: *fact, goal, rule-application, strategy* which are denoted with prefix *r-, f-, g-, ra-, s-,* respectively (Conati, Gertner, & Vanlehn, 2002, p. 13). These nodes are created from the solution graph of the problem or exercise on which students work. In other words, solution graph is the foundation of task-specific part. The structure of solution graph is kept intact in Bayesian network.

*Fact* and *goal* nodes (Conati, Gertner, & Vanlehn, 2002, p. 14) represent the propositions in domain; thus they are called *proposition nodes* (denoted with prefix *pr-*). They express the information that is derived when students apply context rules into solving problems or exercise. That a proposition node gets value 1 (*true*) means that the student can infer such proposition from her/his knowledge and otherwise. The parents of a proposition node are nodes from which it is derived and the real relationship between proposition node and its parents is *leaky-OR* relationship in which the conditional probability of proposition node given its parents equals 1 if at least one of such parents gets *true*. In case that all its parents are *false*, this probability equals the predefined real number *β* so-called a “*leak*”.

*Rule-application* nodes (Conati, Gertner, & Vanlehn, 2002, p. 14) are responsible for aggregating context-rule nodes, proposition nodes and strategy node so as to derive a new proposition node. One of the parents of a rule-application node must be a context-rule node. It implicates whether students can apply the rule into solving their problem or exercise. The relationship between rule-application node and its parents is *noisy-AND* relationship in which the conditional probability of rule-application node given its parents equals 1*–α* only if all such parents gets *true*. The predefined real number *α* is called a “*noise*”. If at least one of its parents gets *false*, this conditional probability equals 0. It means that the student must master all context rules before she/he applies such rules into solving problem or exercise. In case that she/he even knows whole rules, it is possible to assert totally that she/he can apply perfectly rules. Figure [3.2.2.2](#_Figure_I.3.11._Relationship) shows relationship between nodes in task-specific part (Conati, Gertner, & Vanlehn, 2002, p. 14).

context-rule

prop-1

prop-2

prop-*i*

rule-app-1

rule-app-2

**noisy-AND**

*P*(*ra=1* |all parents*=*1) *=*1–*α*

*P*(*ra=*1 |at least one parent*=*0)*=*0

prop-*k*

**leaky-OR**

*P*(*prop=*1 |one or more parents*=*1) *=*1

*P*(*prop=*1 |all parents*=*0)*=β*

**Figure 3.2.2.2.** Relationship between nodes in task-specific part

*Strategy* nodes (Conati, Gertner, & Vanlehn, 2002, p. 15) are used in situation that there are different solutions to a problem. For example, there are two application rules aiming to solve the same goal. When the posterior probability of one application rule lessens the posterior probability of another, it raises the issue so-called mutually exclusive strategy (Conati, Gertner, & Vanlehn, 2002, p. 15), as seen in figure [3.2.2.3](#_Figure_I.3.12._Mutually).

rule1

goal

rule2

rule application1

rule application2

**Figure 3.2.2.3.** Mutually exclusive strategy

The strategy node is associated with goal node in order to come over mutually exclusive situation. Both strategy node and goal node are parent of some goal nodes. Each goal node (child node) is considered as different strategy when student solves a problem and it corresponds with one value of strategy node. Of course, the number of values of strategy node is the same to the number of goal nodes which are its children. The probability of one value of strategy node expresses the frequency of respective strategy that student may choose as the solution for her/his problem. The higher this probability is, the more student prefers to select respective strategy. Strategy node is shown in figure [3.2.2.4](#_Figure_3.2.2.4._Strategy).

rule1

goal

rule2

rule application1

rule application2

strategy

**Figure 3.2.2.4.** Strategy node

**Inference mechanism in Bayesian network**

Suppose a student who solves the problem of finding normal force in the example in figure [3.2.1.1](#_Figure_I.3.8._Sample) chose block A as body. At that time, the fact node *F-A-is-body* gets value 1 (*true*). When the evidence raised by this fact node is entered, the posterior probabilities of all nodes are changed according to propagation way. Such posterior probabilities reflect Bayesian network inference mechanism used to model student’s problem solving. Figure [3.2.2.5](#_Figure_I.3.14._Prior/Posterior) (Conati, Gertner, & Vanlehn, 2002, p. 18) tells us the prior/posterior probabilities of all nodes in the task-specific part (also solution graph) of Bayesian network.

G-find-Nat

RA-try-Newton-2law

G-try-Newton-2law

RA-choose-bodies

G-choose-bodies

RA-choose-separate-bodies

G-define-bodies-A-B

F-A-is-body

G-define-forces-on-A

R-try-Newton-2law

R-choose-bodies

R-choose-separate-bodies

F-B-is-body

S-choose-body-

strategy

RA-choose-compound-body

G-define-compound-AB

**0.99/0.99**

**0.32/0.33**

**0.24/0.25**

**0.11/0.22**

**0.15/1.0**

**0.12/0.24**

**0.16/0.27**

**0.11/0.09**

**0.50/0.51**

**0.50/0.50**

R-choose-compound-body

**0.50/0.51**

**0.50/0.51**

**0.50/0.50**

**Figure 3.2.2.5.** Prior/Posterior probabilities in the task-specific part

### 3.3. SQL-Tutor and constraint-based modeling

**Constraint-based modeling (CBM)**

There are two types of user knowledge: generative and evaluative. Generative knowledge means that user has actual ability about some learning skills. However, in the real situation, students may discriminate between the correct and incorrect solution to a problem before they master such problem. This is the evaluative knowledge. Constraint-Based Modeling (CBM) aims to model evaluative knowledge. A constraint is a pair <*Cr*,*Cs*> denoting *relevance condition* and *satisfaction condition*, respectively (Mayo, 2001, p. 71). Both *Cr* and *Cs* are patterns used to match the states of student’s solutions but *Cs* is more specific than *Cr*.

For example, the *Cr*=(*n*1*+n*2*=\**) of a constraint is defined to match any string of form *n*1*+n*2*=\** where *n* denotes any variable and *\** denotes any string. So some expressions like “1*+*1*=*2”, “7*+*1*=*9”, “*A+B=CD*” match this *Cr* but other expressions like “1234”, “*A=BC*” do not. *Cr* defines the class of student’s solutions.

*Cs* is more specific than *Cr* and it defines the correctness of student’s solutions. An example for *Cs* is *n*1*+n*2*=add*(*n*1*, n*2) where the function *add* is responsible for adding two numbers. So the expression “1*+*1*=*2” matches this *Cs* but the expression “3*+*2*=*6” is wrong.

If student’s solution is matched with both *Cr* and *Cs*, the constraint <*Cr*,*Cs*> is *satisfied* for this solution. If only *Cr* matches the solution, we call that the constraint is *relevant* to solution. If *Cr* does not match this solution, the constraint is *violated*. Following is the matching process (Mayo, 2001, p. 71):

|  |
| --- |
| If *matches*(*student-solution, Cr*)Then  If *matches*(*student-solution, Cs*)Then  *constraint-is-satisfied;*  Else  *constraint-is-relevant;*  End If  Else  *constraint-is-violated;*  End If |

In case that the constraint is violated, the constraint-specific tutoring system can begin.

**Architecture of SQL-Tutor**

SQL-Tutor developed by author (Mitrovic, 1998) is the constraint-specific tutoring system teaching SQL database language. The knowledge base in SQL-Tutor is a set of constraints describing rules of SQL language (Ramakrishnan & Gehrke, 2003, pp. 130-173). The architecture of SQL-Tutor (figure [3.3.1](#_Figure_I.3.15._Architecture)) has three functional models: CBM student modeler, pedagogical module and interface (Mitrovic, 1998, p. 309). As seen in figure [3.3.1](#_Figure_I.3.15._Architecture), SQL-Tutor is an Intelligent Tutoring System (ITS). Please refer to (Mayo, 2001, pp. 2-3) for more details about ITS.

**CBM**

**Student Modeler**

Student Models

Constraints

Databases

Problems

Solutions

**Pedagogical**

**Module**

**Interface**

Students

**Figure 3.3.1.** Architecture of SQL-Tutor

The interface is responsible for interacting with student through graphic user interface (GUI). The CBM student modeler manages and updates student model. There are several databases and a set of problems for each database together with their solutions. Each problem has a concrete difficult level and each student is also assigned by a level of knowledge. CBM student modeler is responsible for increasing student’s level of knowledge if she/he is successful in solving some problems and otherwise her/his level of knowledge is decreased.

The pedagogical module is the most important module. It monitors student continuously and gives some pedagogical decisions (instructions) that help student to improve her/his knowledge. Pedagogical module gives student‘s problem that is appropriate to her/him. It means that it matches student’s level of knowledge with problem’s difficult level. When student solves problem, it sends this solution to CBM student modeler. If the solution is wrong it notices the feedback message, otherwise maybe it gives student the next problem.

**Bayesian network in SQL-Tutor**

SQL-Tutor is enhanced and extended by author (Mayo, 2001) in his PhD thesis. The author (Mayo, 2001, p. 93) added probabilistic student model into SQL-Tutor. So, please focus on how to represent student model by probabilistic approach instead of increasing or decreasing student’s knowledge and how to apply Bayesian network into SQL-Tutor student model (Mayo, 2001, p. 93). The student model is constituted of a set of binary variables (*mastered*1, *mastered*2,…, *masteredn*) where *masteredc* expresses whether the constraint *c* is mastered (*masteredc=*1) by user or not (*masteredc=*0). *P*(*masteredc=*1) is the certain probability that student masters constraint *c*. The initial value of *P*(*masteredc=*1) is the ratio of the frequency that constraint *c* is satisfied to the frequency that constraint *c* is relevant in the past.

After student solves her/his problem and receives the feedback from pedagogical module, the probability *P*(*masteredc=*1) is updated according to following heuristic rules (Mayo, 2001, p. 94):

* If constraint *c* is satisfied then *P*(*masteredc=*1) increases by 10% of the value 1–*P*(*masteredc=*0).
* If constraint *c* is violated and no feedback about *c* is given then *P*(*masteredc=*1) decreases by 20%.
* If constraint *c* is violated but feedback is given about *c* then *P*(*masteredc =*1) increases by 20% of the value 1*–P*(*masteredc=*1).

Instead of using such rules, the author (Mayo, 2001, p. 94) proposes another method which applies Bayesian inference (Wikipedia, Bayesian inference, 2006) to update the probability that constraint is mastered. Let *M* denote student’s mastery of constraint and let *L* denote the outcome of the last student’s solution at this constraint. Both *M* and *L* are binary variables in which *M* takes values 1 (mastered) and 0 (not mastered) and variable *L* takes value 1 (satisfied) and 0 (violated). Suppose the prior probability of student’s mastery is *P*(*M*), the essence of updating such probability is to compute the posterior probability *P*(*M|L*) when outcome *L* is observed. Note that *P*(*M|L*) denotes the probability that student masters (doesn’t master) the constraint given that this constraint is satisfied (violated). The author (Mayo, 2001, p. 95) proposes formula [3.3.1](#_Formula_I.3.3.1._Posterior) to compute *P*(*M|L*).

**Formula 3.3.1.** Posterior probability of student’s mastery

Where *m, l* {0, 1} denote values of *M, L*, respectively and *P*(*L|M*) is the probability that constraint is satisfied (violated) given that student masters (doesn’t master) this constraint. The probability *P*(*L|M*) is considered as the likelihood function of the student’s mastery and defined by experts.

It is necessary to predict the performance of student given the problem *p* on constraint *c*. Let *masteredc* be the binary expressing whether student masters constraint *c*. The binary *relevantISc,p* {0, 1} expresses whether constraint *c* is relevant to the ideal solution of problem *p*. The binary *relevantSSc,p* {0, 1} expresses whether constraint *c* is relevant to the student’s solution of problem *p*. That variable *relevantSSc,p* depends on *relevantISc,p* {0, 1} implicates that the student’s solution must match the ideal solution. The variable *performancec,p* having three values *satisfied, violated* and *not-relevant* denotes the performance of student given the problem *p* on constraint *c*. The variable *performancec,p* depends on both *relevantSSc,p* and *masteredc*. The Bayesian network representing these variables and relationships among them is shown in figure [3.3.2](#_Figure_I.3.16._Bayesian) (Mayo, 2001, p. 98).

**Figure 3.3.2.** Bayesian network in SQL-Tutor

As discussed, the prior probability of *materedc*, *P*0(*materedc=*1), is defined by Bayesian inference (Wikipedia, Bayesian inference, 2006) or heuristic rule. The prior probability of *relevantISc,p* is specified by expert. According to (Mayo, 2001, p. 98), the conditional probability table (CPT) of *relevantSSc,p* given *relevantISc,p* is defined as in table [3.3.1](#_Table_I.3.2._Conditional).

|  |  |  |  |
| --- | --- | --- | --- |
|  | *relevantISc,p* | | |
| *relevantSSc,p* |  | *yes* (1) | *no* (0) |
| *yes* (1) | *αc* | *βc* |
| *no* (0) | 1*–αc* | 1*–βc* |

###### **Table 3.3.1.** Conditional probability table of *relevantSSc,p* given *relevantISc,p*

According to (Mayo, 2001, p. 98), the parameter *αc* (*βc*) denotes the probability of constraint *c* being relevant to the student’s solution given that the student’s solution is (not) relevant to the problem’s ideal solution. It is stated that the parameters *αc*, *βc* indicate the usefulness of ideal solution or the effect of ideal solution on student’s solution. They are defined by experts or as the estimation which is computed from log files. For example, the parameter *αc* is the ratio of the frequency that constraint *c* is relevant to both ideal solution and student’s solution to the frequency that constraint *c* is relevant to ideal solution. The parameter *βc* is the ratio of the frequency that constraint *c* is relevant to student’s solution but not relevant to ideal solution to the frequency that constraint *c* is not relevant to ideal solution.

### 3.4. Data-centric approach

Authors (Cheng, Bell, & Liu, 1997) proposed the considerable method for learning Bayesian network structure from training data. In this method, the correlation between two nodes is measured by the amount of information flow between them. Such measurement is called mutual information (Mutual information, 2014). The higher the mutual information of two nodes is, the more the correlation between them is, in other words, the more likely there is an arc connecting them. The mutual information of two nodes *X* and *Y* is defined as below (Cheng, Bell, & Liu, 1997, p. 2).

Note that notation *log*(*.*)denotes logarithm function and *x*, *y* (s) are possible instances of *X*, *Y*, respectively. Note that *notation P*(*.*) *denotes the probability in this research* and *P*(*x*, *y*) is joint probability of *x*, *y*.

The conditional mutual information is defined as below (Cheng, Bell, & Liu, 1997, p. 2).

Where *C* is a set of nodes and *x*, *y*, *c* (s) are possible instances of *X*, *Y*, *C*, respectively.

Given the threshold ζ, if the conditional mutual information *I*(*X,Y*|*C*) is smaller than ζ then two nodes *X, Y* are d-separated by set *C*.

The authors (Cheng, Bell, & Liu, 1997, pp. 2-3) proposed an algorithm for learning the structure of Bayesian network which includes three phases: *drafting*, *thickening* and *thinning*. However, before discussing about this algorithm, it is necessary to know the concept “*d-separation*” and “*cut-set*”. Give a set *C* and two nodes (*A*, *B*), the statements “*A is d-separated from B by C*”, “*C d-separates A from B*” or “*there is a d-separation between A and B given C*” mean that there is no active (open) undirected path between *A* and *B*. The path between *A* and *B* is active if every node in the path having head-to-head arrows (like *X*→*Z*←*Y*) is in *C* or has a descendant in *C* and every other node in the path is outside *C*. The concept “d-separation” ensures that the evidence about one node doesn’t affect on other node. The smallest set of nodes that d-separates *A* from *B* is called the cut-set of *A* and *B*.

In the first phase, *drafting phase* (Cheng, Bell, & Liu, 1997, p. 2), given the empty ordered set *S* andthe threshold ζ, for each pair of nodes *X* and *Y*, the mutual information *I*(*X,Y*) is computed by above formula. All of these pairs whose *I*(*X,Y*) is larger than ζ are sorted into the set *R* according to their respectively *I*(*X,Y*) in descending order. Starting with picking up the first pair whose *I*(*X,Y*) is largest from *S*; if there is no undirected path between *X* and *Y* (these two nodes are d-separated given empty set) then an undirected arc is added between *X* and *Y*. This is repeated until *S* contain only pairs that aren’t adjacent but are connected via a longer path. The output of this phase is the single-connected network or some unconnected single-connected networks. It means that maybe there is lack of some arcs in networks.

The second phase, *thickening phase* (Cheng, Bell, & Liu, 1997, p. 3), given the remaining pairs (*X, Y*) in *S*, if there is no cut-set that d-separates *X* and *Y* then an arc is added between *X* and *Y* because *X* and *Y* are dependent. The output of this phase is the network that is full of arcs.

After thickening phase, some redundant arcs can occur in networks. For example, two nodes *X* and *Y* are d-separated and there is no cut-set that d-separated them; so an arc is added between them. But more arcs are added to network in thickening phase and there may be cut-sets that d-separate *X* from *Y*. At that time, the arc between *X* and *Y* becomes redundant. So the purpose of the last phase, *thinning phase*, is to remove redundant arcs from network. The *thinning phase* (Cheng, Bell, & Liu, 1997, p. 3) includes two steps:

* Firstly, for each pair of adjacent nodes (*X,Y*), removing temporarily the arc connecting them.
* Secondly, the algorithm tries to find the cut-set that separates *X* from *Y*. If such cut-set exists then this arc is removed permanently from network; otherwise it is kept intact.

The output of *thinning phase* is the final structure of Bayesian network. Now some famous works related to Bayesian network for modeling user were discussed in this section [3](#_I.3._Bayesian_network). The next section [4](#_III.1.3.1._Learning_parameters) discusses about main subject “learning parameters or the evolution of parameters”.

## 4. Learning parameters in Bayesian model

Parameter learning or parameter evolution is essentially to update conditional probability tables (CPT (s)) in Bayesian network (BN) based on issued evidences. In other words, this is to compute posterior probabilities of each node in BN with note that nodes are random (binary) variables and so, the main content of parameter learning is to apply beta function into calculating such posterior probabilities, which is described as below. Note that some proofs, definitions, or formulas in this section [4](#_4._Learning_parameters) are found in the book “Learning Bayesian Networks” by the author (Neapolitan, 2003) from page 293 to page 373 but I rearrange them and prove them again by myself with a few changes according to the purpose of this section [4](#_4._Learning_parameters) – the evolution of parameters in BN. Readers are recommended to read the book “Learning Bayesian Networks” in order to understand comprehensively Bayesian network; this is an excellent book to which I referred. I express my deep gratitude to the author Richard E. Neapolitan for providing the great book.

It is conventional that definitions, theorems, corollaries and lemmas are noted as formulas so that it is easy for readers to follow and look up mathematical formulas.

**Dummy variables and augmented BN**

In continuous case, the conditional probability table (CPT) of each node is replaced by the probability density function (PDF). Recall that CPT is essentially collection of discrete conditional probabilities of each node with attention that node, variable, and random variable have the same meaning in BN context; please see section [2](#_2._Bayesian_network) for more details about CPT. There is a family of PDF which quantifies and updates the strength of conditional dependencies between nodes by natural way is called beta density function, denoted as *β*(*x*; *a*, *b*) or *beta*(*x*; *a*, *b*) with variable *x* and two parameters *a*, *b* (*N=a+b*) where *a* and *b* are positive numbers. Beta density function with two parameters *a* and *b* (Neapolitan, 2003, p. 300) is defined in formula [4.1](#_Formula_III.1.12._Beta).

**Formula 4.1.** Beta density function

Where Γ(.) denotes gamma function (Neapolitan, 2003, p. 298) which is essentially an integral approximated to factorial function as follows:

**Formula 4.2.** Gamma function

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 formula [4.2](#_Formula_III.1.13._Gamma) is equivalent to factorial function,

There is an important property of gamma function which is expressed in formula [4.3](#_Formula_III.1.14._Important) (Neapolitan, 2003, p. 298).

**Formula 4.3.** Important property of gamma function with regard to factorial function

Figure [4.1](#_Figure_III.1.10._Beta) 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 more narrow the graph is.

The integral in interval [0, 1] of the expression inside definition of beta function specified by formula [4.1](#_Formula_III.1.12._Beta) is determined by formula [4.4](#_Formula_III.1.15._Integral) as follow:

**Formula 4.4.** Integral of product expression *xa*(1 – *x*)*b*

Proof,

(due to beta density function specified by formula [4.1](#_Formula_III.1.12._Beta))

The formula [4.4](#_Formula_III.1.15._Integral) is lemma 6.2 in (Neapolitan, 2003, p. 300).

Suppose there is one binary variable *X* in network 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) |

**Formula 4.5.** Conditional probability (relative frequency) of *X* as value of dummy variable *F*

Note, statement “*F* has beta density function *β*(*F*; a, b)” is the same to statement “The probability density function (PDF) of *F* is *β*(*F*; a, b)”.

Please pay attention to the formula[4.5](#_Formula_III.1.16._Conditional), *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 dummy 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 [4.2](#_Figure_III.1.11._The) shows the simplest augmented BN.

*β*(*F*; *a*,*b*) and *F* have space [0,1]

*P*(*X=*1 | *F*) *=* *F*

**Figure 4.2.** The simple augmented BN with only one hypothesis node *X*

It is easy to infer that *P*(*X=*1) *= E*(*F*) where *E*(*F*) is the expectation of *F*.

Proof, owing to the total probability rule in continuous case (see formula [2.5](#_Formula_III.1.1d’._Total)), we have:

Because *F* is beta function, its expectation is , and so we have a very simple but effective formula to compute the probability of *X* as follows:

**Formula 4.6.** Probability of hypothesis *X* as expectation of beta variable *F*

Proof,

(due to formula [4.4](#_Formula_III.1.15._Integral))

(due to formula [4.3](#_Formula_III.1.14._Important))

Please pay attention to formula [4.6](#_Formula_III.1.17._Probability), it is the most essential formula used over the whole section [4](#_4._Learning_parameters). The formula [4.6](#_Formula_III.1.17._Probability) 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 formula [4.6](#_Formula_III.1.17._Probability)) 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 parameters learning*. Note that *P*(*X|*) can be referred as *P*(*X*(*m+*1) *|* ). Figure [4.3](#_Figure_III.1.12._The) depicts this sample  *=* (*X*(1), *X*(2),…, *X*(*m*)).

*P*(*X*(1) | *F*)=*F*

**…**

*P*(*X*(2) | *F*)=*F*

*P*(*X*(*m)* | *F*)=*F*

*β*(*F*; *a*,*b*) and *F* have space [0,1]

**Figure 4.3.** The sample *=*(*X*(1), *X*(2),…, *X*(*m*)) size of *m*

We only survey the case of binomial sample. Thus, having binomial distribution is called binomial sample and the network in figure [4.2](#_Figure_III.1.11._The) 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.

Owing the total probability rule in continuous case (see formula [2.5](#_Formula_III.1.1d’._Total)), we have

(by applying definition of beta function specified by formula [4.1](#_Formula_III.1.12._Beta))

(due to formula [4.4](#_Formula_III.1.15._Integral))

In brief, we have formula [4.7](#_Formula_III.1.17._Expectation) to determine expectation of *Fs*(1 – *F*)*t* as follows:

**Formula 4.7.** Expectation of expression *Fs*(1 – *F*)t

The same proof for formula [4.7](#_Formula_III.1.17._Expectation) is found in (Neapolitan, 2003, p. 306) and formula [4.7](#_Formula_III.1.17._Expectation) is lemma 6.4 in (Neapolitan, 2003, p. 305). The probability of evidences *P*() equals this expectation , which is interpreted as follows:

(because evidence contains independent random variables *X*(1), *X*(2),…, *X*(*m*))

In brief, we have formula [4.8](#_Formula_III.1.18._Probability) to determine the probability *P*() of evidences .

**Formula 4.8.** Probability *P*() of evidences

The similar proof for formula [4.8](#_Formula_III.1.18._Probability) is found in (Neapolitan, 2003, p. 307) and formula [4.8](#_Formula_III.1.18._Probability) is corollary 6.2 in (Neapolitan, 2003, p. 307). The probability *P*() is also called *marginal probability* of evidence sample (see section [2](#_2._Bayesian_network)).

**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*. The beta density function is updated based on evidences as follows:

(According to Bayes’ rule shown in formula [2.1](#_Formula_III.1.1a._Bayes’))

(By applying formulas [4.1](#_Formula_III.1.12._Beta) and [4.7](#_Formula_III.1.17._Expectation))

Briefly, the posterior density function is *β*(*F*; *a+s*, *b+t*) where the prior density function is *β*(*F*; *a*, *b*), which is expressed in formula [4.9](#_Formula_III.1.19._Posterior).

**Formula 4.9.** Posterior beta density function

The similar proof formula [4.9](#_Formula_III.1.19._Posterior) is found in (Neapolitan, 2003, pp. 306-308) and formula [4.9](#_Formula_III.1.19._Posterior) is corollary 6.3 in (Neapolitan, 2003, p. 308). According to formula [4.6](#_Formula_III.1.17._Probability), the posterior probability of *X* is totally determined as below:

**Formula 4.10.** Posterior probability of *X*

Formula [4.10](#_Formula_III.1.21._Posterior) is theorem 6.4 in (Neapolitan, 2003, p. 309). In general, you should merely remember the formulas [4.1](#_Formula_III.1.12._Beta), [4.6](#_Formula_III.1.17._Probability), [4.9](#_Formula_III.1.19._Posterior), [4.10](#_Formula_III.1.21._Posterior) and the way to recognize prior density function, prior probability of *X*, posterior density function, and posterior probability of *X*, respectively. Additionally, formula [4.5](#_Formula_III.1.16._Conditional) attaching beta density function to CPT, which is the base of these formulas, should be considered. Please pay attention that the prior probability implies that the CPT of *X* is represented by beta density function *β*(*F*; *a*, *b*). After receiving evidences, the posterior probability is re-calculated, which implies that the CPT of *X* is evolved (learned) and represented by updated beta density function *β*(*F*; *a+s*, *b+t*). This is the process of parameter learning or the evolution of Bayesian model aforementioned in [beginning of this section 4](#_III.1.3.1._Learning_parameters). The next part will mention this evolution for complex Bayesian model with more than one hypothesis node.

**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 CPT (s) in following figure [4.4](#_Figure_III.1.13._BN) (Neapolitan, 2003, p. 329):

*P*(*X*1*=*1) *P*(*X*1*=*0)

1/2 1/2

*X*1 *P*(*X*2*=*1)

1. 1/2

0 1/2

**(a)**

*β*(*F*11; 1, 1)

*β*(*F*22; 1, 1)

*β*(*F*21; 1, 1)

**(b)**

**Figure 4.4.** BN (a) and complex augmented BN (b)

In figure [4.4](#_Figure_III.1.13._BN), 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 *ci=*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 *ci* parent instances ; in other words, *ci* denotes the size of *PAi* and the size of *Fi*. For example, in figure [4.4](#_Figure_III.1.13._BN), 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 formula [4.11](#_Formula_III.1.22._Conditional) for connecting CPT of variable *Xi* with beta density function of dummy variable *Fi*.

|  |
| --- |
|  |

**Formula 4.11.** Conditional probability (relative frequency) of *Xi* given a parent instance *PAij*, as value of dummy variable in multi-node BN

Formula [4.11](#_Formula_III.1.22._Conditional) is an extension of formula [4.5](#_Formula_III.1.16._Conditional) in multi-node BN and formula [4.11](#_Formula_III.1.22._Conditional) degenerates to formula [4.5](#_Formula_III.1.16._Conditional) if *Xi* has no parent. Note that the beta density function of *Fij* is *β*(*Fij*; *aij*, *bij*) and of course, in figure [4.4](#_Figure_III.1.13._BN), 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 formula [4.12](#_Formula_III.1.21._Beta) as follows:

**Formula 4.12.** Beta density function *β*(*Fij*) corresponding to an instance of a parent of node *Xi*

Note that formulas [4.1](#_Formula_III.1.12._Beta) and [4.12](#_Formula_III.1.21._Beta) have the same meaning for representing beta function except that formula [4.12](#_Formula_III.1.21._Beta) 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:

**Formula 4.13.** Joint beta density function of variable *Xi* having *ci* parent instances

Besides the local parameter independence expressed in formula [4.13](#_Formula_III.1.24._Joint), 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. Formula [4.14](#_Formula_III.1.23._Global) expresses the global parameter independence of all *Fij* (s).

**Formula 4.14.** Global joint beta density function of *n* independent variable *Xi* (s)

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 [4.4](#_Figure_III.1.13._BN). 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:

**Formula 4.15.** Probability of variable *Xi* with respect to its parent instance as expectation of beta variable

The formula [4.15](#_Formula_III.1.26._Probability) is extension of formula [4.6](#_Formula_III.1.17._Probability) when variable *Xi* has parent and both of these formulas express prior probability of variable *Xi*. Following is proof of formula [4.15](#_Formula_III.1.26._Probability).

(due to local parameter independence specified in formula [4.13](#_Formula_III.1.24._Joint) when *Fij* (s) are mutually independent)

The formula [4.15](#_Formula_III.1.26._Probability) is theorem 6.7 proved by the similar way in (Neapolitan, 2003, pp. 334-335) to which I referred. For illustrating formulas [4.11](#_Formula_III.1.22._Conditional) and [4.15](#_Formula_III.1.26._Probability), recall that variables *Fij* (s) and their beta density functions *β*(*Fij*) (s) specify conditional probabilities of *Xi* (s) as in figure [4.4](#_Figure_III.1.13._BN), and so, the CPT (s) in figure [4.4](#_Figure_III.1.13._BN) is interpreted in detailed as follows:

Note that inverted probabilities in CPT (s) 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 [4.4](#_Figure_III.1.13._BN) 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 [4.4](#_Figure_III.1.13._BN), *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* or *evidence sample* or *training data* or *evidences*, 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 [4.4](#_Figure_III.1.13._Updated) is depicted by figure [4.5](#_Figure_III.1.14._Expanded) as below (Neapolitan, 2003, p. 337):

**…**

**(a)**

**(b)**

**Figure 4.5.** 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, formula [4.16](#_Formula_III.1.27._Probability) is used for calculating probability of evidences corresponding to variable *Xi* over *m* trials as follows:

**Formula 4.16.** Probability of evidences corresponding to variable *Xi*

Where,

* Number *ci* 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 formula [4.16](#_Formula_III.1.27._Probability), 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 formula [2.1](#_Formula_III.1.1a._Bayes’))

(applying multiplication rule specified by formula [2.3](#_Formula_III.1.1c._Multiplication) 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 formula [4.17](#_Formula_III.1.28._Probability) for calculating conditional probability *P*(*|F*1, *F*2,…, *Fn*) of evidence sample given *n* vectors *Fi* (s).

**Formula 4.17.** Probability of evidence sample given vectors *Fi*

The formula [4.17](#_Formula_III.1.28._Probability) 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 formula [2.5](#_Formula_III.1.1d’._Total))

(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 formula [4.18](#_Formula_III.1.29._Whole) for determining the whole probability *P*() of evidence sample as product of expectations of binomial trials.

**Formula 4.18.** Whole probability of evidence sample

Formula [4.18](#_Formula_III.1.29._Whole) is theorem 6.11 in (Neapolitan, 2003, p. 343). There is the question “how to determine in formula [4.18](#_Formula_III.1.29._Whole)” and so we have formula [4.19](#_Formula_III.1.30._Expectation) for calculating this expectation by extending formula [4.7](#_Formula_III.1.18._Expectation), as follows:

**Formula 4.19.** Expectation of binomial trials

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 formula [2.1](#_Formula_III.1.1a._Bayes’))

(Due to total probability rule in continuous case, specified by formula [2.5](#_Formula_III.1.1d’._Total). Note that *Fi* = {*Fi*1, *Fi*2,…, })

(due to formula [4.17](#_Formula_III.1.28._Probability))

(applying formula [4.18](#_Formula_III.1.29._Whole) into denominator)

(applying definition of beta density function specified by formula [4.1](#_Formula_III.1.12._Beta) into numerator and applying formula [4.19](#_Formula_III.1.30._Expectation) into denominator, note that *Nij* = *aij* + *bij* and *Mij* = *sij* + *tij*)

(due to definition of beta density function specified in formula [4.1](#_Formula_III.1.12._Beta))

In brief, we have formula [4.20](#_Formula_III.1.31._Posterior) for calculating posterior beta density function *β*(*Fij|*).

**Formula 4.20.** Posterior beta density function in multi-node BN

Note that formula [4.20](#_Formula_III.1.31._Posterior) is an extension of formula [4.9](#_Formula_III.1.19._Posterior) in case of multi-node BN. Formula [4.20](#_Formula_III.1.31._Posterior) is corollary 6.7 proved by similar way in (Neapolitan, 2003, p. 347) to which I referred. Applying formulas [4.15](#_Formula_III.1.26._Probability) and [4.20](#_Formula_III.1.31._Posterior), it is easy to specify the posterior probability *P*(*Xi=*1*|PAij*, ) of variable *Xi* given its parent instance *PAij* as follows:

**Formula 4.21.** Posterior probability of variable *Xi* given its parent instance *PAij*

Where *Nij=aij+bij* and *Mij=sij+tij*.

It is easy to recognize that formula [4.21](#_Formula_III.1.32._Posterior) is an extension of formula [4.10](#_Formula_III.1.20._Posterior) in case of multi-node BN. In general, in case of binomial distribution, if we have the real/trust BN embedded in the expanded augmented network like figure [4.4](#_Figure_III.1.13._BN) 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|*) = . In other words, the CPT of each *Xi* is evolved based on evidences from *P*(*Xi=*1*|PAij*) = to *P*(*Xi=*1*|PAij*,) = . This evolution was mentioned for the previous case (see formula [4.10](#_Formula_III.1.20._Posterior)) in which there are one hypothesis variable *X* in BN. Hence, we conclude that learning parameters or the evolution of Bayesian overlay model (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) becomes very easy if we take advantages of beta density function. Now it is necessary to illustrate learning parameters by following example.

**Example of learning parameters 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 [4.4](#_Figure_III.1.13._BN). Evidence sample (evidence matrix) is shown in table [4.1](#_Table_III.1.5._Evidence) (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.** Evidence sample corresponding to 5 trials (sample of size 5)

In order to interpret evidence sample in table [4.1](#_Table_III.1.5._Evidence), 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 [4.2](#_Table_III.1.6._Posterior) shows counters *sij*, *tij* (s) and posterior density functions calculated based on these counters; please see formula [4.20](#_Formula_III.1.31._Posterior) for more details about updating posterior density functions. For instance, the number of rows (evidences) in table [4.1](#_Table_III.1.5._Evidence) such that *X*2=1 given *X*1=1 is 3, which causes *s*21 = 3 in table [4.2](#_Table_III.1.6._Posterior).

|  |  |
| --- | --- |
| *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 4.2.** 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 formula [4.21](#_Formula_III.1.32._Posterior). Table [4.3](#_Table_III.1.7._Updated) expresses such posterior conditional probabilities as evolutional CPT (s) of *X*1 and *X*2.

|  |
| --- |
|  |

###### **Table 4.3.** Updated CPT (s) of *X*1 and *X*2

Note that inverted probabilities in CPT (s) 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 [4.4](#_Figure_III.1.13._BN) is updated based on evidence sample and it is converted into the evolved BN with full of CPT (s) shown in figure [4.6](#_Figure_III.1.13._Updated) as follows:

P(*X*1=1) *P*(*X*1=0)

5/7 2/7

*X*1 *P*(*X*2=1)

1. 2/3

0 1/3

**(a)**

*β*(*F*11; 5, 2)

*β*(*F*22; 1, 2)

*β*(*F*21; 4, 2)

**(b)**

**Figure 4.6.** Updated version of BN (a) and augmented BN (b)

It is easy to perform learning parameters or the evolution of Bayesian by counting numbers *sij* and *tij* among sample according to expectation of beta density function as in formulas [4.10](#_Formula_III.1.20._Posterior) and [4.21](#_Formula_III.1.32._Posterior) but a problem occurs when data in sample is missing. This problem is solved by expectation maximization (EM) algorithm mentioned in next section [5](#_5._Learning_parameters).

## 5. Learning parameters in case of missing 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](#_Table_III.1.5._Evidence), in which there is the set of 5 evidences *=*{*X*(1)*, X*(2)*, X*(3)*, X*(4)*, X*(5)} along with network in figure [4.4](#_Figure_III.1.13._BN) but the evidences *X*(2) and *X*(5) have not data yet. Table [5.1](#_Table_III.1.8._Evidence) 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 5.1.** 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 [5.2](#_Table_III.1.9._New_1) 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 5.2.** 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 [4.4](#_Figure_III.1.13._BN).

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 [4.4](#_Figure_III.1.13._BN).

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 [5.3](#_Table_III.1.10._Complete).

|  |  |  |  |
| --- | --- | --- | --- |
|  | *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 5.3.** 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 [5.3](#_Table_III.1.10._Complete), we have table [5.4](#_Table_III.1.11._Counters) showing such ultimate results of E-step:

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

###### **Table 5.4.** 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 [5.4](#_Table_III.1.11._Counters)). Table [5.5](#_Table_III.1.12._Posterior) 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 5.5.** 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 [4.4](#_Figure_III.1.13._BN)) 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 expecations 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 [5.4](#_Table_III.1.11._Counters) 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 [5.5](#_Table_III.1.12._Posterior) 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 [5.5](#_Table_III.1.12._Posterior), 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 overlay model in figure [4.4](#_Figure_III.1.13._BN) is converted into the evolutional version specified in figure [5.1](#_Figure_III.1.16._Updated).

P(*X*1=1) *P*(*X*1=0)

0.75 0.25

*X*1 *P*(*X*2=1)

1. 0.61

0 0.50

**(a)**

*β*(*F*11; 9, 3)

*β*(*F*22; 2, 2)

*β*(*F*21; 73/12, 47/12)

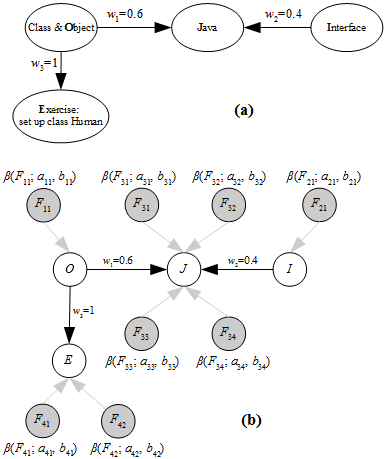
**(b)**

**Figure 5.1.** Updated version of BN (a) and augmented BN (b) in case of missing data

In general, parameter learning or evolution of Bayesian overlay model is described thoroughly in this section [5](#_III.1.3.2._Learning_parameters) and previous section [4](#_4._Learning_parameters). The next section [6](#_6._An_example) is a full example illustrating main aspects of parameter learning.

## 6. An example of learning parameters

Suppose a short Java course (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) is constituted of three concepts such as “Java”, “Class & Object” and “Interface” considered as knowledge variables (hypothesis variables) whose links are aggregation relationships. Additionally, there are an evidence variable: “*Exercise: set up class Human*”. The evidence “Exercise: set up class Human” proves whether or not she/he understands concept “Class & Object”. The number (in range 0…1) that measures the relative importance of each aggregation or evidence is defined as the weight of arc from parent node to child node. All weights concerning the child variable are normalized and used to build up its CPT implied by beta density function. It is logical to initialize weights by uniform distribution. Our work is to define prior density functions and enhance them based on evidences, namely, specifying appropriate posterior density functions. As known, this process is parameter evolution (evolution of Bayesian overlay model) or parameter learning described totally in previous sub-sections [5](#_III.1.3.2._Learning_parameters) and [4](#_III.1.3.1._Learning_parameters). Figure [6.1](#_Figure_III.1.16._BN) depicts network structure as weighted graph (a) and augmented BN (b) of Java course.



**Figure 6.1.** BN structure as weighted graph (a) and augmented BN (b) of Java course

Nodes *J*, *O*, *I* denote knowledge variables (hypothesis variables) “Java”, “Class & Object”, “Interface”, respectively. Node *E* denotes evidence variable “Exercise: set up class Human”. In this example, node *J* has two parents: *O* and *I* which in turn are corresponding to two weights of aggregation relationship: *w*1=0.6, *w*2=0.4. The default weight of diagnostic relationship between *E* and *O* is *w*3=1. Prior conditional probabilities (CPT (s)) of variables *J*, *O*, *I*, and *E* are specified based on these weights according to combination of BN and overlay model (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) with note that {*O*, *I*} is complete set of mutually exclusive variables and so three nodes *J*, *O* and *I* construct a graph of granularity hierarchy (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, pp. 287-288). For instance, the prior conditional probability of *J*=1 given *O*=1, and *I*=1 is:

Prior conditional probability of evidence *E* is specified as follows:

Shortly, it is easy to determine all CPT (s) of variables *J*, *O*, *I*, and *E* as shown in table [6.1](#_Table_III.1.13._All).

|  |  |  |  |
| --- | --- | --- | --- |
| **Real**  **Variable** | **Dummy**  **Variable** | **Density**  **Function** | **Prior**  **Probability** |
| *O* | *F*11 | *β*(*F*11; *a*11, *b*11) | *P*(*O*=1) = 1\*0.5 = 0.5= *a*11/(*a*11+*b*11)  *P*(*O*=0) = 1–*P*(*O*=1) = 0.5 |
| *I* | *F*21 | *β*(*F*21; *a*21, *b*21) | *P*(*I*=1) = *a*21/(*a*21+*b*21) = 1\*1 = 1  *P*(*I*=0) = 1–*P*(*I*=1) = 0 |
| *J* | *F*31 | *β*(*F*31; *a*31, *b31*) | *P*(*J*=1|*O*=1, *I*=1) = 1\*0.6+1\*0.4 = 1 = *a*31/(*a*31+*b*31)  *P*(*J*=0|*O*=1, *I*=1) = 1–*P*(*J*=1|*O*=1, *I*=1) = 0 |
| *J* | *F*32 | *β*(*F*32; *a*32, *b*32) | *P*(*J*=1|*O*=1, *I*=0) = 1\*0.6+0\*0.4 = 0.6 = *a*32/(*a*32+*b*32)  *P*(*J*=0|*O*=1, *I*=0) = 1–*P*(*J*=1|*O*=1, *I*=0) = 0.4 |
| *J* | *F*33 | *β*(*F*33; *a*33, *b*33) | *P*(*J*=1|*O*=0, *I*=1) = 0\*0.6+1\*0.4 = 0.4 = *a*33/(*a*33+*b*33)  *P*(*J*=0|*O*=0, *I*=1) = 1–*P*(*J*=1|*O*=0, *I*=1) = 0.6 |
| *J* | *F*34 | *β*(*F*34; *a*34, *b*34) | *P*(*J*=1|*O*=0, *I*=0) = 0\*0.6+0\*0.4 = 0 = *a*34/(*a*34+*b*34)  *P*(*J*=0|*O*=0, *I*=0) = 1–*P*(*J*=1|*O*=0, *I*=0) = 1 |
| *E* | *F*41 | *β*(*F*41; *a*41, *b*41) | *P*(*E*=1|*O*=1) = 1\*1 = 1 = *a*41/(*a*41+*b*41)  *P*(*E*=0|*O*=1) = 1–*P*(*E*=1|*O*=1) = 0 |
| *E* | *F*42 | *β*(*F*42; *a*42, *b*42) | *P*(*E*=1|*O*=0) = 0\*1 = 0 = *a*42/(*a*42+*b*42)  *P*(*E*=0|*O*=0) = 1–*P*(*E*=1|*O*=0) = 1 |

###### **Table 6.1.** All variables and their density functions, prior probabilities

That *P*(*O*=1) equals 0.5 and *P*(*I*=1) equals 0.5is due to uniform distribution. Now it is necessary to determine prior beta density functions *β*(*Fij*; *aij*, *bij*), which leads to specify parameters *aij* and *bij*. In fact, it is very easy to specify *aij* and *bij* (s) by taking advantages of theorem of “equivalent sample size” when prior conditional probabilities (CPT (s)) are known as in table [6.1](#_Table_III.1.13._All). So, we should glance over the concept “equivalent sample size” (Neapolitan, 2003, p. 351) in BN. Suppose there is the BN and its parameters in full *β*(*Fij*; *aij*, *bij*), for all *i* and *j*, if there exists the number *N* such that satisfying formula [6.1](#_Formula_III.1.33._Equivalent) then, the augmented BN is called to have *equivalent sample size N*.

**Formula 6.1.** Definition of equivalent sample size *N*

Where *P*(*PAij*) is probability of the *jth* parent instance of an *Xi* and it is conventional that if *Xi* has no parent then, *P*(*PAi*1)=1. Formula [6.1](#_Formula_III.1.33._Definition) is equation 6.13 in (Neapolitan, 2003, p. 351). For example, reviewing BN in figure [4.4](#_Figure_III.1.13._BN), given *β*(*F*11; 2, 2), *β*(*F*21; 1,1)*, β*(*F*22; 1,1), we have:

4 = *a*11 + *b*11 = 1\*4 = 4 (*P*(*PA*11)=1 because *X*1 has no parent)

2 = *a*21 + *b*21 = *P*(*X*1=1) \*4 = ½\*4 = 2

2 = *a*22 + *b*22 = *P*(*X*1=0) \*4 = ½\*4 = 2

So, this network has equivalent sample size 4. The theorem of “equivalent sample size”is stated that if there exists the number *N* so that all parameters *aij* and *bij* (s) are satisfied formula [6.2](#_Formula_III.1.34._Theorem) then, the augmented BN has equivalent sample size *N*.

**Formula 6.2.** Theorem of equivalent sample size *N*

It is easy to prove this theorem, we have:

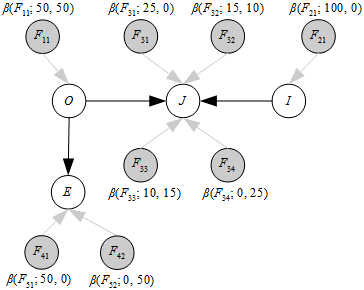
According to definition of equivalent sample size *N*, it implies that the augmented BN has equivalent sample size *N*. The theorem of equivalent sample size is described particularly in (Neapolitan, 2003, p. 353) as theorem 6.14.

Going back Java course example specified in figure [6.1](#_Figure_III.1.16._BN), we suppose that the augmented BN has equivalent sample size *N* = 100. Applying the theorem of equivalent sample size specified in formula [6.2](#_Formula_III.1.34._Theorem), all prior parameters *aij* and *bij* (s) are calculated as in table [6.2](#_Table_III.1.14._All) as follows:

|  |  |
| --- | --- |
| **Density**  **Functions** | **Parameters** |
| *β*(*F*11; *a*11, *b*11) | *a*11 = *P*(*O*=1)\*1\*10 = 0.5\*100 = **50**  *b*11 = *P*(*O*=0)\*1\*10 = 0.5\*100 = **50** |
| *β*(*F*21; *a*21, *b*21) | *a*21 = *P*(*I*=1)\*1\*10 = 1\*100 = **100**  *b*21 = *P*(*I*=0)\*1\*10 = 0\*100 = **0** |
| *β*(*F*31; *a*31, *b*31) | *a*31 = *P*(*J*=1|*O*=1, *I*=1)\**P*(*O*=1, *I*=1)\*100  = *P*(*J*=1|*O*=1, *I*=1)\**P*(*O*=1)\**P*(*I*=1)\*100  = 1\*0.5\*0.5\*100 = **25**  *b*31= *P*(*J*=0|*O*=1, *I*=1)\**P*(*O*=1, *I*=1)\*100  = *P*(*J*=0|*O*=1, *I*=1)\**P*(*O*=1)\**P*(*I*=1)\*100  = 0\*0.5\*0.5\*100 = **0** |
| *β*(*F*32; *a*32, *b*32) | *a*32 = *P*(*J*=1|*O*=1, *I*=0)\**P*(*O*=1, *I*=0)\*100  = *P*(*J*=1|*O*=1, *I*=0)\**P*(*O*=1)\**P*(*I*=0)\*100  = 0.6\*0.5\*0.5\*100 = **15**  *b*32 = *P*(*J*=0|*O*=1, *I*=0)*\*P*(*O*=1, *I*=0)\*100  = *P*(*J*=0|*O*=1, *I*=0)*\*P*(*O*=1)\**P*(*I*=0)\*100  = 0.4\*0.5\*0.5\*100 = **10** |
| *β*(*F*33; *a*33, *b*33) | *a*33 = *P*(*J*=1|*O*=0, *I*=1)\**P*(*O*=0, *I*=1)\*100  = *P*(*J*=1|*O*=0, *I*=1)\**P*(*O*=0)\**P*(*I*=1)\*100  = 0.4\*0.5\*0.5\*100 = **10**  *b*33 = *P*(*J*=0|*O*=0, *I*=1)\**P*(*O*=0, *I*=1)\*100  = *P*(*J*=0|*O*=0, *I*=1)\**P*(*O*=0)\**P*(*I*=1)\*100  = 0.6\*0.5\*0.5\*100 = **15** |
| *β*(*F*34; *a*34, *b*34) | *a*34 = *P*(*J*=1|*O*=0, *I*=0)\**P*(*O*=0, *I*=0)\*100  = *P*(*J*=1|*O*=0, *I*=0)\**P*(*O*=0)\**P*(*I*=0)\*100  = 0\*0.5\*0.5\*100 = **0**  *b*34 = *P*(*J*=0|*O*=0, *I*=0)\**P*(*O*=0, *I*=0)\*100  = *P*(*J*=0|*O*=0, *I*=0)\**P*(*O*=0)\**P*(*I*=0)\*100  = 1\*0.5\*0.5\*100 = **25** |
| *β*(*F*41; *a*41, *b*41) | *a*41 = *P*(*E*=1|*O*=1)\**P*(*O*=1)\*100  = 1\*0.5\*100 = **50**  *b*41 = *P*(*E*=0|*O*=1)\**P*(*O*=1)\*100  = 0\*0.5\*100 = **0** |
| *β*(*F*42; *a*42, *b*42) | *a*42 = *P*(*E*=1|*O*=0)\**P*(*O*=0)\*100  =0\*0.5\*100 = **0**  *b*42 = *P*(*E=0|O=0*)*\*P*(*O=0*)\*100  = 1\*0.5\*100 = **50** |

###### **Table 6.2.** All parameters of prior density functions

Note that prior conditional probabilities *P*(*Xi|PAij*) are shown in table [6.1](#_Table_III.1.13._All). The augmented BN with parameters in full is shown in figure [6.2](#_Figure_III.1.18._Augmented).



**Figure 6.2.** Augmented BN with initial parameters in full

When prior CPT (s) (see table [6.1](#_Table_III.1.13._All)) and prior beta density functions (see table [6.2](#_Table_III.1.14._All)) are specified, the example Java course in this section [6](#_III.1.3.3._An_example) illustrates the evolution of CPT (s) which is essentially updating prior parameters *aij* and *bij* (s) based on evidences as aforementioned in previous sections [4](#_III.1.3.1._Learning_parameters) and [5](#_III.1.3.2._Learning_parameters). Suppose there is an observation: User does well the exercise “*Set up class Human*”. Therefore, we have an evidence *D* = (*E*=1). Because it lacks information about other concepts such as *O*, *J*, *I*, this is case of missing data, aforementioned in previous section [5](#_III.1.3.2._Learning_parameters). Table [6.3](#_Table_III.1.15._Incomplete) shows incomplete sample with the evidence *D* = (*E*=1), in which question mark (?) denotes missing values of variables *O*, *I*, and *J*. EM algorithm mentioned in previous section [5](#_III.1.3.2._Learning_parameters) will be used to estimate these missing values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *O* | *I* | *J* | *E* |
| *D* | ? | ? | ? | 1 |

###### **Table 6.3.** Incomplete sample with evidence *D* = (*E*=1)

Note that missing values (?) are binary and so the evidence *D* is split into many *D*’ (s) according two possible values 0 and 1 of missing values (?). Table [6.4](#_Table_III.1.16._New) shows new split evidences for missing values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *O* | *I* | *J* | *E* | #Occurrences |
| *D*’ | 1 | 1 | 1 | 1 | #*n*1 |
| *D*’ | 1 | 1 | 0 | 1 | #*n*2 |
| *D*’ | 1 | 0 | 1 | 1 | #*n*3 |
| *D*’ | 1 | 0 | 0 | 1 | #*n*4 |
| *D*’ | 0 | 1 | 1 | 1 | #*n*5 |
| *D*’ | 0 | 1 | 0 | 1 | #*n*6 |
| *D*’ | 0 | 0 | 1 | 1 | #*n*7 |
| *D*’ | 0 | 0 | 0 | 1 | #*n*8 |

###### **Table 6.4.** New split evidences for missing values of *O*, *I*, and *J*

Where *#ni* are numbers of possible combinations (occurrences) of binary variables *O*, *I*, and *J*; please see tables [5.2](#_Table_III.1.9._New_1) and [5.3](#_Table_III.1.10._Complete) for knowing more about *#ni*.

It is required to estimate *#ni*, which is the most important task in E-step of EM algorithm. For instance, the *#n*1 is estimated by the probability of *O*=1, *I*=1, and *J*=1 given *E*=1. We have:

(by applying formula [2.8](#_Formula_III.1.2’._Reduced) into joint probability distribution *P*(*O*, *I*, *J*, *E*, *Q*))

(prior probabilities *P*(.) are specified in table [6.1](#_Table_III.1.13._All))

(due to Bayes’ rule specified in formula [2.1](#_Formula_III.1.1._Bayes’))

(by applying formula [2.8](#_Formula_III.1.2’._Reduced) into joint probability distribution *P*(*O*, *I*, *J*, *E*, *Q*))

Similarly, we have:

When occurrence numbers *#ni* (s) are determined, missing data is filled fully and evidence sample shown in table [6.4](#_Table_III.1.16._New) is completed as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *O* | *I* | *J* | *E* | #Occurrences |
| *D*’ | 1 | 1 | 1 | 1 | #*n*1=1 |
| *D*’ | 1 | 1 | 0 | 1 | #*n*2=0 |
| *D*’ | 1 | 0 | 1 | 1 | #*n*3=0 |
| *D*’ | 1 | 0 | 0 | 1 | #*n*4=0 |
| *D*’ | 0 | 1 | 1 | 1 | #*n*5=0 |
| *D*’ | 0 | 1 | 0 | 1 | #*n*6=0 |
| *D*’ | 0 | 0 | 1 | 1 | #*n*7=0 |
| *D*’ | 0 | 0 | 0 | 1 | #*n*8=0 |

###### **Table 6.5.** Complete sample with evidence *D* = (*E*=1)

When missing values are estimated as in table [6.5](#_Table_III.1.17._Complete), it is easy to calculate counters *sij* and *tij* (s) which are ultimate results from E-step of EM algorithm.

* The counter *s*11 (*t*11) is the number of evidences such that *O*=1 (*O*=0), which corresponds to variable *F*11.
* The counter *s*21 (*t*11) is the number of evidences such that *I*=1 (*I*=0), which corresponds to variable *F*21.
* The counter *s*31 (*t*31) is the number of evidences such that *J*=1 (*J*=0) given *O*=1 and *I*=1, which corresponds to variable *F*31.
* The counter *s*32 (*t*32) is the number of evidences such that *J*=1 (*J*=0) given *O*=1 and *I*=0, which corresponds to variable *F*32.
* The counter *s*33 (*t*33) is the number of evidences such that *J*=1 (*J*=0) given *O*=0 and *I*=1, which corresponds to variable *F*33.
* The counter *s*34 (*t*34) is the number of evidences such that *J*=1 (*J*=0) given *O*=0 and *I*=0, which corresponds to variable *F*34.
* The counter *s*41 (*t*41) is the number of evidences such that *E*=1 (*E*=0) given *O*=1, which corresponds to variable *F*41.
* The counter *s*42 (*t*42) is the number of evidences such that *E*=1 (*E*=0) given *O*=0, which corresponds to variable *F*42.

From complete sample in table [6.5](#_Table_III.1.17._Complete), we have:

|  |  |
| --- | --- |
| *s*11=#*n*1=1 | *t*11=0 |
| *s*21=#*n*1=1 | *t*21=0 |
| *s*31=#*n*1=1 | *t*31=0 |
| *s*32=0 | *t*32=0 |
| *s*33=0 | *t*33=0 |
| *s*34=0 | *t*34=0 |
| *s*41=#*n*1=1 | *t*41=0 |
| *s*42=0 | *t*42=0 |

###### **Table 6.6.** Counters *sij* and *tij* (s) from estimated values

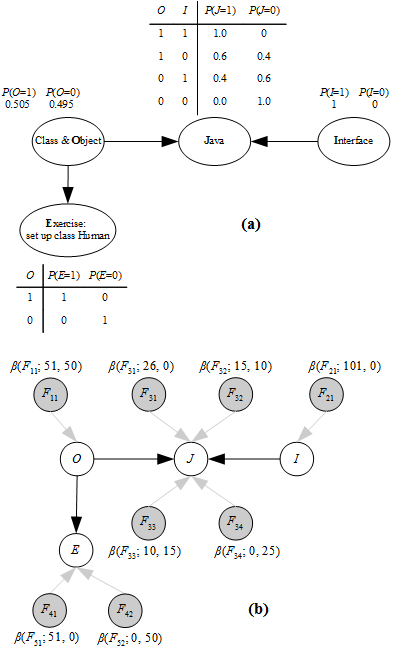
The next step of EM algorithm, M-step is responsible for updating posterior density functions *β*(*Fij|*) (s), which leads to update posterior probabilities *P*(*O*=1*|*), *P*(*I*=1*|*), *P*(*J*=1|*O*=1,*I*=1,), *P*(*J*=1|*O*=1,*I*=0,), *P*(*J*=1|*O*=0,*I*=1,), *P*(*J*=1|*O*=0,*I*=0,), *P*(*E*=1|*O*=1,), *P*(*E*=1|*O*=0,), based on current counters *sij* and *tij* from complete evidence sample (table [6.6](#_Table_III.1.18._Counters)). Table [6.7](#_Table_III.1.19._Posterior) shows results of M-step which are these posterior density functions and posterior probabilities.

|  |
| --- |
| *β*(*F*11*|*) = *β*(*F*11; *a*11+*s*11, *b*11+*t*11) = *β*(*F*11; 50+1, 50+0) = *β*(*F*11; 51, 50)  *β*(*F*21*|*) = *β*(*F*21; *a*21+*s*21, *b*21+*t*21) = *β*(*F*21; 100+1, 0+0) = *β*(*F*21; 101, 0)  *β*(*F*31*|*) = *β*(*F*31; *a*31+*s*31, *b*31+*t*31) = *β*(*F*31; 25+1, 0+0) = *β*(*F*31; 26, 0)  *β*(*F*32*|*) = *β*(*F*32; *a*32+*s*32, *b*32+*t*32) = *β*(*F*32; 15+0, 10+0) = *β*(*F*32; 15, 10)  *β*(*F*33*|*) = *β*(*F*33; *a*33+*s*33, *b*33+*t*33) = *β*(*F*33; 10+0, 15+0) = *β*(*F*33; 10, 15)  *β*(*F*34*|*) = *β*(*F*34; *a*34+*s*34, *b*34+*t*34) = *β*(*F*34; 0+0, 25+0) = *β*(*F*34; 0, 25)  *β*(*F*41*|*) = *β*(*F*41; *a*41+*s*41, *b*41+*t*41) = *β*(*F*41; 50+1, 0+0) = *β*(*F*41; 51, 0)  *β*(*F*42*|*) = *β*(*F*42; *a*42+*s*42, *b*42+*t*42) = *β*(*F*42; 0+0, 50+0) = *β*(*F*42; 0, 50)  *P*(*O*=1*|*) = *E*(*F*11*|*) = 51/(51+50) 0.505  *P*(*I*=1*|*) = *E*(*F*21*|*) = 101/(101+0) = 1  *P*(*J*=1|*O*=1,*I*=1,) = *E*(*F*31*|*) = 26/(26+0) = 1  *P*(*J*=1|*O*=1,*I*=0,) = *E*(*F*32*|*) = 15/(15+10) = 0.6  *P*(*J*=1|*O*=0,*I*=1,) = *E*(*F*33*|*) = 10/(10+15) = 0.4  *P*(*J*=1|*O*=0,*I*=0,) = *E*(*F*34*|*) = 0/(0+25) = 0  *P*(*E*=1|*O*=1,) = *E*(*F*41*|*) = 51/(51+0) = 1  *P*(*E*=1|*O*=0,) = *E*(*F*42*|*) = 0/(0+50) = 0  *P*(*O*=0*|*) = 1–*P*(*O*=1*|*) 0.495  *P*(*I*=0*|*) = 1–*P*(*I*=1*|*) = 0  *P*(*J*=0|*O*=1,*I*=1,) = 1–*P*(*J*=1|*O*=1,*I*=1,) = 0  *P*(*J*=0|*O*=1,*I*=0,) = 1–*P*(*J*=1|*O*=1,*I*=0,) = 0.4  *P*(*J*=0|*O*=0,*I*=1,) = 1–*P*(*J*=1|*O*=0,*I*=1,) = 0.6  *P*(*J*=0|*O*=0,*I*=0,) = 1–*P*(*J*=1|*O*=0,*I*=0,) = 1  *P*(*E*=0|*O*=1,) = 1–*P*(*E*=1|*O*=1,) = 0  *P*(*E*=0|*O*=0,) = 1–*P*(*E*=1|*O*=0,) = 1 |

###### **Table 6.7.** Posterior density functions and posterior probabilities are evolved based on counters *sij* and *tij*

Note that origin parameters such as *a*11=50, *b*11=50, *a*21=100, *b*21=0, *a*31=25, *b*31=0, *a*32=15, *b*32=10, *a*33=10, *b*33=15, *a*34=0, *b*34=25, *a*41=50, *b*41=0, *a*42=0, and *b*42=50 (see figure [6.2](#_Figure_III.1.18._Augmented)) are kept intact in the task of updating posterior density functions *β*(*F*11|), *β*(*F*21|), *β*(*F*31|), *β*(*F*32|), *β*(*F*33|), *β*(*F*34|), *β*(*F*41|), *β*(*F*42|), *β*(*F*51|), and *β*(*F*52|). For example, *β*(*F*11*|*) = *β*(*F*11; *a*11+*s*11, *b*11+*t*11) = *β*(*F*11; 50+1, 50+1) = *β*(*F*11; 51, 51). After the updating task, these parameters are changed into new values; concretely, *a*11=51, *b*11=50, *a*21=101, *b*21=0, *a*31=26, *b*31=0, *a*41=51, and *b*41=0. These parameters updated with new values, which are called posterior parameters, are in turn used for the new iteration of EM algorithm. Please pay attention that such posterior parameters *aij* and *bij* (s) are calculated based on counters *sij* and *tij* (s).

By posterior CPT (s) shown in table [6.7](#_Table_III.1.19._Posterior) which is the ultimate result of EM algorithm, the Bayesian overlay model of Java course in figure [6.2](#_Figure_III.1.18._Augmented) is converted into the evolutional version specified in figure [6.3](#_Figure_III.1.19._Updated).



**Figure 6.3.** Evolutional version of BN (a) and augmented BN (b) for Java course

It is possible to run more iterations for EM algorithm so that the posterior density functions are updated and become more accurate after many iterations because the limit will gains certain value; please see previous section [5](#_III.1.3.2._Learning_parameters). In general, this Java course example is an extension of example in previous section [5](#_III.1.3.2._Learning_parameters), which help us to know clearly combination of Bayesian network and overlay model (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009) so as to construct Bayesian overlay (knowledge) sub-model and applying EM algorithm into making evolution of Bayesian overlay model in case of missing data.

## 7. Conclusion

In general, BN is a powerful mathematical tool for reasoning but it is restricted by unimproved initial parameters. This research focuses on the approach to parameter evolution that uses the EM algorithm for beta functions (Neapolitan, 2003, pp. 293-373). Note that particular features of beta function make this suggestion feasible because it is possible to compute the expectation of beta function which is the conditional probability in BN. Whether the EM converges quickly or not depends on how to pre-define the parameters. So, I specify the initial parameters (*aij, bij*) by weights of arcs.

However, the qualitative model (graph structure) is now fixed. It is more creative to apply machine learning algorithms to enhance entirely the structure of BN. That is learning structure process which will be represented in next researches.

## 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.

Cheng, J., Bell, D. A., & Liu, W. (1997, Januray 1). Learning Belief Networks from Data: An information theory based approach. (F. Golshani, K. Makki, C. Nicholas, & N. Pissinou, Eds.) *Proceedings of the sixth international conference on Information and knowledge management (CIKM '97)*, 325-331.

Conati, C., Gertner, A., & Vanlehn, K. (2002, November 1). Using Bayesian Networks to Manage Uncertainty in Student Modeling. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction, 12*(4), 371-417.

Henze, N. (2000). *Adaptive Hyperbooks: Adaptation for Project-Based Learning Resources.* PhD Thesis, University of Hannover.

Martin, J., & VanLehn, K. (1995, June). Student assessment using Bayesian nets. (B. R. Gaines, D. Heckerman, A. Mamdani, & M. P. Wellman, Eds.) *International Journal of Human-Computer Studies - Special issue: real-world applications of uncertain reasoning, 42*(6), 575-591.

Mayo, M. J. (2001). *Bayesian Student Modelling and Decision-Theoretic Selection of Tutorial Actions in Intelligent Tutoring Systems.* PhD Thesis, University of Canterbury, New Zealand.

Millán, E., & Pérez-de-la-Cruz, J. L. (2002, June). A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction, 12*(2-3), 281-330. doi:10.1023/A:1015027822614

Millán, E., Loboda, T., & Pérez-de-la-Cruz, J. L. (2010, July 29). Bayesian networks for student model engineering. (R. S. Heller, J. D. Underwood, & C.-C. Tsai, Eds.) *Computers & Education, 55*(4), 1663-1683. doi:10.1016/j.compedu.2010.07.010

Mitrovic, A. (1998, March 1). Learning SQL with a computerized tutor. (D. Joyce, & J. Impagliazzo, Eds.) *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education (SIGCSE '98), 30*(1), 307-311.

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.

Nguyen, L. (2009). ZEBRA: A new User Modeling System for Triangular Model of Learners' Characteristics. In G. D. Magoulas, P. Charlton, D. Laurillard, K. Papanikolaou, & M. Grigoriadou (Ed.), *AIED 2009: 14th conference on Artificial Intelligence in Education, Proceedings of the Workshop on “Enabling creative learning design: how HCI, User Modeling and Human Factors Help”* (pp. 42-51). Brighton, United Kingdom: IOS Press Amsterdam, The Netherlands, The Netherlands. Retrieved from https://goo.gl/cVzC6h

Nguyen, L. (2013). *Overview of Bayesian Network.* University of Technology, Ho Chi Minh city, Vietnam. Warri, Delta State, Nigeria: Science Journal Publication. doi:10.7237/sjms/105

Nguyen, L., & Do, P. (2009). Combination of Bayesian Network and Overlay Model in User Modeling. (M. E. Auer, Ed.) *International Journal of Emerging Technologies in Learning (iJET), 4*(4), 41-45. doi:10.3991/ijet.v4i4.684

Oracle. (n.d.). *Java language*. (Oracle Corporation) Retrieved December 25, 2014, from Java website: https://www.oracle.com/java

Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems* (3rd Edition ed.). New York, USA: McGraw-Hill Higher Education.

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

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

Wikipedia. (2014, September 3). *Mutual information*. (Wikimedia Foundation) Retrieved September 16, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Mutual\_information

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)