

# **A User Modeling System for Adaptive Learning**

First edition by Loc Nguyen

Email: [ng\\_phloc@yahoo.com](mailto:ng_phloc@yahoo.com)

Website: <https://sites.google.com/site/ngphloc>

Phone: +84-97-5250362

Research final version: 3.3 build 2009.11.21

Research advanced version: 4.3 build 2015.01.02

The book is originated from a research work submitted for the Degree of Doctor of Philosophy in Computer Science.

The book shows my gratitude to the sponsor Prof. Dr. Dong, Thuy T. B.

Vietnamese research title is “Hệ thống mô hình hóa người học hỗ trợ học thích nghi trong đào tạo từ xa”

## **Acknowledgement**

I would like to show my gratitude to all those who gave me the possibility to complete this thesis. I am grateful to Prof. Dr. Dong, Thuy T. B. for providing invaluable help and encouragement in my work. If it were not for your help, I would not complete the book in time. I have furthermore to thank the Department of Information System – Faculty of Information Technology – University of Science for conveniences during the time of research.

I also want to express my warmest thanks for my family and friends who have supported me in difficult times...

## Acronyms and Notations

$<<(>>)$	Much smaller (larger) than. For example, number 0 is much smaller than number 1000, we have $0 << 1000$ and $1000 >> 0$
$\overline{a,b}$	Integer range from $a$ to $b$ , for example, the expression $i = \overline{1,5}$ means that variable $i$ ranges from 1 to 5 and so, we have $i \in \{1,2,3,4,5\}$
BN	Bayesian Network
BNE	Belief Network Engine
CAT	Computerized Adaptive Testing
CI	Communication Interfaces
CPT	Conditional Probability Table
DAG	Directed Acyclic Graph
$D, \mathcal{D}$	$D$ and $\mathcal{D}$ are used to denote evidence, evidences, evidence sample, data sample, sample, training data, training set, training dataset and corpus
DBN	Dynamic Bayesian Network
$dissim$	Dissimilarity measure
$e$	Euler's number $e \approx 2.71828$
$E(x)$	Probability expectation, concretely, given probability density function $f(x)$ , we have $E(x) = \int_{-\infty}^{+\infty} xf(x)dx$
$e^x, exp(x)$	Exponent function
$f'(x)$	
$d(f(x))$	
$\frac{df(x)}{dx}$	First-order derivative of function $f$ with regard to variable $x$ . Especially, the notation $d(f(x))$ also denotes differential of $f(x)$
$\frac{d}{dx}f(x)$	
$f''(x)$	Second-order derivative of function $f$ with regard to variable $x$
$idf$	inverse document frequency
itemset	Item set
Java	An object-oriented programming language <a href="https://www.oracle.com/java">https://www.oracle.com/java</a>
$L(\theta)$	Likelihood function
$log(x)$	Logarithm function with any base
$log_2(x)$	Logarithm function with base 2

$Li_2(x)$	Dilogarithm function, $Li_2(x) = \sum_{k=1}^{+\infty} \frac{x^k}{k^2} = - \int_0^x \frac{\ln(1-t)}{t} dt$
$LnL(\theta)$	Log-likelihood function
$ln(x)$	Natural logarithm function
litemset	large itemset (frequent itemset)
ME	Mining Engine
MLE	Maximum Likelihood Estimation
PDF	Probability Density Function
$P(.)$	Probability
$sim$	Similarity measure
$tf$	Term frequency
TLM	Triangular Learner Model
UMS	User Modeling Shell/System/Server/Service
Zebra	User modeling system Zebra
.	Dot product or scalar product of two vectors, for example, $(x_1, y_1) \cdot (x_2, y_2) = x_1 * x_2 + y_1 * y_2$
$\beta(x; a, b)$	Beta distribution with two parameters $a, b$
$beta(x; a, b)$	$\beta(x; a, b) = beta(x; a, b) = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}$ Where $\Gamma(.)$ is gamma function
$\gamma$	Euler-Mascheroni constant $\gamma \approx 0.577215$
	Adaptation criterion for evaluating adaptive learning model
$\Gamma(x)$	Gamma function $\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$
$\psi, \psi_1$	Digamma function, trigamma function $\psi(x) = d(\ln(\Gamma(x))) = \frac{\Gamma'(x)}{\Gamma(x)} = \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}} \right) dt$ $\psi_1(x) = \psi'(x) = \int_0^{+\infty} \frac{te^{-xt}}{1-e^{-t}} dt$ Where $\Gamma(.)$ is gamma function
$\frac{\partial f}{\partial x}$	First-order partial derivative of multi-variable function $f$ with regard to variable $x$
$\frac{\partial^2 f}{\partial x^2}$	Second-order partial derivatives of multi-variable function $f$

$\frac{\partial^2 f}{\partial x \partial y}$	
$\nabla f$	Gradient vector of function $f$ , whose components are first-order derivatives of function $f$ . Given function $f$ having $n$ variables $x_1, x_2, \dots, x_n$ then $\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$
$\approx$	Approximation sign, for example: $0.99 \approx 1$
$\int_a^b f(x) dx$	Integral of function $f(x)$ in intervals $[a, b]$ , $[a, b)$ , $(a, b]$ , $(a, b)$

## Preface

Nowadays modern society requires every citizen always updates and improves her/his knowledge and skills necessary to working and researching. E-learning or distance learning gives everyone a chance to study at anytime and anywhere with full support of computer technology and network. Adaptive learning, a variant of e-learning, aims to satisfy the demand of personalization in learning. The adaptive learning system (ALS) is defined as the computer system that has ability to change its action to provide learning content and pedagogic environment/method for every student in accordance with her/his individual characteristics. Therefore, the ultimate goal of this research is to give the best support to learners in their learning path and this is an enthusiastic contribution to research community. Learners' information and characteristics such as knowledge, goal, experience, interest, background are the most important to adaptive system. These characteristics are organized in structure so-called learner model (or user model) and the system or computer software that builds up and manipulates learner model is called user modeling system (or learner modeling system).

This research proposes a learner model that consists of three essential kinds of information about learners such as knowledge, learning style and learning history. Such three characteristics form a triangle and so this learner model is called Triangular Learner Model (TLM). The ideology of TLM is that user characteristics are various and only some information is really necessary to adaptive learning and an optimal user modeling system should choose essential information relating to user's study to build up learner model. According to this ideology, TLM will cover the whole of user's information required by learning adaptation process and give the best support to adaptive learning. The reasons for such assertion are:

- Knowledge, learning styles and learning history are prerequisite for modeling learner.
- While learning history and knowledge change themselves frequently, learning styles are relatively stable. The combination of them ensures the integrity of information about learner.
- User knowledge is domain specific information and learning styles are personal traits. The combination of them supports user modeling system to take full advantages of both domain specific information and domain independent information in user model.

These reasons also strong points of TLM because they reflect the sufficiency and solid of an optimal learner model. Moreover, TLM emphasizes on the inference mechanism by applying Bayesian network and Markov model into modeling user knowledge and learning style. Intelligent deduction is the best feature of TLM instead of providing user information only as normal user modeling system.

I propose an information system that manages TLM. This system is called the user modeling system Zebra. Zebra has two built-in engines: belief network

engine and mining engine. Belief network engine manipulates knowledge and learning styles. Mining engine manipulates learning history.

Although the research focuses on describing TLM and Zebra with regard to user modeling and adaptive learning, *the research is organized as a book* that includes five chapters:

1. Chapter I is a survey of user model, user modeling, and adaptive learning.
2. Chapter II introduces the general architecture of the proposed user modeling system Zebra and TLM.
3. Chapter III describes TLM in detailed. This is the most important chapter.
4. Chapter IV gives some approaches to evaluate TLM and Zebra.
5. Chapter V summarizes the research and discusses future trend of Zebra.

Please pay attention to chapters II and III. Chapter II gives essential aspects of TLM and Zebra according to general viewpoint. Chapter III is the heaviest one filled with a lot of knowledge. It is easy for you to understand TLM and Zebra if you study sub-sections of chapter III cautiously. Zebra is implemented as computer software available at internet link:

<https://sites.google.com/site/ngphloc/st/dissertations/zebra>

In general, the first difference of the book is to introduce my innovative works in user modeling, adaptive learning, data mining, machine learning, and probability. The second difference is that there is computer software Zebra associated to the book.

Have a nice day! Please enjoy the book!

An Giang – Vietnam, January 02, 2015

Loc Nguyen

# Contents

<b>Chapter I. User Model and Adaptive Learning .....</b>	<b>1</b>
<b>I.1. User Model.....</b>	<b>1</b>
I.1.1. Information in user model .....	3
I.1.1.1. Domain specific information .....	3
I.1.1.2. Domain independent information.....	4
I.1.2. Classification of user models.....	5
I.1.2.1. Stereotype model.....	6
I.1.2.2. Overlay model.....	7
I.1.2.3. Differential model .....	8
I.1.2.4. Perturbation model .....	9
I.1.2.5. Plan model .....	9
<b>I.2. Adaptive Learning.....</b>	<b>11</b>
I.2.1. Classification of adaptive learning systems.....	11
I.2.2. Intelligent Tutoring System (ITS).....	13
I.2.2.1. Architecture of ITS .....	14
I.2.2.2. Characteristics of pedagogical module.....	15
I.2.2.3. An example of ITS: ANATOM-TUTOR.....	16
I.2.3. Adaptive Educational Hypermedia System (AEHS) .....	17
I.2.3.1. Architecture of AEHS .....	19
I.2.3.2. Characteristics of AEHS .....	21
I.2.3.3. An example of AEHS: Adaptive Hypermedia for All (AHA!) .....	22
I.2.4. Evaluation of existing ITS and AEHS .....	25
<b>I.3. Bayesian network user model .....</b>	<b>26</b>
I.3.1. KBS hyperbook system.....	26
I.3.1.1. Yet Another Clustering Formalism (YACF).....	29
I.3.1.2. How to define CPT (s) of cluster node and child nodes .....	30
I.3.2. Andes .....	32
I.3.2.1. Solution graph.....	33
I.3.2.2. Bayesian network in Andes .....	35
I.3.3. SQL-Tutor and constraint-based modeling .....	40
I.3.4. Data-centric approach .....	45
<b>Chapter II. Zebra: A User Modeling System for Triangular Learner Model .....</b>	<b>47</b>
<b>II.1. Existing user modeling systems .....</b>	<b>47</b>
II.1.1. Early user modeling systems .....	48
II.1.2. User modeling shells.....	48
II.1.3. User modeling servers .....	50
<b>II.2. Zebra: A User Modeling System for Triangular Learner Model ...</b>	<b>56</b>
II.2.1. Triangular Learner Model .....	57
II.2.2. The architecture of Zebra .....	59
II.2.3. Interaction between Zebra and adaptive applications .....	62
II.2.4. Implementation of Zebra.....	65
<b>II.3. Conclusion .....</b>	<b>91</b>
<b>Chapter III. Three sub-models in Triangular Learner Model .....</b>	<b>92</b>
<b>III.1. Knowledge sub-model .....</b>	<b>92</b>
III.1.1. Combination of Bayesian network and overlay model in building up knowledge sub-model .....	93
III.1.1.1. Bayesian network .....	94
III.1.1.2. Applying Bayesian network to overlay model.....	101

III.1.1.3. SIGMA-gate inference .....	106
III.1.1.4. Evaluation .....	113
III.1.2. Incorporate Bayesian inference into adaptation rules .....	114
III.1.3. Evolution of Bayesian overlay model .....	120
III.1.3.1. Learning parameters in Bayesian model .....	121
III.1.3.2. Learning parameters in case of data missing .....	137
III.1.3.3. An example of learning parameters .....	141
III.1.4. Improving knowledge sub-model by using dynamic Bayesian network .....	153
III.1.4.1. Dynamic Bayesian network .....	154
III.1.4.2. Using dynamic Bayesian network to model user's knowledge .....	156
III.1.4.3. Evaluation .....	171
III.1.5. Specifying prior probabilities .....	172
III.1.5.1. Maximum likelihood estimation .....	174
III.1.5.2. Beta likelihood estimation .....	177
III.1.5.3. Algorithm to solve the equations whose solutions are parameter estimators .....	188
III.1.5.4. An example of how to specify prior probabilities .....	190
III.1.5.5. New version of the equations whose solutions are parameter estimators .....	195
III.1.5.6. Evaluation .....	201
<b>III.2. Learning style sub-model .....</b>	<b>203</b>
III.2.1. What learning styles are .....	203
III.2.2. Learning style families .....	204
III.2.2.1. Constitutionally based learning styles and preferences .....	204
III.2.2.2. The cognitive structure .....	205
III.2.2.3. Stable personal type .....	206
III.2.2.4. Flexible stable learning preference .....	207
III.2.3. Providing adaptation of learning materials to learning styles .....	210
III.2.4. Hidden Markov model .....	212
III.2.5. Applying hidden Markov model into building up learning style sub-model ....	215
III.2.6. Evaluation .....	220
<b>III.3. Learning history sub-model.....</b>	<b>222</b>
III.3.1. Learning concept recommendation based on mining learning history .....	223
III.3.1.1. Approaches of sequential pattern mining .....	226
III.3.1.1.1. Candidate generation-and-test approach .....	226
III.3.1.1.2. Pattern-growth approach .....	230
III.3.1.2. A proposal of breaking sequential pattern in learning context .....	231
III.3.1.3. Evaluation .....	233
III.3.2. Discovering user interests by document classification .....	233
III.3.2.1. Vector model for representing documents .....	235
III.3.2.2. Methods of document classification .....	236
III.3.2.2.1. Document classification based on support vector machine .....	236
III.3.2.2.2. Document classification based on decision tree .....	250
III.3.2.2.3. Document classification based on neural network .....	259
III.3.2.3. Discovering user interests based on document classification .....	271
III.3.2.4. Evaluation .....	273
III.3.3. Constructing user groups or user communities .....	274
III.3.3.1. User model clustering .....	275
III.3.3.2. Overlay model clustering .....	277
III.3.3.2.1. In case that arcs in graph are weighted .....	279
III.3.3.2.2. In case that graph model is Bayesian network .....	282
III.3.3.3. Similarity measures for clustering algorithms .....	284
III.3.3.4. Evaluation .....	294
<b>Chapter IV. Evaluation of Triangular Learner Model.....</b>	<b>295</b>
<b>IV.1. Evaluation of knowledge model .....</b>	<b>295</b>
IV.1.1. Evaluation of Bayesian network .....	296
IV.1.2. Assessment of Bayesian network .....	298
IV.1.2.1. Overview of assessment approaches .....	298
IV.1.2.2. Towards the Computerized Adaptive Testing .....	301
IV.1.2.2.1. Overview of Computerized Adaptive Testing (CAT) .....	302

IV.1.2.2.2. Maximum likelihood estimation (MLE) for CAT.....	305
IV.1.2.2.3. New version of CAT algorithm based on MLE.....	312
<b>IV.2. Evaluation of adaptive learning model.....</b>	<b>330</b>
IV.2.1. Evaluation criterions .....	331
IV.2.1.1. Calculating system criterion $\alpha$ .....	332
IV.2.1.2. Calculating academic criterion $\beta$ .....	334
IV.2.1.3. Calculating adaptation criterion $\gamma$ .....	335
IV.2.2. An evaluation scenario .....	337
<b>Chapter V. Conclusion and Future Trend .....</b>	<b>340</b>
<b>V.1. Conclusion .....</b>	<b>340</b>
<b>V.2. Towards ubiquitous user modeling .....</b>	<b>344</b>
V.2.1. Overview of ubiquitous user modeling.....	345
V.2.1.1. User modeling and context-awareness.....	345
V.2.1.2. Ubiquitous computing.....	346
V.2.1.3. Semantic web .....	349
V.2.2. Knowledge presentation of situation model.....	350
V.2.3. Ubiquitous World.....	355
V.2.3.1. User model ontologies .....	355
V.2.3.2. Ubiquitous World .....	357
V.2.4. Ubiquitous user model service .....	359
V.2.4.1. Architecture of ubiquitous user model service .....	359
V.2.4.2. Main work flow of ubiquitous user model service.....	361
V.2.5. Incorporating Zebra into ubiquitous user model service .....	362
<b>Related Publications .....</b>	<b>366</b>
<b>References .....</b>	<b>368</b>
<b>Appendix .....</b>	<b>383</b>
A. List of Tables .....	383
B. List of Figures.....	386
C. List of Formulas .....	390
D. Index .....	395



# Chapter I. User Model and Adaptive Learning

Because this research proposes a user modeling system for adaptive learning, chapter I will give us a survey of user model and adaptive learning before going into main works in chapters II and III. Essential concepts and methods relevant to user model and adaptive system are described in section I.1 and section I.2, respectively. Because Bayesian network is one important aspect of the research, section I.3 is reserved for introducing some typical Bayesian network models. Your attention please, all terms and concepts mentioned in this chapter are used over the whole research. User model, the main subject of the research, is introduced firstly in section I.1 as below.

## I.1. User Model

Formerly, learning management systems (Wikipedia, Learning management system, 2014) such as Moodle (About Moodle - the open source learning platform, 2014), Sakai (About Sakai Project, 2014) support well for interaction between learner and lesson, learner and teacher. However, every student has individual features such as knowledge, goals, experiences, interests, backgrounds, personal traits, learning styles, learning activities, and study results. So, there is emergent demand for tailoring learning materials such as lessons, exercises, tests to each student. This is personalized learning or adaptive learning process and the system which supports such process was called *adaptive learning system*. Thus, adaptive learning system is able to change its action to provide both learning contents and pedagogic environment/methods appropriate to each student. Adaptive system bases on the “description of learner’s properties” called *user model* or *learner model* so as to make adaptation. In other words, adaptive learning system tunes learning materials and teaching methods to learner model. It is easy to recognize that learner model contains aforementioned features such as knowledge, goals, experiences, interests, backgrounds, personal traits, learning styles, learning activities, study results. The process which gathers information to build up learner model and update it was named: *user modeling* or *learner modeling*. The system that performs user modeling process and manages user model is called *user modeling system*. User model and user modeling system are main subjects in this research.

Note that the term “learning” often refers to electronic learning or *e-learning* in this research if there is no additional explanation. E-learning is also generic term of web-based learning, distance learning and online learning, which is known as study activities together with support of computer and network (Fröschl, 2005, p. 12). In e-learning environment, interaction between learners and learners, between learners and teachers, between learners and learning materials often occurs on network or internet and learning materials are often electronic documents such as files, web pages and software instead of

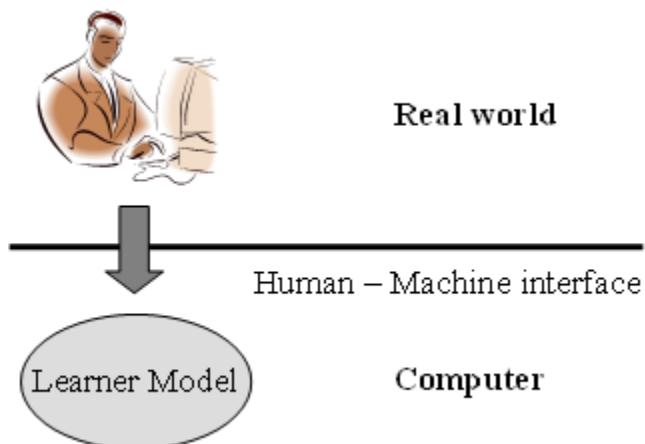
## I.1. User Model

---

traditional paper-based books. Essential concepts of internet and web page are introduced in (Wikipedia, Internet, 2014) and (Wikipedia, Web page, 2014).

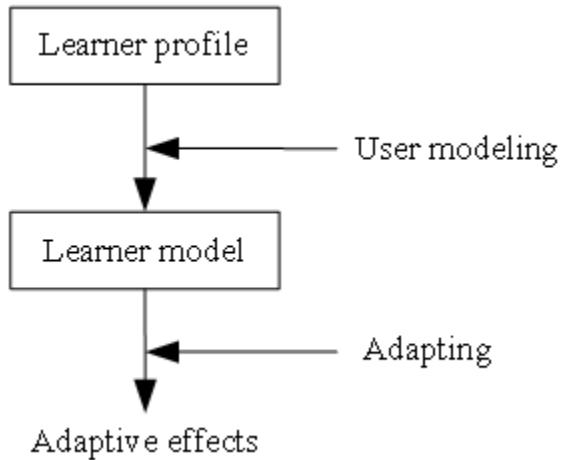
There is slight difference between user modeling and learner modeling as authors (Paiva & Self, 1995, p. 198) stated that “learner modeling is more concerned with diagnosing learner misconceptions and user modeling is more relevant to natural language understanding”. However, in this research, users are considered as learners or students who are diagnosed in their learning process. So, *user model, student model and learner model are synonymous terms* and I will use terms “user model”, “learner model” and “student model” interchangeably.

The terminologies: user model and user profile are often used interchangeably but they have slight difference. A profile contains personal information without inferring or interpreting. User model has a higher level than profile, expresses abstract overview of learner. Moreover, it is able to deduce more extra information about learner from model. User model is often applied in special domain.



**Figure I.1.1.** User modeling

Figure I.1.1 (Fröschl, 2005, p. 10) (Kay, 2001, p. 5) sketches out the user modeling process in which a learner in real world is simplified and simulated as a learner model stored in computer via human-machine interface. Because user modeling system provides valuable information in user model for adaptive learning system to make adaptation, it is possible to say that there is a duality of user modeling and adaptive learning. Figure I.1.2 (Fröschl, 2005, p. 13) (Brusilovsky, 1996, p. 2) depicts classical relationship between user modeling and adapting.



**Figure I.1.2.** Learner profile and learner model in adaptation

Because this research proposes a user modeling system for adaptive learning, it is relevant to both user modeling and adaptive learning. Thus, this chapter is survey of user model and adaptive learning.

Now basic concepts of user model (learner model), user modeling, user modeling system and adaptive learning system were introduced. Details of learner model along with description of learner's essential features and classification of learner model are described in sub-sections [I.1.1](#) and [I.1.2](#). After that, section [I.2](#) will explain adaptive learning system in detailed.

### I.1.1. Information in user model

User model must contain important information about user such as domain knowledge, learning performance, interests, preference, goal, tasks, background, personal traits (learning style, aptitude,...), learning activities, environment (context of work) and other useful features.

Content of learner model can be divided into two categories: domain specific information and domain independent information (Fröschl, 2005, p. 27).

#### I.1.1.1. Domain specific information

Domain specific information (Fröschl, 2005, p. 27) reflects the status and degree of knowledge and skills which student achieved in certain subject or major. Domain specific information is organized as knowledge model. Knowledge model has many elements (concept, topic, subject, etc.) which student needs to learn. Knowledge model can be created by some ways which result many forms. Some widespread forms will be introduced below:

- *Vector model.* Learner's knowledge in domain was modeled in a vector. This vector consists of knowledge items, concepts, topics, or subjects in domain. Each element of vector which is a real number or integer number (ranging within an interval) shows the degree which learner gains knowledge about those knowledge items, concepts, topics or

subjects. A typical example of vector model is  $U=(k_1, k_2, \dots, k_n)$  where each  $k_i$  is a numeric value expresses how much user masters over a knowledge item. Vector model is simplest but very effective.

- *Overlay model.* Learner's knowledge is the subset of expertise's knowledge. Similar to vector model, each element in overlay model is the number which presents learner's knowledge level (see more in subsection I.1.2.2).
- *Fault model.* The drawback of vector model and overlay model is that it cannot describe the lack of learner's knowledge. Fault model can contain learner's errors or bugs and what reasons learners have these errors. Taking out information from fault model, adaptive system can deliver learning material, concepts, subjects or topics that users don't know. Adaptive systems can also give users explanations, annotation to know accurately them or provide users guidance to correct errors.

Besides essential information about domain, there was extra information stored in learner model, such as prior knowledge of learner, records of learning performance, records of test results (Han B. , 2001, p. 13) (Fröschl, 2005, p. 29).

### I.1.1.2. Domain independent information

Besides information about knowledge, domain independence information (Fröschl, 2005, pp. 29-31) may include goals, interests, background and experience, individual traits, aptitudes and demographic information.

- *Interests.* Interest is particularly essential in commercial recommendation system. It is also important in adaptive educational system.
- *Goals.* In most cases, goal expresses learner's purpose; in other words, it is an answer for the question what learners want to achieve in learning course. There are two kinds of goal: long-term and short-term. Long-term goal is relatively permanent in course. Moreover, learner can propose herself/himself long-term plans for lifelong study. By short-term goal, learner intends to solve certain problem such as passing an examination, and doing exercise. Short-term goal is also called as problem-solving goal.
- *Background and experience.* Background includes skills or knowledge that learner gained in the past. Such information affects adaptive process. For example, if student experiences hardships in previous courses then adaptive learning system should deliver high level exercises to her/him.
- *Personal traits.* Personal traits are user's characteristics which together define a learner as an individual. Two basic personal traits are *learning styles* and *aptitudes*.
  - Learning styles are defined as “the composite of characteristic cognitive, affective and psychological factors that serve as relatively stable indicators of how a learner perceives, interacts with and

responds to the learning environment” (Stash, 2007, p. 93). Table I.1.1 shows some common learning styles.

Group	Description
<i>Activist</i>	Activists understand information only if they discussed it and applied it.
<i>Reflector</i>	Reflectors think thoroughly about things before doing any practice.
<i>Pragmatist</i>	Theorists think things through in logical steps, understand different facts into coherent theory (Stash, 2007, p. 106).
<i>Theorist</i>	Pragmatists have practical mind, prefer to try and test techniques relevant to problems (Stash, 2007, p. 106).

**Table I.1.1.** Some common learning styles

- There are eight forms of aptitudes (Fröschl, 2005, p. 30) (Lane, 2000): linguistic, logical/mathematical, spatial, kinesthetic, musical, interpersonal, intrapersonal, naturalist. Table I.1.2 shows eight forms of aptitudes.

Aptitude	Description
<i>Linguistic</i>	Competence to use language
<i>Logical-Mathematical</i>	Competence to use reason, number and logic
<i>Visual-Spatial</i>	Competence to perceive the visual
<i>Bodily-Kinesthetic</i>	Competence to use body effectively and handle objects skillfully
<i>Musical</i>	Competence to create and compose music
<i>Interpersonal</i>	Competence to communicate with other person
<i>Intrapersonal</i>	Competence to self-reflect
<i>Naturalist</i>	Competence to realize flora and fauna

**Table I.1.2.** Eight forms of aptitudes

However, for me, aptitudes are learner’s features not used usually in adaptive process because they are too complex and unpractical to implement in software engineering. Learning styles are more important than aptitudes.

- *Demographic information.* Demographic data includes name, birth day, sex, ID card, etc. In general, demographic information is used to identify person.

## I.1.2. Classification of user models

User models are classified into three main kinds such as stereotype model, overlay model and plan model according to their representations. Differential

model and perturbation model are variants of overlay model. These models are described in sub-section I.1.2.1 (stereotype model), I.1.2.2 (overlay model), I.1.2.3 (differential model), I.1.2.4 (perturbation model), and I.1.2.5 (plan model). Note that there are many approaches to construct these models, ranging from probability and statistics to artificial intelligence, machine learning and data mining. There are some recommended books for generic study of statistics, artificial intelligence, machine learning, and data mining such as “Applied Statistics and Probability for Engineers” by authors (Montgomery & Runger, 2003), “Artificial Intelligence: A Modern Approach” by authors (Russell & Norvig, 2003), “Machine Learning” by author (Mitchell, 1997), and “Data Mining: Concepts and Techniques” by authors (Han & Kamber, 2006). You should refer to these books for concepts in probability, statistics, artificial intelligence, machine learning and data mining occurring frequently in this research.

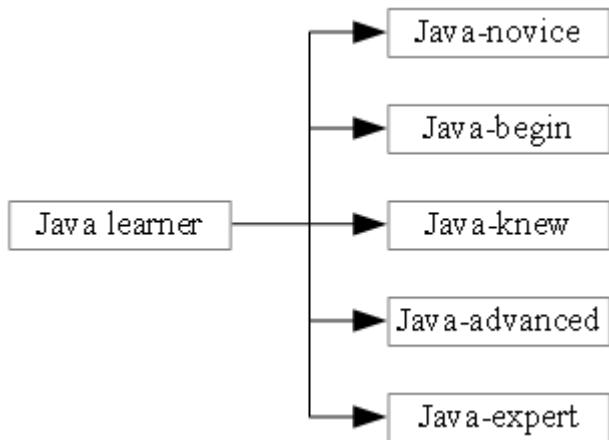
### I.1.2.1. Stereotype model

Stereotype (Rich, 1979) is a set of user’s frequent characteristics. New learner will be classified according to their initial features, each classifier is stereotype. It is able to infer much more new assumptions about user from small amount of information available in stereotype (Fröschl, 2005, p. 34). If information about user is gained in detailed and concretely, assumptions will be changed to become more precise. The term “assumption” refers the system’s belief about user but this belief is not totally reliable, just temporary.

In general, stereotype represents a category or group of learners. There are two kinds of stereotype: *fix* and *default* (Fröschl, 2005, p. 34).

In fix stereotype, learner is assigned to predefined stereotype at abstract level. For example (see figure I.1.3), in Java tutorial course, students are divided into five group, corresponding to five level, each level is more difficult than previous level: novice, begin, known, advanced and expert. Note that Java is popular object-oriented programming language <https://www.oracle.com/java>. After obtaining individual information such as former knowledge, experience; system will assign each student to one of five levels and never change.

In default stereotype, it is more flexible. Therefore, first, learner is assigned to the initial stereotype. It means that initial stereotype has “default” value. System will observe students and gather their performance data, actions, results of tests, etc. in learning process. Finally, system changes the initial stereotype to new more appropriate stereotype. Straightforward, the setting of stereotype is gradually replaced by more precisely and is more fit to learner.



**Figure I.1.3.** An example of stereotypes of Java learner

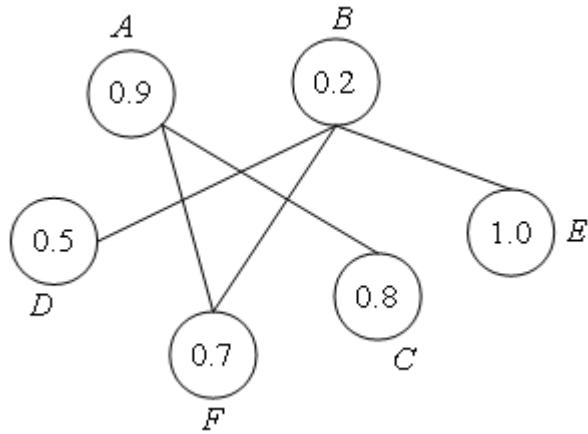
There are three important components in a stereotype: trigger, inference and retraction (Fröschl, 2005, p. 34):

- *Trigger* is used to active a stereotype. In other word, it is a condition (e.g. logic expression) to assign a stereotype to learner. For example: if trigger “don’t know Java” is activated, the stereotype “Java-novice” will be assigned to learner.
- *Inference* is inferring engine, responsible for deducing related information about user from stereotype. For example: if learner is glued to “Java-expert” stereotype, inference engine should take out both essential and extra information such as learner knows object-oriented programming, interface, swing, internationalization problem, and Java pattern.
- *Retraction* conditions are used to deactivate learner’s stereotype. There is a circumstance: student was assigned stereotype “Java-novice” at the beginning of course but after learning process, student knew thoroughly Java, so his stereotype “Java-novice” is no longer suitable. Event “Users do final Java test very well” is condition to retract his stereotype “Java-novice” and he will be assigned a new appropriate stereotype – “Java-expert”.

### I.1.2.2. Overlay model

The essential idea of overlay modeling is that the learner model is the subset of domain model. In other word, the user overlay model is the shot of comprehensive domain model. Domain model is constituted by a set of knowledge elements representing expertise’s knowledge, normally; each element represents a knowledge item, concept, subject or topic in the major. So, the structure of user model “imitates” the structure of domain model. However, each element in user model (corresponding to each element in domain model) has a specific value measuring the user’s knowledge about that element. This value is considered as the mastery of domain element ranging with certain interval.

Straightforward, the domain is decomposed into a set of elements and the overlay model (namely, user model) is simply a set of masteries over those elements. Suppose that the mastery of each element varies from 0 (*not mastered*) to 1 (*mastered*). Then the expert model is the overlay with 1 for each element and the learner model is the overlay with at most 1 for each element.



**Figure I.1.4.** An example of overlay model having six concepts

An example of overlay model is shown in figure I.1.4 in which there are six concepts  $A, B, C, D, E$  and  $F$ ; each concept (element) is attached a number ranging in  $[0, 1]$  indicating user's mastery over such concept. The arc connecting two concepts expresses their relationship. There are many relationships such as prerequisite, dependency, parent-child, cause-effect, aggregation, and diagnostic. Each relationship is specified according to application context. Moreover, arcs can be direct or indirect; figure I.1.4 depicts a indirect overlay model.

Overlay modeling approach was based on domain models which are often constructed as knowledge network or knowledge hierarchical tree. Authors and experts have responsibility for creating domain model. Normally, each concept in domain model is mapped to learning object. Nowadays, there is a trend to build up domain model by ontology (Wikipedia, Ontology (information science), 2014).

Additionally, 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. This research combines overlay model and Bayesian network to construct knowledge model, which is discussed in sub-section III.1.1. Other systems applying Bayesian network into building up user model are introduced in section I.3.

### I.1.2.3. Differential model

Overlay model was based on expert's domain knowledge but there is need for learner/teacher to suppose the knowledge which is necessary to learner. That

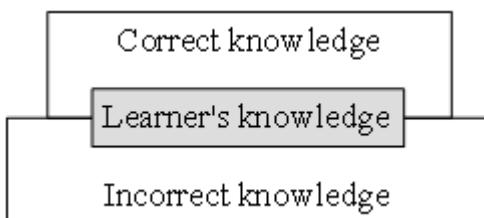
knowledge was called *expected knowledge*. In other word, expected knowledge is domain knowledge that learner should be mastered at the certain time.

Therefore, differential model (Mayo, 2001, p. 58) is basically an overlay on expected knowledge, which in turn is an overlay on expert's domain knowledge. With the overview of top-down methodology, differential model is a variant of overlay model. But in detailed, the differential model is instance of the class "fault model" (see sub-section I.1.1.1) because expected knowledge can be considered as the knowledge that user lacks.

#### I.1.2.4. Perturbation model

Both overlay model and differential model assume that learner's knowledge is the subset of expertise's knowledge. They are not interested in learner's errors caused by misconceptions or lack of knowledge. These errors are considered as *mal-knowledge* or incorrect beliefs.

Perturbation model (Mayo, 2001, p. 59) represents learners as the subset of expert's knowledge (like overlay model) plus their mal-knowledge. Hence, perturbation model is also instance of the class "fault model". This model open up new trend of modeling, so it can support better for adaptive system. In figure I.1.5 (Mayo, 2001, p. 60), learner knowledge shown as shading area includes both correct knowledge and incorrect knowledge (mal-knowledge).



**Figure I.1.5.** Illustration of perturbation model

#### I.1.2.5. Plan model

Plan is a sequence of learners' actions to achieve desires or concrete goals (Fröschl, 2005, p. 35). Plan recognition is based on tracking input user's performance (Kobsa, 1993, p. 3). There is the library consisting of all possible plans. User's actions are regarded and matched to these plans. The plan which is most similar to user's actions is chosen as learner model. This is plan recognition process. In this approach, it is very expensive to create library and requires complex computation and large storage. Furthermore, matching algorithm needs careful implementation and spends much time in executing.

In general, user model has extremely important role in most user-oriented system, especially, adaptive learning system. It is not easy to classify learner models and methods of modeling but useful learner models are described in sub-section I.1.2. However, it is asserted that building up the learner model

must follow three below steps:

- *Initialization* is the first step in user modeling. It gathers information and data about user and it constructs user model from this information. Initialization process also determines structure of user model, reasoning method and storage of user model. There are two common ways to gain data about user so that system can initialize user model: explicit questions and initial tests (Fröschl, 2005, p. 36).
- *Updating* intends to keep user model up-to-date. System can observe user's actions, track user's performance, and analyze user's feedback. Those tasks are done implicitly or explicitly (Fröschl, 2005, p. 37).
- *Reasoning* new information about user out from available data in user model.

Reasoning is complicated but most interesting and so, my research also focuses on learner modeling and reasoning.

This sub-section ends up the survey of user model; you can read document (Nguyen & Do, Learner Model in Adaptive Learning, 2008) for more details about learner model. Adaptive learning system which takes advantage of user model is the main subject in next section [I.2](#). Additionally, existing user modeling systems that construct user model are introduced briefly in section [II.1](#).

## I.2. Adaptive Learning

The traditional learning with live interactions between teacher and students has achieved many successes but nowadays it raises the demand of personalized learning when computer and internet are booming. Learning is mostly associated with activities involving computers and interactive networks simultaneously and users require that learning material/activities should be provided to them in suitable manner. This is origin of adaptive learning domain. For this reason, the adaptive learning system (ALS) must have ability to change its action to provide learning content and pedagogic environment/method for every student in accordance with her/his individual characteristics. Adaptive systems are researched and developed for a long time; there are many kinds of them. So it is very difficult for researchers to analyze them.

Here, I intend to systematize methods and theories in developing adaptive learning systems along with their features (Nguyen, State of the Art of Adaptive Learning, 2009). This section I.2 brings out the entire overview of adaptive learning system.

The term *adaptive* is defined as “able to change when necessary in order to deal with different situations” (Fröschl, 2005, p. 11). In learning context, the adaptive learning system must have ability to change its action to provide learning content and pedagogic environment/method for every student in accordance with her/his individual characteristics such as knowledge, goal, experience, interest, background... when these characteristics vary from person to person and are structured in user model mentioned in previous section I.1. Therefore, adaptive leaning systems tailor learning material to user information. The survey of existing adaptive systems is represented in this section I.2.

Sub-section I.2.1 is to classify existing adaptive systems in their development history. Two modern and popular systems: Intelligent Tutoring System (ITS) and Adaptive Educational Hypermedia System (AEHS) are described in sub-sections I.2.2 and I.2.3; each system is surveyed entirely and enclosed with specific example. Sub-section I.2.4 is the evaluation of existing ITS and AEHS. Note that there is a strong relationship between user model (user modeling system) mentioned in previous section I.1 and adaptive learning system and hence, please pay attention that user model is the heart of modern adaptive system such as ITS and AEHS.

### I.2.1. Classification of adaptive learning systems

Along with the progress of adaptive learning research, there are five main trends of adaptive systems (Fröschl, 2005, pp. 14-18):

- Macro-adaptive system
- Micro-adaptive system
- Aptitude-treatment interactions system (ATI)
- Intelligent tutoring system (ITS)

## I.2. Adaptive Learning

---

- Adaptive Hypermedia System (AHS) or Adaptive Educational Hypermedia System (AEHS)

These systems are introduced in successive as below.

### **Macro-adaptive system**

The early researches on adaptive learning intend to adapt the instructional performances to students on the macro level. Such system is called macro-adaptive system (Fröschl, 2005, p. 14). Students are classified into groups by grades from tests. Students in the same group have similar adaptive instruction. To identify each student with her/his group leads to the poor adaptation. Besides, the groups rarely receive different adaptive instruction.

### **Aptitude-treatment interactions system (ATI)**

As known, e-learning environment serves many persons but is required to be appropriate to each individual. This system adapts specific instructional strategies to specific student's characteristics (aptitudes) such as knowledge, learning styles, intellectual abilities, and cognitive styles. ATI also permits user to control partially or totally the learning process (Mödritscher, Garcia-Barrios, & Gütl, 2004) (Fröschl, 2005, p. 14). User can control learning instruction or content presentation in course. Researches prove that successful level of user's control depends on her/his aptitudes.

### **Micro-adaptive system**

This system, so-called micro-adaptive performs adaptivities on micro level since it discovers and analyzes individuals need to provide user the appropriate instructions (Mödritscher, Garcia-Barrios, & Gütl, 2004) (Fröschl, 2005, p. 15). When student is ongoing learning process, system observes and diagnoses continuously his/her activities (Mödritscher, Garcia-Barrios, & Gütl, 2004). System's efficiency is evaluated on how much the adaptive procedures are tailored to user's needs.

### **Intelligent tutoring system (ITS)**

ITS which is the hybrid approach coordinates aspects of micro-adaptive system and ATI. ITS is implemented by artificial intelligence methods. It aims to resemble the situation in which teacher and student sit down one-on-one and attempt to teach and learn together. ITS considers both user's aptitudes and user's needs. This is the first system applying user modeling techniques. Hence, user information is collected and structured more comprehensively. By the possibility of inferring new information from user model, ITS can perform prominently adaptive strategies. ITS is subdivided into four main components (Mayo, 2001, p. 3) (Fröschl, 2005, p. 16): domain expert, user modeler, tutoring module and user interface which have respective functions (see sub-section I.2.2).

### **Adaptive Hypermedia System (AHS)**

AHS (Fröschl, 2005, p. 17) has also been researched for a long time until now, which is the next generation of ITS. AHS combines adaptive instructional systems (macro-adaptive, ATI, micro-adaptive, ITS) and hypermedia systems. For openers, we should glance over what is hypermedia. Hypertext is defined as a set of nodes of text which are connected by links; each node contains some amount of information (text) and a number of links to other nodes (Henze, 2005, p. 11). Hypermedia is an extension of hypertext, which makes use of multiple forms of media, such as text, video, audio, and graphics (Henze, 2005, p. 11).

Author (Brusilovsky, 1996, p. 1) stated that “AHS can be useful in any application area where the system is expected to be used by people with different goals and knowledge and where the hyperspace is reasonably big. User with different goals and knowledge may be interested in different pieces of information presented on a hypermedia page and may use different links for navigation”. In short, AHS uses the user model containing personal information about her/his goals, interests, and knowledge to adapt the content and navigation in hypermedia space; so it aims to two kinds of adaptation: adaptive presentation and adaptive navigation (see sub-section [I.2.3](#)). For example, if user is a novice, system gives more annotation about the lecture which she/he is studying.

### **Adaptive Educational Hypermedia System (AEHS)**

AEHS is specific AHS applied in learning context. Hypermedia space in AEHS is re-organized and tracked strictly. Moreover, it is kept large enough to be appropriate for teaching because user will be involved in trouble when navigating if hypermedia space is too large. There is separate knowledge space including knowledge items; each item is mapped to hypermedia in hypermedia space. Both knowledge space and hypermedia space constitute the document space. An AEHS consists of document space, user model, observations and adaptation component (Karampiperis & Sampson, 2005, p. 130). We will survey AEHS instead of AHS in sub-section [I.2.3](#).

### **I.2.2. Intelligent Tutoring System (ITS)**

Formerly, intelligent tutoring system (ITS) and artificial intelligence (AI) are areas which have been researched separately. AI developed fast in 1960's; Alan Turing ([http://en.wikipedia.org/wiki/Alan\\_Turing](http://en.wikipedia.org/wiki/Alan_Turing)) thought that computer can “think” as human. Education becomes the fertile ground for applying AI methods since computer plays the role of human teacher. More and more people attends distance courses in the universities and they want to become self-taught mans who prefer a lifelong study; so computer is the best choice. Early ITS is the Computer Assisted Instructional (CAI) system that was generative (Urban-Lurain, 2002). CAI system provides instruction aiming to improve students' skill. It gives students content presentation and records their learning performance but does not care about the knowledge students gained.

User not attached special importance in CAI system becomes the main object in the overall system in the next researches. The system no longer gives only one instructional pattern to all students; it wants to know what types of student are considered and determines which instructions should be presented adaptively to each individual. So, ITS is directed to modeling user. The first modeling approach is stereotype classifying users into groups of characteristics.

The rapid progress in AI supports many powerful mathematical tools for inference. The demand of reasoning new assumptions out of available information in user model is satisfied by using such tools. User's knowledge, needs and aptitudes are included in user model. Until now, ITS is evolved and distinguish from previous CAI system. The implicit assumption about the learner now focused on *learning-by-doing*. ITS is classified as being computer-based, problem-solving monitors, coaches, laboratory instructors and consultants (Urban-Lurain, 2002). The available information in user model, especially knowledge becomes more and more important.

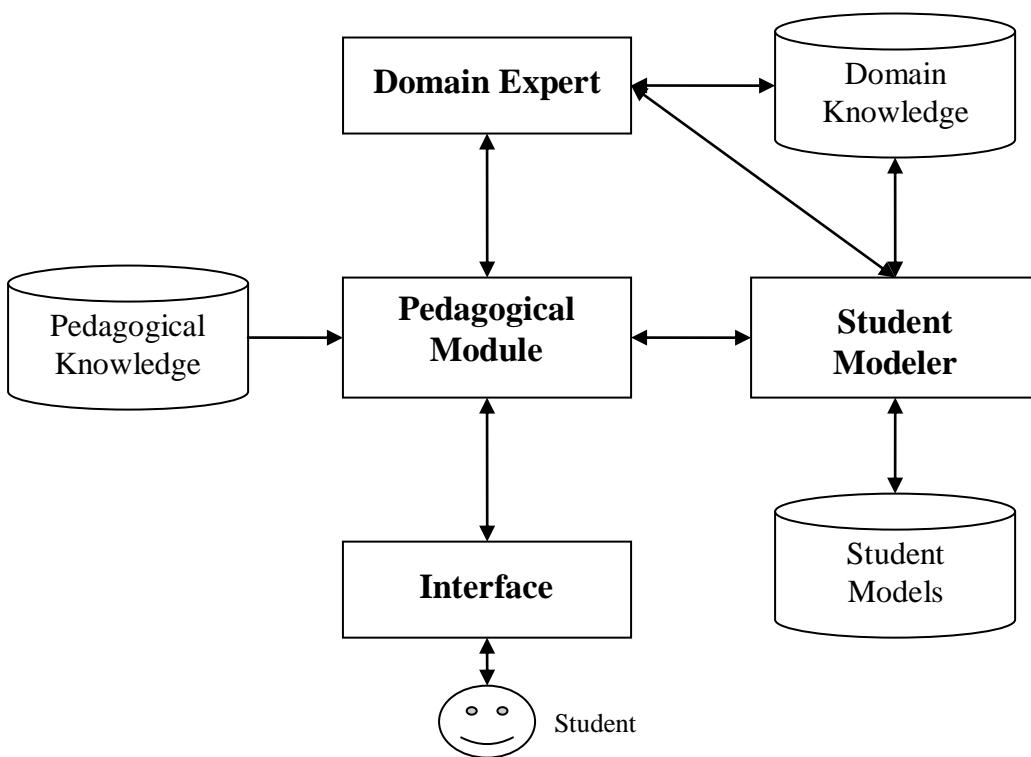
### I.2.2.1. Architecture of ITS

As discussed, ITS is the modern system since it inherits all strong points of micro-adaptive system, ATI and CAI. ITS has components expressing the content taught, adaptive procedures and the techniques for collecting, storing user characteristics and inferring new assumptions from them. General architecture of ITS shown in figure I.2.1 (Mayo, 2001, p. 3) is constituted of four main parts such as domain expert, user modeler, pedagogical module, and user interface as follows:

- *Domain expert* (Mayo, 2001, p. 2) is responsible for structuring, storing and manipulating knowledge space (domain knowledge). The quality of domain knowledge depends on domain expert; it varies from teaching strategies to a considerable amount of knowledge available in learning course. Knowledge space contents many knowledge items which student must be mastered in course. Domain expert supports directly pedagogical module to perform adaptive functions.
- *User modeler* (Mayo, 2001, p. 3) constructs and manages user information represented by user model. This includes long-term information such as goals, demography information, mastered knowledge and short-term information such as whether or not students do exercises, visit a web site,... User modeler also interacts with pedagogical module to catch and log user's tasks. User model can be stored in relation database (Ramakrishnan & Gehrke, 2003, pp. 25-94) or XML files (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008). User model has critical role; if it is bad in that it does not express solidly user's characteristics, the pedagogical module cannot make decisions in the proper way.
- *Pedagogical module* also called tutoring module or didactics module is the centric component in ITS, which adapts instructional procedures to students. This module makes decisions about the teaching process relating to the next problem selection, next topic selection, adaptive presentation of

error messages, and selective highlighting or hiding of text (Mayo, 2001, p. 3). Pedagogical module co-operates with the user modeler and domain expert to draw information about domain knowledge and user when making decisions. Pedagogical module is the heart of ITS, whose characteristics are summarized in sub-section I.2.2.2.

- *User interface* is the component taking full responsibility for user-machine interaction. The user interface is rather necessary when it proposes user friendly environment and gives motivation for student to learn (Mayo, 2001, p. 4). If other parts don't work properly or raises error, user interface can notice or guide user to overcome troubles when using ITS. The interface can improve learning by reducing the cognitive load. While three other parts focus on learning and adaptive procedures, the user interface aims to users.



**Figure I.2.1.** General Architecture of ITS

### I.2.2.2. Characteristics of pedagogical module

Author (Wenger, 1987) stated that “when learning is viewed as successive transitions between knowledge states, the purpose of teaching is accordingly to facilitate the student's traversal of the space of knowledge states”, referred from (Urban-Lurain, 2002). According to (Wenger, 1987), the core of ITS – pedagogical module provides two main adaptive tasks (makes decisions): diagnosis and didactics.

- *Diagnosis*: The ITS “diagnoses” students' states as three levels (Urban-Lurain, 2002):

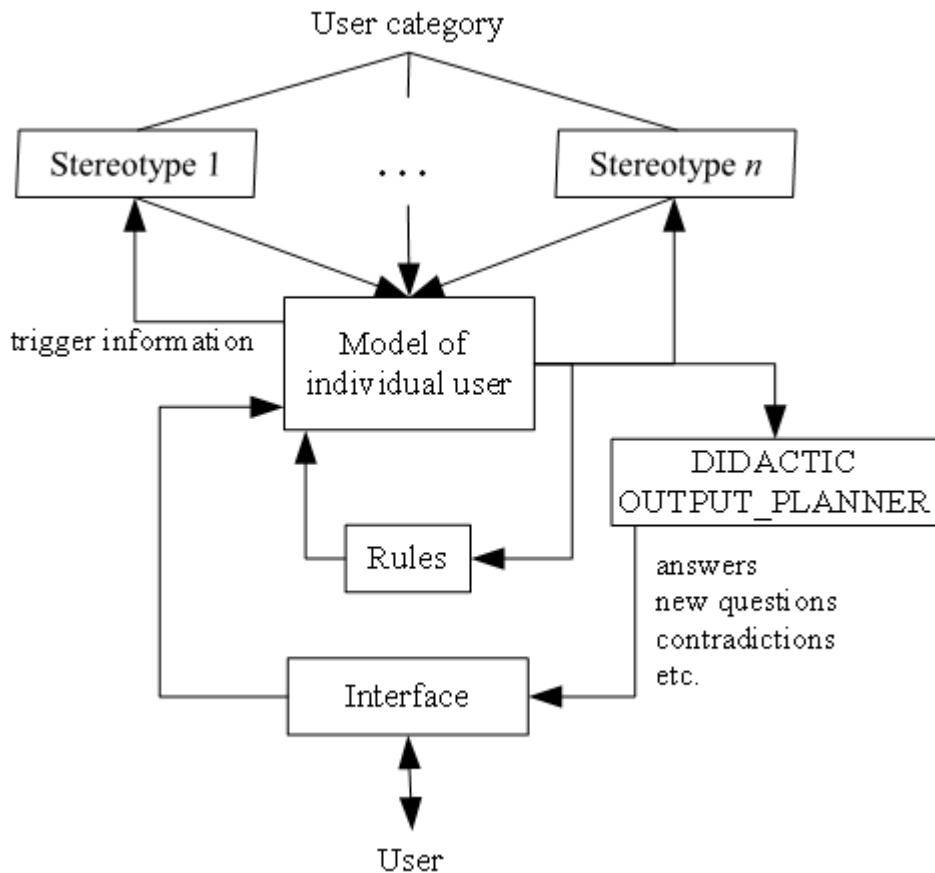
- *Behavioral level*: Learner's knowledge is ignored but her/his behavior is observed (Urban-Lurain, 2002).
- *Epistemic level*: Learner's knowledge state is inferred from her/his observed behavior (Urban-Lurain, 2002).
- *Individual level*: Learner's personal traits, motivational style, self-concept relevant to the domain, etc. are considered (Urban-Lurain, 2002). At this level, students become active learners.
- *Didactics* is the “delivery” aspect of teaching (Urban-Lurain, 2002), which is also referred as making decisions process. The author (Wenger, 1987) claimed that didactics is implemented according to four principles:
  - *Plans of action* are used to guide learner and provide the context for diagnostic operations (Urban-Lurain, 2002).
  - *Strategic contexts*: in which the plans of action are implemented (Urban-Lurain, 2002).
  - *Decision base* contains rules for dispatching learning and system resources according to pre-defined constraints (Urban-Lurain, 2002).
  - *Target level* of the student model: selecting the level at which the teaching takes place (Urban-Lurain, 2002). Depending on user state level, the pedagogical module will make appropriate instructional decisions.

### I.2.2.3. An example of ITS: ANATOM-TUTOR

As the name suggests, ANATOM-TUTOR developed by author (Beaumont, 1994) is the ITS used for anatomy education (specifically for brain, including the visual system, the pupillary light reflex system and the accommodation reflex system). Three important components in ANATOM-TUTOR are ANATOM knowledge base, the didactic module, and the user modeling component corresponding to three modules in the general architecture of ITS: domain expert, pedagogical module and user modeler.

The knowledge base contains anatomical concepts represented in frame-based formalism (Beaumont, 1994, p. 30). Concepts associated to their relations are located in the concept hierarchy. The reasoning is executed by built-in mechanism.

The user modeling component (shown in figure I.2.2) (Beaumont, 1994, p. 35) applies stereotype method to build up user model. First, system classifies users when they answer the initial questions at the start of course. This task will activate default stereotype for individual. Each user's stereotype is refined frequently by surveying and reasoning from observations.



**Figure I.2.2.** User modeling component in ANATOM-TUTOR

The didactic module “teaches” user by providing the adaptive knowledge in form of lessons, explanation, etc. There are two kinds of teaching knowledge (Beaumont, 1994, p. 30):

- Global teaching knowledge refers to the general structure of a lesson (Beaumont, 1994, p. 30).
- Local teaching knowledge refers here to what to do when a student gets into difficulties (Beaumont, 1994, p. 30).

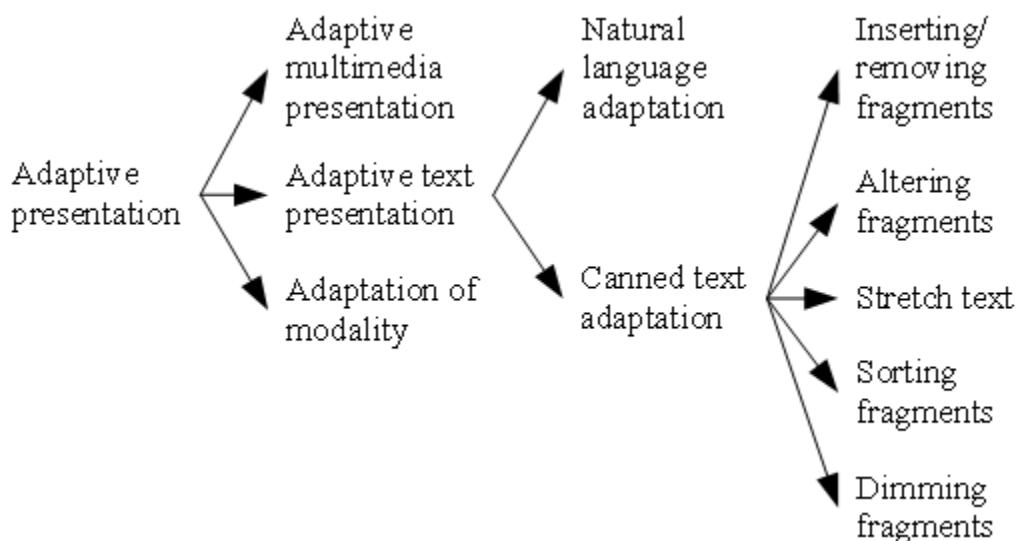
There are many ITS systems but ANATOM-TUTOR is given as typical example because its knowledge base, user model and pedagogical module have coherent interaction with built-in reasoning mechanism. Moreover, medical teaching is worthy to be attached special importance due to humanity.

### I.2.3. Adaptive Educational Hypermedia System (AEHS)

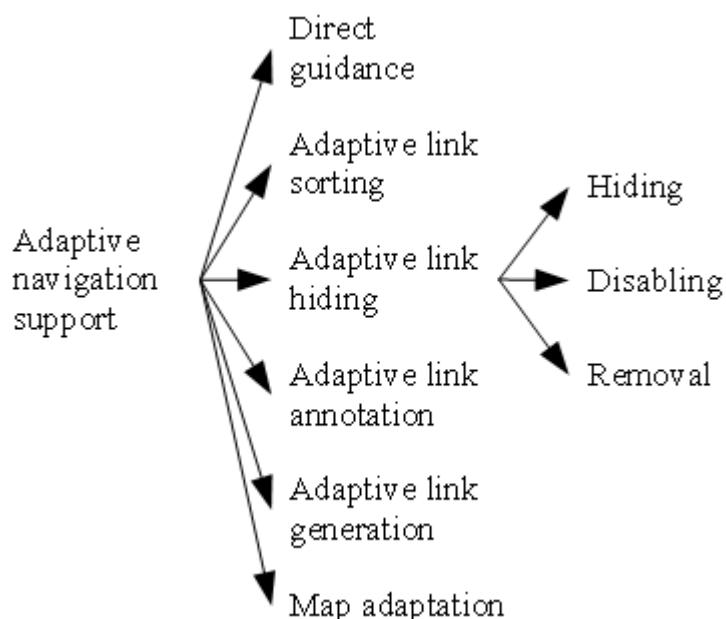
Adaptive Educational Hypermedia System (AEHS) inherits basic components of ITS in respect of implementation but takes advantage of plentiful supplies of learning material in hypermedia space. It is possible to understand that AEHS is specific AHS (see sub-section I.2.1) that serves educational purpose. As discussed in section I.1, adaptation is ability to change system’s behaviors to tune with learner model. When hypermedia is the combination of hypertext and multimedia, AEHS can be known as the system providing learner with learning

material in form of hypertext and multimedia like hyper book and electronic book tailored to learner's preference. According to (Brusilovsky, 1996, pp. 10-13), there are two forms of adaptation: adaptive presentation and adaptive navigation:

- *Adaptive presentation* refers to the information which is shown, in other word, what is shown to the user. Figure I.2.3 depicts adaptive presentation (Brusilovsky, 2001, p. 100).
- *Adaptive navigation* refers to manipulation of the links, thereby; user can navigate through in hypermedia. In other word, it is where user can go. Figure I.2.4 depicts adaptive presentation (Brusilovsky, 2001, p. 100).



**Figure I.2.3.** Adaptive representation



**Figure I.2.4.** Adaptive navigation

Canned text adaptation (De Bra, Stash, & Smits, 2005, p. 4) is the most important case of adaptive presentation. It focuses on processing adaptive text parts called as fragments. There are three main kinds of text adaptation:

- *Conditional text*: Fragments are inserted, removed, altered and dimmed when certain conditions relating user characteristics are met.
- *Stretch text*: Some keywords of document are replaced by longer descriptions according to user's knowledge.
- *Sorting fragments*: Fragments are sorted according to their relevance for the user.

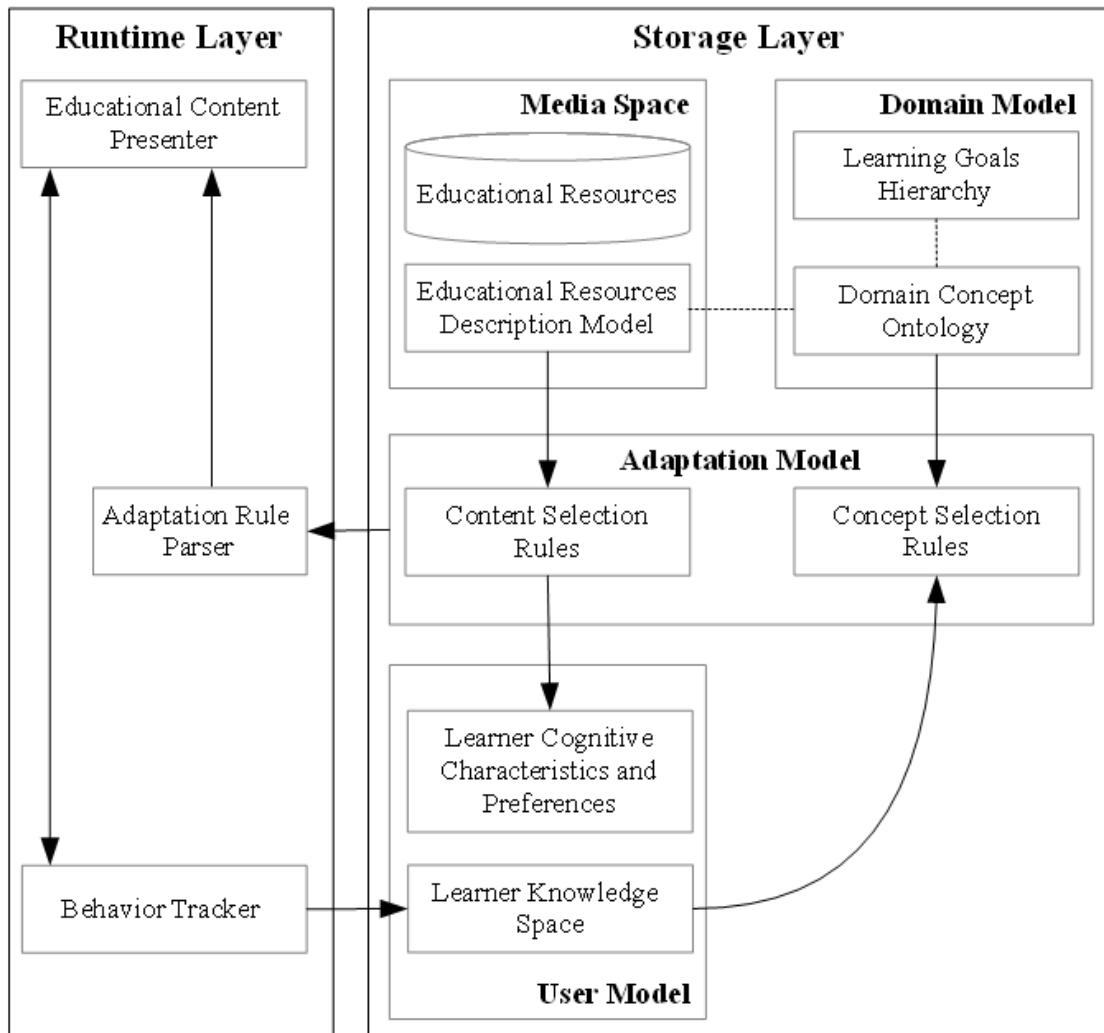
Adaptive navigation (De Bra, Stash, & Smits, 2005, p. 5) supports some following cases of navigations:

- *Direct guidance*: guiding user sequentially through the hypermedia system by two methods:
  - i. Next best: providing a suitable next link.
  - ii. Page sequencing or trails generate a reading sequence through hypermedia space.
- *Adaptive link sorting*: sorting links of hypermedia due to their relevance for user.
- *Adaptive link hiding*: limiting the navigational possibilities by hiding links not suitable to user. Link hiding is implemented by making it invisible or disabling it or removing it.
- *Link annotation*: showing users hints to the content of the pages which the links point to. The annotation might be text, icon or traffic light metaphor. The metaphor is displayed as a colored ball which is annotated the link pointing to a document in hypermedia space. For example, the red ball indicates that document is not recommended to user. The yellow ball has the same meaning to red ball but it is less strict than red ball. The green ball hints that document should be recommended to user. The grey ball indicates that user has already known this document.
- *Link generation*: generating appropriate links so that system prevents user from getting involved in large hyperspace.
- *Map adaptation*: graphical overviews of adaptive links.

### I.2.3.1. Architecture of AEHS

In general, the architecture of AEHS shown in figure I.2.5 (Karampiperis & Sampson, 2005, p. 130) has two layers: *runtime layer* and *storage layer*. Runtime layer has responsibility for presenting adaptive learning material to user and observing user in order to update learner model. Storage layer is the main engine which controls adaptive process with some tasks as follows:

- Initialize and update learner model.
- Choose concepts in domain model, educational resource in Media Space by selection rules.
- Store learning resources, domain ontology, learner model, etc.



**Figure I.2.5.** General architecture of AEHS

As seen in general architecture, storage layer has four models:

- **Media space:** contains learning resources and associated descriptive information (metadata). Learning resources such as lectures, tests, examples, and exercises are often stored as hypertext and hypermedia like (x)html files (W3C, XHTML™ 1.1 - Module-based XHTML - Second Edition, 2010). Media space is also called resource model.
- **Domain model:** constitutes the structure of domain knowledge. Domain model was often shown in the form of graph. Nowadays, researchers intend to build domain model according to ontology (Wikipedia, Ontology (information science), 2014).
- **Adaptation model:** is the centric component which gives effect to adaptation. It contains content selection rules and concept selection rules. We apply content selection rules into choosing suitable educational resource from medial model. On the other hand, concept selection rules are used to choose appropriate concept from domain model. These rules must tune with user model so that the selection gets correct.
- **User model:** contains information and data about user.

The generalized system of AEHS is adaptive education system (AES) in which it is not required to store learning resources in hypermedia space but AEHS is the most popular adaptive learning system.

### I.2.3.2. Characteristics of AEHS

AEHS has many variants in implementation; each of them conforms to specific requirements conditional upon application. It is very difficult to characterize such domain-dependent variants if it has no common language for describing AEHS. Authors (Henze & Nejdl, 2004) use first-order logic (FOL) to define a language for comparing and analyzing AEHS. Therefore, an AHES is a quadruple (DOCS, UM, OBS, AC).

**DOCS** (Henze & Nejdl, 2004, p. 4) refers both hypermedia space and knowledge structure (domain model). DOCS contains the documents which are organized in accordance with domain model representing all relationships between documents. Each document associated information / learning material in hypermedia space such as text, hypertext, and audio has the identifier denoted *doc\_id*. There are logical predicates that show the relationships of documents in domain model. For example (Henze & Nejdl, 2004):

- Predicate *part\_of(doc\_id<sub>1</sub>, doc\_id<sub>2</sub>)* means that *doc\_id<sub>2</sub>* is a part of *doc\_id<sub>1</sub>*.
- Predicate *preq(doc\_id<sub>1</sub>, doc\_id<sub>2</sub>)* means that *doc\_id<sub>2</sub>* is prerequisite of *doc\_id<sub>1</sub>*.
- Predicate *is\_a(doc\_id<sub>1</sub>, doc\_id<sub>2</sub>)* expresses the taxonomy on documents.
- Predicate *is\_dependent(doc\_id<sub>1</sub>, doc\_id<sub>2</sub>)* expresses the dependencies between documents.

So, every document considered as knowledge element or topic, which is the basic unit in DOCS.

**UM** (Henze & Nejdl, 2004, p. 4) is responsible for managing information about user such as knowledge, goals, and learning styles. In short, UM has below functions:

- Initialization is the process which gathers information and data about user and constructs user model from this information.
- Updating user model by using observations OBS.
- Reasoning new information about user out from available data in user model.

User models are classified into: stereotype model, overlay model, differential model, perturbation model, and plan model. For example, the domain (in DOCS) is decomposed into a set of elements and the overlay model is simply a set of masteries over those elements.

Each element in UM represents a user, which denoted logically by identifier *user\_id*. Characteristics assigned to user are expressed by predicates: *has\_property(user id, characteristic x)*, *has\_property(user id, characteristic x,*

*value*). The very important characteristic “knowledge” is expressed by predicates (Henze & Nejdl, 2004, p. 4):

- Predicate *know(doc\_id, user\_id)*: tells us whether user knows document.
- Predicate *know(doc\_id, user\_id, value)*: tells us amount of knowledge user has on document. The variable *value* refers the user’s knowledge level.

**OBS** (Henze & Nejdl, 2004, p. 5). Both system’s runtime behaviors concerning to user and user’s interaction with system are monitored and recorded as observations OBS. For example, how users did the test, whether users visited course web pages and how long users studied online are considered as observations OBS used to update user model. Hence, the objects of OBS for modeling observations are the users and observations. Suppose systems recognized that user *user\_id* visited the document *doc\_id*; this observation is expressed by predicates such as *obs(doc\_id, user\_id)* and *obs(doc\_id, user\_id, value)*. The following predicate is more complex, in which the *value* can tell us how many times user visited the document.

**AC** (Henze & Nejdl, 2004, p. 5), the most important (adaptive) component, contains rules supporting adaptive procedures. In other words, adaptive functionality is a set of rules describing AEHS’s runtime behaviors. Suppose there is a functionality which is to decide whether or not recommend a document for learning to student. This functionality which belongs to the group determining the learning state of documents is described as the rule: “student should learn a document if she/he has already visited (learned) all prerequisites of this document. This rule is expressed as FOL predicate (Henze & Nejdl, 2004, p. 5):

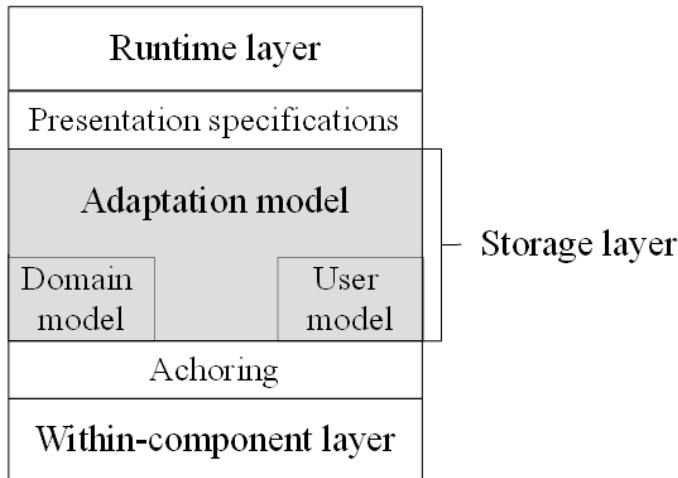
$$\begin{aligned} \forall user\_id, \forall doc\_id_1 (\forall doc\_id_2, & \text{preq}(doc\_id_1, doc\_id_2) \\ \Rightarrow obs(doc\_id_2, user\_id, & \text{visited})) \\ \Rightarrow learning\_state(doc\_id_1, user\_id, & \text{recommended\_for\_reading}) \end{aligned}$$

### I.2.3.3. An example of AEHS: Adaptive Hypermedia for All (AHA!)

Adaptive Hypermedia for All (AHA!) developed by author Paul De Bra (De Bra & Calvi, 1998) is a Java-based open source software which is based on Dexter Hypertext Reference Model (Halasz & Schwartz, 1990, p. 3) and aims to general-purpose adaptive hypermedia system. The AHA! architecture so-called AHAM complies with the standards of general architecture of AEHS, please the architecture of AEHS in sub-section I.2.3.1. The reference model AHAM (De Bra, Houben, & Wu, 1999, p. 3) has three layers: runtime layer, storage layer, within-component layer.

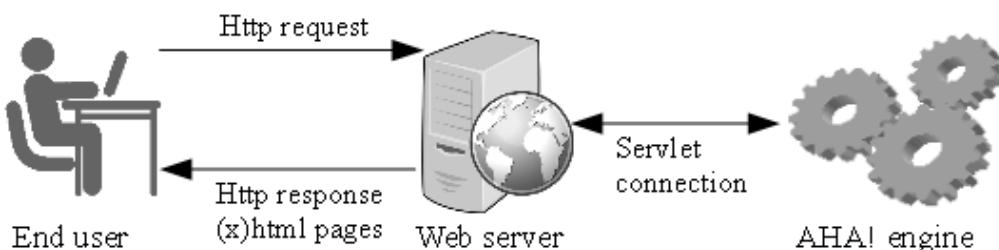
- Runtime layer has the same function to user interface module of ITS, which interacts with users. This layer must be implemented according to specifications in “Presentation specifications” (De Bra, Houben, & Wu, 1999, p. 3).

- Storage layer which is the heart of AHA! has three models: domain model (DM), user model (UM), adaptation model (AM). These models are managed by AHA! engine discussed later.
- Within-component layer describes the internal data objects, e.g. the resources (x)html (W3C, XHTML™ 1.1 - Module-based XHTML - Second Edition, 2010) linked to concepts. AHA! accesses these objects through “anchoring”.



**Figure I.2.6.** AHAM – The architecture of AHA!

The implemented framework of AHA! (De Bra, Smits, & Stash, 2006) is constituted of Java web server, connection servlets (Java Servlet, 2014) and AHA! engine. When users request a concept / document, the engine will return adaptive learning material consisting of (x)html pages, possibly other media objects. In brief, AHA!, a web-based system, receives user HTTP request (Wikipedia, Hypertext Transfer Protocol, 2014) and sends back HTTP response containing adaptive instructions. Figure I.2.6 (De Bra, Houben, & Wu, 1999, p. 3) and figure I.2.7 (De Bra, Smits, & Stash, 2006) depict the architecture AHAM and the implemented framework of AHA!, respectively.



**Figure I.2.7.** Implemented framework of AHA!

**The AHA! engine** (De Bra, Smits, & Stash, 2006) manipulates three main models as below:

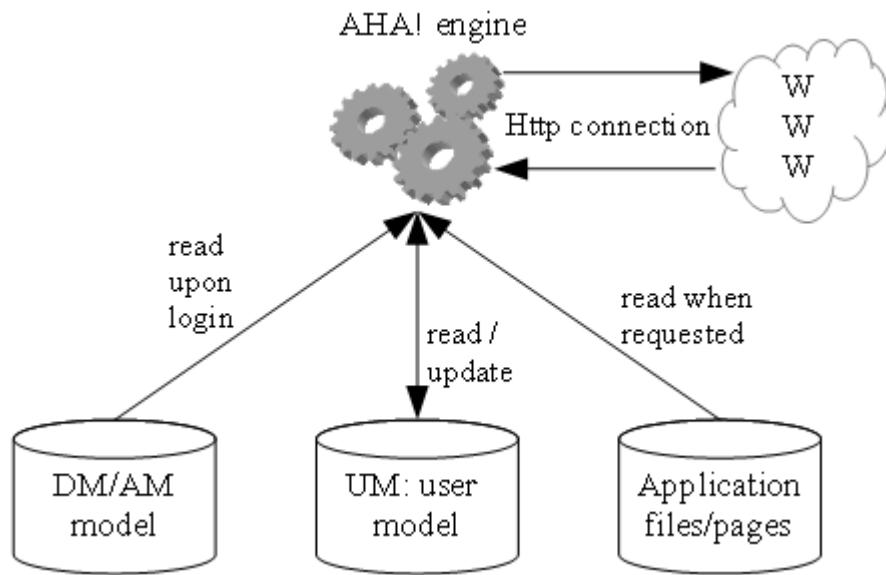
- Domain model (DM) contains domain concepts associated web learning resources. The relationships between concepts are also specified.

## I.2. Adaptive Learning

---

- User model (UM) describes user information. UM is built up by applying overlay modeling approach. Hence, UM is the mastered subset of DM.
- Adaptation model (AM) contains adaptive rules; each rule is associated to a domain concept and used to update UM when executing.

The combination of DM and AM represents a model of the *conceptual* structure of the application. So AHA! engine has responsibility for executing rules, updating user model and filtering retrieved resources. The AHA! engine is shown in figure I.2.8 (De Bra, Smits, & Stash, 2006).



**Figure I.2.8.** AHA! engine

Moreover, AHA! engine also manages application files, resource files in (x)html. DM, UM and AM are stored in relation database (Ramakrishnan & Gehrke, 2003, pp. 25-94) or XML files (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008). The engine manipulates them through three abstract tiers (De Bra, Smits, & Stash, 2006) as below:

- Concept tier: to create concepts, find concepts and link concepts to resources.
- Profile tier: to create user profiles, find a profile and destroy profiles. Note that AHA! identifies user profile with user model.
- Log tier: to record user's interactions with system, e.g. where and when user accessed some concepts.

**Adaptive procedure** is processed by the engine described above. The interaction between user and AHA! happens through HTTP protocol; whenever users send the HTTP request since the click on the link in course pages, adaptive process occurs as following steps (De Bra, Smits, & Stash, 2006):

1. Finding the request *concept* in DM and getting the resource (web pages, exercises, tests, etc.) associated with this concept.
2. Retrieving UM.

3. Executing adaptive rules (in AM) attached to the request concept. These rules will update attributes of UM (users' characteristics). The executing process spreads over AM since the update can trigger other rules associated updated attributes. So this process continues until no rule is triggered.
4. The resources associated to request concept are retrieved and filtered by adaptive rules in AM so that they are suitable to users (tuned to UM).

#### I.2.4. Evaluation of existing ITS and AEHS

We already described architecture and basic features of adaptive learning systems in which ITS and AEHS are researched and developed continuously nowadays. When AEHS and ITS are compared together, their architectures are very similar. For example, adaptation component in AEHS “plays the role” of pedagogical module in ITS. However, AEHS has some prominent advantages when it makes use of web technology. Hypermedia space in AEHS is more plentiful and convenient with non-linear navigation than knowledge space in ITS. Moreover, the interface in AEHS is more friendly than ITS due to using web page and HTTP protocol (Wikipedia, Hypertext Transfer Protocol, 2014) as means of interaction between user-machine in learning environment. That client-server architecture is implemented perfectly in AEHS through HTTP protocol will provide user the collaborative environment in e-learning; learners can share their experiments over network.

AEHS has some variants such as Adaptive Educational Web-Based System (AEWBS) and Adaptive Educational Intelligent Web-Based System (AEIWBS) focusing on web technology and artificial intelligence techniques but the ideology of such variants does not go beyond AEHS. So, I don't include them in this chapter.

There is a strong relationship between user model and adaptive system when adaptive system takes advantages of user model so as to make adaptation effects; both of them are surveyed particularly in this section I.2 and previous section I.1. As aforementioned in sub-section I.1.2.2, overlay model is essentially graph model whose nodes are knowledge elements and arcs express relationships of nodes. This research uses Bayesian approach to combine overlay model and Bayesian network to construct so-called Bayesian overlay model (see sub-section III.1.1) which is essentially a variant of overlay model. Because Bayesian network is important aspect of the research, next section I.3 introduces some typical user modeling systems that apply Bayesian network.

## I.3. Bayesian network user model

Note that the user modeling system proposed in this research uses not only Bayesian network (Nguyen, Overview of Bayesian Network, 2013) but also other techniques such as hidden Markov model (Schmolze, 2001) and data mining to construct and exploit user model. However Bayesian network approach is very important in the design of user modeling system. So I reserve this section I.3 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 sub-section III.1.1.

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 I.3 are based on Bayesian network and so, you can read the report (Nguyen, Overview of Bayesian Network, 2013) which is a good introduction to generic Bayesian network together with basic concepts, inference and learning techniques.

### I.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  $i^{th}$  component of this vector is the conditional probability expressing how user masters the  $KI_i$  (Henze, 2000, p. 46).

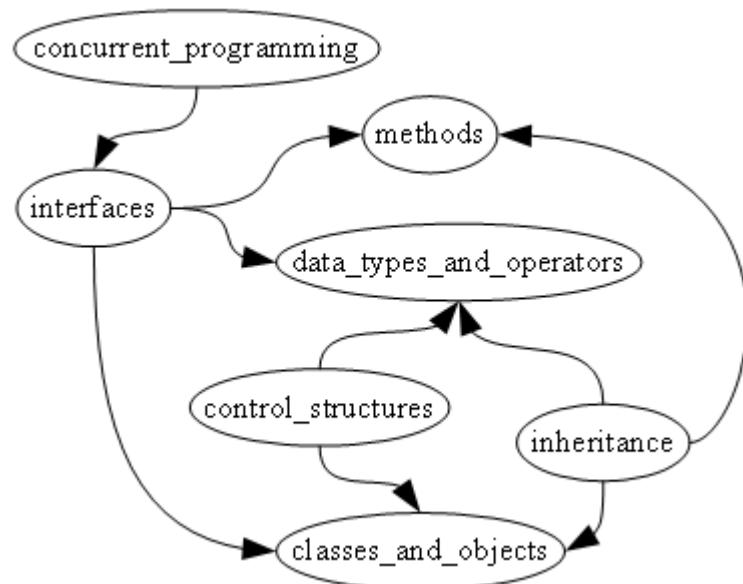
$$KV(U) = \{ P(KI_1/\mathcal{D}), P(KI_2/\mathcal{D}), \dots, P(KI_n/\mathcal{D}) \}$$

Where  $KI_1, KI_2, \dots, KI_n$  denotes knowledge items and  $\mathcal{D}$  is evidence that system observes about user in learning process. Note that *notation  $P(\cdot)$  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  $Z$  intervening 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.

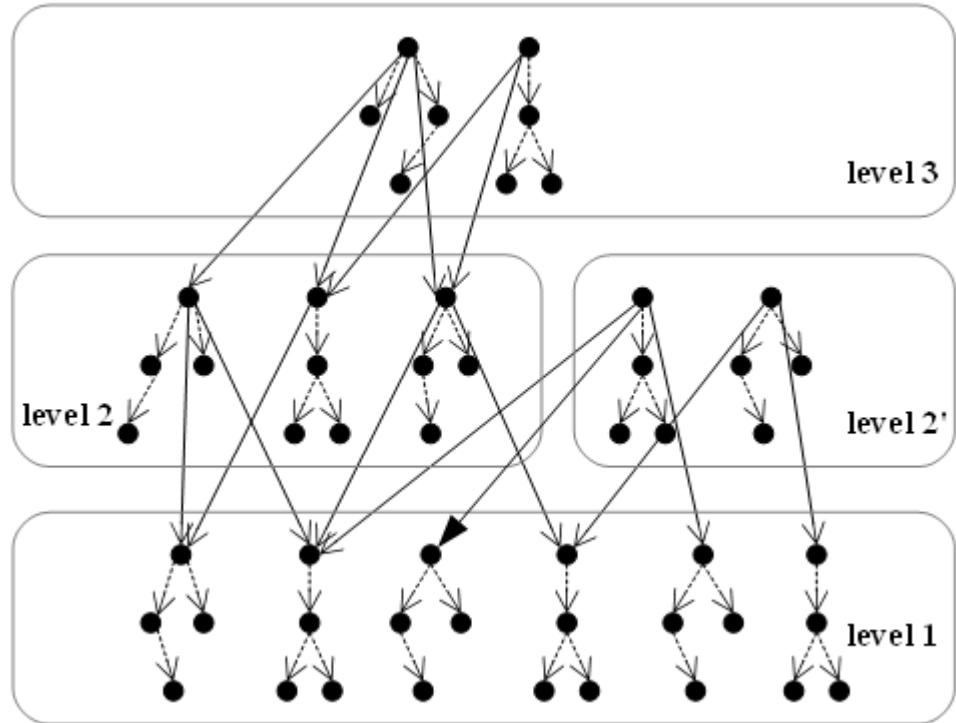
- The first level includes top-most  $KI(s)$ .
- The second level includes  $KI(s)$  that require top-most  $KI(s)$ . This level is further divided into two parts: one that is prerequisite for some third-level  $KI(s)$  and one that is not required by any third-level  $KI$ .
- The third level includes  $KI(s)$  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 I.3.1 (Henze, 2000, p. 52) depicts an example of dependency graph mentioning basic concepts of Java programming language (Oracle, Java language).



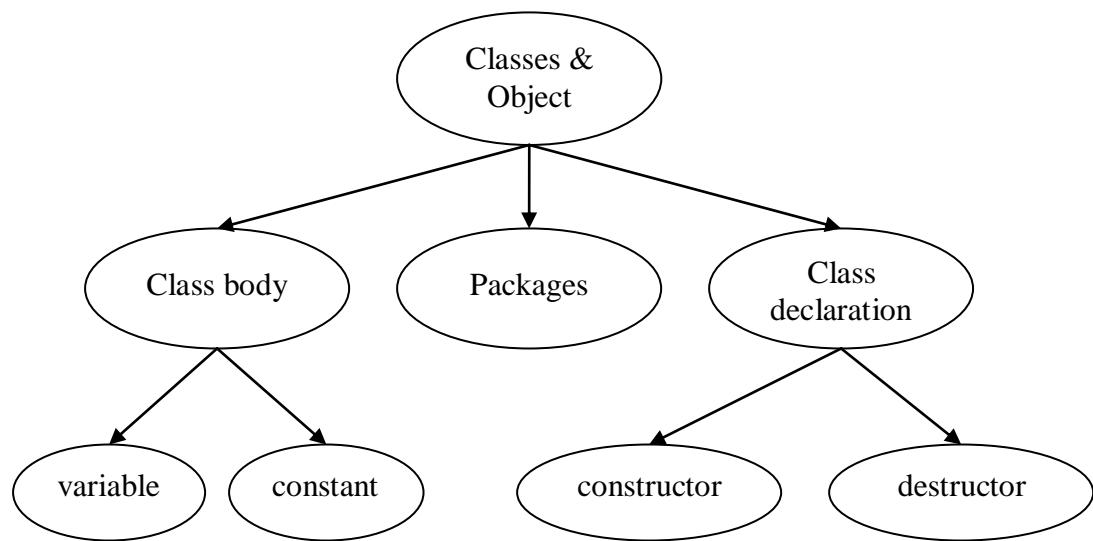
**Figure I.3.1.** An example of dependency graph

Figure I.3.2 (Henze, 2000, p. 53) expresses levels of  $KI(s)$ .



**Figure I.3.2.** 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 I.3.3 is an example of 1 root tree.



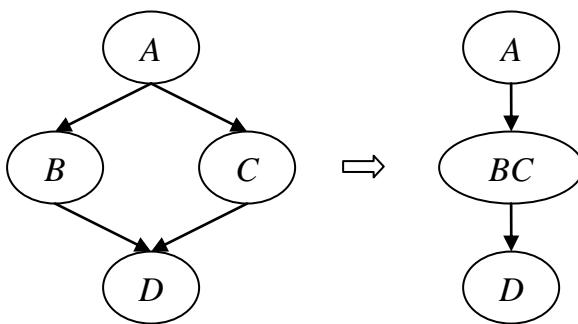
**Figure I.3.3.** Root tree

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 I.3.4 (Henze, 2000, p. 54)



**Figure I.3.4.** 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).

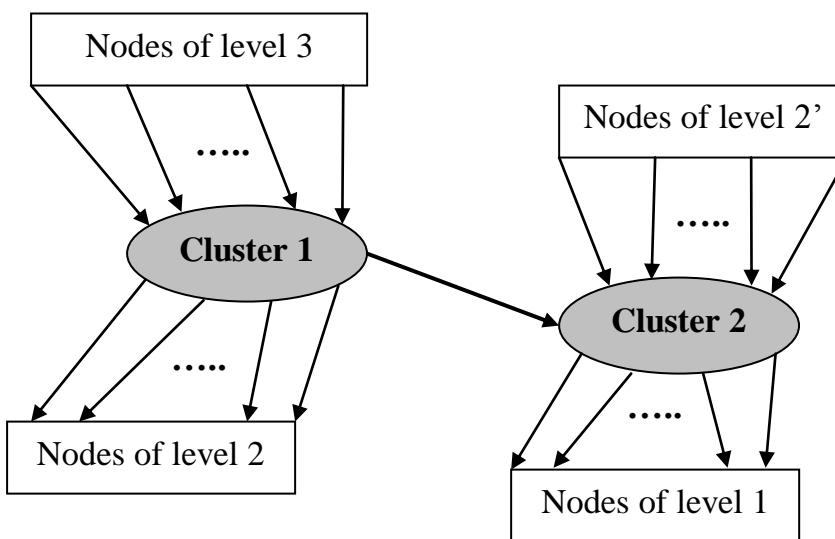
### I.3.1.1. Yet Another Clustering Formalism (YACF)

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 change, 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  $KI_3$ .

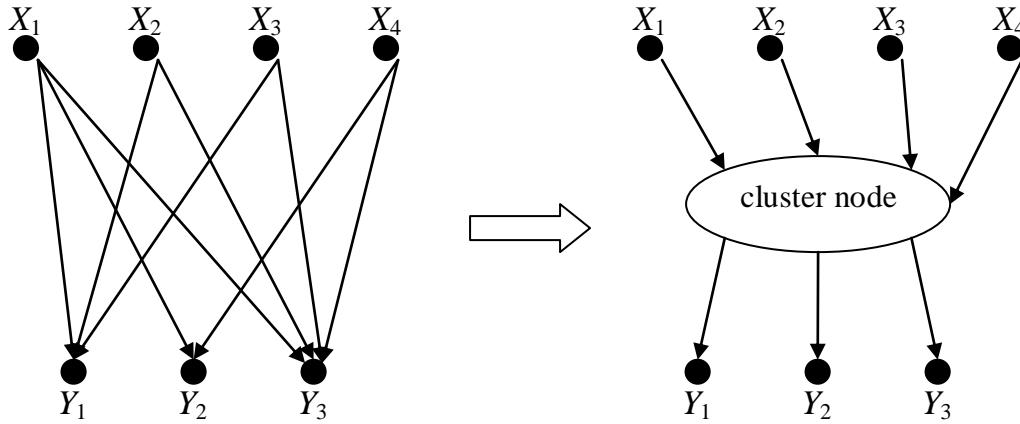
The excellence of YACF method is only to specify the conditional probability tables (CPT) of cluster node and child nodes. It isn't necessary to re-construct whole network; the structure of network and the CPT (s) of parent nodes are keep in origin. Figure I.3.5 (Henze, 2000, p. 56) describes YACF method.



**Figure I.3.5.** YACF method

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

Suppose  $X_1, X_2, \dots, X_N$  are parent nodes and  $Y_1, Y_2, \dots, Y_M$  are child nodes and  $H$  is cluster node (see in figure I.3.6) (Henze, 2000, p. 57). Each  $Y_i$  where  $i = \overline{1, M}$  is dependent on at least one  $X_k$  where  $k = \overline{1, N}$ . Let  $1Y_i, \dots, LY_i$  denote the part of the range of  $H$  which carries information for node  $Y_i$ . Note that if  $KI$  has for values as discussed  $\{E, F, A, N\}$  then the set  $\{1, \dots, L\}$  becomes  $\{E, F, A, N\}$  and values  $1Y_i, \dots, LY_i$  are often denoted  $E\_Y_i, F\_Y_i, A\_Y_i$  and  $N\_Y_i$ .



**Figure I.3.6.** Cluster node

The CPT of cluster node  $H$  is defined as the  $\prod_{l=1}^N |R(X_l)| \times \prod_{i=1}^M |R(Y_i)|$  matrix where  $R(\cdot)$  denotes the range of given variable. The author (Henze, 2000, p. 56) proposed formula I.3.1.1 for calculating conditional probability of cluster node.

$$P(H = k|Y_1, \dots, Y_M) \\ = \begin{cases} \frac{1}{M}, & \text{if } k = \text{best\_grade}(X_l \text{ (s) on which } Y_i \text{ is dependent}) \\ 0, & \text{else} \end{cases}$$

*Formula I.3.1.1.* Condition probability of cluster node

Where *best\_grade* return the maximum value of parent variables  $X_l$  (s) on which  $Y_i$  is dependent. Table I.3.1 (Henze, 2000, p. 57) shows the conditional probability table (CPT) of cluster node  $H$ , based on formula I.3.1.1.

$X_1, \dots, X_k, \dots, X_n$	$P(H=E Y_1/\dots) \dots P(H=N Y_1/\dots)$	...	$P(H=E Y_M/\dots) \dots P(H=N Y_M/\dots)$
$X_1=E, \dots, X_k=E, \dots, X_n=E$	1/M    0    0    0	...	1/M    0    0    0
.	.	.	.
$X_1=F, \dots, X_k=N, \dots, X_n=E$	0    1/M    0    0	...	1/M    0    0    0
.	.	.	.
$X_1=N, \dots, X_k=E, \dots, X_n=N$	1/M    0    0    0	...	0    0    0    1/M
.	.	.	.

**Table I.3.1.** CPT of a YACF cluster node

The conditional probability of child node  $Y_i$  where  $i = \overline{1, M}$  given cluster node  $H$  is defined in the following (Henze, 2000, p. 57):

$$P(Y_i|H = h(Y_i)) = \begin{cases} \left(\frac{1}{|R(Y_i)|}, \dots, \frac{1}{|R(Y_i)|}\right), & \text{if } h \notin \{1Y_i, \dots, LY_i\} \\ P(Y_i|X = h(Y_i)), & \text{else} \end{cases}$$

*Formula I.3.1.2.* Conditional probability of child node  $Y_i$

Table I.3.2 (Henze, 2000, p. 58) shows conditional probability of an arbitrary child node  $Y$  given cluster node  $H$ , based on formula I.3.1.2.

Cluster node ( $H$ )	$P(Y=E H=...)$	$P(Y=F H=...)$	$P(Y=A H=...)$	$P(Y=N H=...)$
$H_1=E$	0.25	0.25	0.25	0.25
$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$
$H_{i-1}=N$	0.25	0.25	0.25	0.25
$H_i=E$	<b>0.8</b>	<b>0.2</b>	<b>0.0</b>	<b>0.0</b>
$H_i=A$	<b>0.2</b>	<b>0.6</b>	<b>0.2</b>	<b>0.0</b>
$H_i=F$	<b>0.0</b>	<b>0.2</b>	<b>0.6</b>	<b>0.2</b>
$H_i=N$	<b>0.0</b>	<b>0.0</b>	<b>0.2</b>	<b>0.8</b>
(range of $H$ holding evidence for $Y$ )				
$H_{i+1}=E$	0.25	0.25	0.25	0.25
$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$
$H_n=N$	0.25	0.25	0.25	0.25

**Table I.3.2.** CPT of a child node  $Y$  given cluster node  $H$

The conditional probability of child node  $Y_i$  given parent nodes  $X_1, \dots, X_N$  is defined by (Henze, 2000, p. 59) as below:

$$P(Y_i|X_1, \dots, X_N) = (M - 1) \frac{1}{L * N} + \frac{1}{N} P(Y_i|H = h(Y_i))$$

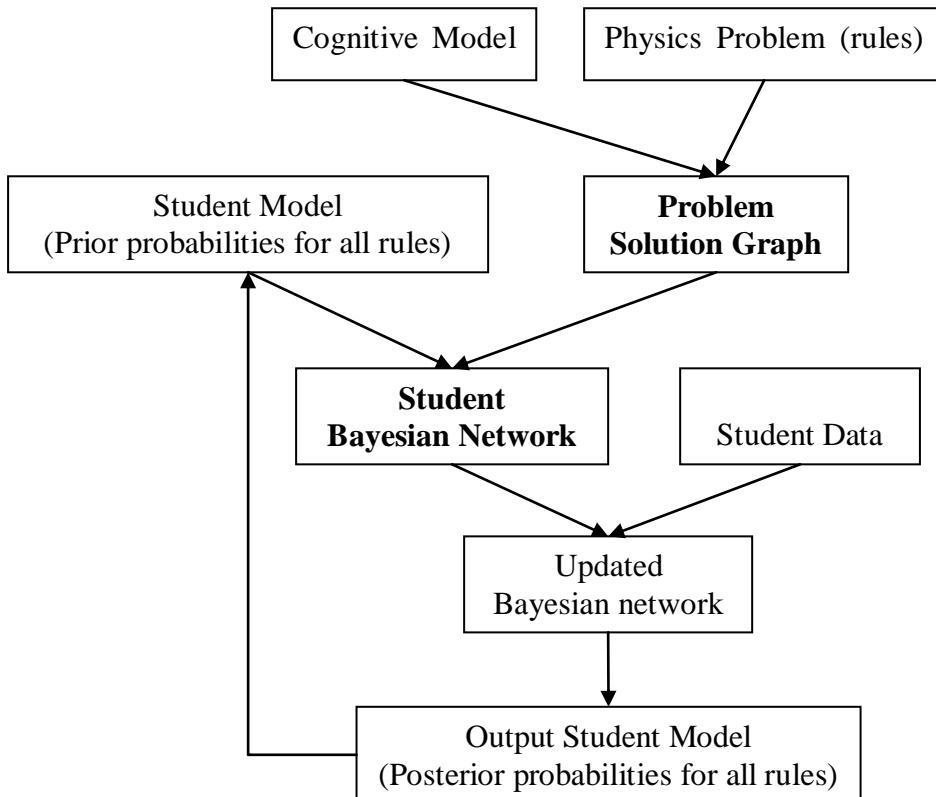
*Formula I.3.1.3.* Conditional probability of child node  $Y_i$  given parent nodes  $X_1, \dots, X_N$

When conditional probability of each child node  $Y_i$  is calculated according to formula I.3.1.3 which is core of YACF method, the Bayesian network without cycles is totally determined.

### I.3.2. Andes

Andes is developed by authors (Conati, Gertner, & Vanlehn, 2002) is the intelligent tutoring system (ITS, see sub-section I.2.2) that helps students to solve physical problems or exercises. 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 so-called *solution graph* which is then converted into Bayesian network. The solution graph is initialized right before student solves a problem. Figure I.3.7 depicts student modeling in intelligent tutoring system OLAE (Martin & VanLehn, 1995, p. 580) which is applied into Andes.



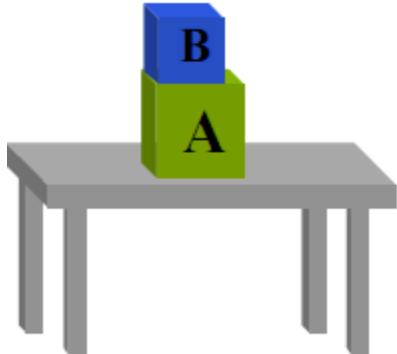
**Figure I.3.7.** Student modeling in Andes

### I.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 I.3.8 below:



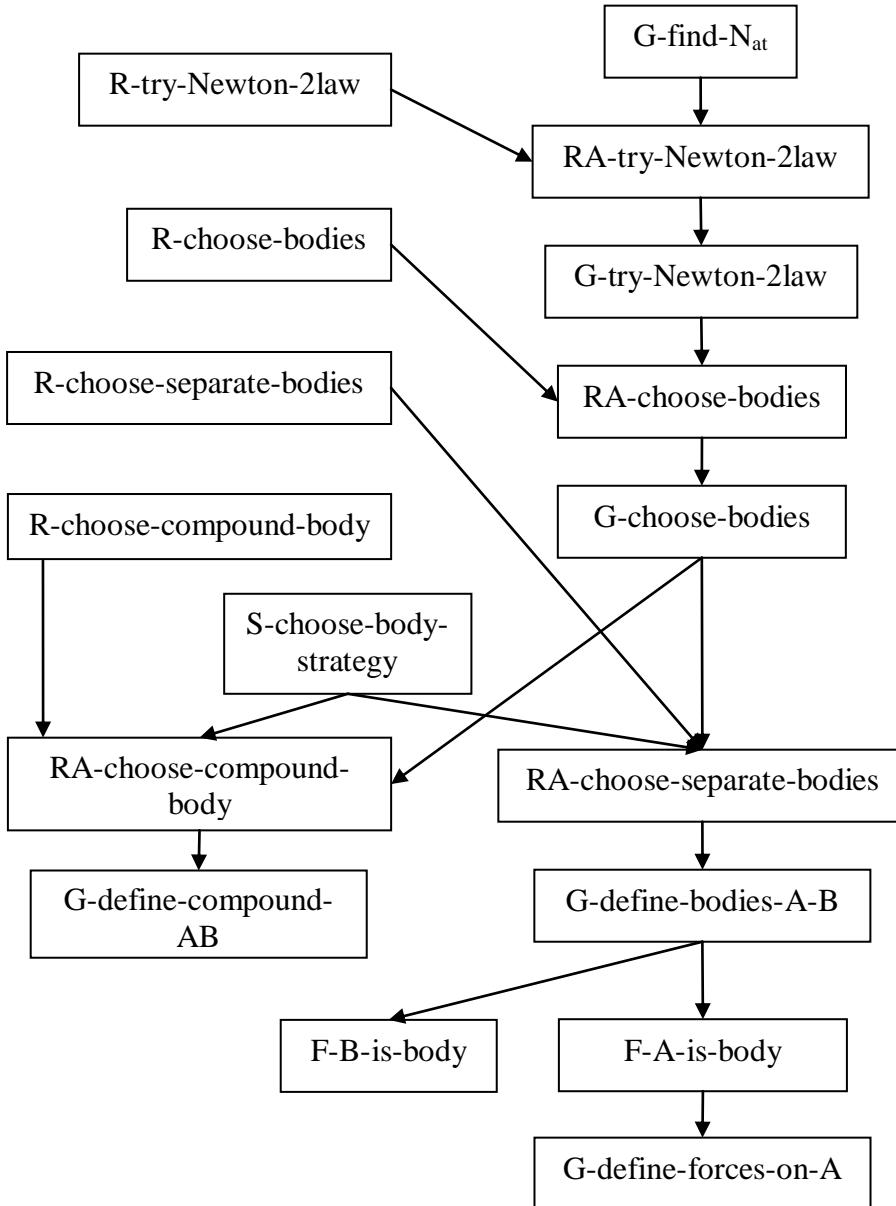
Block A of mass 50kg rests on top of a table. Another block B of mass 10kg rests on top of block A.

What is the normal force exerted by the table on block A?

(Conati, Gertner, & Vanlehn, 2002, p. 11)

**Figure I.3.8.** 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  $F = m\vec{a}$ " (Conati, Gertner, & Vanlehn, 2002, p. 11). The solution graph is shown in figure I.3.9 (Conati, Gertner, & Vanlehn, 2002, p. 11).



**Figure I.3.9.** 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-N<sub>at</sub>*, *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  $F = m\vec{a}$ , respectively. These nodes and relationships among them are used to construct the task-specific part of Bayesian network.

### I.3.2.2. Bayesian network in Andes

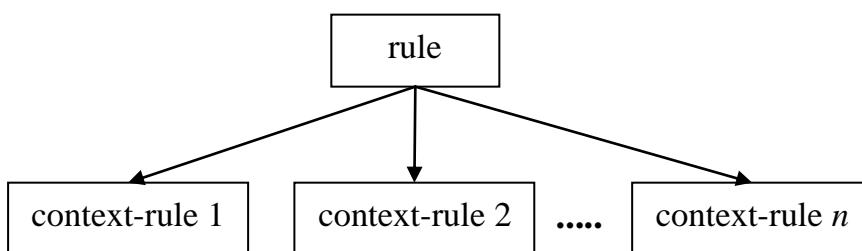
The Bayesian network in Andes includes two parts (Conati, Gertner, & Vanlehn, 2002, p. 12): one part so-called *domain-general part* that encodes the

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 I.3.10 (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.



**Figure I.3.10.** Relationship between rule and context-rule nodes

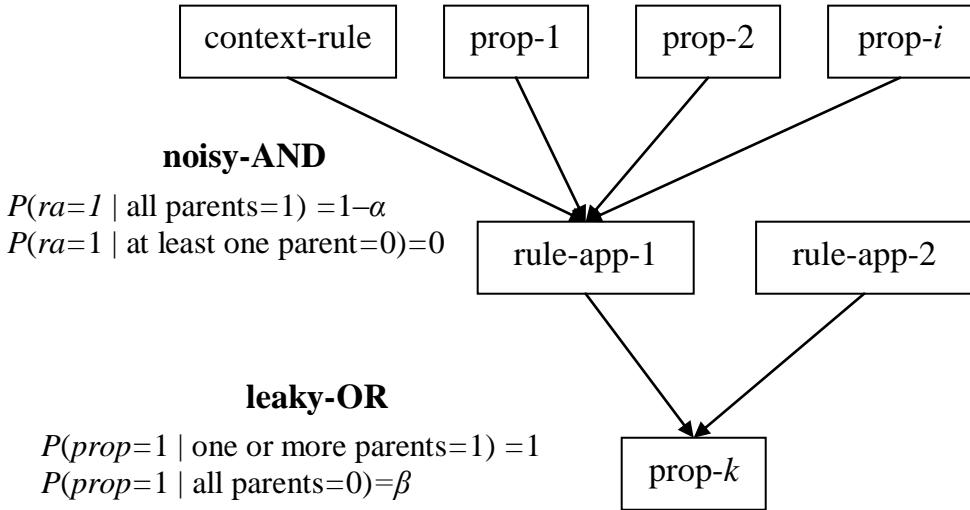
The conditional probability  $P(\text{context-rule}=\text{mastered} \mid \text{rule}=\text{mastered})$  equals 1 because if the student masters the general knowledge then she/he can apply it to solve any problems or exercise. The conditional probability  $P(\text{context-rule}=\text{mastered} \mid \text{rule}=\text{not-mastered})$  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.

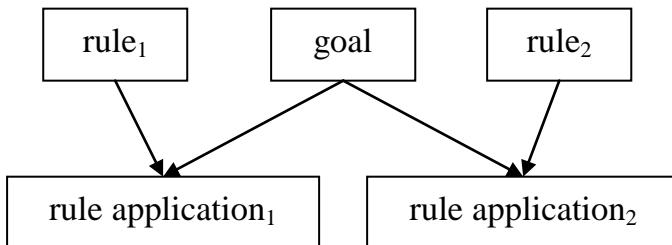
*Fact* and *goal* nodes (Conati, Gertner, & Vanlehn, 2002, p. 14) represent the propositions in domain; thus they are called *proposition nodes* (denoted with prefix *pr-*). They express the information that is derived when students apply context rules into solving problems or exercise. That a proposition 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  $\beta$  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  $1-\alpha$  only if all such parents gets *true*. The predefined real number  $\alpha$  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 I.3.11 shows relationship between nodes in task-specific part (Conati, Gertner, & Vanlehn, 2002, p. 14).



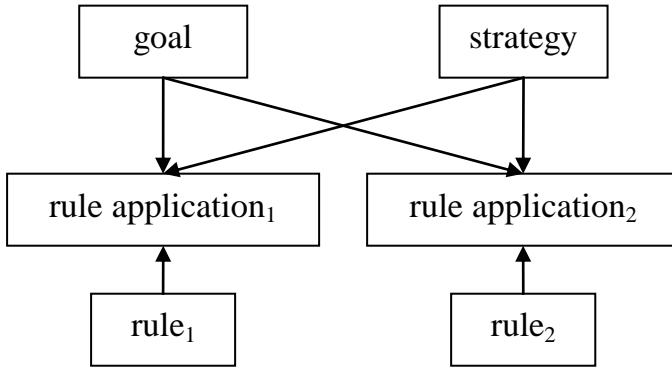
**Figure I.3.11.** Relationship between nodes in task-specific part

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 I.3.12.



**Figure I.3.12.** Mutually exclusive strategy

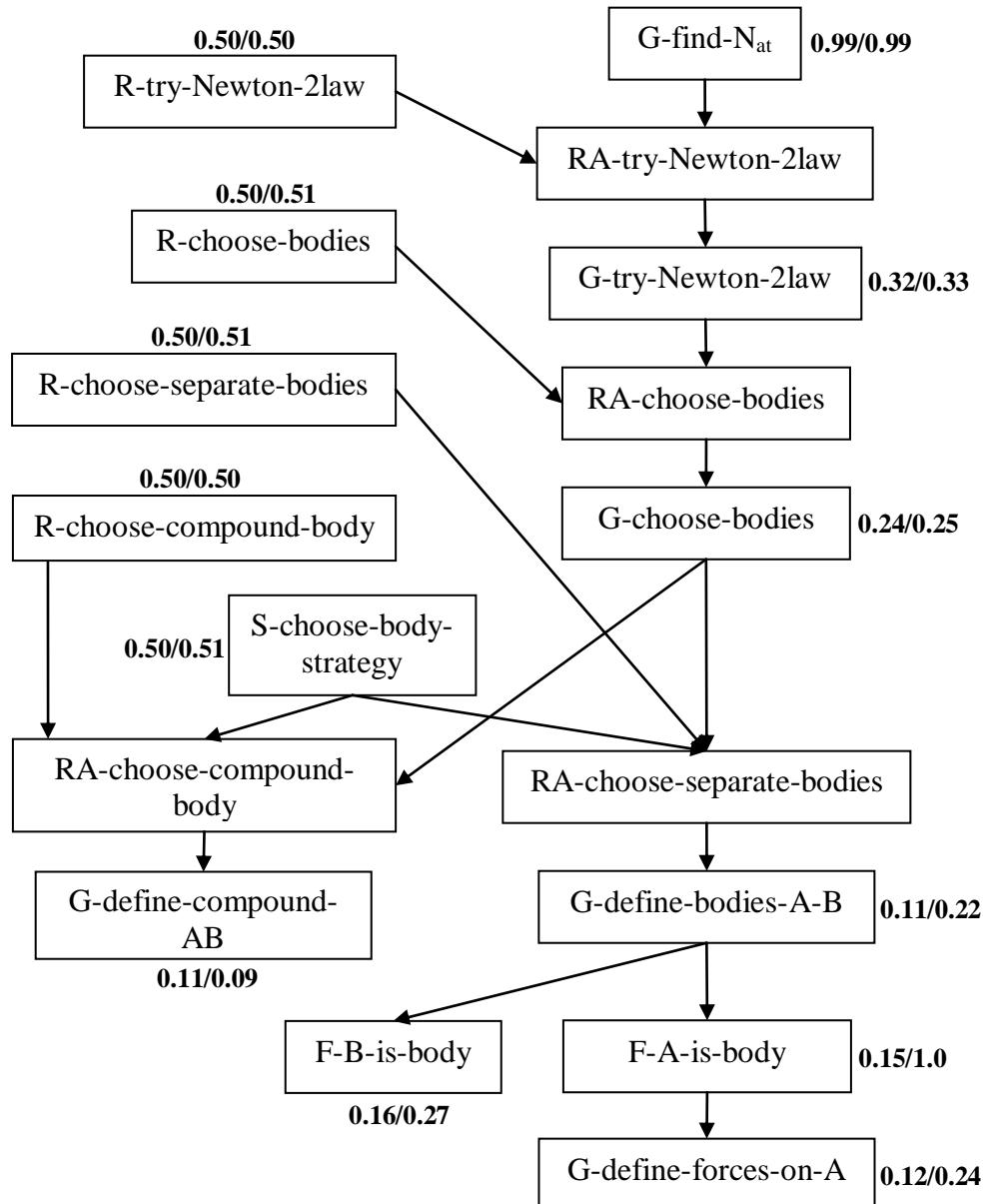
The strategy node is associated with goal node in order to come over mutually exclusive situation. Both strategy node and goal node are parent of some goal nodes. Each goal node (child node) is considered as different strategy when student solve a problem and it corresponds with one value of strategy 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 is this probability, the more do student prefers to select respective strategy. Strategy node is shown in figure I.3.13.



**Figure I.3.13.** Strategy node

### Inference mechanism in Bayesian network

Suppose a student who solves the problem of finding normal force in the example in figure I.3.8 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 that derive to evidence become higher and otherwise. Figure I.3.14 (Conati, Gertner, & Vanlehn, 2002, p. 18) tells us the prior/posterior probabilities of all nodes in the task-specific part (also solution graph) in Bayesian network.



**Figure I.3.14.** Prior/Posterior probabilities in the task-specific part

### I.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  $\langle Cr, Cs \rangle$  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.

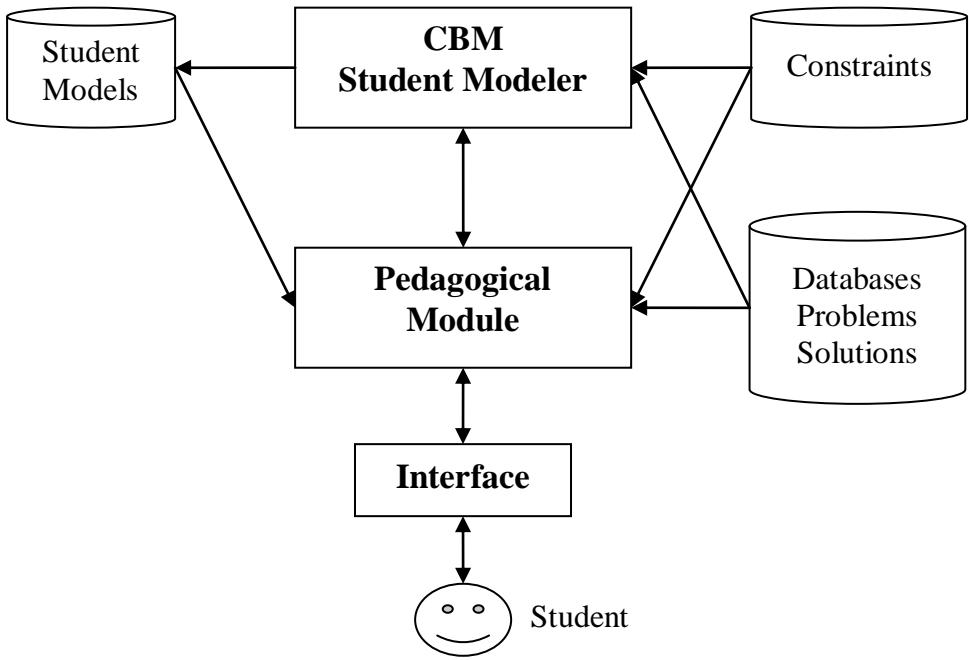
If student’s solution is matched with both  $Cr$  and  $Cs$ , the constraint  $\langle Cr, Cs \rangle$  is *satisfied* for this solution. If only  $Cr$  matches the solution, we call that the constraint is *relevant* to solution. If both  $Cr$  and  $Cs$  don’t match this solution, the constraint is *violated*. Following is the matching process (Mayo, 2001, p. 71):

```
If matches(student-solution, Cr) Then
    If not_matches(student-solution, Cs) Then
        constraint-is-violated;
    Else
        constraint-is-satisfied;
    End If
Else
    constraint-is-relevant;
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 I.3.15) has three functional models: CBM student modeler, pedagogical module and interface (Mitrovic, 1998, p. 309).



**Figure I.3.15.** 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 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 give some pedagogical decisions (instructions) that helps students to improve their 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 students solve 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 representing 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 ( $mastered_1, mastered_2, \dots, mastered_n$ ) where  $mastered_c$  ( $c = \overline{1, n}$ ) expresses whether the constraint  $c$  is mastered ( $mastered_c=1$ ) by user or not ( $mastered_c=0$ ).  $P(mastered_c=1)$  is the certain probability that student masters constraint  $c$ . The initial value of  $P(mastered_c=1)$  is the ratio of the frequency

that constraint  $c$  is satisfied to the frequency that constraint  $c$  is relevant in the past.

$$P_0(\text{mastered}_c = 1) = \frac{\text{The frequency that } c \text{ is satisfied}}{\text{The frequency that } c \text{ is relevant}}$$

After student solves her/his problem and receives the feedback from pedagogical module, the probability  $P(\text{mastered}_c=1)$  is updated according to following heuristic rules (Mayo, 2001, p. 94):

- If constraint  $c$  is satisfied then  $P(\text{mastered}_c=1)$  increases by 10% of the value  $1-P(\text{mastered}_c=0)$ .
- If constraint  $c$  is violated and no feedback about  $c$  is given then  $P(\text{mastered}_c=1)$  decreases by 20%.
- If constraint  $c$  is violated but feedback is given about  $c$  then  $P(\text{mastered}_c=1)$  increases by 20% of the value  $1-P(\text{mastered}_c=1)$ .

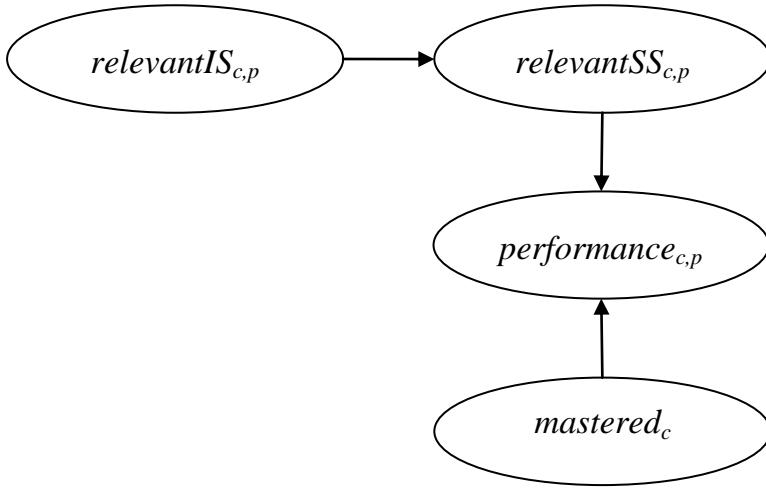
Instead of using such rules, the author (Mayo, 2001, p. 94) proposes another method which applies Bayesian inference (Wikipedia, Bayesian inference, 2006) to update the probability that constraint is mastered. Let  $M$  denote student's mastery of constraint and let  $L$  denote the outcome of the last 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 I.3.3.1 to compute  $P(M/L)$ .

$$P(M = m|L = l) = \frac{P(L = l|M = m)P(M = m)}{P(L = l|M = 1)P(M = 1) + P(L = l|M = 0)P(M = 0)}$$

#### Formula I.3.3.1. Posterior probability of student's mastery

Where  $m, l \in \{0, 1\}$  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  $\text{mastered}_c$  be the binary expressing whether student masters constraint  $c$ . The binary  $\text{relevantIS}_{c,p} \in \{0, 1\}$  expresses whether constraint  $c$  is relevant to the ideal solution of problem  $p$ . The binary  $\text{relevantSS}_{c,p} \in \{0, 1\}$  expresses whether constraint  $c$  is relevant to the student's solution to problem  $p$ . That variable  $\text{relevantSS}_{c,p}$  depends on  $\text{relevantIS}_{c,p} \in \{0, 1\}$  implicates that the student's solution must match the ideal solution. The variable  $\text{performance}_{c,p}$  having three values *satisfied*, *violated* and *not-relevant* denotes the performance of student given the problem  $p$  on constraint  $c$ . The variable  $\text{performance}_{c,p}$  depends on both  $\text{relevantSS}_{c,p}$  and  $\text{mastered}_c$ . The Bayesian network representing these variables and relationships among them is shown in figure I.3.16 (Mayo, 2001, p. 98).



**Figure I.3.16.** Bayesian network in SQL-Tutor

As discussed, the prior probability of *mastered<sub>c</sub>*,  $P_0(\text{mastered}_c=1)$ , is defined by Bayesian inference (Wikipedia, Bayesian inference, 2006) or heuristic rule. The prior probability of *relevantIS<sub>c,p</sub>* is specified by expert. According to (Mayo, 2001, p. 98), the conditional probability table (CPT) of *relevantSS<sub>c,p</sub>* given *relevantIS<sub>c,p</sub>* is defined as in table I.3.2.

		<i>relevantIS<sub>c,p</sub></i>	
		<i>yes</i> (1)	<i>no</i> (0)
<i>relevantSS<sub>c,p</sub></i>	<i>yes</i> (1)	$\alpha_c$	$\beta_c$
	<i>no</i> (0)	$1-\alpha_c$	$1-\beta_c$

**Table I.3.3.** Conditional probability table of *relevantSS<sub>c,p</sub>* given *relevantIS<sub>c,p</sub>*

According to (Mayo, 2001, p. 98), the parameter  $\alpha_c$  ( $\beta_c$ ) 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  $\alpha_c$ ,  $\beta_c$  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  $\alpha_c$  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  $\beta_c$  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.

$$\alpha_c = \frac{\text{Frequency that } c \text{ relevant to both student's solution and ideal solution}}{\text{Frequency that } c \text{ relevant to ideal solution}}$$

$$\beta_c = \frac{\text{Frequency that } c \text{ relevant to student's solution but not relevant to ideal solution}}{\text{Frequency that } c \text{ not relevant to ideal solution}}$$

### I.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, 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).

$$I(X, Y) = \sum_{x,y} P(x, y) \log \left( \frac{P(x, y)}{P(x)P(y)} \right)$$

Note that notation  $\log(\cdot)$  denotes natural logarithm function and  $x, y$  (s) are possible instances of  $X, Y$ , respectively. Note that *notation  $P(\cdot)$  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).

$$I(X, Y|C) = \sum_{x,y,c} P(x, y, c) \log \left( \frac{P(x, y|c)}{P(x|c)P(y|c)} \right)$$

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

Given the threshold  $\zeta$ , if the conditional mutual information  $I(X, Y|C)$  is smaller than  $\zeta$  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 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  $X \rightarrow Z \leftarrow Y$ ) 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  $S$  and the threshold  $\zeta$ , 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  $\zeta$  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 this chapter gave us a survey of user model, adaptive learning system along with methods and systems that construct user model. It is easy to recognize that user model is the heart of adaptive system; hence, user model and user modeling system are the main subjects in this research. The next chapter [II](#) gives an overview of my main work, which is to propose a learner model and a user modeling system that manipulates such learner model (see section [II.2](#)). Chapter [III](#) will describe my works in the most detailed.

## Chapter II. Zebra: A User Modeling System for Triangular Learner Model

The core of adaptive system, as aforementioned in previous section I.1, is the user model that is the representation of information about an individual. User model is necessary for an adaptive system to provide the adaptation effect, i.e., to behave differently for different users. The system that collects user information to build up user model and reasons out new assumptions about user is called user modeling system (UMS). Based on the survey of user model and adaptive learning mentioned in previous chapter I, there are two main tendencies towards implementing UMS: domain-dependent UMS and domain-independent UMS. The latter is called generic UMS known widely but my approach focuses on the domain-dependent UMS applied into adaptive e-learning especially. The reason is that domain-independent UMS is too generic to “cover” all learners’ characteristics in e-learning, which may cause unpredictable bad consequences in adaptation process. Note that user is considered as learner in e-learning context. Many users’ characteristics can be modeled but each characteristic is in accordance with respective modeling method. It is impossible to model all learners’ characteristics in the same UMS because of such reason “there is no modeling method fit all characteristics”.

To overcome these obstacles and difficulties, I propose the new model of learner “Triangular Learner Model (TLM)” composed by three main learners’ characteristics: knowledge, learning style and learning history. TLM with such three underlying characteristics will cover the whole of learner’s information required by learning adaptation process. The UMS which builds up and manipulates TLM is also described in detail and named Zebra (Nguyen, ZEBRA: A new User Modeling System for Triangular Model of Learners’ Characteristics, 2009). I also propose the new architecture of an adaptive application and the interaction between such application and Zebra.

Before discussing main topic, we should glance over existing UMS (s) in section II.1. Section II.2 described the Zebra – my modeling system in detailed. Section II.3 is the conclusion.

### II.1. Existing user modeling systems

User modeling system (UMS) is defined as the system that collects user information to build up user model and reason out new assumptions about user. UMS (s) have a long evolutionary process from early user modeling systems embedded into specific application to user modeling shells and user modeling servers which are separated from adaptive system and communicate with adaptive system according to client-server architecture. Note that the term “UMS” indicates both user modeling shell and user modeling server in this section II.1.

### **II.1.1. Early user modeling systems**

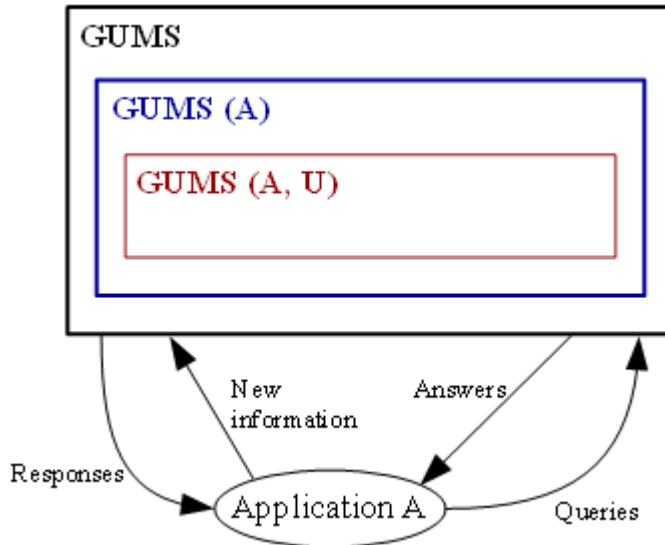
Early UMS (s) concentrate on question-answer (dialog) system and human-computer interaction. They are components embedded in concrete application. An example for such dialog system is GRUNDY developed by author (Rich, 1979) in her PhD thesis. GRUNDY plays the role of book recommender in the library when it calculates recommendation of books, based on assumptions about users' personal traits. Such traits which are educational and intellectual level, preference for thrill, fast-moving plots or romance, tolerance for descriptions of sexuality, violence and suffering, etc. are represented as user model (Fröschl, 2005, p. 59). GRUNDY use stereotype method (Rich, 1979) to build up user model, based on users' answers to questions during their first usage of system. For example, if user has a male first name, GRUNDY infers a high sex tolerance and a low one for romance.

### **II.1.2. User modeling shells**

There is a need for developing UMS (s) as separated components whose functionality is not dependent on any adaptive application. Such UMS (s) are called user modeling shell. The term “shell” is borrowed from the field of expert system; thereby, the purpose of shells is to separate user modeling functionality from adaptive application.

User modeling shell goes towards generic purpose but it is not totally independent on application and often integrated into application when it is deployed. Examples of user modeling shell are GUMS, UMT, PROTUM, TAGUS, um.

**GUMS** is the abbreviation of “General User Modeling System” developed by authors (Finin & Drager, 1986). It is a first modeling shell that abstracts information about user by allowing defining the simple stereotype (Rich, 1979) hierarchies in form a tree of structure. Each stereotype is associated facts and rules describing system's reasoning about it (Fröschl, 2005, p. 60). Users are classified into such stereotypes. The initial stereotype of user is assigned by system and can be altered later according to observations about her/him. Both GUMS and GRUNDY apply stereotype method into modeling user. Moreover GUMS interacts with specific application by storing facts that application provides and answering any queries of application concerning assumptions about user. GUMS ensures the consistency of new facts with old assumptions about user in user model. If any inconsistency takes place, GUMS will inform to application by a response (see figure II.1) (Finin & Drager, 1986, p. 225).



**Figure II.1.** The architecture of GUMS

Where A, GUMS(A) and GUMS(A, U) denote a application A, modeling system for application A, and model for user U in application A, respectively (Finin & Drager, 1986, p. 225).

**UMT** (User Modeling Tool) developed by authors (Brajnik & Tasso, 1994) models information about users as stereotypes which contain their assumptions in form of attribute-value pairs (Brajnik & Tasso, 1994, p. 41). UMT allows developers/specialists to define hierarchical user stereotypes, triggers, rules for user model inferences and contradiction conditions (Kobsa, 2001, p. 50). The first time user interacts with system, trigger is activated to assign her/his into an initial stereotype. The assumptions of initial stereotype are added into user model by applying inference rules. Some of these assumptions can be retracted whenever contradictions occur.

**PROTUM** (PROlog based Tool for User Modeling) developed by author (Vergara, 1994) is written by the functional language PROLOG, please see (Wikipedia, Prolog, 2014) for surveying language PROLOG. PROTUM is more powerful than UMT although it uses stereotype method to model user like UMT did because its hierarchy of stereotypes is not limited to tree structure and assumptions about users are not based on attribute-value pairs like UMT (Fröschl, 2005, p. 63). Moreover PROTUM determines the activation rates of triggers in order to activate or deactivate stereotypes which have already assigned to user. These rates are used to resolve conflicts between inconsistent assumptions of two activated stereotypes (Fröschl, 2005, p. 63).

**TAGUS** developed by authors (Paiva & Self, 1995) also applies stereotype method into modeling user. Thus, it allows defining the hierarchical stereotype but each assumption about user in stereotype is represented in first-order formulas with meta-operators expressing the type of assumption such as belief, and goal (Kobsa, 2001, p. 51). TAGUS also has inference mechanism and truth

maintenance discovering contradictions of assumptions and diagnosing unexpected user's behaviors (Kobsa, 2001, p. 51).

**um** developed by author (Kay, 1995) aims to provide the library of user modeling functionalities in which assumptions about users' knowledge, goal, background, etc. are represented in attribute-value pairs. So um is often considered as um toolkit. User model is composed of pieces of information called as components accompanied by the set of evidences for user verification. There are four types of components: preferences, knowledge, beliefs, and attributes. Each evidence consists of four parts: reliability class, source identifier, part identifier, and timestamp (Kay, 1995, pp. 166-168). Source identifier refers to the computer program that produces evidence. Part identifier is the sub-part or sub-function of such program. Timestamp tells us when evidence is produced or collected. Reliability class is used to classify evidences and there are five types of reliability class such as observation, stereotype activation, rule invocation, user input and told to the user (Kobsa, 2001, p. 51). So um combines stereotype method and rule-based method in implementation.

### II.1.3. User modeling servers

User modeling server has the same purpose with user modeling shell when both of them aim to separate user modeling functionality from adaptive system. However user modeling server is totally independent on application. It is not integrated into applications and interacts with applications through inter-process communication (Kobsa, 2007, p. 3). It can reside on the different site from application's site and serve more than one instance of application at the same time. The communication between user modeling server and adaptive application is based on client-server architecture in which modeling server is responsible for answering application's requests. Examples of user modeling server are BGP-MS, Doppelgänger, CUMMULATE, Personis.

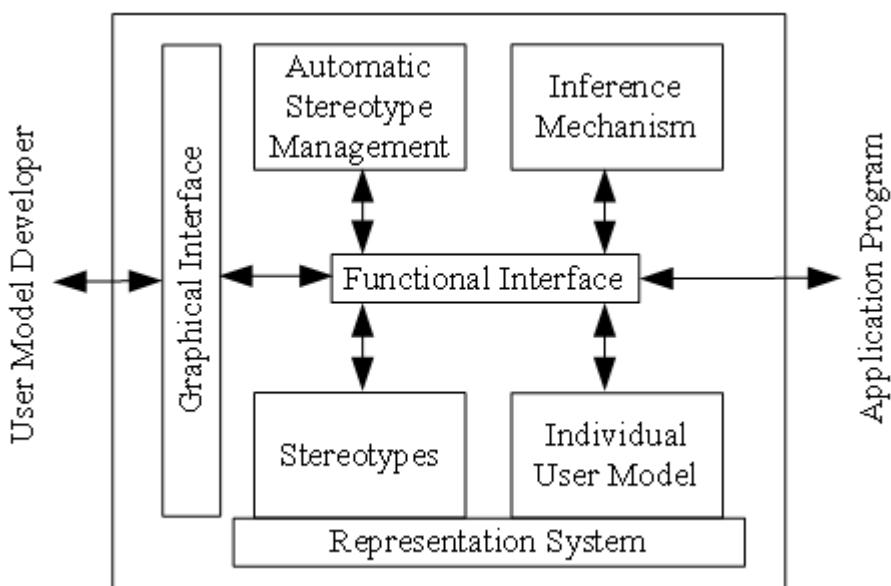
**BGP-MS** developed by authors (Kobsa & Pohl, 1995) is the user modeling server taking interest in user's knowledge, belief and goal. Note that the name BGP-MS is abbreviation of Belief, Goal and Plan Maintenance System (Kobsa & Pohl, 1995, p. 4). Although BGP-MS is really a user modeling shell, it can be considered as user modeling server because communication protocol inside BGP-MS allows it to be deployed as network server which responses many adaptive applications (Kobsa, 2007, p. 5). BGP-MS receives user's observations provided by adaptive application and processes internal operations of classification and calculation based on these observations (Fröschl, 2005, p. 63). BGP-MS uses stereotype method, natural language dialogs and questionnaires to build up user model. BGP-MS has four essential components (Fröschl, 2005, p. 64):

- *Individual user model* contains assumptions about user.
- *Stereotype* component manages the hierarchy of stereotypes. Both *stereotypes* and *individual user model* are based on the *representation*

*system* which is the integrated suite of knowledge representation mechanism for representing assumptions about user. Representation system is based on the conceptual knowledge representation language SB-ONE (Kobsa & Pohl, 1995, p. 16). SB-ONE has two levels such as general level and individualized level. In the general level, representational elements are general concepts and general attribute descriptions. In the individualized level, assumptions about user are represented based on using individualized concepts and individualized attribute descriptions which are linked to respective general concepts and general attribute descriptions specified in the general level (Kobsa, 1991, p. 71).

- *Automatic stereotype management* is responsible for the activation and deactivation of assigned stereotypes of an individual user model (Fröschl, 2005, p. 64).

These main components communicate together through the *functional interface*. Developer interacts with BGP-MS by the *graphic interface*. The architecture of BGP-MS is represented in figure II.2 (Fröschl, 2005, p. 64).



**Figure II.2.** The architecture of BGP-MS

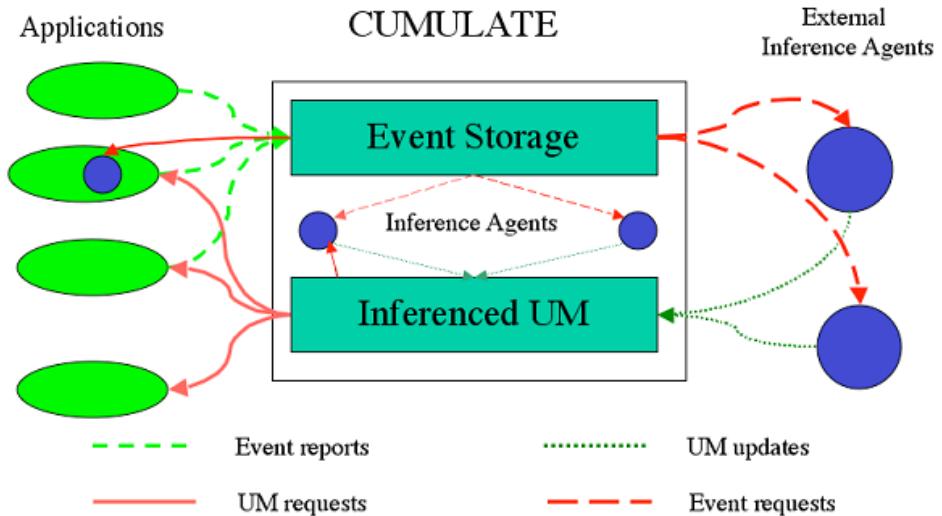
As seen in figure II.2, the adaptive system communicates with main components through functional interface and BGP-MS defines the new communication protocol called KN-IPCMS where KN-IPCMS is abbreviation of KoNstanz Inter-Process Communication Management System (Kobsa & Pohl, 1995, p. 12).

**Doppelgänger** developed by author (Orwant, 1991) is the server that monitors users' actions and discovers patterns from these actions. Basing on such patterns, Doppelgänger aims to deliver user a personalized daily newspaper; it provides news in which user can be interested. The architecture of

Doppelgänger is split into two levels: sensor level and sever level (Fröschl, 2005, pp. 66-68).

- *Sensor level.* There are sensors having responsibility for gathering information about user. Sensors can be either software or hardware. There are specific techniques inside sensors in order to extract valuable information from users' actions. Each sensor has its own specific purpose, for example, one gathers amount of time of computer use and another tracks user's physical location (Fröschl, 2005, p. 66). User model is stored as LISP-like list, for example, a piece of user model expressing “it is likely that user Orwant prefers to read the news topic *Olympics*” is described as follows: “(*object orwant news (object preferences (assertion (likes topic olympics (strength 0.8) (probability 0.9)) (confidence 0.8) (technique beta))...)*)...” (Orwant, 1995, p. 6). Please see (Wikipedia, Lisp (programming language), 2014) for surveying programming language LISP.
- *Server level.* Doppelgänger makes inferences on information provided by sensors. For example, when sensors provide events such as “user often cares about stock market index in the evening and he usually reads sport news after lunch”, Doppelgänger can predict user's habit and tell the news recommendation system such habit. The author (Orwant, 1995, pp. 10-23) uses three learning techniques for inference tasks such as predicting user's preference by beta distribution, modeling time event by linear prediction, and predicting physical location by Markov model. Markov model is introduced in (Fosler-Lussier, 1998) and (Schmolze, 2001).

**CUMMULATE** developed by authors (Brusilovsky, Sosnovsky, & Shcherbinina, 2005) is a generic student-modeling server in which information about user is represented on two levels: *event storage* and *inferred user model*. Student actions being monitored are sent to event storage by a standard HTTP-based event-reporting protocol with note that HTTP protocol is introduced in (Wikipedia, Hypertext Transfer Protocol, 2014). Such actions are considered events. CUMMULATE adds a timestamp to each event and stores it permanently in event storage (Brusilovsky, Sosnovsky, & Shcherbinina, 2005, p. 2). The event storage allows several *inference agents* that process events in different ways and convert these events into the inferred user model, for example, some agents monitor user's knowledge and others predict user's interests. The architecture of CUMMULATE is open to a variety of *external inference agents* that receive requests from event storage and manipulate user model. CUMMULATE is really a combination of event-driven approach and agent-based approach, which is prominent in distributed e-learning environment. Figure II.3 shows its architecture (Brusilovsky, Sosnovsky, & Shcherbinina, 2005, p. 3) (Brusilovsky, 2004, p. 5).

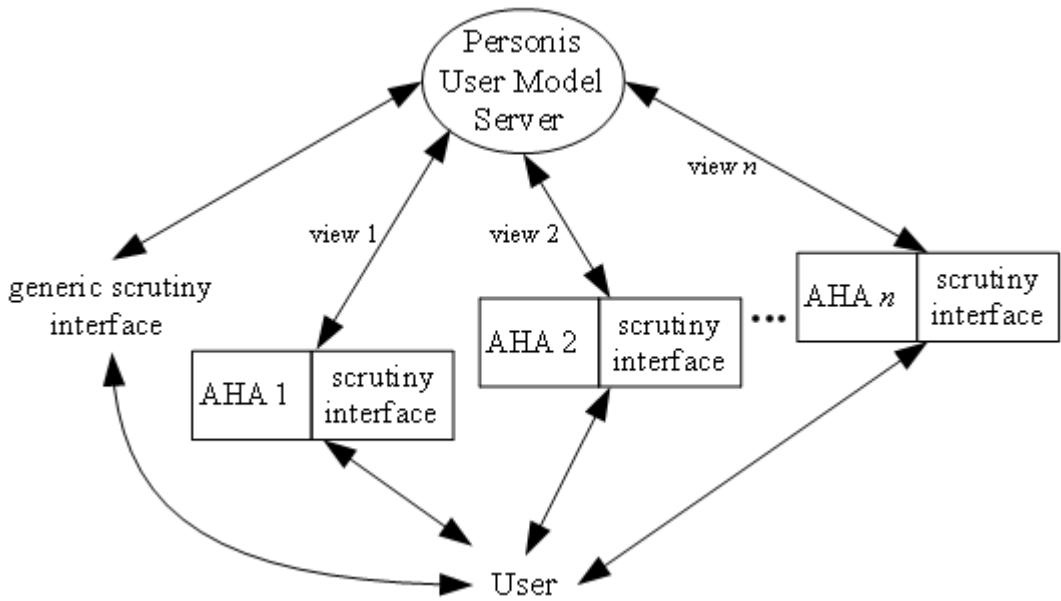


**Figure II.3.** The architecture of CUMMULATE

**Personis** developed by authors (Kay, Kummerfeld, & Lauder, 2002) is the modeling server whose considerable feature is to allow user to control and scrutinize her/his model (Kay, Kummerfeld, & Lauder, 2002, p. 203) (Fröschl, 2005, p. 68). Such user model is called scrutable user model. Personis is based on um toolkit but more complicated than um toolkit. The high-level architecture of Personis is divided into four parts such as *the server itself*, *generic scrutiny tools*, *adaptive hypermedia applications*, and *views* (Kay, Kummerfeld, & Lauder, 2002, pp. 205-207).

- The *server itself* is responsible for managing user model.
- A set of *generic scrutiny tools* allow users to see and manipulate their own user models. Users interact with these tools through a generic scrutiny interface. This interface is application-independent.
- A set of *adaptive hypermedia applications* (AHA) are denoted  $AHA_1, AHA_2, \dots, AHA_n$ . Each AHA includes two parts: first, the core enables user to do some learning tasks and second, the scrutiny interface associated to the core enables the core to interact with scrutiny tools. This interface is similar to generic scrutiny interface except that it belongs to the context of AHA.
- There is a set of *views* of user model; each view is available to each AHA and responsible for defining the components used by such AHA with note that components are parts of user model (Kay, Kummerfeld, & Lauder, 2002, p. 206). Applications can use either the same or different views.

Figure II.4 shows the high-level architecture of Personis.



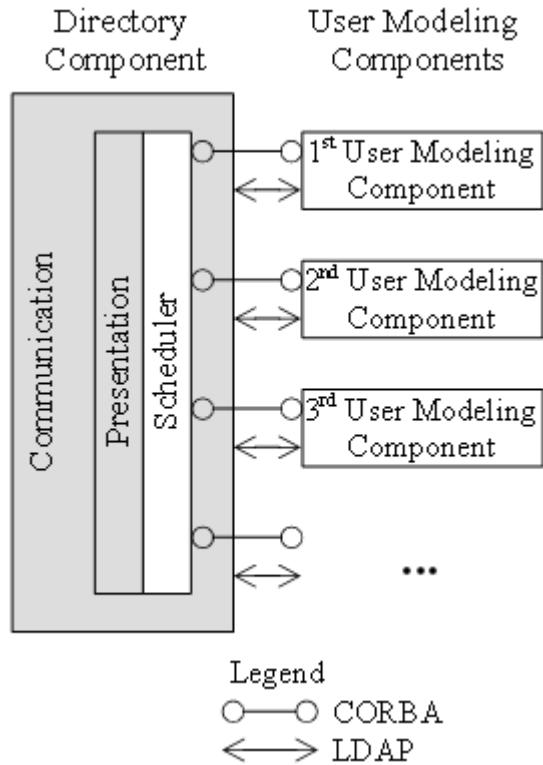
**Figure II.4.** The architecture of Personis

**LDAP-UMS** is the user modeling servers based on Lightweight Directory Access Protocol (LDAP), published by (Fink, 2004) in his PhD thesis. It focuses on data handling and enhances user modeling functionality by enabling user model to achieve “pluggable” feature. UMS uses a directory structure of LDAP to manage user’s information spreading across a network. Please see (Wikipedia, Lightweight Directory Access Protocol, 2014) for surveying LDAP. The architecture of LDAP-UMS inherits the LDAP directory server; so, UMS is composed of several pluggable user modeling components and can be accessed by external clients. The core of architecture is the *Directory Component* attached its three sub-components: *Communication*, *Representation* and *Scheduler* (Fink, 2004, pp. 75-76).

- *Communication* sub-component handles the communication between external clients and *Directory Component*, between *Directory Component* and *User Modeling Components*. Each *User Modeling Component* is dedicated to perform user modeling tasks such as collecting user information and inferring new assumptions about user. The *Directory Component* and *User Modeling Components* interact together via CORBA (Wikipedia, Common Object Request Broker Architecture, 2014) and LDAP.
- *Representation* sub-component is responsible for managing the directory contents.
- *Scheduler* sub-component is responsible for wrapping the underlying LDAP server with a component interface and harmonizing different sub-systems with *User Modeling Components* (Fink, 2004, p. 75).

This architecture allows developer to add more self-developed *User Modeling Components* to LDAP-UMS. So LDAP-UMS is the open and flexible user modeling server, which becomes powerful one among modern user modeling

servers. Figure II.5 shows the architecture of LDAP-UMS (Fink, 2004, p. 75) (Fröschl, 2005, p. 71).



**Figure II.5.** The architecture of LDAP-UMS

Now existing user modeling systems were introduced briefly in this section II.1 and almost of them aims to serve requests for generic user information. The next section II.2 is to propose a user modeling system which is designed and implemented for specific situation in which users (main modeled objects) are learners in learning environment such as web sites in order to gain high performance and precise reasoning for new information from learner model.

## II.2. Zebra: A User Modeling System for Triangular Learner Model

Existing user modeling systems (UMS) develop fast in recent years; they are trending towards servers that give support to adaptive applications with fully response to queries about user information available in user model. However I recognize that generic UMS (s) described in previous section II.1 are too generic to describe all fine characteristics of user when she/he is learner in e-learning context. Especially in situation that our research focuses on domain of e-learning, such UMS (s) are proved to be less effective in providing assumptions about user to adaptive learning applications. In learning environment, users who play role of learners must be modeled by special method. The content of learner model can be divided into two categories: domain specific information and domain independent information. **Domain specific information** is knowledge that learner achieved in certain subjects. Otherwise, **domain independent information** includes personal traits not related to domain knowledge such as interests, learning styles, and demographic information. Each kind of information is in accordance with respective modeling method. For example, knowledge model can be created by overlay method, which called **overlay model**. So it is impossible to model all learners' characteristics because of the reason "there is no modeling method fit all characteristics". Moreover, almost UMS (s) require effective inference techniques in their modeling tasks but this is impossible if we cannot recognize which individual characteristics are important.

To overcome these obstacles and difficulties, I propose the new learner model that contains three most important characteristics of user: knowledge (K), learning styles (LS) and learning history (LH). Such three characteristics form a triangle; so my model is called **Triangular Learner Model (TLM)**. TLM with three underlying characteristics will cover the whole of user's information required by learning adaptation process. The reasons for such assertion are:

- Knowledge, learning styles and learning history are prerequisite for modeling learner.
- While learning history and knowledge change themselves frequently, learning styles are relatively stable. The combination of them ensures the integrity of information about learner.
- User's knowledge is domain specific information and learning styles are personal traits. The combination of them supports UMS to take full advantages of both domain specific information and domain independent information in user model.

I also introduce the architecture of UMS which builds up TLM; it is named **Zebra**. The name "Zebra" implicates that my UMS will run fast and be powerful like African zebra (Wikipedia, Zebra, 2014).

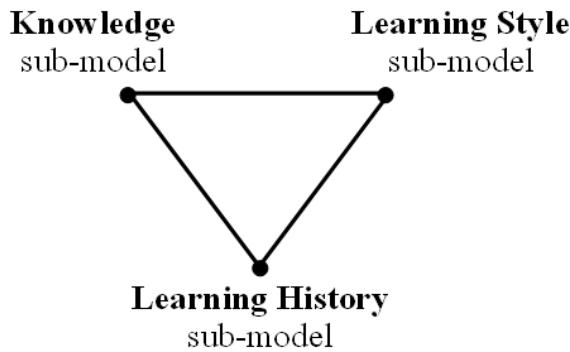
This section II.2 includes four sub-sections as follows:

- Sub-section II.2.1 describes TLM in detailed.
- Sub-section II.2.2 describes the architecture of Zebra.

- Sub-section [II.2.3](#) tells us the interaction between Zebra and adaptive learning systems like AHS and AEHS.
- Because Zebra is implemented as computer software, sub-section [II.2.4](#) aims to the implementation of Zebra.

### **II.2.1. Triangular Learner Model**

Triangular Learner Model (TLM) is constituted of three basic features of user: knowledge, learning styles and learning history which are considered as three apexes of a triangle (see figure [II.6](#)). Hence TLM has three sub-models: *knowledge sub-model*, *learning style sub-model* and *learning history sub-model*.



**Figure II.6.** Triangular Learner Model

I propose the method that combines overlay method and Bayesian network to build up **knowledge sub-model**. In overlay method, the domain is decomposed into a set of knowledge elements and the overlay model (namely, user model) is simply a set of masteries over those elements. The Bayesian network (see sub-section [III.1.1.1](#)) is the directed acyclic graph (DAG) in which nodes are linked together by arcs; each arc expresses the relationship between two node. The strengths of relationships are quantified by Conditional Probability Table (CPT). The combination between overlay model and Bayesian network is done through following steps:

1. The structure of overlay model is translated into Bayesian network, each user knowledge element becomes a node in Bayesian network.
2. Each relationship between domain elements in overlay model becomes a conditional dependence assertion signified by CPT of each node in Bayesian network.

So knowledge sub-model is called as Bayesian overlay sub-model, which is described particularly in section [III.1](#). The proposed approach that combines overlay model and Bayesian network so as to build up Bayesian overlay sub-model is described particularly in sub-section [III.1.1](#).

As aforementioned in sub-section [I.1.1.2](#), **learning styles** are defined as “the composite of characteristic cognitive, affective and psychological factors that serve as relatively stable indicators of how a learner perceives, interacts

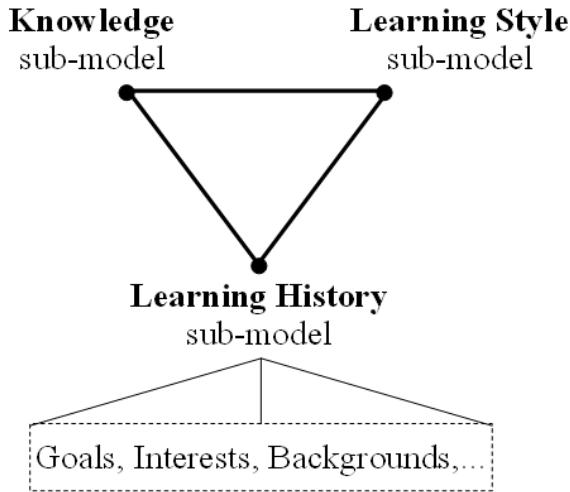
with and responds to the learning environment” (Stash, 2007, p. 93). There are many models of learning styles in theory of psychology such as Dunn and Dunn (Dunn & Dunn, 1996), Witkin (Witkin, Moore, Goodenough, & Cox, 1977), Riding (Riding & Rayner, 1998), Myers-Briggs (Wikipedia, Myers-Briggs Type Indicator, 2014), Kolb (Kolb & Kolb, 2005), Honey-Mumford (Honey & Mumford, 2000), and Felder-Silverman (Felder & Silverman, 1988). I choose Honey-Mumford model and Felder-Silverman model as principal models and construct them by hidden Markov model (please see sub-section [III.2.4](#) for more details about hidden Markov model). According to Honey-Mumford model and Felder-Silverman model, learning styles are classified into following dimensions:

- *Verbal/Visual.* Verbal students like learning materials in text form. Otherwise visual students prefer to images, pictures, video, etc.
- *Active/Reflective.* Active students understand information only if they discussed it and applied it. Reflective students think thoroughly about things before doing any practice.
- *Theorist/Pragmatist.* Theorists think things through in logical steps, understand different facts into coherent theory (Stash, 2007, p. 106). Pragmatists have practical mind, prefer to try and test techniques relevant to problems (Stash, 2007, p. 106).

For modeling learning style using Hidden Markov Model (HMM), we must define states, observations and the relationship between states and observations in context of learning style. So each learning style is now considered as a state. The essence of state transition in HMM is the change of user’s learning style, thus, it is necessary to recognize which learning styles are most suitable to user. After monitoring users’ learning process, we collect observations about them and then discover their styles by using inference mechanism in HMM. So learning style sub-model is modeled as HMM, which is described particularly in section [III.2](#).

The last sub-model stores and manipulates learner’s **learning history** in form of XML data files (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008). *All learners’ actions: learning materials access, duration of computer use, doing exercise, taking an examination, doing test, communicating with teachers or classmates, etc. are logged in learning history sub-model.* School reports also recorded in this sub-model. We consider this sub-model as a feature of learners because every student has individual learning process in her/his life and the data about such learning process are recorded as pieces of information in learning history sub-model. Information in this sub-model is necessary for data mining in e-learning to discover not only knowledge and learning styles but also other learners’ characteristics such as interests, backgrounds, and goals. The mining engine in the core of Zebra often uses this sub-model for many mining tasks. For this reason, this sub-model is drawn as the apex at the bottom of triangle in architecture of TLM. This implicates that learning history sub-model is the most important sub-model in TLM when it is considered as the basic of two other sub-models. Figure [II.7](#)

shows the extended TLM in which learning history sub-model is the root for attaching more learners' characteristics such as interests, backgrounds, and goals to TLM.



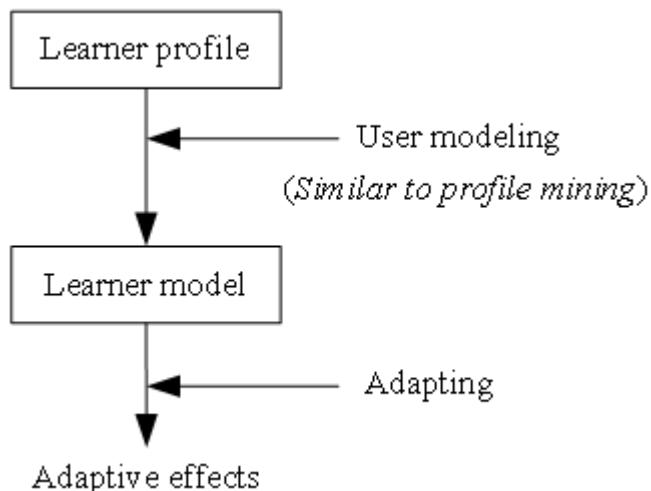
**Figure II.7.** extended Triangular Learner Model

Learning history sub-model is described particularly in section III.3.

The successive sub-section II.2.2 describes the architecture of Zebra.

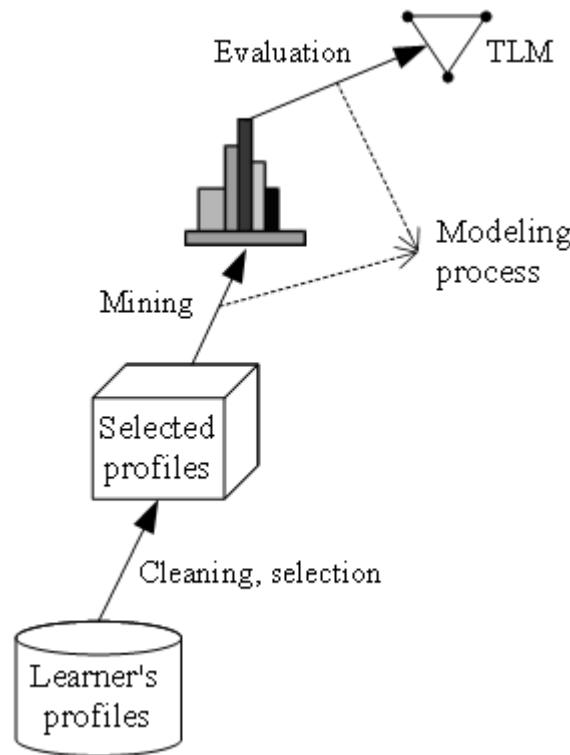
### II.2.2. The architecture of Zebra

The essence of user modeling systems is mining user's profile to discover valuable patterns in form of user's features. These features which are personal traits or characteristics in learning context navigate adaptive applications to give support to user in her/his learning path. As aforementioned, the user modeling system that manipulates TLM is called Zebra. The purpose of Zebra is to mine user's learning profile to build up her/his TLM. Hence Zebra has the inside *mining engine*. Figure II.8 indicates that user modeling (Brusilovsky, 1996, p. 2) is similar to profile mining.



**Figure II.8.** Modeling task is similar to profile mining task

As aforementioned in section I.1, terms such as user model, student model and learner model have the same meaning because users are considered as learners or students in this research.



**Figure II.9.** The modeling process in Zebra

As seen in figure II.9, the goal of modeling process in Zebra is to mine users' profiles but this process gets higher level than traditional data mining when it results out TML which is the most structured and valuable knowledge about users.

Moreover Zebra must implement the powerful inference mechanism to reason learners' new assumptions (or characteristics) out TML. In next section III.1, I propose two methods: Bayesian network combined overlay model and hidden Markov model to infer learners' knowledge and learning styles. Both Bayesian network (see sub-section III.1.1.1) and Markov model (see sub-section III.2.4) are special cases of belief network. In general, belief network (Murphy, 1998) is directed acyclic graphs in which nodes represent variables, arcs signify direct dependencies between the linked variables, and the strengths of these dependencies are quantified by conditional probabilities. Belief network is the robust mathematical tools appropriate to reasoning based on evidences. Zebra must have another inside engine – the *belief network engine*.

Therefore, the core of Zebra is the composition of two engines: *mining engine* (ME) and *belief network engine* (BNE).

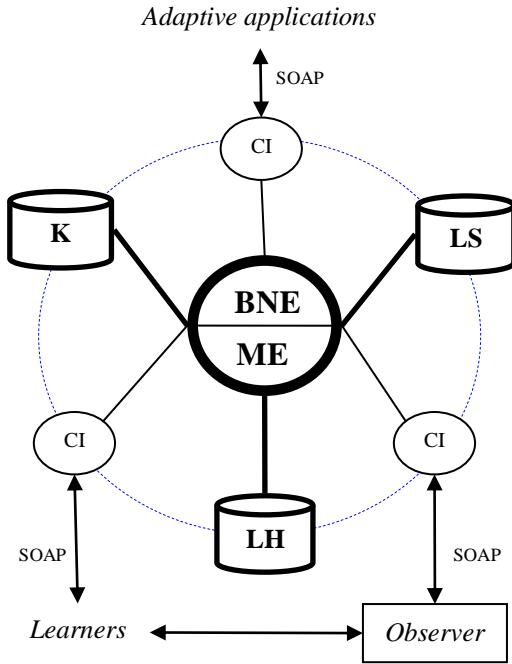
- **Mining engine (ME)** is responsible for collecting learners' data, monitoring their actions, structuring and updating TLM. Mining engine also provides important information to belief network engine; it is

considered as input for belief network engine. In short, mining engine creates TLM by applying mining algorithms, for example, it is possible to modeling user's learning path by using sequential pattern mining. Mining engine has three other important functionalities that are *to discover some other characteristics* beyond knowledge and learning styles (such as interests, goals, learning context) and *to support learning concept recommendation* and *to support collaborative learning*. The last functionality is the extension of Zebra.

- **Belief network engine (BNE)** is responsible for inferring new user information from TLM by using deduction mechanism available in belief network. This engine applies both Bayesian network and hidden Markov model into its tasks. Two sub-models: knowledge and learning style are managed by this engine.

Zebra also provides *communication interfaces (CI)* that allows users and adaptive systems to see or modify restrictedly their TLM. Adaptive applications also interact with Zebra by these interfaces. *Communication interfaces* can be implemented as web services used widely on internet. According to World Wide Web Consortium (<http://www.w3.org>), a web service is defined as a software system designed to support interoperable machine-to-machine interaction over a network (W3C, Web Services Architecture, 2004). It has an interface described in a machine-processable format such as WSDL (W3C, Web Services Description Language (WSDL) 1.1, 2001). Other systems interact with the web service in a manner prescribed by its description using SOAP messages (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000), typically conveyed using HTTP (Wikipedia, Hypertext Transfer Protocol, 2014) with an XML (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) serialization in conjunction with other web-related standards. When complying with web service standard, it is possible to publish CI (s) on internet for third-parties to communicate with Zebra more effectively.

There is external program so-called *observer* having responsibility for tracking learners' actions. Observer catches and delivers user observations to Zebra. Observer interacts with Zebra through CI.



**Figure II.10.** The architecture of Zebra

Figure II.10 depicts the architecture of Zebra in which ME and BNE denote mining engine and belief network engine, respectively. In addition K, LS and LH denote knowledge sub-model, learning styles sub-model and learning history sub-model, respectively. Finally, CI denotes communication interface.

Zebra is implemented as computer software available at internet link <https://sites.google.com/site/ngphloc/st/dissertations/zebra>. The next sub-section II.2.3 mentions the interaction between Zebra and adaptive applications.

### II.2.3. Interaction between Zebra and adaptive applications

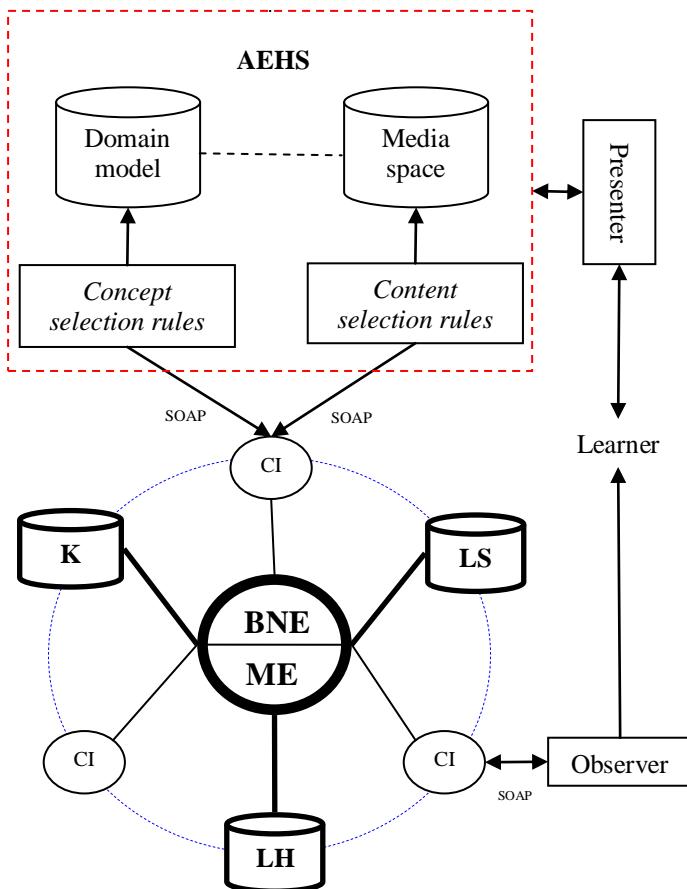
Zebra aims to support adaptive learning applications; so in this sub-section II.2.3 we should glance over what adaptive applications are and discuss about how Zebra interacts with such applications. As aforementioned in sub-section I.2.3, the most popular adaptive learning system supporting personalized learning environment is adaptive education hypermedia system (**AEHS**). Please see sub-section I.2.3 for more details about AEHS. Remind that the storage layer of AEHS has four models:

- *Media space or resource model*: contains learning resources (lectures, tests, examples, exercises, etc.) and associated descriptive information (metadata).
- *Domain model*: constitutes the structure of domain knowledge so-called domain model. Domain model was often represented in the form of graph.
- *Adaptation model*: is the centric component which gives effect to adaptation. It contains *content selection rules* and *concept selection rules*. We apply content selection rules into choosing suitable

educational resources from media space. On the other hand, concept selection rules are used to choose appropriate concept from domain. These rules must obey user model so that the selection gets correct.

- *User model*: information and data about user.

Here, AEHS is regarded as an example for illustrating prominent traits of adaptive learning system. I propose the new approach in which the *user model* is removed from AEHS and becomes the TLM managed by the user modeling system Zebra. Now AEHS owns only three components: resource model, domain model and adaptation model. The reason is that learner model (TLM) becomes too complex to be maintained by AEHS and AEHS should only focus on improving adaptation process and the performance of system will get enhanced when AEHS takes full advantage of functionalities of Zebra. All operations relating TLM are executed by Zebra instead of AEHS. AEHS interacts with Zebra via communication interfaces (CI) according to SOAP protocol (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000). Figure II.11 shows the new architecture of AEHS and the interaction between AEHS and Zebra. There are only three instances of CI but the number of them is not limited in practice.



**Figure II.11.** The new architecture of AEHS and the interaction between AEHS and Zebra

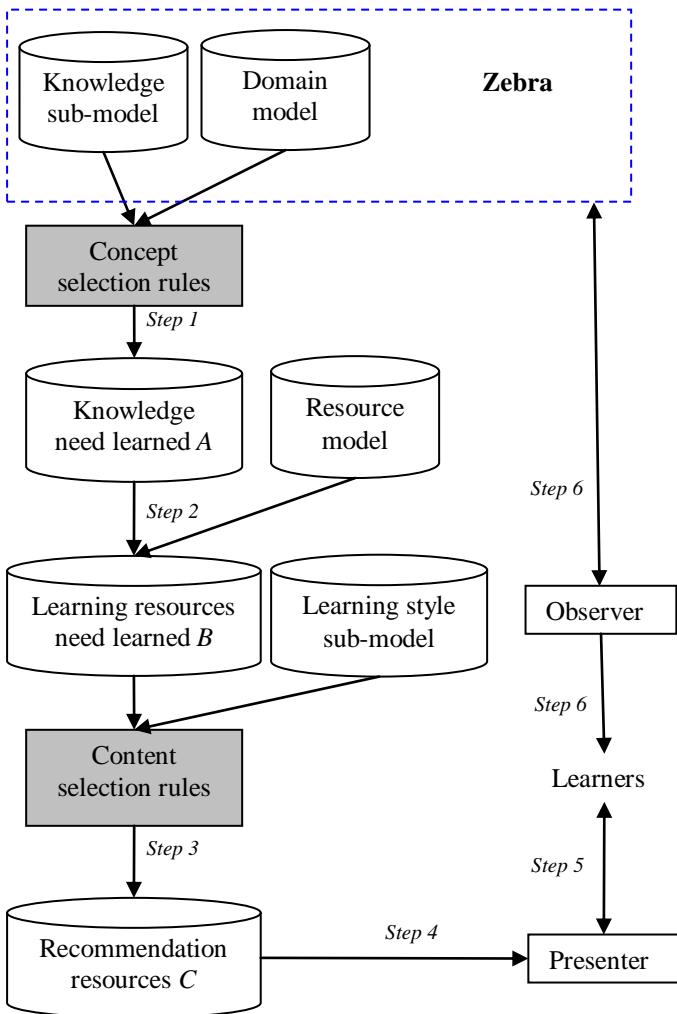
In figure II.11, *observer* is external program that tracks learners' actions and *presenter* is any application or web site that introduces or presents learning resources to learners.

The adaptation process performed by adaptation components includes two main sub-processes: concept selection process and content selection process.

- In *concept selection process*, concept selection rules are used to match learners' knowledge to concepts in domain model. In other words, these concepts are filtered to find ones that is necessary for learners to learn in their course.
- In *content selection process*, learning resources are selected from resource model based on content selection rules that match learners' learning styles to attributes of resources in resource space. In other words, this process finds the resources that learner preferred or suitable to learner.

The adaptation process shown in figure II.12 includes following steps:

1. *Step 1*: The projection of domain model onto knowledge sub-model by using concept selection rules results in a set of domain knowledge called *A* that student has to learn. This is concept selection process.
2. *Step 2*: *A* is used as filter to choose a set of learning resources called *B* that relating to *A*.
3. *Step 3*: The projection of *B* onto learning styles sub-model by using content selection rules results in a sub-set of learning resources (lectures, exercises, tests, etc.) called *C* tailoring to learner's preferences. This is content selection process. *C* is considered as a set of recommendation resources.
4. *Step 4*: *C* is shown in content presenter. Presenter can be human-machine interfaces, web sites, learning management system (LMS), teaching support applications, etc. Please see (Wikipedia, Learning management system, 2014) for more details about LMS.
5. *Step 5*: Learner studies *C* by interacting with content presenter.
6. *Step 6*: Observer monitors learners in order to catch and delivers learner's observations to Zebra. Zebra uses such observations to update TLM.



**Figure II.12.** Steps in adaptation process with support of Zebra

Because Zebra is implemented as computer software, sub-section [II.2.4](#) describes the implementation of Zebra.

#### II.2.4. Implementation of Zebra

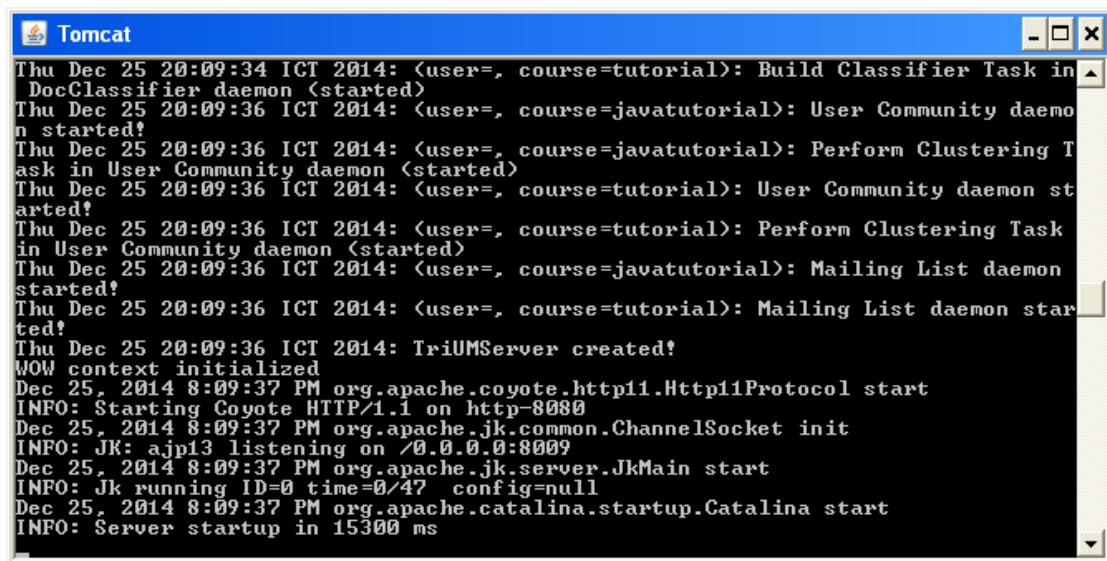
Zebra is implemented as a computer software working as a server (Nguyen, ZEBRA: A new User Modeling System for Triangular Model of Learners' Characteristics, 2009). Zebra server is written by Java language (Oracle) <https://www.oracle.com/java> that is a object-oriented programming language. The version of Java language when Zebra is built up is 1.6 updated 3. There is convention that the term “*Zebra server*” indicates the software server that implements Zebra. **Triangular Learner Model** (TLM), **belief network engine** (BNE), and **mining engine** (ME) are implemented as services or *daemons* inside Zebra server. Daemon (Wikipedia, Daemon (computing), 2014) is defined as a computer program running implicitly in an operating system such as Windows, Unix, and Linux. There are 9 main daemons: knowledge static Bayesian network daemon, knowledge dynamic Bayesian network daemon, learning style daemon, learning history data daemon, learning concept recommendation daemon, learning path daemon, discovering user interests

daemon (document classification + user searching), constructing user communities daemon, and mailing list daemon. Essentially, TLM and two engines (BNE and ME) are decomposed into such 9 daemons.

Learning history sub-model of TLM stores coarse information in XML files (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) or relation database (Ramakrishnan & Gehrke, 2003, pp. 25-94). Besides implementing Zebra architecture and TLM, Zebra server provides many utility tools for managing learners and supporting adaptive learning. Zebra server can be downloaded via internet link as follows:

<https://sites.google.com/site/ngphloc/st/dissertations/zebra>

When you startup Zebra server, it runs as underlying service as shown in figure II.13. Note that Zebra server runs inside Apache Tomcat web server (Apache, 1999).



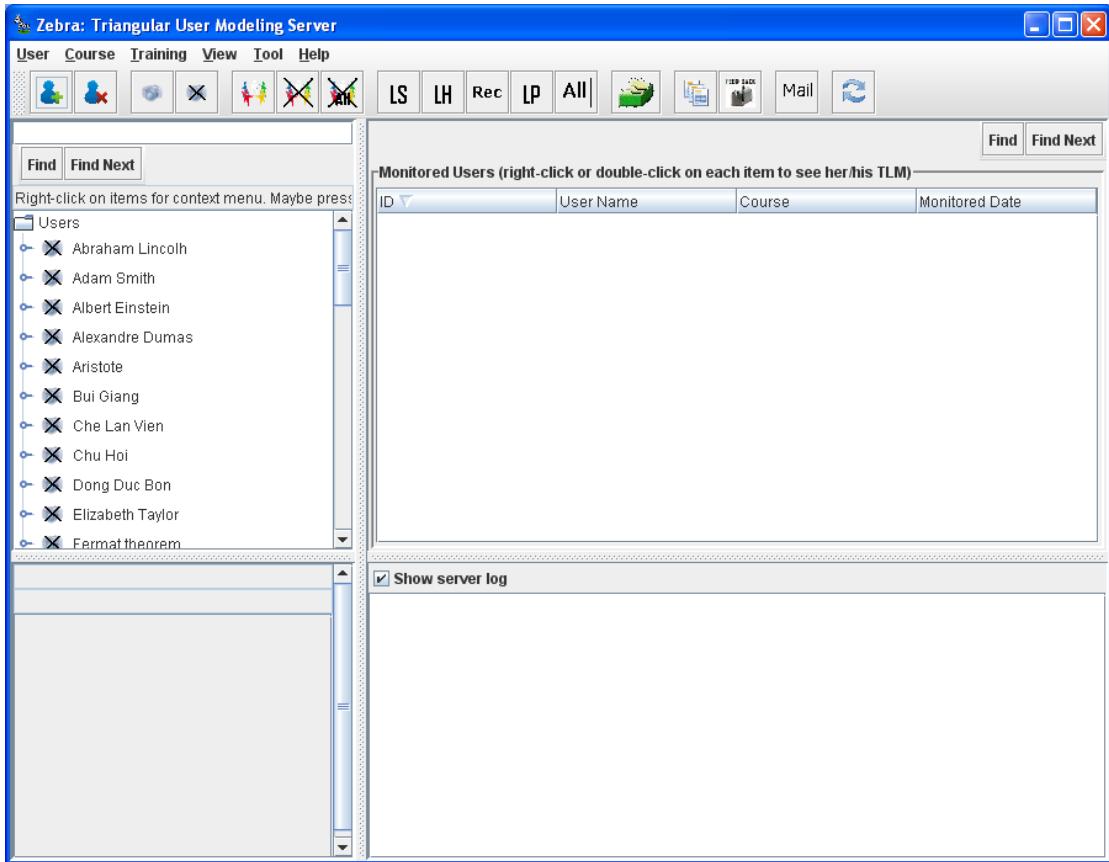
```

Tomcat
Thu Dec 25 20:09:34 ICT 2014: <user=, course=tutorial>: Build Classifier Task in DocClassifier daemon <started>
Thu Dec 25 20:09:36 ICT 2014: <user=, course=javatutorial>: User Community daemon started!
Thu Dec 25 20:09:36 ICT 2014: <user=, course=javatutorial>: Perform Clustering Task in User Community daemon <started>
Thu Dec 25 20:09:36 ICT 2014: <user=, course=tutorial>: User Community daemon started!
Thu Dec 25 20:09:36 ICT 2014: <user=, course=tutorial>: Perform Clustering Task in User Community daemon <started>
Thu Dec 25 20:09:36 ICT 2014: <user=, course=javatutorial>: Mailing List daemon started!
Thu Dec 25 20:09:36 ICT 2014: <user=, course=tutorial>: Mailing List daemon started!
Thu Dec 25 20:09:36 ICT 2014: TriUMServer created!
WOW context initialized
Dec 25, 2014 8:09:37 PM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
Dec 25, 2014 8:09:37 PM org.apache.jk.common.ChannelSocket init
INFO: JK: ajp13 listening on /0.0.0.0:8009
Dec 25, 2014 8:09:37 PM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/47 config=null
Dec 25, 2014 8:09:37 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 15300 ms

```

**Figure II.13.** Zebra is running

Zebra server is often manipulated via Zebra control panel shown in figure II.14.



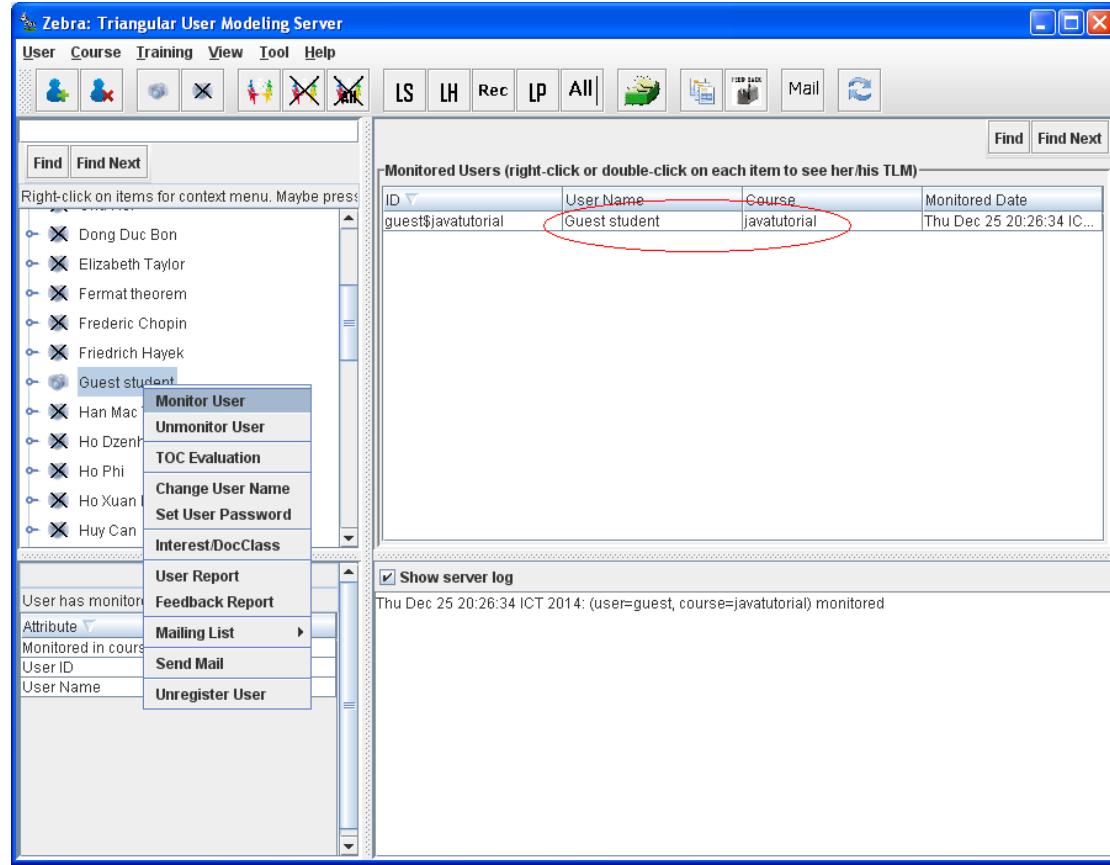
**Figure II.14.** Zebra control panel

The main purpose of Zebra server is to manage learners. So, you can choose a learner appearing in the left column of control panel and monitor her/him. For example, learner “*Guest student*” is monitored in the “Java course” as shown in figure II.15. Java course is designed and written as “Java tutorial” (Oracle, The Java™ Tutorials, 2009) which is web-based learning course teaching Java programming language to learners. It contains knowledge items, concepts, HTML lectures and online tests. Please read (W3Schools, 1999) and (W3C, XHTML™ 1.1 - Module-based XHTML - Second Edition, 2010) for more details about HTML and XHTML. The Java tutorial is available at <http://docs.oracle.com/javase/tutorial> (Oracle, The Java™ Tutorials, 2009). There are many courses like “Java course” built in Zebra server. Teachers and experts can create their own courses. Each learner is monitored according to the course she/he studies because her/his knowledge model is constructed by combination of overlay model and Bayesian network; please see sub-model III.1.1 for more details about knowledge sub-model. It is conventional that default user monitored by Zebra server is Guest student and users are learners in learning context. Guest student has identification “*guest*” and her/his TLM inside Java course is named “*guest\$javatutorial*” when the identification of Java course (Java tutorial) is “*javatutorial*”. If there are many courses, each learner owns many TLM (s) too. For example, if Zebra server manages two courses such as Java course identified by “*javatutorial*” and Oracle course identified by “*oracletutorial*”, learner John indentified by “*john*” will have two TLM (s) such

## II.2. Zebra: A User Modeling System for Triangular Learner Model

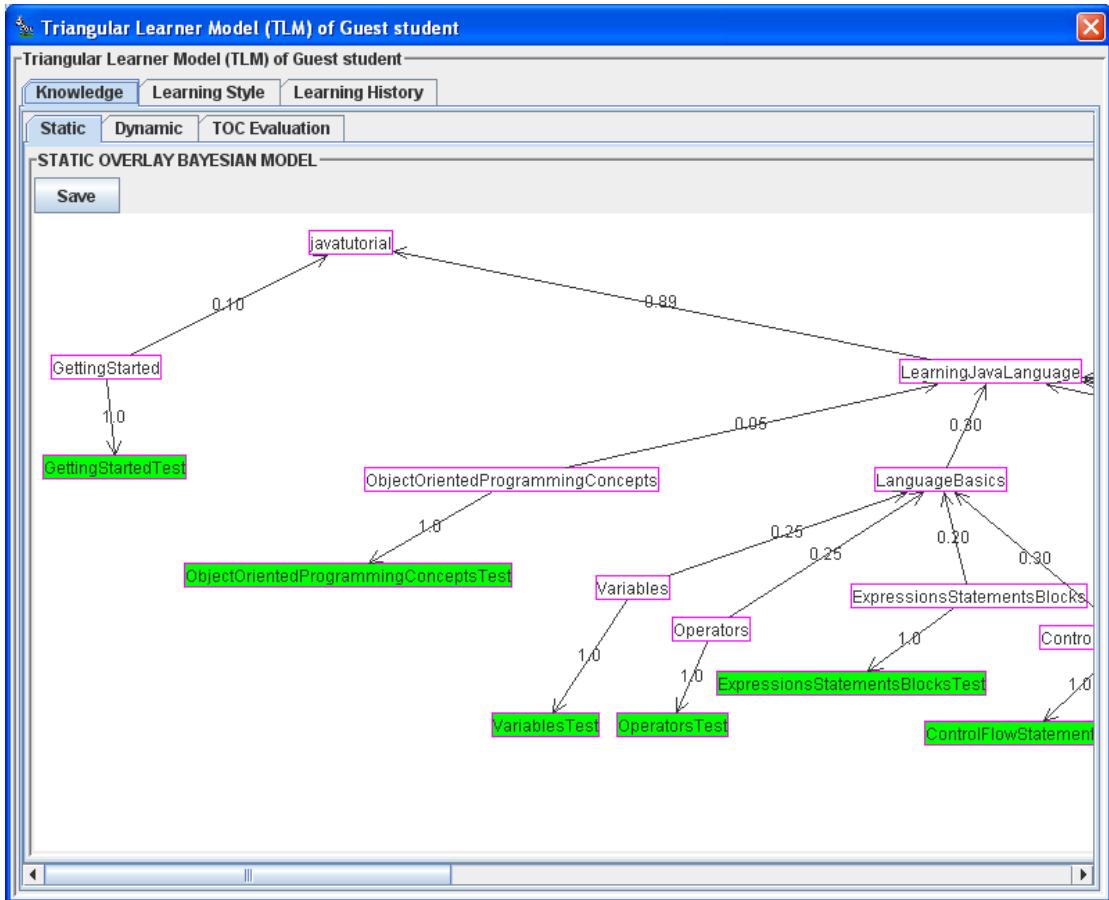
---

as “john\$javatutorial” and “john\$oracletutorial”. It is concluded that each TLM always associates with a course.



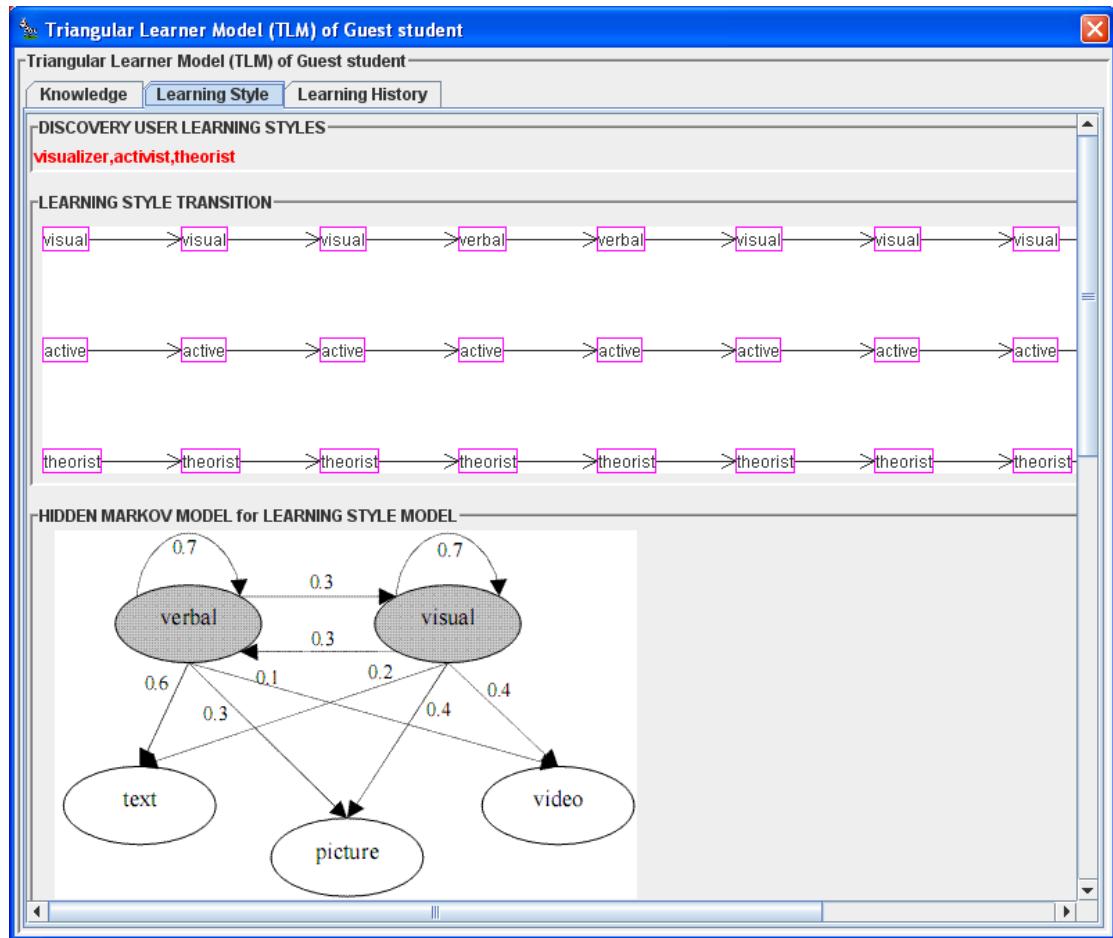
**Figure II.15.** Monitoring a learner

The Guest student is appeared in the center of the control panel when she/he is monitored. If you double-click on the “Guest student”, her/his TLM managed by two engines BNE and ME is opened as shown in figure II.16.



**Figure II.16.** TLM of given learner

There are three tabs “Knowledge”, “Learning style”, and “Learning History” shown in figure II.16, visualizing three sub-models of TLM such as knowledge sub-model, learning style sub-model, and learning history sub-model. Knowledge sub-model is constructed by Bayesian overlay model which is combination of overlay model and Bayesian network. Figure II.16 expresses static Bayesian overlay model in which nodes represent concepts, knowledge items and evidences; please see sub-section III.1.1 for more details about Bayesian overlay model. Learning style sub-model is constructed by hidden Markov model (see section III.2) shown in figure II.17. BNE is responsible for manipulating knowledge sub-model and learning style sub-model.



**Figure II.17.** Learning style sub-model constructed by hidden Markov model

Learning history sub-model stores coarse information in XML files, as shown in figure II.18. ME is responsible for manipulating learning history sub-model. Please see section III.3 for more details about learning history sub-model.

The screenshot shows the 'Triangular Learner Model (TLM) of Guest student' window. The 'Learning History' tab is selected. Below it, there are two sections: 'HISTORY DATA OF guest' and 'ALL USERS' HISTORY DATA'. Both sections display XML data tables.

**HISTORY DATA OF guest:**

Column 1	Column 2
@id	guest
<session id=5443E953B8CB58EF09736D2DAB1A2CEC>	5443E953B8CB58EF09736D2DAB1A2CEC
@id	
<record>	
@id	
<conceptname>	javatutorial.GettingStarted
<accessdate>	Sat May 17 11:58:43 ICT 2014
<resource>	file:/javatutorial/xhtml/GettingStarted.xhtml
<fragment>	false
<record>	
<session id=C7895B17130A86294CAADEE99DB2C537>	
<session id=E1DE40EA231DC5410012F9BE60002EB8>	
<session id=EA709425463144A8F81FECC0DD26A096>	

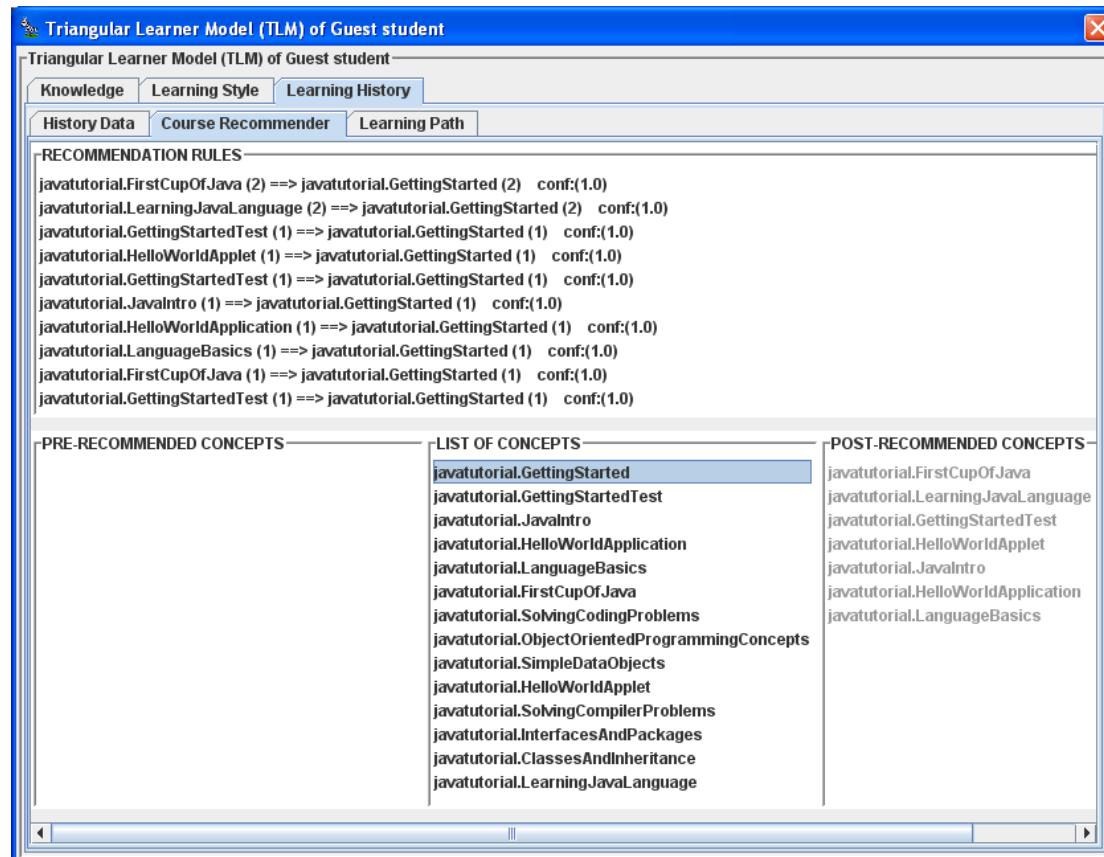
**ALL USERS' HISTORY DATA:**

Column 1	Column 2
@name	javatutorial
<attributes>	
<user id=guest>	
@id	guest
<session id=5443E953B8CB58EF09736D2DAB1A2CEC>	5443E953B8CB58EF09736D2DAB1A2CEC
@id	
<record>	
@id	
<conceptname>	javatutorial.GettingStarted
<accessdate>	Sat May 17 12:28:56 ICT 2014
<resource>	file:/javatutorial/xhtml/GettingStarted.xhtml
<fragment>	false
<record>	
<session id=C7895B17130A86294CAADEE99DB2C537>	
<session id=E1DE40EA231DC5410012F9BE60002EB8>	
<session id=EA709425463144A8F81FECC0DD26A096>	

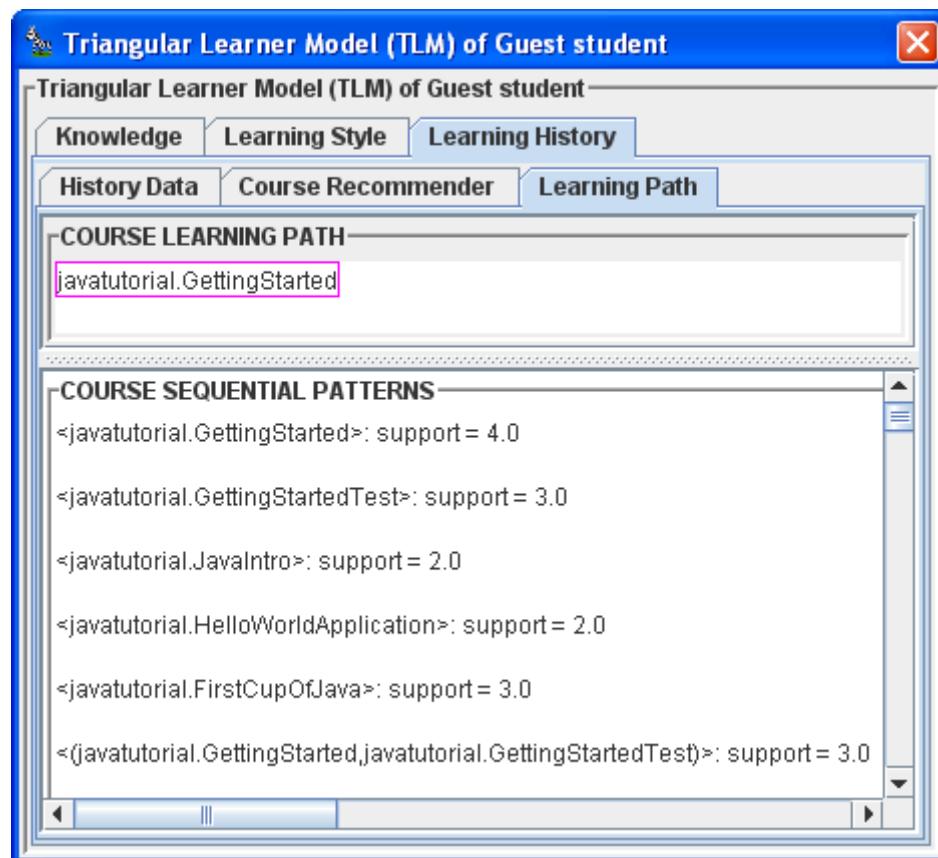
**Figure II.18.** Learning history sub-model stores coarse information in XML files

The first extended function of learning history sub-model is learning concept and learning path recommendation based on sequential pattern mining (see subsection III.3.1), which is illustrated in figures II.19 and II.20.

## II.2. Zebra: A User Modeling System for Triangular Learner Model



**Figure II.19.** Learning concept recommendation



**Figure II.20.** Learning path recommendation

The second extended function of learning history sub-model is discovering user interests based on document classification; please see sub-section [III.3.2](#) for more details about this function. Such function can be performed via Zebra control panel, as shown in figure [II.21](#).

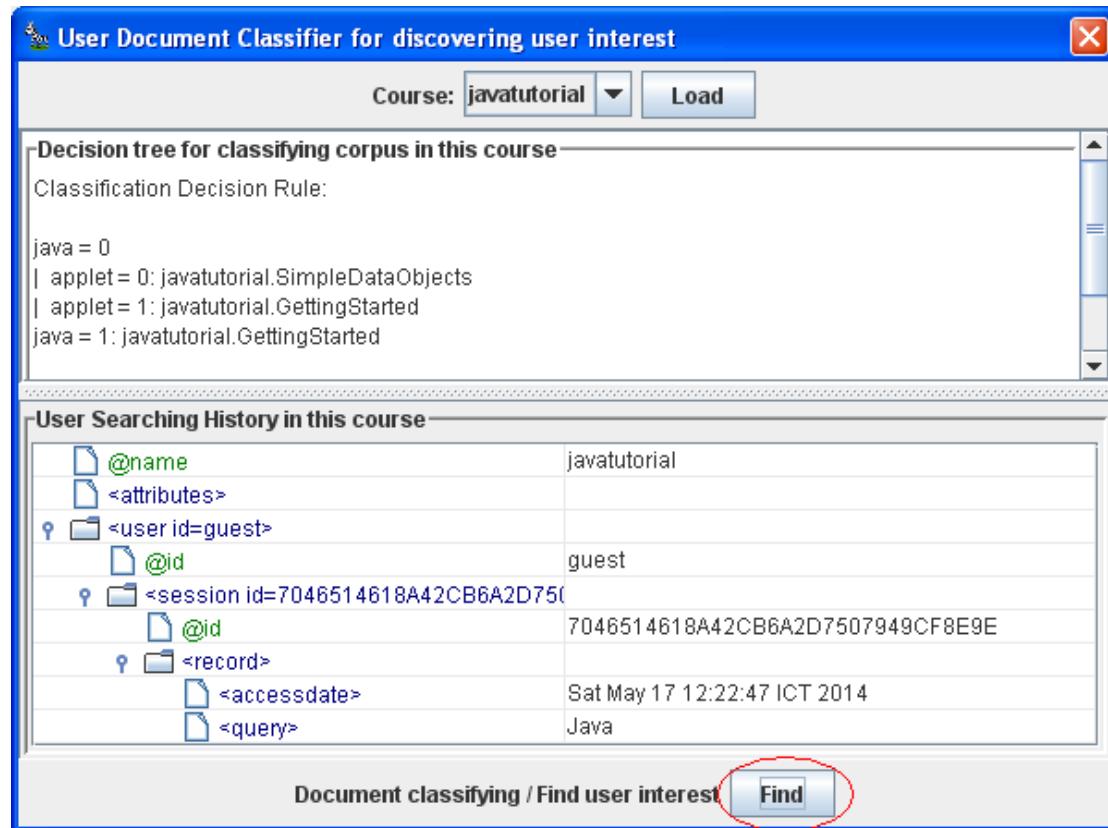
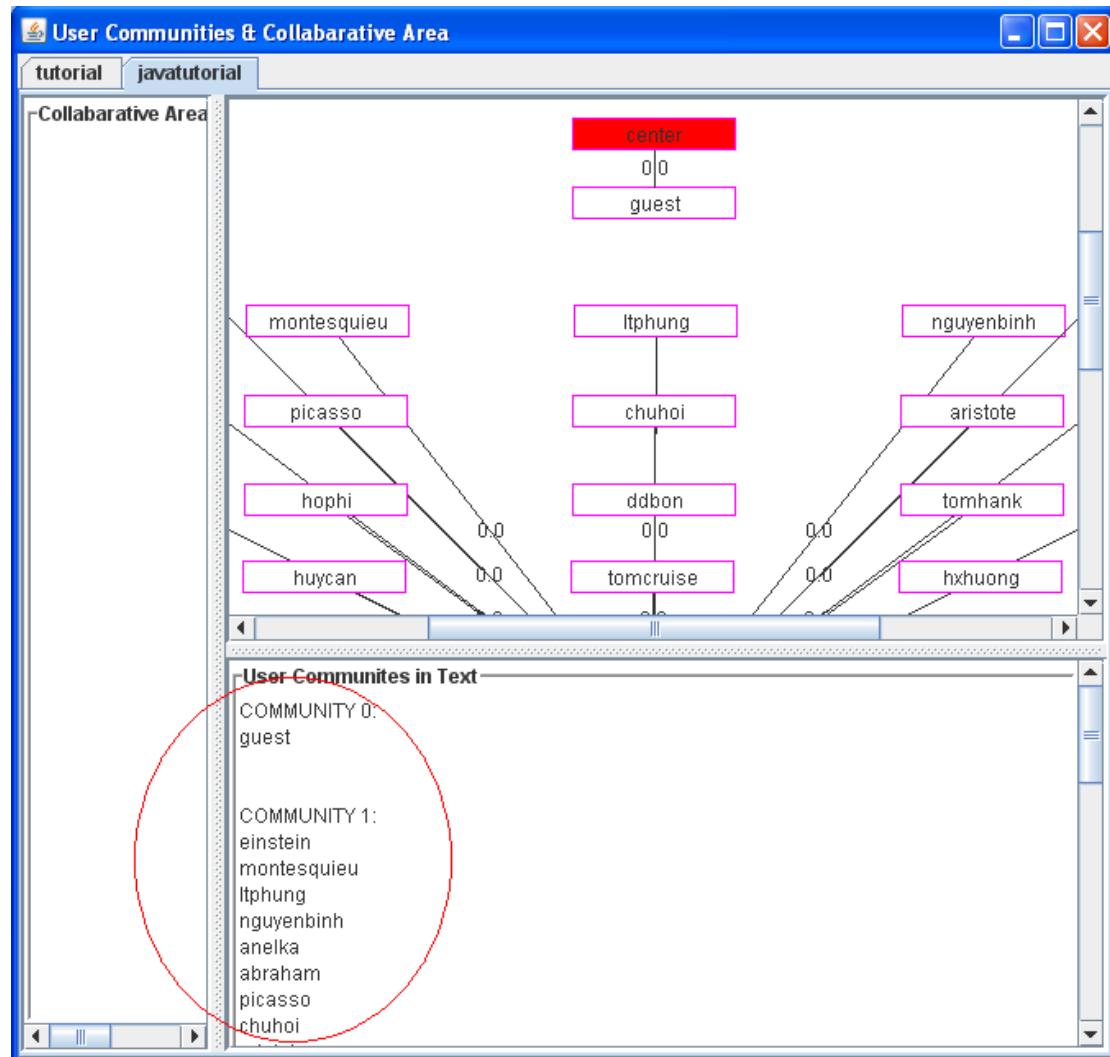
**Figure II.21.** Discovering user interests based on document classification

Figure [II.22](#) shows the third extended function of learning history sub-model which is constructing user groups or user communities.

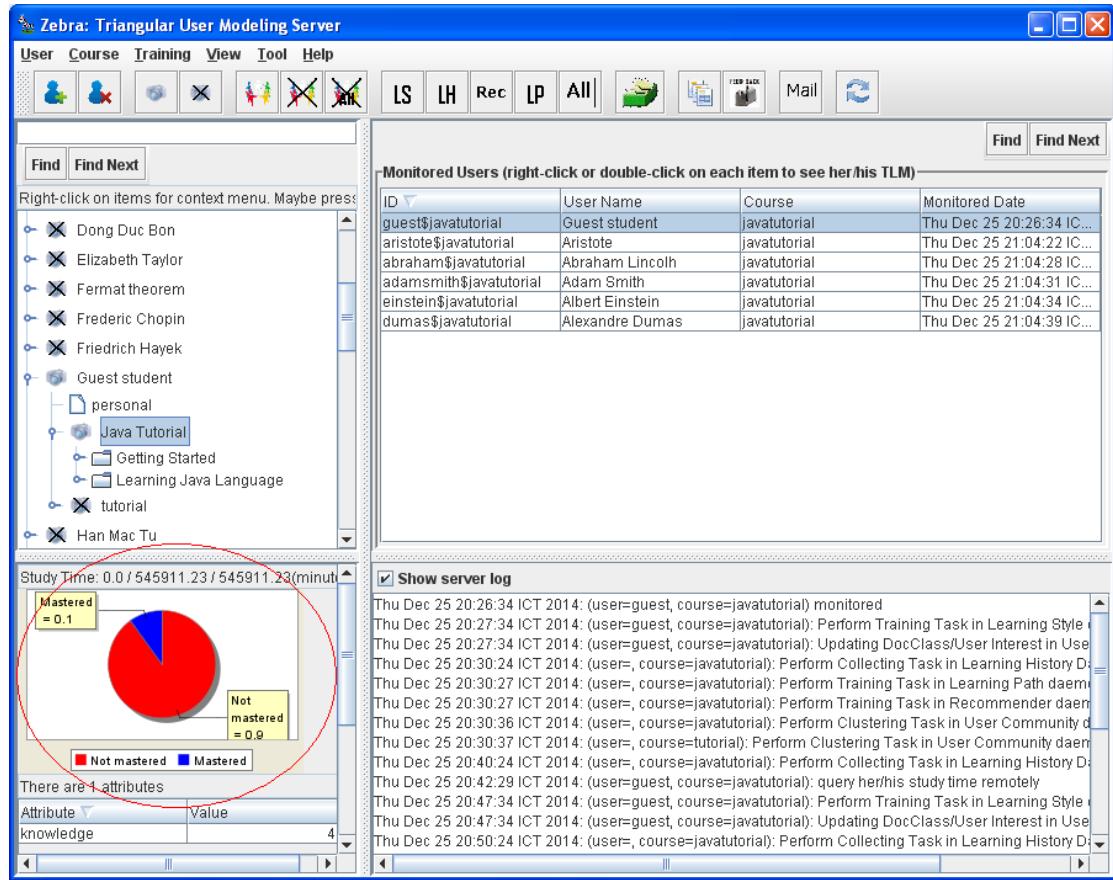


**Figure II.22.** Constructing user groups

As seen figure II.22, there are two communities 0 and 1. Community 0 has only user “Guest student”.

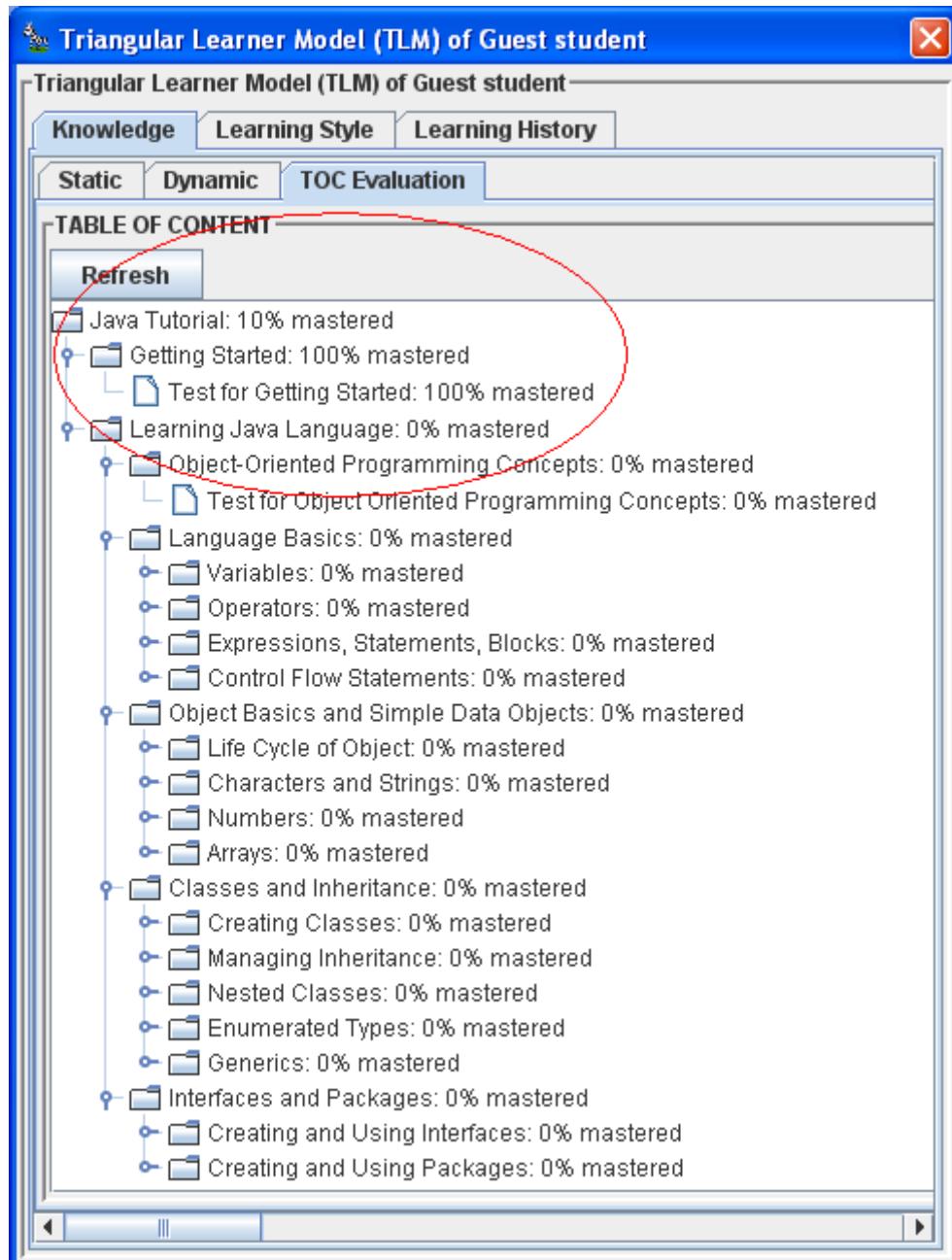
Recall that there are three tabs “Knowledge”, “Learning style”, and “Learning History” shown in figure II.16, visualizing three sub-models of TLM such as knowledge sub-model, learning style sub-model, and learning history sub-model. You can explore these tabs in order to comprehend TLM.

According to knowledge sub-model, it is easy to evaluate how much knowledge the Guest student gains. Figure II.23 expresses evaluation of knowledge of Guest student. According to pie chart in figure II.23, blue area represents the percentage of mastery over given concept or course. It is easy to recognize that the Guest student masters the Java course about 10% (the indicator *mastered*=0.1).

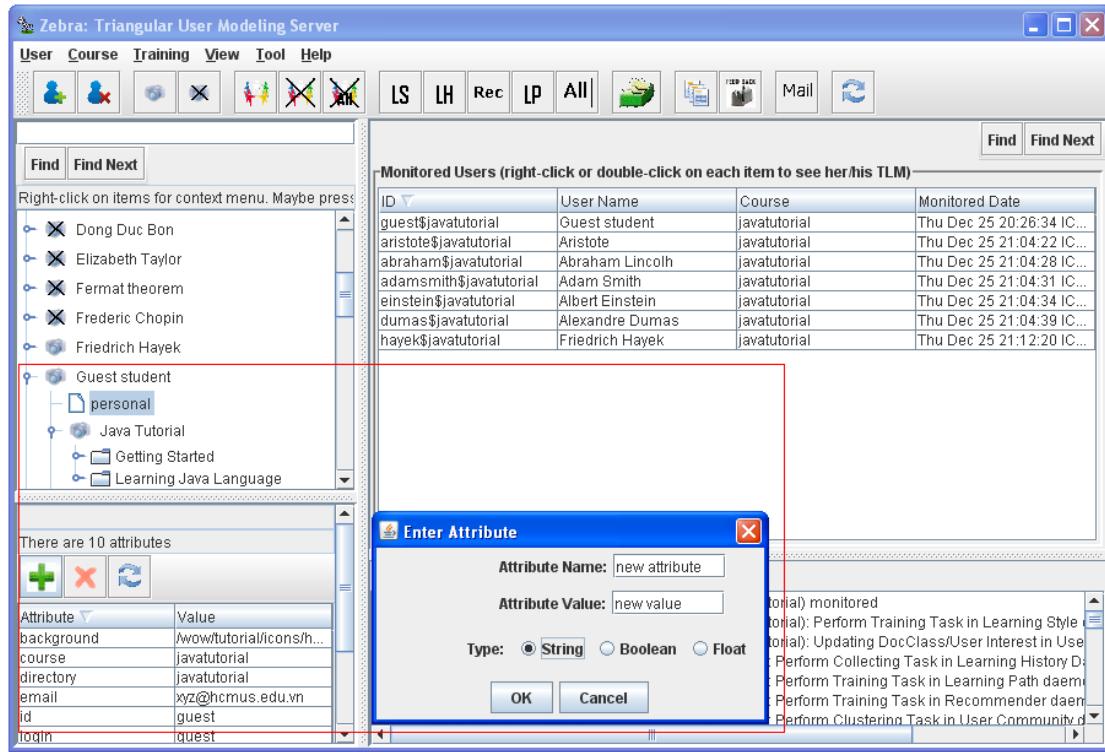


**Figure II.23.** Evaluation of learner's knowledge

Learner's knowledge can be evaluated in detailed as shown in figure [II.24](#).



**Figure II.24.** Evaluation of learner's knowledge in detailed Personal information can be added, deleted, and updated via the control panel, as shown in figure II.25.



**Figure II.25.** Updating personal information

Learner's studying process and studying results are structured in so-called *learning report* shown in figure II.26.

The screenshot shows a window titled "User Total Report" with a blue header bar. The main title is "Report on 'Guest student' in 'javatutorial' course". Below it, a "TOC" section lists 11 items as numbered links:

1. [Person Information](#)
2. [Learning Style](#)
3. [Interest](#)
4. [Learning Path](#)
5. [Your Community](#)
6. [Course Evaluation - Study Result](#)
7. [Study Time](#)
8. [Test Log](#)
9. [Access Log](#)
10. [Search Log](#)
11. [User Feedback](#)

---

**Person Information**

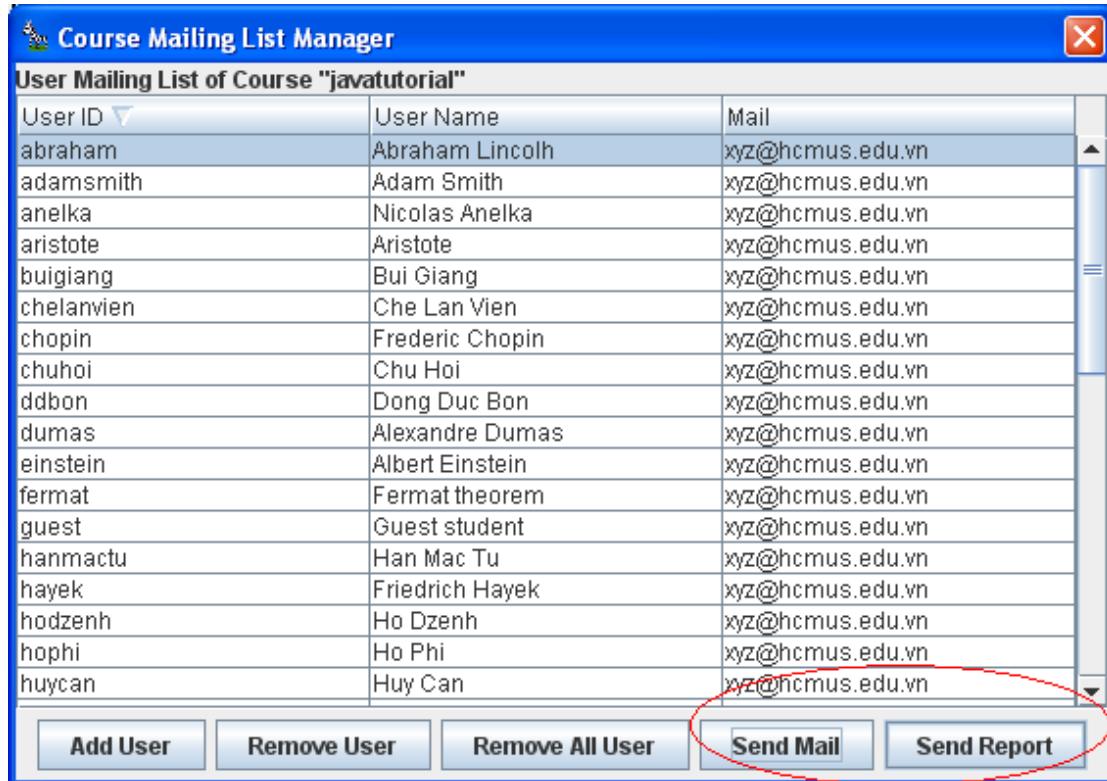
id	guest
login	guest
name	Guest student

[Save](#) [Send Mail](#)

**Figure II.26.** Learning report

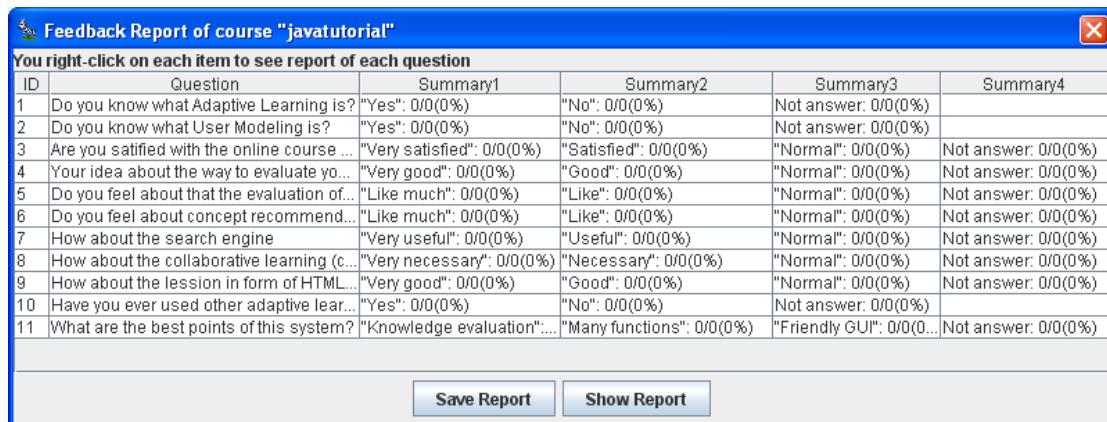
As seen in figure II.26, learning report contains 11 items that describes comprehensively all information about learner.

Zebra server provides a so-called mailing list tool that allows us to send learning report and mails to users. You can also add/remove users to/from mailing list via mailing list tool. Figure II.27 shows the mailing list tool.



**Figure II.27.** Mailing list tool

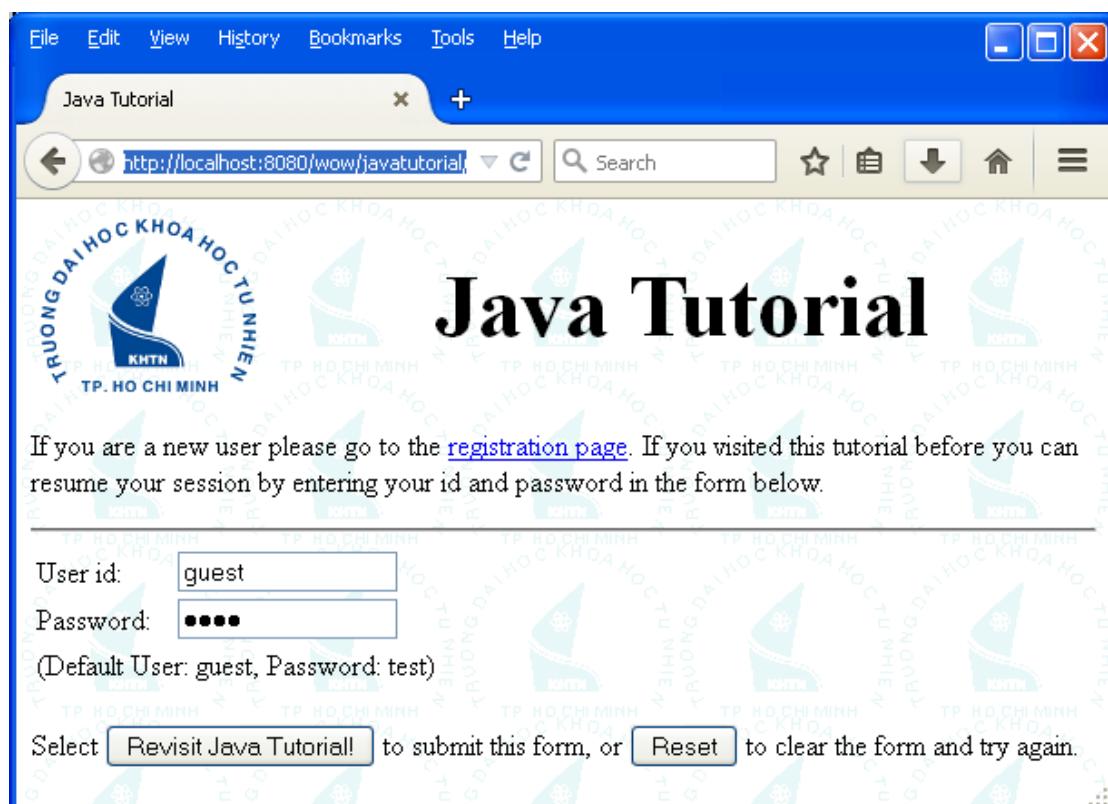
Zebra server also provides a useful utility tool that collects users' feedbacks so as to evaluate itself and adaptive learning system. Concretely, Zebra server issues a questionnaire consisting of a number of questions. These questions help Zebra server to survey users' opinions and feelings about Zebra. Users answer these questions as online feedbacks. Zebra server collects such feedbacks and analyzes them so as to evaluate itself and adaptive learning system according to pre-defined measures. Please see section IV.2 for more details about evaluation of user modeling system and adaptive learning system. Figure II.28 gives an example of feedback report after Zebra server collects such feedbacks.



**Figure II.28.** Feedback report

Now main aspects of the user modeling system Zebra are described. It is necessary to describe the adaptive learning system which interacts with Zebra according to figures II.11 and II.12. In other words, the adaptive learning system takes advantages of users' information provided by Zebra. Please see section I.2 for more details about adaptive learning systems. The adaptive learning system interacting with Zebra server is WOW which is the modified version of system AHA!; please see sub-section I.2.3.3 for more details about AHA!. The system AHA! is developed by the author (De Bra & Calvi, 1998) and so, the WOW system shows gratitude to the author Paul De Bra. Note that both AHA! and WOW are adaptive educational hypermedia system (AEHS); please see sub-section I.2.3 for more details about AEHS. Recall that WOW interacts with Zebra via communication interfaces (CI). CI supports many protocols such as HTTP (Wikipedia, Hypertext Transfer Protocol, 2014) and SOAP (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000).

The main aspect of WOW is to support evaluation of user knowledge based on Bayesian overlay model. According to sub-section III.1.2, that dummy attribute is added into each knowledge item allows WOW to execute adaptation rule following inference mechanism inside Bayesian overlay model built in Zebra. The corporation between WOW and Bayesian inference is strong point of WOW. After running Zebra server as shown in figure II.13, you execute WOW by opening a browser pointing to WOW location, for example, <http://localhost:8080/wow/javatutorial> because WOW is web application written by Java Servlet (Wikipedia, Java Servlet, 2014). Figure II.29 shows WOW login web page (Wikipedia, Web page, 2014) in which Guest student studies.



**Figure II.29.** WOW login web page

Figure II.30 is a typical adaptive learning web page that WOW gives to learners. Left column of the web page shows three important parts supporting adaptive navigation (see sub-section I.2.3) as follows:

- *Links to online HTML lectures (web pages) associated concepts* that learners should master. Please read (W3Schools, 1999) and (W3C, XHTML™ 1.1 - Module-based XHTML - Second Edition, 2010) for more details about HTML and XHTML.
- *Knowledge evaluation for current concept* according to Bayesian inference. The online HTML lectures are adaptive to the knowledge evaluation, for example, if learner does not enough knowledge to study advanced concepts, such advanced concepts are shown as gray or disabled items. As seen in figure II.30, because Guest student mastered absolutely the concept “Getting Started” (mastered = 1), it is sufficient for her/him to study next concept “Learning Java language” and so, the concept “Learning Java language” is enabled.
- *Learning path and learning concept recommendation.* When Guest student mastered absolutely the concept “Getting Started”, she/he is recommended to study concepts: “First Cup of Java”, “Learning Java Language”, etc.

Right column of the web page shows two parts:

- The upper part lists utility tools such as “TOC”, “Glossary”, “Search”, “Feedback”, “User Query”, and “Collaborative” which help learners to take full advantage of WOW.
- The lower part is the current online HTML lecture (web page) associated concepts that learners study. This part supports adaptive presentation. Please see sub-section I.2.3 for more details about adaptive presentation and adaptive navigation.

**Java Tutorial**

Welcome Guest student (xyz@hcmus.edu.vn) - [TOC](#) - [Glossary](#) - [Search](#) - [Feedback](#) - [User Query](#) - [Collaborative](#) - [Change Password](#) - [Log off](#)

## Trail: Getting Started

The lessons in this trail give a quick introduction to Java programming. They tell you what Java is and provide you with an opportunity to compile and run some simple Java programs. Finally, they give you the background knowledge you need to understand how the programs work.

**Having Trouble Getting Started?** Try the detailed instructions in *Your First Cup of Java* for Microsoft Windows, for UNIX, or for Mac OS.

**Learning Path**

Getting Started

**Recommended Concepts**

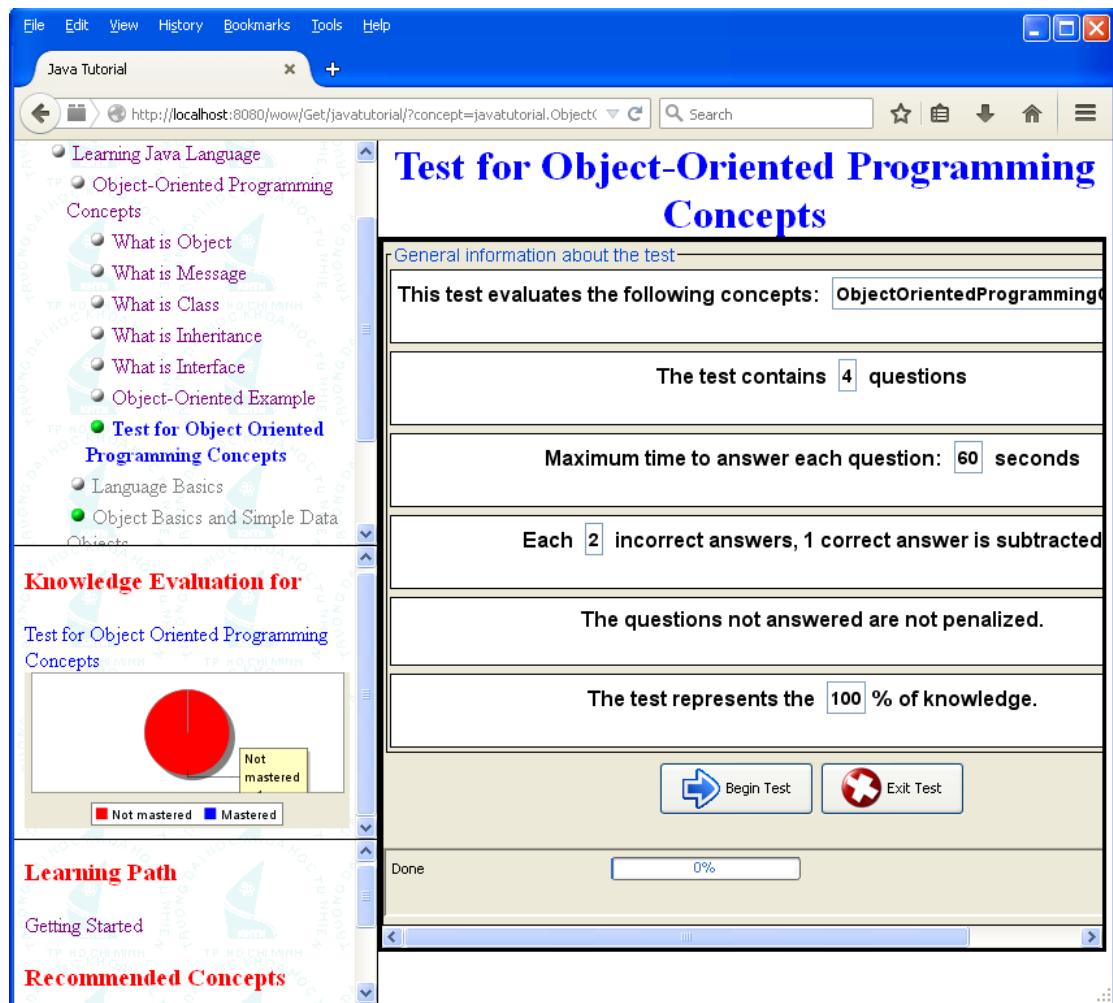
Should be read before:

Should be read after:

- First Cup of Java
- Learning Java Language

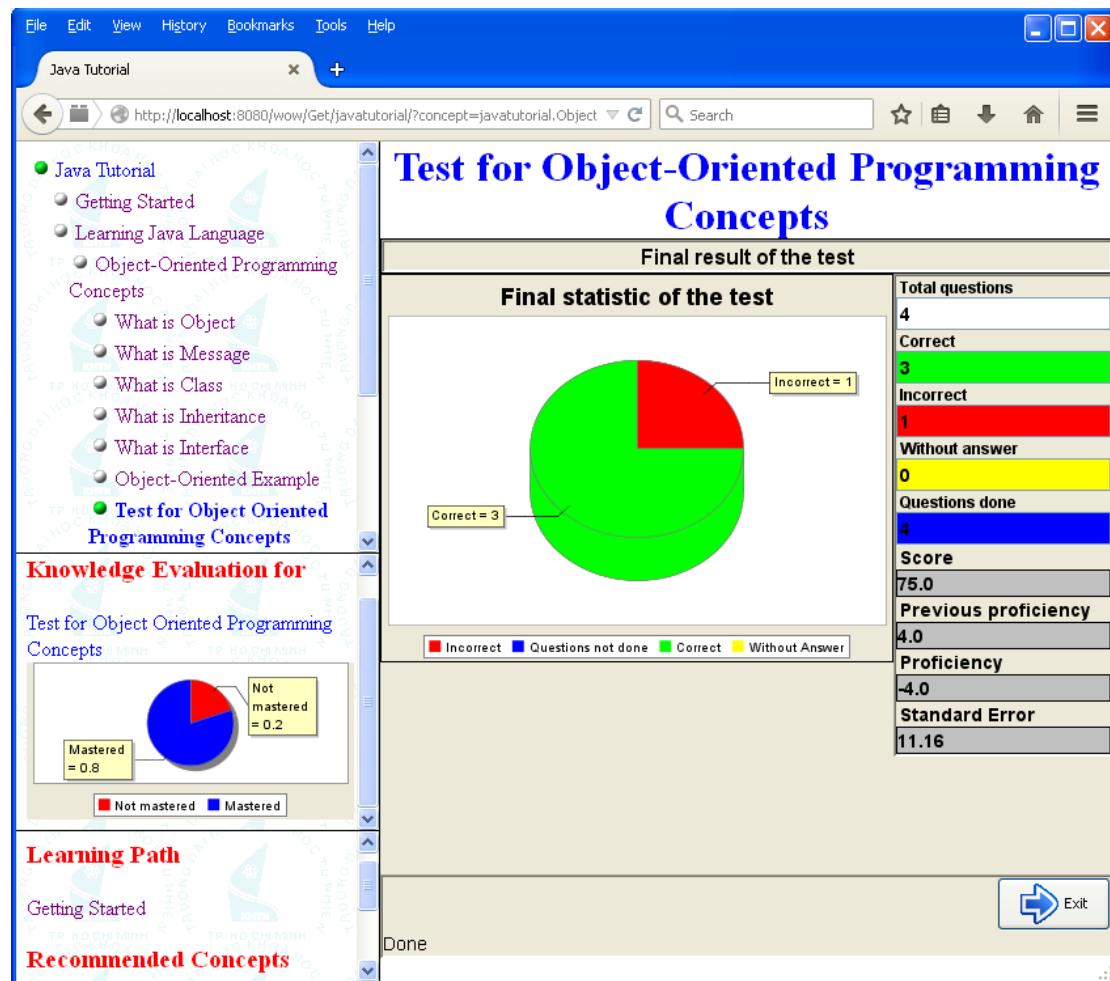
**Figure II.30.** Typical adaptive learning web page supported by WOW

As seen in figure II.30, the knowledge evaluation value represented by indicator “mastered” is 1 with regard to concept “Getting Started”, which means that Guest student comprehends concept “Getting Started”. Essentially, indicator “mastered” is posterior probability of concept “Getting Started” inside the Bayesian overlay model. The relationship between knowledge evaluation and inference mechanism inside Bayesian overlay model implies the corporation between WOW and Zebra server. Suppose Guest student does the online test “Test for Object Oriented Programming Concepts” after she/he studies concept “Object Oriented Programming Concepts” under the concept “Learning Java Language”. Figure II.31 shows that Guest student does such online test with note that the indicator “mastered” is 0. It means that Guest student does not master the concept “Object Oriented Programming Concepts” yet. Note that the online test module inside AHA! is developed by Universidad de Córdoba (<http://www.uco.es>), Spain, 2008.



**Figure II.31.** Leaner does online test via WOW

After doing the online test, Guest student's knowledge is enhanced and the indicator “mastered” for the test “Test for Object Oriented Programming Concepts” is 0.8, as shown in figure II.32.



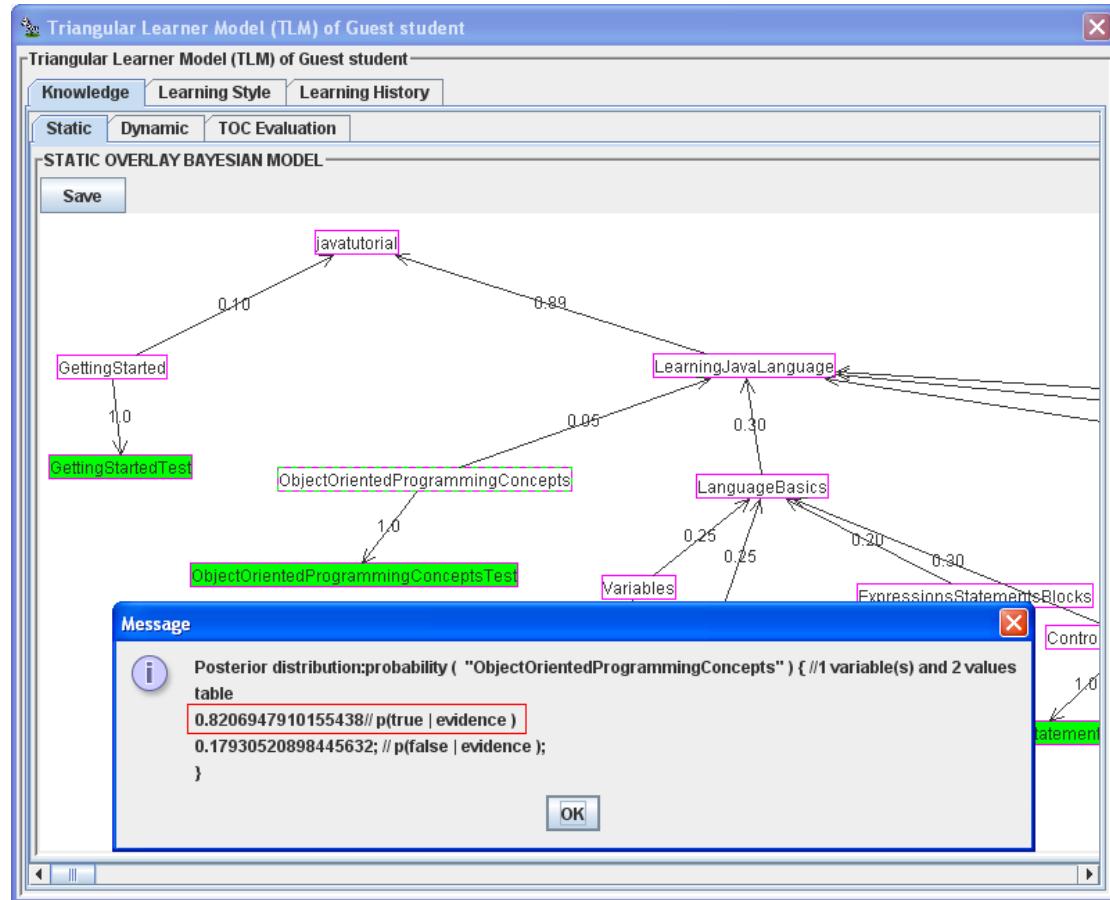
**Figure II.32.** Result of online test

The indicator “mastered” for the concept “Object Oriented Programming Concepts” becomes 0.821. It means that Guest student masters the concept “Object Oriented Programming Concepts” about 82.1% after she/he finishes successfully the test “Test for Object Oriented Programming Concepts”. Therefore, figure II.33 shows such her/his improvement of knowledge.

The screenshot shows a web browser window titled "Java Tutorial". The URL is <http://localhost:8080/wow/Get/javatutorial?concept=javatutorial.Object>. The page content includes a sidebar with navigation links like "Java Tutorial", "Getting Started", "Learning Java Language", "Object-Oriented Programming Concepts", and "Test for Object Oriented Programming Concepts". Below this is a section titled "Knowledge Evaluation for Object-Oriented Programming Concepts" featuring a pie chart. The chart indicates that 82.1% of the concept is mastered (blue) and 17.9% is not mastered (red). The master status is also listed as "Mastered = 0.821" and "Not mastered = 0.179". The main content area is titled "Java Tutorial" and "Lesson: Object-Oriented Programming Concepts". It contains text about object-oriented programming concepts and three questions with diamond icons: "What Is an Object?", "What Is a Message?", and "What Is a Class?".

**Figure II.33.** Knowledge evaluation is increased after learner finishes test successfully

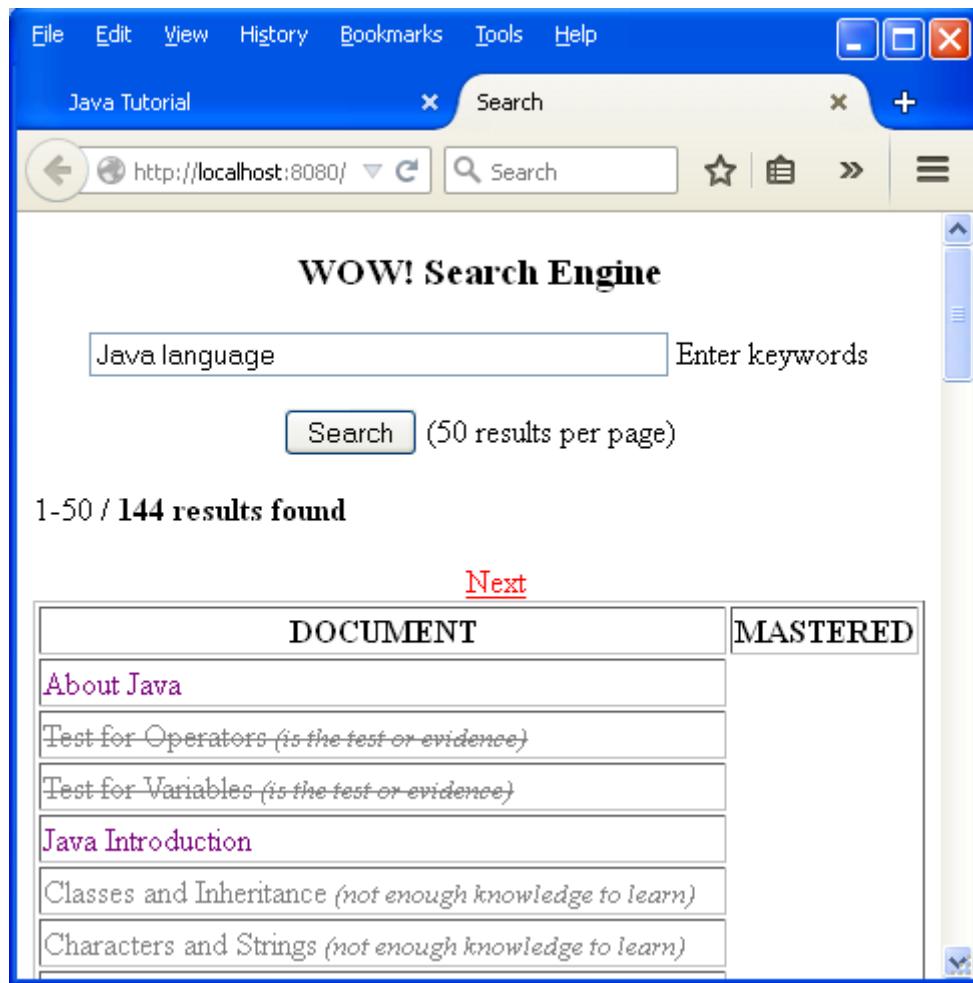
There is a question “How to determine that Guest student achieves 82.1% knowledge of concept “Object Oriented Programming Concepts”; in other words, how to determined indicator “mastered” = 1”. The answer of this question implicates the corporation between WOW and Zebra server. Shortly, the test “Test for Object Oriented Programming Concepts” is evidence inside Bayesian overlay model. Based on the test result (see figure II.32), inference mechanism inside Bayesian overlay model is executed and the posterior probability represented by indicator “mastered” gets 0.821. Figure II.34 expresses the posterior probability of concept “Object Oriented Programming Concepts” and it is  $0.820694791 \approx 0.821$ .



**Figure II.34.** Posterior probability of given concept

Note that Bayesian overlay model is built in Zebra server and so, it is easy to recognize that figure II.34 shows a partial function of Zebra control panel. Therefore, it is possible to conclude that there is a corporation between Zebra server and WOW. Note that figures II.16 and II.34 show the same function of Zebra server which manipulates the TLM.

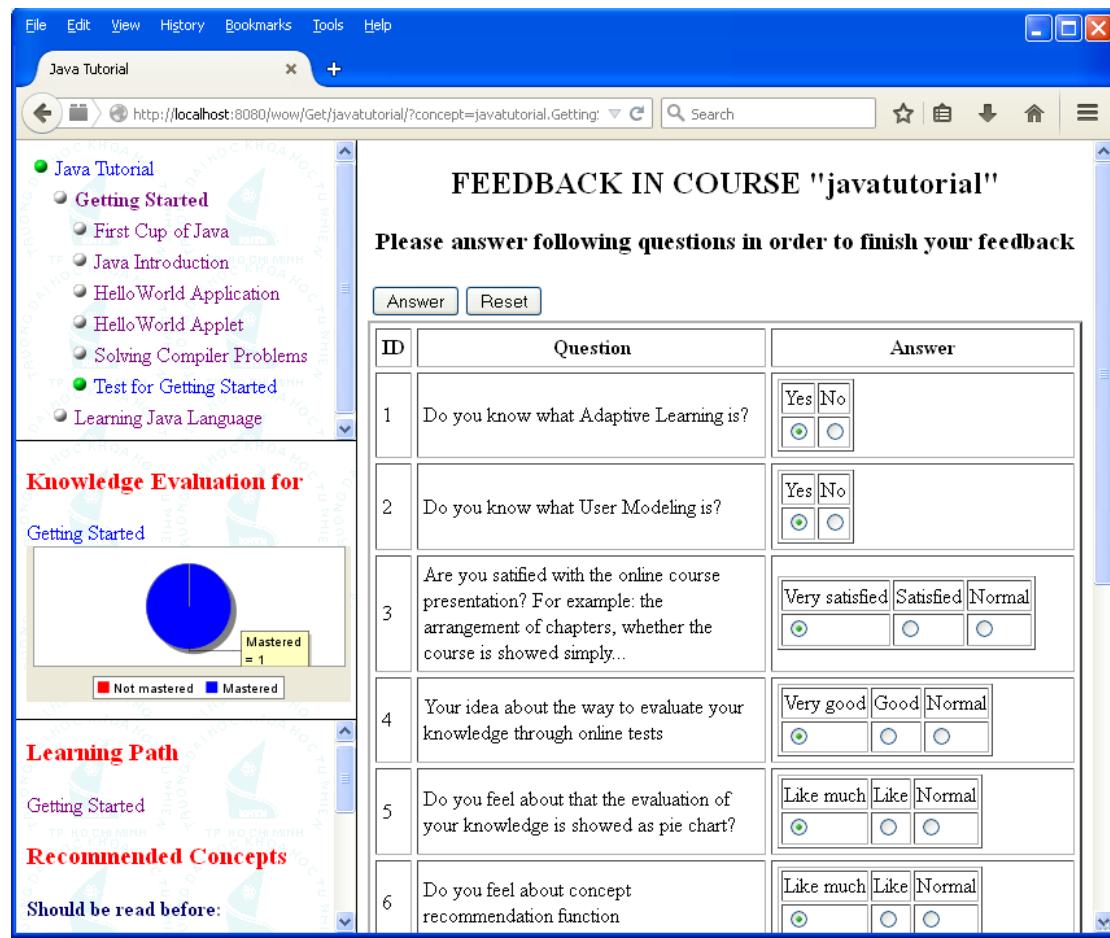
Additionally, WOW provides a search engine that allows learners to search information relevant to Java course. Figure II.35 shows this search engine.



**Figure II.35.** Search engine in WOW

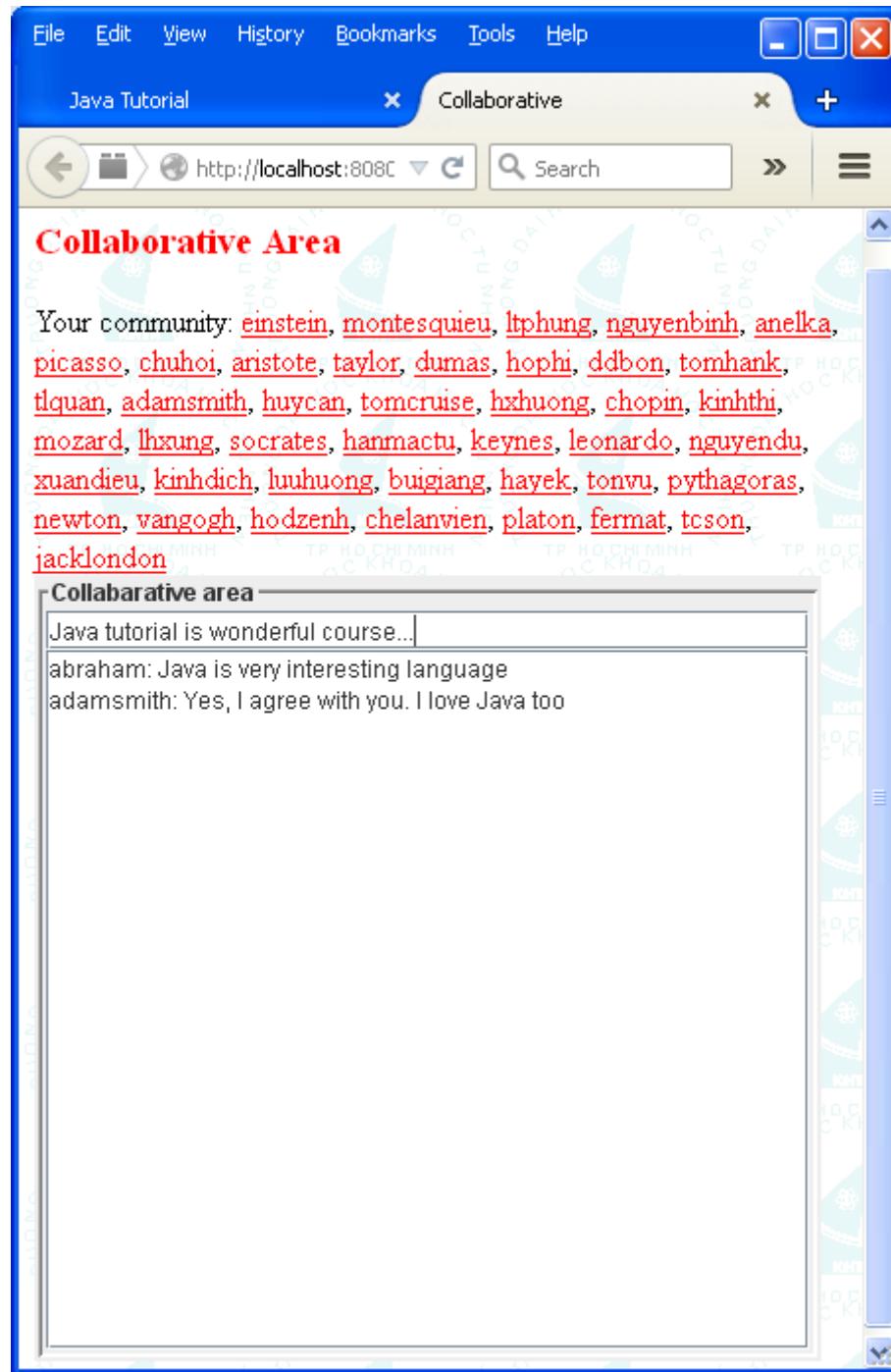
Based on aforementioned knowledge evaluation (see figure II.34), the search engine also supports adaptive recommendation when concepts resulted from a search task will be disabled or stroke out if learner has not enough knowledge to study these concepts. As seen in figure II.35, concepts that learner has not enough knowledge to study are “Classes and Inheritance” and “Characters and Strings”.

Recall that Zebra server collects users’ feedbacks in order to evaluate itself and WOW (see figure II.28). Of course, users must send feedbacks to Zebra. WOW provides the utility tool that help learners to send their feedbacks to Zebra server. Figure II.36 shows the WOW feedback tool.



**Figure II.36.** WOW feedback tool

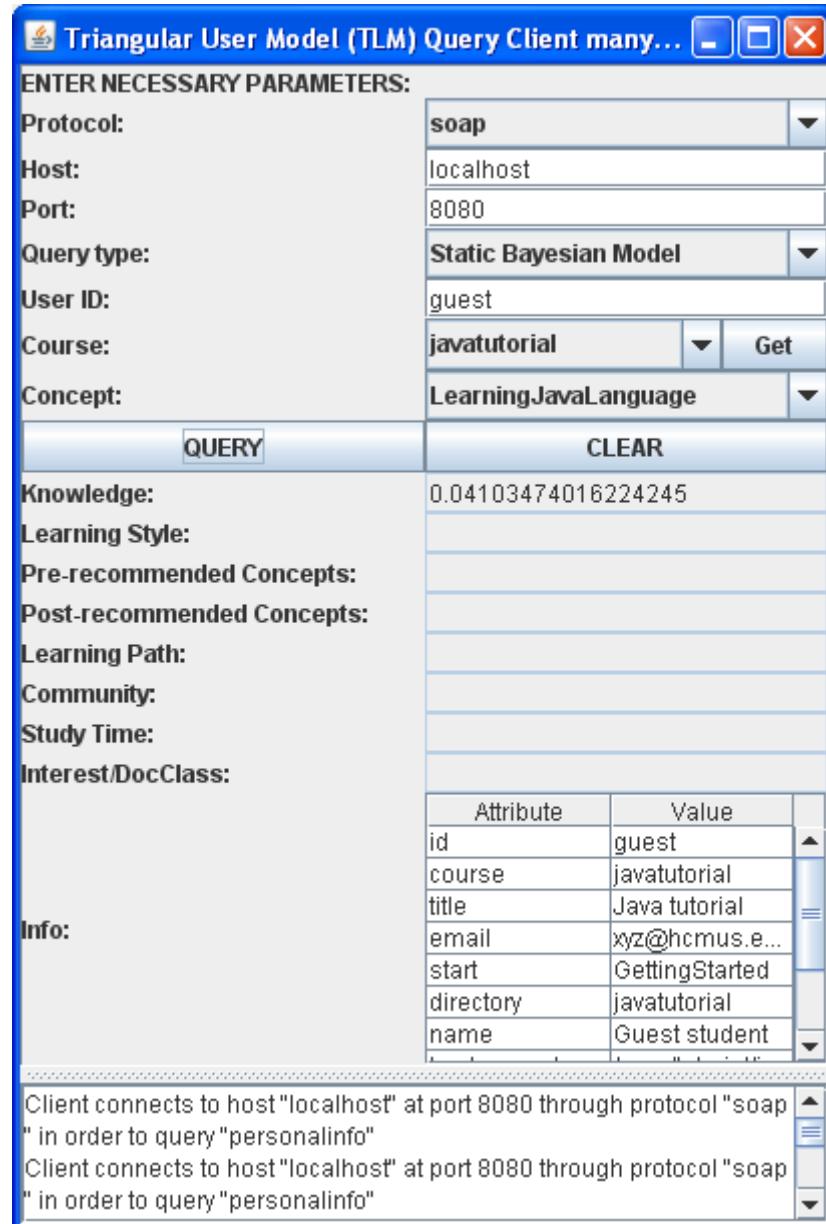
Recall that the third extended function of learning history sub-model is constructing user groups or user communities (see figure II.22). The WOW system helps learners to interact with other learners in the same community via *collaborative area* shown in figure II.37.



**Figure II.37.** Collaborative area for collaborative learning

Now functions of adaptive learning system WOW were described and WOW can be considered as the thick client of Zebra server. WOW interacts with Zebra server and takes advantages of users' information received from Zebra server so as to change its action to provide both learning contents and pedagogic environment/methods appropriate to each learner. There is another client so-called thin client that also queries users' information from Zebra server. Note that this thin client interact with Zebra server via communication interfaces (CI). CI supports many protocols such as HTTP (Wikipedia, Hypertext Transfer Protocol, 2014) and SOAP (W3C, Simple Object Access

Protocol (SOAP) 1.1, 2000). Figure II.38 shows the thin client connecting to Zebra server.



**Figure II.38.** Thin client connecting Zebra server

This sub-section II.2.4 ends up the main section II.2 that proposes Zebra – the user modeling system for [Triangular Learner Model \(TLM\)](#). The successive section II.3 is the conclusion of chapter II.

### II.3. Conclusion

In this chapter II, the “[Triangular Learner Model \(TLM\)](#)” composed by three underlying characteristics: knowledge, learning style and learning history aims to cover the whole of learner’s information required by learning adaptation process. Hence TLM has three sub-models: knowledge sub-model, learning style sub-model and learning history sub-model which are considered as three apexes of a triangle. Next chapter III is the comprehensive description of these sub-models, which is the most detailed chapter in this research.

In general, UMS which builds up and manipulates TLM so-called [Zebra](#) includes the core engine and a set of communication interfaces. The core engine is the composition of two sub-engines: mining engine and belief network engine. Mining engine applies data mining algorithms into discovering and structuring TLM. Belief network engine is responsible for inferring new personal traits from TLM by using deduction mechanism available in belief network. Communication interfaces (CI) allow users to see or modify restrictedly their TLM and adaptive applications also interact with Zebra through them. I also propose the new architecture of adaptive education hypermedia system (AEHS) that interacts with Zebra. Thus, please understand thoroughly these basic concepts and architecture of Zebra before going to chapter III which will describe particularly TLM and Zebra with note that chapter III is the longest one containing a lot of knowledge.

## Chapter III. Three sub-models in Triangular Learner Model

The previous chapter II gives us a general architecture of [Triangular Learner Model](#) (TLM) and its user modeling system [Zebra](#). TLM is composed of three sub-models such as knowledge sub-model, learning style sub-model and learning history sub-model. This chapter is the most detailed one that describes thoroughly these sub-models together with evaluations on each of them. Thus, it includes three main sections [III.1](#), [III.2](#), and [III.3](#) which focus on knowledge sub-model, learning style sub-model, and learning history sub-model, respectively. Please pay attention to the coherence of such three main sections together with respective sub-models. Now the knowledge sub-model is described in the first section [III.1](#) as below.

### III.1. Knowledge sub-model

The core of adaptive system is user model containing personal information such as knowledge, learning styles, and goals which are requisite for learning personalized process; please review previous sections [I.1](#) and [I.2](#) for more details about user model and adaptive learning. There are many modeling approaches, for example, stereotype, overlay, and plan recognition but they do not bring out the solid method for reasoning from user model. This research introduces the statistical method that combines Bayesian network and overlay modeling so that it is able to infer user's knowledge from evidences collected during user's learning process. Note that the knowledge sub-model is one among three sub-models constituting the Triangular Learner Model (TLM) mentioned in previous sub-section [II.2.1](#). The strong point of this model is the inference mechanism when it is constructed in form of Bayesian network; so it has ability to predict and assess user knowledge through observing their actions like accessing lectures, doing exercise, and doing test. Assessing user knowledge is not easy in traditional education that teacher and student teach and learn face-to-face. In the complex teaching curriculum, almost teachers can't assess user knowledge unless they use powerful math tools which are implemented in this sub-model. This section [III.1](#) has five main parts:

1. Combination of Bayesian network and overlay model in building up knowledge sub-model (sub-section [III.1.1](#)).
2. Incorporating Bayesian inference into adaptation rules (sub-section [III.1.2](#)).
3. Evolution of Bayesian overlay model (sub-section [III.1.3](#)).
4. Improving knowledge sub-model by using dynamic Bayesian network (sub-section [III.1.4](#)).
5. Specifying prior probabilities (sub-section [III.1.5](#)).

The first sub-section [III.1.1](#) is the most important; it describes in detailed how to construct knowledge sub-model by applying Bayesian network. Sub-section

[III.1.2](#) discusses about how to take advantage of knowledge sub-model so as to perform adaptive learning tasks. Sub-sections [III.1.3](#) and [III.1.4](#) discusses two approaches to improve Bayesian network: parameter learning by beta function together with expectation maximization (EM) algorithm and structure learning by using dynamic Bayesian network to model temporal knowledge. Sub-section [III.1.5](#) is an extra part which mentions how to specify prior probabilities more accurately so as to enhance the inference process in Bayesian network. Note that knowledge sub-model is managed by belief network engine (BNE) inside the user modeling system Zebra described particularly in previous section [II.2](#).

### **III.1.1. Combination of Bayesian network and overlay model in building up knowledge sub-model**

User modeling is the new trend of enhancing the adaptability of e-learning system. User models are classified into: stereotype model, overlay model, differential model, perturbation model, and plan model (see previous sub-section [I.1.2](#)).

- Stereotype (Rich, 1979) is a set of user's frequent characteristics. In general, stereotype represents a category or group of learners.
- In overlay modeling (see previous sub-section [I.1.2.2](#)), learner model is the subset of domain model. The domain is decomposed into a set of elements and the overlay model is simply a set of masteries over those elements.
- Differential model (Mayo, 2001, p. 58) is basically an overlay on expected knowledge, which in turn is an overlay on expert's domain knowledge.
- Perturbation model (Mayo, 2001, p. 59) represents learners as the subset of expert's knowledge plus their mal-knowledge.

Modeling user must follow three below steps:

- Initialization is the process which gathers information and data about user and constructs user model from this information.
- Updating intends to keep user model up-to-date.
- Reasoning new information about user out from available data in user model.

Reasoning is complex but essential and interesting, especially, there is need to deal with uncertain or imprecise information in user modeling. For example, answering the question: "The student failed the exam, so most probably she/he doesn't master the knowledge" is involved in processing uncertain information. Approaches which solve this problem primarily base on theory of artificial intelligence (AI) or statistics. Both AI and statistics have particular advantages and drawbacks but statistical method is appropriate to evaluate learner's performance by collecting evidences. Bayesian network which is the marriage between Bayesian inference and graph theory has a solid mathematical fundamental. Additionally, overlay model can represent very clearly user's knowledge. In this sub-section [III.1.1](#), I propose the combination of Bayesian

network and overlay model so that it is able to take full advantages of strong points of both of them (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009). First, survey of Bayesian inference and Bayesian network is discussed in sub-section III.1.1.1. After that main method of applying Bayesian network into overlay model, the core of my approach, is described in sub-section III.1.1.2. Sub-section III.1.1.3 is the SIGMA-gate theorem which is the base of such proposed method for specifying Bayesian network parameters. You can ignore sub-section III.1.1.1 if knowing Bayesian network but please pay attention to two main sub-sections III.1.1.2 and III.1.1.3. Evaluation of the combination between Bayesian network and overlay model is mentioned in sub-section III.1.1.4. Note that the knowledge sub-model created by combining overlay model and Bayesian network is called *Bayesian overlay model*, *Bayesian network model*, *Bayesian model*, or *Bayesian network*, in brief.

### **III.1.1.1. Bayesian network**

This sub-section III.1.1.1 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 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 III.1.1a as follows:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}$$

*Formula III.1.1a. 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  $\mathcal{D}$  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. It is also hypothesis’ initial value.
- $P(H|D)$ , 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 the cornerstone of Bayesian inference.
- $P(D|H)$  is conditional probability of occurring evidence  $D$  when hypothesis was true. 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  $P(D) = \sum_H P(D|H)P(H)$ , otherwise  $f(D) = \int f(D|H)f(H)dH$  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  $P(\cdot)$  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 expresses in formulas [III.1.1b](#) and [III.1.1c](#), respectively as follows:

$$P(X \cup Y) = P(X) + P(Y) - P(X \cap Y)$$

*Formula III.1.1b. Additional rule*

$$P(X \cap Y) = P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X)$$

*Formula III.1.1c. Multiplication rule*

Where notations  $\cup$  and  $\cap$  denote union operator and intersection operator in set theory (Wikipedia, Set (mathematics), 2014). Your attention please, when  $X$  and  $Y$  are numerical variables, notations  $\cup$  and  $\cap$  also denote operators “*or*” and “*and*” in theory logic (Rosen, 2012, pp. 1-12). If  $X$  and  $Y$  are mutually

independent (mutually exclusive) then,  $X \cup Y$  and  $X \cap Y$  are often denoted as  $X+Y$  and  $XY$ , respectively and so, we have:

$$\begin{aligned} P(X + Y) &= P(X) + P(Y) \\ P(XY) &= P(X, Y) = P(X)P(Y) \end{aligned}$$

Given a complete set of mutually exclusive events  $X_1, X_2, \dots, X_n$  such that

$$\begin{aligned} X_1 \cup X_2 \cup \dots \cup X_n &= X_1 + X_2 + \dots + X_n = \Omega \text{ where } \Omega \text{ is probability space} \\ X_i \cap X_j &= \emptyset, \forall i, j \end{aligned}$$

The total probability rule (Montgomery & Runger, 2003, p. 44) is specified in formula III.1.1d as follows:

$$\begin{aligned} P(Y) &= P(Y|X_1)P(X_1) + P(Y|X_2)P(X_2) + \dots + P(Y|X_n)P(X_n) \\ &= \sum_{i=1}^n P(Y|X_i)P(X_i) \end{aligned}$$

*Formula III.1.1d.* Total probability rule

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

$$P(Y) = \int_X P(Y|X)P(X)dX$$

*Formula III.1.1d'.* 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 III.1.1a) and total probability rule (formulas III.1.1d and III.1.1d') because they are used frequently over the whole research.

### Bayesian network (BN)

Bayesian network (BN) (Neapolitan, 2003, p. 40) (Nguyen, 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 sub-section III.1.3.1 – 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 the derivative of CDF; in other words, CDF is the integral of PDF. In practical statistics, PDF is used more commonly than CDF is used and so, PDF is used over the whole research. Note, notation  $P(\cdot)$  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 III.1.1e is the definition of CDF and PDF.

$$\text{Continuous case: } \begin{cases} F(X_0) = P(X \leq X_0) = \int_{-\infty}^{X_0} f(X) dX \\ \int_{-\infty}^{+\infty} f(X) dX = 1 \end{cases}$$

$$\text{Discrete case: } \begin{cases} F(X_0) = P(X \leq X_0) = \sum_{X \leq X_0} P(X) \\ f(X) = P(X) \text{ and } \sum_X P(X) = 1 \end{cases}$$

*Formula III.1.1e.* Definition of cumulative distribution function (CDF) and probability density function (PDF)

Because this sub-section III.1.1.1 focuses on BN, please read (Montgomery & Runger, 2003, pp. 98-103) for more details about CDF and PDF.

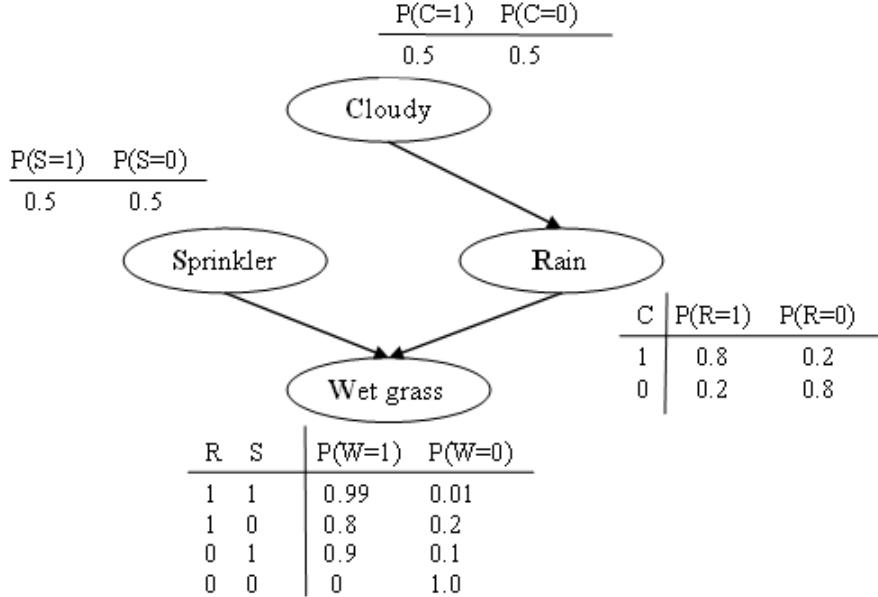
Now please pay attention to the concept CPT because it occurs very frequently in the research; you can understand simply that CPT is essentially collection of discrete conditional probabilities of each node (variable). It is easy to infer that CPT is discrete form of PDF. When one node is conditionally 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 III.1.1, event “cloudy” is cause of event “rain” which in turn is cause of “grass is wet” (Murphy, 1998). So we have three causal relationships of: 1-cloudy to rain, 2- rain to wet grass, 3- sprinkler to wet grass. This model is expressed below by BN with four nodes and three arcs

### III.1. Knowledge sub-model

---

corresponding to four events and three relationships. Every node has two possible values True (1) and False (0) together its CPT.



**Figure III.1.1.** Bayesian network (a classic example about “wet grass”)

Note that random variables  $C$ ,  $S$ ,  $R$ , and  $W$  denote phenomena or event 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, 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 III.1.1. 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 and structure learning will be mentioned in subsections III.1.3, III.1.4, and III.1.5. This sub-section III.1.1 focuses the new approach to construct BN with both structure and parameters.

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 letters  $x_i$  and  $PA(x_i)$  to name a node and a set of its parent, correspondingly. Let  $X$  be vector which was constituted of all  $x_i$ ,  $X = (x_1, x_2, \dots, x_n)$ . The **Global Joint Probability Distribution** (GJPD)  $P(X)$  being product of all local CPD (s) or CPT (s) is formulated as:

$$P(X) = P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | PA(x_i))$$

*Formula III.1.2.* Global joint probability distribution of random vector

Suppose  $\Omega_i$  is the subset of  $PA(x_i)$  such that  $x_i$  must depend conditionally and directly on every variable in  $\Omega_i$ . In other words, there is always an arc from each variable in  $\Omega_i$  to  $x_i$  and no intermediate node between them. Thus, formula III.1.2 becomes:

$$P(X) = P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | \Omega_i)$$

*Formula III.1.2'.* Reduced global joint probability distribution of random vector

Note that  $P(x_i | \Omega_i)$  in formula III.1.2' is the CPT of  $x_i$ . According to Bayesian rule, given evidence (random variables)  $\mathcal{D}$ , the posterior probability  $P(x_i | \mathcal{D})$  of variable  $x_i$  is computed in formula III.1.3 as below:

$$P(x_i | \mathcal{D}) = \frac{P(\mathcal{D} | x_i)P(x_i)}{P(\mathcal{D})}$$

*Formula III.1.3.* Posterior probability of variable  $x_i$  given evidence  $\mathcal{D}$

Where  $P(x_i)$  is prior probability of random variable  $x_i$  and  $P(\mathcal{D} | x_i)$  is conditional probability of occurring  $\mathcal{D}$  when  $x_i$  was true and  $P(\mathcal{D})$  is probability of occurring  $\mathcal{D}$  together all mutually exclusive cases of  $X$ . From formulas III.1.2' and III.1.3, we gain formula III.1.3' as follows:

$$P(x_i | \mathcal{D}) = \frac{\sum_{X \setminus (\{x_i\} \cup \mathcal{D})} P(x_1, x_2, \dots, x_n)}{\sum_{X \setminus \mathcal{D}} P(x_1, x_2, \dots, x_n)}$$

*Formula III.1.3'.* Advanced posterior probability of variable  $x_i$  given evidence  $\mathcal{D}$

Where  $X \setminus (\{x_i\} \cup \mathcal{D})$  and  $X \setminus \mathcal{D}$  are all possible values  $X = (x_1, x_2, \dots, x_n)$  with fixing (excluding)  $\{x_i\} \cup \mathcal{D}$  and fixing (excluding)  $\mathcal{D}$ , respectively. Note that evidence  $\mathcal{D}$  including at least one random variable  $x_i$  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 III.1.3' is the base for inference inside Bayesian network, which is used over the whole research. It is easy to infer that formulas III.1.3 and III.1.3' are extensions of Bayes' rule specified by formula III.1.1a.

From figure III.1.1 of wet grass example and according to formula III.1.2, we have:

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

Applying formula III.1.2',  $P(S|C)=P(S)$  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 III.1.2' we have formula III.1.4 for determining global joint probability distribution of “wet grass” Bayesian network as follows:

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

*Formula III.1.4.* 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 III.1.1, suppose  $W$  becomes evidence variable which is observed the fact that the grass is wet, so,  $W$  has value 1. There is request for answering the question: how to determine which cause (sprinkler or rain) is more possible for wet grass. Hence, we will calculate two posterior probabilities of  $R (=1)$  and  $S (=1)$  in condition  $W (=1)$ . Such probabilities called *explanations* for  $W$  are simple forms of formula III.1.3', expended by formulas III.1.5 and III.1.6 as follows:

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

*Formula III.1.5.* Posterior probability of rain given evidence “wet grass”

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

*Formula III.1.6.* Posterior probability of sprinkler given evidence “wet grass”

Note that the numerator in the right side of formula III.1.5 is the sum of possible probabilities  $P(C, R = 1, S, W = 1)$  over possible values of  $C$  and  $S$ . Concretely, we have an interpretation for the numerator as follows:

$$\begin{aligned} \sum_{C,S} P(C, R = 1, S, W = 1) \\ = & P(C = 1, R = 1, S = 1, W = 1) \\ & + P(C = 1, R = 1, S = 0, W = 1) \\ & + P(C = 0, R = 1, S = 1, W = 1) \\ & + P(C = 0, R = 1, S = 0, W = 1) \end{aligned}$$

Applying formula III.1.4 for global joint probability distribution of “wet grass” Bayesian network, we have:

$$\begin{aligned}
 & \sum_{C,S} P(C, R = 1, S, W = 1) \\
 &= (P(C = 1) * P(S = 1) * P(R = 1|C = 1) \\
 &\quad * P(W = 1|R = 1, S = 1)) \\
 &\quad + (P(C = 1) * P(S = 0) * P(R = 1|C = 1) \\
 &\quad * P(W = 1|R = 1, S = 0)) \\
 &\quad + (P(C = 0) * P(S = 1) * P(R = 1|C = 0) \\
 &\quad * P(W = 1|R = 1, S = 1)) \\
 &\quad + (P(C = 0) * P(S = 0) * P(R = 1|C = 0) \\
 &\quad * P(W = 1|R = 1, S = 0))
 \end{aligned}$$

It is easy to infer that there is the same interpretation for numerators and denominators in right sides of formulas III.1.5 and III.1.6 and the previous formula III.1.3' is also understood simply by this way when  $\{C, S\} = \{C, R, S, W\} \setminus \{R, W\}$  and fixing  $\{R, W\}$ .

In fact, formulas III.1.5 and III.1.6 are expansion of formula III.1.3'. Applying formula III.1.4 to formulas III.1.5 and III.1.6, we have:

$$P(R = 1|W = 1) = \frac{0.4475}{0.7695} = 0.581 < 0.614 = \frac{0.4725}{0.7695} = P(S = 1|W = 1)$$

It is concluded that sprinkler is the most likely cause of wet grass.

### III.1.1.2. Applying Bayesian network to overlay model

The basic idea of overlay modeling is that the user model is the subset of domain model. Straightforward, the domain is decomposed into a set of knowledge elements and the overlay model (namely, user model) is simply a set of masteries over those elements. Suppose that the mastery of each element varies from 0 (not mastered) to 1 (mastered), which is considered as the weight of such element. Then the expert model is the overlay with 1 for each element and the learner model is the overlay with at most 1 for each element (see sub-section I.1.2.2). In general, overlay model is essentially a graph constituted of a set of nodes known as knowledge elements and a set of arcs (see figure I.1.4). Each node is quantified by a so-called mastery number ranging from 0 to 1. Each arc connecting two nodes expresses a relationship specified according to application context and there are many relationships such as prerequisite, dependency, parent-child, cause-effect, aggregation, and diagnostic. The research focuses on aggregation and diagnostic relationship. The aggregation relationship from parent node  $A$  to child node  $B$  expresses that the mastery of node  $A$  contributes partially and exclusively to the whole mastery of node  $B$ . The diagnostic relationship from node  $X$  to node  $D$  expresses that the mastery of node  $D$  is evidence for determining the mastery of node  $X$ . The graph containing one child node and a set of direct parent nodes with constraint that all relationships are aggregations is *sigma graph* which is researched particularly in next sub-section III.1.1.3.

### III.1. Knowledge sub-model

---

Although overlay model is the simple but powerful method to represent user model, it does not provide the way to infer user's knowledge from evidences collected in user's learning process. Overlay modeling should associate with other statistical approach in solving this problem and Bayesian network (BN) is the best choice. So, I combined BN and overlay model by following steps (Nguyen & Do, Combination of Bayesian Network and Overlay Model in User Modeling, 2009):

1. The structure of overlay model is considered as BN. Thus, knowledge elements in domain become variables (or nodes) in BN. Instead of using the weight of each element as above, I assign the probability to each variable for estimating the mastery of knowledge. All variables are binary (0 – not mastered and 1 – mastered). Note, knowledge item, knowledge element and concept are synonym terms.
2. The aggregation relationships between knowledge elements are known as the conditional dependence assertions in BN. Accordingly, each node has a CPT.
3. All knowledge elements are defined as hidden variables (hypothesis). Other learning objects or events (such as tests, exams, exercises, user's feedback, and user's activities) which are used to assess or evaluate user's performance in learning process are consider as evidence variables. We must add them to Bayesian network along with determining the conditional dependence relationship between them and remaining hidden variable, namely, specifying their CPT (s). Inferring user's knowledge is to compute posterior probability of hidden variables according to formula III.1.3' when evidence variables change their values. This process can be known as knowledge diagnosis.

The combination of overlay model and Bayesian network results out *Bayesian overlay model* or Bayesian model or Bayesian network in brief. As three steps above, it is necessary to solve two main problems:

- Specifying the structure of model including nodes and arcs. This task is development of qualitative model done by experts or by learning algorithms. Experts may be teachers, lecturers, supervisors, etc.
- Specifying the important parameters which are CPT (s) of all variables. This task called development of quantitative model is described right now.

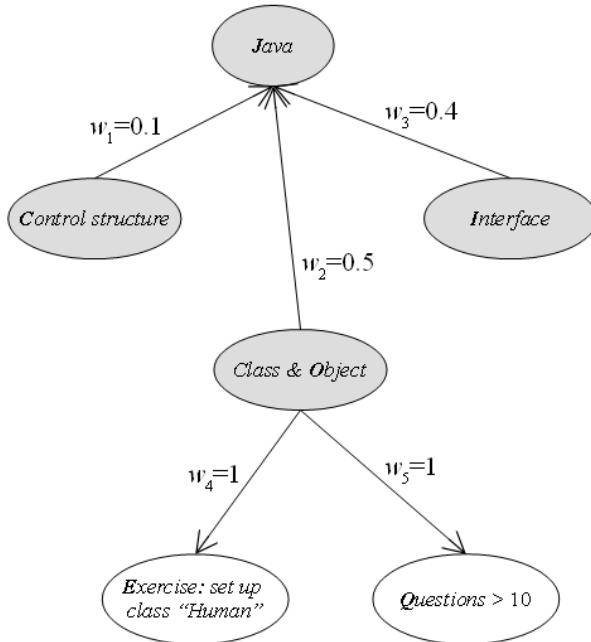
Three proposed steps to combine overlay model and BN is illustrated by an application of "how to design online Java course in order to teach students and assess their study results" with note that Java is popular object-oriented programming language <https://www.oracle.com/java>. Suppose Java course is constituted of four concepts considered as hidden nodes whose links are aggregation relationships. Hidden nodes represent learning concepts such as "**Java**", "**Control structure**", "**Class & Object**" and "**Interface**" with note that target node "**Java**" represents whole course. Additionally, there are two evidence nodes such as "**Questions > 10**" and "**Exercise: set up class Human**". That learner asks more than 10 questions is to tell how much her/his amount of knowledge. Like that, evidence "**Exercise: set up class Human**" proves whether

or not she/he understands concept “*Class & Object*”. All nodes and arcs constitute a graph with note that every node is binary random variable. The number in range [0, 1] that measures the relative importance of each relationship is defined by expert or teacher. In other words, this is the weight of arc from parent node to child node. All weights concerning the child variable will build up its CPT. Sum of weights of all arcs to each child node should be 1. It means that each weight is normalized.

Your attention please, the relationship between hidden variable and evidence variable must be from hidden variable to evidence variable because the process that computes posterior probability of hidden variable with evidence is the knowledge diagnosis. So, evidence variable has no child and its parents must be hidden variables. In short, there are two kinds of relationships:

- Aggregation relationships among hidden variables.
- Diagnostic relationships of hidden variables to evidences. The mastery of hidden concepts effects on the trust of evidences. However, if learner failed an examination, it is not sure about her/his lack of knowledge or ability because she/he can make a mistake unexpectedly.

If we understand deeply BN, both aggregation relationship and diagnostic relationship are variants of cause-effect relationship. Figure III.1.2 depicts the weighted graph representing Java course with note that such graph is the structure of Bayesian overlay model.



**Figure III.1.2.** Structure of Bayesian overlay model for Java course

Note that evidence nodes are unshaded; otherwise, hidden nodes are shaded. Now it is necessary to specify CPT (s) of variables in Bayesian network (Bayesian overlay model).

### Specifying CPT (s) of variables

It is easy to recognize that this weighted graph or overlay model is a composite sigma graph when the target node  $J$  (Java) is the aggregation of nodes  $C$  (Control structure),  $O$  (Class & Object), and  $I$  (Interface) and arcs express aggregation relationships. What we need to do now is to transform the weighted graph into Bayesian network by applying SIGMA-gate inference with attention that sigma graph and SIGMA-gate inference are described particularly in next sub-section [III.1.1.3](#). In this example, target node  $J$  has three source parents such as  $C$ ,  $O$ , and  $I$  which in turn are corresponding to three weights of aggregation relationships such as  $w_1=0.1$ ,  $w_2=0.5$  and  $w_3=0.4$ . These weights are also the prior probabilities (CPT) of nodes  $C$ ,  $O$  and  $I$  which are specified in table [III.1.1](#).

$P(C=1) = w_1 = 0.1$	$P(C=0) = 0.9$
$P(O=1) = w_2 = 0.5$	$P(O=0) = 0.5$
$P(I=1) = w_3 = 0.4$	$P(I=0) = 0.6$

**Table III.1.1.** Prior probabilities (CPT) of nodes  $C$ ,  $O$  and  $I$

It is easy to infer that target node  $J$  is the sigma sum of source nodes  $C$ ,  $O$  and  $I$ . By applying SIGMA-gate inference, the conditional probabilities or CPT of node  $J$  is totally determined. For example, the conditional probability of  $J=1$  given  $C=1$ ,  $O=1$ , and  $I=1$  is computed by formula [III.1.7](#) as follows:

$$\begin{aligned}
 P(J = 1 | C = 1, O = 1, I = 1) \\
 &= h_1 P(C = 1) + h_2 P(O = 1) + h_3 P(I = 1) \\
 \text{Where } h_1 &= \begin{cases} 1 & \text{if } C = J \\ 0 & \text{else} \end{cases}, h_2 = \begin{cases} 1 & \text{if } O = J \\ 0 & \text{else} \end{cases}, \text{and } h_3 = \begin{cases} 1 & \text{if } I = J \\ 0 & \text{else} \end{cases} \\
 \Rightarrow P(J = 1 | C = 1, O = 1, I = 1) \\
 &= 1 * P(C = 1) + 1 * P(O = 1) + 1 * P(I = 1) \\
 &= 1 * 0.1 + 1 * 0.5 + 1 * 0.4 = 1.0
 \end{aligned}$$

*Formula III.1.7.* Conditional probability of  $J=1$  given  $C=1$ ,  $O=1$ , and  $I=1$

Table [III.1.2](#) shows the CPT of node  $J$ , which is calculated by the same way to formula [III.1.7](#). Recall that the formula [III.1.7](#) is based on the theorem of SIGMA-gate inference which will be introduced in next sub-section [III.1.1.3](#).

$P(J=1   C=1, O=1, I=1) = 1*0.1 + 1*0.5 + 1*0.4 = 1.0$
$P(J=1   C=1, O=1, I=0) = 1*0.1 + 1*0.5 + 0*0.4 = 0.6$
$P(J=1   C=1, O=0, I=1) = 1*0.1 + 0*0.5 + 1*0.4 = 0.5$
$P(J=1   C=1, O=0, I=0) = 1*0.1 + 0*0.5 + 0*0.4 = 0.1$
$P(J=1   C=0, O=1, I=1) = 0*0.1 + 1*0.5 + 1*0.4 = 0.9$
$P(J=1   C=0, O=1, I=0) = 0*0.1 + 1*0.5 + 0*0.4 = 0.5$
$P(J=1   C=0, O=0, I=1) = 0*0.1 + 0*0.5 + 1*0.4 = 0.4$
$P(J=1   C=0, O=0, I=0) = 0*0.1 + 0*0.5 + 0*0.4 = 0.0$
$P(J=0   C=1, O=1, I=1) = 0*0.1 + 0*0.5 + 0*0.4 = 0.0$
$P(J=0   C=1, O=1, I=0) = 0*0.1 + 0*0.5 + 1*0.4 = 0.4$

$P(J=0   C=1, O=0, I=1) = 0*0.1 + 1*0.5 + 0*0.4 = 0.5$
$P(J=0   C=1, O=0, I=0) = 0*0.1 + 1*0.5 + 1*0.4 = 0.9$
$P(J=0   C=0, O=1, I=1) = 1*0.1 + 0*0.5 + 0*0.4 = 0.1$
$P(J=0   C=0, O=1, I=0) = 1*0.1 + 0*0.5 + 1*0.4 = 0.5$
$P(J=0   C=0, O=0, I=1) = 1*0.1 + 1*0.5 + 0*0.4 = 0.6$
$P(J=0   C=0, O=0, I=0) = 1*0.1 + 1*0.5 + 1*0.4 = 1.0$

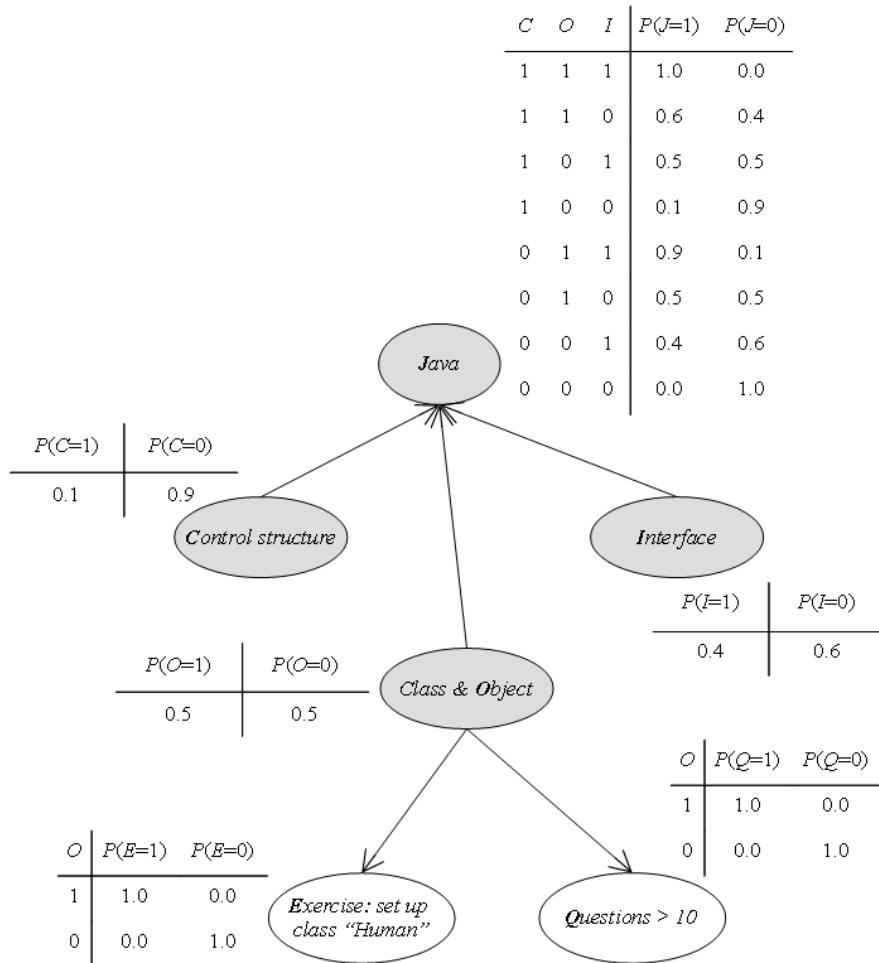
**Table III.1.2.** CPT of node  $J$ 

Let  $w_4 = 0.8$  and  $w_5 = 0.2$  be weights of diagnostic relationships from hidden node  $O$  to evidence nodes  $E$  (Exercise: set up class Human) and  $Q$  (Questions > 10), the CPT (s) of  $E$  and  $Q$  are totally determined as in table III.1.3.

$P(E=1   O=1) = 1$	$P(E=0   O=1) = 0$
$P(E=1   O=0) = 0$	$P(E=0   O=0) = 1$
$P(Q=1   O=1) = 1$	$P(Q=0   O=1) = 0$
$P(Q=1   O=0) = 0$	$P(Q=0   O=0) = 1$

**Table III.1.3.** CPT (s) of evidence nodes  $E$  and  $Q$ 

Hence, the Java course overlay model is transformed totally into Bayesian network (Bayesian overlay model) as figure III.1.3:



**Figure III.1.3.** Bayesian network (Bayesian overlay model) of Java course with full of CPT (s)

Note that evidence nodes are unshaded; otherwise, hidden nodes are shaded. When Bayesian network is determined, it is easy to infer or assess user's knowledge based on inference mechanism (see sub-section [III.1.1.1](#)) inside Bayesian network.

### Inferring user's knowledge

Suppose a leaner did well the exercise “Set up class *Human*” and asked more than 10 questions. That is to say the occurrence of two evidences, namely,  $E=1$  and  $Q=1$ . It is necessary to answer the question: How mastered is learner over the concept “Java”? Thus, the posterior conditional  $P(J=1 | C, O, I, E=1, Q=1)$  of hidden variables  $J$  with fixed events  $E=1$  and  $Q=1$  must be computed. According to formula [III.1.3](#), we have:

$$P(J = 1 | C, O, I, E = 1, Q = 1) = \frac{\sum_{C,O,I} P(J = 1, C, O, I, E = 1, Q = 1)}{\sum_{C,O,I,E,Q} P(J = 1, C, O, I, E, Q)}$$

Where  $P(J, C, O, I, E, Q)$  is global joint probability distribution. Applying formula [III.1.2](#), we have:

$$P(J, C, O, I, E, Q) = P(C) * P(O) * P(I) * P(E|O) * P(Q|O) * P(J|C, O, I)$$

Applying all CPT (s) in table [III.1.1](#), [III.1.2](#), and [III.1.3](#), it is able to determined  $P(J, C, O, I, E, Q)$ . After that, we compute  $P(J=1 / C, O, I, E=1, Q=1)$  to answer above question.

Note, the set of all parents of a hidden node is the complete set of mutually exclusive hidden variables and the set of all evidence nodes which are children of a hidden node is the complete set of mutually exclusive evidence variables.

Now the combination approach to construct Bayesian model is described thoroughly in this sub-section [III.1.1.2](#). Next sub-section [III.1.1.3](#) is the theoretical proof for such proposed approach.

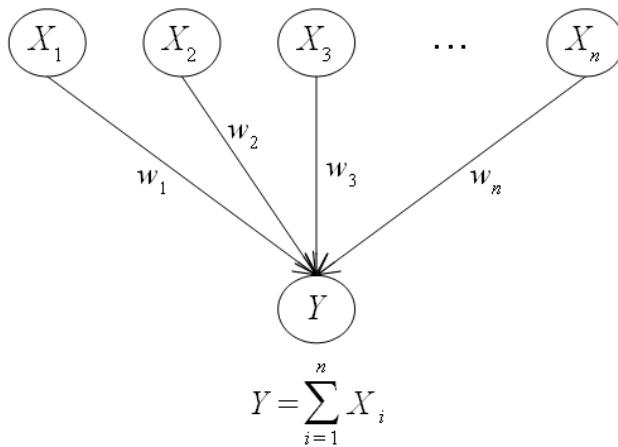
### III.1.1.3. SIGMA-gate inference

The main task of applying Bayesian network into overlay model mentioned in previous sub-section [III.1.1.2](#) is to specify or determine CPT (s) of variables. This is the learning parameter problem when CPT (s) are considered as quantitative parameters. So, the essence of formula [III.1.7](#) specifying conditional probability of random variable is merely the learning parameter problem. I propose a theorem of SIGMA-gate inference in this sub-section [III.1.1.3](#) which is the base of formula [III.1.7](#), also the base of the combination of overlay model and Bayesian network mentioned in previous sub-section [III.1.1.2](#). Given a Bayesian network consisting of a directed acyclic graph (DAG), we narrow the learning parameter problem “how to determine CPT (s) of such Bayesian network” into special DAG so-called aggregation graph in which child node is the aggregation of parent nodes; in other words, each arc expresses an aggregation relationship. If parent nodes are mutually independent, aggregation network becomes *sigma graph*. In other words, all parent nodes in sigma graph constitute a complete set of mutually exclusive

random variables. Given sigma graph contains a set of parent nodes  $X_1, X_2, \dots, X_n$  and a child node  $Y$  where  $Y$  is the sigma sum of all  $X_i$  (s) and each arc  $X_i \rightarrow Y$  is weighted.

$$Y = \bigcup_{i=1}^n X_i = \sum_{i=1}^n X_i \text{ where } X_i \cap X_j = \emptyset, \forall 1 \leq i, j \leq n$$

Note that  $X_i$  and  $X_j$  are binary random variables and so we use notation  $X_i \cap X_j = \emptyset$  to denote that  $X_i$  (s) are mutually independent. Hence, the *sigma sum* is interpreted that node  $Y$  is exclusive aggregation or exclusive union of nodes  $X_i$  (s) and so, the sigma sum sign  $\sum$  does not express arithmetical addition. Figure III.1.4 depicts sigma graph.



**Figure III.1.4.** Sigma graph

It is easy to recognize that nodes in Java course network such as  $J$  (Java),  $C$  (Control structure),  $O$  (Class & Object), and  $I$  (Interface) together with aggregation relationships depicted in figure III.1.2 mentioned in previous subsection III.1.1.2 compose a sigma graph.

Now we prefer to use terms “sigma sum” instead of “exclusive aggregation” or “exclusive union” as usual. In sigma graph, parent node is called *partial node* or *source node* and child node is called *aggregative node* or *target node*. Note that statement “target node  $Y$  is aggregation of sources node  $X_i$  (s)” is the same to statement “each source node  $X_i$  is integrated into target node  $Y$ ”.

The main problem is how to transform sigma graph into Bayesian network; in other words, it is necessary to calculate all CPT (s) attached to nodes. Such problem is also called parameter learning. The theorem of SIGMA-gate inference will be stated, which helps us to solve this problem.

Suppose every node is binary, SIGMA-gate inference associated to sigma graph is based on three assumptions:

- *Aggregation inhibition*: Given an aggregation relationship denoted by arc  $X \rightarrow Y$ , there is a factor  $I$  that inhibits  $X$  from integrated into  $Y$ . Factor  $I$  is called inhibition of  $X$ . That the inhibition  $I$  is turned off is the prerequisite of  $X$  integrated into  $Y$ .

$$I = 0 \Leftrightarrow I \text{ turned OFF}$$

$$I = 1 \Leftrightarrow I \text{ turned ON}$$

### III.1. Knowledge sub-model

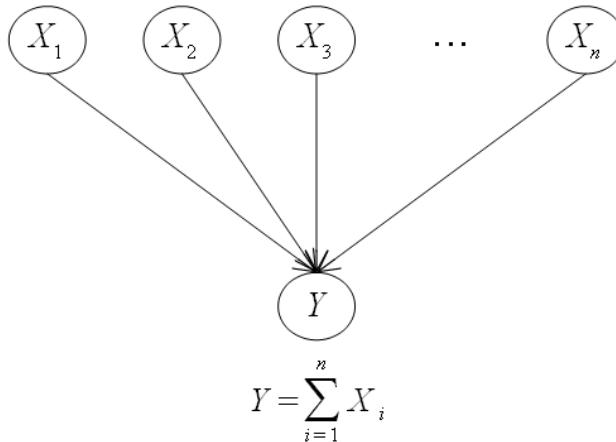
---

- *Inhibition independence*: Inhibitions are mutually independent. For example, inhibition  $I_1$  of  $X_1$  is independent from inhibition  $I_2$  of  $X_2$ .
- *Sigma condition*: Suppose we have a set of aggregation relationships in which  $Y$  is the aggregation of many sources  $X_1, X_2, \dots, X_n$  and let  $I_i$  be the inhibition of  $X_i$ , the sigma condition states that “the target  $Y$  is the sigma sum of all sources  $X_i$  (s)”. So the sigma condition is merely the main aspect of sigma graph as aforementioned. Sigma condition is formulated in formula III.1.8 as follows:

$$Y = \bigcup_{i=1}^n X_i = \sum_{i=1}^n X_i \text{ where } X_i \cap X_j = \emptyset, \forall 1 \leq i, j \leq n$$

*Formula III.1.8.* Sigma condition

Concepts “aggregation inhibition”, “inhibition independence” and “sigma condition” are inspired from concepts “cause inhibition”, “exception independence” and “accountability” of noisy OR-gate model (Neapolitan, 2003, p. 157) in Bayesian network inference. Figure III.1.5 also depicts sigma condition.



**Figure III.1.5.** Sigma condition

Suppose we have  $n$  sources  $X_1, X_2, \dots, X_n$  and one target  $Y$ . According to “aggregation inhibition” and “inhibition independence” assumptions and let  $I_i$  be the inhibition of  $X_i$ . Let  $A_i$  be dummy variable so that  $A_i$  is *ON* ( $=1$ ) if  $X_i$  is equal to 1 and  $I_i$  is *OFF* ( $=0$ ), we have:

$$P(A_i = \text{ON} | X_i = 1, I_i = \text{OFF}) = 1$$

$$P(A_i = \text{ON} | X_i = 1, I_i = \text{ON}) = 0$$

$$P(A_i = \text{ON} | X_i = 0, I_i = \text{OFF}) = 0$$

$$P(A_i = \text{ON} | X_i = 0, I_i = \text{ON}) = 0$$

$$P(A_i = \text{OFF} | X_i = 1, I_i = \text{OFF}) = 0$$

$$P(A_i = \text{OFF} | X_i = 1, I_i = \text{ON}) = 1$$

$$P(A_i = \text{OFF} | X_i = 0, I_i = \text{OFF}) = 1$$

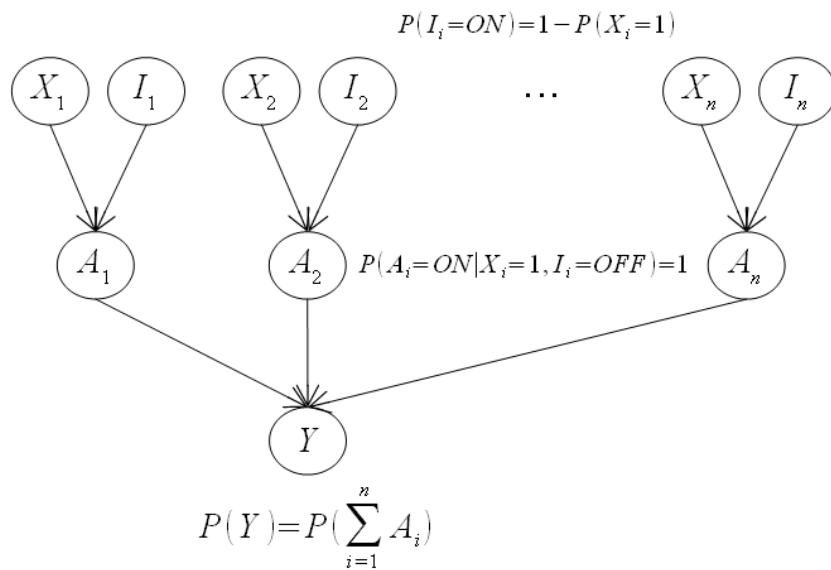
$$P(A_i = \text{OFF} | X_i = 0, I_i = \text{ON}) = 1$$

According to sigma condition, the condition probability of  $Y$  is the probability of sigma sum of all  $A_i$  (s) where  $A_i$  (s) are mutually independent, as seen in formula III.1.9 as follows:

$$P(Y) = P\left(\bigcup_{i=1}^n A_i\right) = P\left(\sum_{i=1}^n A_i\right) \text{ where } A_i \cap A_j = \emptyset, \forall 1 \leq i, j \leq n$$

*Formula III.1.9.* Probability of sigma sum

Because  $A_i$  (s) are binary random variables, the sigma sum  $\sum_{i=1}^n A_i$  is interpreted that variable  $Y$  is exclusive aggregation or exclusive union of variables  $A_i$  (s). Figure III.1.6 depicts SIGMA-gate model with regard to dummy variables  $A_i$  (s).



**Figure III.1.6.** SIGMA-gate model

Now the strength of each aggregation relationship  $X_i \rightarrow Y$  is quantified by the CPT  $P(Y | X_i)$ . Suppose sources  $(X_1, X_2, \dots, X_i, \dots, X_n)$  become evidences having values  $(x_1, x_2, \dots, x_i, \dots, x_n)$ . Let  $P(X_i = 1) = p_i$  be the probability of  $X_i = 1$ , the probability of inhibition of  $X_i$  is the inverse of  $P(X_i = 1)$ .

$$P(I_i=ON) = 1 - P(X_i=1) = 1 - p_i$$

$$P(I_i=OFF) = P(X_i=1) = p_i$$

Please pay attention that the set  $(X_1, X_2, \dots, X_i, \dots, X_n)$  is a partition of probability space; in other words, sum of all probabilities  $P(X_i = 1)$  must be equal to 1.

$$\sum_{i=1}^n P(X_i = 1) = \sum_{i=1}^n p_i = 1$$

Let  $K$  be the set of  $i$  (s) such that  $X_i = 1$

$$\forall i \in K, X_i = 1$$

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

### III.1. Knowledge sub-model

---

$$\begin{aligned}
& P(Y|X_1, X_2, \dots, X_i, \dots, X_n) \\
&= P\left(\sum_{i=1}^n A_i \mid X_1, X_2, \dots, X_i, \dots, X_n\right) \text{ (due to SIGMA condition)} \\
&= \sum_{i=1}^n P(A_i|X_1, X_2, \dots, X_i, \dots, X_n) \text{ (because } A_i \text{ (s) are mutually independent)} \\
&= \sum_{i=1}^n P(A_i|X_i) \text{ (because } A_i \text{ is only dependent on } X_i)
\end{aligned}$$

Suppose  $Y$  is instantiated as 1, we have:

$$\begin{aligned}
& P(Y = 1|X_1 = x_1, X_2 = x_2, \dots, X_i = x_i, \dots, X_n = x_n) \\
&= \sum_{i=1}^n P(A_i = ON|X_i = x_i) \\
&= \sum_{i=1}^n (P(A_i = ON|X_i = x_i, I_i = ON)P(I_i = ON) \\
&\quad + P(A_i = ON|X_i = x_i, I_i = OFF)P(I_i = OFF))
\end{aligned}$$

Applying the total probability rule (see formula III.1.1d) with regard to the set  $K$  of  $i$  (s) such that  $X_i = 1$ , we have:

$$\begin{aligned}
& P(Y = 1|X_1 = x_1, X_2 = x_2, \dots, X_i = x_i, \dots, X_n = x_n) \\
&= \left( \sum_{i \in K} (P(A_i = ON|X_i = 1, I_i = ON)P(I_i = ON) \right. \\
&\quad \left. + P(A_i = ON|X_i = 1, I_i = OFF)P(I_i = OFF)) \right) \\
&\quad + \left( \sum_{i \notin K} (P(A_i = ON|X_i = 0, I_i = ON)P(I_i = ON) \right. \\
&\quad \left. + P(A_i = ON|X_i = 0, I_i = OFF)P(I_i = OFF)) \right) \\
&= \sum_{i \in K} (0 * (1 - p_i) + 1 * p_i) + \sum_{i \notin K} (0 * (1 - p_i) + 0 * p_i) \\
&= \sum_{i \in K} p_i = \sum_{i \in K} P(X_i = 1) = \sum_{i=1}^n h_i P(X_i = 1)
\end{aligned}$$

Where,

$$h_i = \begin{cases} 1 & \text{if } X_i = 1 \\ 0 & \text{if } X_i \neq 1 \end{cases}$$

In conclusion, the theorem of SIGMA-gate inference states that *given target variable  $Y$  which is sigma sum of mutually independent source variables  $X_i$  (s),*

*the probability of  $Y$  is the sum of prior probabilities of  $X_i$  (s) which are equal to  $Y$  as follows:*

$$P(Y|X_1, X_2, \dots, X_n) = \sum_{i=1}^n h_i P(X_i = Y)$$

Where,

$$h_i = \begin{cases} 1 & \text{if } X_i = Y \\ 0 & \text{if } X_i \neq Y \end{cases}$$

*Formula III.1.10. Theorem of SIGMA-gate inference*

Please pay attention that the sum  $\sum_{i=1}^n P(X_i = 1)$  must be equal to 1 because the set  $(X_1, X_2, \dots, X_i, \dots, X_n)$  is a partition of probability space.

Going back issued problem with sigma graph when probabilities of source nodes  $X_i$  (s) are not specified; in other words, probabilities  $P(X_i)$  are not defined but each aggregation arc is weighted. Suppose  $w_i$  is the weight of arc  $X_i \rightarrow Y$  from node  $X_i$  to  $Y$ , the theorem of SIGMA-gate inference (formula III.1.10) restates that *given target variable  $Y$  which is sigma sum of mutually independent source variables  $X_i$  (s), the probability of  $Y$  is the sum of weights  $w_i$  (s) with condition that the respective  $X_i$  (s) are equal to  $Y$ .*

$$P(Y|X_1, X_2, \dots, X_n) = \sum_{i=1}^n h_i w_i$$

Where,

$$h_i = \begin{cases} 1 & \text{if } X_i = Y \\ 0 & \text{if } X_i \neq Y \end{cases}$$

*Formula III.1.11. Theorem of SIGMA-gate inference with weighted graph*

Please pay attention that sum of all  $w_i$  (s) must be equal to 1; in other words all weights  $w_i$  (s) are normalized,  $\sum_{i=1}^n w_i = 1$ . It is very easy to prove this variant of SIGMA-gate inference theorem. That the arc  $X_i \rightarrow Y$  is weighted implies that the prior probability of  $(X_i = 1)$  is equal to  $w_i$ .

$$\begin{aligned} P(X_i = 1) &= w_i \\ P(X_i = 0) &= 1 - w_i \end{aligned}$$

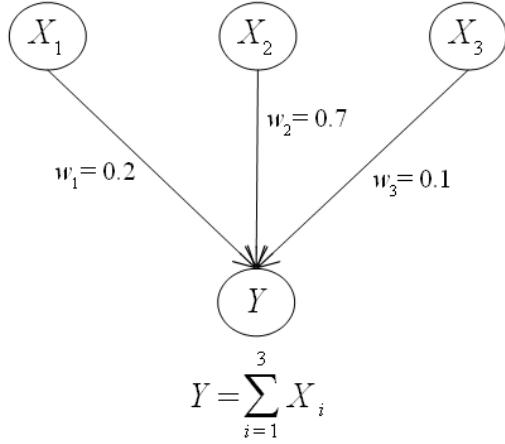
Therefore, we have:

$$P(Y = 1|X_1, X_2, \dots, X_n) = \sum_{i=1}^n h_i P(X_i = 1) = \sum_{i=1}^n h_i w_i$$

Where,

$$h_i = \begin{cases} 1 & \text{if } X_i = Y \\ 0 & \text{if } X_i \neq Y \end{cases}$$

In general, the proof of SIGMA-gate inference is inspired from noisy OR-gate model (Neapolitan, 2003, p. 157) in Bayesian network. Now it is easy to transform the weighted sigma graph into Bayesian network. Given example where sigma graph has one target node  $Y$  and three source nodes  $X_1, X_2$  and  $X_3$  whose weights are  $w_1 = 0.2, w_2 = 0.7$  and  $w_3 = 0.1$ , respectively. Figure III.1.7 depicts given weighted sigma graph.



**Figure III.1.7.** An example of sigma graph

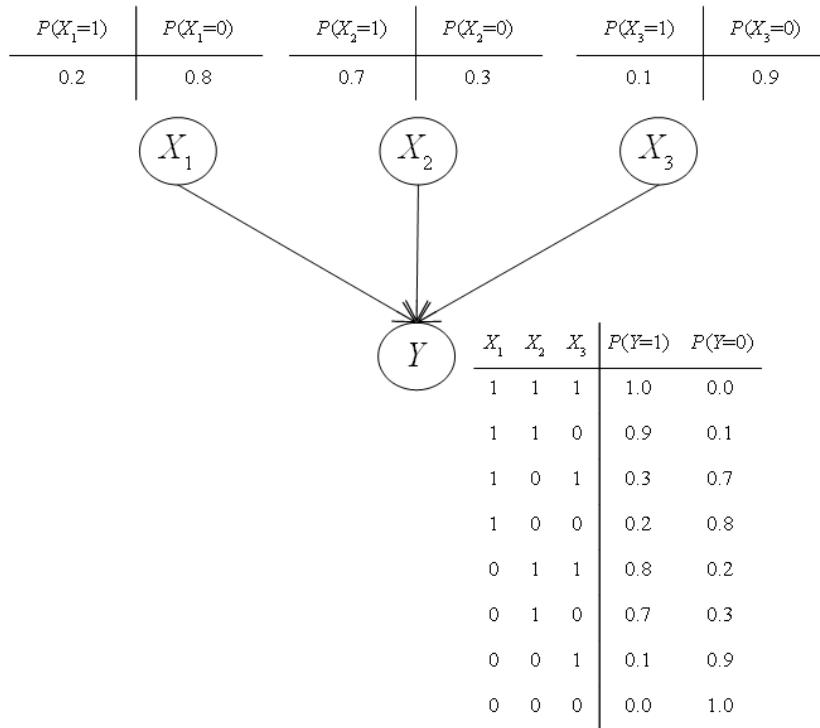
Applying theorem of SIGMA-gate inference specified in formula III.1.11, it is possible to determine prior probabilities of source nodes  $X_1$ ,  $X_2$  and  $X_3$  and conditional probability (CPT) of target node  $Y$  as below:

$$\begin{aligned} P(X_1=1) &= w_1 = 0.2 & P(X_1=0) &= 0.8 \\ P(X_2=1) &= w_2 = 0.7 & P(X_2=0) &= 0.3 \\ P(X_3=1) &= w_3 = 0.1 & P(X_3=0) &= 0.9 \end{aligned}$$

$$\begin{aligned} P(Y_1=1 | X_1=1, X_2=1, X_3=1) &= 1*0.2 + 1*0.7 + 1*0.1 = 1.0 \\ P(Y_1=1 | X_1=1, X_2=1, X_3=0) &= 1*0.2 + 1*0.7 + 0*0.1 = 0.9 \\ P(Y_1=1 | X_1=1, X_2=0, X_3=1) &= 1*0.2 + 0*0.7 + 1*0.1 = 0.3 \\ P(Y_1=1 | X_1=1, X_2=0, X_3=0) &= 1*0.2 + 0*0.7 + 0*0.1 = 0.2 \\ P(Y_1=1 | X_1=0, X_2=1, X_3=1) &= 0*0.2 + 1*0.7 + 1*0.1 = 0.8 \\ P(Y_1=1 | X_1=0, X_2=1, X_3=0) &= 0*0.2 + 1*0.7 + 0*0.1 = 0.7 \\ P(Y_1=1 | X_1=0, X_2=0, X_3=1) &= 0*0.2 + 0*0.7 + 1*0.1 = 0.1 \\ P(Y_1=1 | X_1=0, X_2=0, X_3=0) &= 0*0.2 + 0*0.7 + 0*0.1 = 0.0 \end{aligned}$$

$$\begin{aligned} P(Y_1=0 | X_1=1, X_2=1, X_3=1) &= 0*0.2 + 0*0.7 + 0*0.1 = 0.0 \\ P(Y_1=0 | X_1=1, X_2=1, X_3=0) &= 0*0.2 + 0*0.7 + 1*0.1 = 0.1 \\ P(Y_1=0 | X_1=1, X_2=0, X_3=1) &= 0*0.2 + 1*0.7 + 0*0.1 = 0.7 \\ P(Y_1=0 | X_1=1, X_2=0, X_3=0) &= 0*0.2 + 1*0.7 + 1*0.1 = 0.8 \\ P(Y_1=0 | X_1=0, X_2=1, X_3=1) &= 1*0.2 + 0*0.7 + 0*0.1 = 0.2 \\ P(Y_1=0 | X_1=0, X_2=1, X_3=0) &= 1*0.2 + 0*0.7 + 1*0.1 = 0.3 \\ P(Y_1=0 | X_1=0, X_2=0, X_3=1) &= 1*0.2 + 1*0.7 + 0*0.1 = 0.9 \\ P(Y_1=0 | X_1=0, X_2=0, X_3=0) &= 1*0.2 + 1*0.7 + 1*0.1 = 1.0 \end{aligned}$$

Figure III.1.8 depicts Bayesian network transformed from weighted sigma graph.



**Figure III.1.8.** Bayesian network transformed from sigma graph

The example of transforming sigma graph into Bayesian network ends up this sub-section III.1.1.3 with full of explanations of SIGMA-gate theorem. Next sub-section III.1.1.4 is the evaluation of proposed method to combine overlay model and Bayesian network which is based on SIGMA-gate inference.

### III.1.1.4. Evaluation

There is no doubt that the combination of BN and overlay model gives us the appropriate approach for user modeling but it has two disadvantages:

- The expense of data storage is high. A BN which has  $n$  variables together  $n$  CPT (s) with  $2^n$  parameters (values in CPT (s)) under constraint: “each variable is binary (0 and 1)”. If variables are not binary, the number of parameters is large, so, it is difficult to store them in memory.
- The computation of posterior probability which is basis of inference consumes much time when executing in runtime because it is rather complex.

The first cause which is the inherent attribute of BN can be only restricted by programming technique when implementing network and it would be best to declare binary variables. On the other hand, it is the done to use CPT instead of continuous probability density/distribution function for solving the second problem.

As already discussed, the structure and parameters (CPT (s)) in my model are fixed and specified by experts. However, they must be evolved after each occurrence of evidence, please see sub-section III.1.3 discussing evolution of Bayesian overlay model. When machine learning is concerned, structural

learning is process of gradual improving the structure of model and correspondingly, parametric learning is process of changing the parameters so as to be more suitable. I will discuss the improvement on qualitative model (structure) and quantitative model (parameters) in sub-sections III.1.4 and III.1.5. Note that the knowledge sub-model created by combining Bayesian network and overlay model is called as Bayesian overlay model or Bayesian model or Bayesian network in brief. Before discussing improvement of Bayesian model, the incorporation of inference mechanism in Bayesian network into adaptive system is described in successive sub-section III.1.2.

#### III.1.2. Incorporate Bayesian inference into adaptation rules

Adaptive hypermedia system (AHS) and adaptive educational hypermedia system (AEHS) are powerful adaptive system, which aims to provide users the adaptation effect based on their characteristics; please see sub-sections I.2.1 and I.2.3 for more details about AHS and AEHS. In other words, AEHS ensures that hypermedia contents including links and information web pages (Wikipedia, Web page, 2014) are offered and adapted to each individual user. AHA!, a typical AHS, developed by the author Paul De Debra (De Bra & Calvi, 1998) is an open Adaptive Hypermedia for All that is suitable for many different applications; it aims to generic purpose. The architecture of AHA! based on Dexter Hypertext Reference Model (Halasz & Schwartz, 1990, p. 3) has some prominences but the inside user model is built up by overlay method in which the domain is decomposed into a set of elements and the overlay is simply a set of masteries over those elements. Although overlay model (sub-section I.1.2.2) is easy to represent user information, there is no inference mechanism for reasoning out new assumptions about user. All AHS (s) have the adaptation model containing adaptation rules but I use AHA! as a sample AHS for my method. So I propose a new way to incorporating Bayesian inference into AHA! so that it is able to improve modeling functionality in AHA!.

##### AHA! models and adaptation rules

I have introduced AHA! in sub-section I.2.3.3 but now we should survey it in detailed. The AHA! engine (De Bra, Smits, & Stash, 2006) manipulates three main models such as domain model, user model and adaptation model. Such models are described as below.

*Domain model* (De Bra, Stash, & Smits, 2005, p. 9) consists of concepts which are topics in the application domain. Each concept has:

- A unique identifier (c-id).
- A description structure called “*concept information*” (c-info) is constituted of three fixed parts, namely a *sequence of anchors*, a *presentation specification* and a set of *attribute-value pairs*.

The sequence of anchors describes sub-structures within a concept. These can be used as anchor point for links such as the source or destination of links. The

presentation specification navigates the runtime layer how to display concept's content. The attribute-value pairs are arbitrary. Each of them tells us optional information of concept and depends on idea of experts. Some usual attributes are shown in below:

- *access*: when a concept is accessed, this attributed is triggered. Attribute "access" is the starting point of process of executing adaptation rules.
- *knowledge*: amount of knowledge that user are mastered over concept.
- *visited*: this attribute indicates whether user visited the page associated with concept or not.

*User model* contains important information about user such as knowledge, learning style, goals, and background. User model is used as basis of adaptive process. In overlay method, user model is the subset of domain model. Namely, domain is decomposed into a set of concepts and overlay model is simply the set of masteries over these concepts. In this section III.1.2, user model is implicated as overlay model.

*Adaptation model* (AM) is responsible for associating user model and domain model to generate adaptation. AM contains a set of adaptation rules used to tailor learning concept & material to user characteristics in user model. Rules are categorized into two classes associated two purposes:

- Updating user model.
- Defining adaptation effect by setting presentation specification. For example, if user is mastered over concept "Control Structure", she/he should be recommended concept "Loop" in programming language course.

In the overlay model, with each concept in domain model, there is corresponding concept in user model with the same name. So the expression "*concept.attribute*" refers both the domain model attribute and user model attribute. The adaptation rules whose purpose is to update user model are called *event-condition-action* (ECA) rules because (De Bra, Stash, & Smits, 2005, p. 11):

- They are triggered by an *event*, e.g. users access a concept by following a link.
- The *condition* which is Boolean expression is evaluated. The expression has terms which are domain/user model attributes in form "*concept.attribute*".
- When the condition is satisfied, the *action* is executed. The action performs an update to attribute value and can be trigger another rule. This is propagation mechanism.

For example, the ECA rule "when the concept *C* is accessed and it was visited before, its attribute *knowledge* is set to be 100%" is interpreted as below:

Event:	access( <i>C</i> )
Condition:	<i>C.visited</i> = true
Action:	<i>C.knowledge</i> = 100%

### III.1. Knowledge sub-model

---

The adaptation rules whose purpose is to define adaptation effect are called condition-action (CA) rules. They have main responsibility for showing adaptive presentation:

- CA rules have no event. They are checked when the engine need to deliver learning objects to user.
- The condition in the form of Boolean expression in which terms are attributes is evaluated similarly to ECA rules.
- *Action* results in adaptive effect according to presentation specification.

For example, the CA rule “If user knowledge about concept *C* reaches or exceeds 50% then user is provided advanced subjects about *C*. Otherwise, user should pay attention to basic explanations”.

Condition: *C.knowledge > 50*

Action: True status: providing advanced subjects about *C*

False status: providing basic explanations about *C*

This rule can be interpreted in XML form as follows:

```
<if expr="C.knowledge > 50">
<block>
    Here advanced subjects about C for advanced user
</block>
<block>
    Here basic explanations about C for novice
</block>
</if>
```

### Incorporating Bayesian inference into adaptation rules

There are two techniques of deductive logic: forward reasoning and backward reasoning (De Bra, Smits, & Stash, The Design of AHA!, 2006):

- Suppose a student is recommended to learn concept “Class & Object” before concept “Interface & Package” in Java course with note that Java is a popular object-oriented programming language <https://www.oracle.com/java>. There is the prerequisite relationship in which “Class & Object” is the precondition of “Interface & Package”. Namely, the “recommended” attribute of “Interface & Package” becomes *true* if and only if the “knowledge” attribute of “Class & Object” reaches 50%. Whenever the “knowledge” attribute changes its values, the “recommended” attribute is checked and can be re-assigned new value (“true”). This technique is called forward reasoning because the “recommended” attribute is determined as soon as possible and thus before it is actually needed, regardless of user requirement.
- Otherwise, the “recommended” attribute is only determined when students require learning “Interface & Package”, meaning we check whether the “knowledge” attribute reaches 50%. If user does not ask, “recommended” attribute is not considered even the “knowledge” is changed. This technique is called backward reasoning because the

attribute is not pre-computed but it is tracked back when actually needed.

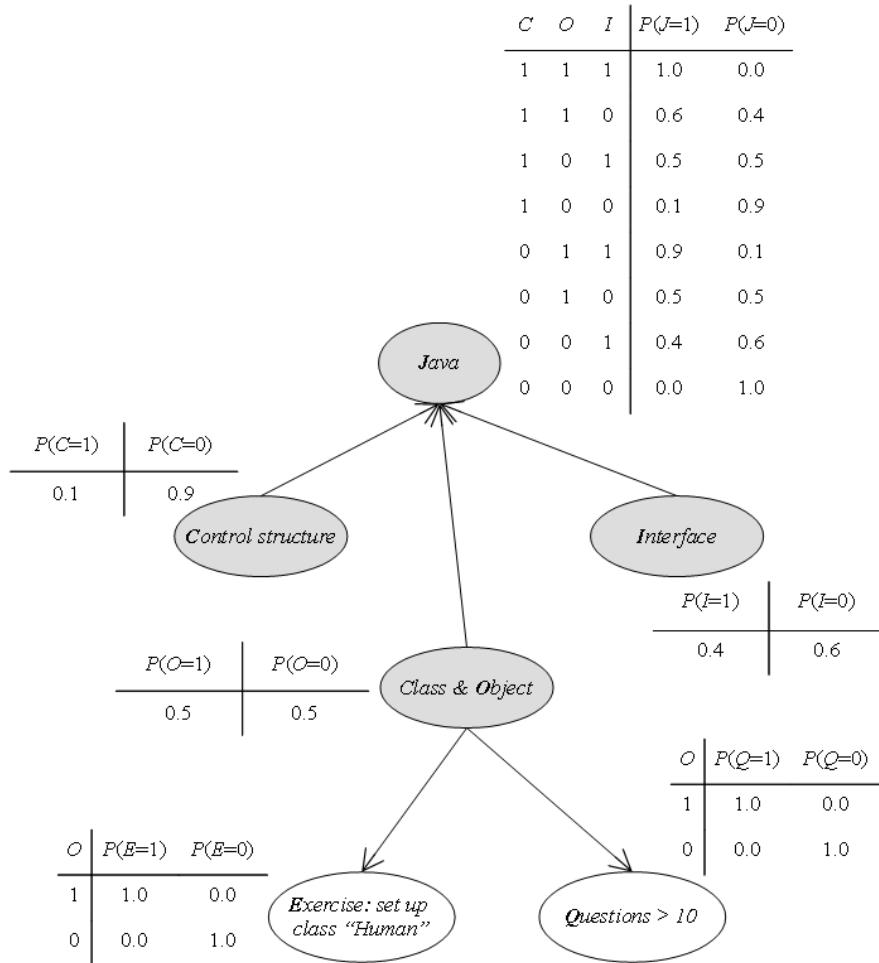
Forward reasoning and backward reasoning have both advantages and drawbacks (De Bra, Smits, & Stash, The Design of AHA!, 2006):

- The strong point of forward reasoning is that condition is evaluated immediately by checking the attribute value because it was already stored. That respond time is very fast is very useful for real-time applications. Otherwise, the drawback is that the attributes are changed many times even they are not considered yet (before actually needed).
- The advantage of backward reasoning is to avoid the drawback of backward reasoning. The condition is only checked when actually needed. But drawback is that checking attribute takes more time than forwarding reasoning because it is necessary to perform more operations in reasoning process.

#### *Dummy attribute and backward reasoning*

Reviewing Java course aforementioned in previous sub-section [III.1.1.2](#) is constituted of three concepts considered as knowledge variables whose links are dependency relationships. Additionally, there are two evidence variables: “*Questions > 10*” and “*Exercise: set up class Human*”. That learner asks more than 10 questions is to tell how much her/his amount of knowledge. Like that, 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. Figure [III.1.9](#) depicts the Bayesian overlay model of the Java course in which CPT (s) are specified according to combination method mentioned in previous sub-section [III.1.1.2](#). In order to be convenient for reader to keep tract important problems, figure [III.1.9](#) is a repetition of figure [III.1.3](#).

### III.1. Knowledge sub-model



**Figure III.1.9.** Bayesian overlay model of Java course with full of CPT (s)

Note that five concepts (nodes)  $O, I, J, E, Q$  denote knowledge variables and evidences: **Class & Object**, **Java**, **Interface**, “**Exercise: set up class Human**” and “**Questions > 10**”, respectively. Concepts  $E, Q$  have the new attribute *is\_evidence* indicating whether  $E, Q$  are fit to become evidences or not. Concepts  $O, I, J$  have the new dummy attribute *do\$bayes\$infer*. It called “dummy” because attribute *do\$bayes\$infer* does not exists actually in user model. It is only used as the denotation for backward reasoning with Bayesian network inference. Namely,  $J.\text{do\$bayes\$infer}$  is the posterior probability tell us how mastered leaner is over the concept  $J$ . Please see sub-section III.1.1.1 for knowing posterior probability.

I defined two ECA rules for updating evidences “**Questions > 10**”, “**Exercise: set up class Human**” and one CA rule setting presentation specification.

- *The 1<sup>st</sup> ECA rule:* If user asks more than 10 questions then the attribute  $Q.\text{is\_evidence}$  is set to be *true*.
- *The 2<sup>nd</sup> ECA rule:* If user does exercise “**Exercise: set up class Human**” the attribute  $E.\text{is\_evidence}$  is set to be *true*.
- *The 3<sup>rd</sup> CA rule:* If the probability of user knowledge about concept Java reaches or exceeds 0.5 then user is provided advanced subjects about Java. Otherwise, user should pay attention to basic explanations. The

probability of user knowledge about concept Java is denoted as dummy attribute  $J.\text{do\$bayes\$infer}$ .

These rules are interpreted in table [III.1.4](#) as follows:

	Event	Condition	Action
<i>Rule 1<sup>st</sup></i>	access(Q)	Q.visited > 10	Q.is_evidence = true
<i>Rule 2<sup>nd</sup></i>	access(E)	E.visited > 10	E.is_evidence = true
<i>Rule 3<sup>rd</sup></i>		$J.\text{do\$bayes\$infer} \geq 0.5$	<ul style="list-style-type: none"> <li>- <i>True status</i>: providing advanced subjects about Java.</li> <li>- <i>False status</i>: providing basic explanations about Java.</li> </ul>

**Table III.1.4.** ECA and CA rules for Bayesian inference

The 3<sup>rd</sup> rule is translated in XML form as follows:

```
<if expr="J.\text{do\$bayes\$infer} > 0.5">
<block>
    Here advanced subjects about Java for advanced user
</block>
<block>
    Here basic explanations about Java for novice
</block>
</if>
```

In the 3<sup>rd</sup> rule, attribute  $J.\text{do\$bayes\$infer}$  is not stored and not checked instantly but whenever adaptive engine requires to determine its value, it will be tracked back and computed by Bayesian inference. Therefore, this is backward reasoning. For example, after the 1<sup>st</sup> and 2<sup>nd</sup> rules are executed, concepts  $Q$  and  $E$  become actual evidences. According to the formula [III.1.3](#)' for calculating posterior probability in Bayesian network, the attribute  $J.\text{do\$bayes\$infer}$  that represents user's knowledge about concept  $J$  is determined as follows:

$$\begin{aligned} J.\text{do\$bayes\$infer} &= P(J = 1 | Q = 1, E = 1) \\ &= \frac{\sum_{C,O,I} P(J = 1, C, O, I, E = 1, Q = 1)}{\sum_{J,C,O,I} P(J, C, O, I, E = 1, Q = 1)} \end{aligned}$$

Where  $P(J, C, O, I, E, Q)$  is global joint probability distribution. According to formula [III.1.2](#)' for specifying global joint probability distribution in effective way, we have:

$$P(J, C, O, I, E, Q) = P(C) * P(O) * P(I) * P(E|O) * P(Q|O) * P(J|C, O, I)$$

My method (Nguyen, Incorporating Bayesian Inference into Adaptation Rules in AHA architecture, 2009) is to use dummy attribute to execute backward reasoning. Whenever it is required to compute adaptation, the dummy attribute of a domain element (variable) which denotes the posterior probability representing user's knowledge about this domain element is considered. Then adaptation rules will refer to such dummy attribute in order to decide adaptation strategies. It is possible to extend my method into other inferences such as neural network and hidden Markov model (see sub-section [III.2.4](#));

hence, there is no need to change the main technique and what we do is to add new dummy attributes to domain element.

The approach to combine overlay model and Bayesian network is proposed in sub-section [III.1.1.2](#), proved in sub-section [III.1.1.3](#) and applied in sub-section [III.1.2](#). Now it is necessary to improve Bayesian model and there are two improvements of Bayesian network such as parameter learning and structure learning. Next sub-section [III.1.3](#) discusses evolution of Bayesian overlay model which is essentially learning CPT (s) inside Bayesian network with note that CPT (s) are quantitative parameters of Bayesian network. Structure learning is mentioned in sub-section [III.1.4](#).

#### **III.1.3. Evolution of Bayesian overlay model**

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 previous sub-section [III.1.1](#). 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 sub-section [III.1.3](#) discusses 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 almost 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 (see sub-section [III.1.1](#)) will make use of each method's strong points and restraints drawbacks.

- The structure of overlay model is translated into BN, each user knowledge element becomes an node in BN.
- Each aggregation relationship between domain element in overlay model becomes an conditional dependence assertion signified by CPT of each node in BN.
- Domain elements are defined as hidden nodes and other learning objects which are used to assess user's performance are consider 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. 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. Sub-section III.1.3.1 discusses about main subject “learning parameters or the evolution of parameters” (Nguyen & Do, Evolution of parameters in Bayesian Overlay Model, 2009). Sub-section III.1.3.2 mentions how to learn parameters in case of data missing. Sub-section III.1.3.3 gives an example of learning parameters. Some works related to Bayesian network for modeling user are discussed in section I.3 and most of them do not have mechanism for the evolution of BN like this research.

### III.1.3.1. 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 sub-section III.1.3.1 can be found out 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 little bit changes according to the purpose of this sub-section III.1.3.1 – the evolution of parameters in BN. 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 mathematics formulas. The [appendix C](#) lists all formulas in this research.

#### 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 sub-section III.1.1.1 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  $\beta(x; a, b)$  or  $beta(x; a, b)$  with variable  $x$  and two parameters  $a, b$  ( $N=a+b$ ) where  $a$  and  $b$  should be integer number greater than 0. Beta density function with two parameters  $a$  and  $b$  (Neapolitan, 2003, p. 300) is defined in formula III.1.12.

$$\beta(x; a, b) = beta(x; a, b) = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} x^{a-1} (1-x)^{b-1}$$

$$(N = a + b \text{ and } 0 \leq x \leq 1)$$

*Formula III.1.12. Beta density function*

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

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

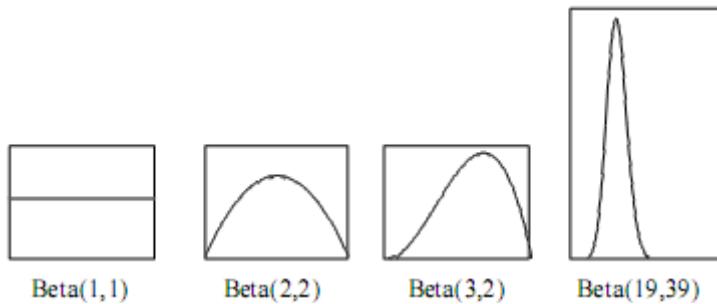
*Formula III.1.13.* Gamma function

It is conventional that  $e^{(\cdot)}$  and  $\exp(\cdot)$  denote exponent function and  $e \approx 2.71828$  is Euler's number. If  $x$  is positive integer, gamma function in formula III.1.13 is equivalent to factorial function,  $\Gamma(x) = (x - 1)!$ . There is an important property of gamma function which is expressed in formula III.1.14 (Neapolitan, 2003, p. 298).

$$\frac{\Gamma(x + 1)}{\Gamma(x)} = x$$

*Formula III.1.14.* Important property of gamma function with regard to factorial function

Figure III.1.10 shows beta density function with various parameters  $a$  and  $b$ .



**Figure III.1.10.** 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. Higher value of “ $a$ ” is, higher ratio of success is, so, the graph leans forward right. Higher value of “ $a+b$ ” is, the more the mass concentrates around  $a/(a+b)$  and the more narrow the graph is.

The integral in interval  $[0, 1]$  of the expression  $x^{a-1}(1-x)^{b-1}$  inside definition of beta function specified by formula III.1.12 is determined by formula III.1.15 as follow:

$$\int_0^1 x^a (1-x)^b dx = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)}$$

*Formula III.1.15.* Integral of product expression  $x^a(1-x)^b$

Proof,

$$\begin{aligned}
 \int_0^1 x^a (1-x)^b dx &= \int_0^1 \frac{\Gamma(a+1+b+1)}{\Gamma(a+1)\Gamma(b+1)} x^a (1-x)^b \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+1+b+1)} dx \\
 &= \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \int_0^1 \beta(x; a+1, b+1) dx \\
 &\quad (\text{by formula III. 1.12}) \\
 &= \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} P(0 \leq x \leq 1) = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \\
 &\quad (\text{due to } P(0 \leq x \leq 1) = 1)
 \end{aligned}$$

The formula III.1.15 is noted as lemma 6.2 in (Neapolitan, 2003, p. 300).

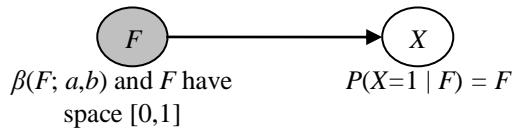
Suppose there is one binary variable  $X$  in network and the probability distribution of  $X$  is considered as relative frequency having values in space  $[0, 1]$  which is the range of variable  $F$ . A dummy variable  $F$  (whose space consists of numbers in  $[0, 1]$ , of course) is added to each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

$$P(X=1|F) = F, \text{ where } F \text{ has beta density function } \beta(F; a, b)$$

*Formula III.1.16.* Conditional probability (relative frequency) of  $X$  as value of dummy variable  $F$

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

Please pay attention to the formula III.1.16,  $P(X=1|F) = F$  implicating that  $F$  represents relative 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. When hypothesis variable  $X$  is attached by dummy variable  $F$  then, variable  $F$ , the probability  $P(X=1|F) = F$ , and beta function  $\beta(F; a, b)$  share the same purpose and all of them represent CPT of  $X$ . Figure III.1.11 shows the simplest augmented BN.



**Figure III.1.11.** 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 III.1.1d’), we have:

$$P(X = 1) = \int_0^1 P(X = 1|F)\beta(F)dF = \int_0^1 F\beta(F)dF = E(F)$$

### III.1. Knowledge sub-model

---

Due to  $F$  is beta function, its expectation  $E(F) = \frac{a}{N}$ , and so we have a very simple but effective formula to compute the probability of  $X$  as follows:

$$P(X = 1) = E(F) = \frac{a}{N}$$

*Formula III.1.17.* Probability of hypothesis  $X$  as expectation of beta variable  $F$

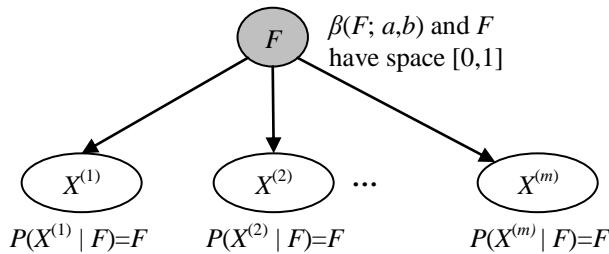
Proof,

$$\begin{aligned} P(X = 1) &= E(F) = \int_0^1 F\beta(F)dF = \int_0^1 F \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} F^{a-1} (1-F)^{b-1} df \\ &= \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \int_0^1 F^a (1-F)^{b-1} = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+1)\Gamma(b)}{\Gamma(N+1)} \\ &\quad \text{(due to formula III.1.15)} \\ &= \frac{\Gamma(N)}{\Gamma(N+1)} \frac{\Gamma(a+1)}{\Gamma(a)} = \frac{a}{N} \\ &\quad \text{(due to formula III.1.14)} \end{aligned}$$

Please pay attention to formula III.1.17, it is the most essential formula used over the whole sub-section III.1.3.1. The formula III.1.17 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  $u^{th}$  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  $\mathcal{D} = (X^{(1)}, X^{(2)}, \dots, X^{(m)})$  which is also called the sample of size  $m$ . Hence,  $\mathcal{D}$  is known as a *sample* or an *evidence vector* and we often implicate  $\mathcal{D}$  as a collection of evidences. Given this sample,  $\beta(F)$  is called the prior density function, and  $P(X^{(u)} = 1) = a/N$  (due to formula III.1.17) is called prior probability of  $X^{(u)}$ . It is necessary to determine the posterior density function  $\beta(F/\mathcal{D})$  and the posterior probability of  $X$ , namely  $P(X/\mathcal{D})$ . The nature of this process is the parameters learning. Note that  $P(X/\mathcal{D})$  can be referred as  $P(X^{(m+1)} / \mathcal{D})$ . Figure III.1.12 depicts this sample  $\mathcal{D} = (X^{(1)}, X^{(2)}, \dots, X^{(m)})$ .



**Figure III.1.12.** The sample  $\mathcal{D} = (X^{(1)}, X^{(2)}, \dots, X^{(m)})$  size of  $m$

We only surveyed in the case of binomial sample, in other words,  $\mathcal{D}$  having binomial distribution is called binomial sample and the network in figure III.1.11 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 III.1.1d'), we have

$$\begin{aligned} E(F^s(1 - F)^t) &= \int_0^1 F^s(1 - F)^t \beta(F) dF \\ &= \int_0^1 F^s(1 - F)^t \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} F^{a-1}(1 - F)^{b-1} dF \end{aligned}$$

(by applying definition of beta function specified in formula III.1.12)

$$\begin{aligned} &= \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \int_0^1 F^{a+s-1}(1 - F)^{b+t-1} dF = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a+b+s+t)} \\ &\quad (\text{due to formula III.1.15}) \\ &= \frac{\Gamma(N)}{\Gamma(N+M)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)} \\ &\quad (\text{due to } N = a + b \text{ and } M = s + t) \end{aligned}$$

In brief, we have formula III.1.18 to determine expectation of  $F^s(1 - F)^t$  as follows:

$$E(F^s(1 - F)^t) = \frac{\Gamma(N)}{\Gamma(N+M)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)}$$

*Formula III.1.18. Expectation of expression  $F^s(1 - F)^t$*

The same proof for formula III.1.18 can be found in (Neapolitan, 2003, p. 306) and formula III.1.18 is called lemma 6.4 in (Neapolitan, 2003, p. 305). The probability of evidences  $P(\mathcal{D})$  equals this expectation  $E(F^s(1 - F)^t)$ , which is interpreted as follows:

$$\begin{aligned} P(\mathcal{D}) &= \int_0^1 P(\mathcal{D}|F) \beta(F) dF = \int_0^1 \left( \prod_{i=1}^m P(X^{(i)}|F) \right) \beta(F) dF \\ &\quad (\text{because evidence } \mathcal{D} \text{ contains independent random variables } X^{(1)}, X^{(2)}, \dots, X^{(m)}) \\ &= \int_0^1 F^s(1 - F)^t \beta(F) dF \\ &\quad \left( \text{due to binomial trials } \prod_{i=1}^m P(X^{(i)}|F) = F^s(1 - F)^t \right) \\ &= E(F^s(1 - F)^t) \end{aligned}$$

In brief, we have formula III.1.19 to determine the probability  $P(\mathcal{D})$  of evidences  $\mathcal{D}$ .

$$P(\mathcal{D}) = E(F^s(1-F)^t) = \frac{\Gamma(N)}{\Gamma(N+M)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)}$$

*Formula III.1.19.* Probability  $P(\mathcal{D})$  of evidences  $\mathcal{D}$

The similar proof for formula III.1.19 can be found out in (Neapolitan, 2003, p. 307) and formula III.1.19 is called corollary 6.2 in (Neapolitan, 2003, p. 307). The probability  $P(\mathcal{D})$  is also called *marginal probability* of evidence sample  $\mathcal{D}$  (see sub-section III.1.1.1).

### Computing posterior density function and posterior probability

Now, we need to compute the posterior density function  $\beta(F|\mathcal{D})$  and the posterior probability  $P(X=1|\mathcal{D})$ . It is essential to determine the probability distribution of  $X$ . The beta density function is updated based on evidences  $\mathcal{D}$  as follows:

$$\begin{aligned} \beta(F|\mathcal{D}) &= \frac{P(\mathcal{D}|F)\beta(F)}{P(\mathcal{D})} \\ &= \frac{F^s(1-F)^t\beta(F)}{E(F^s(1-F)^t)} \\ &\quad \left( \text{due to } P(\mathcal{D}|F) = \prod_{i=1}^m P(X^{(i)}|F) = F^s(1-F)^t \text{ and } P(\mathcal{D}) \right. \\ &\quad \left. = E(F^s(1-F)^t) \text{ specified in formula III.1.19} \right) \\ &= \frac{F^s(1-F)^t \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} F^{a-1}(1-F)^{b-1}}{\frac{\Gamma(N)}{\Gamma(N+M)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)}} \\ &\quad \left( \text{By applying formulas III.1.12 and III.1.18} \right) \\ &= \frac{\Gamma(N+M)}{\Gamma(a+s)\Gamma(b+t)} F^{a+s-1}(1-F)^{b+t-1} = \beta(F; a+s, b+t) \end{aligned}$$

Briefly, the posterior density function is  $\beta(F; a+s, b+t)$  where the prior density function is  $\beta(F; a, b)$ , which is expressed in formula III.1.20.

$$\beta(F|\mathcal{D}) = \beta(F; a+s, b+t)$$

*Formula III.1.20.* Posterior beta density function

The similar proof formula III.1.20 can be found in (Neapolitan, 2003, pp. 306-308) and formula III.1.20 is called corollary 6.3 in (Neapolitan, 2003, p. 308). According to formula III.1.17, the posterior probability of  $X$  is totally determined as below:

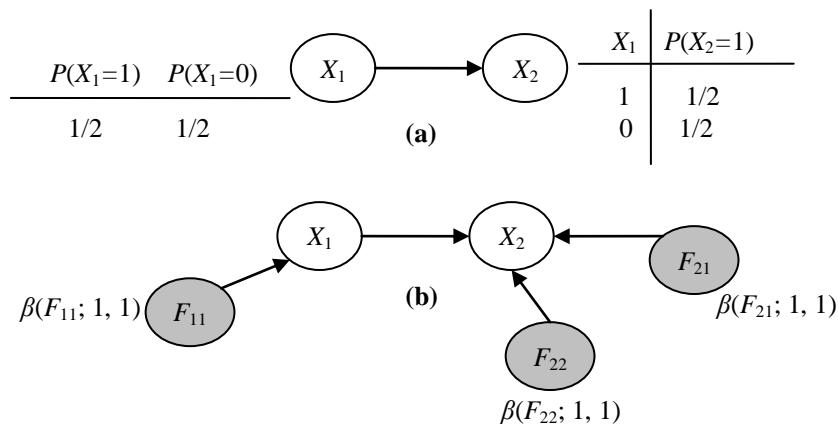
$$P(X = 1|\mathcal{D}) = E(F|\mathcal{D}) = \frac{a + s}{a + s + b + t} = \frac{a + s}{N + M}$$

*Formula III.1.21.* Posterior probability of  $X$

Formula III.1.21 is essentially theorem 6.4 in (Neapolitan, 2003, p. 309). In general, you should merely remember the formulas III.1.12, III.1.17, III.1.20, III.1.21 and the way to recognize prior density function, prior probability of  $X$ , posterior density function, and posterior probability of  $X$ , respectively. Additionally, formula III.1.16 attaching beta density function to CPT, which is the base of these formulas, should be considered. Please pay attention that the prior probability  $P(X = 1) = \frac{a}{N}$  implies that the CPT of  $X$  is represented by beta density function  $\beta(F; a, b)$ . After receiving evidences, the posterior probability is re-calculated,  $P(X = 1|\mathcal{D}) = \frac{a+s}{N+M}$  which implies that the CPT of  $X$  is evolved (learned) and represented by updated beta density function  $\beta(F; a+s, b+t)$ . This is the process of parameter learning or the evolution of Bayesian model aforementioned in [beginning of this section III.1.3.1](#). 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 III.1.13 (Neapolitan, 2003, p. 329):



**Figure III.1.13.** BN (a) and complex augmented BN (b)

In figure III.1.13, 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)  $X_i$ , we add dummy parent nodes to  $X_i$ , obeying two cases below:

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

### III.1. Knowledge sub-model

---

2. If  $X_i$  has a set of  $k_i$  parent nodes and each parent node is binary, we add a set of  $c_i=2k_i$  dummy variables  $\{F_{i1}, F_{i2}, \dots, F_{ic_i}\}$  which, in turn, correspond to instances of parent nodes of  $X_i$ , namely  $\{PA_{i1}, PA_{i2}, PA_{i3}, \dots, PA_{ic_i}\}$  where each  $PA_{ij}$  is an instance of a parent node of  $X_i$  with note that each binary parent node has two instances (0 and 1, for example). For convenience, each  $PA_{ij}$  is called a parent instance of  $X_i$  and we let  $PA_i=\{PA_{i1}, PA_{i2}, PA_{i3}, \dots, PA_{ic_i}\}$  be the vector or collection of parent instances of  $X_i$ . We also let  $F_i=\{F_{i1}, F_{i2}, \dots, F_{ic_i}\}$  be the respective vector or collection of dummy variables  $F_{i1}$  (s) attached to  $X_i$ . It is conventional that each  $X_i$  has  $c_i$  parent instances ( $c_i \geq 0$ ); in other words,  $c_i$  denotes the size of  $PA_i$  and the size of  $F_i$ . For example, in figure III.1.13, 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 III.1.22 for connecting CPT of variable  $X_i$  and beta density function of dummy variable  $F_i$ .

$$P(X_i = 1 | PA_{ij}, F_{i1}, F_{i2}, \dots, F_{ij}, \dots, F_{ic_i}) = P(X_i = 1 | PA_{ij}, F_{ij}) = F_{ij}$$

*Formula III.1.22.* Conditional probability (relative frequency) of  $X_i$  given a parent instance  $PA_{ij}$ , as value of dummy variable in multi-node BN

Formula III.1.22 is an extension of formula III.1.16 in multi-node BN and formula III.1.22 degenerate to formula III.1.16 if  $X_i$  has no parent. Note that the beta density function of  $F_{ij}$  is  $\beta(F_{ij}; a_{ij}, b_{ij})$  and of course, in figure III.1.13, 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  $F_{ij}$  is specified in formula III.1.23 as follows:

$$\beta(F_{ij}) = \beta(F_{ij}; a_{ij}, b_{ij}) = \frac{\Gamma(N_{ij})}{\Gamma(a_{ij})\Gamma(b_{ij})} F_{ij}^{a_{ij}-1} (1 - F_{ij})^{b_{ij}-1}$$

(Where  $N_{ij} = a_{ij} + b_{ij}$ )

*Formula III.1.23.* Beta density function  $\beta(F_{ij})$  corresponding to an instance of a parent of node  $X_i$

Note that formulas III.1.12 and III.1.23 have the same meaning for representing beta function except that formula III.1.23 is used in multi-node BN. Variables  $F_{ij}$  (s) attached to the same  $X_i$  have no parent and are mutually independent, so, it is very easy to compute the joint beta density function  $\beta(F_{i1}, F_{i2}, \dots, F_{ic_i})$  with regard to node  $X_i$  as follows:

$$\beta(F_i) = \beta(F_{i1}, F_{i2}, \dots, F_{ic_i}) = \beta(F_{i1})\beta(F_{i2}) \dots \beta(F_{ic_i}) = \prod_{j=1}^{c_i} \beta(F_{ij})$$

*Formula III.1.24.* Joint beta density function of variable  $X_i$  having  $c_i$  parent instances

Besides the local parameter independence expressed in formula III.1.24, we have global parameter independence if reviewing all variables  $X_i$  (s) with note that all respective  $F_{ij}$  (s) over entire augmented BN are mutually independent. Formula III.1.25 expresses the global parameter independence of all  $F_{ij}$  (s).

$$\begin{aligned}\beta(F_1, F_2, \dots, F_i, \dots, F_n) &= \beta\left(\frac{F_{11}, F_{12}, \dots, F_{1c_1}, F_{21}, F_{22}, \dots, F_{2c_2}, \dots}{F_{i1}, F_{i2}, \dots, F_{ic_i}, \dots, F_{n1}, F_{n2}, \dots, F_{nc_n}}\right) \\ &= \prod_{i=1}^n \beta(F_{i1}, F_{i2}, \dots, F_{ic_i}) = \prod_{i=1}^n \prod_{j=1}^{c_i} \beta(F_{ij})\end{aligned}$$

*Formula III.1.25.* Global joint beta density function of  $n$  independent variable  $X_i$  (s)

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

All variables  $X_i$  and their dummy variables form the complex augmented BN representing the trust BN in figure III.1.13. In the trust BN, the conditional probability of variable  $X_i$  with respect to its parent instance  $PA_{ij}$ , in other words, the  $ij^{th}$  conditional distribution is the expected value of  $F_{ij}$  as below:

$$P(X_i = 1 | PA_{ij}) = E(F_{ij}) = \frac{a_{ij}}{N_{ij}}$$

*Formula III.1.26.* Probability of variable  $X_i$  with respective to its parent instance as expectation of beta variable

The formula III.1.26 is extension of formula III.1.17 when variable  $X_i$  has parent and both of these formulas express prior probability of variable  $X_i$ . Following is proof of formula III.1.26.

$$\begin{aligned}P(X_i = 1 | PA_{ij}) &= \int_0^1 \dots \int_0^1 P(X_i = 1 | PA_{ij}, F_{i1}, \dots, F_{ij}, \dots, F_{ic_i}) \beta(F_{i1}, \dots, F_{ij}, \dots, F_{ic_i}) \\ &\quad dF_{i1} \dots dF_{ij} \dots dF_{ic_i} \\ &= \int_0^1 \dots \int_0^1 P(X_i = 1 | PA_{ij}, F_{i1}, \dots, F_{ij}, \dots, F_{ic_i}) \beta(F_{i1}) \dots \beta(F_{ij}) \dots \beta(F_{ic_i}) \\ &\quad dF_{i1} \dots dF_{ij} \dots dF_{ic_i}\end{aligned}$$

(due to local parameter independence specified in formula III.1.24 when  $F_{ij}$  (s) are mutually independent)

### III.1. Knowledge sub-model

---

$$\begin{aligned}
&= \int_0^1 \dots \int_0^1 F_{ij} \beta(F_{i1}) \dots \beta(F_{ij}) \dots \beta(F_{ic_i}) dF_{i1} \dots dF_{ij} \dots dF_{ic_i} \\
&\quad (\text{due to } P(X_i = 1 | PA_{ij}, F_{i1}, \dots, F_{ij}, \dots, F_{ic_i}) \\
&\quad = F_{ij} \text{ specified in formula III.1.22}) \\
&= \int_0^1 F_{ij} \beta(F_{ij}) dF_{ij} = E(F_{ij}) = \frac{a_{ij}}{N_{ij}}
\end{aligned}$$

The formula [III.1.26](#) noted as theorem 6.7 which is proved by the similar way in (Neapolitan, 2003, pp. 334-335). For illustrating formulas [III.1.22](#) and [III.1.26](#), recall that variables  $F_{ij}$  (s) and their beta density functions  $\beta(F_{ij})$  (s) specify conditional probabilities of  $X_i$  (s) as in figure [III.1.13](#), and so, the CPT (s) in figure [III.1.13](#) is interpreted in detailed as follows:

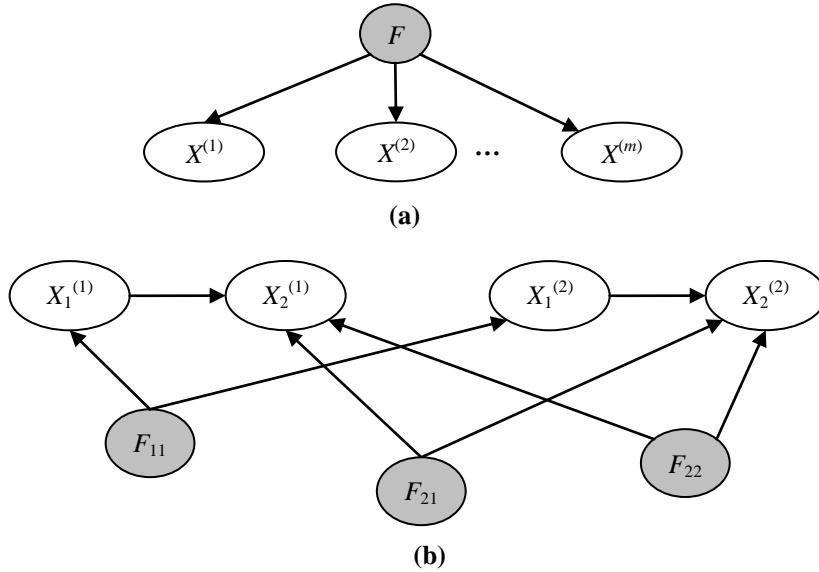
$$\begin{aligned}
P(X_1 = 1 | F_{11}) &= F_{11} \Rightarrow P(X_1 = 1) = E(F_{11}) = \frac{1}{1+1} = \frac{1}{2} \\
P(X_2 = 1 | X_1 = 1, F_{21}) &= F_{21} \Rightarrow P(X_2 = 1 | X_1 = 1) = E(F_{21}) = \frac{1}{1+1} = \frac{1}{2} \\
P(X_2 = 1 | X_1 = 0, F_{22}) &= F_{22} \Rightarrow P(X_2 = 1 | X_1 = 0) = E(F_{22}) = \frac{1}{1+1} = \frac{1}{2}
\end{aligned}$$

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  $X_i$  (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  $u^{th}$  trial which is BN like figure [III.1.13](#) is represented as a random vector  $X^{(u)}$  containing all evidence variables in network. Vector  $X^{(u)}$  is also called the  $u^{th}$  evidence (vector) of entire BN. Suppose  $X^{(u)}$  has  $n$  components or partial evidences  $X_i^{(u)}$  when BN has  $n$  nodes; in figure [III.1.13](#),  $n = 2$ . Note that evidence  $X_i^{(u)}$  is considered as random variable like  $X_i$ .

$$X^{(u)} = \begin{pmatrix} X_1^{(u)} \\ X_2^{(u)} \\ \vdots \\ X_n^{(u)} \end{pmatrix}$$

It is easy to recognize that each component  $X_i^{(u)}$  represents the  $u^{th}$  evidence of node  $X_i$  in the BN. The  $m$  trials constitute the sample of size  $m$  which is the set of random vectors denoted as  $\mathcal{D} = \{X^{(1)}, X^{(2)}, \dots, X^{(m)}\}$ .  $\mathcal{D}$  is also called *evidence matrix* or *evidence sample* or *training data* or *evidences*, in brief. We review only in case of binomial sample, it means that  $\mathcal{D}$  is the binomial BN sample of size  $m$ . For example, this sample corresponding to the network in figure [III.1.13](#) is depicted by figure [III.1.14](#) as below (Neapolitan, 2003, p. 337):



**Figure III.1.14.** Expanded binomial BN sample of size  $m$

After occurring  $m$  trials, the augmented BN was updated and dummy variables' density functions and hypothesis variables' conditional probabilities changed. We need to compute the posterior density function  $\beta(F_{ij}/\mathcal{D})$  of each dummy variable  $F_{ij}$  and the posterior condition probability  $P(X_i=1/PA_{ij}, \mathcal{D})$  of each variable  $X_i$ . Note that evidence vectors  $X^{(u)}$  (s) are mutually independent with all given  $F_{ij}$  (s). It is easy to infer that given fixed  $i$ , all evidences  $X_i^{(u)}$  corresponding to variable  $X_i$  are mutually independent. Based on binomial trials and mentioned mutual independence, we have formula III.1.27 for calculating probability of evidences corresponding to variable  $X_i$  over  $m$  trials as follows:

$$P(X_i^{(1)}, X_i^{(2)}, \dots, X_i^{(M)} | PA_i, F_i) = \prod_{u=1}^m P(X_i^{(u)} | PA_i, F_i) = \prod_{j=1}^{c_i} (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}}$$

*Formula III.1.27. Probability of evidences corresponding to variable  $X_i$*

Where,

- Number  $c_i$  is the number of parent instances of  $X_i$ . In binary case, each  $X_i^{(u)}$ 's parent node has two instances/values, namely, 0 and 1.
- Counter  $s_{ij}$ , respective to  $F_{ij}$ , is the number of all evidences among  $m$  trials such that variable  $X_i = 1$  and  $PA_{ij} = 1$ . Counter  $t_{ij}$ , respective to  $F_{ij}$ , is the number of all evidences among  $m$  trials such that variable  $X_i = 1$  and  $PA_{ij} = 0$ . Note that  $s_{ij}$  and  $t_{ij}$  are often called counters or count numbers.
- $PA_i = \{PA_{i1}, PA_{i2}, PA_{i3}, \dots, PA_{ic_i}\}$  is the vector of parent instances of  $X_i$  and  $F_i = \{F_{i1}, F_{i2}, \dots, F_{ic_i}\}$  is the respective vector of variables  $F_{i1}$  (s) attached to  $X_i$ .

From formula III.1.27, it is easy to compute conditional probability  $P(\mathcal{D}/F_1, F_2, \dots, F_n)$  of evidence sample  $\mathcal{D}$  given  $n$  vectors  $F_i$  (s) with assumption that BN has  $n$  variables  $X_i$  (s) as follows:

### III.1. Knowledge sub-model

---

$$\begin{aligned}
P(\mathcal{D}|F_1, F_2, \dots, F_n) &= P\left(X^{(1)}, X^{(2)}, \dots, X^{(m)}|F_1, F_2, \dots, F_n\right) \\
&= \prod_{u=1}^m P(X^{(u)}|F_1, F_2, \dots, F_n) \\
&\quad (\text{due to evidence vectors } X^{(u)} \text{ (s) are mutually independent}) \\
&= \prod_{u=1}^m \frac{P(X^{(u)}, F_1, F_2, \dots, F_n)}{P(F_1, F_2, \dots, F_n)} \\
&\quad (\text{due to Bayes' rule specified in formula III.1.1a}) \\
&= \prod_{u=1}^m \frac{P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)}, F_1, F_2, \dots, F_n)}{P(F_1, F_2, \dots, F_n)} \\
&= \prod_{u=1}^m \frac{P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)}|F_1, F_2, \dots, F_n)P(F_1, F_2, \dots, F_n)}{P(F_1, F_2, \dots, F_n)} \\
&\quad (\text{applying multiplication rule specified by formula III.1.1c into the numerator}) \\
&= \prod_{u=1}^m P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)}|F_1, F_2, \dots, F_n) \\
&= \prod_{u=1}^m \prod_{i=1}^n P(X_i^{(u)}|PA_i, F_i) \\
&\quad (\text{because } X_i^{(u)} \text{ (s) are mutually independent given } F_i \text{ (s) and each } X_i \text{ depends only on } F_i) \\
&= \prod_{i=1}^n \prod_{u=1}^m P(X_i^{(u)}|PA_i, F_i) = \prod_{i=1}^n \prod_{j=1}^{c_i} (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}} \\
&\quad \left( \text{due to formula III.1.27: } \prod_{u=1}^m P(X_i^{(u)}|PA_i, F_i) = \prod_{j=1}^{c_i} (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}} \right)
\end{aligned}$$

In brief, we have formula III.1.28 for calculating conditional probability  $P(\mathcal{D}|F_1, F_2, \dots, F_n)$  of evidence sample  $\mathcal{D}$  given  $n$  vectors  $F_i$  (s).

$$P(\mathcal{D}|F_1, F_2, \dots, F_n) = \prod_{i=1}^n \prod_{j=1}^{c_i} (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}}$$

*Formula III.1.28. Probability of evidence sample  $\mathcal{D}$  given vectors  $F_i$*

The formula III.1.28 is lemma 6.8 which is proved by similar way in (Neapolitan, 2003, pp. 338-339). It is necessary to calculate the whole probability  $P(\mathcal{D})$  of evidence sample  $\mathcal{D}$ , we have:

$$\begin{aligned}
P(\mathcal{D}) &= P(X^{(1)}, X^{(2)}, \dots, X^{(m)}) = \prod_{u=1}^m P(X^{(u)}) = \prod_{u=1}^m P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)}) \\
&\quad (\text{due evidence vectors } X^{(u)} \text{ (s) are independent})
\end{aligned}$$

$$\begin{aligned}
 &= \prod_{u=1}^m \int_{F_1} \dots \int_{F_n} P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)} | F_1, F_2, \dots, F_n) \beta(F_1, F_2, \dots, F_n) \\
 &\quad dF_1 dF_2 \dots dF_n \\
 &\quad (\text{due to total probability rule in continuous case, please see formula III.1.1d'}) \\
 &= \prod_{u=1}^m \int_{F_1} \dots \int_{F_n} \prod_{i=1}^n P(X_i^{(u)} | PA_i, F_i) \prod_{i=1}^n \beta(F_i) \\
 &\quad dF_1 dF_2 \dots dF_n \\
 &\quad (\text{Because } X_i^{(u)} \text{ (s) are mutually independent given } F_i \text{ (s) and each } X_i \text{ depends only on } F_i. \text{ Moreover, all } F_i \text{ (s) are mutually independent}) \\
 &= \prod_{u=1}^m \int_{F_1} \dots \int_{F_n} \left( \prod_{i=1}^n P(X_i^{(u)} | PA_i, F_i) \beta(F_i) \right) dF_1 dF_2 \dots dF_n \\
 &= \prod_{u=1}^m \prod_{i=1}^n \int_{F_i} P(X_i^{(u)} | PA_i, F_i) \beta(F_i) dF_i \\
 &= \prod_{i=1}^n \prod_{u=1}^m \int_{F_i} P(X_i^{(u)} | PA_i, F_i) \beta(F_i) dF_i \\
 &= \prod_{i=1}^n \prod_{j=1}^{c_i} \int_0^1 (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}} \beta(F_{ij}) dF_{ij} \\
 &= \prod_{i=1}^n \prod_{j=1}^{c_i} E((F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}})
 \end{aligned}$$

(due to binomial trials)

In brief, we have formula III.1.29 for determining the whole probability  $P(\mathcal{D})$  of evidence sample  $\mathcal{D}$  as product of expectations of binomial trials.

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

*Formula III.1.29. Whole probability of evidence sample  $\mathcal{D}$*

Formula III.1.29 can be found out as theorem 6.11 in (Neapolitan, 2003, p. 343). There is the question “how to determine  $E((F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}})$  in formula III.1.29” and so we have formula III.1.30 for calculating this expectation by extending formula III.1.18, as follows:

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

*Formula III.1.30. Expectation of binomial trials*

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

When both condition probability  $P(\mathcal{D}|F_1, F_2, \dots, F_n)$  and whole probability  $P(\mathcal{D})$  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  $\beta(F_{ij}|\mathcal{D})$  and the posterior probability  $P(X_i=1|PA_{ij}, \mathcal{D})$  for each variable  $X_i$  in BN. In fact, we have:

$$\begin{aligned}
 \beta(F_{ij}|\mathcal{D}) &= \frac{P(\mathcal{D}|F_{ij})\beta(F_{ij})}{P(\mathcal{D})} \\
 &= \frac{\left( \int_0^1 \dots \int_0^1 P(E|F_1, F_2, \dots, F_n) \prod_{kl \neq ij} \beta(F_{kl}) dF_{kl} \right) \beta(F_{ij})}{P(\mathcal{D})} \\
 &\quad (\text{due to Bayes' rule specified in formula III.1.1a}) \\
 &= \frac{\left( \int_0^1 \dots \int_0^1 \left( \prod_{u=1}^n \prod_{v=1}^{c_u} (F_{uv})^{s_{uv}} (1 - F_{uv})^{t_{uv}} \right) \left( \prod_{kl \neq ij} \beta(F_{kl}) dF_{kl} \right) \right) \beta(F_{ij})}{P(\mathcal{D})} \\
 &\quad (\text{due to total probability rule in continuous case, formula III.1.1d'}) \\
 &= \frac{\left( F_{ij} \right)^{s_{ij}} \left( 1 - F_{ij} \right)^{t_{ij}} \left( \int_0^1 \dots \int_0^1 \prod_{kl \neq ij} (F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}} \beta(F_{kl}) dF_{kl} \right) \beta(F_{ij})}{P(\mathcal{D})} \\
 &= \frac{\left( F_{ij} \right)^{s_{ij}} \left( 1 - F_{ij} \right)^{t_{ij}} \left( \prod_{kl \neq ij} \int_0^1 (F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}} \beta(F_{kl}) dF_{kl} \right) \beta(F_{ij})}{P(\mathcal{D})} \\
 &= \frac{\left( F_{ij} \right)^{s_{ij}} \left( 1 - F_{ij} \right)^{t_{ij}} \left( \prod_{kl \neq ij} E((F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}}) \right) \beta(F_{ij})}{\prod_{k=1}^n \prod_{l=1}^{c_k} E((F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}})} \\
 &\quad (\text{applying formula III.1.29 into denominator}) \\
 &= \frac{\left( F_{ij} \right)^{s_{ij}} \left( 1 - F_{ij} \right)^{t_{ij}} \left( \prod_{kl \neq ij} E((F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}}) \right) \beta(F_{ij})}{\prod_{kl} E((F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}})} \\
 &= \frac{\left( F_{ij} \right)^{s_{ij}} \left( 1 - F_{ij} \right)^{t_{ij}} \beta(F_{ij})}{E\left(\left( F_{ij} \right)^{s_{ij}} \left( 1 - F_{ij} \right)^{t_{ij}}\right)} \\
 &= \frac{\left( F_{ij} \right)^{s_{ij}} \left( 1 - F_{ij} \right)^{t_{ij}} \frac{\Gamma(N_{ij})}{\Gamma(a_{ij})\Gamma(b_{ij})} \left( F_{ij} \right)^{a_{ij}-1} \left( 1 - F_{ij} \right)^{b_{ij}-1}}{\frac{\Gamma(N_{ij})}{\Gamma(N_{ij} + M_{ij})} \frac{\Gamma(a_{ij} + s_{ij})\Gamma(b_{ij} + t_{ij})}{\Gamma(a_{ij})\Gamma(b_{ij})}}
 \end{aligned}$$

(applying definition of beta density function specified in formula III.1.12 into numerator and applying formula III.1.30 into denominator, note that  $N_{ij} = a_{ij} + b_{ij}$  and  $M_{ij} = s_{ij} + t_{ij}$ )

$$= \frac{\Gamma(N_{ij} + M_{ij})}{\Gamma(a_{ij} + s_{ij})\Gamma(b_{ij} + t_{ij})} \left( F_{ij} \right)^{a_{ij} + s_{ij} - 1} \left( 1 - F_{ij} \right)^{b_{ij} + t_{ij} - 1}$$

$$= \text{beta}(F_{ij}; a_{ij} + s_{ij}, b_{ij} + t_{ij})$$

(due to definition of beta density function specified in formula III.1.12)

In brief, we have formula III.1.31 for calculating posterior beta density function  $\beta(F_{ij}/\mathcal{D})$ .

$$\beta(F_{ij} | \mathcal{D}) = \text{beta}(F_{ij}; a_{ij} + s_{ij}, b_{ij} + t_{ij})$$

*Formula III.1.31.* Posterior beta density function in multi-node BN

Note that formula III.1.31 is an extension of formula III.1.20 in case of multi-node BN. Formula III.1.31 is corollary 6.7 which is proved by similar way in (Neapolitan, 2003, p. 347). Applying formulas III.1.26 and III.1.31, it is easy to specify the posterior probability  $P(X_i=1/PA_{ij}, \mathcal{D})$  of variable  $X_i$  given its parent instance  $PA_{ij}$  as follows:

$$P(X_i = 1 | PA_{ij}, \mathcal{D}) = E(F_{ij} | \mathcal{D}) = \frac{a_{ij} + s_{ij}}{N_{ij} + M_{ij}}$$

*Formula III.1.32.* Posterior probability of variable  $X_i$  given its parent instance  $PA_{ij}$

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

It is easy to recognize that formula III.1.32 is an extension of formula III.1.21 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 such as figure III.1.13 and each dummy node  $F_{ij}$  has a prior beta distribution  $\beta(F_{ij}; a_{ij}, b_{ij})$  and each hypothesis node  $X_i$  has the prior conditional probability  $P(X_i=1/PA_{ij}) = E(F_{ij}) = \frac{a_{ij}}{N_{ij}}$ , the parameter learning process based on a set of evidences is to update the posterior density function  $\beta(F_{ij}/\mathcal{D})$  and the posterior conditional probability  $P(X_i=1/PA_{ij}, \mathcal{D})$ . Indeed, we have  $\beta(F_{ij}/\mathcal{D}) = \text{beta}(F_{ij}; a_{ij}+s_{ij}, b_{ij}+t_{ij})$  and  $P(X_i=1/PA_{ij}, \mathcal{D}) = E(F_{ij}/\mathcal{D}) = \frac{a_{ij}+s_{ij}}{N_{ij}+M_{ij}}$ . In other words, the CPT of each  $X_i$  is evolved based on evidences from  $P(X_i=1/PA_{ij}) = \frac{a_{ij}}{N_{ij}}$  to  $P(X_i=1/PA_{ij}, \mathcal{D}) = \frac{a_{ij}+s_{ij}}{N_{ij}+M_{ij}}$ . This evolution was mentioned for the previous case (see formula III.1.21) in which there are one hypothesis variable  $X$  in BN. Hence, we conclude that learning parameters or the evolution of Bayesian overlay model 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  $\mathcal{D}=\{X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}\}$  owing to network in figure III.1.13. Evidence sample (evidence matrix)  $\mathcal{D}$  is shown in table III.1.5 (Neapolitan, 2003, p. 358).

	X <sub>1</sub>	X <sub>2</sub>
--	----------------	----------------

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

**Table III.1.5.** Evidence sample corresponding to 5 trials (sample of size 5)

In order to interpret evidence sample  $\mathcal{D}$  in table III.1.5, for instance, the first evidence (vector)  $\mathbf{X}^{(1)} = \begin{pmatrix} X_1^{(1)} = 1 \\ X_2^{(1)} = 1 \end{pmatrix}$  implies that variable  $X_2=1$  given  $X_1=1$  occurs in the first trial. We need to compute all posterior density functions  $\beta(F_{11}|\mathcal{D})$ ,  $\beta(F_{21}|\mathcal{D})$ ,  $\beta(F_{22}|\mathcal{D})$  and all posterior conditional probabilities  $P(X_1=1|\mathcal{D})$ ,  $P(X_2=1|X_1=1,\mathcal{D})$ ,  $P(X_2=1|X_1=0,\mathcal{D})$  from prior density functions  $\beta(F_{11}; 1, 1)$ ,  $\beta(F_{21}; 1, 1)$ ,  $\beta(F_{22}; 1, 1)$ . As usual, let counter  $s_{ij}$  ( $t_{ij}$ ) be the number of evidences among 5 trials such that variable  $X_i = 1$  and  $PA_{ij} = 1$  ( $PA_{ij} = 0$ ), the following table III.1.6 shows counters  $s_{ij}$ ,  $t_{ij}$  (s) and posterior density functions calculated based on these counters; please see formula III.1.31 for more details about updating posterior density functions. For instance, the number of rows (evidences) in table III.1.5 such that  $X_2=1$  given  $X_1=1$  is 3, which causes  $s_{21} = 3$  in table III.1.6.

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

**Table III.1.6.** Posterior density functions calculated based on count numbers  $s_{ij}$  and  $t_{ij}$

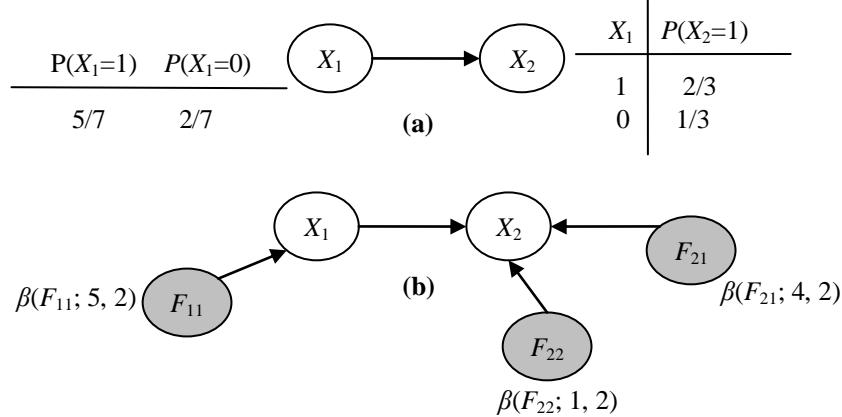
When posterior density functions are determined, it is easy to compute posterior conditional probabilities  $P(X_1=1|\mathcal{D})$ ,  $P(X_2=1|X_1=1,\mathcal{D})$ , and  $P(X_2=1|X_1=0,\mathcal{D})$  as conditional expectations of  $F_{11}$ ,  $F_{21}$ , and  $F_{22}$ , respectively according to formula III.1.32. Table III.1.7 expresses such posterior conditional probabilities as evolutional CPT (s) of  $X_1$  and  $X_2$ .

$P(X_1 = 1 \mathcal{D}) = E(F_{11} \mathcal{D}) = \frac{5}{5+2} = \frac{5}{7}$
$P(X_2 = 1 X_1 = 1, \mathcal{D}) = E(F_{21} \mathcal{D}) = \frac{4}{4+2} = \frac{2}{3}$
$P(X_2 = 1 X_1 = 0, \mathcal{D}) = E(F_{22} \mathcal{D}) = \frac{1}{1+2} = \frac{1}{3}$

**Table III.1.7.** Updated CPT (s) of  $X_1$  and  $X_2$

Note that inverted probabilities in CPT (s) such as  $P(X_1=0|\mathcal{D})$ ,  $P(X_2=0|X_1=1,\mathcal{D})$  and  $P(X_2=0|X_1=0,\mathcal{D})$  are not mentioned because  $X_i$  (s) are binary variables and so,  $P(X_1=0|\mathcal{D}) = 1 - P(X_1=1|\mathcal{D}) = 2/7$ ,  $P(X_2=0|X_1=1,\mathcal{D}) = 1 - P(X_2=1|X_1=1,\mathcal{D}) = 1/3$  and  $P(X_2=0|X_1=0,\mathcal{D}) = 1 - P(X_2=1|X_1=0,\mathcal{D}) = 2/3$ .

Now BN in figure III.1.13 is updated based on evidence sample  $\mathcal{D}$  and it is converted into the evolved BN with full of CPT (s) shown in figure III.1.15 as follows:



**Figure III.1.15.** Updated version of BN (a) and augmented BN (b)

It is easy to perform learning parameters or the evolution of Bayesian by counting numbers  $s_{ij}$  and  $t_{ij}$  among sample according to expectation of beta density function as in formulas III.1.21 and III.1.32 but there is a problem issues when there is data missing in sample. This problem is solved by expectation maximization (EM) algorithm mentioned in next sub-section III.1.3.2.

### III.1.3.2. Learning parameters in case of data missing

In practice there are some evidences in  $\mathcal{D}$  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 III.1.5, in which there is the set of 5 evidences  $\mathcal{D}=\{X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}\}$  along with network in figure III.1.13 but the evidences  $X^{(2)}$  and  $X^{(5)}$  have not data yet. Table III.1.8 shows such missing data (Neapolitan, 2003, p. 359).

	$X_1$	$X_2$
$X^{(1)}$	$X_1^{(1)} = 1$	$X_2^{(1)} = 1$
$X^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = v_1?$

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

**Table III.1.8.** Evidence sample with data missing

As known, count numbers  $s_{21}$ ,  $t_{21}$  and  $s_{22}$ ,  $t_{22}$  can't be computed directly, it means that it is not able to compute directly the posterior density functions  $\beta(F_{11}|\mathcal{D})$ ,  $\beta(F_{21}|\mathcal{D})$ , and  $\beta(F_{22}|\mathcal{D})$ . 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 III.1.9 shows new split evidences for missing data.

	$X_1$	$X_2$	#Occurrences
$\mathbf{X}^{(1)}$	$X_1^{(1)} = 1$	$X_2^{(1)} = 1$	1
$\mathbf{X}^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = 1$	$\#n_{11}$
$\mathbf{X}^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = 0$	$\#n_{10}$
$\mathbf{X}^{(3)}$	$X_1^{(3)} = 1$	$X_2^{(3)} = 1$	1
$\mathbf{X}^{(4)}$	$X_1^{(4)} = 1$	$X_2^{(4)} = 0$	1
$\mathbf{X}^{(5)}$	$X_1^{(5)} = 0$	$X_2^{(5)} = 1$	$\#n_{21}$
$\mathbf{X}^{(5)}$	$X_1^{(5)} = 0$	$X_2^{(5)} = 0$	$\#n_{20}$

**Table III.1.9.** 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 III.1.13.

$$\#n_{11} = P(X_2 = 1|X_1 = 1) = E(F_{21}) = \frac{a_{21}}{a_{21} + b_{21}} = \frac{1}{2}$$

$$\#n_{10} = P(X_2 = 0|X_1 = 1) = 1 - P(X_2 = 1|X_1 = 1) = 1 - \frac{1}{2} = \frac{1}{2}$$

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 III.1.13.

$$\#n_{21} = P(X_2 = 1|X_1 = 0) = E(F_{22}) = \frac{a_{22}}{a_{22} + b_{22}} = \frac{1}{2}$$

$$\#n_{20} = P(X_2 = 0|X_1 = 0) = 1 - P(X_2 = 1|X_1 = 0) = 1 - \frac{1}{2} = \frac{1}{2}$$

When  $\#n_{11}$ ,  $\#n_{10}$ ,  $\#n_{21}$ , and  $\#n_{20}$  are determined, missing data is filled fully and evidence sample  $\mathcal{D}$  is completed as in table III.1.10.

	$X_1$	$X_2$	#Occurrences
$\mathbf{X}^{(1)}$	$X_1^{(1)} = 1$	$X_2^{(1)} = 1$	1
$\mathbf{X}^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = 1$	1/2
$\mathbf{X}^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = 0$	1/2

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

**Table III.1.10.** 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  $\mathcal{D}$  in table III.1.10, we have:

$$\begin{aligned} s_{11} &= 1 + \frac{1}{2} + \frac{1}{2} + 1 + 1 = 4 & t_{11} &= \frac{1}{2} + \frac{1}{2} = 1 \\ s_{21} &= 1 + \frac{1}{2} + 1 = \frac{5}{2} & t_{21} &= \frac{1}{2} + 1 = \frac{3}{2} \\ s_{22} &= \frac{1}{2} & t_{22} &= \frac{1}{2} \end{aligned}$$

**Table III.1.11.** 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  $\beta(F_{11}|\mathcal{D})$ ,  $\beta(F_{21}|\mathcal{D})$ , and  $\beta(F_{22}|\mathcal{D})$ , which leads to update posterior probabilities  $P(X_1=1|\mathcal{D})$ ,  $P(X_2=1|X_1=1,\mathcal{D})$ , and  $P(X_2=1|X_1=0,\mathcal{D})$ , based on current counters  $s_{11}, t_{11}, s_{21}, t_{21}, s_{22}$ , and  $t_{22}$  from complete evidence sample  $\mathcal{D}$  (table III.1.11). Table III.1.12 shows results of M-step which are posterior density functions  $\beta(F_{11}|\mathcal{D})$ ,  $\beta(F_{21}|\mathcal{D})$ , and  $\beta(F_{22}|\mathcal{D})$  along with posterior probabilities (updated CPT) such as  $P(X_1=1|\mathcal{D})$ ,  $P(X_2=1|X_1=1,\mathcal{D})$ , and  $P(X_2=1|X_1=0,\mathcal{D})$ . 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 III.1.13) are kept intact.

$$\begin{aligned} \beta(F_{11}|\mathcal{D}) &= \beta(F_{11}; a_{11} + s_{11}, b_{11} + t_{11}) = \beta(F_{11}; 1 + 4, 1 + 1) = \beta(F_{11}; 5, 2) \\ \beta(F_{21}|\mathcal{D}) &= \beta(F_{21}; a_{21} + s_{21}, b_{21} + t_{21}) = \beta\left(F_{21}; 1 + \frac{5}{2}, 1 + \frac{3}{2}\right) \\ &= \beta\left(F_{21}; \frac{7}{2}, \frac{5}{2}\right) \\ \beta(F_{22}|\mathcal{D}) &= \beta(F_{22}; a_{22} + s_{22}, b_{22} + t_{22}) = \beta\left(F_{22}; 1 + \frac{1}{2}, 1 + \frac{1}{2}\right) \\ &= \beta\left(F_{22}; \frac{3}{2}, \frac{3}{2}\right) \end{aligned}$$

$$\begin{aligned}
 P(X_1 = 1 | \mathcal{D}) &= E(F_{11} | \mathcal{D}) = \frac{5}{5+2} = \frac{5}{7} \\
 P(X_2 = 1 | X_1 = 1, \mathcal{D}) &= E(F_{21} | \mathcal{D}) = \frac{7/2}{7/2 + 5/2} = \frac{7}{12} \\
 P(X_2 = 1 | X_1 = 0, \mathcal{D}) &= E(F_{22} | \mathcal{D}) = \frac{3/2}{3/2 + 3/2} = \frac{1}{2}
 \end{aligned}$$

**Table III.1.12.** Posterior density functions and posterior probabilities are updated in M-step of EM algorithm

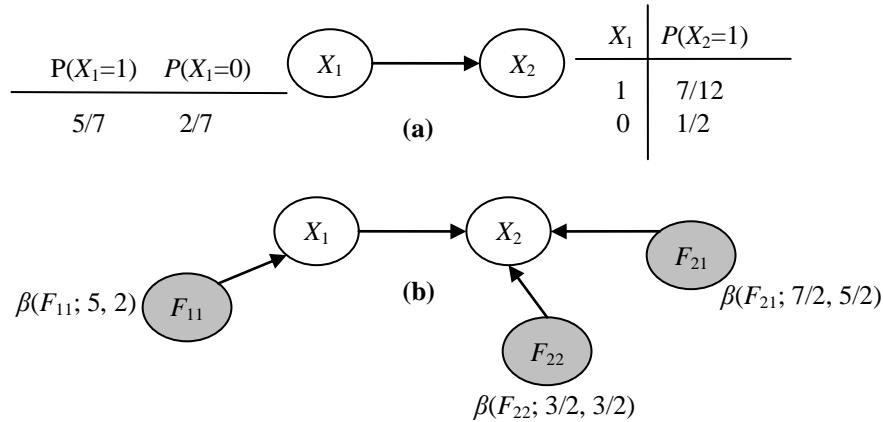
If there are more evidences, the process of such two steps (E-step and M-step) repeated more and more brings out the EM algorithm. In general, EM algorithm is the iterative algorithm having many iterations and each iteration has these steps. Given the  $k^{th}$  iteration in EM algorithm whose two steps such as E-step and M-step are summarized as follows:

1. *E-step.* Missing values are estimated based on expectations of  $F_{ij}$  with regard to previous ( $k-1^{th}$ ) parameters  $a_{ij}$  and  $b_{ij}$ . Current ( $k^{th}$ ) counters  $s_{ij}$  and  $t_{ij}$  are calculated with estimated values (of such missing values).
2. *M-step.* Posterior density functions and posterior probabilities (CPT) are updated based on current ( $k^{th}$ ) counters  $s_{ij}$  and  $t_{ij}$ . Of course,  $a_{ij}$  and  $b_{ij}$  are updated because they are parameters of (beta) density functions. Terminating algorithm if the stop condition (for example, the number of iterations approaches  $k$  times or there is no missing values) becomes true, otherwise, reiterating step 1.

After  $k^{th}$  iteration, the limit

$$\lim_{k \rightarrow +\infty} E(F_{ij} | \mathcal{D})^{(k)} = \lim_{k \rightarrow +\infty} \frac{a_{ij}^{(k)} + s_{ij}^{(k)}}{a_{ij}^{(k)} + s_{ij}^{(k)} + b_{ij}^{(k)} + t_{ij}^{(k)}}$$

will approach a certain limit. Note, the upper script  $(k)$  denotes the  $k^{th}$  iteration. Don't worry about the case of infinite iterations, we will obtain optimal probability  $P(X_i=1/PA_i, \mathcal{D}) = \lim_{k \rightarrow +\infty} E(F_{ij} | \mathcal{D})^{(k)}$  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). After applying EM algorithm, the Bayesian overlay model in figure III.1.13 is converted into the evolutional version specified in figure III.1.16.



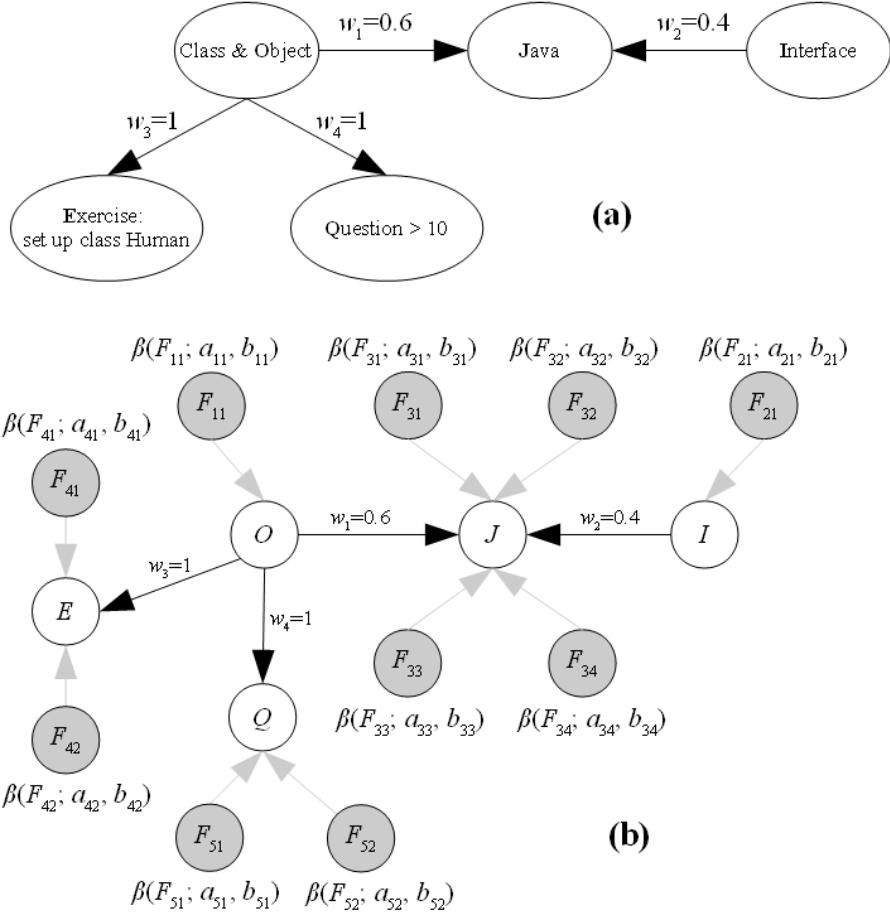
**Figure III.1.16.** Updated version of BN (a) and augmented BN (b) in case of data missing

In general parameter learning or evolution of Bayesian overlay model is described thoroughly in this sub-section III.1.3.2 and previous sub-section III.1.3.1. The next sub-section III.1.3.3 is a full example illustrating main aspects of parameter learning.

### III.1.3.3. An example of learning parameters

Suppose a short Java course (see sub-section III.1.1.2) 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 two evidence variables: “*Questions > 10*” and “*Exercise: set up class Human*”. That learner asks more than 10 questions is to tell how much her/his amount of knowledge. Like that, 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 III.1.3.2 and III.1.3.1. Figure III.1.17 depicts network structure as weighted graph (a) and augmented BN (b) of Java course.

### III.1. Knowledge sub-model



**Figure III.1.17.** 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. Nodes  $E, Q$  denote evidence variables “Exercise: set up class Human” and “Questions > 10” respectively. In this example, node  $J$  has two parents:  $O$  and  $I$  which in turn are corresponding to two weights of aggregation relationship:  $w_1=0.6, w_2=0.4$ . Similarly, the weights of two diagnostic relationships between  $E$  and  $O$ , between  $Q$  and  $O$  are  $w_3=0.8$  and  $w_4=0.2$ . Prior conditional probabilities (CPT (s)) of variables  $J, O, I, E$ , and  $Q$  are specified based on these weights according to theorem of SIGMA-gate inference specified in formula III.1.11 with note that  $\{O, I\}$  is complete set of mutually exclusive variables and so three nodes  $J, O$  and  $I$  construct a sigma graph (see sub-section III.1.1.3). For instance, the prior conditional probability of  $J=1$  given  $O=1$ , and  $I=1$  is:

$$P(J = 1|O = 1, I = 1) = 1 * w_1 + 1 * w_2 = 1 * 0.6 + 1 * 0.4 = 1$$

In similar, it is easy to determine all CPT (s) of variables  $J, O, I, E$  and  $Q$  as shown in table III.1.13.

Real Variable	Dummy Variable	Density Function	Prior Probability
$O$	$F_{11}$	$\beta(F_{11}; a_{11}, b_{11})$	$P(O=1) = 1 * 0.5 = 0.5 = a_{11}/(a_{11}+b_{11})$ $P(O=0) = 1 - P(O=1) = 0.5$
$I$	$F_{21}$	$\beta(F_{21}; a_{21}, b_{21})$	$P(I=1) = a_{21}/(a_{21}+b_{21}) = 1 * 0.5 = 0.5$

			$P(I=0) = 1 - P(I=1) = 0.5$
$J$	$F_{31}$	$\beta(F_{31}; a_{31}, b_{31})$	$P(J=1 O=1, I=1) = 1 * 0.6 + 1 * 0.4 = 1 = a_{31}/(a_{31}+b_{31})$ $P(J=0 O=1, I=1) = 1 - P(J=1 O=1, I=1) = 0$
$J$	$F_{32}$	$\beta(F_{32}; a_{32}, b_{32})$	$P(J=1 O=1, I=0) = 1 * 0.6 + 0 * 0.4 = 0.6 = a_{32}/(a_{32}+b_{32})$ $P(J=0 O=1, I=0) = 1 - P(J=1 O=1, I=0) = 0.4$
$J$	$F_{33}$	$\beta(F_{33}; a_{33}, b_{33})$	$P(J=1 O=0, I=1) = 0 * 0.6 + 1 * 0.4 = 0.4 = a_{33}/(a_{33}+b_{33})$ $P(J=0 O=0, I=1) = 1 - P(J=1 O=0, I=1) = 0.6$
$J$	$F_{34}$	$\beta(F_{34}; a_{34}, b_{34})$	$P(J=1 O=0, I=0) = 0 * 0.6 + 0 * 0.4 = 0 = a_{34}/(a_{34}+b_{34})$ $P(J=0 O=0, I=0) = 1 - P(J=1 O=0, I=0) = 1$
$E$	$F_{41}$	$\beta(F_{41}; a_{41}, b_{41})$	$P(E=1 O=1) = 1 * 1 = 1 = a_{41}/(a_{41}+b_{41})$ $P(E=0 O=1) = 1 - P(E=1 O=1) = 0$
$E$	$F_{42}$	$\beta(F_{42}; a_{42}, b_{42})$	$P(E=1 O=0) = 0 * 1 = 0 = a_{42}/(a_{42}+b_{42})$ $P(E=0 O=0) = 1 - P(E=1 O=0) = 1$
$Q$	$F_{51}$	$\beta(F_{51}; a_{51}, b_{51})$	$P(Q=1 O=1) = 1 * 1 = 1 = a_{51}/(a_{51}+b_{51})$ $P(Q=0 O=1) = 1 - P(Q=1 O=1) = 0$
$Q$	$F_{52}$	$\beta(F_{52}; a_{52}, b_{52})$	$P(Q=1 O=0) = 0 * 1 = 0 = a_{52}/(a_{52}+b_{52})$ $P(Q=0 O=0) = 1 - P(Q=1 O=0) = 1$

**Table III.1.13.** All variables and their density functions, prior probabilities

That  $P(O=1)$  equals 0.5 and  $P(I=1)$  equals 0.5 is due to uniform distribution. Now it is necessary to determine prior beta density functions  $\beta(F_{ij}; a_{ij}, b_{ij})$ , which leads to specify parameters  $a_{ij}$  and  $b_{ij}$ . In fact, it is very easy to specify  $a_{ij}$  and  $b_{ij}$  (s) by taking advantages of theorem of “equivalent sample size” when prior conditional probabilities (CPT (s)) are known as in table III.1.13. 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  $\beta(F_{ij}; a_{ij}, b_{ij})$ , for all  $i$  and  $j$ , if there exists the number  $N$  such that satisfying formula III.1.33 then, the augmented BN is called to have *equivalent sample size N*.

$$N_{ij} = a_{ij} + b_{ij} = P(PA_{ij}) * N \quad (\forall i, j)$$

*Formula III.1.33.* Definition of equivalent sample size  $N$

Where  $P(PA_{ij})$  is probability of the  $j^{th}$  parent instance of an  $X_i$  and it is conventional that if  $X_i$  has no parent then,  $P(PA_{i1})=1$ . Formula III.1.33 is noted as equation 6.13 in (Neapolitan, 2003, p. 351). For example, reviewing BN in figure III.1.13, given  $\beta(F_{11}; 2, 2), \beta(F_{21}; 1, 1), \beta(F_{22}; 1, 1)$ , we have:

$$\begin{aligned} 4 &= a_{11} + b_{11} = 1 * 4 = 4 \quad (P(PA_{11})=1 \text{ because } X_1 \text{ has no parent}) \\ 2 &= a_{21} + b_{21} = P(X_1=1) * 4 = 1/2 * 4 = 2 \\ 2 &= a_{22} + b_{22} = P(X_1=0) * 4 = 1/2 * 4 = 2 \end{aligned}$$

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  $a_{ij}$  and  $b_{ij}$  (s) are satisfied III.1.34 then, the augmented BN has equivalent sample size  $N$ .

$$\begin{aligned} a_{ij} &= P(X_i = 1 | PA_{ij}) * P(PA_{ij}) * N \\ b_{ij} &= P(X_i = 0 | PA_{ij}) * P(PA_{ij}) * N \end{aligned}$$

*Formula III.1.34.* Theorem of equivalent sample size  $N$

### III.1. Knowledge sub-model

---

It is easy to prove this theorem, we have:

For all  $i$  and  $j$ ,  $N_{ij} = a_{ij} + b_{ij}$

$$\begin{aligned}
 &= P(X_i = 1 | PA_{ij}) * P(PA_{ij}) * N + P(X_i = 0 | PA_{ij}) * P(PA_{ij}) \\
 &\quad * N = P(PA_{ij}) * N * (P(X_i = 1 | PA_{ij}) + P(X_i = 0 | PA_{ij})) \\
 &= P(PA_{ij}) * N * (P(X_i = 1 | PA_{ij}) + 1 - P(X_i = 1 | PA_{ij})) \\
 &= P(PA_{ij}) * N
 \end{aligned}$$

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.

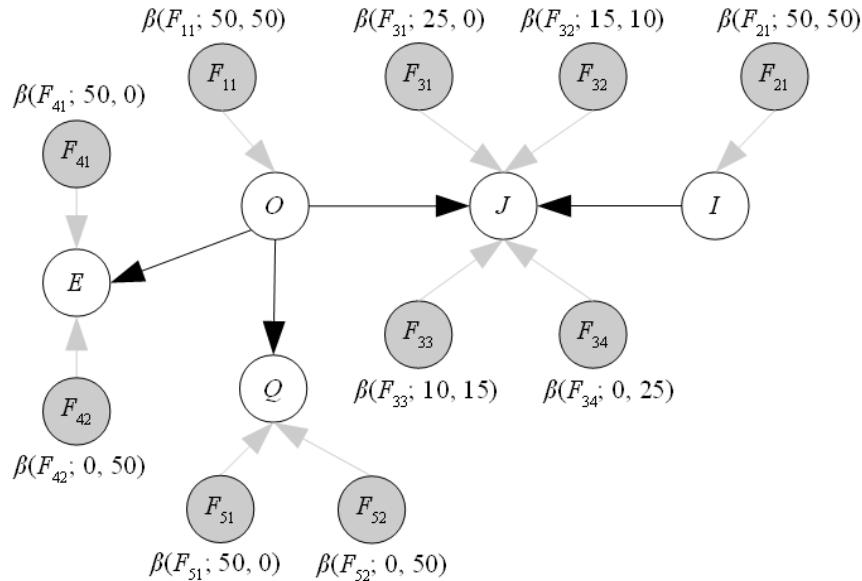
Going back Java course example specified in figure III.1.17, we suppose that the augmented BN has equivalent sample size  $N = 100$ . Applying the theorem of equivalent sample size specified in formula III.1.34, all prior parameters  $a_{ij}$  and  $b_{ij}$  (s) are calculated as in table III.1.14 as follows:

<b>Density Functions</b>	<b>Parameters</b>
$\beta(F_{11}; a_{11}, b_{11})$	$a_{11} = P(O=1)*1*10 = 0.5*100 = \mathbf{50}$ $b_{11} = P(O=0)*1*10 = 0.5*100 = \mathbf{50}$
$\beta(F_{21}; a_{21}, b_{21})$	$a_{21} = P(I=1)*1*10 = 0.5*100 = \mathbf{50}$ $b_{21} = P(I=0)*1*10 = 0.5*100 = \mathbf{50}$
$\beta(F_{31}; a_{31}, b_{31})$	$a_{31} = P(J=1 O=1, I=1)*P(O=1, I=1)*100$ $= P(J=1 O=1, I=1)*P(O=1)*P(I=1)*100$ $= 1*0.5*0.5*100 = \mathbf{25}$ $b_{31} = P(J=0 O=1, I=1)*P(O=1, I=1)*100$ $= P(J=0 O=1, I=1)*P(O=1)*P(I=1)*100$ $= 0*0.5*0.5*100 = \mathbf{0}$
$\beta(F_{32}; a_{32}, b_{32})$	$a_{32} = P(J=1 O=1, I=0)*P(O=1, I=0)*100$ $= P(J=1 O=1, I=0)*P(O=1)*P(I=0)*100$ $= 0.6*0.5*0.5*100 = \mathbf{15}$ $b_{32} = P(J=0 O=1, I=0)*P(O=1, I=0)*100$ $= P(J=0 O=1, I=0)*P(O=1)*P(I=0)*100$ $= 0.4*0.5*0.5*100 = \mathbf{10}$
$\beta(F_{33}; a_{33}, b_{33})$	$a_{33} = P(J=1 O=0, I=1)*P(O=0, I=1)*100$ $= P(J=1 O=0, I=1)*P(O=0)*P(I=1)*100$ $= 0.4*0.5*0.5*100 = \mathbf{10}$ $b_{33} = P(J=0 O=0, I=1)*P(O=0, I=1)*100$ $= P(J=0 O=0, I=1)*P(O=0)*P(I=1)*100$ $= 0.6*0.5*0.5*100 = \mathbf{15}$
$\beta(F_{34}; a_{34}, b_{34})$	$a_{34} = P(J=1 O=0, I=0)*P(O=0, I=0)*100$ $= P(J=1 O=0, I=0)*P(O=0)*P(I=0)*100$ $= 0*0.5*0.5*100 = \mathbf{0}$ $b_{34} = P(J=0 O=0, I=0)*P(O=0, I=0)*100$ $= P(J=0 O=0, I=0)*P(O=0)*P(I=0)*100$

	$= 1 * 0.5 * 0.5 * 100 = \mathbf{25}$
$\beta(F_{41}; a_{41}, b_{41})$	$a_{41} = P(E=1 O=1)*P(O=1)*100$ $= 1 * 0.5 * 100 = \mathbf{50}$ $b_{41} = P(E=0 O=1)*P(O=1)*100$ $= 0 * 0.5 * 100 = \mathbf{0}$
$\beta(F_{42}; a_{42}, b_{42})$	$a_{42} = P(E=1 O=0)*P(O=0)*100$ $= 0 * 0.5 * 100 = \mathbf{0}$ $b_{42} = P(E=0 O=0)*P(O=0)*100$ $= 1 * 0.5 * 100 = \mathbf{50}$
$\beta(F_{51}; a_{51}, b_{51})$	$a_{51} = P(Q=1 O=1)*P(O=1)*100$ $= 1 * 0.5 * 100 = \mathbf{50}$ $b_{51} = P(Q=0 O=1)*P(O=1)*100$ $= 0 * 0.5 * 100 = \mathbf{0}$
$\beta(F_{52}; a_{52}, b_{52})$	$a_{52} = P(Q=1 O=0)*P(O=0)*100$ $= 0 * 0.5 * 100 = \mathbf{0}$ $b_{52} = P(Q=0 O=0)*P(O=0)*100$ $= 1 * 0.5 * 100 = \mathbf{50}$

**Table III.1.14.** All parameters of prior density functions

Note that prior conditional probabilities  $P(X_i/PA_{ij})$  are shown in table III.1.13. The augmented BN with parameters in full is shown in figure III.1.18.



**Figure III.1.18.** Augmented BN with initial parameters in full

When prior CPT (s) (see table III.1.13) and prior beta density functions (see table III.1.14) are specified, the example Java course in this sub-section III.1.3.3 illustrates the evolution of CPT (s) which is essentially updating prior parameters  $a_{ij}$  and  $b_{ij}$  (s) based on evidences as aforementioned in previous sub-sections III.1.3.1 and III.1.3.2. Suppose we have 2 evidences  $X^{(1)}=(E^{(1)}=1, Q^{(1)}=1)$  and  $X^{(2)}=(E^{(2)}=0, Q^{(2)}=0)$ .

### III.1. Knowledge sub-model

---

- The first observation: User does well the exercise “*Set up class Human*” and asks more than 10 questions in course, corresponding to evidence  $X^{(1)}=(E^{(1)}=1, Q^{(1)}=1)$ .
- The second observation: User does not well the exercise “*Set up class Human*” and asks less than 10 questions at another time, corresponding to evidence  $X^{(2)}=(E^{(2)}=0, Q^{(2)}=0)$ .

Because it lacks information about other concepts such as  $O, J, I$ , this is case of data missing aforementioned in previous sub-section III.1.3.2. Table III.1.15 shows incomplete sample  $\mathcal{D}$  with two evidences  $X^{(1)}=(E^{(1)}=1, Q^{(1)}=1)$  and  $X^{(2)}=(E^{(2)}=0, Q^{(2)}=0)$ , in which question mark (?) denotes missing values of variables  $O, I$ , and  $J$ . Algorithm EM mentioned in previous sub-section III.1.3.2 will be used to estimate these missing values.

	$O$	$I$	$J$	$E$	$Q$
$X^{(1)}$	?	?	?	1	1
$X^{(2)}$	?	?	?	0	0

**Table III.1.15.** Incomplete sample with two evidences ( $E=1, Q=1$ ) and ( $E=0, Q=0$ )

Note that missing values (?) are binary and so evidences  $X^{(1)}$  and  $X^{(2)}$  are split into  $X'^{(1)}$  (s) and  $X'^{(2)}$  (s) according two possible values 0 and 1 of missing values (?). Table III.1.16 shows new split evidences for missing values.

	$O$	$I$	$J$	$E$	$Q$	#Occurrences
$X'^{(1)}$	1	1	1	1	1	# $n_{11}$
$X'^{(1)}$	1	1	0	1	1	# $n_{12}$
$X'^{(1)}$	1	0	1	1	1	# $n_{13}$
$X'^{(1)}$	1	0	0	1	1	# $n_{14}$
$X'^{(1)}$	0	1	1	1	1	# $n_{15}$
$X'^{(1)}$	0	1	0	1	1	# $n_{16}$
$X'^{(1)}$	0	0	1	1	1	# $n_{17}$
$X'^{(1)}$	0	0	0	1	1	# $n_{18}$
$X'^{(2)}$	1	1	1	1	0	# $n_{21}$
$X'^{(2)}$	1	1	0	1	0	# $n_{22}$
$X'^{(2)}$	1	0	1	1	0	# $n_{23}$
$X'^{(2)}$	1	0	0	1	0	# $n_{24}$
$X'^{(2)}$	0	1	1	1	0	# $n_{25}$
$X'^{(2)}$	0	1	0	1	0	# $n_{26}$
$X'^{(2)}$	0	0	1	1	0	# $n_{27}$
$X'^{(2)}$	0	0	0	1	0	# $n_{28}$

**Table III.1.16.** New split evidences for missing values of  $O, I$ , and  $J$

Where  $\#n_{ij}$  are numbers of possible combinations (occurrences) of binary variables  $O, I$ , and  $J$ ; please see tables III.1.9 and III.1.10 for knowing more about  $\#n_{ij}$ .

It is required to estimate  $\#n_{ij}$ , which is the most important task in E-step of EM algorithm. For instance, the  $\#n_{11}$  is estimated by the probability of  $O=1$ ,  $I=1$ , and  $J=1$  given  $E=1$  and  $Q=1$ . We have:

$$\begin{aligned}
 & P(E = 1, Q = 1) \\
 &= P(O = 1, I = 1, J = 1, E = 1, Q = 1) \\
 &\quad + P(O = 1, I = 1, J = 0, E = 1, Q = 1) \\
 &\quad + P(O = 1, I = 0, J = 1, E = 1, Q = 1) \\
 &\quad + P(O = 1, I = 0, J = 0, E = 1, Q = 1) \\
 &\quad + P(O = 0, I = 1, J = 1, E = 1, Q = 1) \\
 &\quad + P(O = 0, I = 1, J = 0, E = 1, Q = 1) \\
 &\quad + P(O = 0, I = 0, J = 1, E = 1, Q = 1) \\
 &\quad + P(O = 0, I = 0, J = 0, E = 1, Q = 1) \\
 &= P(O = 1)P(I = 1)P(J = 1|O = 1, I = 1)P(E = 1|O = 1)P(Q = 1|O = 1) \\
 &\quad + P(O = 1)P(I = 1)P(J = 0|O = 1, I = 1)P(E = 1|O = 1)P(Q = 1|O = 1) \\
 &\quad + P(O = 1)P(I = 0)P(J = 1|O = 1, I = 0)P(E = 1|O = 1)P(Q = 1|O = 1) \\
 &\quad + P(O = 1)P(I = 0)P(J = 0|O = 1, I = 0)P(E = 1|O = 1)P(Q = 1|O = 1) \\
 &\quad + P(O = 0)P(I = 1)P(J = 1|O = 0, I = 1)P(E = 1|O = 0)P(Q = 1|O = 0) \\
 &\quad + P(O = 0)P(I = 1)P(J = 0|O = 0, I = 1)P(E = 1|O = 0)P(Q = 1|O = 0) \\
 &\quad + P(O = 0)P(I = 0)P(J = 1|O = 0, I = 0)P(E = 1|O = 0)P(Q = 1|O = 0) \\
 &\quad + P(O = 0)P(I = 0)P(J = 0|O = 0, I = 0)P(E = 1|O = 0)P(Q = 1|O = 0) \\
 &\quad \text{(by applying formula III.1.2' into joint probability distribution } P(O, I, J, E, Q)) \\
 &= 0.5 * 0.5 * 1 * 1 * 1 \\
 &\quad + 0.5 * 0.5 * 0 * 1 * 1 \\
 &\quad + 0.5 * 0.5 * 0.6 * 1 * 1 \\
 &\quad + 0.5 * 0.5 * 0.4 * 1 * 1 \\
 &\quad + 0.5 * 0.5 * 0.4 * 0 * 0 \\
 &\quad + 0.5 * 0.5 * 0.6 * 0 * 0 \\
 &\quad + 0.5 * 0.5 * 0 * 0 * 0 \\
 &\quad + 0.5 * 0.5 * 1 * 0 * 0 \\
 &\quad \text{(prior probabilities } P(\cdot) \text{ are specified in table III.1.13)} \\
 &= 0.25 + 0 + 0.15 + 0 + 0 + 0 + 0 = 0.5
 \end{aligned}$$

$$\begin{aligned}
 & P(E = 0, Q = 0) \\
 &= P(O = 1, I = 1, J = 1, E = 1, Q = 0) \\
 &\quad + P(O = 1, I = 1, J = 0, E = 1, Q = 0) \\
 &\quad + P(O = 1, I = 0, J = 1, E = 1, Q = 0) \\
 &\quad + P(O = 1, I = 0, J = 0, E = 1, Q = 0) \\
 &\quad + P(O = 0, I = 1, J = 1, E = 1, Q = 0) \\
 &\quad + P(O = 0, I = 1, J = 0, E = 1, Q = 0) \\
 &\quad + P(O = 0, I = 0, J = 1, E = 1, Q = 0) \\
 &\quad + P(O = 0, I = 0, J = 0, E = 1, Q = 0) \\
 &= P(O = 1)P(I = 1)P(J = 1|O = 1, I = 1)P(E = 0|O = 1)P(Q = 0|O = 1) \\
 &\quad + P(O = 1)P(I = 1)P(J = 0|O = 1, I = 1)P(E = 0|O = 1)P(Q = 0|O = 1) \\
 &\quad + P(O = 1)P(I = 0)P(J = 1|O = 1, I = 0)P(E = 0|O = 1)P(Q = 0|O = 1) \\
 &\quad + P(O = 1)P(I = 0)P(J = 0|O = 1, I = 0)P(E = 0|O = 1)P(Q = 0|O = 1) \\
 &\quad + P(O = 0)P(I = 1)P(J = 1|O = 0, I = 1)P(E = 0|O = 0)P(Q = 0|O = 0) \\
 &\quad + P(O = 0)P(I = 1)P(J = 0|O = 0, I = 1)P(E = 0|O = 0)P(Q = 0|O = 0) \\
 &\quad + P(O = 0)P(I = 0)P(J = 1|O = 0, I = 1)P(E = 0|O = 0)P(Q = 0|O = 0) \\
 &\quad + P(O = 0)P(I = 0)P(J = 0|O = 0, I = 1)P(E = 0|O = 0)P(Q = 0|O = 0)
 \end{aligned}$$

### III.1. Knowledge sub-model

---

$$\begin{aligned}
& + P(O = 0)P(I = 0)P(J = 0|O = 0, I = 0)P(E = 0|O = 0)P(Q = 0|O = 0) \\
& \quad (\text{by applying formula III.1.2' into joint probability distribution } P(O, I, J, E, Q)) \\
& = 0.5 * 0.5 * 1 * 0 * 0 \\
& + 0.5 * 0.5 * 0 * 0 * 0 \\
& + 0.5 * 0.5 * 0.6 * 0 * 0 \\
& + 0.5 * 0.5 * 0.4 * 0 * 0 \\
& + 0.5 * 0.5 * 0.4 * 1 * 1 \\
& + 0.5 * 0.5 * 0.6 * 1 * 1 \\
& + 0.5 * 0.5 * 0 * 1 * 1 \\
& + 0.5 * 0.5 * 1 * 1 * 1 \\
& \quad (\text{prior probabilities } P(\cdot) \text{ are specified in table III.1.13}) \\
& = 0 + 0 + 0 + 0 + 0.1 + 0.15 + 0 + 0.25 = 0.5
\end{aligned}$$

$$\begin{aligned}
\#n_{11} &= P(O = 1, I = 1, J = 1|E = 1, Q = 1) \\
&= \frac{P(O = 1, I = 1, J = 1, E = 1, Q = 1)}{P(E = 1, Q = 1)} \\
&\quad (\text{due to Bayes' rule specified in formula III.1.1a}) \\
&= \frac{P(O = 1)P(I = 1)P(J = 1|O = 1, I = 1)P(E = 1|O = 1)P(Q = 1|O = 1)}{P(E = 1, Q = 1)} \\
&\quad (\text{by applying formula III.1.2' into joint probability distribution } P(O, I, J, E, Q)) \\
&= \frac{0.25}{0.5} = 0.5
\end{aligned}$$

Similarly, we have:

$$\begin{aligned}
\#n_{12} &= \frac{P(O = 1, I = 1, J = 0, E = 1, Q = 1)}{P(E = 1, Q = 1)} = \frac{0}{0.5} = 0 \\
\#n_{13} &= \frac{P(O = 1, I = 0, J = 1, E = 1, Q = 1)}{P(E = 1, Q = 1)} = \frac{0.15}{0.5} = 0.3 \\
\#n_{14} &= \frac{P(O = 1, I = 0, J = 0, E = 1, Q = 1)}{P(E = 1, Q = 1)} = \frac{0.1}{0.5} = 0.2 \\
\#n_{15} &= \frac{P(O = 0, I = 1, J = 1, E = 1, Q = 1)}{P(E = 1, Q = 1)} = \frac{0}{0.5} = 0 \\
\#n_{16} &= \frac{P(O = 0, I = 1, J = 0, E = 1, Q = 1)}{P(E = 1, Q = 1)} = \frac{0}{0.5} = 0 \\
\#n_{17} &= \frac{P(O = 0, I = 0, J = 1, E = 1, Q = 1)}{P(E = 1, Q = 1)} = \frac{0}{0.5} = 0 \\
\#n_{18} &= \frac{P(O = 0, I = 0, J = 0, E = 1, Q = 1)}{P(E = 1, Q = 1)} = \frac{0}{0.5} = 0 \\
\#n_{21} &= \frac{P(O = 1, I = 1, J = 0, E = 0, Q = 0)}{P(E = 0, Q = 0)} = \frac{0}{0.5} = 0 \\
\#n_{22} &= \frac{P(O = 1, I = 1, J = 0, E = 0, Q = 0)}{P(E = 0, Q = 0)} = \frac{0}{0.5} = 0 \\
\#n_{23} &= \frac{P(O = 1, I = 0, J = 1, E = 0, Q = 0)}{P(E = 0, Q = 0)} = \frac{0}{0.5} = 0
\end{aligned}$$

$$\begin{aligned}\#n_{24} &= \frac{P(O = 1, I = 0, J = 0, E = 0, Q = 0)}{P(E = 0, Q = 0)} = \frac{0}{0.5} = 0 \\ \#n_{25} &= \frac{P(O = 0, I = 1, J = 1, E = 0, Q = 0)}{P(E = 0, Q = 0)} = \frac{0.1}{0.5} = 0.2 \\ \#n_{26} &= \frac{P(O = 0, I = 1, J = 0, E = 0, Q = 0)}{P(E = 0, Q = 0)} = \frac{0.15}{0.5} = 0.3 \\ \#n_{27} &= \frac{P(O = 0, I = 0, J = 1, E = 0, Q = 0)}{P(E = 0, Q = 0)} = \frac{0}{0.5} = 0 \\ \#n_{28} &= \frac{P(O = 0, I = 0, J = 0, E = 0, Q = 0)}{P(E = 0, Q = 0)} = \frac{0.25}{0.5} = 0.5\end{aligned}$$

When occurrence numbers  $\#n_{ij}$  (s) are determined, missing data is filled fully and evidence sample  $\mathcal{D}$  shown in table III.1.16 is completed as follows:

	$O$	$I$	$J$	$E$	$Q$	#Occurrences
$X^{(1)}$	1	1	1	1	1	0.5
$X^{(1)}$	1	1	0	1	1	0
$X^{(1)}$	1	0	1	1	1	0.3
$X^{(1)}$	1	0	0	1	1	0.2
$X^{(1)}$	0	1	1	1	1	0
$X^{(1)}$	0	1	0	1	1	0
$X^{(1)}$	0	0	1	1	1	0
$X^{(1)}$	0	0	0	1	1	0
$X^{(2)}$	1	1	1	1	0	0
$X^{(2)}$	1	1	0	1	0	0
$X^{(2)}$	1	0	1	1	0	0
$X^{(2)}$	1	0	0	1	0	0
$X^{(2)}$	0	1	1	1	0	0.2
$X^{(2)}$	0	1	0	1	0	0.3
$X^{(2)}$	0	0	1	1	0	0
$X^{(2)}$	0	0	0	1	0	0.5

**Table III.1.17.** Complete sample with two evidences ( $E=1, Q=1$ ) and ( $E=1, Q=0$ )

When missing values are estimated as in table III.1.17, it is easy to calculate counters  $s_{ij}$  and  $t_{ij}$  (s) which are ultimate results from E-step of EM algorithm.

- The counter  $s_{11}$  ( $t_{11}$ ) are numbers of evidences such that  $O=1$  ( $O=0$ ), which corresponds to variable  $F_{11}$ .
- The counter  $s_{21}$  ( $t_{11}$ ) are numbers of evidences such that  $I=1$  ( $I=0$ ), which corresponds to variable  $F_{21}$ .
- The counter  $s_{31}$  ( $t_{31}$ ) are numbers of evidences such that  $J=1$  ( $J=0$ ) given  $O=1$  and  $I=1$ , which corresponds to variable  $F_{31}$ .
- The counter  $s_{32}$  ( $t_{32}$ ) are numbers of evidences such that  $J=1$  ( $J=0$ ) given  $O=1$  and  $I=0$ , which corresponds to variable  $F_{32}$ .

### III.1. Knowledge sub-model

---

- The counter  $s_{33}$  ( $t_{33}$ ) are numbers of evidences such that  $J=1$  ( $J=0$ ) given  $O=0$  and  $I=1$ , which corresponds to variable  $F_{33}$ .
- The counter  $s_{34}$  ( $t_{34}$ ) are numbers of evidences such that  $J=1$  ( $J=0$ ) given  $O=0$  and  $I=0$ , which corresponds to variable  $F_{34}$ .
- The counter  $s_{41}$  ( $t_{41}$ ) are numbers of evidences such that  $E=1$  ( $E=0$ ) given  $O=1$ , which corresponds to variable  $F_{41}$ .
- The counter  $s_{42}$  ( $t_{42}$ ) are numbers of evidences such that  $E=1$  ( $E=0$ ) given  $O=0$ , which corresponds to variable  $F_{42}$ .
- The counter  $s_{51}$  ( $t_{51}$ ) are numbers of evidences such that  $Q=1$  ( $Q=0$ ) given  $O=1$ , which corresponds to variable  $F_{51}$ .
- The counter  $s_{52}$  ( $t_{52}$ ) are numbers of evidences such that  $Q=1$  ( $Q=0$ ) given  $O=0$ , which corresponds to variable  $F_{52}$ .

Please pay attention that posterior parameters  $a_{ij}$  and  $b_{ij}$  (s) are calculated based on counters  $s_{ij}$  and  $t_{ij}$  (s). From complete sample  $\mathcal{D}$  in table III.1.17, we have:

$s_{11}=0.5+0.3+0.2=1$	$t_{11}=0.5+0.3+0.2=1$
$s_{21}=0.5+0.3+0.2=1$	$t_{21}=0.3+0.2+0.5=1$
$s_{31}=0.5$	$t_{31}=0$
$s_{32}=0.3$	$t_{32}=0.2$
$s_{33}=0.2$	$t_{33}=0.3$
$s_{34}=0$	$t_{34}=0.5$
$s_{41}=0.5+0.3+0.2=1$	$t_{41}=0$
$s_{42}=0.2+0.3+0.5=1$	$t_{42}=0$
$s_{51}=0.5+0.3+0.2=1$	$t_{51}=0$
$s_{52}=0$	$t_{52}=0.2+0.3+0.5=1$

**Table III.1.18.** Counters  $s_{ij}$  and  $t_{ij}$  (s) from estimated values

The next step of EM algorithm, M-step is responsible for updating posterior density functions  $\beta(F_{ij}/\mathcal{D})$  (s), which leads to update posterior probabilities  $P(O=1/\mathcal{D})$ ,  $P(I=1/\mathcal{D})$ ,  $P(J=1|O=1,I=1,\mathcal{D})$ ,  $P(J=1|O=1,I=0,\mathcal{D})$ ,  $P(J=1|O=0,I=1,\mathcal{D})$ ,  $P(J=1|O=0,I=0,\mathcal{D})$ ,  $P(E=1|O=1,\mathcal{D})$ ,  $P(E=1|O=0,\mathcal{D})$ ,  $P(Q=1|O=1,\mathcal{D})$ ,  $P(Q=1|O=0,\mathcal{D})$ , based on current counters  $s_{ij}$  and  $t_{ij}$  from complete evidence sample  $\mathcal{D}$  (table III.1.18). Table III.1.19 shows results of M-step which are these posterior density functions and posterior probabilities. Note that origin parameters such as  $a_{11}=50$ ,  $b_{11}=50$ ,  $a_{21}=50$ ,  $b_{21}=50$ ,  $a_{31}=25$ ,  $b_{31}=0$ ,  $a_{32}=15$ ,  $b_{32}=10$ ,  $a_{33}=10$ ,  $b_{33}=15$ ,  $a_{34}=0$ ,  $b_{34}=25$ ,  $a_{41}=40$ ,  $b_{41}=10$ ,  $a_{42}=10$ ,  $b_{42}=40$ ,  $a_{51}=10$ ,  $b_{51}=40$ ,  $a_{52}=40$ , and  $b_{52}=10$  (see table III.1.14) are kept intact.

$$\begin{aligned}
 \beta(F_{11}/\mathcal{D}) &= \beta(F_{11}; a_{11}+s_{11}, b_{11}+t_{11}) = \beta(F_{11}; 50+1, 50+1) = \beta(F_{11}; 51, 51) \\
 \beta(F_{21}/\mathcal{D}) &= \beta(F_{21}; a_{21}+s_{21}, b_{21}+t_{21}) = \beta(F_{21}; 50+1, 50+1) = \beta(F_{21}; 51, 51) \\
 \beta(F_{31}/\mathcal{D}) &= \beta(F_{31}; a_{31}+s_{31}, b_{31}+t_{31}) = \beta(F_{31}; 25+0.5, 0+0) = \beta(F_{31}; 25.5, 0) \\
 \beta(F_{32}/\mathcal{D}) &= \beta(F_{32}; a_{32}+s_{32}, b_{32}+t_{32}) = \beta(F_{32}; 15+0.3, 10+0.2) = \beta(F_{32}; 15.3, 10.2) \\
 \beta(F_{33}/\mathcal{D}) &= \beta(F_{33}; a_{33}+s_{33}, b_{33}+t_{33}) = \beta(F_{33}; 10+0.2, 15+0.3) = \beta(F_{33}; 10.2, 15.3) \\
 \beta(F_{34}/\mathcal{D}) &= \beta(F_{34}; a_{34}+s_{34}, b_{34}+t_{34}) = \beta(F_{34}; 0+0, 25+0.5) = \beta(F_{34}; 0, 25.5) \\
 \beta(F_{41}/\mathcal{D}) &= \beta(F_{41}; a_{41}+s_{41}, b_{41}+t_{41}) = \beta(F_{41}; 50+1, 0+0) = \beta(F_{41}; 51, 0) \\
 \beta(F_{42}/\mathcal{D}) &= \beta(F_{42}; a_{42}+s_{42}, b_{42}+t_{42}) = \beta(F_{42}; 0+1, 50+0) = \beta(F_{42}; 1, 50)
 \end{aligned}$$

$$\begin{aligned}\beta(F_{51}/\mathcal{D}) &= \beta(F_{51}; a_{51}+s_{51}, b_{51}+t_{51}) = \beta(F_{51}; 50+1, 0+0) = \beta(F_{51}; 51, 0) \\ \beta(F_{52}/\mathcal{D}) &= \beta(F_{52}; a_{52}+s_{52}, b_{52}+t_{52}) = \beta(F_{52}; 0+0, 50+1) = \beta(F_{52}; 0, 51)\end{aligned}$$

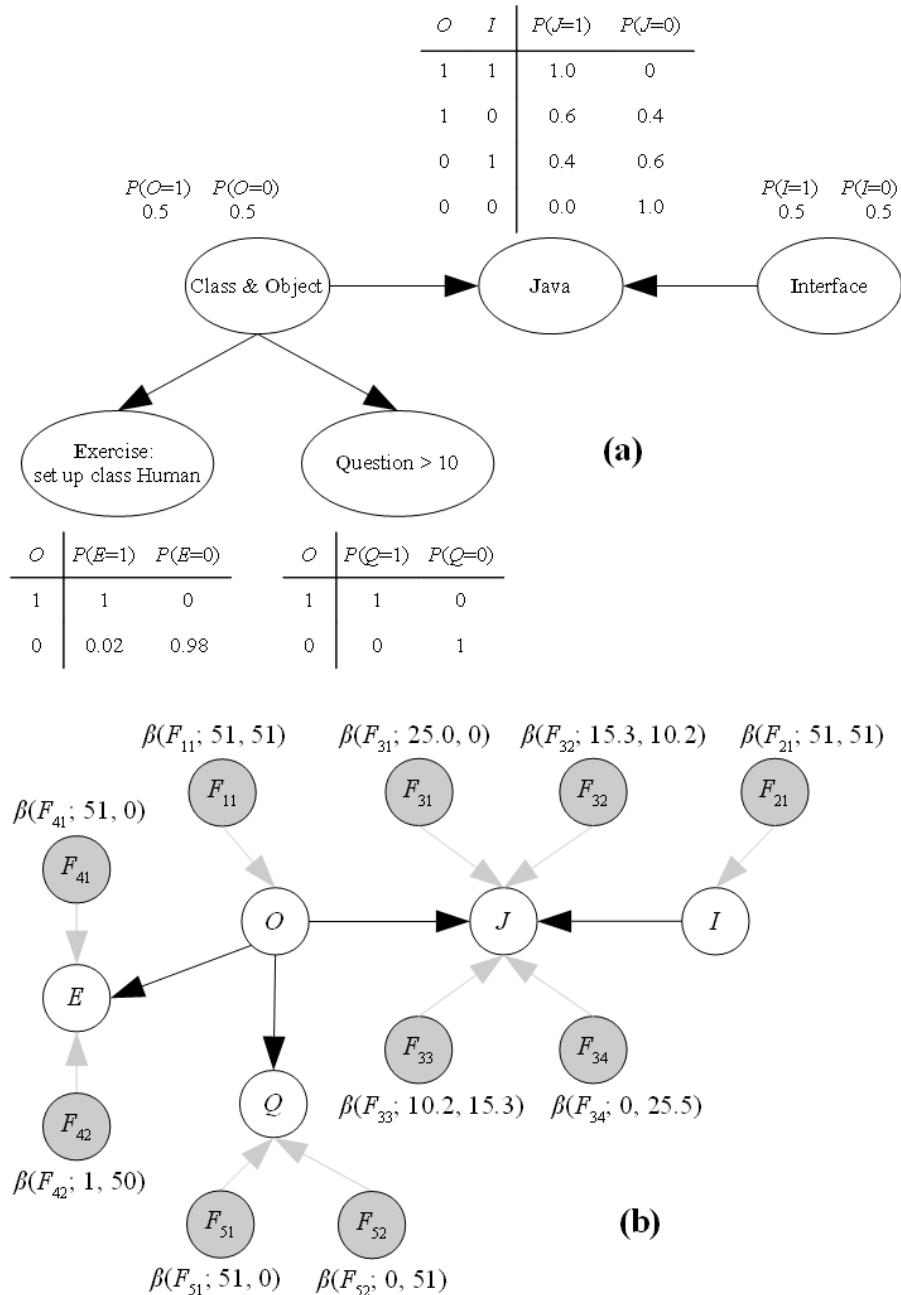
$$\begin{aligned}P(O=1/\mathcal{D}) &= E(F_{11}/\mathcal{D}) = 51/(51+51) = 0.5 \\ P(I=1/\mathcal{D}) &= E(F_{21}/\mathcal{D}) = 51/(51+51) = 0.5 \\ P(J=1|O=1, I=1, \mathcal{D}) &= E(F_{31}/\mathcal{D}) = 25.5/(25.5+0) = 1 \\ P(J=1|O=1, I=0, \mathcal{D}) &= E(F_{32}/\mathcal{D}) = 15.3/(15.3+10.2) = 0.6 \\ P(J=1|O=0, I=1, \mathcal{D}) &= E(F_{33}/\mathcal{D}) = 10.2/(10.2+15.3) = 0.4 \\ P(J=1|O=0, I=0, \mathcal{D}) &= E(F_{34}/\mathcal{D}) = 0/(0+25.5) = 0 \\ P(E=1|O=1, \mathcal{D}) &= E(F_{41}/\mathcal{D}) = 51/(51+0) = 1 \\ P(E=1|O=0, \mathcal{D}) &= E(F_{42}/\mathcal{D}) = 1/(1+50) \approx 0.02 \\ P(Q=1|O=1, \mathcal{D}) &= E(F_{51}/\mathcal{D}) = 51/(51+0) = 1 \\ P(Q=1|O=0, \mathcal{D}) &= E(F_{52}/\mathcal{D}) = 0/(0+51) = 0\end{aligned}$$

$$\begin{aligned}P(O=0/\mathcal{D}) &= 1-P(O=1/\mathcal{D}) = 0.5 \\ P(I=0/\mathcal{D}) &= 1-P(I=1/\mathcal{D}) = 0.5 \\ P(J=0|O=1, I=1, \mathcal{D}) &= 1-P(J=1|O=1, I=1, \mathcal{D}) = 0 \\ P(J=0|O=1, I=0, \mathcal{D}) &= 1-P(J=1|O=1, I=0, \mathcal{D}) = 0.4 \\ P(J=0|O=0, I=1, \mathcal{D}) &= 1-P(J=1|O=0, I=1, \mathcal{D}) = 0.6 \\ P(J=0|O=0, I=0, \mathcal{D}) &= 1-P(J=1|O=0, I=0, \mathcal{D}) = 1 \\ P(E=0|O=1, \mathcal{D}) &= 1-P(E=1|O=1, \mathcal{D}) = 0 \\ P(E=0|O=0, \mathcal{D}) &= 1-P(E=1|O=0, \mathcal{D}) = 0.98 \\ P(Q=0|O=1, \mathcal{D}) &= 1-P(Q=1|O=1, \mathcal{D}) = 0 \\ P(Q=0|O=0, \mathcal{D}) &= 1-P(Q=1|O=0, \mathcal{D}) = 1\end{aligned}$$

**Table III.1.19.** Posterior density functions and posterior probabilities are evolved based on counters  $s_{ij}$  and  $t_{ij}$

By posterior CPT (s) shown in table III.1.19 which is the ultimate result of EM algorithm, the Bayesian overlay model of Java course in figure III.1.18 is converted into the evolutional version specified in figure III.1.19.

### III.1. Knowledge sub-model



**Figure III.1.19.** Evolutional version of BN (a) and augmented BN (b) for Java course

If we continue to apply EM algorithm whenever observed evidences are raised, the posterior density functions are updated and become more accurate after many iterations because the limit  $\lim_{k \rightarrow +\infty} \frac{a_{ij}^{(k)} + s_{ij}^{(k)}}{a_{ij}^{(k)} + s_{ij}^{(k)} + b_{ij}^{(k)} + t_{ij}^{(k)}}$  will gain certain value; please see previous sub-section III.1.3.2. In general, this Java course example is an extension of example in previous sub-section III.1.3.2, which help us to know clearly combination of Bayesian network and overlay model (sub-section III.1.1.2) so as to construct Bayesian overlay (knowledge) sub-model and applying EM algorithm into making evolution of Bayesian overlay model in case of data missing.

Sub-section [III.1.3](#) focusing on evolution of Bayesian overlay model ends up here. In general, BN is a powerful mathematical tool for reasoning but it is restricted by unimproved initial parameters. This sub-section [III.1.3](#) suggests the approach to parameter evolution that uses the EM algorithm for beta functions. Note that the 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 ( $a_{ij}, b_{ij}$ ) by weights of arcs; please understand deeply sigma graph and theorem of SIGMA-gate inference mentioned in previous sub-sections [III.1.1.2](#) and [III.1.1.3](#). SIGMA-gate inference is the basic theory for transforming weights of arcs into parameters ( $a_{ij}, b_{ij}$ ) and CPT (s).

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 sub-section [III.1.4](#). Your attention please, as aforementioned there are two ways to improve BN such as parameter learning which was mentioned in sub-section [III.1.3](#) and structure learning which will be described in next sub-section [III.1.4](#). Hence, sub-section [III.1.4](#) proposed an interesting and innovative approach to improve (and construct) knowledge sub-model that allows to monitor chronologically users' process of gaining knowledge by using dynamic Bayesian network.

### **III.1.4. Improving knowledge sub-model by using dynamic Bayesian network**

Normal Bayesian network (BN) described in previous sub-section [III.1.3](#) is effective approach to make reasoning on knowledge model but dynamic Bayesian network (DBN) is more robust than BN for modeling users' knowledge when DBN allows monitoring user's process of gaining knowledge and evaluating her/his knowledge (Neapolitan, 2003, pp. 272-279). However the size of DBN becomes numerous when the process continues for a long time; thus, performing probabilistic inference will be inefficient. Moreover the number of transition dependencies among points in time is too large to compute posterior marginal probabilities when doing inference in DBN. Note that normal BN introduced in previous sub-section [III.1.1.1](#) is known as static BN opposite to DBN mention in this sub-section [III.1.4](#) and it is conventional that BN is implicated as static BN if there is no additional explanation.

To overcome these difficulties, I propose the new algorithm that both the size of DBN and the number of conditional probability tables (CPT) in DBN are kept intact (not changed) when the process continues for a long time. This method includes six steps: initializing DBN, specifying transition weights, re-constructing DBN, normalizing weights of dependencies, re-defining CPT(s) and probabilistic inference (Nguyen, A New Algorithm for Modeling and Inferring User's Knowledge by Using Dynamic Bayesian Network, 2014). My

algorithm also solves the problem of *temporary slip* and *lucky guess*: “learner does (doesn’t) know a particular subject but there is solid evidence convincing that she/he doesn’t (does) understand it; this evidence just reflects a temporary slip (or lucky guess)”. The document (Reye, 2004) is the good description of concepts “temporal slip” and “lucky guess”.

As aforementioned in sub-section III.1.1, my solution of building up knowledge is to combine BN and overlay model and so such knowledge sub-model is called Bayesian overlay (sub-) model. The main purpose of this sub-section III.1.4 is to improve or re-structure Bayesian overlay model by the proposed algorithm based on DBN, which implicates that the essence of such algorithm is structure learning approach for Bayesian network. Before describing the algorithm to improve Bayesian overlay model by using DBN in sub-section III.1.4.2, we should glance over what dynamic Bayesian network is in sub-section III.1.4.1. Sub-section III.1.4.3 is the evaluation.

### III.1.4.1. Dynamic Bayesian network

Bayesian network (BN) provides a powerful inference mechanism based on evidences but it cannot model temporal relationships between variables. It only represents directed acyclic graph (DAG) at a certain time point. In some situations, capturing the dynamic (temporal) aspect is very important; especially in e-learning context it is very necessary to monitor chronologically users’ process of gaining knowledge. So the purpose of dynamic Bayesian network (DBN) is to model the temporal relationships among variables (Neapolitan, 2003, p. 272); in other words, it represents DAG in the time series.

Suppose we have some finite number  $T$  of time points, let  $x_i[t]$  be the (temporal) variable representing the value of  $x_i$  at time  $t$  where  $0 \leq t \leq T$ . Let  $X[t] = \{x_1[t], x_2[t], \dots, x_n[t]\}$  be the temporal random vector denoting the random vector  $X = \{x_1, x_2, \dots, x_n\}$  at time  $t$ . A DBN is defined as a BN containing variables that comprise  $T$  variable vectors  $X[t]$  and determined by following specifications (Neapolitan, 2003, p. 273):

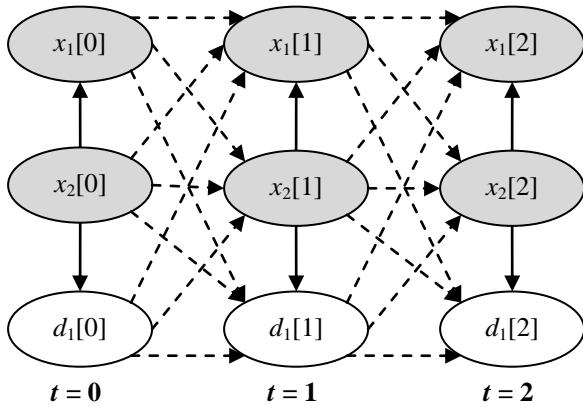
- An initial BN  $G_0 = \{X[0], P(X[0])\}$  at first time  $t = 0$ .
- A transition BN is a template consisting of a transition DAG  $G_{\rightarrow}$  containing variables in  $X[t-1] \cup X[t]$  and a transition probability distribution  $P_{\rightarrow}(X[t]/X[t-1])$  with  $t \geq 1$ . Recall that notation  $\cup$  denotes union operator in set theory (Wikipedia, Set (mathematics), 2014).

In short, the DBN consists of the initial DAG  $G_0$  and the transition DAG  $G_{\rightarrow}$  evaluated at time  $t$  where  $0 \leq t \leq T$ . The **Distributed Global Joint Probability Distribution** of DBN so-called DGJPD is product of conditional probability distribution of  $G_0$  and product of all  $P_{\rightarrow}$  (s) valued at all time points, which is specified in formula III.1.35 as follows:

$$P(X[0], X[1], \dots, X[T]) = P_0(X[0]) \prod_{t=1}^T P_{\rightarrow}(X[t]/X[t-1])$$

*Formula III.1.35. Distributed global joint probability distribution of DBN*

Where  $P_0(\cdot)$  is the conditional probability distribution (CPD) of  $G_0$ ; please see sub-section III.1.1.1 for more details about CPD and conditional probability table (CPT). Note that the transition (temporal) probability can be considered the transition (temporal) dependency. An example of DBN is shown in figure III.1.20.



**Figure III.1.20.** DBN for  $t = 0, 1, 2$

Note, non-evidence variables (hypothesis variables) are shaded; otherwise, evidence variables ( $d_1[0]$ ,  $d_1[1]$ ,  $d_1[2]$ ) are not shaded. Dash lines - - - denotes transition probabilities or transition dependencies of  $G_{\rightarrow}$  between consecutive points in time. In figure III.1.20, we have  $X[0] = \{x_1[0], x_2[0], x_3[0]\}$ ,  $X[1] = \{x_1[1], x_2[1], x_3[1]\}$  and  $X[2] = \{x_1[2], x_2[2], x_3[2]\}$  where  $x_3[0] = d_1[0]$ ,  $x_3[1] = d_1[1]$ , and  $x_3[2] = d_1[2]$ . Please pay attention to temporal random vector  $X[t]$  because it is used over this whole sub-section III.1.4.

The essence of learning DBN is to specify the initial BN and the transition probability distribution  $P_{\rightarrow}$ . It is possible to specify the transition probability distribution  $P_{\rightarrow}$  by applying the scored-based approach that selects optimal probabilistic network according to some criterions; this is a backward or forward selection or the leaps and bounds algorithms (Hastie, Tibshirani, & Friedman, 2009, pp. 57-60). It is also possible to use greedy search (Neapolitan, 2003, p. 513) or MCMC algorithm (Neapolitan, 2003, pp. 453-462) to select the best output DBN. Authors (Friedman, Murphy, & Russell, 1998) propose criterions such as BIC score and BDe score to select and learn DBN from complete and incomplete data; this approach uses the structural expectation maximization (SEM) algorithm that combines network structure and parameter into single expectation maximization (EM) process (Friedman, Murphy, & Russell, 1998). Some other algorithms such as Baum Welch algorithm (Mills, 1997) take advantages of the similarity of DBN and hidden Markov model (HMM) in order to learn DBN from the aspects of HMM when HMM is the simple case of DBN. In general, learning DBN is an extension of learning static BN and there are two main BN learning approaches (Neapolitan, 2003, pp. 441-606):

- Scored-based approach: given scoring criterion  $\delta$  assigned to every BN, which BN gains highest  $\delta$  is the best BN. This criterion  $\delta$  is computed as

- the posterior probability over whole BN given training data set (Neapolitan, 2003, p. 445).
- Constraint-based approach (Neapolitan, 2003, p. 541): given a set of constraints, which BN satisfies over all such constraints is the best BN. Constraints are defined as rules relating to Markov condition (Neapolitan, 2003, p. 31). Markov condition or Markov property will be mentioned in next sub-section [III.1.4.2](#).

Please refer to (Murphy, 2002) for more details about learning DBN. These approaches can give the precise results with the best-learned DBN but they become inefficient when the number of variables gets huge. It is impossible to learn DBN by the same way done in case of static BN when the training data is enormous. Moreover, these approaches cannot response in real time if there is requirement of creating DBN from continuous and instant data stream. Following are drawbacks of inference in DBN and the proposal of this research.

### **Drawbacks of inferences in DBN**

Formula [III.1.35](#) is considered as extension of formula [III.1.2](#); so, the posterior probability of each temporal variable is now computed by using Distributed Global Joint Probability Distribution (DGJPD) in formula [III.1.35](#) which is much more complex than normal Global Joint Probability Distribution (GJPD) in formula [III.1.2](#). Whenever the posterior of a variable evaluated time point  $t$  needs to be computed, all temporal random vectors  $X[0], X[1], \dots, X[t]$  must be included for executing Bayes' rule because DGJPD is product of all transition  $P_{\rightarrow}(s)$  evaluated at  $t$  points in time. Suppose the initial DAG has  $n$  variables ( $X[0] = \{x_1[0], x_2[0], \dots, x_n[0]\}$ ), there are  $n*(t+1)$  temporal variables concerned in time series (0, 1, 2, ...,  $t$ ). It is impossible to take into account such an extremely large number of temporal variables in  $X[0] \cup X[1] \cup \dots \cup X[t]$ . In other words, the size of DBN becomes numerous when the process continues for a long time; thus, performing probabilistic inference will be inefficient.

Moreover suppose  $G_0$  has  $n$  variables, we must specify  $n*n$  transition dependencies between variables  $x_i[t-1] \in X[t-1]$  and variables  $x_i[t] \in X[t]$ . Through  $t$  time points, there are  $n*n*t$  transition dependencies. So it is impossible to compute effectively the transition probability distribution  $P_{\rightarrow}(X[t]/X[t-1])$  and the DGJPD in formula [III.1.35](#). Thus, the proposed algorithm described in next sub-section [III.1.4.2](#) overcomes the drawbacks of DBN.

### **III.1.4.2. Using dynamic Bayesian network to model user's knowledge**

To overcome drawbacks of DBN, I propose the new algorithm that both the size of DBN and the number of CPT(s) in DBN are kept intact (not changed) when the process continues for a long time. However we should glance over some definitions before discussing our method. Suppose the DAG has  $n$  variables, given  $pa_i[t]$  is a set of parents of  $x_i$  at time point  $t$ , namely parents of

$x_i[t]$ , the transition probability distribution is computed as below (Neapolitan, 2003, p. 274):

$$P_{\rightarrow}(X[t]|X[t-1]) = \prod_{i=1}^n P_{\rightarrow}(x_i[t]|pa_i[t])$$

*Formula III.1.36.* Transition probability distribution

Note,  $pa_i[t]$  is subset of union of  $X[t-1]$  and  $X[t]$ ,  $pa_i[t] \subseteq X[t-1] \cup X[t]$ . Applying formula III.1.36 for all  $X$  given  $t$ , suppose we have:

$$P_{\rightarrow}(X[t]|X[t-1], X[t-2], \dots, X[0]) = P_{\rightarrow}(X[t]|X[t-1])$$

*Formula III.1.37.* Markov property according to transition probability distribution

If the DBN meets fully formula III.1.37, it has *Markov property*, namely, given previous time point  $t-1$ , the conditional probability of current time point  $t$  is only dependent on the previous time point  $t-1$ , not relevant to any further past time point ( $t-2, t-3, \dots, 0$ ). Furthermore, the DBN is *stationary* if  $P_{\rightarrow}(X[t]/X[t-1])$  is the same for all  $t \geq 1$ .

Suppose DBN is stationary and has Markov property, I propose a new algorithm for modeling and inferring user's knowledge by using DBN; thus, each time there are occurrences of evidences, DBN is re-constructed and probabilistic inference is done by six following steps:

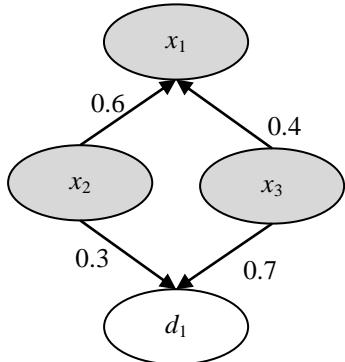
- |   |
|---|
| Step 1. <i>Initializing DBN</i> : All variables (nodes) and relationships (arcs) among variables of initial Bayesian network $G_0$ must be specified. The strength of relationship is considered as weight of arc.  |
| Step 2. <i>Specifying transition weights</i> : The transition weights expressing conditional transition probabilities of current network $G_t$ given previous network $G_{t-1}$ are specified based on factors slip and guess.                            |
| Step 3. <i>(Re-)constructing DBN</i> : The DBN at current time point $t$ given previous time point $t-1$ is (re-)constructed based on current network $G_t$ , previous network $G_{t-1}$ and transition weights.  |
| Step 4. <i>Normalizing weights of relationships</i> : All weights which express relationships among variables inside DBN at current time point $t$ are normalized so that sum of these weights are equal to 1.  |
| Step 5. <i>(Re-)defining CPT (s)</i> : All CPT (s) of DBN at current time point $t$ are updated based on normalized weights (in step 4) and posterior probabilities. Note that the posterior probabilities were computed in step 6 of previous iteration. |
| Step 6. <i>Probabilistic inference</i> : The posterior probabilities at current time point $t$ are computed according to Bayesian inference. These probabilities will be used to update CPT (s) in step 5 of next iteration.                              |

**Table III.1.20.** Six steps of new algorithm for modeling and inferring user's knowledge by using DBN

As seen in table III.1.20, six steps are repeated whenever evidences occur. Each iteration gives the view of DBN at certain point in time. After  $t^{th}$  iteration, the posterior marginal probability of random vector  $X$  in DBN will approach a certain limit; it means that DBN converges at that time.

Because there are an extremely large number of variables included in DBN for a long time, we focus a subclass of DBN in which network in different time steps are connected only through non-evidence variables ( $x_i$ ).

Suppose there is learning course in which the domain model has four knowledge elements  $x_1, x_2, x_3, d_1$ . The item  $d_1$  is the evidence that tells us how learners are mastered over  $x_1, x_2, x_3$ . This domain model is represented as a BN having three non-evidence variables  $x_1, x_2, x_3$  and one evidence variable  $d_1$ . The weight of an arc from parent variable to child variable represents the strength of relationship among them. For instance, the weight of arc from  $x_2$  to  $x_1$  measures the relevant importance of  $x_2$  in  $x_1$ . This BN with full of weights regarded as an example for our algorithm is shown in figure III.1.21. Note that there are many relationships in BN such as prerequisite, dependency, parent-child, cause-effect, aggregation, and diagnostic; each relationship is specified according to application context (see sub-sections I.1.2.2 and III.1.1.2).

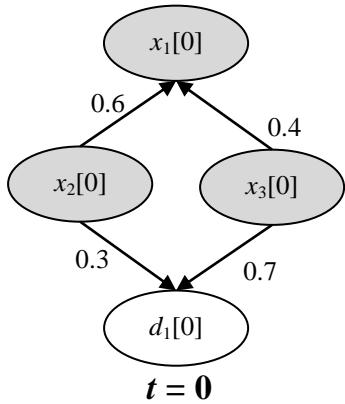


**Figure III.1.21.** The BN sample with full of weights

In figure III.1.21, non-evidence variables (hypothesis variables) are shaded; otherwise, evidence variable ( $d_1$ ) is not shaded; moreover weights (0.6, 0.4, 0.3, 0.7) are shown nearby arcs. The remaining part of this sub-section III.1.4.2 will describe six steps of the proposed algorithm in detailed.

### Step 1: Initializing DBN

If time point  $t > 0$  then jumping to step 3. Otherwise, all variables (nodes) and relationships (arcs) among variables of initial BN  $G_0$  must be specified. The strength of relationship is considered as weight of arc. Figure III.1.22 shows the initial BN  $G_0$  with full of weights when time point  $t = 0$ . Note that weights (0.6, 0.4, 0.3, 0.7) are shown nearby arcs and these weights are called *ordinary weights* in order to distinguish them from transition weight mentioned in step 2.



**Figure III.1.22.** Initial BN  $G_0$  derived from BN in figure III.1.21

In figure III.1.22, non-evidence variables (hypothesis variables) are shaded; otherwise, evidence variable ( $d_1[0]$ ) is not shaded and so we have  $X[0]=\{x_1[0], x_2[0], x_3[0], x_4[0]\}$  where  $x_4[0]=d_1[0]$  is the evidence. Note  $X[t]$  is the temporal random vector  $X=\{x_1, x_2, x_3, x_4=d_1\}$  at time  $t$ ; please see sub-section III.1.4.1 for basic concepts about DBN.

### Step 2: Specifying transition weight

Given two factors: *slip* and *guess* where *slip* (*guess*) factor expresses the situation that user does (doesn't) know a particular subject but there is solid evidence convincing that she/he doesn't (does) understand it; this evidence just reflects a temporary slip (or lucky guess). *Slip* factor is essentially probability that user has known concept/subject  $x$  before but she/he forgets it now. Otherwise *guess* factor is essentially probability that user hasn't known concept/subject  $x$  before but she/he knows it now. Please read the document (Reye, 2004) which is the good description of concepts “*slip*” and “*guess*”. Suppose  $x[t-1]$  and  $x[t]$  denote the user's state of knowledge about  $x$  at two consecutive time points  $t-1$  and  $t$ , respectively. Both  $x[t-1]$  and  $x[t]$  are temporal variables referring the same knowledge element  $x$ . Factors *slip* and *guess* are formulated by formula III.1.38 as follows:

$$\begin{aligned} \text{slip} &= P(\text{not } x[t] | x[t-1]) \\ \text{guess} &= P(x[t] | \text{not } x[t-1]) \end{aligned}$$

*Formula III.1.38.* Formula of slip factor and guess factor

Where  $0 \leq \text{slip}, \text{guess} \leq 1$  and the sign “*not*” denotes negation operator, for example, *not*  $x=0$  if  $x=1$ .

So the conditional probability (named  $a$ ) of event that user knows  $x[t+1]$  given event that she/he has already known  $x[t]$  has value  $1-\text{slip}$ . The conditional probability  $a$  is proved and expressed in formula III.1.39.

$$a = P(x[t] | x[t-1]) = 1 - P(\text{not } x[t] | x[t-1]) = 1 - \text{slip}$$

*Formula III.1.39.* Conditional probability  $a$

The bias  $b$  is defined as differences of an amount of knowledge user gains about  $x$  between time points  $t-1$  and  $t$ . Formula III.1.40 represents the bias  $b$  of user knowledge.

$$b = \frac{1}{1 + P(x[t]|not\ x[t-1])} = \frac{1}{1 + guess}$$

*Formula III.1.40.* The bias  $b$  of user knowledge

The smaller the guess factor is, the larger the bias  $b$  is, which implies that user's knowledge is really enhanced in chronologic order.

Now the weight  $w$  expressing strength of (temporal) dependency between  $x[t-1]$  and  $x[t]$  is defined as product of the conditional probability  $a$  and the bias  $b$ . Note that (temporal) dependency is known as a relationship when there are many relationships such as prerequisite, dependency, parent-child, cause-effect, aggregation, and diagnostic (see sub-sections I.1.2.2 and III.1.1.2). The weight  $w$  is specified by formula III.1.41.

$$w = a * b = (1 - slip) \frac{1}{1 + guess}$$

*Formula III.1.41.* The weight  $w$  as product of conditional probability  $a$  and bias  $b$

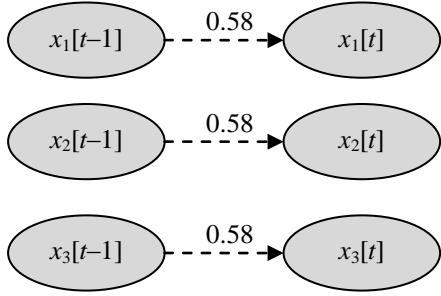
Expanding to temporal random vectors,  $w$  is considered as the weight of arcs from temporal vector  $X[t-1]$  to temporal vector  $X[t]$ . Thus the weight  $w$  implicates that the conditional transition probability of  $X[t]$  given  $X[t-1]$  satisfies both Markov property and stationary property; please see formula III.1.42 as follows:

The weight  $w$  implicates that

$$P_{\rightarrow}(X[t]|X[t-1]) = P_{\rightarrow}(X[t+1]|X[t]), \forall t \geq 1$$

*Formula III.1.42.* The weight  $w$  implicates that the conditional transition probability satisfies both Markov property and stationary property

So  $w$  is called *temporal weight* or *transition weight* and all transition dependencies have the same weight  $w$ . Suppose  $slip = 0.3$  and  $guess = 0.2$  in our example, we have  $w = (1 - 0.3) \frac{1}{1+0.2} = 0.58$  according to formula III.1.41. Figure III.1.23 expresses transition weights between two points in time  $t-1$  and  $t$  with regard to variables in figures III.1.21 and III.1.22.



**Figure III.1.23.** Transition weights

In figure III.1.23, dash lines - - - denotes transition probabilities or transition dependencies between consecutive points in time ( $t-1$  and  $t$ ).

### Step 3: Re-constructing DBN

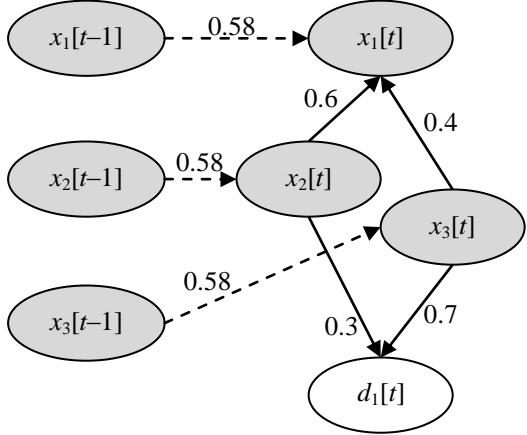
Because our DBN is stationary and has Markov property (see formulas III.1.37 and III.1.42), we only focus its previous adjoining state at any point in time. We concern DBN at two consecutive time points  $t-1$  and  $t$ . For each time point  $t$ , we create a new BN  $G'[t]$  whose variables include all variables in  $X[t-1] \cup X[t]$  except evidences in  $X[t-1]$ .  $G'[t]$  is called *expended BN* at time point  $t$ . The set of such variables constituting  $G'[t]$  is denoted  $Y$  which is represented in formula III.1.43 with assumption that there are  $n$  variables and  $k$  evidences among such  $n$  variables.

$$\begin{aligned} Y &= (X[t-1] \cup X[t]) \setminus \mathcal{D}[t-1] \\ &= \{x_1[t-1], x_2[t-1], \dots, x_n[t-1], x_1[t], x_2[t], \dots, x_n[t]\} \\ &\quad \setminus \{d_1[t-1], d_2[t-1], \dots, d_k[t-1]\} \end{aligned}$$

*Formula III.1.43.* The set  $Y$  of all variables (nodes) of expended BN  $G'[t]$  at time point  $t$

Where  $\mathcal{D}[t-1] = \{d_1[t-1], d_2[t-1], \dots, d_k[t-1]\}$  is the set of evidences at time point  $t-1$ . Recall that the sign “\” denotes the subtraction (excluding) in set theory (Wikipedia, Set (mathematics), 2014).

A very important fact to which you should pay attention is that all relationships among variables in  $X[t-1]$  are removed from  $G'[t]$ . It means that no arc (or CPT) relevant to  $X[t-1]$  exists in  $G'[t]$  now. However each couple of variables  $x_i[t-1]$  and  $x_i[t]$  has a transition dependency which is added to  $G'[t]$ . The strength of such dependency is the temporal weight  $w$  specified in formula III.1.41. Hence every  $x_i[t]$  in  $X[t]$  has an additional parent which in turn is the variable  $x_{i-1}[t]$  in  $X[t-1]$  and the temporal relationship among them are weighted. In other words, vector  $X[t-1]$  becomes the input of vector  $X[t]$ . It is easy to infer that the expended BN  $G'[t]$  is a representation of DBN at time point  $t$ , as shown in figure III.1.24. The key of proposed algorithm in this subsection III.1.4.2 is to construct and improve such expended BN  $G'[t]$  and so, this step (step 3) aims to construct  $G'[t]$  while the essence of next steps 4, 5, and 6 is to improve  $G'[t]$ . It is possible to consider expended BN  $G'[t]$  as the DBN at certain time point  $t$ .



**Figure III.1.24.** DBN or expended BN with full of weights at time point  $t$

In figure III.1.24, non-evidence variables (hypothesis variables) are shaded; otherwise, evidence variable ( $d_1[t]$ ) is not shaded. Dash lines - - - denotes transition probabilities or transition dependencies between consecutive points in time. The set of all variables of expended BN  $G'[t]$  is  $Y = \{x_1[t-1], x_2[t-1], x_3[t-1], x_1[t], x_2[t], x_3[t], x_4[t]\}$  where  $x_4[t] = d_1[t]$  is the evidence. We consider  $Y$  as random vector of the whole expended BN  $G'[t]$ .

#### Step 4: Normalizing weights of relationships

Suppose  $x_1[t]$  has two parents  $x_2[t]$  and  $x_3[t]$ . Suppose ordinary weights of two arcs from  $x_2[t]$ ,  $x_3[t]$  to  $x_1[t]$  are  $w_2$ ,  $w_3$ , respectively. The essence of these weights is the strength of relationships inside random vector  $X[t]$  such that:

$$w_2 + w_3 = 1$$

Now in expended BN (DBN at time point  $t$ ), the transition weight  $w_1$  of temporal arc from  $x_1[t-1]$  to  $x_1[t]$  is specified according to formula III.1.41.

$$w_1 = a * b = (1 - \text{slip}) \frac{1}{1 + \text{guess}}$$

The weights  $w_1$ ,  $w_2$ ,  $w_3$  must be normalized because sum of them is larger than or equal to 1,  $w_1 + w_2 + w_3 \geq 1$  because of  $w_2 + w_3 = 1$  and  $w_1 \geq 0$ . Formula III.1.44 indicates the way to normalize ordinary weights  $w_2$ ,  $w_3$  and transition weight (temporal weight)  $w_1$  such that  $w_1+w_2+w_3=1$ .

$$\begin{aligned} w_2 &= w_2(1 - w_1) \\ w_3 &= w_3(1 - w_1) \end{aligned}$$

*Formula III.1.44.* Normalizing weights  $w_2$ ,  $w_3$

Suppose  $S$  is the sum of  $w_1$ ,  $w_2$  and  $w_3$ , we have:

$$\begin{aligned} S &= w_1 + w_2(1 - w_1) + w_3(1 - w_1) = w_1 + (w_2 + w_3)(1 - w_1) \\ &= w_1 + (1 - w_1) = 1 \end{aligned}$$

Expanding formula III.1.44 on general case, suppose variable  $x_i[t]$  has  $k-1$  weights  $w_{i2}, w_{i3}, \dots, w_{ik}$  corresponding to  $k-1$  parents and a transition weight  $w_{i1}$  of temporal relationship between  $x_i[t-1]$  and  $x_i[t]$ . We have formula III.1.45 for normalizing ordinary weights and transition weight in general case.

$$\begin{aligned} w_{i2} &= w_{i2}(1 - w_{i1}) \\ w_{i3} &= w_{i3}(1 - w_{i1}) \\ &\vdots \\ w_{ik} &= w_{ik}(1 - w_{i1}) \end{aligned}$$

*Formula III.1.45.* Normalizing weights in general case

After normalizing weights following formula [III.1.45](#), transition weight  $w_{i1}$  is kept intact but other weights  $w_{ij}$  ( $j > 1$ ) get smaller. So the meaning of formula [III.1.45](#) is to focus on transition probability and knowledge accumulation.

Because this formula is a suggestion, you can define the other one by yourself. Let  $w_i[t]$  be the set of normalized weights relevant to a variable  $x_i[t]$ , we have:

$$w_i[t] = \{w_{i1}, w_{i2}, w_{i3}, \dots, w_{ik}\} \text{ where } w_{i1} + w_{i2} + \dots + w_{ik} = 1$$

Going back the DBN (expended BN) constructed in step 3 (see figure [III.1.24](#)), it is easy to recognize that only weights relevant to variable  $x_1[t]$  require to be normalized because  $x_1[t]$  has two parent variables  $x_2[t], x_3[t]$  and one additional parent variable  $x_1[t-1]$  from random vector  $X[t-1]$  at previous time point  $t-1$ . Let  $w_{11}=0.58$ ,  $w_{12}=0.6$ , and  $w_{13}=0.4$  be transition weight of temporal dependency of  $x_1[t-1]$  and  $x_1[t]$ , ordinary weight of relationship from  $x_2[t]$  to  $x_1[t]$ , and ordinary weight of relationship from  $x_3[t]$  to  $x_1[t]$ , respectively. Ordinary weights are normalized by applying formula [III.1.45](#) as follows:

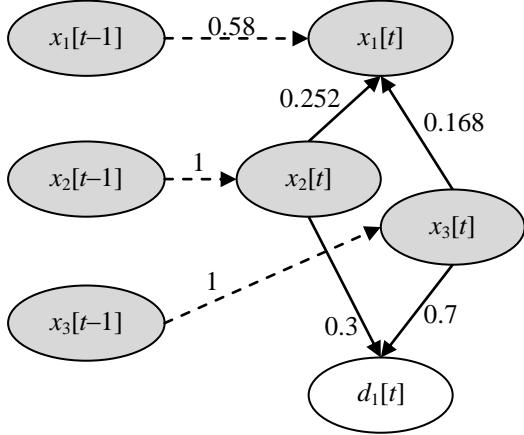
$$\begin{aligned} w_{12} &= w_{12}(1 - w_{11}) = 0.6 * (1 - 0.58) = 0.252 \\ w_{13} &= w_{13}(1 - w_{11}) = 0.4 * (1 - 0.58) = 0.168 \end{aligned}$$

Note that if a variable has only one temporal parent node, the weight of respective transition dependency should be changed into 1; this implicates that the transition dependency is the main cause of user's knowledge accumulation but you can specify the normalization by yourself. For example,  $x_2[t]$  has only one temporal parent node  $x_2[t-1]$  and so transition weight  $w_{21}=0.58$  of temporal dependency between  $x_2[t-1]$  and  $x_2[t]$  is changed into  $w_{21}=1$ . Table [III.1.21](#) lists weights and normalized weights relating to  $x_1[t]$ .

$x_1[t]$	$w_{11}=0.58$	$w_{12}=0.6$	$w_{13}=0.4$
$x_1[t]$ (normalized)	$w_{11}=0.58$	$w_{12}=0.252$	$w_{13}=0.168$
$x_2[t]$	$w_{21}=0.58$		
$x_2[t]$ (normalized)	$w_{21}=1$		
$x_3[t]$	$w_{31}=0.58$		
$x_3[t]$ (normalized)	$w_{31}=1$		

**Table III.1.21.** The weights relating  $x_1[t]$  are normalized

The following figure [III.1.25](#) shows the variant of expended BN (or DBN) at time point  $t$  (shown in figure [III.1.24](#)) whose weights are normalized.



**Figure III.1.25.** DBN or expended BN (at time point  $t$ ) whose weights are normalized

In figure III.1.25, non-evidence variables (hypothesis variables) are shaded; otherwise, evidence variable ( $d_1[t]$ ) is not shaded. Dash lines - - - denotes transition probabilities or transition dependencies between consecutive points in time. Now structure of DBN is determined as in figure III.1.25 and hence, it is required to specify parameters of DBN, namely CPT(s). Step 5 of proposed algorithm will mention how to specify CPT(s) based on normalized weights and posterior probabilities.

### Step 5: Re-defining CPT(s)

There are two random vectors  $X[t-1]$  and  $X[t]$ . So defining CPT(s) of DBN includes: determining CPT for each variable  $x_i[t-1] \in X[t-1]$  and re-defining CPT for each variable  $x_i[t] \in X[t]$ .

1. *Determining CPT (s) of  $X[t-1]$ .* The CPT of  $x_i[t-1]$  includes posterior probabilities which were computed in step 6 of previous iteration at time point  $t-1$ . The posterior probability of variable  $x_i[t-1]$  given  $\mathcal{D}[t-1] = \{d_1[t-1], d_2[t-1], \dots, d_k[t-1]\}$  is the set of evidences at time point  $t-1$  is:

$$P(x_i[t-1]|\mathcal{D}[t-1]) = \frac{\sum_{X[t-1] \setminus (\{x_i[t-1]\} \cup \mathcal{D}[t-1])} P(x_1[t-1], x_2[t-1], \dots, x_n[t-1])}{\sum_{X[t-1] \setminus \mathcal{D}[t-1]} P(x_1[t-1], x_2[t-1], \dots, x_n[t-1])}$$

*Formula III.1.46.* Posterior probability of each  $x_i[t-1]$  given evidences  $\mathcal{D}[t-1]$

Formula III.1.46 is the same to formula III.1.48 specified in step 6 when these posterior probabilities were computed in step 6 of previous iteration at time point  $t-1$ . Thus, please surveying step 6 for more details about probabilistic inference and posterior probabilities when there are some complicated details in formula III.1.46 which are not explained here yet except that formula III.1.46 is derived from formula III.1.3' described in sub-section III.1.1.1.

Going back our example of DBN shown in figure III.1.25, table III.1.22 expresses CPT (s) of  $x_1[t-1]$ ,  $x_2[t-1]$ , and  $x_3[t-1]$  as follows:

$P(x_1[t-1]=1)$	$\alpha_1$
$P(x_1[t-1]=0)$	$1-\alpha_1$
$P(x_2[t-1]=1)$	$\alpha_2$
$P(x_2[t-1]=0)$	$1-\alpha_2$
$P(x_3[t-1]=1)$	$\alpha_3$
$P(x_3[t-1]=0)$	$1-\alpha_3$

**Table III.1.22.** CPT (s) of  $X[t-1] = \{x_1[t-1], x_2[t-1], x_3[t-1]\}$

There is an interesting thing that CPT (s) specified in table III.1.22 at current time point  $t$  are essentially prior probabilities which in turn are posterior probabilities at previous time point  $t-1$  calculated based on formula III.1.48 in step 6. So prior probabilities  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  are posterior probabilities of  $x_1$ ,  $x_2$ , and  $x_3$ , respectively computed at previous iteration. Evaluation of  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  will be illustrated in step 6.

2. *Re-defining CPT(s) of  $X[t]$ .* Suppose  $pa_i[t] = \{y_{i1}, y_{i2}, \dots, y_{ik}\}$  is a set of parents of  $x_i[t]$  at current time point  $t$  and  $w_i[t] = \{w_{i1}, w_{i2}, \dots, w_{ik}\}$  is a set of normalized weights which expresses the strength of relationships between  $x_i$  and such  $pa_i[t]$  where  $pa_i[t] \subseteq X[t-1] \cup X[t]$ . Note that  $w_i[t]$  is specified in step 4. The conditional probability of variable  $x_i[t]$  given its parents  $pa_i[t]$  is denoted  $P(x_i[t] | pa_i[t])$ . So  $P(x_i[t] | pa_i[t])$  represents the CPT of  $x_i[t]$ . Suppose relationships between  $x_i$  and its  $pa_i[t]$  are interpreted as aggregation relationships so that  $x_i$  and its  $pa_i[t]$  constitute a sigma graph (see sub-section III.1.1.3). Applying theorem of SIGMA-gate inference specified in formula III.1.11, we have formula III.1.47 for computing the condition probability  $P(x_i[t] | pa_i[t])$  as follows:

$$P(x_i[t] | pa_i[t]) = \sum_{j=1}^k h_{ij} w_{ij} \text{ where } h_{ij} = \begin{cases} 1 & \text{if } y_{ij} = x_i[t] \\ 0 & \text{else} \end{cases}$$

$$P(\text{not } x_i[t] | pa_i[t]) = 1 - P(x_i[t] | pa_i[t])$$

*Formula III.1.47.* Conditional probability of each variable  $x_i[t]$  at current time point  $t$

Going back our example of DBN (whose weights are normalized) at current time point  $t$  shown in figure III.1.25, for instance, the variable  $x_1[t]$  has three parent variables  $x_1[t-1]$ ,  $x_2[t]$  and  $x_3[t]$  and so we have  $pa_1[t] = \{x_1[t-1], x_2[t], x_3[t]\}$ . Applying formula III.1.47, the conditional probability of  $x_1[t]=1$  ( $x_1[t]=0$ ) given  $x_1[t-1]=1$ ,  $x_2[t]=1$  and  $x_3[t]=1$  is:

$$\begin{aligned} P(x_1[t] = 1 | x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\ = 1 * 0.58 + 1 * 0.252 + 1 * 0.168 = 1 \end{aligned}$$

### III.1. Knowledge sub-model

---

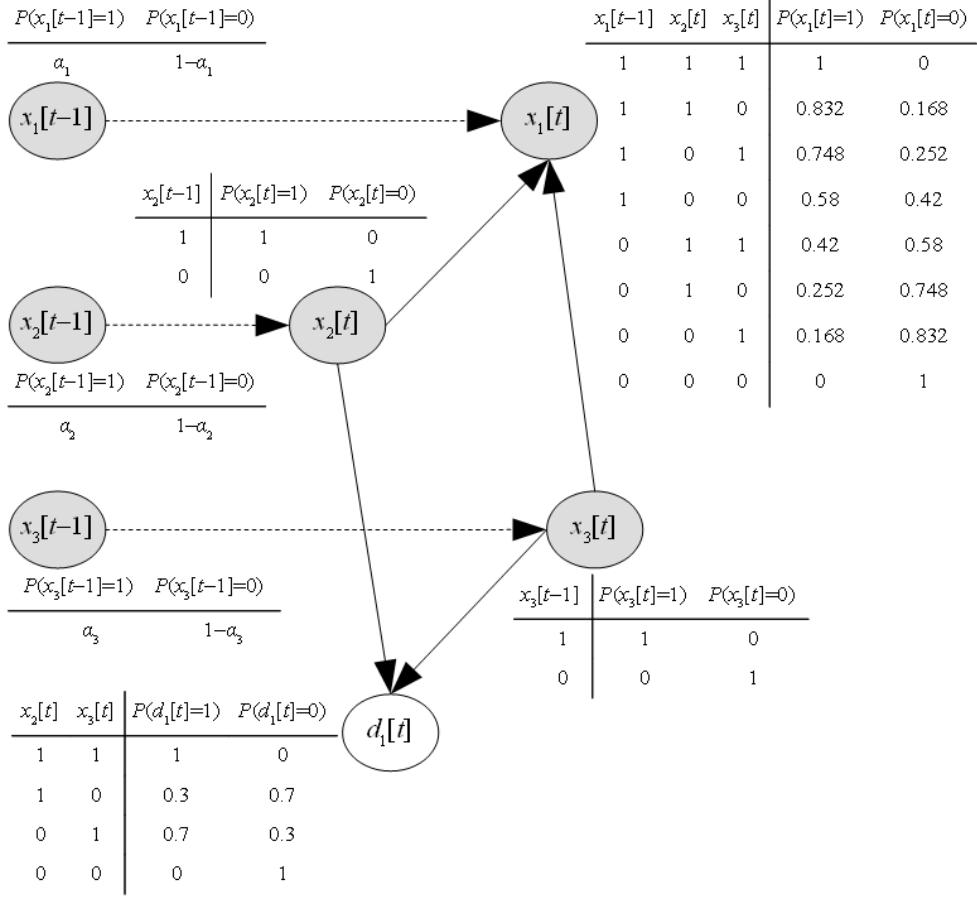
$$\begin{aligned}
P(x_1[t] = 0 | x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
= 1 - P(x_1[t] = 1 | x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
= 0
\end{aligned}$$

Similarly, it is easy to calculate conditional probabilities of  $x_2[t]$  and  $x_3[t]$ . Table III.1.23 shows CPT (s) of  $X[t] = \{x_1[t], x_2[t], x_3[t], d_1[t]\}$ .

$P(x_1[t]=1   x_1[t-1]=1, x_2[t]=1, x_3[t]=1) = 1*0.58+1*0.252+1*0.168 = 1$
$P(x_1[t]=1   x_1[t-1]=1, x_2[t]=1, x_3[t]=0) = 1*0.58+1*0.252+0*0.168 = 0.832$
$P(x_1[t]=1   x_1[t-1]=1, x_2[t]=0, x_3[t]=1) = 1*0.58+0*0.252+1*0.168 = 0.748$
$P(x_1[t]=1   x_1[t-1]=1, x_2[t]=0, x_3[t]=0) = 1*0.58+0*0.252+0*0.168 = 0.58$
$P(x_1[t]=1   x_1[t-1]=0, x_2[t]=1, x_3[t]=1) = 0*0.58+1*0.252+1*0.168 = 0.42$
$P(x_1[t]=1   x_1[t-1]=0, x_2[t]=1, x_3[t]=0) = 0*0.58+1*0.252+0*0.168 = 0.252$
$P(x_1[t]=1   x_1[t-1]=0, x_2[t]=0, x_3[t]=1) = 0*0.58+0*0.252+1*0.168 = 0.168$
$P(x_1[t]=1   x_1[t-1]=0, x_2[t]=0, x_3[t]=0) = 0*0.58+0*0.252+0*0.168 = 0$
$P(x_1[t]=0   x_1[t-1]=1, x_2[t]=1, x_3[t]=1) = 0$
$P(x_1[t]=0   x_1[t-1]=1, x_2[t]=1, x_3[t]=0) = 0.168$
$P(x_1[t]=0   x_1[t-1]=1, x_2[t]=0, x_3[t]=1) = 0.252$
$P(x_1[t]=0   x_1[t-1]=1, x_2[t]=0, x_3[t]=0) = 0.42$
$P(x_1[t]=0   x_1[t-1]=0, x_2[t]=1, x_3[t]=1) = 0.58$
$P(x_1[t]=0   x_1[t-1]=0, x_2[t]=1, x_3[t]=0) = 0.748$
$P(x_1[t]=0   x_1[t-1]=0, x_2[t]=0, x_3[t]=1) = 0.832$
$P(x_1[t]=0   x_1[t-1]=0, x_2[t]=0, x_3[t]=0) = 1$
$P(x_2[t]=1   x_2[t-1]=1) = 1*1 = 1$
$P(x_2[t]=1   x_2[t-1]=0) = 0*0 = 0$
$P(x_2[t]=0   x_2[t-1]=1) = 0$
$P(x_2[t]=0   x_2[t-1]=0) = 1$
$P(x_3[t]=1   x_3[t-1]=1) = 1*1 = 1$
$P(x_3[t]=1   x_3[t-1]=0) = 0*0 = 0$
$P(x_3[t]=0   x_3[t-1]=1) = 0$
$P(x_3[t]=0   x_3[t-1]=0) = 1$
$P(d_1[t]=1   x_2[t]=1, x_3[t]=1) = 1*0.3+1*0.7 = 1$
$P(d_1[t]=1   x_2[t]=1, x_3[t]=0) = 1*0.3+0*0.7 = 0.3$
$P(d_1[t]=1   x_2[t]=0, x_3[t]=1) = 0*0.3+1*0.7 = 0.7$
$P(d_1[t]=1   x_2[t]=0, x_3[t]=0) = 0*0.3+0*0.7 = 0$
$P(d_1[t]=0   x_2[t]=1, x_3[t]=1) = 0$
$P(d_1[t]=0   x_2[t]=1, x_3[t]=0) = 0.7$
$P(d_1[t]=0   x_2[t]=0, x_3[t]=1) = 0.3$
$P(d_1[t]=0   x_2[t]=0, x_3[t]=0) = 1$

**Table III.1.23.** CPT (s) of  $X[t] = \{x_1[t], x_2[t], x_3[t], d_1[t]\}$

When CPT (s) of  $X[t-1]$  and  $X[t]$  are determined, the DBN at current time point  $t$  shown in figure III.1.25 is totally determined and so, figure III.1.26 shows the DBN or expended BN at time point t with full of CPT (s).



**Figure III.1.26.** DBN at time point  $t$  with full of CPT (s)

As seen in figure III.1.26, the CPT (s) of  $X[t-1] = \{x_1[t-1], x_2[t-1], x_3[t-1]\}$  at current time point  $t$  includes posterior probabilities  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  of  $x_1[t-1]$ ,  $x_2[t-1]$ , and  $x_3[t-1]$  at previous time point  $t-1$ . The last step of proposed algorithm (step 6) describes how to calculate these posterior probabilities based on probabilistic inference inside BN.

### Step 6: Probabilistic inference

The probabilistic inference in our DBN at current time point  $t$  (extended BN, please see steps 3 and 4) can be done similarly as we do in normal BN by using the formula III.1.3'. It is essential to compute the posterior probabilities of non-evidence variable in  $X[t]$ . This decrease significantly expense of computation regardless of a large number of variables in DBN for a long time. At any time point, it is only to examine  $2*n$  variables if the DAG has  $n$  variables instead of including  $2*n*t$  variables and  $n*n*t$  transition probabilities given time point  $t$ . Given  $\mathcal{D}[t] = \{d_1[t], d_2[t], \dots, d_k[t]\}$  is the set of evidences at current time point  $t$ , each posterior probability of  $x_i[t] \in X[t]$  is computed by formula III.1.48 as follows:

$$P(x_i[t]|\mathcal{D}[t]) = \frac{\sum_{X[t] \setminus \{x_i[t]\} \cup \mathcal{D}[t]} P(x_1[t], x_2[t], \dots, x_n[t])}{\sum_{X[t] \setminus \mathcal{D}[t]} P(x_1[t], x_2[t], \dots, x_n[t])}$$

*Formula III.1.48.* Posterior probability of each  $x_i[t]$  given evidences  $\mathcal{D}[t]$

Where  $X[t] \setminus (\{x_i[t]\} \cup \mathcal{D}[t])$  and  $X[t] \setminus \mathcal{D}[t]$  are all possible values  $X[t] = (x_1[t], x_2[t], \dots, x_n[t])$  with fixing  $\{x_i[t]\} \cup \mathcal{D}[t]$  and fixing  $\mathcal{D}[t]$ , respectively. Note that the sign “\” denotes the subtraction (excluding) in set theory (Wikipedia, Set (mathematics), 2014).

*Such posterior probabilities are used for determining CPT (s) of DBN in step 5 of next iteration (time point  $t+1$ ).* Please pay attention to the reciprocal relationship between step 5 and step 6 when such reciprocal relationship based on these posterior probabilities and transition weight specified in step 2 are two crucial problems of the proposed algorithm. Back to our example of DBN shown in figure III.1.26, posterior probabilities of  $x_1[t]$ ,  $x_2[t]$  and  $x_3[t]$  are  $\alpha_1$ ,  $\alpha_2$  and  $\alpha_3$ , respectively, which are mentioned in step 5 and shown in table III.1.22. If the posterior probabilities are the same as before (previous iteration) then DBN converges when all posterior probabilities of variables  $x_i[t]$  gain stable values at any time. If so we can stop algorithm; otherwise turning back step 1.

Going back our example of DBN shown in figure III.1.26, given evidence sample  $\mathcal{D}[t] = \{x_1[t-1]=1, x_2[t-1]=1, x_3[t-1]=1, d_1[t]=1\}$ , it is required to compute posterior probabilities  $\alpha_1=P(x_1[t]=1|\mathcal{D}[t])$ ,  $\alpha_2=P(x_2[t]=1|\mathcal{D}[t])$  and  $\alpha_3=P(x_3[t]=1|\mathcal{D}[t])$ . Using 4-evidence sample  $\mathcal{D}[t]$  is convenient for illustration when  $x_1[t-1]$ ,  $x_2[t-1]$ , and  $x_3[t-1]$  are not real evidences and so we should use  $d_1[t]$  as evidence in realistic implementation.

By applying formula III.1.2', we have global joint probability distribution as follows:

$$\begin{aligned} P(x_1[t-1], x_2[t-1], x_3[t-1], x_1[t], x_2[t], x_3[t], d_1[t]) \\ = P(x_1[t-1]) * P(x_2[t-1]) * P(x_3[t-1]) \\ * P(x_2[t]|x_2[t-1]) * P(x_3[t]|x_3[t-1]) \\ * P(x_1[t]|x_1[t-1], x_2[t], x_3[t]) * P(d_1[t]|x_2[t], x_3[t]) \end{aligned}$$

Suppose  $P(x_1[t-1]=1) = P(x_2[t-1]=1) = P(x_3[t-1]=1) = 0.5$ , the global joint probability distribution becomes:

$$\begin{aligned} P(x_1[t], x_2[t], x_3[t], d_1[t]) \\ = 0.125 * P(x_2[t]|x_2[t-1]=1) * P(x_3[t]|x_3[t-1]=1) \\ * P(x_1[t]|x_1[t-1]=1, x_2[t], x_3[t]) * P(d_1[t]=1|x_2[t], x_3[t]) \end{aligned}$$

Note that probabilities  $P(x_1[t-1]=1)$ ,  $P(x_2[t-1]=1)$ , and  $P(x_3[t-1]=1)$  are always determined because they are also essentially posterior probabilities  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  calculated at step 6 in previous time point  $t-1$ .

The posterior probability of evidence sample  $\mathcal{D}[t] = \{d_1[t]=1\}$  so-called marginal probability is:

$$\begin{aligned} P(\mathcal{D}[t]) \\ = P(x_1[t]=1, x_2[t]=1, x_3[t]=1, d_1[t]=1) \\ + P(x_1[t]=1, x_2[t]=1, x_3[t]=0, d_1[t]=1) \\ + P(x_1[t]=1, x_2[t]=0, x_3[t]=1, d_1[t]=1) \\ + P(x_1[t]=1, x_2[t]=0, x_3[t]=0, d_1[t]=1) \end{aligned}$$

$$\begin{aligned}
 & +P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 0, x_3[t] = 0, d_1[t] = 1) \\
 & = 0.125 * 1 * 1 * 1 * 1 \\
 & +0.125 * 1 * 0 * 0.832 * 0.3 \\
 & +0.125 * 0 * 1 * 0.748 * 0.7 \\
 & +0.125 * 0 * 0 * 0.58 * 0 \\
 & +0.125 * 1 * 1 * 0 * 1 \\
 & +0.125 * 1 * 0 * 0.168 * 0.3 \\
 & +0.125 * 0 * 1 * 0.252 * 0.7 \\
 & +0.125 * 0 * 0 * 0.42 * 0 \\
 & = 0.125 + 0 + 0 + 0 + 0 + 0 + 0 = 0.125
 \end{aligned}$$

We also have:

$$\begin{aligned}
 & P(x_1[t] = 1, \mathcal{D}[t]) \\
 & = P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\
 & +P(x_1[t] = 1, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 1, x_2[t] = 0, x_3[t] = 0, d_1[t] = 1) \\
 & = 0.125 + 0 + 0 + 0 = 0.125
 \end{aligned}$$

$$\begin{aligned}
 & P(x_2[t] = 1, \mathcal{D}[t]) \\
 & = P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\
 & = 0.125 + 0 + 0 + 0 = 0.125
 \end{aligned}$$

$$\begin{aligned}
 & P(x_3[t] = 1, \mathcal{D}[t]) \\
 & = P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 1, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\
 & = 0.125 + 0 + 0 + 0 = 0.125
 \end{aligned}$$

Now it is easy to compute posterior probabilities  $\alpha_1 = P(x_1[t]=1|\mathcal{D}[t])$ ,  $\alpha_2 = P(x_2[t]=1|\mathcal{D}[t])$ , and  $\alpha_3 = P(x_3[t]=1|\mathcal{D}[t])$ .

$$\begin{aligned}
 \alpha_1 &= P(x_1[t] = 1|\mathcal{D}[t]) = \frac{P(x_1[t] = 1, \mathcal{D}[t])}{P(\mathcal{D}[t])} = \frac{0.125}{0.125} = 1 \\
 \alpha_2 &= P(x_2[t] = 1|\mathcal{D}[t]) = \frac{P(x_2[t] = 1, \mathcal{D}[t])}{P(\mathcal{D}[t])} = \frac{0.125}{0.125} = 1 \\
 \alpha_3 &= P(x_3[t] = 1|\mathcal{D}[t]) = \frac{P(x_3[t] = 1, \mathcal{D}[t])}{P(\mathcal{D}[t])} = \frac{0.125}{0.125} = 1
 \end{aligned}$$

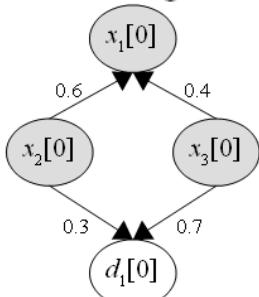
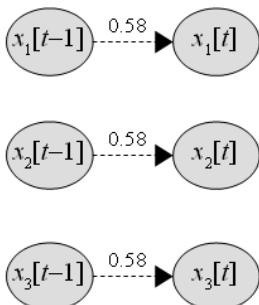
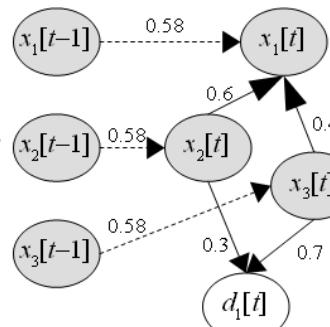
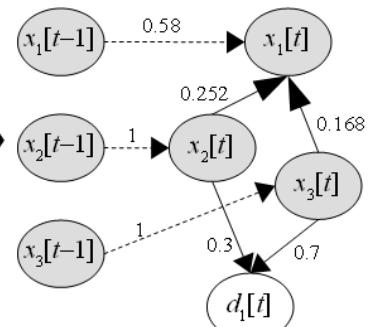
That  $\alpha_1=1$ ,  $\alpha_2=1$ , and  $\alpha_3=1$  indicates that user mastered all knowledge items (hypothesis variables)  $x_1$ ,  $x_2$  and  $x_3$  at time point  $t$ . Table III.1.24 summarizes

the posterior probabilities  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  which are the results of probabilistic inference, used as prior probabilities (CPT (s)) of DBN in step 5 of next iteration.

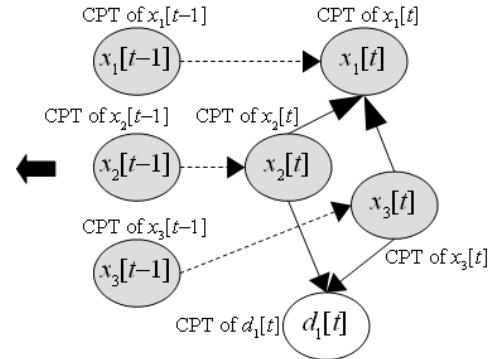
$\alpha_1 = P(x_1[t]=1   \mathcal{D}[t]) = 1$
$\alpha_2 = P(x_2[t]=1   \mathcal{D}[t]) = 1$
$\alpha_3 = P(x_3[t]=1   \mathcal{D}[t]) = 1$

**Table III.1.24.** The results of probabilistic inference – posterior probabilities

In general, the proposed algorithm to construct DBN based on Markov property and stationary property is iterative process, which has many iterations. Each iteration includes six aforementioned steps creating a closed cycle in which the prior probabilities at previous iteration (step 6) become posterior probabilities at current iteration (step 5) so that the DBN (expended BN) is re-constructed (step 3) and improved continuously. The closed cycle is executed continuously whenever evidences occur and it only stops when DBN converges, at that time, the posterior probabilities gain stable values and structure of DBN is stable too. Figure III.1.27 shows such cycle.

**1. Initializing DBN**

**2. Specifying transition weights**

**3. Re-constructing**

**4. Normalizing weights**


$$\begin{aligned} a_1 &= P(x_1[t]=1 | D[t]) = 1 \\ a_2 &= P(x_2[t]=1 | D[t]) = 1 \\ a_3 &= P(x_3[t]=1 | D[t]) = 1 \end{aligned}$$

**6. Probabilistic inference**

**5. Re-defining CPT (s)**

**Figure III.1.27.** The cycle of proposed algorithm to construct DBN

### III.1.4.3. Evaluation

My basic idea is to minimize the size of DBN and the number of transition probabilities in order to decrease expense of computation when the process of inference continues for a long time. Suppose DBN is stationary and has Markov property, I define two factors: *slip* and *guess* to specify the same weight for all transition relationships (temporal relationship) among time points instead of specifying a large number of transition probabilities. The augmented DBN composed at given time point  $t$  has just two random vectors  $X[t-1]$  and  $X[t]$ ; so , it is only to examine  $2*n$  variables if the DAG has  $n$  variables instead of including  $2*n*t$  variables and  $n*n*t$  transition probabilities. That specifying

*slip* factor and *guess* factor will solve the problem of temporary slip and lucky guess.

The process of inference including six steps is done in succession through many iterations, the result of current iteration will be input for next iteration. After  $t^{\text{th}}$  iteration DBN will converge when the posterior probabilities of all variables  $x_i[t]$  gain stable values regardless of the occurrence of a variety of evidences.

Instead of using DBN to re-construct network, another approach described in next sub-section [III.1.5](#) is applied into improving the quality of inference mechanism in knowledge sub-model. This approach is to analyze training data so as to determine the prior probabilities of nodes in network as precisely as possible. In other words, network structure is not modified and only conditional probability tables (CPT) are improved. The new approach is introduced in next sub-section [III.1.5](#) – specifying prior probabilities of [Bayesian overlay model](#). Both such approach and the method “how to make evolution of Bayesian overlay model” described in previous sub-sections [III.1.3.1](#) and [III.1.3.2](#) are parameter learning techniques (opposite to structure learning technique based on DBN mentioned in this sub-section [III.1.4](#)) but their difference will be made clear. Please read next sub-section [III.1.5](#) in order to release your curiousness.

#### **III.1.5. Specifying prior probabilities**

Bayesian network (BN) provides the solid inference mechanism when convincing the hypothesis by collecting evidences. BN is instituted of two models such as qualitative model quantitative model. The qualitative model is its structure and the quantitative model is its parameters, namely conditional probability tables (CPT) whose entries are probabilities quantifying relationships among variables in network; please see previous sub-section [III.1.1.1](#) for more details about BN and CPT. The quality of CPT depends on the initialized values of its entries. Such initial values are prior probabilities. Because the beta function provides some conveniences when specifying CPT (s), this function is used as the basic distribution in my method. The main problem of defining prior probabilities is how to estimate parameters in beta distribution. It is slightly unfortunate when the equations whose solutions are parameter estimators are differential equations and it is too difficult to solve them. By applying the Maximum Likelihood Estimation (MLE) technique, I invent the simple equations so that differential equations are eliminated and it is much easier to estimate parameters in case that such parameters are positive integer numbers (Nguyen, Specifying Prior Probabilities in Bayesian Network by Maximum Likelihood Estimation method, 2014). Thus, I also propose the algorithm to find out the approximate solutions of these simple equations. Recall that there are two ways to improve BN such as parameter learning so-called evolution of [Bayesian overlay model](#) which was mentioned in previous sub-section [III.1.3](#) and structure learning based on dynamic Bayesian network which was described in previous sub-section [III.1.4](#). The proposed algorithm specifying prior probabilities introduced in this sub-section [III.1.5](#) is essentially

a parameter learning technique like evolution of Bayesian network mentioned in previous sub-section [III.1.3](#) but their difference is explained as follows:

- *Specifying prior probabilities* is to construct parameters (CPT (s)) of BN based on training data or data sample when BN has not CPT (s) yet.
- *Evolution of Bayesian network* is to improve or update CPT (s) when BN has already CPT (s), which means that specifying prior probabilities is always done before evolution of Bayesian network.

Recall that Bayesian network (BN) is the directed acyclic graph (DAG) constituted of a set of nodes representing random variables and a set of directed arcs representing relationships among nodes; please see sub-section [III.1.1.1](#) for more details about BN. The strengths of relationship are quantified by conditional probabilities. Each node owns a conditional probability table (CPT) that measures the impact of all its parents on it. Such CPT (s) are called the parameters of BN. Note that each entry in a CPT is a conditional probability. The problem which must be solved is how to initialize these parameters so as to be optimal. It means that we should specify prior probability.

Every node  $X$  in BN is a binary random variable. As aforementioned in sub-section [III.1.3.1](#), each variable  $X$  is attached by a dummy variable  $F$  so that the probability density function (PDF) of such variable  $F$  represents CPT of  $X$ . Please see sub-section [III.1.1.1](#) for more details about CPT and PDF. The PDF of  $F$  conforms beta density function  $\beta(F; a, b)$  where  $a$  and  $b$  are two parameters. In other words,  $F$  has beta density function  $\beta(F; a, b)$  which is expressed by formula [III.1.49](#).

$$\beta(F; a, b) = \text{beta}(F; a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} F^{a-1} (1 - F)^{b-1}$$

*Formula III.1.49. Beta density function  $\beta(F; a, b)$*

Where  $\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$  denotes gamma function which is expressed by formula [III.1.13](#). Please see sub-section [III.1.3.1](#) for more details about dummy variable  $F$  and beta density function. Because it is convenient for readers to keep track main content in this sub-section [III.1.5](#), formula [III.1.49](#) is replication of formula [III.1.12](#) for specifying beta density function. The expectation  $E(F)$  and the variance  $Var(F)$  of dummy variable  $F$  are:

$$E(F) = \frac{a}{a + b} \text{ and } Var(F) = \frac{ab}{(a + b)^2(a + b + 1)}$$

The reason we choose beta density function as the probability distribution attached to every variable  $X$  in BN is that the prior probability of variable  $X$  is the expectation of  $F$  and it is very easy to compute this value as follows:

$$P(X = 1) = E(F) = E(\beta(a, b)) = \frac{a}{a + b} = \frac{a}{N} \\ (N = a + b)$$

*Formula III.1.50. Probability of variable  $X$  as expectation of beta variable  $F$*

Because it is convenient for readers to keep track main content in this sub-section III.1.5, formula III.1.50 is the same to formula III.1.17. Note that the equation  $E(F) = E(\beta(a, b))$  implicates that dummy variable  $F$  is identified with its beta distribution  $\beta(a, b)$ . We need to compute the posterior probability of variable  $X$  denoted as  $P(X=1|\mathcal{D})$  where  $\mathcal{D}$  is the set of evidences in which the number of evidences having value 1 is  $s$  and the number of evidences having value 0 is  $t$ . Formula III.1.51 specifies posterior probability of  $X$  as conditional expectation of beta variable  $F$ .

$$P(X = 1|\mathcal{D}) = E(F|\mathcal{D}) = E(\beta(a, b)|\mathcal{D}) = \frac{a + s}{a + b + s + t} = \frac{a + s}{N + M}$$

*Formula III.1.51.* Posterior probability of variable  $X$  as conditional expectation of beta variable  $F$

Where  $N=a+b$  and  $M=s+t$ .

The reason that formula III.1.51 is the replication of formula III.1.21 is to give readers convenience for keeping track main content in this sub-section III.1.5. Note that the equation  $E(F|\mathcal{D}) = E(\beta(a, b)|\mathcal{D})$  implicates that dummy variable  $F$  is identified with its beta distribution  $\beta(a, b)$ .

I recognize that beta distribution provides us convenience when specifying CPT (s) in BN. It is essential to count the number of evidences so as to compute the posterior probabilities. However, the quality of CPT is also dependent on the prior probability and so; the considerable problem is involved in how to estimate two parameters of beta distribution  $a$  and  $b$  because the prior probability is derived from them,  $P(X = 1) = E(F) = \frac{a}{a+b}$ .

Sub-section III.1.5.1 discusses some basic concepts of maximum likelihood estimation (MLE) technique. Sub-section III.1.5.2 discusses applying MLE into beta distribution (beta density function). Sub-section III.1.5.3 – the main sub-section describes my proposed algorithm to estimate two parameters  $a$  and  $b$  of beta distribution. Sub-section III.1.5.3 also proposes the simple equations whose solutions are estimates of positive parameters  $a$  and  $b$ . Sub-section III.1.5.4 illustrates the proposed algorithm by example. Sub-section III.1.5.5 introduces a new version of simple equations mentioned in sub-section III.1.5.4. Sub-section III.1.5.6 is evaluation of the algorithm proposed in the main sub-section III.1.5.3.

### III.1.5.1. Maximum likelihood estimation

Let  $\Theta$  and  $X$  be the hypothesis and observation variable, respectively. Suppose  $x_1, x_2, \dots, x_n$  are instances of variable  $X$  in training data and they are observed independently. In study of statistics, training data is called sample which is constituted of these observations or evidences  $x_i$  (s) and such  $x_i$  (s) are considered as independent and identically distributed (i.i.d) random variables; which means that  $x_i$  (s) and  $X$  have the same probability distribution or the same probability density function (PDF). Please see formula III.1.1e for more details about PDF and probability distribution. According multiplication rule

(see formula III.1.1c) in probability theory, the likelihood function  $L(\Theta)$  is the joint probability which is the product of condition probabilities of instances  $x_i$ , given hypothesis variable  $\Theta$  (Lynch, 2007, p. 36). Formula III.1.52 expresses the likelihood function  $L(\Theta)$  with regard to variable  $\Theta$ .

$$L(\Theta) = P(X | \Theta) = \prod_{i=1}^n P(x_i | \Theta)$$

*Formula III.1.52. Likelihood function*

Where  $P(x_i | \Theta)$  is the conditional probability of instance  $x_i$  given the hypothesis  $\Theta$ . Suppose  $\Theta = \{\theta_1, \theta_2, \dots, \theta_k\}$  is the vector of parameters specifying the distribution of  $X$  (density function of  $X$ ) denoted  $f$ , it is required to estimate the parameter vector and its standard deviation in distribution  $f$  so that the likelihood function takes the maximum value. Thus, this method is called maximum likelihood estimation (MLE). The parameter vector that maximizes likelihood function is called *optimal parameter vector* or *parameter vector estimator* denoted  $\widehat{\Theta}$ , as shown in formula III.1.53. If  $\widehat{\Theta}$  was evaluated, it can be considered *parameter vector estimate*. It is possible to use terms such as “optimal parameter vector”, “parameter vector estimator”, and “parameter vector estimate” in exchangeable. We can remove the word “vector” inside these terms if we do not focus on the fact that  $\widehat{\Theta}$  is a vector; in other words,  $\widehat{\Theta}$  can be called as optimal parameter, parameter estimator, and parameter estimate.

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}} L(\Theta) = \underset{\Theta}{\operatorname{argmax}} \left( \prod_{i=1}^n P(x_i | \Theta) \right)$$

*Formula III.1.53. Optimal parameter vector*

Because it is too difficult to work with the likelihood function in the form of product of condition probabilities, it is necessary to take logarithm of  $L(\Theta)$  so as to transform the likelihood function from form of repeated multiplication like formula III.1.52 into form of repeated addition like formula III.1.54 (Lynch, 2007, p. 38). The natural logarithm of  $L(\Theta)$  so-called log-likelihood is denoted  $LnL(\Theta)$ , as shown in formula III.1.54.

$$\begin{aligned} LnL(\Theta) &= \ln \left( \prod_{i=1}^n P(x_i | \Theta) \right) = \sum_{i=1}^n \ln(P(x_i | \Theta)) \\ \widehat{\Theta} &= \underset{\Theta}{\operatorname{argmax}} LnL(\Theta) = \underset{\Theta}{\operatorname{argmax}} \left( \sum_{i=1}^n \ln(P(x_i | \Theta)) \right) \end{aligned}$$

*Formula III.1.54. Log-likelihood function and optimal parameter vector*

Where  $\ln(\cdot)$  denotes natural logarithm function.

The essence of maximizing the likelihood function is to find the peak of the curve of  $LnL(\Theta)$  (Lynch, 2007, p. 38). This can be done by setting the first-order partial derivative of  $LnL(\Theta)$  with respect to each parameter  $\theta_i$  to 0 and solving this equation to find out parameter  $\theta_i$ . The number of equations corresponds with the number of parameters. If all parameters are found, in other words, the optimal parameter vector  $\widehat{\Theta} = \{\widehat{\theta}_1, \widehat{\theta}_2, \dots, \widehat{\theta}_k\}$  is defined then the optimal distribution  $f(\widehat{\Theta})$  is known clearly. Each  $\widehat{\theta}_i$  is also called a *parameter estimator*; on the other hand  $\widehat{\theta}_i$  can be considered *parameter estimate* or *optimal parameter* if it is evaluated as numeric value. It is possible to use terms such as “optimal parameter”, “parameter estimator”, and “parameter estimate” in exchangeable.

The accuracy of parameter estimator is measured by its standard error (Montgomery & Runger, 2003, p. 225) and thus; another important issue is how to determine the standard error in distribution  $f$  when we have already computed all parameters and standard error is standard deviation of parameter estimator. It is very fortunate when the second-order derivative of the log-likelihood function denoted  $\frac{\partial^2 LnL}{\partial \Theta \partial \Theta^T}$  can be computed and it is used to determine the variances of parameters. If distribution  $f$  has only one parameter, the second-order derivative  $\frac{\partial^2 LnL}{\partial \Theta \partial \Theta^T}$  is scalar, otherwise it is a matrix so-called Hessian matrix. The negative expectation of Hessian matrix is called the *information matrix* which in turn is inverted so as to construct *co-variance matrix* denoted  $Var(\Theta)$  (Lynch, 2007, p. 40). Formula III.1.55 specifies the covariance matrix of parameter vector  $\Theta$ .

$$Var(\Theta) = \left( -E \left( \frac{\partial^2 LnL}{\partial \Theta \partial \Theta^T} \right) \right)^{-1}$$

*Formula III.1.55. Co-variance matrix of parameter vector*

Elements on co-variance matrix diagonal are variances of the parameters and the square root of each variance is a standard error. Note that  $\left( -E \left( \frac{\partial^2 LnL}{\partial \Theta \partial \Theta^T} \right) \right)^{-1}$  is exactly so-called Cramer-Rao lower bound of co-variance matrix but we can consider approximately  $\left( -E \left( \frac{\partial^2 LnL}{\partial \Theta \partial \Theta^T} \right) \right)^{-1}$  as co-variance matrix when  $\widehat{\Theta}$  is derived from likelihood function and  $\widehat{\Theta}$  is unbiased estimator (Zivot, 2009, p. 11). Please read (Lynch, 2007, pp. 35-43) and (Zivot, 2009) for more details about MLE.

Next sub-section III.1.5.2 – “Beta likelihood estimation” discusses how to apply MLE into beta distribution (beta density function).

### III.1.5.2. Beta likelihood estimation

As discussed, each variable  $X$  in BN is attached by a dummy variable  $F$  and variable  $F$ , in turn, has beta distribution  $\beta(F; a, b)$  specified in formula III.1.49. For mathematical notations, we often use letters such as “ $X$ ”, “ $Y$ ”, “ $Z$ ”, “ $x$ ”, “ $y$ ”, and “ $z$ ” to denote random variable and it is very rare to use letter “ $F$ ” to denote variable as we used to do in this research. Therefore, your attention please, we will use letter “ $X$ ” to denote variable conforming the beta distribution instead of using letter “ $F$ ” as usual. Variable  $X$  mentioned in this sub-section III.1.5.2 does not implicate to random variable (node) in BN and it really indicate to dummy variable  $F$  although you can identify random variable (node) in BN with dummy variable  $F$  via high level point of view. So the beta density function specified by previous formula III.1.49 is re-written as follows:

$$f(X; a, b) = \beta(X; a, b) = \text{beta}(X; a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} X^{a-1} (1 - X)^{b-1}$$

*Formula III.1.56.* Beta density function (beta distribution)  $\beta(X; a, b)$

Where  $\Gamma(\cdot)$  denotes gamma function which is expressed by formula III.1.13.

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

Note that  $e^{(\cdot)}$  and  $\exp(\cdot)$  denote exponent function and  $e \approx 2.71828$  is Euler's number.

Beta density function is based on gamma function and there is another so-called *digamma function* is also defined via gamma function. Formula III.1.57 is definition of digamma function  $\psi(x)$ . We will know later that beta density function is also relevant to digamma function.

$$\psi(x) = d \left( \ln(\Gamma(x)) \right) = \frac{\Gamma'(x)}{\Gamma(x)}$$

*Formula III.1.57.* Definition of digamma function

Note that  $\ln(\cdot)$  denotes natural logarithm function. According to formula III.1.57, digamma function is the derivative of natural logarithm of gamma function.

The integral form of digamma function is specified by formula III.1.58 (Medina & Moll, 2009, p. 114):

$$\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)} = \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-xt}}{1 - e^{-t}} \right) dt$$

*Formula III.1.58.* Integral form of digamma function

Suppose variable  $x$  is non-zero, we have:

$$\begin{aligned}
 \psi(x+1) &= \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-(x+1)t}}{1-e^{-t}} \right) dt = \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}} + e^{-xt} \right) dt \\
 &= \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}} \right) dt + \int_0^{+\infty} e^{-xt} dt = \psi(x) - \frac{1}{x} e^{-xt} \Big|_0^{+\infty} \\
 &= \psi(x) + \frac{1}{x}
 \end{aligned}$$

Briefly, the recurrence formula of digamma function is specified by formula III.1.59.

$$\psi(x+1) = \psi(x) + \frac{1}{x}$$

*Formula III.1.59.* Recurrence formula of digamma function

Formula III.1.59 shows recurrence relation (Wikipedia, Digamma function, 2014) of digamma function, which implicates that it is very easy to compute  $\psi(x)$  if variable  $x$  is positive integer. Thus, it is necessary to calculate the starting positive value  $\psi(1)$ , we have:

$$\begin{aligned}
 \psi(1) &= \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-t}}{1-e^{-t}} \right) dt = \int_0^{+\infty} \frac{e^{-t}}{t} dt - \int_0^{+\infty} \frac{e^{-t}}{1-e^{-t}} dt \\
 &= \int_0^{+\infty} e^{-t} d(\ln t) - \int_0^{+\infty} d(\ln(1-e^{-t})) \\
 &= e^{-t} \ln t \Big|_0^{+\infty} + \int_0^{+\infty} e^{-t} \ln t dt - \ln(1-e^{-t}) \Big|_0^{+\infty} \\
 &= \lim_{t \rightarrow +\infty} (e^{-t} \ln t) - \lim_{t \rightarrow 0} (e^{-t} \ln t) - \gamma - \lim_{t \rightarrow +\infty} \ln(1-e^{-t}) + \lim_{t \rightarrow 0} \ln(1-e^{-t}) \\
 &\quad (\text{due to Euler-Mascheroni constant } \gamma = -\int_0^{+\infty} e^{-t} \ln t dt) \\
 &= -\gamma + \lim_{t \rightarrow +\infty} (e^{-t} \ln t) - \lim_{t \rightarrow 0} (e^{-t} \ln t) + \lim_{t \rightarrow 0} \ln(1-e^{-t}) \\
 &= -\gamma + \lim_{t \rightarrow +\infty} \frac{\ln t}{e^t} + \lim_{t \rightarrow 0} (\ln(1-e^{-t}) - e^{-t} \ln t) \\
 &= -\gamma + \lim_{t \rightarrow +\infty} \frac{\ln t}{e^t} + \lim_{t \rightarrow 0} \ln \frac{e^{\ln(1-e^{-t})}}{e^{e^{-t} \ln t}}
 \end{aligned}$$

(due to transformation with regard to indeterminate form (Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}
 &= -\gamma + \lim_{t \rightarrow +\infty} \frac{\ln t}{e^t} + \lim_{t \rightarrow 0} \ln \frac{1-e^{-t}}{t^{e^{-t}}} \\
 &= -\gamma + \lim_{t \rightarrow +\infty} \frac{\ln t}{e^t} + \ln \left( \lim_{t \rightarrow 0} \frac{1-e^{-t}}{t^{e^{-t}}} \right) \\
 &= -\gamma + \lim_{t \rightarrow +\infty} \frac{d(\ln t)}{d(e^t)} + \ln \left( \lim_{t \rightarrow 0} \frac{1-e^{-t}}{t^{e^{-t}}} \right)
 \end{aligned}$$

$$\begin{aligned}
 & \text{(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))} \\
 & = -\gamma + \lim_{t \rightarrow +\infty} \frac{1}{te^t} + \ln \left( \lim_{t \rightarrow 0} \frac{1 - e^{-t}}{t^{e^{-t}}} \right) \\
 & = -\gamma + \ln \left( \lim_{t \rightarrow 0} \frac{1 - e^{-t}}{t^{e^{-t}}} \right) \\
 & = -\gamma + \ln \left( \lim_{t \rightarrow 0} \frac{e^{-t}}{g'(t)} \right) \text{ where } g(t) = t^{e^{-t}}
 \end{aligned}$$

Note that  $\gamma \approx 0.577215$  is Euler-Mascheroni constant, please read (Weisstein, Euler-Mascheroni Constant) for more detailed about Euler-Mascheroni constant.

$$\gamma = - \int_0^{+\infty} e^{-x} \ln x dx = \lim_{n \rightarrow +\infty} \left( \sum_{k=1}^n \frac{1}{k} - \ln(n) \right) \approx 0.577215$$

We have:

$$\ln(g(t)) = e^{-t} \ln t \Rightarrow \frac{g'(t)}{g(t)} = \frac{e^{-t}(1 - tlnt)}{t} \Rightarrow g'(t) = \frac{t^{e^{-t}} e^{-t}(1 - tlnt)}{t}$$

It implies that:

$$\begin{aligned}
 \psi(1) & = -\gamma + \ln \left( \lim_{t \rightarrow 0} \frac{e^{-t} t}{t^{e^{-t}} e^{-t}(1 - tlnt)} \right) = \gamma + \ln \left( \lim_{t \rightarrow 0} \frac{1}{t^{e^{-t}-1}(1 - tlnt)} \right) \\
 & = -\gamma + \ln \left( \frac{1}{\lim_{t \rightarrow 0} (t^{e^{-t}-1}) \lim_{t \rightarrow 0} (1 - tlnt)} \right)
 \end{aligned}$$

We also have:

$$\lim_{t \rightarrow 0} (t^{e^{-t}-1}) = \lim_{t \rightarrow 0} \exp \left( \frac{e^{-t} - 1}{1/lnt} \right) = \exp \left( \lim_{t \rightarrow 0} \frac{e^{-t} - 1}{1/lnt} \right)$$

(due to transformation with regard to indeterminate form (Wikipedia, Indeterminate form, 2014))

$$= \exp \left( \lim_{t \rightarrow 0} \frac{d(e^{-t} - 1)}{d(1/lnt)} \right) = \exp \left( \lim_{t \rightarrow 0} \frac{-e^{-t}}{\frac{1}{t(lnt)^2}} \right)$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}
 & = \exp \left( \lim_{t \rightarrow 0} e^{-t} t (lnt)^2 \right) = \exp \left( \lim_{t \rightarrow 0} t (lnt)^2 \right) = \exp \left( \lim_{t \rightarrow 0} \frac{(lnt)^2}{1/t} \right) \\
 & = \exp \left( \lim_{t \rightarrow 0} \frac{d((lnt)^2)}{d(1/t)} \right) = \exp \left( \lim_{t \rightarrow 0} \frac{2lnt/t}{-1/t^2} \right)
 \end{aligned}$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

### III.1. Knowledge sub-model

---

$$\begin{aligned}
&= \exp\left(-\lim_{t \rightarrow 0} 2tlnt\right) = \exp\left(-\lim_{t \rightarrow 0} \frac{2lnt}{1/t}\right) \\
&= \exp\left(-\lim_{t \rightarrow 0} \frac{d(2lnt)}{d(1/t)}\right) = \exp\left(\lim_{t \rightarrow 0} \frac{2/t}{1/t^2}\right)
\end{aligned}$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$= \exp\left(\lim_{t \rightarrow 0} 2t\right) = \exp(0) = 1$$

We also have:

$$\lim_{t \rightarrow 0} (1 - tlnt) = 1 - \lim_{t \rightarrow 0} (tlnt) = 1 - \lim_{t \rightarrow 0} \frac{lnt}{1/t}$$

$$= 1 - \lim_{t \rightarrow 0} \frac{d(lnt)}{d(1/t)} = 1 + \lim_{t \rightarrow 0} \frac{1/t}{1/t^2}$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$= 1 + \lim_{t \rightarrow 0} t = 1 + 0 = 1$$

Therefore, it implies that:

$$\psi(1) = -\gamma + \ln\left(\frac{1}{\lim_{t \rightarrow 0} (t^{e^{-t}-1}) \lim_{t \rightarrow 0} (1 - tlnt)}\right) = -\gamma + \ln\left(\frac{1}{1 * 1}\right) = -\gamma$$

Briefly, the value  $\psi(1)$  is always equal to  $\gamma$ . Given  $x$  is positive integer, formula III.1.58 is replaced by formula III.1.60 for calculating digamma function in case of positive integer number.

$$\begin{aligned}
\psi(1) &= -\gamma \\
\psi(x) &= -\gamma + \sum_{k=1}^{x-1} \frac{1}{k} \\
(x \text{ positive integer and } x \geq 2)
\end{aligned}$$

*Formula III.1.60. Digamma function in case of positive integer variable Proof,*

$$\begin{aligned}
\forall x \geq 2, \psi(x) &= \psi(x-1) + \frac{1}{x-1} = \psi(x-2) + \frac{1}{x-2} + \frac{1}{x-1} \\
&\quad (\text{by applying formula III.1.59}) \\
&= \psi(x-3) - \frac{1}{x+3} - \frac{1}{x+2} - \frac{1}{x+1} \\
&= \dots = \psi(x-(x-1)) + \underbrace{\frac{1}{x-(x-1)} + \frac{1}{x-(x-2)} + \dots + \frac{1}{x-1}}_{x-1} \\
&== \psi(x-(x-1)) + \underbrace{\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{x-1}}_{x-1}
\end{aligned}$$

$$= \psi(1) + \sum_{k=1}^{x-1} \frac{1}{k} = -\gamma + \sum_{k=1}^{x-1} \frac{1}{k}$$

Let  $\psi_1(x)$  be the first-order of digamma function, we have:

$$\psi_1(x) = \psi'(x) = \frac{d \left( \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}} \right) dt \right)}{dx} = \int_0^{+\infty} \frac{d \left( \frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}} \right)}{dx} dt$$

(because function  $\frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}}$  is continuous and differentiable in open interval  $(0, +\infty)$  with regard to variable  $x$ )

$$= \int_0^{+\infty} \frac{d \left( -\frac{e^{-xt}}{1-e^{-t}} \right)}{dx} dt = \int_0^{+\infty} \frac{te^{-xt}}{1-e^{-t}} dt$$

We also have:

$$\begin{aligned} \psi_1(x+y) &= \frac{d\psi(x+y)}{dx} = \frac{d\psi(x+y)}{dy} = \int_0^{+\infty} \frac{d \left( -\frac{e^{-(x+y)t}}{1-e^{-t}} \right)}{dx} dt \\ &= \int_0^{+\infty} \frac{d \left( -\frac{e^{-(x+y)t}}{1-e^{-t}} \right)}{dy} dt = \int_0^{+\infty} \frac{te^{-(x+y)t}}{1-e^{-t}} dt \end{aligned}$$

Function  $\psi_1(x)$  is also called *trigamma* function; please refer to documents (Weisstein, Polygamma Function) and (Weisstein, Trigamma Function) for more detailed about trigamma function. Briefly, formula III.1.61 expresses trigamma function.

$$\begin{aligned} \psi_1(x) &= \int_0^{+\infty} \frac{te^{-xt}}{1-e^{-t}} dt \\ \psi_1(x+y) &= \int_0^{+\infty} \frac{te^{-(x+y)t}}{1-e^{-t}} dt \end{aligned}$$

### Formula III.1.61. Trigamma function

Suppose variable  $x$  is non-zero, we have:

$$\begin{aligned} \psi_1(x+1) &= \int_0^{+\infty} \frac{te^{-(x+1)t}}{1-e^{-t}} dt = \int_0^{+\infty} \left( \frac{te^{-xt}}{1-e^{-t}} - te^{-xt} \right) dt \\ &= \int_0^{+\infty} \frac{te^{-xt}}{1-e^{-t}} dt - \int_0^{+\infty} te^{-xt} dt = \psi_1(x) + \frac{1}{x} \int_0^{+\infty} td(e^{-xt}) dt \\ &= \psi_1(x) + \frac{1}{x} te^{-xt} \Big|_0^{+\infty} - \frac{1}{x} \int_0^{+\infty} e^{-xt} dt \end{aligned}$$

### III.1. Knowledge sub-model

---

$$\begin{aligned}
&= \psi_1(x) + \frac{1}{x} te^{-xt} \Big|_0^{+\infty} + \frac{1}{x^2} e^{-xt} \Big|_0^{+\infty} \\
&= \psi_1(x) + \frac{1}{x} \lim_{t \rightarrow +\infty} (te^{-xt}) - \frac{1}{x} \lim_{t \rightarrow 0} (te^{-xt}) + \frac{1}{x^2} \lim_{t \rightarrow +\infty} e^{-xt} - \frac{1}{x^2} \lim_{t \rightarrow 0} e^{-xt} \\
&= \psi_1(x) + \frac{1}{x} \lim_{t \rightarrow +\infty} (te^{-xt}) - \frac{1}{x^2} = \psi_1(x) - \frac{1}{x^2} + \frac{1}{x} \lim_{t \rightarrow +\infty} \frac{t}{e^{xt}} \\
&= \psi_1(x) - \frac{1}{x^2} + \frac{1}{x} \lim_{t \rightarrow +\infty} \frac{d(t)}{d(e^{xt})} = \psi_1(x) - \frac{1}{x^2} + \frac{1}{x} \lim_{t \rightarrow +\infty} \frac{1}{xe^{xt}}
\end{aligned}$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$= \psi_1(x) - \frac{1}{x^2}$$

Briefly, the recurrence formula of trigamma function is specified by formula III.1.62.

$$\psi_1(x+1) = \psi_1(x) - \frac{1}{x^2}$$

*Formula III.1.62. Recurrence formula of trigamma function*

Formula III.1.62 shows recurrence relation (Wikipedia, Polygamma function, 2014) of trigamma function, which implicates that it is very easy to compute  $\psi_1(x)$  if variable  $x$  is positive integer. Thus, it is necessary to calculate the starting positive value  $\psi_1(1)$ , we have:

$$\begin{aligned}
\psi_1(x) &= \psi'(x) = \frac{d}{dx} \left( \frac{\Gamma'(x)}{\Gamma(x)} \right) \\
&= \frac{\Gamma''(x)\Gamma(x) - \Gamma'(x)\Gamma'(x)}{\Gamma(x)\Gamma(x)} = \frac{\Gamma''(x)}{\Gamma(x)} - \left( \frac{\Gamma'(x)}{\Gamma(x)} \right)^2 \\
&= \frac{\Gamma''(x)}{\Gamma(x)} - (\psi(x))^2 \\
\Rightarrow \psi_1(x) &= \frac{\Gamma''(1)}{\Gamma(1)} - (\psi(1))^2 = \frac{\Gamma''(1)}{\Gamma(1)} - \gamma^2
\end{aligned}$$

We have:

$$\Gamma(1) = \int_0^{+\infty} e^{-t} dt = -e^{-t} \Big|_0^{+\infty} = \lim_{t \rightarrow +\infty} (-e^{-t}) + 1 = 0 + 1 = 1$$

$$\begin{aligned}
\Gamma'(x) &= \frac{d(\int_0^{+\infty} t^{x-1} e^{-t} dt)}{dx} = \int_0^{+\infty} t^{x-1} e^{-t} lnt dt \\
\Rightarrow \Gamma''(x) &= \frac{d(\Gamma'(x))}{dx} = \frac{d(\int_0^{+\infty} t^{x-1} e^{-t} lnt dt)}{dx} = \int_0^{+\infty} t^{x-1} e^{-t} (lnt)^2 dt \\
\Rightarrow \Gamma''(1) &= \int_0^{+\infty} e^{-t} (lnt)^2 dt = \gamma^2 + \frac{\pi^2}{6}
\end{aligned}$$

Where  $\gamma \approx 0.577215$  is Euler-Mascheroni constant (Weisstein, Euler-Mascheroni Constant). The evaluation  $\int_0^{+\infty} e^{-t} (\ln t)^2 dt = \gamma^2 + \frac{\pi^2}{6}$  is found out in (Weisstein, Euler-Mascheroni Constant).

It implies that

$$\psi_1(x) = \frac{\Gamma''(1)}{\Gamma(1)} - \gamma^2 = \frac{\gamma^2 + \frac{\pi^2}{6}}{1} - \gamma^2 = \frac{\pi^2}{6}$$

Briefly, the value  $\psi_1(1)$  is always equal to  $\frac{\pi^2}{6}$ . Given  $x$  is positive integer, formula III.1.61 is replaced by formula III.1.63 for calculating trigamma function in case of positive integer number, as follows:

$$\begin{aligned}\psi_1(1) &= \frac{\pi^2}{6} \\ \psi_1(x) &= \frac{\pi^2}{6} - \sum_{k=1}^{x-1} \frac{1}{k^2}\end{aligned}$$

( $x$  positive integer and  $x \geq 2$ )

*Formula III.1.63.* Trigamma function in case of positive integer variable

Proof,

$$\begin{aligned}\forall x \geq 2, \psi_1(x) &= \psi_1(x-1) - \frac{1}{x-1} = \psi_1(x-2) - \left( \frac{1}{(x-2)^2} + \frac{1}{(x-1)^2} \right) \\ &\quad \text{(by applying formula III.1.62)} \\ &= \psi_1(x-3) - \left( \frac{1}{(x-3)^2} + \frac{1}{(x-2)^2} + \frac{1}{(x-1)^2} \right) \\ &= \dots \\ &= \psi_1(x-(x-1)) - \underbrace{\left( \frac{1}{(x-(x-1))^2} + \frac{1}{(x-(x-2))^2} + \dots + \frac{1}{(x-1)^2} \right)}_{x-1} \\ &== \psi_1(x-(x-1)) - \underbrace{\left( \frac{1}{1^2} + \frac{1}{2^2} + \dots + \frac{1}{(x-1)^2} \right)}_{x-1} \\ &= \psi_1(1) - \sum_{k=1}^{x-1} \frac{1}{k^2} = \frac{\pi^2}{6} - \sum_{k=1}^{x-1} \frac{1}{k^2}\end{aligned}$$

The beta function (Wikipedia, Beta function, 2014) denoted  $B(x, y)$  is a special function defined as below:

$$B(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

*Formula III.1.64.* Beta function  $B(x, y)$

### III.1. Knowledge sub-model

---

Please distinguish beta density function  $\beta(X; a, b)$  specified in formulas III.1.56, III.1.49, and III.1.12 known as probability density function (PDF) from beta function  $B(x, y)$  specified by formula III.1.64.

The first-order partial derivative of  $B(x, y)$  is determined as follows:

$$\begin{aligned}\frac{\partial B(x, y)}{\partial x} &= \Gamma(y) \left( \frac{\Gamma'(x)\Gamma(x+y) - \Gamma(x)\Gamma'(x+y)}{(\Gamma(x+y))^2} \right) \\ &= \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \frac{\Gamma'(x)\Gamma(x+y) - \Gamma(x)\Gamma'(x+y)}{\Gamma'(x)\Gamma(x+y) - \Gamma(x)\Gamma'(x+y)} \\ &= B(x, y) \frac{\Gamma'(x)\Gamma(x+y) - \Gamma(x)\Gamma'(x+y)}{\Gamma(x)\Gamma(x+y)} \\ &= B(x, y) \left( \frac{\Gamma'(x)}{\Gamma(x)} - \frac{\Gamma'(x+y)}{\Gamma(x+y)} \right)\end{aligned}$$

Due to digamma function  $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$ , we have:

$$\frac{\partial B(x, y)}{\partial x} = B(x, y)(\psi(x) - \psi(x+y))$$

*Formula III.1.65.* First-order partial derivative of beta function  $B(x, y)$

The digamma function is always determined by formulas III.1.58 and III.1.60. Substituting beta function  $B(x, y)$  specified formula III.1.64 into formula III.1.56, the beta density function is re-written:

$$f(X; a, b) = \beta(X; a, b) = \frac{1}{B(a, b)} X^{a-1} (1-X)^{b-1}$$

*Formula III.1.66.* Beta density function with regard to beta function

Now we specify the likelihood function of beta distribution as below:

$$\begin{aligned}L(a, b) &= \prod_{i=1}^n \beta(x_i; a, b) = \prod_{i=1}^n \frac{1}{B(a, b)} x_i^{a-1} (1-x_i)^{b-1} \\ &= \frac{1}{B^n(a, b)} \left( \prod_{i=1}^n x_i^{a-1} \right) \left( \prod_{i=1}^n (1-x_i)^{b-1} \right)\end{aligned}$$

Take the logarithm of  $L(a, b)$ , we have the log-likelihood function for beta distribution as follows:

$$\begin{aligned}LnL(a, b) &= n \ln \left( \frac{1}{B(a, b)} \right) + \sum_{i=1}^n ((a-1) \ln(x_i)) + \sum_{i=1}^n ((b-1) \ln(1-x_i)) \\ &= -n \ln(B(a, b)) + (a-1) \sum_{i=1}^n \ln(x_i) + (b-1) \sum_{i=1}^n \ln(1-x_i)\end{aligned}$$

*Formula III.1.67.* Log-likelihood function of beta density function (beta distribution)

Please pay attention to formula [III.1.67](#) because formula [III.1.67](#) is specific case of formula [III.1.54](#) mentioned in previous sub-section [III.1.5.1](#); thus, MLE is applied into beta distribution.

Note that  $LnL(a, b) = -\infty$  if any instance  $x_i$  is equal to 1 or 0. In practice, we should assign a very large number to  $LnL(a, b)$  in this case, instead of keeping the infinity.

Two parameters  $a$  and  $b$  must be determined so that they maximize the log-likelihood function. Thus, by taking two first-order partial derivatives of log-likelihood function specified in formula [III.1.67](#) corresponding to two parameters and applying formulas [III.1.65](#), we have:

$$\begin{aligned}\frac{\partial LnL(a, b)}{\partial a} &= -n \frac{1}{B(a, b)} \frac{\partial B(a, b)}{\partial a} + \sum_{i=1}^n \ln(x_i) \\ &= -n(\psi(a) - \psi(a + b)) + \sum_{i=1}^n \ln(x_i)\end{aligned}$$

*Formula III.1.68.* First-order partial derivative of log-likelihood function of beta density function with regard to parameter  $a$

$$\begin{aligned}\frac{\partial LnL(a, b)}{\partial b} &= -n \frac{1}{B(a, b)} \frac{\partial B(a, b)}{\partial b} + \sum_{i=1}^n \ln(1 - x_i) \\ &= -n(\psi(b) - \psi(a + b)) + \sum_{i=1}^n \ln(1 - x_i)\end{aligned}$$

*Formula III.1.69.* First-order partial derivative of log-likelihood function of beta density function with regard to parameter  $b$

Where  $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$  is digamma function by specified by formulas [III.1.58](#) and [III.1.60](#). Note that notation  $\frac{\partial f}{\partial x}$  denotes first-order partial derivative of multi-variable function  $f$  with regard to variable  $x$ .

Please pay attention to formulas [III.1.68](#) and [III.1.69](#) for determining two first-order partial derivatives of log-likelihood function of beta distribution. Setting such two partial derivatives equal 0 so as to find out two parameters  $a$  and  $b$ , we have a set of equations whose two solutions are the values of  $a$  and  $b$ :

$$\begin{aligned} \left\{ \begin{array}{l} \frac{\partial \ln L(a, b)}{\partial a} = 0 \\ \frac{\partial \ln L(a, b)}{\partial b} = 0 \end{array} \right. &\Leftrightarrow \begin{cases} -n(\psi(a) - \psi(a+b)) + \sum_{i=1}^n \ln(x_i) = 0 \\ -n(\psi(b) - \psi(a+b)) + \sum_{i=1}^n \ln(1-x_i) = 0 \end{cases} \\ &\Leftrightarrow \begin{cases} \psi(a) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(x_i) \\ \psi(b) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(1-x_i) \end{cases} \end{aligned}$$

*Formula III.1.70.* The set of differential equations for estimating parameters  $a$  and  $b$

Formula [III.1.70](#) shows the set of differential equations for estimating parameters  $a$  and  $b$ . In the next sub-section [III.1.5.3](#), I invent the simple form of such equations when parameters  $a$  and  $b$  are positive integer numbers. I also propose the algorithm to find out the approximate solutions.

Before going into the main sub-section [III.1.5.3](#), it is necessary to glance over the co-variance matrix  $\text{Var}(a, b)$  of parameters of beta density function mentioned in previous sub-section [III.1.5.1](#) although the co-variance matrix is not important subject in the research. Let  $H(a, b)$  be the second-order partial derivative matrix so-called Hessian matrix, we have:

$$H(a, b) = \begin{pmatrix} \frac{\partial \ln L}{\partial^2 a} & \frac{\partial \ln L}{\partial a \partial b} \\ \frac{\partial \ln L}{\partial a \partial b} & \frac{\partial \ln L}{\partial^2 b} \end{pmatrix}$$

Note, the bracket  $(.)$  denotes matrix.

Basing on formulas [III.1.68](#) and [III.1.69](#), we can determine four second-order partial derivatives of log-likelihood function as follows:

$$\frac{\partial \ln L}{\partial^2 a} = \frac{\partial \psi(a+b)}{\partial a} - n \frac{\partial \psi(a)}{\partial a} = \psi_1(a+b) - n\psi_1(a)$$

$$\frac{\partial \ln L}{\partial a \partial b} = \frac{\partial \psi(a+b)}{\partial b} = \psi_1(a+b)$$

$$\frac{\partial \ln L}{\partial b \partial a} = \frac{\partial \psi(a+b)}{\partial a} = \psi_1(a+b)$$

$$\frac{\partial \ln L}{\partial^2 b} = \frac{\partial \psi(a+b)}{\partial b} - n \frac{\partial \psi(b)}{\partial b} = \psi_1(a+b) - n\psi_1(b)$$

Where  $\psi_1(.)$  denotes trigamma function specified by formulas [III.1.61](#) and [III.1.63](#). According to formula [III.1.55](#), the co-variance matrix  $\text{Var}(a, b)$  is the inversion of negative expectation of Hessian matrix.

$$\text{Var}(a, b) = (-E(H(a, b)))^{-1}$$

$$\begin{aligned}
 &= \left( \begin{matrix} -E(\psi_1(a+b) - n\psi_1(a)) & \psi_1(a+b) \\ \psi_1(a+b) & \psi_1(a+b) - n\psi_1(b) \end{matrix} \right)^{-1} \\
 &= \left( \begin{matrix} -(\psi_1(a+b) - n\psi_1(a)) & \psi_1(a+b) \\ \psi_1(a+b) & \psi_1(a+b) - n\psi_1(b) \end{matrix} \right)^{-1}
 \end{aligned}$$

(Because trigamma functions  $\psi_1(a)$ ,  $\psi_1(b)$ , and  $\psi_1(a+b)$  are only dependent on parameters  $a$  and  $b$ , the expectation of  $H(a, b)$  is merely  $H(a, b)$ )

$$\begin{aligned}
 &= \left( \begin{matrix} n\psi_1(a) - \psi_1(a+b) & -\psi_1(a+b) \\ -\psi_1(a+b) & n\psi_1(b) - \psi_1(a+b) \end{matrix} \right)^{-1} \\
 &= \frac{1}{n^2\psi_1(a)\psi_1(b) - n\psi_1(a+b)(\psi_1(a) + \psi_1(b))} \\
 &\quad * \left( \begin{matrix} n\psi_1(b) - \psi_1(a+b) & \psi_1(a+b) \\ \psi_1(a+b) & n\psi_1(a) - \psi_1(a+b) \end{matrix} \right)
 \end{aligned}$$

Briefly, formula III.1.71 specifies the co-variance matrix of parameters of beta density function as follows:

$$Var(a, b) = A \left( \begin{matrix} n\psi_1(b) - \psi_1(a+b) & \psi_1(a+b) \\ \psi_1(a+b) & n\psi_1(a) - \psi_1(a+b) \end{matrix} \right)$$

*Formula III.1.71.* Co-variance matrix of parameters of beta density function

Where  $\psi_1(\cdot)$  denotes trigamma function and,

$$A = \frac{1}{n^2\psi_1(a)\psi_1(b) - n\psi_1(a+b)(\psi_1(a) + \psi_1(b))}$$

Formula III.1.71 is concrete case of formula III.1.55 when probability distribution is beta distribution. If parameters  $a$  and  $b$  are positive integers, the trigamma function  $\psi_1(\cdot)$  is calculated simply according to formula III.1.63; this is the ultimate purpose of this sub-section III.1.5.2.

The roots of diagonal elements are the standard deviations (standard errors) of parameter estimates. Let  $\sigma(\hat{a})$  and  $\sigma(\hat{b})$  be the standard errors of optimal parameters  $\hat{a}$  and  $\hat{b}$  where  $\hat{a}$  and  $\hat{b}$  are solutions of equations specified by formula III.1.70, we have:

$$\begin{aligned}
 \sigma(\hat{a}) &= \sqrt{A(n\psi_1(\hat{b}) - \psi_1(\hat{a} + \hat{b}))} \\
 \sigma(\hat{b}) &= \sqrt{A(n\psi_1(\hat{a}) - \psi_1(\hat{a} + \hat{b}))}
 \end{aligned}$$

*Formula III.1.72.* Standard errors of parameter estimates of beta distribution

Where  $\psi_1(\cdot)$  denotes the trigamma function and,

$$A = \frac{1}{n^2\psi_1(a)\psi_1(b) - n\psi_1(a+b)(\psi_1(a) + \psi_1(b))}$$

Formula III.1.72 specifying standard errors of parameter estimates ends up this sub-section III.1.5.2 mentioning applying MLE technique into beta distribution.

Now the next sub-section III.1.5.3 will mention the proposed approach to solve the set of differential equations specified in formula III.1.62.

### III.1.5.3. Algorithm to solve the equations whose solutions are parameter estimators

As specified by formula III.1.70, the parameter estimators  $\hat{a}$  and  $\hat{b}$  are solutions of two equations:

$$\begin{cases} \psi(a) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(x_i) \\ \psi(b) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(1-x_i) \end{cases}$$

Where  $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)} = \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}} \right) dt$

Obviously, these equations are differential functions whose solutions are families of functions. Because it is too difficult to solve such equations, I find out the simple form of them when parameters  $a$  and  $b$  are positive integer numbers and hence, differential functions are eliminated from the simple form.

Suppose that parameters  $a$  and  $b$  are positive integer numbers. Expanding the expression  $\psi(a) - \psi(a+b)$ , we have:

$$\begin{aligned} \psi(a) - \psi(a+b) &= \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-at}}{1-e^{-t}} \right) dt - \int_0^{+\infty} \left( \frac{e^{-t}}{t} - \frac{e^{-(a+b)t}}{1-e^{-t}} \right) dt \\ &= \int_0^{+\infty} \left( \frac{e^{-(a+b)t}}{1-e^{-t}} - \frac{e^{-at}}{1-e^{-t}} \right) dt = \int_0^{+\infty} \frac{e^{-at}(e^{-bt}-1)}{1-e^{-t}} dt \\ &= e^a \int_0^{+\infty} (e^{-bt}-1) \frac{e^{-t}}{1-e^{-t}} dt = e^a \int_{-\infty}^0 ((1-e^x)^b - 1) dx \end{aligned}$$

Due to,

$$x = \ln(1 - e^{-t}) \Rightarrow \begin{cases} e^{-t} = 1 - e^x \\ dx = \frac{e^{-t}}{1-e^{-t}} dt \\ x \rightarrow 0 \text{ when } t \rightarrow +\infty \text{ and } x \rightarrow -\infty \text{ when } t \rightarrow 0 \end{cases}$$

Expanding the polynomial  $(1 - e^x)^b$  with note that  $b$  is positive integer number, we also have

$$(1 - e^x)^b = \sum_{k=0}^b (-1)^k C_b^k e^{kx}$$

Where  $C_b^k$  is the combination taken  $k$  of  $b$  elements,

$$C_b^k = \binom{b}{k} = \frac{b!}{k!(b-k)!}$$

Hence, we have

$$\begin{aligned}\psi(a) - \psi(a+b) &= e^a \int_{-\infty}^0 \left( \left( \sum_{k=0}^b (-1)^k C_b^k e^{kx} \right) - 1 \right) dx \\ &= e^a \int_{-\infty}^0 \left( \sum_{k=1}^b (-1)^k C_b^k e^{kx} \right) dx = e^a \sum_{k=1}^b \left( (-1)^k C_b^k \int_{-\infty}^0 e^{kx} dx \right) \\ &= e^a \sum_{k=1}^b \left( (-1)^k C_b^k \left( \frac{e^{kx}}{k} \Big|_{-\infty}^0 \right) \right) = e^a \sum_{k=1}^b \frac{(-1)^k}{k} C_b^k\end{aligned}$$

In the similar way, we have:

$$\psi(b) - \psi(a+b) = e^b \sum_{k=1}^a \frac{(-1)^k}{k} C_a^k$$

The equations whose solutions are parameter estimators becomes two following equations:

$$\begin{cases} e^a \sum_{k=1}^b \frac{(-1)^k}{k} C_b^k = \frac{1}{n} \sum_{i=1}^n \ln(x_i) \\ e^b \sum_{k=1}^a \frac{(-1)^k}{k} C_a^k = \frac{1}{n} \sum_{i=1}^n \ln(1-x_i) \end{cases} \Leftrightarrow \begin{cases} G_1(a, b) = L_1 \\ G_2(a, b) = L_2 \end{cases}$$

*Formula III.1.73.* Two simplest equations for estimating positive integer parameters  $a$  and  $b$

Where,

$$G_1(a, b) = e^a \sum_{k=1}^b \frac{(-1)^k}{k} C_b^k \text{ and } G_2(a, b) = e^b \sum_{k=1}^a \frac{(-1)^k}{k} C_a^k$$

$$L_1 = \frac{1}{n} \sum_{i=1}^n \ln(x_i) \text{ and } L_2 = \frac{1}{n} \sum_{i=1}^n \ln(1-x_i)$$

Obviously, formula III.1.73 is the simplest form from which all differential functions are removed. However, the number of solutions of equations in formula III.1.73 is large and it is very difficult to find out them. Suppose there is the restriction:

*“The range of variables  $a$  and  $b$  is from 1 to  $n$  where  $n$  is the whole positive number and not greater than the number of evidences in training data”.*

I propose the iterative algorithm that each pair values  $(a_i, b_i)$  which are values of variables  $a$  and  $b$  are fed to  $G_1, G_2$  at each iteration. Two biases  $\Delta_1 = G_1(a_i,$

$b_i) - L_1$  and  $\Delta_2 = G_2(a, b_i) - L_2$  are computed. The normal bias is the root of sum of the second power  $\Delta_1$  and the second power of  $\Delta_2$  and so we have  $\Delta = \sqrt{\Delta_1^2 + \Delta_2^2}$ . The pair  $(\hat{a}, \hat{b})$  whose normal bias  $\Delta$  is minimum are chosen as the parameter estimators. The algorithm is described in table III.1.25 as below:

```

minΔ = +∞
â = ī = 1 (uniform distribution )
For a=1 to n do
    For b=1 to n do
        Δ1=G1(a, b)-L1
        Δ2=G2(a, b)-L2
        Δ=√Δ12 + Δ22
        If Δ < minΔ then
            minΔ=Δ
            â=a
            ī=b
        End If
    End For a
End For b
(â and ī are optimal parameters)

```

**Table III.1.25.** Iterative algorithm solving simplest equations specified by formula III.1.73 for estimating parameters  $a$  and  $b$

Where  $min\Delta$  denotes minimum bias.

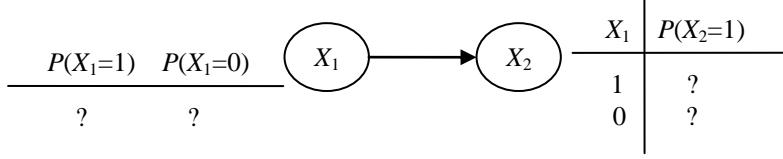
Note that the power and combination functions occurring in  $G_1$  and  $G_2$  become unexpectedly huge when the range of  $a$  and  $b$  is wide; so the upper bound  $n$  should not be large. With respect to beta distribution, the probability of variable in Bayesian network is the expectation of beta distribution, namely,  $P(X) = E(\beta(a, b)) = \frac{a}{a+b}$ ; please see formula III.1.50 for knowing probability of  $X$  as expectation of beta distribution. The ratio  $\frac{a}{a+b}$  is not so dependent on the amplitude of  $a$  or  $b$ ; for example, the ratio  $\frac{5}{5+7}$  whose parameters  $a$  and  $b$  equal 5 and 7, respectively is the same to the ratio  $\frac{10}{10+14}$  whose parameters  $a$  and  $b$  equal 10 and 14, respectively. That is why we do not need to define the range of  $a$  and  $b$  to be so wide.

The next sub-section III.1.5.4 describes an example illustrating proposed algorithm in table III.1.25.

#### III.1.5.4. An example of how to specify prior probabilities

Suppose there is the BN having two variables  $X_1$ ,  $X_2$  and one arc which links them together. Variables  $X_1$  and  $X_2$  obey beta distribution. We need to specify the prior CPT (s), namely the prior conditional probabilities  $P(X_1=1)$ ,

$P(X_2=1|X_1=1)$  and  $P(X_2=1|X_1=0)$ . Figure III.1.28 depicts the example of BN including such variables  $X_1$  and  $X_2$  without CPT (s).



**Figure III.1.28.** Bayesian network without CPT (s)

In figure III.1.28, question marks (?) indicate undefined conditional probabilities.

Let  $\beta_1(a_1, b_1)$ ,  $\beta_2(a_2, b_2)$ ,  $\beta_3(a_3, b_3)$  be beta distributions of conditional probabilities  $P(X_1=1)$ ,  $P(X_2=1|X_1=1)$  and  $P(X_2=1|X_1=0)$ . Applying formula III.1.50 for specifying probability of  $X$  as expectation of beta distribution, we have:

$$P(X_1 = 1) = E(\beta_1(a_1, b_1)) = \frac{a_1}{a_1 + b_1}$$

$$P(X_2 = 1 | X_1 = 1) = E(\beta_2(a_2, b_2)) = \frac{a_2}{a_2 + b_2}$$

$$P(X_2 = 1 | X_1 = 0) = E(\beta_2(a_3, b_3)) = \frac{a_3}{a_3 + b_3}$$

It is necessary to determine three parameter pairs  $(a_1, b_1)$ ,  $(a_2, b_2)$  and  $(a_3, b_3)$  of three beta distributions  $\beta_1$ ,  $\beta_2$  and  $\beta_3$ , respectively. Suppose we perform 5 trials of a random process, the outcome of  $i^{th}$  trial denoted  $D^{(i)}$  is considered as an evidence in which  $X_1$  and  $X_2$  obtains value 0 or 1. So we have the vector of 5 evidences  $\mathcal{D} = (D^{(1)}, D^{(2)}, D^{(3)}, D^{(4)}, D^{(5)})$ . Table III.1.26 shows these evidences.

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

**Table III.1.26.** The evidences corresponding to 5 trials

According to the proposed algorithm described in table III.1.25, let  $L_{ij}$ ,  $G_{ij}$ ,  $\Delta_{ij}$ ,  $\Delta_i$  be the values of  $L_j$ ,  $G_j$ ,  $\Delta_j$ ,  $\Delta$  with respect to  $\beta_i$  where  $i = \overline{1,3}$  and  $j = \overline{1,2}$ . We have:

- $L_{11} = \frac{1}{n} \sum_{i=1}^n \ln(x_i)$  and  $L_{12} = \frac{1}{n} \sum_{i=1}^n \ln(1 - x_i)$  where  $x_i$  is the instances of  $X_1$ .
- $L_{21} = \frac{1}{n} \sum_{i=1}^n \ln(x_i)$  and  $L_{22} = \frac{1}{n} \sum_{i=1}^n \ln(1 - x_i)$  where  $x_i$  is the instances of  $X_2$  given  $X_1=1$ .
- $L_{31} = \frac{1}{n} \sum_{i=1}^n \ln(x_i)$  and  $L_{32} = \frac{1}{n} \sum_{i=1}^n \ln(1 - x_i)$  where  $x_i$  is the instances of  $X_2$  given  $X_1=0$ .

### III.1. Knowledge sub-model

---

- $G_{11}(a_1, b_1) = e^{a_1} \sum_{k=1}^{b_1} \frac{(-1)^k}{k} C_{b_1}^k$  and  $G_{12}(a_1, b_1) = e^{b_1} \sum_{k=1}^{a_1} \frac{(-1)^k}{k} C_{a_1}^k$
- $G_{21}(a_2, b_2) = e^{a_2} \sum_{k=1}^{b_2} \frac{(-1)^k}{k} C_{b_2}^k$  and  $G_{22}(a_2, b_2) = e^{b_2} \sum_{k=1}^{a_2} \frac{(-1)^k}{k} C_{a_2}^k$
- $G_{31}(a_3, b_3) = e^{a_3} \sum_{k=1}^{b_3} \frac{(-1)^k}{k} C_{b_3}^k$  and  $G_{32}(a_3, b_3) = e^{b_3} \sum_{k=1}^{a_3} \frac{(-1)^k}{k} C_{a_3}^k$
- $\Delta_{11} = G_{11} - L_{11}$ ,  $\Delta_{12} = G_{12} - L_{12}$  and  $\Delta_1 = \sqrt{\Delta_{11}^2 + \Delta_{12}^2}$
- $\Delta_{21} = G_{21} - L_{21}$ ,  $\Delta_{22} = G_{22} - L_{22}$  and  $\Delta_2 = \sqrt{\Delta_{21}^2 + \Delta_{22}^2}$
- $\Delta_{31} = G_{31} - L_{31}$ ,  $\Delta_{32} = G_{32} - L_{32}$  and  $\Delta_3 = \sqrt{\Delta_{31}^2 + \Delta_{32}^2}$

In order to be convenient for computation, we have some conventions. Due to  $\ln(0) \rightarrow -\infty$ ,  $\ln(0)$  is represented as a extremely large number, for example,  $-1000$ . So it is possible to approximate  $\ln(0)$  to  $-1000$  and we have  $\ln(0) \approx -1000$ . Expanding the range of function  $\ln(X)$  when  $X$  is binary variable, if  $X = 0$  then  $\ln(X=0) \approx -1000$ , otherwise  $\ln(X=1) \approx +1000$ . From above evidences shown in table III.1.26, it is easy to compute  $L_{ij}$ . For example, we have:

$$\begin{aligned} L_{11} &= \frac{1}{5} (4\ln(X_1 = 1) + \ln(X_1 = 0)) = \frac{1}{5} (4 * 1000 - 1000) = 600 \\ L_{12} &= \frac{1}{5} (4\ln(1 - (X_1 = 1)) + \ln(1 - (X_1 = 0))) = \frac{1}{5} (4\ln(0) + \ln(1)) \\ &= \frac{1}{5} (4 * (-1000) + 1000) = -600 \end{aligned}$$

In the similar way, all  $L_{ij}$  are determined. Table III.1.27 shows values of  $L_{ij}$  corresponding to beta density functions  $\beta_1$ ,  $\beta_2$ , and  $\beta_3$ .

$\beta_1$	$\beta_2$	$\beta_3$
$L_{11} = 600$	$L_{21} = 500$	$L_{31} = -1000$
$L_{12} = -600$	$L_{22} = -500$	$L_{32} = 1000$

**Table III.1.27.** The values of  $L_{ij}$  corresponding to beta density function  $\beta_1$ ,  $\beta_2$ , and  $\beta_3$

Suppose the range of all parameters is from 1 to 4. By applying the proposed algorithm described in table III.1.25, it is easy to compute the normal biases. For example, given  $a_1 = 1$  and  $b_1 = 1$ , we have:

$$G_{11}(a_1 = 1, b_1 = 1) = e^1 \sum_{k=1}^1 \frac{(-1)^k}{k} C_1^k = -e C_1^1 = -e \approx -2.72$$

$$G_{12}(a_1 = 1, b_1 = 1) = e^1 \sum_{k=1}^1 \frac{(-1)^k}{k} C_1^k = -e C_1^1 = -e \approx -2.72$$

$$\Delta_{11} = G_{11} - L_{11} = -2.72 - 600 = -602.72$$

$$\Delta_{12} = G_{12} - L_{12} = -2.72 + 600 = 597.28$$

$$\Delta_1 = \sqrt{\Delta_{11}^2 + \Delta_{12}^2} = \sqrt{(-602.72)^2 + (597.28)^2} \approx 848.54$$

Following table III.1.28 shows normal biases of all possible values of  $(a_1, b_1)$ .

$a_1$	$b_1$	$G_{11}$	$G_{12}$	$\Delta_{11}$	$\Delta_{12}$	$\Delta_1$
-------	-------	----------	----------	---------------	---------------	------------

1	1	-2.72	-2.72	-602.72	597.28	848.54
1	2	-7.39	-4.08	-607.39	595.92	850.91
1	3	-20.09	-4.98	-620.09	595.02	859.39
1	4	-54.6	-5.66	-654.6	594.34	884.16
2	1	-4.08	-7.39	-604.08	592.61	846.23
2	2	-11.08	-11.08	-611.08	588.92	848.67
2	3	-30.13	-13.55	-630.13	586.45	860.81
2	4	-81.9	-15.39	-681.9	584.61	898.19
3	1	-4.98	-20.09	-604.98	579.91	838.04
3	2	-13.55	-30.13	-613.55	569.87	837.37
3	3	-36.82	-36.82	-636.82	563.18	850.12
3	4	-100.1	-41.84	-700.1	558.16	895.36
4	1	-5.66	-54.6	-605.66	545.4	815.04
<b>4</b>	<b>2</b>	<b>-15.39</b>	<b>-81.9</b>	<b>-615.39</b>	<b>518.1</b>	<b>804.45</b>
4	3	-41.84	-100.1	-641.84	499.9	813.55
4	4	-113.75	-113.75	-713.75	486.25	863.64

**Table III.1.28.** The normal biases of  $(a_1, b_1)$  with respect to  $\beta_1$

The normal biases of all possible values of  $(a_2, b_2)$  with respect to  $\beta_2$  are shown in following table [III.1.29](#).

$a_2$	$b_2$	$G_{21}$	$G_{22}$	$\Delta_{21}$	$\Delta_{22}$	$\Delta_2$
1	1	-2.72	-2.72	-502.72	497.28	707.12
1	2	-7.39	-4.08	-507.39	495.92	709.49
1	3	-20.09	-4.98	-520.09	495.02	718.00
1	4	-54.60	-5.66	-554.60	494.34	742.93
2	1	-4.08	-7.39	-504.08	492.61	704.81
2	2	-11.08	-11.08	-511.08	488.92	707.28
2	3	-30.13	-13.55	-530.13	486.45	719.49
2	4	-81.90	-15.39	-581.90	484.61	757.26
3	1	-4.98	-20.09	-504.98	479.91	696.65
3	2	-13.55	-30.13	-513.55	469.87	696.07
3	3	-36.82	-36.82	-536.82	463.18	709.02
3	4	-100.10	-41.84	-600.10	458.16	755.00
4	1	-5.66	-54.60	-505.66	445.40	673.85
<b>4</b>	<b>2</b>	<b>-15.39</b>	<b>-81.90</b>	<b>-515.39</b>	<b>418.10</b>	<b>663.66</b>
4	3	-41.84	-100.10	-541.84	399.90	673.44
4	4	-113.75	-113.75	-613.75	386.25	725.17

**Table III.1.29.** The normal biases of  $(a_2, b_2)$  with respect to  $\beta_2$

The normal biases of all possible values of  $(a_3, b_3)$  with respect to  $\beta_3$  are shown in following table [III.1.30](#).

$a_3$	$b_3$	$G_{31}$	$G_{32}$	$\Delta_{31}$	$\Delta_{32}$	$\Delta_3$
1	1	-2.72	-2.72	997.28	-1002.72	1414.22

### III.1. Knowledge sub-model

---

1	2	-7.39	-4.08	992.61	-1004.08	1411.90
1	3	-20.09	-4.98	979.91	-1004.98	1403.65
1	4	-54.60	-5.66	945.40	-1005.66	1380.27
2	1	-4.08	-7.39	995.92	-1007.39	1416.58
2	2	-11.08	-11.08	988.92	-1011.08	1414.30
2	3	-30.13	-13.55	969.87	-1013.55	1402.83
<b>2</b>	<b>4</b>	<b>-81.90</b>	<b>-15.39</b>	<b>918.10</b>	<b>-1015.39</b>	<b>1368.92</b>
3	1	-4.98	-20.09	995.02	-1020.09	1425.00
3	2	-13.55	-30.13	986.45	-1030.13	1426.27
3	3	-36.82	-36.82	963.18	-1036.82	1415.17
3	4	-100.10	-41.84	899.90	-1041.84	1376.69
4	1	-5.66	-54.60	994.34	-1054.60	1449.44
4	2	-15.39	-81.90	984.61	-1081.90	1462.86
4	3	-41.84	-100.10	958.16	-1100.10	1458.86
4	4	-113.75	-113.75	886.25	-1113.75	1423.33

**Table III.1.30.** The normal biases of  $(a_3, b_3)$  with respect to  $\beta_3$

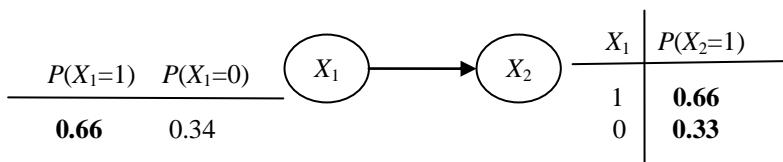
From above tables III.1.28, III.1.29, and III.1.30, we recognize that when  $(a_1, b_1)=(4,2)$ ,  $(a_2, b_2)=(4,2)$  and  $(a_3, b_3)=(2,4)$ , the normal biases of distributions  $\beta_1$ ,  $\beta_2$  and  $\beta_3$ , respectively become minimum. So the parameter estimators  $(\hat{a}_1, \hat{b}_1)$ ,  $(\hat{a}_2, \hat{b}_2)$  and  $(\hat{a}_3, \hat{b}_3)$  corresponding to distributions  $\beta_1$ ,  $\beta_2$  and  $\beta_3$  are (4,2), (4,2) and (2,4), respectively. So the prior conditional probabilities  $P(X_1=1)$ ,  $P(X_2=1|X_1=1)$  and  $P(X_2=1|X_1=0)$  are determined:

$$P(X_1 = 1) = \frac{\hat{a}_1}{\hat{a}_1 + \hat{b}_1} = \frac{4}{4+2} = 0.66$$

$$P(X_2 = 1 | X_1 = 1) = \frac{\hat{a}_2}{\hat{a}_2 + \hat{b}_2} = \frac{4}{4+2} = 0.66$$

$$P(X_2 = 1 | X_1 = 0) = \frac{\hat{a}_3}{\hat{a}_3 + \hat{b}_3} = \frac{2}{4+2} = 0.33$$

When these prior probabilities were calculated, the Bayesian network is totally determined with full of prior CPT (s) as in figure III.1.29.



**Figure III.1.29.** Bayesian network with full of prior CPT (s)

The figure III.1.29 shows the ultimate result of the interesting algorithm mentioned mainly in sub-section III.1.5.3 for specifying prior CPT (s) of Bayesian network. Such interesting algorithm takes advantages of simple equations whose solutions are estimates of positive parameter  $a$  and  $b$ . The successive sub-section III.1.5.5 gives an enjoyable subject that is to recommend a new version of these equations.

### III.1.5.5. New version of the equations whose solutions are parameter estimators

According to formula III.1.73 aforementioned in sub-section III.1.5.3, the parameter estimators  $\hat{a}$  and  $\hat{b}$  are solutions of two equations as follows:

$$\begin{cases} e^a \sum_{k=1}^b \frac{(-1)^k}{k} C_b^k = \frac{1}{n} \sum_{i=1}^n \ln(x_i) \\ e^b \sum_{k=1}^a \frac{(-1)^k}{k} C_a^k = \frac{1}{n} \sum_{i=1}^n \ln(1 - x_i) \end{cases}$$

Although these equations are very simple, a question is issued “Are there another version of these equations?”. This sub-section III.1.5.5 proves that the new version of these equations exists.

As specified by formula III.1.70, the parameter estimators  $\hat{a}$  and  $\hat{b}$  are solutions of two equations:

$$\begin{cases} \psi(a) - \psi(a + b) = \frac{1}{n} \sum_{i=1}^n \ln(x_i) \\ \psi(b) - \psi(a + b) = \frac{1}{n} \sum_{i=1}^n \ln(1 - x_i) \end{cases}$$

According to formula III.1.60, given  $a$  and  $b$  are positive integers, the digamma function  $\psi(x)$  is:

$$\psi(x) = -\gamma + \sum_{k=1}^{x-1} \frac{1}{k}$$

We have:

$$\begin{aligned}
 & \left\{ \begin{array}{l} \psi(a) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(x_i) \\ \psi(b) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(1-x_i) \end{array} \right. \\
 & \Leftrightarrow \left\{ \begin{array}{l} n(\psi(a) - \psi(a+b)) = \sum_{i=1}^n \ln(x_i) \\ n(\psi(b) - \psi(a+b)) = \sum_{i=1}^n \ln(1-x_i) \end{array} \right. \\
 & \Leftrightarrow \left\{ \begin{array}{l} n \left( -\gamma + \sum_{k=1}^{a-1} \frac{1}{k} + \gamma - \sum_{k=1}^{a+b-1} \frac{1}{k} \right) = \sum_{i=1}^n \ln(x_i) \\ n \left( -\gamma + \sum_{k=1}^{b-1} \frac{1}{k} + \gamma - \sum_{k=1}^{a+b-1} \frac{1}{k} \right) = \sum_{i=1}^n \ln(1-x_i) \end{array} \right. \\
 & \Leftrightarrow \left\{ \begin{array}{l} -n \sum_{k=a}^{a+b-1} \frac{1}{k} = \sum_{i=1}^n \ln(x_i) \\ -n \sum_{k=b}^{a+b-1} \frac{1}{k} = \sum_{i=1}^n \ln(1-x_i) \end{array} \right. \\
 & \Rightarrow \left\{ \begin{array}{l} \exp \left( -n \sum_{k=a}^{a+b-1} \frac{1}{k} \right) = \prod_{i=1}^n x_i \\ \exp \left( -n \sum_{k=b}^{a+b-1} \frac{1}{k} \right) = \prod_{i=1}^n (1-x_i) \end{array} \right.
 \end{aligned}$$

(By taking exponent function of both sides of these equations)

Briefly, the parameter estimators  $\hat{a}$  and  $\hat{b}$  are solutions of two following equations specified by formula III.1.74.

$$\left\{ \begin{array}{l} \exp \left( -n \sum_{k=a}^{a+b-1} \frac{1}{k} \right) = \prod_{i=1}^n x_i \\ \exp \left( -n \sum_{k=b}^{a+b-1} \frac{1}{k} \right) = \prod_{i=1}^n (1-x_i) \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} G_1(a, b) = L_1 \\ G_2(a, b) = L_2 \end{array} \right.$$

*Formula III.1.74.* New version of equations for estimating positive integer parameters  $a$  and  $b$

Where,

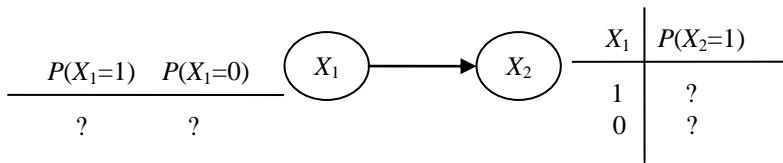
$$0 \leq x \leq 1$$

$$G_1(a, b) = \exp\left(-n \sum_{k=a}^{a+b-1} \frac{1}{k}\right) \text{ and } G_2(a, b) = \exp\left(-n \sum_{k=b}^{a+b-1} \frac{1}{k}\right)$$

$$L_1 = \prod_{i=1}^n x_i \text{ and } L_2 = \prod_{i=1}^n (1 - x_i)$$

Formula III.1.74 is not absolutely simpler than formula III.1.73 but it is more convenient to evaluate formula III.1.74 because logarithm functions are removed from formula III.1.74. The iterative algorithm described in table III.1.25 is applied into formula III.1.74 in order to find out parameter estimators  $\hat{a}$  and  $\hat{b}$ .

It is necessary to give an example for illustrating the iterative algorithm with regard to formula III.1.74. Going back previous example shown in figure III.1.28, recall that there is the BN having two variables  $X_1$ ,  $X_2$  and one arc which links them together. We need to specify the prior CPT (s), namely the prior conditional probabilities  $P(X_1=1)$ ,  $P(X_2=1|X_1=1)$  and  $P(X_2=1|X_1=0)$ . For convenience, figure III.1.30 is the replication of figure III.1.28.



**Figure III.1.30.** Bayesian network without CPT (s)

In figure III.1.30, question marks (?) indicate undefined conditional probabilities.

Let  $\beta_1(a_1, b_1)$ ,  $\beta_2(a_2, b_2)$ ,  $\beta_3(a_3, b_3)$  be beta distributions of conditional probabilities  $P(X_1=1)$ ,  $P(X_2=1|X_1=1)$  and  $P(X_2=1|X_1=0)$ . Applying formula III.1.50 for specifying probability of  $X$  as expectation of beta distribution, we have:

$$P(X_1 = 1) = E(\beta_1(a_1, b_1)) = \frac{a_1}{a_1 + b_1}$$

$$P(X_2 = 1 | X_1 = 1) = E(\beta_2(a_2, b_2)) = \frac{a_2}{a_2 + b_2}$$

$$P(X_2 = 1 | X_1 = 0) = E(\beta_3(a_3, b_3)) = \frac{a_3}{a_3 + b_3}$$

It is necessary to determine three parameter pairs  $(a_1, b_1)$ ,  $(a_2, b_2)$  and  $(a_3, b_3)$  of three beta distributions  $\beta_1$ ,  $\beta_2$  and  $\beta_3$ , respectively. Suppose we perform 5 trials of a random process, the outcome of  $i^{th}$  trial denoted  $D^{(i)}$  is considered as an evidence in which  $X_1$  and  $X_2$  obtains value 0 or 1. So we have the vector of 5 evidences  $\mathcal{D} = (D^{(1)}, D^{(2)}, D^{(2)}, D^{(3)}, D^{(4)}, D^{(5)})$ . For convenience, table III.1.31 is replication of table III.1.26 showing these evidences.

	$X_1$	$X_2$
$D^{(1)}$	$X_1 = 1$	$X_2 = 1$
$D^{(2)}$	$X_1 = 1$	$X_2 = 1$

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

**Table III.1.31.** The evidences corresponding to 5 trials

According to the proposed algorithm described in table III.1.25, let  $L_{ij}$ ,  $G_{ij}$ ,  $\Delta_{ij}$ ,  $\Delta_i$  be the values of  $L_j$ ,  $G_j$ ,  $\Delta_j$ ,  $\Delta$  with respect to  $\beta_i$  where  $i = \overline{1,3}$  and  $j = \overline{1,2}$ . We have:

- $L_{11} = \prod_{i=1}^n x_i$  and  $L_{12} = \prod_{i=1}^n (1 - x_i)$  where  $x_i$  is the instances of  $X_1$ .
- $L_{21} = \prod_{i=1}^n x_i$  and  $L_{22} = \prod_{i=1}^n (1 - x_i)$  where  $x_i$  is the instances of  $X_2$  given  $X_1=1$ .
- $L_{31} = \prod_{i=1}^n x_i$  and  $L_{32} = \prod_{i=1}^n (1 - x_i)$  where  $x_i$  is the instances of  $X_2$  given  $X_1=0$ .
- $G_{11}(a_1, b_1) = \exp\left(-n \sum_{k=a_1}^{a_1+b_1-1} \frac{1}{k}\right)$  and  $G_{12}(a_1, b_1) = \exp\left(-n \sum_{k=b_1}^{a_1+b_1-1} \frac{1}{k}\right)$
- $G_{21}(a_2, b_2) = \exp\left(-n \sum_{k=a_2}^{a_2+b_2-1} \frac{1}{k}\right)$  and  $G_{22}(a_2, b_2) = \exp\left(-n \sum_{k=b_2}^{a_2+b_2-1} \frac{1}{k}\right)$
- $G_{31}(a_3, b_3) = \exp\left(-n \sum_{k=a_3}^{a_3+b_3-1} \frac{1}{k}\right)$  and  $G_{32}(a_3, b_3) = \exp\left(-n \sum_{k=b_3}^{a_3+b_3-1} \frac{1}{k}\right)$
- $\Delta_{11} = G_{11} - L_{11}$ ,  $\Delta_{12} = G_{12} - L_{12}$  and  $\Delta_1 = \sqrt{\Delta_{11}^2 + \Delta_{12}^2}$
- $\Delta_{21} = G_{21} - L_{21}$ ,  $\Delta_{22} = G_{22} - L_{22}$  and  $\Delta_2 = \sqrt{\Delta_{21}^2 + \Delta_{22}^2}$
- $\Delta_{31} = G_{31} - L_{31}$ ,  $\Delta_{32} = G_{32} - L_{32}$  and  $\Delta_3 = \sqrt{\Delta_{31}^2 + \Delta_{32}^2}$

Let  $D_1$ ,  $D_2$ , and  $D_3$  be the set of  $x_i$  (s) that are instances of  $X_1$ ,  $X_2$  given  $X_1=1$ , and  $X_2$  given  $X_1=0$ , respectively. From evidences expressed in table III.1.31, we have:

$$\begin{aligned} D_1 &= \{x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1, x_5 = 0\} \\ D_2 &= \{x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0\} \\ D_3 &= \{x_1 = 0\} \end{aligned}$$

Each instance  $x_i$  will be modified so that products  $\prod_{i=1}^n x_i$  and  $\prod_{i=1}^n (1 - x_i)$  avoid getting zero frequently.

- If  $x_i$  equals 1, it is subtracted by a very small number  $\varepsilon$ , for example, given  $\varepsilon=0.1$ ,  $x_i = x_i - 0.1 = 1 - 0.1 = 0.9$ .
- If  $x_i$  equals 0, it is added by a very small number  $\varepsilon$ , for example,  $\varepsilon=0.1$ ,  $x_i = x_i + 0.1 = 0 + 0.1 = 0.1$ .

Thus, we have:

$$\begin{aligned} D_1 &= \{x_1 = 0.9, x_2 = 0.9, x_3 = 0.9, x_4 = 0.9, x_5 = 0.1\} \\ D_2 &= \{x_1 = 0.9, x_2 = 0.9, x_3 = 0.9, x_4 = 0.1\} \\ D_3 &= \{x_1 = 0.1\} \end{aligned}$$

Suppose the range of all parameters is from 1 to 4. By applying the proposed algorithm described in table III.1.25, it is easy to compute the normal biases. For example, given  $a_1 = 1$  and  $b_1 = 1$ , we have:

$$L_{11}(a_1 = 1, b_1 = 1) = \prod_{x_i \in D_1} x_i = 0.9 * 0.9 * 0.9 * 0.9 * 0.1 \approx 0.0656$$

$$\begin{aligned}
 L_{12}(a_1 = 1, b_1 = 1) &= \prod_{x_i \in D_1} (1 - x_i) \\
 &= (1 - 0.9) * (1 - 0.9) * (1 - 0.9) * (1 - 0.9) * (1 - 0.1) \\
 &\approx 0.0001 \\
 G_{11}(a_1 = 1, b_1 = 1) &= \exp\left(-5 \sum_{k=1}^{1+1-1} \frac{1}{k}\right) = \exp\left(-5 * \frac{1}{1}\right) \approx 0.0067 \\
 G_{12}(a_1 = 1, b_1 = 1) &= \exp\left(-5 \sum_{k=1}^{1+1-1} \frac{1}{k}\right) = \exp\left(-5 * \frac{1}{1}\right) \approx 0.0067 \\
 \Delta_{11} &= G_{11} - L_{11} = 0.0067 - 0.0656 \approx -0.0589 \\
 \Delta_{12} &= G_{12} - L_{12} = 0.0067 - 0.0001 \approx 0.0066 \\
 \Delta_1 &= \sqrt{\Delta_{11}^2 + \Delta_{12}^2} = \sqrt{(-0.0589)^2 + (0.0066)^2} \approx 0.0592
 \end{aligned}$$

Following table III.1.32 shows normal biases of all possible values of  $(a_1, b_1)$ .

<b><math>a_1</math></b>	<b><math>b_1</math></b>	<b><math>L_{11}</math></b>	<b><math>L_{12}</math></b>	<b><math>G_{11}</math></b>	<b><math>G_{12}</math></b>	<b><math>\Delta_{11}</math></b>	<b><math>\Delta_{12}</math></b>	<b><math>\Delta_1</math></b>
1	1	0.0656	0.0001	0.0067	0.0067	-0.0589	0.0066	0.0592
1	2	0.0656	0.0001	0.0006	0.0821	-0.0651	0.0820	0.1047
1	3	0.0656	0.0001	0.0001	0.1889	-0.0655	0.1888	0.1998
1	4	0.0656	0.0001	0.0000	0.2865	-0.0656	0.2864	0.2938
2	1	0.0656	0.0001	0.0821	0.0006	0.0165	0.0005	0.0165
2	2	0.0656	0.0001	0.0155	0.0155	-0.0501	0.0154	0.0524
2	3	0.0656	0.0001	0.0044	0.0541	-0.0612	0.0540	0.0816
2	4	0.0656	0.0001	0.0016	0.1054	-0.0640	0.1053	0.1232
3	1	0.0656	0.0001	0.1889	0.0001	0.1233	0.0000	0.1233
<b>3</b>	<b>2</b>	<b>0.0656</b>	<b>0.0001</b>	<b>0.0541</b>	<b>0.0044</b>	<b>-0.0115</b>	<b>0.0044</b>	<b>0.0123</b>
3	3	0.0656	0.0001	0.0199	0.0199	-0.0457	0.0198	0.0498
3	4	0.0656	0.0001	0.0087	0.0458	-0.0570	0.0457	0.0730
4	1	0.0656	0.0001	0.2865	0.0000	0.2209	-0.0001	0.2209
4	2	0.0656	0.0001	0.1054	0.0016	0.0398	0.0015	0.0398
4	3	0.0656	0.0001	0.0458	0.0087	-0.0198	0.0086	0.0216
4	4	0.0656	0.0001	0.0224	0.0224	-0.0432	0.0223	0.0486

**Table III.1.32.** The normal biases of  $(a_1, b_1)$  with respect to  $\beta_1$

The normal biases of all possible values of  $(a_2, b_2)$  with respect to  $\beta_2$  are shown in following table III.1.33.

<b><math>a_2</math></b>	<b><math>b_2</math></b>	<b><math>L_{21}</math></b>	<b><math>L_{22}</math></b>	<b><math>G_{21}</math></b>	<b><math>G_{22}</math></b>	<b><math>\Delta_{21}</math></b>	<b><math>\Delta_{22}</math></b>	<b><math>\Delta_2</math></b>
1	1	0.0729	0.0009	0.0183	0.0183	-0.0546	0.0174	0.0573
1	2	0.0729	0.0009	0.0025	0.1353	-0.0704	0.1344	0.1518
1	3	0.0729	0.0009	0.0007	0.2636	-0.0722	0.2627	0.2725
1	4	0.0729	0.0009	0.0002	0.3679	-0.0727	0.3670	0.3741
2	1	0.0729	0.0009	0.1353	0.0025	0.0624	0.0016	0.0625
2	2	0.0729	0.0009	0.0357	0.0357	-0.0372	0.0348	0.0509

### III.1. Knowledge sub-model

---

2	3	0.0729	0.0009	0.0131	0.0970	-0.0598	0.0961	0.1132
2	4	0.0729	0.0009	0.0059	0.1653	-0.0670	0.1644	0.1775
3	1	0.0729	0.0009	0.2636	0.0007	0.1907	-0.0002	0.1907
3	2	0.0729	0.0009	0.0970	0.0131	0.0241	0.0122	0.0270
3	3	0.0729	0.0009	0.0436	0.0436	-0.0293	0.0427	0.0518
3	4	0.0729	0.0009	0.0224	0.0849	-0.0505	0.0840	0.0980
4	1	0.0729	0.0009	0.3679	0.0002	0.2950	-0.0007	0.2950
4	2	0.0729	0.0009	0.1653	0.0059	0.0924	0.0050	0.0925
<b>4</b>	<b>3</b>	<b>0.0729</b>	<b>0.0009</b>	<b>0.0849</b>	<b>0.0224</b>	<b>0.0120</b>	<b>0.0215</b>	<b>0.0246</b>
4	4	0.0729	0.0009	0.0479	0.0479	-0.0250	0.0470	0.0532

**Table III.1.33.** The normal biases of  $(a_2, b_2)$  with respect to  $\beta_2$

The normal biases of all possible values of  $(a_3, b_3)$  with respect to  $\beta_3$  are shown in following table [III.1.34](#).

<b><math>a_3</math></b>	<b><math>b_3</math></b>	<b><math>L_{31}</math></b>	<b><math>L_{32}</math></b>	<b><math>G_{31}</math></b>	<b><math>G_{32}</math></b>	<b><math>\Delta_{31}</math></b>	<b><math>\Delta_{32}</math></b>	<b><math>\Delta_3</math></b>
1	1	0.1	0.9	0.3679	0.3679	0.2679	-0.5321	0.5957
1	2	0.1	0.9	0.2231	0.6065	0.1231	-0.2935	0.3183
1	3	0.1	0.9	0.1599	0.7165	0.0599	-0.1835	0.1930
<b>1</b>	<b>4</b>	<b>0.1</b>	<b>0.9</b>	<b>0.1245</b>	<b>0.7788</b>	<b>0.0245</b>	<b>-0.1212</b>	<b>0.1237</b>
2	1	0.1	0.9	0.6065	0.2231	0.5065	-0.6769	0.8454
2	2	0.1	0.9	0.4346	0.4346	0.3346	-0.4654	0.5732
2	3	0.1	0.9	0.3385	0.5580	0.2385	-0.3420	0.4169
2	4	0.1	0.9	0.2771	0.6376	0.1771	-0.2624	0.3166
3	1	0.1	0.9	0.7165	0.1599	0.6165	-0.7401	0.9633
3	2	0.1	0.9	0.5580	0.3385	0.4580	-0.5615	0.7246
3	3	0.1	0.9	0.4569	0.4569	0.3569	-0.4431	0.5690
3	4	0.1	0.9	0.3867	0.5397	0.2867	-0.3603	0.4604
4	1	0.1	0.9	0.7788	0.1245	0.6788	-0.7755	1.0306
4	2	0.1	0.9	0.6376	0.2771	0.5376	-0.6229	0.8228
4	3	0.1	0.9	0.5397	0.3867	0.4397	-0.5133	0.6759
4	4	0.1	0.9	0.4679	0.4679	0.3679	-0.4321	0.5675

**Table III.1.34.** The normal biases of  $(a_3, b_3)$  with respect to  $\beta_3$

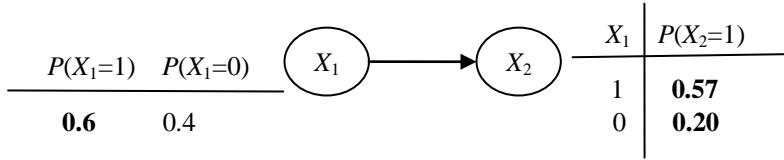
From above tables [III.1.32](#), [III.1.33](#), and [III.1.34](#), we recognize that when  $(a_1, b_1)=(3,2)$ ,  $(a_2, b_2)=(4,3)$  and  $(a_3, b_3)=(1,4)$ , the normal biases of distributions  $\beta_1$ ,  $\beta_2$  and  $\beta_3$ , respectively become minimum. So the parameter estimators  $(\hat{a}_1, \hat{b}_1)$ ,  $(\hat{a}_2, \hat{b}_2)$  and  $(\hat{a}_3, \hat{b}_3)$  corresponding to distributions  $\beta_1$ ,  $\beta_2$  and  $\beta_3$  are  $(3,2)$ ,  $(4,3)$  and  $(1,4)$ , respectively. So the prior conditional probabilities  $P(X_1=1)$ ,  $P(X_2=1|X_1=1)$  and  $P(X_2=1|X_1=0)$  are determined:

$$P(X_1 = 1) = \frac{\hat{a}_1}{\hat{a}_1 + \hat{b}_1} = \frac{3}{3+2} = 0.6$$

$$P(X_2 = 1 | X_1 = 1) = \frac{\hat{a}_2}{\hat{a}_2 + \hat{b}_2} = \frac{4}{4+3} \approx 0.57$$

$$P(X_2 = 1 | X_1 = 0) = \frac{\hat{a}_3}{\hat{a}_3 + \hat{b}_3} = \frac{1}{1 + 4} = 0.2$$

When these prior probabilities were calculated, the Bayesian network is totally determined with full of prior CPT (s) as in figure III.1.31.



**Figure III.1.31.** Bayesian network with full of prior CPT (s)

The figure III.1.31 shows the ultimate result of applying the iterative algorithm mentioned in sub-section III.1.5.3 into the simple equations specified by formula III.1.74 in this sub-section III.1.5.5. It is easy to recognize that this result is similar to the one (shown in figure III.1.29) that is produced from equations specified by formula III.1.73. The deviation between two results is caused by computational bias.

In general, the iterative algorithm for solving simple equations specified by formulas III.1.73 and III.1.74 is the result of applying MLE method into beta density function. Sub-section III.1.5.6 is evaluation of the iterative algorithm which is considered as an implementation of MLE method for specifying prior probabilities of BN.

### III.1.5.6. Evaluation

The basic idea of MLE is to solve the equation formed by setting the first-order derivation of log-likelihood function equal 0. I apply MLE into beta distribution so as to determine the equations for computing two parameters of beta distribution. Because beta distribution is more complex to estimate its parameters than other distributions like binomial distribution and normal distribution, I invent the simple form of MLE equations in case that parameters are positive integer numbers. Moreover, I propose the algorithm so as to find out approximate solutions of these equations. This is iterative algorithm in which a number of parameters are surveyed and the parameter whose bias with respect to the actual solution is minimum is the approximate solution. It is impossible to find out the precise solutions but it is easy and feasible to implement our algorithm as computer program. Although the simple form of MLE equations and the iterative algorithm are my two contributions but the most important is the invention of simple MLE equations. The ideology behind the simple equations is that we focus on finding out discrete parameters instead of making effort to solve differential equations; moreover, the proposed iterative algorithm is only appropriate to such ideology. I think that the invention is worthy to publish worldwide. Additionally, the new version of these simple equations is also proposed, which gives convenience for solving these equations.

### III.1. Knowledge sub-model

---

In comparison with dynamic Bayesian network (DBN) method, this MLE approach is simpler but it cannot monitor chronologically users' process of gaining knowledge and the convergence of DBN gives the best prediction on users' mastery over learning materials. MLE method is appropriate to static Bayesian network associated with available training data.

This sub-section III.1.5 ends up the comprehensive description of knowledge sub-model – the apex of [Triangular Learner Model](#) (TLM), which represents domain-specific user information. The next section III.2 describes learning style sub-model – another apex of TLM, which represents [domain-independent information](#). Similarly, I also use hidden Markov model to construct learning style sub-model with attention that both Bayesian network (see sub-section III.1.1.1) and hidden Markov model (see sub-section III.2.4) are kinds of belief network (Murphy, 1998) and so, knowledge sub-model and learning style sub-model are managed by belief network engine (BNE) inside user modeling system [Zebra](#). Releasing your curiousness and surveying learning style sub-model with next section III.2.

## III.2. Learning style sub-model

Adaptive learning systems are developed rapidly in recent years and the “heart” of such systems is user model. Recall that user model aforementioned in section I.1 is the representation of information about an individual that is essential for an adaptive system to provide the adaptation effect, i.e., to behave differently for different users. There are some main features in user model such as knowledge, goals, learning styles, interests, background... but knowledge, learning styles and goals are features attracting researchers’ attention in adaptive e-learning domain. Contrary to knowledge model described in previous section III.1 focusing on domain-specific information about users, learning styles represent domain-independent information. Learning styles were surveyed in psychological theories but it is slightly difficult to model them in the domain of computer science because learning styles are too unobvious to represent them and there is no solid inference mechanism for discovering users’ learning styles now. Moreover, researchers in domain of computer science will get confused by so many psychological theories about learning style when choosing which theory is appropriate to adaptive system.

In this section III.2, I give the overview of learning styles for answering the question “what are learning styles?” and then propose the new approach to model and discover students’ learning styles by using hidden Markov model (HMM). In other words, learning style sub-model is structured by HMM. HMM is such a powerful statistical tool that it allows us to predict users’ learning styles from observed evidences about them; please see sub-section III.2.4 for more details about HMM.

In general, learning style sub-model is one among three sub-models that constituting Triangular Learner Model (TLM). It and knowledge sub-model mention in previous section III.1 are managed by belief network engine (BNE). Please see section II.2 for more details about Triangular Learner Model and belief network engine. Now it is necessary to start reading sub-section III.2.1 as an introduction of some basic concepts of learning styles.

### III.2.1. What learning styles are

People have different views upon the same situation, the way they perceive and estimate the world is different. So their responses to around environment are also different. For example, look at the way students prefers to study a lesson. Some have a preference for listening to instructional content (so-called *auditory* learner), some for perceiving materials as picture (*visual* learner), some for interacting physically with learning material (*tactile kinesthetic* learner), some for making connections to personal and to past learning experiences (*internal kinesthetic* learner). Such characteristics about user cognition are called learning styles but learning styles are wider than what we think about them.

Learning styles are defined as “the composite of characteristic cognitive, affective and psychological factors that serve as relatively stable indicators of

how a learner perceives, interacts with and responds to the learning environment” (Stash, 2007, p. 93). Learning style is the important factor in adaptive learning, which is the navigator helping teacher/computer to deliver the best instructions to students.

There are many researches and descriptions about learning style but only minorities of them are valuable and applied widely in adaptive learning. The descriptions of learning style (so-called learning style models) are categorized criteria such as theoretical importance, wide spread use, and influence on other learning style models (Stash, 2007, p. 94). Learning style models are organized within the families as follows (Stash, 2007, p. 95):

- Constitutionally based learning styles and preferences: Dunn and Dunn model (Dunn & Dunn, 1996).
- The cognitive structure: Witkin model (Witkin, Moore, Goodenough, & Cox, 1977) and Riding model (Riding & Rayner, 1998).
- Stable personality type: Myers-Briggs Type Indicator (Wikipedia, Myers-Briggs Type Indicator, 2014).
- Flexibly learning preferences: Kolb’s Learning Style Inventory, Honey and Mumford model, Felder-Silverman model, Pask model, and Vermunt model.

In next sub-section [III.2.2](#), we discuss about such learning style families. In general, learning styles are analyzed comprehensively in theory of psychology but there are few of researches on structuring learning styles by mathematical tools to predict/infer users’ styles. Former researches often give users questionnaires and then analyze their answers in order to discover their styles but there are so many drawbacks of question-and-answer techniques, i.e., not questions enough, confusing questions, and users’ wrong answers. Hence, such technique is not a possible solution. It is essential to use another technique that provides more powerful inference mechanism. So, I propose the new approach which uses hidden Markov model to discover and represent users’ learning styles in sub-section [III.2.5](#). Sub-section [III.2.6](#) is the evaluation of proposed approach. We should pay attention to some issues of providing adaptation of learning materials to learning styles concerned in sub-section [III.2.3](#). Additionally, it is necessary to read sub-section [III.2.4](#) which is the brief survey of hidden Markov model (HMM).

#### **III.2.2. Learning style families**

As aforementioned, there are four learning style families such as constitutionally based learning styles and preferences, cognitive structure, stable personality type, and flexibly learning preferences which are described briefly as below.

##### **III.2.2.1. Constitutionally based learning styles and preferences**

Learning styles in this family are fixed and difficult to change (Stash, 2007, p. 95). This family has the famous model “Dunn and Dunn model” developed by

authors Rita Dunn and Kenneth Dunn (Dunn & Dunn, 1996). With Dunn and Dunn model, learning style is divided into 5 major strands (Stash, 2007, pp. 95-96):

- *Environmental*: incorporates user preferences for sound, light, temperature, etc.
- *Emotional*: considers user motivation, persistence, responsibility, etc.
- *Sociological*: discovers user preference for learning alone, in pairs, as member of group.
- *Physiological*: surveys perceptual strengths such as visual, auditory, kinesthetic, and tactile.
- *Psychological*: focusing on user's psychological traits namely incorporates the information-processing elements of global versus analytic and impulsive versus reflective behaviors.

The physiological strand classifies learning styles into modalities as below:

- *Auditory*: Preference to listen to instructional content.
- *Visual (Picture)*: Preference to perceive materials as pictures.
- *Verbal (Text)*: Preference to perceive materials as text.
- *Tactile Kinesthetic*: Preference to interact physically with learning material.
- *Internal Kinesthetic*: Preference to make connections to personal and to past learning experiences.

The psychological strand classifies learning styles into modalities as below:

- *Impulsive*: Preference to try out new material immediately.
- *Reflective*: Preference to take time to think about a problem.
- *Global*: Preference to get the 'big picture' first, details second.
- *Analytical*: Preference to process information sequentially: details first, working towards the 'big picture'.

### **III.2.2.2. The cognitive structure**

In this family, learning styles are considered as "structural properties of cognitive system itself" (Stash, 2007, p. 98). So styles are linked to particular personality features, which implicates that cognitive styles are deeply embedded in personality structure (Stash, 2007, p. 98). There are two models in this family: Witkin model (Witkin, Moore, Goodenough, & Cox, 1977) and Riding model (Riding & Rayner, 1998).

#### **Witkin model**

The main aspect in Witkin model (Witkin, Moore, Goodenough, & Cox, 1977) is the bipolar dimensions of *field-dependence / field-independence* (FD/FI) in which:

- *Field-dependence* (FD) person process information globally and attend to the most prominent cues regardless of their relevance (Stash, 2007, p. 99). In general, they see the global picture, ignore details and approach the task more holistically. They often get confused with non-linear learning, so, they require guided navigation in hypermedia space.

### III.2. Learning style sub-model

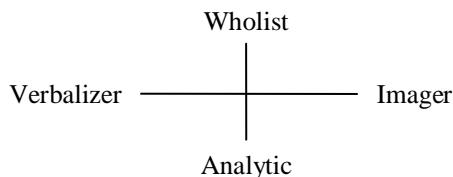
---

- *Field-independency* (FI) person are highly analytic, care more inherent cues in the field and are able to extract the relevant cues necessary to complete a task (Stash, 2007, p. 99). In general, they focus on details and learn more sequentially. They can set learning path themselves and have no need of guidance.

#### Riding model

Riding model (Riding & Rayner, 1998) identifies learning styles into two dimensions: *Wholist-Analytic* and *Verbalizer-Imager*, as seen in figure III.2.1 (Stash, 2007, p. 100).

- *Wholist-Analytic* dimension expresses how an individual cognitively organize information into either whole or parts (Stash, 2007, p. 100). *Wholist* tends to perceive globally before focusing on details. Otherwise, *analytic* tends to perceive everything as the collection of parts and focusing on such parts.
- *Verbalizer-Imager* dimension expresses how an individual tends to perceive information, as either text or picture. *Verbalizer* prefers to text. *Imager* prefers to picture.



**Figure III.2.1.** Two dimensions in Riding model

#### III.2.2.3. Stable personal type

The models in this family have a common focus upon learning style as one part of the observable expression of a relatively stable personality type (Stash, 2007, p. 101). We will glance over the famous model in this family: Myers-Briggs Type Indicator. The Myers-Briggs Type Indicator developed by authors Katharine Cook Briggs and Isabel Briggs Myers (Wikipedia, Myers-Briggs Type Indicator, 2014), first published in 1921, involves four different pairs of opposite preferences for how person focus and interact with around environment (Stash, 2007, pp. 101-102):

1. How does a person relate to the world?
  - a. *Extravert*: try things out, focus on the world around, like working in teams.
  - b. *Introvert*: think things through, focus on the inner world of ideas, prefer to work alone.
2. How does a person absorb/process information?
  - a. *Sensor*: concrete, realistic, practical, detail-oriented, focus on events and procedures.

- b. *Intuitive*: abstract, imaginative, concept-oriented, focus on meanings and possibilities.
- 3. How does a person make decisions?
  - a. *Thinker*: skeptical, tend to make decisions based on logic and rules.
  - b. *Feeler*: appreciative, tend to make decisions based on personal and human considerations.
- 4. How does a person manage her/his life?
  - a. *Judger*: organized, set and follow agendas, make decisions quickly.
  - b. *Perceiver*: disorganized, adapt to change environment, gather more information before making a decision.

### **III.2.2.4. Flexible stable learning preference**

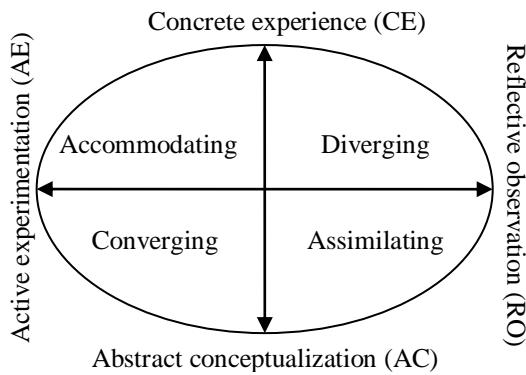
With models in this family, learning style is not a fixed trait but is a differential preference for learning, which changes slightly from situation to situation (Stash, Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System, 2007, p. 102). There are five typical models in this family: Kolb's Learning Style Inventory, Honey and Mumford model, Felder-Silverman model, Pask model, and Vermunt model.

#### **Kolb Learning Style Inventory**

Kolb Learning Style Inventory also known as Kolb's model was developed by David A. Kolb, an American educational theorist (Wikipedia, David A. Kolb, 2014). The first version of Kolb's model was published in 1969; advanced versions 2, 2a, 3, and 3.1 were published in 1985, 1993, 1999, and 2005, respectively (Kolb & Kolb, 2005). The author David A. Kolb stated that "Learning is the process whereby knowledge is created through the transformation of experience. Knowledge results from the combination of grasping experience and transforming it" (Stash, 2007, p. 103). The center of Kolb's model is the four-stage cycle of learning which contains four stages in learning process: *Concrete Experience* (CE - feeling), *Abstract Conceptualization* (AC - thinking), *Active Experimentation* (AE - doing), and *Reflective Observation* (RO - watching) (Stash, 2007, p. 103). The four-stage cycle is concretized as below:

1. Learner makes acquainted with the concrete situation, accumulates the experience (CE- feeling).
2. Learner observes reflectively (RO - watching) herself/himself.
3. She/he conceptualizes what she/he watches (observations) into abstract concepts (AC - thinking).
4. She/he experiments actively such concepts and gets the new experience (AE - doing). The cycle repeats again.

Figure III.2.2 (Stash, 2007, p. 103) depicts this four-stage cycle in learning process.



**Figure III.2.2.** Four-stage learning process of Kolb's model

Based on four stages, there are four learning styles: accommodating, assimilating, diverging and converging. Each couple of these stages constitutes a style, for example, CE and AE combine together in order to generate accommodating style.

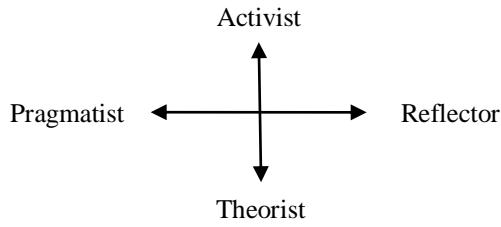
- *Accommodating* (CE/AE): emphasizes concrete experience and active experimentation (Stash, 2007, p. 104). Learners prefer to apply learning material in new situations so that they solve real problems. A typical question for this style is “What if?”.
- *Assimilating* (AC/RO): prefers abstract conceptualization and reflective observation (Stash, 2007, p. 104). Learners respond to information presented in an organized, logical fashion and benefit if they have time for reflection. A typical question for this style is “What?”.
- *Converging* (AC/AE): relies primarily on abstract conceptualization and active experimentation (Stash, 2007, p. 104). Learners respond to having opportunities to work actively on well-defined tasks and to learn by trial-and-error in an environment that allows them to fail safely. A typical question for this style is “How?”.
- *Diverging* (CE/RO): emphasizes concrete experience and reflective observation (Stash, 2007, p. 104). Learners respond well to explanations of how course material relates to their experience, their interests, and their future careers. A typical question for this style is “Why?”.

### Honey and Mumford model

According to Peter Honey and Alan Mumford (Honey & Mumford, 2000), the authors of this model, there are four learning styles (Stash, 2007, pp. 105-106):

- *Activist*: learners are open-minded and comprehend new information by doing something with it.
- *Reflector*: learners prefer to think about new information first before acting on it.
- *Theorist*: learners think things through in logical steps, assimilate different facts into coherent theory.
- *Pragmatist*: learners have practical mind, prefer to try and test techniques relevant to problems.

Figure III.2.3 depicts learning styles in Honey and Mumford model.



**Figure III.2.3.** Learning styles in Honey and Mumford model

### Felder-Silverman model

This model developed by Felder and Silverman (Felder & Silverman, 1988) involves following dimensions:

- *Active/Reflective*. Active students understand information only if they discussed it, applied it. Reflective students think thoroughly about things before doing any practice.
- *Sensing/Intuitive*. Sensing students learn from concrete tasks related to problems and facts that could be solved by well-behaved methods. They are keen on details. Intuitive students discover alternate possibilities and relationships by themselves, working with abstractions and formulas.
- *Verbal/Visual*. Verbal students like learning materials in text form. Otherwise visual student prefer to images, pictures, videos, etc.
- *Sequential/Global*. Sequential students structure their learning process by logically chained steps, each step following from previous one. Global students prefer to learn in random jumps. They can solve complicated problem but don't know clearly how they did it.

### Pask model

Pask model developed by author Pask (Pask, 1976) who stated that there are two distinguishable strategies that learners prefer to learn (Stash, 2007, p. 108):

- *Wholist*: Learners understand problems by building up a global view.
- *Serialist*: Learners prefer to details of activities, facts and follow a step-by-step learning procedure.

### Vermunt model

According to Vermunt (Vermunt, 1996), the author of this model, there are four learning styles (Stash, 2007, p. 108):

- *Meaning-oriented*: Learners prefer to get theory before go to examples. This style is similar to assimilating style of Kolb's model.
- *Application-directed*: Learners prefer to know the purpose of information before get theory. This style is similar to accommodating style of Kolb's model.
- *Undirected*: similar to FD style of Wikin model.
- *Reproduction-oriented*: similar to FI style of Wikin model.

Now learning styles and their models were briefly introduced with essential concepts which are very necessary for us to understand application or role of learning styles in learning adaptation and user modeling. Some issues of providing adaptation of learning materials to learning styles concerned in next sub-section [III.2.3](#).

#### **III.2.3. Providing adaptation of learning materials to learning styles**

Learning styles are discovered and explored in psychological domain but how they are incorporated into adaptive systems? We must solve the problem of “matching” learning materials with users’ learning styles. The teacher must recognize styles of students and then provide individually them teaching methods associated personal learning materials such as lessons, exercises, and test. Such teaching method is called learning strategy or instructional strategy or adaptive strategy. Although there are many learning style models but they share some common features, such as the modality *visual (text)/visual (picture)* in Dunn and Dunn model is similar to *verbalizer/imager* dimension in Riding model and *verbal-visual* dimension in Felder-Silverman model. Strategies are supposed according to common features of model because it is too difficult to describe comprehensively all features of model. Features of all models (learning styles) can be categorized into two groups: perception group and understanding group which are enumerated together with adaptive strategies as below:

**Perception group:** This group related learners’ perception includes:

- The *visual(picture) / visual(text)* modality in Dunn and Dunn model is similar to the *verbalizer/imager* dimension in Riding model and *verbal-visual* dimension in Felder-Silverman model. Instructional strategy is that the teacher should recommend textual materials to verbalizer and pictorial materials to imager.
- The *sensing/intuitive* dimension in Felder-Silverman model is identical to the *sensor/intuitive* dimension in Myer Briggs Type Indicator. Sensing learners are recommended examples before expositions, otherwise, expositions before examples for intuitive learners
- The *perceptive-judging* dimension in Myer Briggs Type Indicator. Perceptive learners are provided rich media such as the integrative use of pictures, tables and diagram. Otherwise, judging learners are provided lean materials.
- The *impulsive/reflective* modality in Dunn and Dunn model is similar to the *activist/reflector* dimension in Honey and Mumford model, the *active/reflective* dimension in Felder-Silverman model and the *extravert/introvert* of Myers-Briggs Type Indicator. Active (also impulsive, extravert) learners are provided activity-oriented approach: showing content of activity and links to example, theory and exercise. Reflective (also introvert) learners are provided example-oriented

approach: showing content of example and links to theory, exercise and activity.

- The *theorist/pragmatist* dimension of Honey and Mumford model. Theorists are provided theory-oriented approach: showing content of theory and links to example, exercise and activity. Pragmatists are provided exercise-oriented approach: showing content of exercise and links to example, theory and activity.
- The *accommodating/assimilating* dimension of Kolb model is similar to *application-directed/ meaning-oriented* dimension of Vermunt model. The adaptive strategy for accommodating style is to provide application-based information to learners; otherwise, theory-based information for assimilating style.

**Understanding group:** This group related to the way learners comprehend knowledge includes:

- The *global/analytical* modality in Dunn and Dunn model is similar to *wholist-analytic* dimension in Riding model, *global/sequential* dimension in Felder-Silverman model, *wholist-serialist* dimension in Pask model. Global (also wholist) learners are provided breadth-first structure of learning material. Otherwise, analytical (also analytic, sequential, serialist) learners are recommended depth-first structure of learning materials. For the breadth-first structure, after a learner has already known all the topics at the same level, other descendant topics at lower level are recommended to her/him. For the depth-first structure, after a learner has already known a given topic  $T_1$  and all its children (topic) at lower level, the sibling topic of  $T_1$  (namely  $T_2$ , at same level with  $T_1$ ) will be recommended to her/him.
- The *FD/FI* dimension in Wikin model is correlated with *undirected/reproduction-oriented* dimension in Vermunt model. FD learners are provided breadth-first structure of materials, guided navigation, illustration of ideas with visual materials, advance organizer and system control. FI learners are provided depth-first structure of materials or navigational freedom, user control and individual environment.

The adaptive strategy (for learning style) is the sequence of adaptive rules which define how adaptation to learning styles is performed. Learning style strategies is classified into three following forms (Stash, Cristea, & De Bra, 2007, pp. 321-322):

- *Selection of information:* Information (learning materials) is presented in various types such as text, audio, video, graph, and picture. Depending on user's learning styles, an appropriate type will be chosen to provide to user. For example, verbalizers are recommended text and imagers are suggested pictures and graphs. This form supports adaptation techniques such as adaptive presentation, altering fragments, and stretch text.

- *Ordering information or providing different navigation paths:* The order in which learning materials suggested to users is tuned with learning styles. For active learners, learning materials are presented in the order: activity→example→theory→exercise. For reflective learner, this order is changed such as example→theory→exercise→activity. This form is corresponding to link adaptation techniques: direct guidance, link sorting, link hiding, link annotation.
- *Providing learners with navigation support tools:* Different learning tools are supported to learners according to their learning styles. For example, in Witkin model, FD learners are provided tools such as concept map, graphic path indicator. Otherwise FI learners are provided with a control option showing a menu from which they can choose in any order because they have high self-control.

There are two type of strategy (Stash, Cristea, & De Bra, 2005, p. 3):

- *Instructional strategy* is itself, which contains adaptive rules and is in three above forms such as selection of information, ordering information, and providing learners with navigation support tools.
- *Instructional meta-strategy* is strategy which is used to observe user actions and infer their learning styles. Thus, meta-strategy is applied in order to define strategy.

*My approach is an instructional meta-strategy, which applies Markov model to infer users' learning styles.* Before discussing about main techniques in sub-section III.2.5, it is necessary to glance over hidden Markov model in sub-section III.2.4.

#### III.2.4. Hidden Markov model

There are many real-world phenomena (so-called states) that we would like to model in order to explain our observations. Often, given sequence of observations symbols, there is demand of discovering real states. For example, there are some states of weather: *sunny*, *cloudy*, *rainy* (Fosler-Lussier, 1998, p. 1). Based on observations such as wind speed, atmospheric pressure, humidity, and temperature, it is possible to forecast the weather by using hidden Markov model (HMM). Before discussing about HMM, we should glance over the definition of Markov model (MM). First, MM is the statistical model which is used to model the stochastic process. MM is defined as below (Schmolze, 2001):

- Given a finite set of state  $S=\{s_1, s_2, \dots, s_n\}$  whose cardinality is  $n$ . Let  $\Pi$  be the *initial state distribution* where  $\pi_i \in \Pi$  represents the probability that the stochastic process begins in state  $s_i$ . In other words  $\pi_i$  is the initial probability of state  $s_i$ , where  $\sum_{s_i \in S} \pi_i = 1$ .
- The stochastic process which is modeled gets only one state from  $S$  at all times. The process is denoted as a finite vector  $P=(x_1, x_2, \dots, x_u)$  whose element  $x_i$  is a state ranging in space  $S$ . Note that  $x_i \in S$  is one of states in the finite set  $S$ ,  $x_i$  is identical to  $s_i$ . Moreover, the process must meet

fully the *Markov property*, namely, given previous state  $x_{k-1}$  of process  $P$ , the conditional probability of current state  $x_k$  is only dependent on the previous state  $x_{k-1}$ , not relevant to any further past state ( $x_{k-2}, x_{k-3}, \dots, 0$ ). In other words,  $P(x_k / x_{k-1}, x_{k-2}, x_{k-3}, \dots, x_0) = P(x_k / x_{k-1})$  with note that  $P(\cdot)$  also denotes probability in this research. Such process is called first-order Markov process. Note that Markov property was mentioned in previous section [III.1.4.2](#).

- At each lock time, the process transitions to the next state based upon the *transition probability distribution*  $a_{ij}$  which depends only on the previous state. So  $a_{ij}$  is the probability that, the process change the current state  $s_i$  to next state  $s_j$ . The probability of transitioning from any given state to some next state is 1, we have  $\forall s_i \in S, \sum_{s_j \in S} a_{ij} = 1$ . All transition probabilities  $a_{ij}$  (s) constitute the *transition probability matrix*  $A$ .

Briefly, MM is the triple  $\langle S, A, \Pi \rangle$ . In typical MM, states are observed directly by users and transition probability matrix is the unique parameters. Otherwise, hidden Markov model (HMM) is similar to MM except that the underlying states become hidden from observer, they are hidden parameters. HMM adds more output parameters which are called observations. Each state (hidden parameter) has the conditional probability distribution upon such observations. HMM is responsible for discovering hidden parameters (states) from output parameters (observations), given the stochastic process. The HMM has further properties as below (Schmolze, 2001):

- There is the second stochastic process which produces *observations* correlating hidden states. Suppose there is a finite set of possible observations  $\Theta = \{\theta_1, \theta_2, \dots, \theta_m\}$  whose cardinality is  $m$ .
- There is a probability distribution of producing a given observation in each state. Let  $b_i(k)$  be the probability of observation  $\theta_k$  when the second stochastic process is in state  $s_i$ . The sum of probabilities of all observations which observed in a certain state is 1, we have  $\forall s_i \in S, \sum_{\theta_k \in \Theta} b_i(k) = 1$ . All probabilities of observations  $b_i(k)$  constitute the *observation probability matrix*  $B$ .

Thus, HMM is the 5-tuple  $\Delta = \langle S, \Theta, A, B, \Pi \rangle$ . Back to weather example, suppose you need to predict how whether is tomorrow: *sun* or *cloud* or *rain* since you know only observations about the humidity: *dry*, *dryish*, *damp*, *soggy*. The HMM is totally determined based on its components  $S, \Theta, A, B$ , and  $\Pi$  according to weather example. We have  $S = \{sun, cloud, rain\}$ ,  $\Theta = \{dry, dryish, damp, soggy\}$ . Transition probability matrix  $A$  is shown in table [III.2.1](#).

		weather today		
		sunny	cloudy	rainy
weather yesterday	sunny	0.5	0.25	0.25
	cloudy	0.4	0.2	0.4
	rainy	0.1	0.7	0.2

**Table III.2.1.** Transition probability matrix  $A$ 

Initial state distribution specified as uniform distribution is shown in table III.2.2.

	<i>sunny</i>	<i>cloudy</i>	<i>rainy</i>
	0.5	0.5	0.5

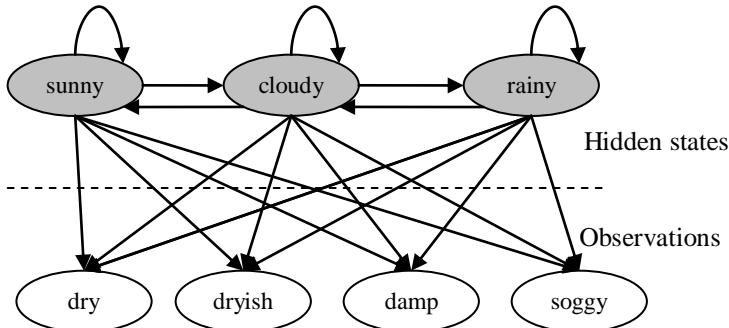
**Table III.2.2.** Uniform initial state distribution  $\Pi$ 

Observation probability matrix  $B$  is shown in table III.2.3.

		Humidity			
		<i>dry</i>	<i>dryish</i>	<i>damp</i>	<i>soggy</i>
weather	<i>sunny</i>	0.60	0.20	0.15	0.05
	<i>cloudy</i>	0.25	0.25	0.25	0.25
	<i>rainy</i>	0.05	0.10	0.35	0.50

**Table III.2.3.** Observation probability matrix  $B$ 

The whole HMM is depicted in figure III.2.4.


**Figure III.2.4.** HMM of weather forecast (hidden states are shaded)

Please see (Schmolze, 2001) and (Fosler-Lussier, 1998) for surveying hidden Markov model. You can refer to an excellent document “A tutorial on hidden Markov models and selected applications in speech recognition” written by author (Rabiner, 1989) for advanced details about HMM.

### Uncovering problem and Viterbi algorithm

Given HMM  $\Delta$  and a sequence of observations  $O = \{o_1 \rightarrow o_2 \rightarrow \dots \rightarrow o_k\}$  where  $o_i \in \Theta$ , how to find the sequence of states  $U = \{u_1 \rightarrow u_2 \rightarrow \dots \rightarrow u_k\}$  where  $u_i \in S$  so that  $U$  is most likely to have produced the observation sequence  $O$ . This is the uncovering problem: which sequence of state transitions is most likely to have led to given sequence of observations. It means that given the selective criterion  $P(U|O, \Delta)$  which is the conditional probability of sequence  $U$  with respect to sequence  $O$  and model  $\Delta$ , it is required to find out  $U$  by maximizing criterion  $P(U|O, \Delta)$  and we have  $U = \underset{U}{\operatorname{argmax}}(P(U|O, \Delta))$ . We can apply brute-force strategy: “go through all possible such  $U$  and pick the one leading to

maximizing criterion  $P(U/O, \Delta)$ " but this strategy is impossible if the number of states is huge. In this situation, Viterbi algorithm (Schmolze, 2001) is the effective solution. Instead of describing details of Viterbi algorithm, I only use it to predict learner's styles given observations about her/him.

The next sub-section [III.2.5](#) describes the new approach which uses HMM to discover and represent users' learning styles (Nguyen, A New Approach for Modeling and Discovering Learning Styles by Using Hidden Markov Model, 2013).

### **III.2.5. Applying hidden Markov model into building up learning style sub-model**

For modeling learning style (LS) using HMM we should determine states, observations and the relationship between states and observations in context of learning style. In other words, we must define five components  $S, \Theta, A, B, \Pi$ . Each learning style is now considered as a state. The essence of state transition in HMM is the change of user's learning style, thus, it is necessary to recognize the learning styles which are most suitable to user. After monitoring users' learning process, we collect observations about them and then discover their styles by using inference mechanism in HMM, namely Viterbi algorithm (Schmolze, 2001). Suppose we choose Honey-Mumford model and Felder-Silverman model as principal models which are presented by HMM. We have three dimensions: *Verbal/Visual*, *Activist/Reflector*, *Theorist/Pragmatist* which are modeled as three HMM(s):  $\Delta_1, \Delta_2, \Delta_3$  respectively. For example, in  $\Delta_1$ , there are two states: *Verbal* and *Visual*; so  $S_1=\{\text{verbal}, \text{visual}\}$ . We have:

- $\Delta_1 = \langle S_1, \Theta_1, A_1, B_1, \Pi_1 \rangle$ .
- $\Delta_2 = \langle S_2, \Theta_2, A_2, B_2, \Pi_2 \rangle$ .
- $\Delta_3 = \langle S_3, \Theta_3, A_3, B_3, \Pi_3 \rangle$ .

We are responsible for defining states ( $S_i$ ), initial state distributions ( $\Pi_i$ ), transition probability matrices ( $A_i$ ), observations ( $\Theta_i$ ), observation probability matrices ( $B_i$ ) through five steps:

1. **Defining states:** each state is corresponding to a leaning style.  
 $S_1=\{\text{verbal}, \text{visual}\}$ ,  
 $S_2=\{\text{activist}, \text{reflector}\}$ ,  
 $S_3=\{\text{theorist}, \text{pragmatist}\}$ .
2. **Defining initial state distributions:** Uniform probability distribution is used for each  $\Pi_i$ .  
 $\Pi_1 = \{0.5, 0.5\}$ ; it means that  $P(\text{verbal}) = P(\text{visual}) = 0.5$   
 $\Pi_2 = \{0.5, 0.5\}$ ;  $P(\text{activist}) = P(\text{reflector}) = 0.5$   
 $\Pi_3 = \{0.5, 0.5\}$ ;  $P(\text{theorist}) = P(\text{pragmatist}) = 0.5$
3. **Defining transition probability matrices:** Suppose that learners tend to keep their styles; so the conditional probability of a current state on

previous state is high if both current state and previous state have the same value and otherwise. For example,  $P(s_i=\text{verbal} | s_{i-1}=\text{verbal}) = 0.7$  is obviously higher than  $P(s_i=\text{verbal} | s_{i-1}=\text{visual}) = 0.3$ . Transition probability matrices are shown in table III.2.4.

$A_1$	<i>verbal</i>	<i>visual</i>
<i>verbal</i>	0.7	0.3
<i>visual</i>	0.3	0.7

$A_2$	<i>activist</i>	<i>reflector</i>
<i>activist</i>	0.7	0.3
<i>reflector</i>	0.3	0.7

$A_3$	<i>theorist</i>	<i>pragmatist</i>
<i>theorist</i>	0.7	0.3
<i>pragmatist</i>	0.3	0.7

**Table III.2.4.** Transition probability matrices:  $A_1, A_2, A_3$

4. Defining **observations**. There is a relationship between learning object learned by users and their learning styles. Three attributes are assigned to each learning object such as lecture, lesson, example, and test.

- *Format* attribute indicating the format of learning object has three values: *text, picture, video*.
- *Type* attribute telling the type of learning object has four values: *theory, example, exercise, puzzle*.
- *Interactive* attribute indicates the “interactive” level of learning object. The more interactive learning object is, the more learners interact together in their learning path. This attribute has three values corresponding to three levels: *low, medium, high*.

Whenever a student selects a learning object (LO), it raises observations depending on the attributes of learning object. We must account for the values of the attributes selected. For example, if a student selects a LO which has *format* attribute being *text*, *type* attribute being *theory*, *interactive* attribute being *low*, there are considerable observations: *text, theory, low* (interaction). So, it is possible to infer that she/he is a theorist.

The dimension *Verbal/Visual* is involved in *format* attribute. The dimensions *Activist/Reflector* and *Theorist/Pragmatist* relate to both *type* attribute and *interactive* attribute. So we have:

- $\Theta_1 = \{ \text{text, picture, video} \}$
- $\Theta_2 = \{ \text{theory, example, exercise, puzzle, low (interaction), medium (interaction), high (interaction)} \}$
- $\Theta_3 = \{ \text{theory, example, exercise, puzzle, low (interaction), medium (interaction), high (interaction)} \}$

5. Defining **observation probability matrices**. Different observations (attributes of LO) effect on states (learning styles) in different degrees. Because the “weights” of observation vary according to states, there is a question: “How to specify weights?”. If we can specify these “weights”, it is easy to determine observation probability matrices.

In the Honey-Mumford model and Felder-Silverman model, verbal students prefer to text material and visual students prefer to pictorial materials. The weights of observations: *text, picture, video* on state *Verbal* are in descending order. Otherwise, the weights of observations: *text, picture, video* on state *Visual* are in ascending order. Such weights themselves are observation probabilities. We can define these weights as below:

- $P(\text{text} \mid \text{verbal}) = 0.6, P(\text{picture} \mid \text{verbal}) = 0.3, P(\text{video} \mid \text{verbal}) = 0.1.$
- $P(\text{text} \mid \text{visual}) = 0.2, P(\text{picture} \mid \text{visual}) = 0.4, P(\text{video} \mid \text{visual}) = 0.4.$

There are some differences in specifying observation probabilities of dimensions *Activist/Reflector* and *Theorist/Pragmatist*. As discussed, active learners are provided activity-oriented approach: showing content of activity (such as puzzle, game...) and links to example, theory and exercise. Reflective learners are provided example-oriented approach: showing content of example and links to theory, exercise and activity (such as puzzle, game...). The weights of observations: *puzzle, example, theory, exercise* on state *Activist* are in descending order. The weights of observations: *example, theory, exercise, puzzle* on state *Reflector* are in descending order. However, activists tend to learn high interaction materials and reflectors prefer to low interaction materials. So the weight of observations: *low* (interaction), *medium* (interaction), *high* (interaction) on state *Activist* get values: 0, 0, 1, respectively. Otherwise, the weight of observations: *low* (interaction), *medium* (interaction), *high* (interaction) on state *Reflector* get values: 1, 0, 0, respectively. We have:

- $P(\text{puzzle} \mid \text{activist}) = 0.4, P(\text{example} \mid \text{activist}) = 0.3, P(\text{theory} \mid \text{activist}) = 0.2, P(\text{exercise} \mid \text{activist}) = 0.1.$   
 $P(\text{low} \mid \text{activist}) = 0, P(\text{medium} \mid \text{activist}) = 0, P(\text{high} \mid \text{activist}) = 1.$
- $P(\text{example} \mid \text{reflector}) = 0.4, P(\text{theory} \mid \text{reflector}) = 0.3, P(\text{exercise} \mid \text{reflector}) = 0.2, P(\text{puzzle} \mid \text{reflector}) = 0.1.$   
 $P(\text{low} \mid \text{reflector}) = 1, P(\text{medium} \mid \text{reflector}) = 0, P(\text{high} \mid \text{reflector}) = 0.$

Because the sum of conditional probabilities of observations on each state equals 1, we should normalize above probabilities.

- $P(\text{puzzle} \mid \text{activist}) = 0.4*4/7 = 0.22, P(\text{example} \mid \text{activist}) = 0.3*4/7 = 0.17, P(\text{theory} \mid \text{activist}) = 0.2*4/7 = 0.11, P(\text{exercise} \mid \text{activist}) = 0.1*4/7 = 0.05.$   
 $P(\text{low} \mid \text{activist}) = 0*3/7 = 0, P(\text{medium} \mid \text{activist}) = 0*3/7 = 0, P(\text{high} \mid \text{activist}) = 1*3/7 = 0.42.$

- $P(example \mid reflector) = 0.4*4/7 = 0.22$ ,  $P(theory \mid reflector) = 0.3*4/7 = 0.17$ ,  $P(exercise \mid reflector) = 0.2*4/7 = 0.11$ ,  $P(puzzle \mid reflector) = 0.1*4/7 = 0.05$ .

$$P(low \mid reflector) = 1*3/7 = 0.42, P(medium \mid reflector) = 0*3/7 = 0, P(high \mid reflector) = 0*3/7 = 0.$$

According to Honey and Mumford model, *theorists* are provided theory-oriented approach: showing content of theory and links to example, exercise and puzzle; *pragmatists* are provided exercise-oriented approach: showing content of exercise and links to example, theory and puzzle. Thus, the conditional probabilities of observations: *example*, *theory*, *exercise*, *puzzle*, *low* (interaction), *medium* (interaction), *high* (interaction) on states: *theorists*, *pragmatists* are specified by the same technique discussed above. Observation probability matrices are shown in table III.2.5.

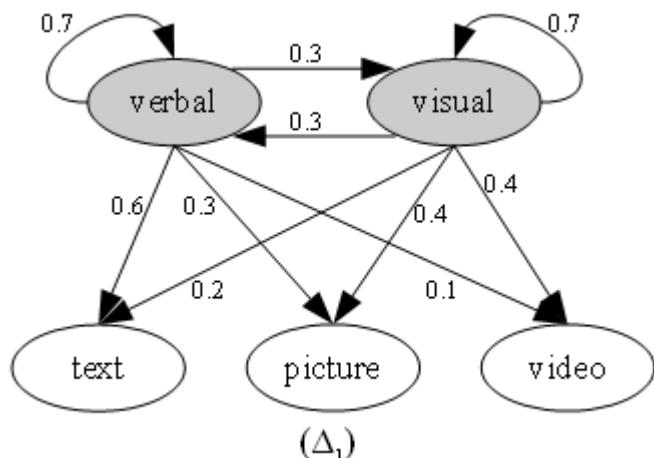
$B_1$	<i>text</i>	<i>picture</i>	<i>video</i>
<i>verbal</i>	0.6	0.3	0.1
<i>visual</i>	0.2	0.4	0.4

$B_2$	<i>theory</i>	<i>example</i>	<i>exercise</i>	<i>puzzle</i>	<i>low</i>	<i>medium</i>	<i>high</i>
<i>activist</i>	0.11	0.17	0.05	0.22	0	0	0.42
<i>reflector</i>	0.17	0.22	0.11	0.05	0.42	0	0

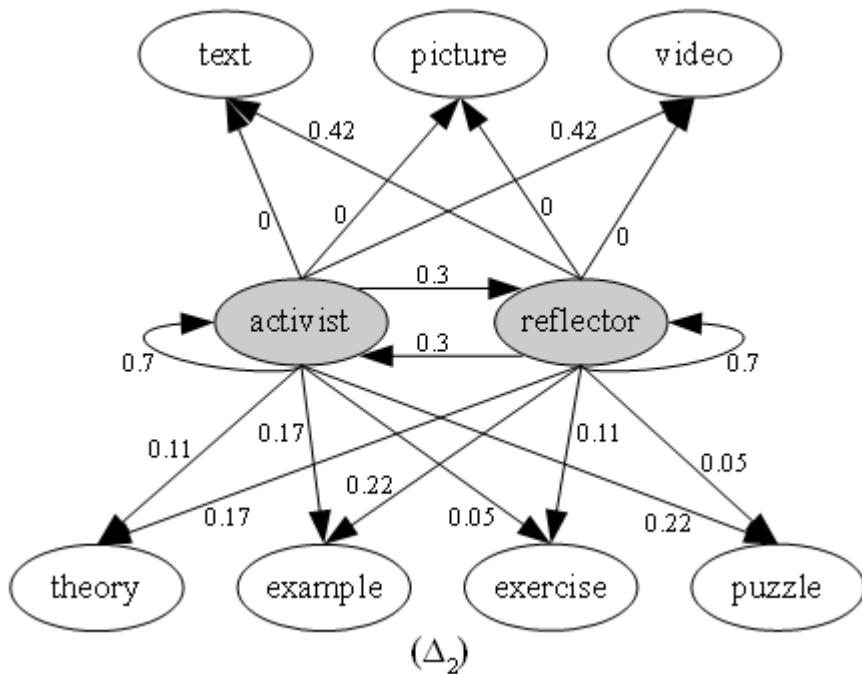
$B_3$	<i>theory</i>	<i>example</i>	<i>exercise</i>	<i>puzzle</i>	<i>low</i>	<i>medium</i>	<i>high</i>
<i>pragmatist</i>	0.11	0.17	0.22	0.05	0.04	0.08	0.30
<i>theorist</i>	0.22	0.17	0.11	0.05	0.3	0.08	0.04

**Table III.2.5.** Observation probability matrices:  $B_1$ ,  $B_2$ ,  $B_3$

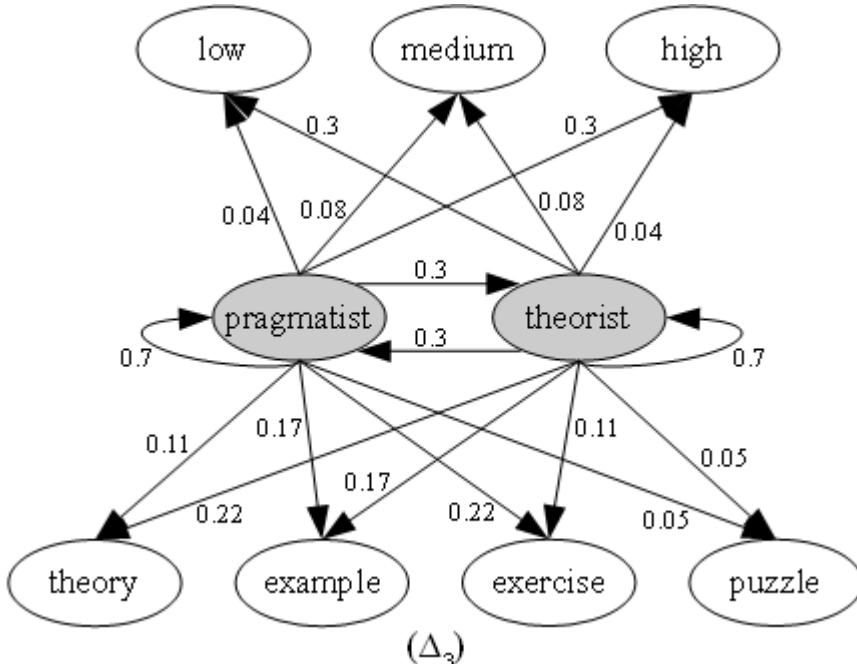
Now three HMM (s):  $\Delta_1$ ,  $\Delta_2$ ,  $\Delta_3$  corresponding to three dimensions of learning styles: *Verbal/Visual*, *Activist/Reflector*, *Pragmatist/Theorist* are represented respectively in figures III.2.5, III.2.6, and III.2.7.



**Figure III.2.5.** HMM (s) of learning styles  $\Delta_1$  (hidden states are shaded)



**Figure III.2.6.** HMM (s) of learning styles  $\Delta_2$  (hidden states are shaded)



**Figure III.2.7.** HMM (s) of learning styles  $\Delta_3$  (hidden states are shaded)

### An example for inferring student's learning styles

Suppose the learning objects that a student selects in session 1, 2 and 3 are  $LO_1$ ,  $LO_2$  and  $LO_3$  respectively, as seen in table III.2.6.

	Format	Type	Interactive
$LO_1$	picture	theory	not assigned

$LO_2$	text	example	not assigned
$LO_3$	text	not assigned	low

**Table III.2.6.** Learning objects selected

It is easy to recognize the sequence of user observations from the attributes *format*, *type*, *interactive*, as shown in table [III.2.7](#).

HMM – Dimension	Sequence of observations
$\Delta_1$ : Dimension Verbal/Visual	picture → text → text
$\Delta_2$ : Dimension Activist/Reflector	theory → example → low
$\Delta_3$ : Dimension Pragmatist/Theorist	theory → example → low

**Table III.2.7.** Sequence of student observations

Using Viterbi algorithm for each HMM, it is possible to find corresponding sequence of state transitions that is most suitable to have produced such sequence of observations, as shown in table [III.2.8](#).

HMM - Dimension	Sequence of observations	Sequence of state transitions	Student style
$\Delta_1$	picture → text → text	visual → verbal	verbal
$\Delta_2$	theory → example → low	reflector → reflector → reflector	reflector
$\Delta_3$	theory → example → low	theorist → theorist → theorist	theorist

**Table III.2.8.** Sequence of state transitions

It is easy to deduce that this student is a verbal, reflective and theoretical person. Since then, adaptive learning systems will provide appropriate instructional strategies to her/him.

Table [III.2.8](#) shows the ultimate result of proposed approach mentioned in this sub-section [III.2.5](#). The next sub-section [III.2.6](#) is the evaluation of the proposed approach.

### III.2.6. Evaluation

HMM and Viterbi algorithm (Schmolze, 2001) provide the way to model and predict users' learning styles. I propose five steps to realize and apply HMM into two learning style models: Honey-Mumford and Felder-Silverman, in which styles are considered states and user's selected learning objects are tracked as observations. The sequence of observations becomes the input of Viterbi algorithm for inferring the real style of learner. It is possible to extend my approach into other learning style models such as Witkin, Riding, and Kolb. Moreover, there is no need to alter main techniques except that we should specify new states correlating with new learning styles and add more attributes to learning objects.

This section [III.2](#) ends up the comprehensive description of both knowledge sub-model and learning sub-model, two of three components constituting [Triangular Learner Model](#) (TLM). At this time, we have known thoroughly both [specific-domain](#) and [independent-domain](#) information such as user

knowledge and personal traits available in knowledge sub-model (see section [III.1](#)) and learning style sub-model, respectively. Such information is fine information analyzed or extracted from more essential information that is manipulated by the third sub-model so-called learning history sub-model which is described in next section [III.3](#). Are you ready to survey the most important sub-model which will be discussed in next section [III.3](#)?

### III.3. Learning history sub-model

Learning history sub-model is the basic sub-model among three sub-models constituting the [Triangular Learner Model](#) (TLM) when two remaining sub-models such as knowledge and learning style were expressed in previous sections [III.1](#) and [III.2](#), respectively. Learning history is defined as “a transcript of all learners’ actions such as learning materials access, duration of computer use, doing exercise, taking an examination, doing test, communicating with teachers or classmates”. This sub-model has four main functions:

1. *Providing necessary information for two remaining sub-models:* learning style sub-model and knowledge sub-model described in previous sections [III.1](#) and [III.2](#) so that they perform inference tasks. For example, knowledge sub-model needs learning evidences like learner’s results of test, frequency of accessing lectures so as to assess learner’s mastery of concrete knowledge item or concept. Such coarse information is organized and stored as structured data such as XML files (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) or relation database (Ramakrishnan & Gehrke, 2003, pp. 25-94) by learning history sub-model.
2. *Supporting learning concept recommendation.*
3. Mining learners’ educational data in order to *discover other learners’ characteristics* such as interests, background, and goals.
4. *Supporting collaborative learning* through constructing learner groups.

This model is managed by [mining engine](#) (ME) which almost always uses mining techniques with note that ME is the one among two essential engines inside the user modeling system Zebra; please see sub-section [II.2.2](#) for more details about Zebra. ME is very important for collecting learners’ data, monitoring their actions, structuring and updating TLM.

The first function (providing necessary information for remaining sub-models) is implied in previous sections [III.1](#) and [III.2](#) about knowledge sub-model and learning style sub-model. So I have just described in this section [III.3](#) the approaches that learning history sub-model applies to perform recommendation tasks, discover another learners’ characteristic, namely user interests and construct learner groups. This sub-model is an open model that allows developer/programmer to plug other functions into it. For example, a programmer can develop a new component that discovers learner’s goals by using mining techniques and attach such component to learning history sub-model. This sub-model is very necessary for extending Triangular Learner Model (see figure [II.7](#)) and so, it is easy to recognize that functions 2, 3, 4 are extended functions of Zebra.

This section includes three following sub-sections such as learning concept recommendation (sub-section [III.3.1](#)), discovering user interests (sub-section [III.3.2](#)) and constructing user groups (sub-section [III.3.3](#)) that correspond to functions 2, 3, 4 of this sub-model. It is conventional these functions 2, 3, and 4 are called the first, second, and third *extended functions*, respectively. Now the research of learning history sub-model is started with sub-section [III.3.1](#) which

mentions how to recommend learning concepts based on mining learning history.

### **III.3.1. Learning concept recommendation based on mining learning history**

Supporting learning concept recommendation is a function among three extended functions of learning history sub-model and **mining engine** (ME) and so, this sub-section **III.3.1** is reserved for describing this function in detailed. Recall that such three extended functions are to perform recommendation tasks, to discover user interests and to construct learner groups. Because the concept recommendation is based on sequential pattern mining, it is necessary to introduce sequential pattern mining, firstly. Sequential pattern mining is new trend in data mining domain with many useful applications, especially commercial application but it also results surprised effect in adaptive learning. Suppose there is an adaptive e-learning website, a student accesses learning materials / does exercises relating domain concepts in sessions. Her/his learning sequences which are lists of concepts accessed after total study sessions construct the learning sequence database  $S$ . Note that  $S$  is stored in learning history sub-model or extracted from learning history sub-model.  $S$  is mined to find the sequences which are expected to be learned frequently or preferred by student. Such sequences called sequential patterns are used to recommend appropriate concepts / learning objects to students in his next visits. It results in enhancing the quality of adaptive learning system. This process is sequential pattern mining. In this section, I also propose an approach to break sequential pattern  $s=\langle c_1, c_2, \dots, c_m \rangle$  into association rules including left-hand and right-hand in form  $c_i \rightarrow c_j$ . Left-hand is considered as source concept, right-hand is treated as recommended concept available to students (Nguyen & Do, Learning Concept Recommendation based on Sequential Pattern Mining, 2009).

Recommendation methods are categorized into three different trends:

- Rule-based filtering system is based on manually or automatically generated decision rules that are used to recommend items to users (Ricci, Rokach, Shapira, & Kantor, 2011, pp. 48-60).
- Content-based filtering system recommends items that are considered appropriate to user information in his profile (Ricci, Rokach, Shapira, & Kantor, 2011, pp. 73-100).
- Collaborative filtering system is considered as social filtering when it matches the rating of a current user for items with those of similar users in order to produce recommendations for new items (Ricci, Rokach, Shapira, & Kantor, 2011, pp. 107-184).

Sequential pattern mining belongs to collaborative filtering family. User does not rate explicitly items but his series of chosen items are recorded as sequences to construct the sequence database which is mined to find frequently repeated patterns he can choose in future. In learning context, items can be domain concepts / learning objects which students access or learn. First, we

### III.3. Learning history sub-model

---

should glance over basic concepts and what is sequential pattern mining along with its application, especially in adaptive learning.

Suppose  $I = \{i_1, i_2, i_3, \dots, i_n\}$  is a set of all *items*. An *itemset* is a subset of  $I$ , denoted  $x_i = (i_u, \dots, i_v)$ , if  $x_i$  has only one item  $i_k$ , it can be denoted  $x_i = i_k$ , omitting the brackets. An *sequence* is an ordered list of itemsets, denoted  $s = \langle x_1, x_2, \dots, x_m \rangle$ , where  $x_i$  is an itemset,  $x_i$  is also called an *element* in sequence  $s$ . The term “ordered list” means that sequence  $s_1 = \langle x_1, x_2, x_3 \rangle$  is distinguished from sequence  $s_2 = \langle x_1, x_3, x_2 \rangle$  although both of them have the same elements. An item has only been once in an element of sequence but can occur repeatedly many times in different elements in sequences (Han, Pei, & Yan, 2005, p. 184). The number of items occurring in sequence is the length of sequence. Sequence with length  $l$  is called *l-sequence*. Please read the document (Han, Pei, & Yan, 2005) for more basic concepts about sequence mining and please distinguish sequences mentioned in this sub-section III.3.1 from sequences of student observations and sequences of state transitions aforementioned in tables III.2.7 and III.2.8 in previous sub-section III.2.5 – “Applying hidden Markov model into building up learning style sub-model”.

Sequence  $s_1 = \langle x_1, x_2, \dots, x_n \rangle$  is called sub-sequence of  $s_2 = \langle y_1, y_2, \dots, y_m \rangle$  denoted  $s_1 \subseteq s_2$  or  $s_2 \supseteq s_1$  if there is the ordered series of indexes  $k_1, k_2, \dots, k_n$  so that  $1 \leq k_1 \leq k_2 \leq \dots \leq k_n \leq m$  and

$$x_1 \subseteq y_{k_1}, x_2 \subseteq y_{k_2}, \dots, x_n \subseteq y_{k_n}$$

Assertion “ $s_1$  is sub-sequence of  $s_2$ ” is equivalent to “ $s_2$  is super-sequence of  $s_1$ ” For example, suppose that:

- A set of all items  $I = \{i_1, i_2, i_3, i_4\}$
- Six itemsets  $x_1 = i_1, x_2 = (i_1, i_3), x_3 = (i_1, i_2, i_3), x_4 = (i_1, i_2, i_3, i_4), x_5 = i_3, x_6 = (i_1, i_2)$
- Three sequences  $s_1, s_2$  denoted as  $s_1 = \langle x_1, x_2 \rangle, s_2 = \langle x_1, x_2, x_3 \rangle, s_3 = \langle x_5, x_6 \rangle$  but  $s_1, s_2$  are often shown in detailed forms:  $s_1 = \langle i_1(i_1, i_3) \rangle, s_2 = \langle i_1(i_1, i_3)(i_1, i_2, i_3) \rangle, s_3 = \langle i_3(i_1, i_2) \rangle$

We have:

- The sequences  $s_1, s_2, s_3$  are 5-length, 6-length, 3-length sequences, respectively. Note, item  $i_1$  occurs 2 times in sequence  $s_1$ , so it contributes 2 to length of  $s_1$
- The sequence  $s_3$  is sub-sequence of  $s_2$  due to  $x_5 \subseteq x_2$  and  $x_6 \subseteq x_3$ . The sequence  $s_1$  is not sub-sequence of  $s_2$  because  $x_1 \subseteq x_2$  but  $x_4$  is not subset of any subset in  $s_2$ .

Suppose the sequential database  $S$  has a set of records  $\langle sid, s \rangle$  where  $sid$  is an identifier of sequence  $s$ . A record  $\langle sid, s \rangle$  is said to contain sequence  $\alpha$  if  $\alpha \subseteq s$ . The support of sequence  $\alpha$  is the fraction of total records containing  $\alpha$ , denoted  $support(\alpha) = |\{\langle sid, s \rangle / \alpha \subseteq s\}| / |S|$ . Similar to association rules, given the number  $min\_sup$  as support threshold, if the support of  $\alpha$  is greater than or

equals  $\text{min\_sup}$ , namely  $\text{support}(\alpha) \geq \text{min\_sup}$  then  $\alpha$  is called large or *frequent sequence*. A sequence is maximal if it has no super-sequence. The *maximal frequent sequence* is called **sequential pattern**, it means that all super-sequences of sequential pattern are infrequent. In case that property “maximal” is not paid attention, frequent sequence is considered as sequential pattern. If  $s$  is infrequent sequence but all its sub-sequences are frequent,  $s$  is called *minimal infrequent sequence*. Please read the document (Han, Pei, & Yan, 2005) for more basic concepts about sequence mining.

Totally, given a sequence database  $S$  and the threshold  $\text{min\_sup}$ , sequential pattern mining is to find all complete set of sequential patterns in  $S$ . Note that  $S$  is structured by learning history sub-model.

### Issue of learning concepts / objects recommendation

Suppose there are compulsory concepts (subjects) in Java course: **data type**, **package**, **class & OOP**, **selection structure**, **virtual machine**, **loop structure**, **control structure**, and **interface** which in turn denoted as  $d, p, o, s, v, l, c, f$ . These concepts are considered as items in sequence database. Note that Java is a popular object-oriented programming language <https://www.oracle.com/java>. At our e-learning website, students access learning material relating such domain concepts in sessions, each session contains only one itemset and is ordered by time. Each student’s learning sequence is constituted of itemsets accessed in all her/his sessions. Table III.3.1 shows sequence database  $S$  created from these learning sequences which are constructed from learning sessions, in turn.

Student	Session	Concept accessed
1	Aug 5 10:20:01	$o$
1	Aug 5 10:26:12	$f$
2	Aug 6 08:20:01	$d, p$
2	Aug 6 14:15:01	$o$
2	Aug 6 15:00:00	$s, l, c$
3	Aug 7 12:30:00	$o, v, c$
4	Aug 8 07:14:20	$o$
4	Aug 8 07:40:25	$s, c$
4	Aug 8 10:17:20	$f$
5	Aug 8 10:26:15	$f$

ID	Learning sequences	Length
1	$\langle of \rangle$	2
2	$\langle (dp)o(slc) \rangle$	6
3	$\langle (ovc) \rangle$	3
4	$\langle o(sc)f \rangle$	4
5	$\langle f \rangle$	1

**Table III.3.1.** Learning sessions → learning sequences

Students accessed learning material in their past sessions, how system recommends appropriate domain concepts to student for next visits. It issues the application of mining sequential patterns. In learning context, sequential pattern is a sequence of concepts / learning materials that students prefer to study or access regularly. My solution includes two steps as below:

1. Applying techniques of mining user learning data to find learning sequential patterns.
2. Breaking such patterns into concepts / learning materials which are recommended to users.

We browse some methods to mine sequential patterns in sub-section III.3.1.1. The approach to break patterns into concepts is proposed in sub-section III.3.1.2. Sub-section III.3.1.3 is the evaluation. Note, sequences mentioned in the research are considered as learning sequences by default.

### III.3.1.1. Approaches of sequential pattern mining

There are two main approaches of sequential pattern mining (Han, Pei, & Yan, 2005, p. 183):

1. *Candidate generation-and-test approach* based on algorithm Apriori (Han & Kamber, 2006, pp. 248-253) is also classified into two categories: horizontal and vertical data format methods. Candidate generation-and-test approach is mentioned in next part III.3.1.1.1.
2. *Pattern-growth approach* based on pattern-growth mining of frequent patterns in transaction database. Pattern-growth approach is mentioned in part III.3.1.1.2.

#### III.3.1.1.1. Candidate generation-and-test approach

This approach is essentially an extension of associate rule discovering algorithm Apriori satisfying the statement “*every non-empty sub-sequence of a frequent sequence is a frequent sequence*” (and vice versa) considered as downward closure property (Han, Pei, & Yan, 2005, p. 5). This approach has two trends:

- *Horizontal data format* based sequential pattern mining with typical algorithms such as AprioriAll (Agrawal & Srikant, 1995), and GSP (Srikant & Agrawal, 1996).
- *Vertical data format* based sequential pattern mining with typical algorithm: SPADE (Zaki, 2001).

**AprioriAll** algorithm (Agrawal & Srikant, 1995, pp. 3-5) is the most similar to Apriori algorithm but applied for sequential pattern mining and has 3 main phases: large itemset phase, transformation phase, sequence phase and maximal phase. Sequence phase is the most important.

1. Large itemset phase (Agrawal & Srikant, 1995, p. 3): The **support of itemset**  $x_i$  is defined as the fraction of total sequences which containing  $x_i$ . Note that  $x_i$  is contained in a sequence if any itemset in sequence contains it. In case  $x_i$  occurs many times in the same sequence, its support is increased only once. Itemset  $x_i$  is called *frequent itemset* or *large itemset* or *litemset* in brief if its support satisfies the support threshold ( $\geq \text{min\_sup}$ ). If sequence  $s$  is recognized as large sequence (frequent sequence) then all its itemsets are litemsets.

Suppose all itemset in sequence database  $S$  are litemsets, the length of sequence is re-defined as the number of litemsets contained in it.

Obviously, both supports of 1-sequence  $\langle l_i \rangle$  and litemset  $l_i$  are the same.

This phase finds all itemsets in  $S$  with support threshold  $\text{min\_sup}$ . They are considered as atomic elements in sequences and can be mapped to integer in convenient (Agrawal & Srikant, 1995, p. 3). Let threshold  $\text{min\_sup}=25\%$ , there are just itemsets  $o, s, c, (sc), f$  from learning sequence database  $S$  shown in table III.3.1. Recall that letters  $o, s, c$ , and  $f$  denote Java course concepts: class & OOP, selection structure, control structure, and interface, respectively. All itemsets are mapped to integers for convenience and each of them looks like as single item appropriate to re-definition of sequence length. Table III.3.2 shows itemsets mapped to integers.

<i>itemsets</i>	<i>mapped to</i>
$o$	1
$s$	2
$c$	3
$(sc)$	4
$f$	5

**Table III.3.2.** Large itemsets are mapped to integers

2. Transformation phase (Agrawal & Srikant, 1995, p. 4): In this phase, each itemset  $x_i$  in sequence database  $S$  are replaced by itemset  $l_i$  where  $x_i \subseteq l_i$  ( $l_i$  contains  $x_i$ ). If a sequence has no itemset, it will be removed from sequence database  $S$  but it still contributes to the count of sequences. Table III.3.3 shows transformed (learning) sequences.

ID	Original sequence	Transformed sequence	Mapped
1	$\langle of \rangle$	$\langle of \rangle$	$\langle 15 \rangle$
2	$\langle (dp)o(slc) \rangle$	$\langle o(sc(sc)) \rangle$	$\langle 1(234) \rangle$
3	$\langle (ovc) \rangle$	$\langle (oc) \rangle$	$\langle (13) \rangle$
4	$\langle o(sc)f \rangle$	$\langle o(sc(sc))f \rangle$	$\langle 1(234)5 \rangle$
5	$\langle f \rangle$	$\langle f \rangle$	$\langle 5 \rangle$

**Table III.3.3.** Transformed sequences

3. Sequence phase (Agrawal & Srikant, 1995, pp. 4-5): The purpose of sequence phase is to find all large sequences. This is iterative process that scans database many times to find the large  $k$ -sequences where  $k$  is increased until no large sequence can be found. Let  $L_k$  is a set of large  $k$ -sequence. Each  $k^{th}$  iterative process includes three steps as follows:

- a. At the beginning of  $k^{th}$  scan, all sequences in  $L_{k-1}$  are joined to generate  $C_k$ , the set of new potential large  $k$ -sequences which is also called candidate sequences.
- b. After that, the supports of candidate sequences in  $C_k$  are computed and assessed whether they satisfy  $\text{min\_sup}$  so that it is

possible to determine the actually large sequences. The set of actually large  $k$ -sequences is  $L_k$ .

- c.  $L_k$  is re-used as the seed for generating  $L_{k+1}$ .

The technique for generating candidate sequence  $C_k$  (in step a) is similar to join operator in SQL-language as follows (Agrawal & Srikant, 1995, p. 5):

```

INSERT INTO  $C_k$ 
SELECT  $p.litemset_1, p.litemset_2, \dots, p.litemset_{k-1}, q.litemset_{k-1}$ 
FROM  $L_{k-1} p, L_{k-1} q$ 
WHERE  $p.litemset_1 = q.litemset_1$  AND  $p.litemset_2 = q.litemset_2$  AND
... AND  $p.litemset_{k-2} = q.litemset_{k-2}$ 

```

Because all sub-sequences of a large sub-sequence are also large and vice versa, it is necessary to prune off sequence  $c \in C_k$  whose sub-sequences do not occur in  $L_{k-1}$ . Table III.3.4 shows the task of candidate generation.

Large 3-sequences	Candidate 4-sequences	Candidate 4-sequence (pruned)
$\langle 123 \rangle$	$\langle 1234 \rangle$	$\langle 1234 \rangle$
$\langle 124 \rangle$	$\langle 1243 \rangle$	
$\langle 134 \rangle$	$\langle 1345 \rangle$	
$\langle 135 \rangle$	$\langle 1354 \rangle$	
$\langle 234 \rangle$		

**Table III.3.4.** Candidate generation

Finally, when all larger sequences  $L$  were discovered, we must find maximal sequences among  $L$  because sequential patterns are maximal large sequences. Suppose  $n$  is the length of longest sequences in  $L$ , such operator is done as follows (Agrawal & Srikant, 1995, p. 4):

```

for ( $i = n; i > 1; i=i-1$ )
    foreach  $i$ -sequence  $s_i$  do “remove sub-sequences of  $s_i$  from  $L$ ”

```

In this example there is only one larger sequence  $\langle 1234 \rangle$ , so it also is maximal. The larger sequence  $\langle 1234 \rangle$  is mapped inversely as  $\langle osc(sc) \rangle$ . Please see table III.3.2 for mapping litemsets to integers; recall that letters  $o$ ,  $s$ ,  $c$ , and  $f$  denote Java course concepts: class & OOP, selection structure, control structure, and interface, respectively.

**GSP (Generalized Sequential Pattern)** algorithm (Srikant & Agrawal, 1996) is a variant of AprioriAll. It also adopts multiple passes over database, uses previous frequent sequences as seeds to generate candidate sequences in each pass and test them to find the actually frequent sequences which re-used for

next pass (Han, Pei, & Yan, 2005, p. 6), like AprioriAll algorithm. However, there are some differences:

- There is no large itemset phase and transformation phase. The length of sequence is actually the number of its items (not itemsets) like original definition. So all frequent sequences in  $L_1$  have only one item in form  $\langle x \rangle$ .
- The way to generate candidate set produces total possible combination of sequences. For example,  $L_1 = \{\langle a \rangle, \langle b \rangle, \langle c \rangle\}$  having three 1-sequences generate fifteen 2-sequences,  $C_2 = \{\langle aa \rangle, \langle ab \rangle, \langle ac \rangle, \langle ba \rangle, \langle bb \rangle, \langle bc \rangle, \langle ca \rangle, \langle cb \rangle, \langle cc \rangle, \langle (ab) \rangle, \langle (ac) \rangle, \langle (ba) \rangle, \langle (bc) \rangle, \langle (ca) \rangle, \langle (cb) \rangle\}$ . If  $L_1$  have  $n$  1-sequence, it generates  $n^2 + n*(n-1)/2$  elements in candidate set  $C_2$ .

GSP finds more frequent sequences than AprioriAll because of generating full of candidate sets but it can raise the problem of the large number of candidate sequences. Suppose there are 300 large sequences of 1-length, algorithm will deliver  $300^2 + 300*299/2 = 134,850$  candidate 2-sequences.

AprioriAll and GPS generate the candidate sequences by scanning database in horizontal data format. That scanning row by row requires accessing database many times. It takes the consequence of downward performance. Whereas **SPADE** (Sequential PAttern Discovery using Equivalent classes) mines frequent sequences by considering sequence database as vertical format data set. The SPADE algorithm (Zaki, 2001) has below steps:

1. Cracking the sequence database into a vertical format data set in which each single itemset associates with the sequence identifier (*sid*) and its individual event identifier (*eid*) to form the new record (Han, Pei, & Yan, 2005, pp. 190-191). For example, a sequence record  $\{1, \langle itemset_1, itemset_2, itemset_3 \rangle\}$  is broken into three records:  $\{1, 1, \langle itemset_1 \rangle\}, \{1, 2, \langle itemset_2 \rangle\}, \{1, 3, \langle itemset_3 \rangle\}$ . Note in simple case, itemset is replaced by item if we consider that each itemset has only one item. Table III.3.5 shows how to crack sequence database (table III.3.1) into vertical format data sets.

<i>sid</i>	<i>eid</i>	itemsets (item)
1	1	<i>of</i>
2	1	<i>dp</i>
2	2	<i>o</i>
2	3	<i>slc</i>
3	1	<i>ovc</i>
4	1	<i>o</i>
4	2	<i>sc</i>
4	3	<i>f</i>
5	1	<i>f</i>

**Table III.3.5.** Cracking sequence database into vertical format data sets

2. The frequent 1-sequences are found by exploring frequent itemsets. Then, each candidate  $k$ -sequence is formed by joining two frequent  $(k-1)$ -sequences if they share the same  $sid$  and their event identifiers ( $eid$ ) follow the sequential order. The length of frequent sequences is growth until reaching maximal length or it is not able to find any more frequent itemsets. Table III.3.6 shows how to join two frequent itemsets.

<i>o</i>		<i>f</i>		<i>of</i>		
sid	eid	sid	eid	sid	eid( <i>o</i> )	eid( <i>f</i> )
1	1	1	1	1	1	1
2	2	4	3	4	1	3
3	1	5	1			
4	1					

**Table III.3.6.** (1). frequent itemsets ( $o$ ) and ( $f$ ) are associated with pairs ( $sid, eid$ ). (2). 2-sequence  $\langle of \rangle$  are found by joining itemsets ( $o$ ), ( $f$ ) in (1)

SPADE reduce frequency of global database access because candidate sequence generation is based on local vertical format data set in which itemsets and sub-sequence are represented by their pair ( $sid, eid$ ). However, nature of SPADE is similar to AprioriAll and GPS, the weakness of exploring breadth-first search and generating large candidate sequences still exists in SPADE (Han, Pei, & Yan, 2005, p. 9). This weakness is overcome by pattern-growth approach mentioned in next part [III.3.1.1.2](#).

### *III.3.1.1.2. Pattern-growth approach*

Huge candidate sets generated in AprioriAll, GSP, SPADE are caused by the large number of elements in seed set. Mining separately frequent sequences with disjointed database for the purpose of reduce the number of elements is idea of **FreeSpan** (**Frequent pattern-projected Sequential pattern**) algorithm (Han, Pei, Mortazavi-Asl, Chen, Dayal, & Hsu, 2000).

Let a sequence be  $s = \langle s_1, s_2, \dots, s_k \rangle$ , the itemset  $\bigcup_{i=1}^k s_i$  is defined as *projected itemset* of  $s$ . The algorithm's methodology obeys the property “if an itemset  $X$  is infrequent, sequences whose projected itemset contains  $X$  is infrequent too”. The FreeSpan algorithm has two main steps as follows (Han, Pei, Mortazavi-Asl, Chen, Dayal, & Hsu, 2000, p. 4):

1. Determine a descending ordered set of all frequent item  $f\_list = \{x_1, x_2, \dots, x_n\}$ . Note that  $support(x_1) > support(x_2) > \dots > support(x_n)$ . According to  $f\_list$ , all frequent sequences can be divided into  $n$  disjointed subset  $FS_i$  following condition:  $FS_i$  contains items  $x_i$  (s) but no item after  $x_i$ .

2. Each subset  $FS_i$  is found separately by mining particularly  $x_i$ -projected database by other algorithms such as GSP, AprioriAll, and SPADE where  $x_i$ -projected database is constructed by removing all items (after  $x_i$ ) from each sequence of original database. Frequent item matrix can be constructed so as to enhance FreeSpan algorithm (Han, Pei, Mortazavi-Asl, Chen, Dayal, & Hsu, 2000, p. 2). For example, if sequence  $s$  in original database contains  $x_i$ , it is replaced by sub-sequence  $s'$  which is derived by removing  $x_i$  from  $s$ .

For example, from sequence database shown in table III.3.1,  $f\_list$  is determined along with the support of each item in form  $\{x_1:support(x_1); x_2:support(x_2); \dots; x_n:support(x_n)\}$  as follows:

$$f\_list = \{o:4, c:3, f:3, s:2, d:1, p:1, v:1, l:1\}$$

Let  $min\_sup$  be 25%, applying FreeSpan for  $o, c$ -projected databases, we have results as below:

	Sequences	Frequent sequences (mined by frequent item matrix, GPS, AprioriAll, SPADE, etc.)
$o$ -projected database	$\langle o \rangle, \langle o \rangle, \langle o \rangle, \langle o \rangle$	$\langle o \rangle:4$
$c$ -projected database	$\langle o \rangle, \langle oc \rangle, \langle (oc) \rangle, \langle oc \rangle$	$\langle c \rangle:3, \langle oc \rangle:2$

**Table III.3.7.** Frequent sequences mined from projected databases

FreeSpan is more efficient than Apriori-like algorithms since it projects recursively a large database into smaller databases according to current frequent sequences. Generation of longer sequences is done on such small databases leading to a smaller set of candidate. However, FreeSpan can create redundant projected databases, which affects performance.

This sub-section III.3.1.1.2 ends up the brief survey of sequential pattern mining. Next sub-section III.3.1.2 proposes the method to break sequential pattern into sequential rules supporting learning concept recommendation with attention that learning sequential patterns are mined from learning history sub-model and [mining engine \(ME\)](#) of [Zebra](#) server is responsible for the mining task. Please see sub-section II.2.2 for more details about ME and Zebra.

### III.3.1.2. A proposal of breaking sequential pattern in learning context

Given sequence database (shown in table III.3.1) managed by learning history sub-model, suppose we discovered the sequential pattern  $\langle osc(sc) \rangle$  which means:

“*class & OOP*”–”*selection structure*”–”*control structure*”–”*selection structure, control structure*”

and some other patterns as results from aforementioned mining algorithms such as [AprioriAll](#), [GSP](#), and [SPADE](#). We accept that these patterns like the learning “routes” that student preferred or learned often in past but in the next time if a

student chooses the concept “*control structure*”, the adaptive learning system should recommend which next concepts in above patterns? So the patterns should be broken into association rules with their confidence. For example, breaking above pattern  $\langle osc(sc) \rangle$  follows steps:

1. Breaking entire  $\langle osc(sc) \rangle$  into itemsets such as  $o$ ,  $s$ ,  $c$ ,  $(sc)$  and determining all possible large 2-sequences which are 2-arrangement of all itemsets following the criterion: order of 2-sequences must comply with the order of sequential pattern. There are six large 2-sequences:  $\langle os \rangle$ ,  $\langle oc \rangle$ ,  $\langle o(sc) \rangle$ ,  $\langle sc \rangle$ ,  $\langle s(sc) \rangle$ ,  $\langle c(sc) \rangle$ . Thus, we have six rules derived from these large 2-sequences in form: “*left-hand itemset  $\rightarrow$  right-hand itemset*”, for example, rule “ $s \rightarrow c$ ” derived from 2-sequence  $\langle sc \rangle$ .
2. Computing the confidences of such rules and sorting them according to these measures. The confidence of a rule is the ratio of the support of 2-sequences and the support of left-hand itemset,  $confidence(x \rightarrow y) = support(\langle xy \rangle) / support(\langle x \rangle)$ . The rules whose confidence is less than threshold  $min\_conf$  is removed in order to ensure that remains are strong rules. We called these rules as **sequential rules** and these confidences as sequential confidences, as shown in table III.3.8.

sequential rules	sequential confidences
$o \rightarrow s$	40%
$o \rightarrow c$	40%
$o \rightarrow sc$	40%
$s \rightarrow c$	0%
$s \rightarrow sc$	0%
$c \rightarrow sc$	0%

**Table III.3.8.** Sequential rules

Now, if student choose the concept (itemset)  $x$ , system will find whole rules broken from all sequential patterns and the *left-hand itemset* of each rule must contain  $x$ . Then, these rules are sorted by their confidences in descending order. Final outcome is an ordered list of *right-hand itemsets* (concepts) of rules, which are recommended to students. The top concept (itemset) in such list is referred as the most necessary concept. For example, according to sequential rules (shown in table III.3.8) resulted from the proposed method, if a user choose concept “*class & OOP*” denoted  $o$ , we recommend her/his other concepts  $s$  and  $c$  which are right-hand itemsets of the highest-confidence rules  $o \rightarrow s$  and  $o \rightarrow c$ . Table III.3.9 shows these recommended concepts  $s$  and  $c$ .

Concept	Confidence (rate)
“ <i>selection structure</i> ”	40%
“ <i>control structure</i> ”	40%

**Table III.3.9.** Recommended learning concepts constructed from sequential rules

This methodology is similar to mining association rules but it achieves high performance and precise prediction since it derived from result of sequential pattern mining process. The sequential rules stick close on user's learning "route" because they are mined in accordance with sequential pattern and pay attention to the sequence order (Nguyen & Do, Learning Concept Recommendation based on Sequential Pattern Mining, 2009).

Next sub-section [III.3.1.3](#) is the evaluation of proposed method of breaking sequential patterns.

### **III.3.1.3. Evaluation**

Although sequential pattern mining is applied in e-commercial for customer purchase but the extracted patterns can be used to predict user's learning "route", which is fundamental of learning object recommendation. There are two sequential pattern mining approaches: candidate generation-and-test, pattern-growth. The first which is essentially a refinement of the Apriori-like is easy to implement but causes a problem of large candidate set and so, leads to performance downfall and requirement of both complex computation and huge storage. Especially, in e-learning environment, there are thousands of students; speed and performance are very important factors. The second which is a divide-and-conquer solution intends to reduce the number of candidate sequences. So, it is more efficient.

Last, I propose the technique to break sequential patterns into rules containing recommendable concepts / learning materials. The ideology of this approach is derived from association rule mining but it is more efficient than association rules.

Learning concept recommendation is the useful extended function of [Zebra](#), which is supported by mining engine (ME) and learning history sub-model. The second extended function discovering user interests is described in next section [III.3.2](#), which is also supported by ME and learning history sub-model.

### **III.3.2. Discovering user interests by document classification**

User interest is one of personal traits attracting researchers' attention in user modeling and user profiling. User interests are known simply as user's preferences, likes, and dislikes on something. User interest is not specified in Triangular Learner Model (TLM) shown in figure [II.6](#) when TLM consists of knowledge sub-model (see section [III.1](#)), learning style sub-model (see section [III.2](#)) and learning history sub-model (in this section [III.3](#)). However, user interest is a feature of extended TLM and managed by learning history sub-model, as shown in figure [II.7](#). Similar to learning concept recommendation mentioned in previous sub-section [III.3.1](#), discovering user interest is the second extended function of learning history sub-model and performed by [mining engine](#) (ME); please see section [II.2](#) for more details about ME and the

### III.3. Learning history sub-model

---

user modeling system Zebra. As aforementioned at [the beginning of this section III.3](#), there are three extended functions executed in learning history sub-model such as learning concept recommendation, discovering user interests, and constructing learner groups.

User interest competes with user knowledge to become the most important characteristics in user model. Adaptive systems need to know user interests so that provide adaptation to user. For example, adaptive learning systems tailor learning materials (lessons, examples, exercises, tests, etc.) to user interests. I propose a new approach for discovering user interest based on document classification (Nguyen, A Proposal Discovering User Interests by Support Vector Machine and Decision Tree on Document Classification, 2009). The basic idea is to consider user interests as classes of documents. The process of classifying documents is also the process of discovering user interests. There are two new points of view:

- The series of user access in her/his history are modeled as documents. So user is referred indirectly to as “document”.
- User interests are classes such documents are belong to.

My approach includes four following steps, as shown in table [III.3.10](#).

1. Documents in training corpus are represented according to *vector model*. Each element of vector is product of term frequency and inverse document frequency. However the inverse document frequency can be removed from each element for convenience.
2. *Classifying training corpus* by applying classification methods such as decision tree, support vector machine, and neural network. *Classifiers* must be constructed by these methods; for example, classification rules are drawn from decision tree and weight vectors are drawn from support vector machine. Classification rules and weight vectors are typical classifiers.
3. Mining user’s access history to *find maximum frequent itemsets*. Each itemset is considered an *interesting document* and its member items are considered as terms. Such interesting documents are modeled as vectors.
4. Applying classifiers (see step 2) into these interesting documents (see step 3) order to choose which classes are most suitable to these interesting documents. *Such classes are user interests*.

**Table III.3.10.** Proposed approach to discover user interests

This approach bases on document classification but it also relates to information retrieval in the manner of representing documents. Hence sub-section [III.3.2.1](#) discusses about vector model for representing documents, according to step 1. Supervised learning methods such as support vector machine, decision tree, and neural network and their application on document classification are mentioned in sub-section [III.3.2.2](#), according to step 2 (Nguyen, Discovering User Interests by Document Classification, 2010). The main approach to discover user interest is described in sub-section [III.3.2.3](#), according to steps 3 and 4. Sub-section [III.3.2.4](#) is the evaluation.

### III.3.2.1. Vector model for representing documents

The main purpose of this sub-section is to introduce vector model of document according to step 1 of proposed approach to discover user interests as described in table III.3.10. Suppose corpus  $\mathcal{D}$  is the composition of documents,  $\mathcal{D} = \{D_1, D_2, \dots, D_n\}$ . *Corpus*, which is the terminology in subject of information retrieval (Manning, Raghavan, & Schütze, 2009, p. 4), shares the similar meaning to other terms such as evidences, evidence sample, data sample, sample, training data, training set, and training dataset. Every document  $D_i$  contains a set of key words so-called *terms*. The number of times a term occurs in a document is called *term frequency* ( $tf$ ). Given the document  $D_i$  and term  $t_j$ , the term frequency  $tf_{ij}$  measuring the importance of term  $t_j$  within document  $D_i$  is defined by formula III.3.1 as follows:

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{ik}}$$

*Formula III.3.1.* Term frequency

Where  $n_{ij}$  is the number of occurrences of term  $t_j$  in document  $D_i$ , and the denominator is the sum of number of occurrences of all terms in document  $D_i$ .

Suppose we need to search documents which are most relevant to the query having term  $t_j$ . The simple way is to choose documents which have highest term frequency  $tf_{ij}$ . However in situation that  $t_j$  is not a good term to distinguish between relevant and non-relevant documents, other terms occurring rarely are better ones to distinguish between relevant and non-relevant documents. This will tend to incorrectly emphasize documents containing term  $t_j$  more, without giving enough weight to other meaningful terms. So the *inverse document frequency* ( $idf$ ) is a measure of general importance of the term. It is used to decrease the weight of terms occurring frequently and increase the weight of terms occurring rarely. The inverse document frequency of term  $t_j$  is the ratio of the size of corpus to the number of documents that  $t_j$  occurs. The inverse document frequencies exist if all documents are indexed. Formula III.3.2 specifies inverse document frequency.

$$idf_j = \log \frac{|\mathcal{D}|}{|\{D_i : (t_j \in D_i) \text{ and } (D_i \in \mathcal{D})\}|}$$

*Formula III.3.2.* Inverse document frequency

Where  $|\mathcal{D}|$  is the total number of documents in corpus  $\mathcal{D}$  and  $|\{D_i : (t_j \in D_i) \text{ and } (D_i \in \mathcal{D})\}|$  is the number of documents  $D_i$  (s) containing term  $t_j$  with condition that such documents belong to corpus  $\mathcal{D}$ . The logarithm function  $\log(\cdot)$  is used to normalize  $idf_j$  so that it is less than 1.

The weight of term  $t_j$  in document  $D_i$  is defined as product of  $tf_{ij}$  and  $idf_j$ , which is specified by formula III.3.3 as follows:

$$w_{ij} = tf_{ij} * idf_j$$

*Formula III.3.3.* The weight  $w_{ij}$  of term  $t_j$  in document  $D_i$

This weight measures the importance of a term in a document over the corpus. It increases proportionally to the number of times a term occurs in the document but is offset by the frequency of this term in the corpus. In general this weight balances the importance of two measures: term frequency and inverse document frequency.

Suppose there are  $p$  terms  $\{t_1, t_2, \dots, t_p\}$ , each document  $D_i$  is modeled as the vector which is composed of weights of such terms. *Formula III.3.4* expresses the document vector which is the result of step 1 of proposed approach for discovering user interests (see table [III.3.10](#)).

$$D_i = (w_{i1}, w_{i2}, w_{i3}, \dots, w_{ip})$$

*Formula III.3.4.* Vector model of document  $D_i$

Where the weights  $w_{ij} = tf_{ij} * idf_j$  specified by formula [III.3.3](#).

If inverse document frequencies are ignored (corpus is not indexed), document vector  $D_i$  can be represented shortly by a set of term frequencies. *Formula III.3.4'* specifies such brief representation of document vector.

$$D_i = (tf_{i1}, tf_{i2}, tf_{i3}, \dots, tf_{ip})$$

*Formula III.3.4'.* Document vector as a set of term frequencies

The corpus becomes a matrix  $n \times p$ , which have  $n$  rows and  $p$  columns with respect to  $n$  documents and  $p$  terms.  $D_i$  is called *document vector*.

The essence of document classification is to use supervised learning algorithms in order to classify corpus into groups of documents; each group is labeled, for example, +1 and -1. I apply three methods namely support vector machine, decision tree and neural network into document classification. These methods are described in successive sub-section [III.3.2.2](#).

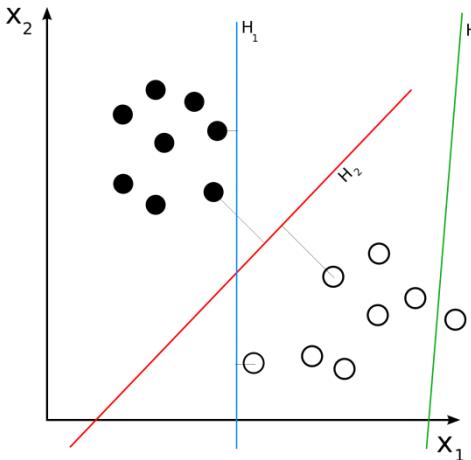
### III.3.2.2. Methods of document classification

This sub-section [III.3.2.2](#) describes three supervised learning methods such as support vector machine (part [III.3.2.2.1](#)), decision tree (part [III.3.2.2.2](#)), neural network (part [III.3.2.2.3](#)) together with their application on document classification, according to step 2 of proposed approach for discovering user interest as described in table [III.3.10](#).

#### III.3.2.2.1. Document classification based on support vector machine

**Support vector machine (SVM)** (Law, 2006) is a supervised learning algorithm for classification and regression. Given a set of  $p$ -dimensional vectors in vector space, SVM finds the *separating hyperplane* that splits vector

space into sub-set of vectors; each separated sub-set (so-called data set) is assigned one class. There is the condition for this separating hyperplane: “it must maximize the margin between two sub-sets”. Figure III.3.1 (Wikibooks, 2008) shows separating hyperplanes  $H_1$ ,  $H_2$ , and  $H_3$  in which only  $H_2$  gets maximum margin according to this condition.



**Figure III.3.1.** Separating hyperplanes

Suppose we have some  $p$ -dimensional vectors; each of them belongs to one of two classes. We can find many  $p-1$  dimensional hyperplanes that classify such vectors but there is only one hyperplane that maximizes the margin between two classes. In other words, the nearest between a point in one side of this hyperplane and other side of this hyperplane is maximized. Such hyperplane is called *maximum-margin hyperplane* and it is considered as the SVM classifier.

Let  $\{X_1, X_2, \dots, X_n\}$  be the training set of  $n$  vectors  $X_i$  (s) and let  $y_i = \{+1, -1\}$  be the class label of vector  $X_i$ . Each  $X_i$  is also called a data point with attention that *vectors can be identified with data points and data points indicate documents* mentioned in sub-section III.3.2.1. Data point can be called *point*, in brief. It is necessary to determine the maximum-margin hyperplane that separates data points belonging to  $y_i=+1$  from data points belonging to  $y_i=-1$  as clear as possible.

According to theory of geometry, arbitrary hyperplane is represented as a set of points satisfying *hyperplane equation* specified by formula III.3.5.

$$W \cdot X_i - b = 0$$

**Formula III.3.5.** Hyperplane equation

Where the sign “.” denotes the dot product or scalar product and  $W$  is *weight vector* perpendicular to hyperplane and  $b$  is the *bias*. Vector  $W$  is also called perpendicular vector or normal vector and it is used to specify hyperplane. Suppose  $W=(w_1, w_2, \dots, w_p)$  and  $X_i=(x_{i1}, x_{i2}, \dots, x_{ip})$ , the scalar product  $W \cdot X_i$  is:

$$W \cdot X_i = X_i \cdot W = w_1 x_{i1} + w_2 x_{i2} + \dots + w_p x_{ip} = \sum_{j=1}^p w_j x_{ij}$$

### III.3. Learning history sub-model

---

Given scalar value  $w$ , the multiplication of  $w$  and vector  $X_i$  denoted  $wX_i$  is a vector as follows:

$$wX_i = (wx_{i1}, wx_{i2}, \dots, wx_{ip})$$

Please distinguish scalar product  $W \cdot X_i$  and multiplication  $wX_i$ .

The essence of SVM method is to find out weight vector  $W$  and bias  $b$  so that the hyperplane equation specified by formula III.3.5 expresses the maximum-margin hyperplane that maximizes the margin between two classes of training set.

The value  $\frac{b}{|W|}$  is the offset of the (maximum-margin) hyperplane from the origin along the weight vector  $W$  where  $|W|$  or  $\|W\|$  denotes length or module of vector  $W$ .

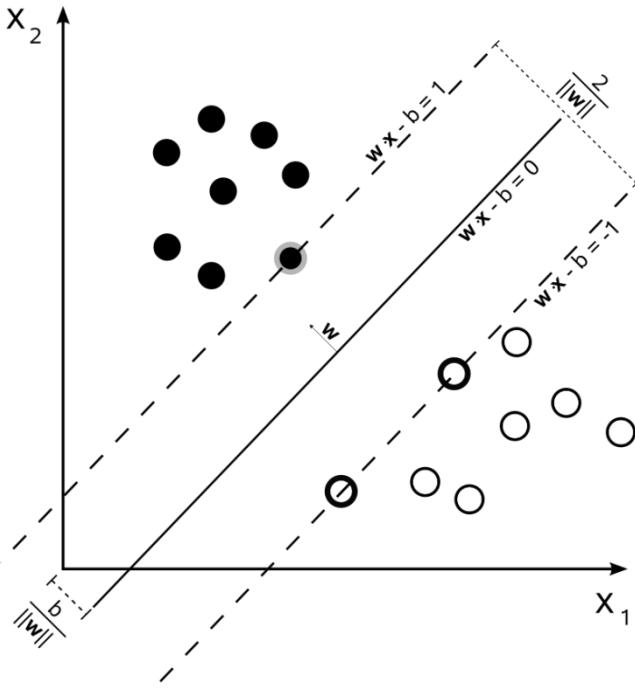
$$|W| = \|W\| = \sqrt{w_1^2 + w_2^2 + \dots + w_p^2} = \sqrt{\sum_{i=1}^p w_i^2}$$

Note that we use two notations  $|.|$  and  $\|.\|$  for denoting the length of vector. Additionally, the value  $\frac{2}{|W|}$  is the width of the margin as seen in figure III.3.2. To determine the margin, two parallel hyperplanes are constructed, one on each side of the maximum-margin hyperplane. Such two parallel hyperplanes are represented by two hyperplane equations, as shown in formula III.3.6 as follows.

$$\begin{aligned} W \cdot X_i - b &= 1 \\ W \cdot X_i - b &= -1 \end{aligned}$$

*Formula III.3.6. Equations of two parallel hyperplanes*

Figure III.3.2 (Wikibooks, 2008) illustrates maximum-margin hyperplane, weight vector  $W$  and two parallel hyperplanes. As seen in the figure III.3.2, the margin is limited by such two parallel hyperplanes. Exactly, there are two margins (each one for a parallel hyperplane) but it is convenient for referring both margins as the unified single margin as usual.



**Figure III.3.2.** Maximum-margin hyperplane, parallel hyperplanes and weight vector  $W$

To prevent vectors falling into the margin, all vectors belonging to two classes  $y_i=1$  and  $y_i=-1$  have two following constraints, respectively:

$$\begin{cases} W \cdot X_i - b \geq 1 & \text{(for } X_i \text{ belonging to class } y_i = +1) \\ W \cdot X_i - b \leq -1 & \text{(for } X_i \text{ belonging to class } y_i = -1) \end{cases}$$

As seen in figure III.3.2, vectors (data points) belonging to classes  $y_i=+1$  and  $y_i=-1$  are depicted as black circles and white circles. Such two constraints are unified into the so-called classification constraint specified by formula III.3.7 as follows:

$$y_i(W \cdot X_i - b) \geq 1 \Leftrightarrow 1 - y_i(W \cdot X_i - b) \leq 0$$

#### Formula III.3.7. Classification constraint

As known,  $y_i=+1$  and  $y_i=-1$  represent two classes of data points. It is easy to infer that maximum-margin hyperplane which is the result of SVM method is the classifier that aims to determine which class (+1 or -1) a given data point  $X$  belongs to. Your attention please, each data point  $X_i$  in training set was assigned by a class  $y_i$  before and maximum-margin hyperplane constructed from the training set is used to classify any different data point  $X$ .

Because maximum-margin hyperplane is defined by weight vector  $W$ , it is easy to recognize that the essence of constructing maximum-margin hyperplane is to solve the **constrained optimization problem** as follows:

$$\underset{W,b}{\text{minimize}} \frac{1}{2} |W|^2 \text{ subject to } y_i(W \cdot X_i - b) \geq 1, \forall i = \overline{1, n}$$

Where  $|W|$  is the length of weight vector  $W$  and  $y_i(W \cdot X_i - b) \geq 1$  is the classification constraint specified by formula III.3.7. The reason of minimizing

$\frac{1}{2}|W|^2$  is that distance between two parallel hyperplanes is  $\frac{2}{|W|}$  and we need to maximize such distance in order to maximize the margin for maximum-margin hyperplane. Then maximizing  $\frac{2}{|W|}$  is to minimize  $\frac{1}{2}|W|$ . Because it is complex to compute the length  $|W|$ , we substitute  $\frac{1}{2}|W|^2$  for  $\frac{1}{2}|W|$  when  $|W|^2$  is equal to the scalar product  $W \cdot W$  as follows:

$$|W|^2 = \|W\|^2 = W \cdot W = w_1^2 + w_2^2 + \cdots + w_p^2$$

The constrained optimization problem is re-written, shown in formula III.3.8 as below:

$$\begin{cases} \underset{W,b}{\text{minimize}} \ f(W,b) = \frac{1}{2}|W|^2 \\ \text{subject to } g_i(W,b) = 1 - y_i(W \cdot X_i - b) \leq 0, \forall i = \overline{1,n} \end{cases}$$

*Formula III.3.8.* Constrained optimization problem for constructing maximum-margin hyperplane

Where  $f(W,b) = \frac{1}{2}|W|^2$  is called target function with regard to two variables  $W$  and  $b$  although it is only dependent on variable  $W$ . Function  $g_i(W,b) = 1 - y_i(W \cdot X_i - b)$  is called constraint function with regard to two variables  $W$  and  $b$  and it is derived from the classification constraint specified by formula III.3.7. Because training set  $\{X_1, X_2, \dots, X_n\}$  has  $n$  data points  $X_i$  (s), there are  $n$  constraints  $g_i(W,b) \leq 0$ . The *Lagrangian function* (Boyd & Vandenberghe, 2009, p. 215) is constructed from constrained optimization problem specified by formula III.3.8. Let  $L(W, b, \lambda)$  be Lagrangian function, we have formula III.3.9 for representing Lagrangian function as follows:

$$\begin{aligned} L(W,b,\lambda) &= f(W,b) + \sum_{i=1}^n \lambda_i g_i(W,b) \\ &= \frac{1}{2}|W|^2 + \sum_{i=1}^n \lambda_i (1 - y_i(W \cdot X_i - b)) \\ &= \frac{1}{2}|W|^2 + \sum_{i=1}^n \lambda_i - W \cdot \left( \sum_{i=1}^n \lambda_i y_i X_i \right) + b \sum_{i=1}^n \lambda_i y_i \end{aligned}$$

*Formula III.3.9.* Lagrangian function

Where  $\lambda$  is  $n$ -component vector  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$  and each  $\lambda_i \geq 0$  is called Lagrange multiplier or Karush-Kuhn-Tucker multiplier (Wikipedia, Karush–Kuhn–Tucker conditions, 2014) or dual variables. Note that sign “.” denotes scalar product and every training data point  $X_i$  (s) was assigned by a class  $y_i$  before.

Suppose  $(W^*, b^*)$  is solution of constrained optimization problem specified by formula III.3.8 then, the pair  $(W^*, b^*)$  is minimum point of target function  $f(W, b)$  or target function  $f(W, b)$  gets minimum at  $(W^*, b^*)$  with all constraints

$g_i(W, b) \leq 0, \forall i = \overline{1, n}$ . Note that  $W^*$  is called *optimal weight vector* and  $b^*$  is called *optimal bias*. It is easy to infer that the pair  $(W^*, b^*)$  represents the maximum-margin hyperplane and it is possible to identify  $(W^*, b^*)$  with the maximum-margin hyperplane. The ultimate goal of SVM method is to find out  $W^*$  and  $b^*$ . According to Lagrangian duality theorem (Boyd & Vandenberghe, 2009, p. 216) (Jia, 2013, p. 8), the pair  $(W^*, b^*)$  is the extreme point of Lagrangian function as follows:

$$(W^*, b^*) = \underset{W, b}{\operatorname{argmin}} L(W, b, \lambda)$$

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \left( \min_{W, b} L(W, b, \lambda) \right)$$

#### Formula III.3.10. Lagrangian duality problem

Now it is necessary to solve the Lagrangian duality problem represented by formula III.3.10 to find out  $W^*$ . Thus, the Lagrangian function  $L(W, b, \lambda)$  is minimized with respect to the primal variables  $W, b$  and maximized with respect to the dual variables  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ , in turn. If gradient of  $L(W, b, \lambda)$  is equal to zero then,  $L(W, b, \lambda)$  will get minimum value with note that gradient of a multi-variable function is the vector whose components are first-order partial derivative of such function. Thus, setting the gradient of  $L(W, b, \lambda)$  with respect to  $W$  and  $b$  to zero, we have:

$$\begin{cases} \frac{\partial L(W, b, \lambda)}{\partial W} = 0 \\ \frac{\partial L(W, b, \lambda)}{\partial b} = 0 \end{cases} \Leftrightarrow \begin{cases} W - \sum_{i=1}^n \lambda_i y_i X_i = 0 \\ \sum_{i=1}^n \lambda_i y_i = 0 \end{cases} \Leftrightarrow \begin{cases} W = \sum_{i=1}^n \lambda_i y_i X_i \\ \sum_{i=1}^n \lambda_i y_i = 0 \end{cases}$$

In general,  $W^*$  is determined by formula III.3.11 as follows:

$$\begin{cases} W^* = \sum_{i=1}^n \lambda_i y_i X_i \\ \sum_{i=1}^n \lambda_i y_i = 0 \end{cases}$$

#### Formula III.3.11. Formula for determining optimal weight vector $W^*$

It is required to determine Lagrange multipliers  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$  in order to evaluate  $W^*$ . Substituting formula III.3.11 into Lagrangian function  $L(W, b, \lambda)$  specified by formula III.3.9, we have:

$$\begin{aligned} L(\lambda) &= \min_{W, b} L(W, b, \lambda) \\ &= \frac{1}{2} \left( \sum_{i=1}^n \lambda_i y_i X_i \right)^2 + \sum_{i=1}^n \lambda_i - \left( \sum_{i=1}^n \lambda_i y_i X_i \right) \cdot \left( \sum_{i=1}^n \lambda_i y_i X_i \right) \\ &\quad (\text{due to } L(W, b, \lambda) \text{ gets minimum at } W = \sum_{i=1}^n \lambda_i y_i X_i \text{ and } \sum_{i=1}^n \lambda_i y_i = 0) \end{aligned}$$

$$\begin{aligned}
 &= -\frac{1}{2} \left( \sum_{i=1}^n \lambda_i y_i X_i \right) \cdot \left( \sum_{i=1}^n \lambda_i y_i X_i \right) + \sum_{i=1}^n \lambda_i \\
 &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (X_i \cdot X_j) + \sum_{i=1}^n \lambda_i
 \end{aligned}$$

Where  $L(\lambda)$  is called *dual function* represented by formula III.3.12.

$$L(\lambda) = \min_{W,b} L(W,b,\lambda) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (X_i \cdot X_j) + \sum_{i=1}^n \lambda_i$$

*Formula III.3.12.* Dual function for determining Lagrange multipliers

According to Lagrangian duality problem represented by formula III.3.10,  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$  is calculated as the maximum point of dual function  $L(\lambda)$ . If the gradient of  $L(\lambda)$  is equal to zero then,  $L(\lambda)$  will get global maximum value because  $L(\lambda)$  is quadratic function of variable  $\lambda$ . Thus, setting the gradient of dual function  $L(\lambda)$  with respect to Lagrange multipliers  $\lambda_i$  (s) to zero, we have:

$$\begin{cases} \frac{\partial L(\lambda)}{\partial \lambda_1} = 0 \\ \frac{\partial L(\lambda)}{\partial \lambda_2} = 0 \\ \vdots \\ \frac{\partial L(\lambda)}{\partial \lambda_n} = 0 \end{cases} \Leftrightarrow \begin{cases} -\sum_{j=1}^n \lambda_1 y_1 y_j (X_1 \cdot X_j) + 1 = 0 \\ -\sum_{j=1}^n \lambda_2 y_2 y_j (X_2 \cdot X_j) + 1 = 0 \\ \vdots \\ -\sum_{j=1}^n \lambda_n y_n y_j (X_n \cdot X_j) + 1 = 0 \\ 1 \end{cases}$$

$$\Leftrightarrow \lambda_i = \frac{1}{(y_i X_i) \cdot (\sum_{j=1}^n y_j X_j)}, \forall i = \overline{1, n}$$

*Formula III.3.13.* Lagrange multipliers

Substituting Lagrange multiplier specified by formula III.3.13 into formula III.3.11, the optimal weight vector  $W^*$  is totally determined by formula III.3.14.

$$W^* = \sum_{i=1}^n \frac{1}{X_i \cdot (\sum_{j=1}^n y_j X_j)} X_i$$

*Formula III.3.14.* Optimal weight vector  $W^*$

The maximum-margin hyperplane (SVM classifier) is formed by both weight vector  $W$  and the bias  $b$ . According to Karush-Kuhn-Tucker condition (Wikipedia, Karush–Kuhn–Tucker conditions, 2014) (Boyd & Vandenberghe, 2009, p. 243), every bias  $b_i$  satisfies following constrained equation known as the complementary slackness of Karush-Kuhn-Tucker condition.

$$g_i(W, b_i) = 1 - y_i (W \cdot X_i - b_i) = 0, \forall i = \overline{1, n}$$

$$\Rightarrow W \cdot X_i - b_i = \frac{1}{y_i} = y_i, \forall i = \overline{1, n}$$

$$\Rightarrow b_i = W \cdot X_i - y_i, \forall i = \overline{1, n}$$

As usual, the optimal bias  $b^*$  is calculated as the mean of all bias  $b_i$  (s) over  $n$  training data points (Wikipedia, Support vector machine, 2014).

$$b^* = \frac{1}{n} \sum_{i=1}^n b_i = \frac{1}{n} \sum_{i=1}^n (W \cdot X_i - y_i)$$

When optimal weight vector  $W^*$  was determined, it is easy to compute the optimal bias  $b^*$ . Formula III.3.15 describes how to calculate optimal bias  $b^*$  according to mean method.

$$b^* = \frac{1}{n} \sum_{i=1}^n (W^* \cdot X_i - y_i)$$

*Formula III.3.15.* Optimal bias  $b^*$  according to mean method

When both optimal weight vector  $W^*$  and optimal bias  $b^*$  are determined, the maximum-margin hyperplane known as SVM classifier is totally determined. According to formula III.3.5, the equation of maximum-margin hyperplane is expressed in formula III.3.16 as follows:

$$W^* \cdot X - b^* = 0$$

*Formula III.3.16.* Equation of maximum-margin hyperplane (SVM classifier)

Where  $W^*$  and  $b^*$  are specified by formulas III.3.14 and III.3.15.

$$W^* = \sum_{i=1}^n \frac{1}{X_i \cdot (\sum_{j=1}^n y_j X_j)} X_i$$

$$b^* = \frac{1}{n} \sum_{i=1}^n (W^* \cdot X_i - y_i)$$

Recall that  $y_i = +1$  and  $y_i = -1$  represent two classes of data points.

For any data point  $X$ , classification rule derived from maximum-margin hyperplane (SVM classifier) is used to classify such data point  $X$ . In context of this sub-section III.3.2, data points are documents mentioned in sub-section III.3.2.1. Let  $R$  be the classification rule, formula III.3.17 specifies the classification rule as the sign function of point  $X$ .

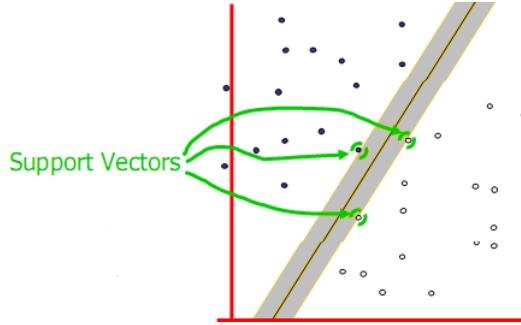
$$R \stackrel{\text{def}}{=} \text{sign}(W^* \cdot X - b^*) = \begin{cases} +1 & \text{if } W^* \cdot X - b^* \geq 0 \\ -1 & \text{if } W^* \cdot X - b^* < 0 \end{cases}$$

*Formula III.3.17.* Classification rule derived from maximum-margin hyperplane (SVM classifier)

### III.3. Learning history sub-model

After evaluating  $R$  with regard to  $X$ , if  $R(X) = 1$  then,  $X$  belongs to class +1; otherwise,  $X$  belongs to class -1. This is the simple process of data classification.

The SVM method now is described thoroughly but why this method is called support vector machine (SVM)? There are some vectors (data points) closest to the maximum-margin hyperplane. These vectors also lie on parallel hyperplanes. They contribute much to constructing the maximum margin and so they are called support vectors shown in figure III.3.3 (Moore, 2001, p. 5). That is the reason that this approach is called support vector machine (SVM).



**Figure III.3.3.** Support vectors

There is lack of accuracy if the optimal bias  $b^*$  is calculated via mean method as by formula III.3.15 because it is not exactly the bias of maximum-margin hyperplane. Therefore, I propose an iterative method for specifying the optimal bias  $b^*$ . For each training data point  $X_i$  in training set, the bias  $b_i = W^* \cdot X_i - y_i$  is calculated and the separating hyperplane  $H_i$  is determined as follows:

$$H_i \stackrel{\text{def}}{=} W^* \cdot X - b_i = W^* \cdot X - (W^* \cdot X_i - y_i) = 0$$

The classification rule  $R_i$  is derived from  $H_i$  as follows:

$$R_i \stackrel{\text{def}}{=} \text{sign}(W^* \cdot X - b_i)$$

The rule  $R_i$  is used to classify all data point  $X_i$  (s) in training set and all the classes resulted from  $R_i$  are compare to the real classes  $y_i$  (s) of these  $X_i$  (s). Let  $n_i$  be the number of wrong cases that the classes resulted from  $R_i$  are different from the real classes  $y_i$  (s). Which bias  $b_i$  causes the smallest  $n_i$  is the optimal bias  $b^*$ . Table III.3.11 describes the proposed iterative method to determine optimal bias  $b^*$ .

$n_{min} = +\infty$ $b^* = 0$ For each $X_i$ in training set $b_i = W^* \cdot X_i - y_i$ $H_i \stackrel{\text{def}}{=} W^* \cdot X - b_i = 0$ $R_i \stackrel{\text{def}}{=} \text{sign}(W^* \cdot X - b_i)$ $R_i$ is used to classify training set $n_i :=$ the number of wrong classification cases If $n_i < n_{min}$ Then $n_{min} = n_i$ $b^* = b_i$
--

```

        End If
    End For
    
```

**Table III.3.11.** Proposed iterative method to determine optimal bias  $b^*$

Now the **application of SVM into document classification** is described right here.

Recall that given corpus  $\mathcal{D} = \{D_1, D_2, \dots, D_n\}$  in which there are  $p$  terms  $\{t_1, t_2, \dots, t_p\}$ , every document  $D_i$  is modeled by a document vector as follows:

$$D_i = (w_{i1}, w_{i2}, w_{i3}, \dots, w_{ip})$$

Where weight  $w_{ij} = tf_{ij} * idf_j$  is the product of term frequency and inverse document frequency; please see formula III.3.4 for more details about document vector. Term frequency ( $tf$ ) and inverse document frequency ( $idf$ ) are mentioned in formulas III.3.1 and III.3.2.

If inverse document frequencies are ignored, document vector  $D_i$  can be represented by a set of term frequencies.

$$D_i = (tf_{i1}, tf_{i2}, tf_{i3}, \dots, tf_{ip})$$

Please see formulas III.3.4' for more about the short representation of document vector. For convenience, it is preferred to use such short representation of document vector over the whole sub-section III.3.2.

Given  $k$  classes  $\{c_1, c_2, \dots, c_k\}$ , there is demand of classifying documents into such classes. The technique of classification based on SVM is *two-class* classification in which the classes are  $+1$  and  $-1$  for  $y_i=+1$  and  $y_i=-1$ , respectively. So we need to determine the unique maximum-margin hyperplane referred to as *two-class classifier* with attention that such hyperplane is identified with the couple of optimal weight vector and optimal bias. It is possible to extend *two-class classification* to *k-class classification* by constructing  $k$  two-class classifier. It means that we must specify  $k$  couples of optimal weight vector  $W_i^*$  and bias  $b_i^*$ . Each couple  $(W_i^*, b_i^*)$  being a *two-class classifier* is the representation of class  $c_i$ . Each classification rule  $R_i$  is constructed based on classifier  $(W_i^*, b_i^*)$ . The process of finding classifier  $(W_i^*, b_i^*)$  in training corpus  $\mathcal{D}$  and constructing classification  $R_i$  was described in formulas III.3.14, III.3.15, and III.3.17 before. Table III.3.12 shows  $k$  classes along with  $k$  classifiers and  $k$  classification rules.

Class	Classifier	Classification rule
$c_1$	$(W_1^*, b_1^*)$	$R_1 = sign(W_1^* \cdot X - b_1^*)$
$c_2$	$(W_2^*, b_2^*)$	$R_2 = sign(W_2^* \cdot X - b_2^*)$
$\vdots$	$\vdots$	$\vdots$
$c_k$	$(W_k^*, b_k^*)$	$R_k = sign(W_k^* \cdot X - b_k^*)$

**Table III.3.12.**  $k$  couple  $(W_i^*, b_i^*)$  corresponds with  $k$  class  $\{c_1, c_2, \dots, c_k\}$   
For instance, classifying document  $D = (tf_1, tf_2, \dots, tf_p)$  is described as below:

### III.3. Learning history sub-model

---

1. For each classification rule  $R_i = W_i^* \cdot X + b_i^*$ , substituting each  $D$  into such rule. It means that vector  $X$  in such rule is replaced by document  $D$  and we get the classified value  $R_i(D)$ .
2. If there is a sub-set of rules  $\{R_{i_1}, R_{i_2}, \dots, R_{i_r}\}$  whose values  $\{R_{i_1}(D), R_{i_2}(D), \dots, R_{i_r}(D)\}$  equals 1 then, it is concluded that document  $D$  is belongs to  $r$  classes  $\{c_{i_1}, c_{i_2}, \dots, c_{i_r}\}$  where  $\{c_{i_1}, c_{i_2}, \dots, c_{i_r}\} \subseteq \{c_1, c_2, \dots, c_k\}$ . Document  $D$  is often belong to one class ( $r=1$ ) but there are some applications in which a document can be categorized into more than one class.

Table III.3.13 shows how to classify document with multi-classes.

Classification rule evaluation	Value
$R_1(D) = sign(W_1^* \cdot D - b_1^*)$	= +1 then $D \in c_1$ = -1 then $D \notin c_1$
$R_2(D) = sign(W_2^* \cdot D - b_2^*)$	= +1 then $D \in c_2$ = -1 then $D \notin c_2$
:	:
$R_k(D) = sign(W_k^* \cdot D - b_k^*)$	= +1 then $D \in c_k$ = -1 then $D \notin c_k$

**Table III.3.13.** Classifying document  $D$  with multi-classes

It is necessary to have an example for illustrating how to classify documents by SVM. Given a set of classes  $C = \{\text{computer science}, \text{math}\}$ , a set of terms  $T = \{\text{computer}, \text{derivative}\}$  and the corpus  $\mathcal{D} = \{\text{doc1.txt}, \text{doc2.txt}, \text{doc3.txt}, \text{doc4.txt}\}$ . The training corpus (training data) is shown in following table III.3.14 in which cell  $(i, j)$  indicates the number of times that term  $j$  (column  $j$ ) occurs in document  $i$  (row  $i$ ); in other words, each cell represents a term frequency and each row represents a document vector. Please see sub-section III.3.2.1 for more details about document vectors. Note that each document in this section belongs to only one class ( $r=1$ ).

	<i>computer</i>	<i>derivative</i>	<i>class</i>
<i>doc1.txt</i>	20	55	math
<i>doc2.txt</i>	20	20	computer science
<i>doc3.txt</i>	15	30	math
<i>doc4.txt</i>	35	10	computer science

**Table III.3.14.** Term frequencies of documents (SVM)

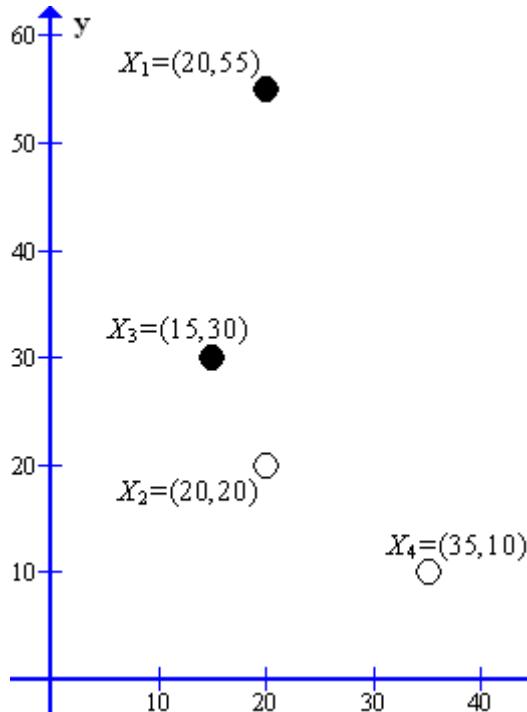
Let  $X_i$  be data points representing documents  $\text{doc1.txt}, \text{doc2.txt}, \text{doc3.txt}, \text{doc4.txt}, \text{doc5.txt}$ . We have  $X_1=(20,55)$ ,  $X_2=(20,20)$ ,  $X_3=(15,30)$ , and  $X_4=(35,10)$ . Let  $y_i=+1$  and  $y_i=-1$  represent classes “math” and “computer science”, respectively. Let  $x$  and  $y$  represent terms “computer” and “derivative”, respectively and so, for example, it is interpreted that the data point  $X_1=(20,55)$  has abscissa  $x=20$  and ordinate  $y=55$ . Therefore, term

frequencies from table III.3.14 is interpreted as SVM input training corpus shown in table III.3.15.

	$x$	$y$	$y_i$
$X_1$	20	55	+1
$X_2$	20	20	-1
$X_3$	15	30	+1
$X_4$	35	10	-1

**Table III.3.15.** Training corpus (SVM)

Data points  $X_1$ ,  $X_2$ ,  $X_3$ , and  $X_4$  are depicted in figure III.3.4 in which classes “math” ( $y_i=+1$ ) and “computer” ( $y_i=-1$ ) are represented by shading and hollow circles, respectively. Note that some figures (like figure III.3.4) in this research are drawn by the software *Graph* <http://www.padawan.dk> developed by author Ivan Johansen (Johansen, 2012).



**Figure III.3.4.** Data points in training data (SVM)

By applying formulas III.3.14 and III.3.15 into training corpus shown in table III.3.15, it is easy to calculate optimal weight vector  $W^*$  and optimal bias  $b^*$ . We have:

$$\begin{aligned}
 \sum_{j=1}^4 y_j X_j &= y_1 X_1 + y_2 X_2 + y_3 X_3 + y_4 X_4 \\
 &= 1 * (20, 55) - 1 * (20, 20) + 1 * (15, 30) - 1 * (35, 10) \\
 &= (-20, 55)
 \end{aligned}$$

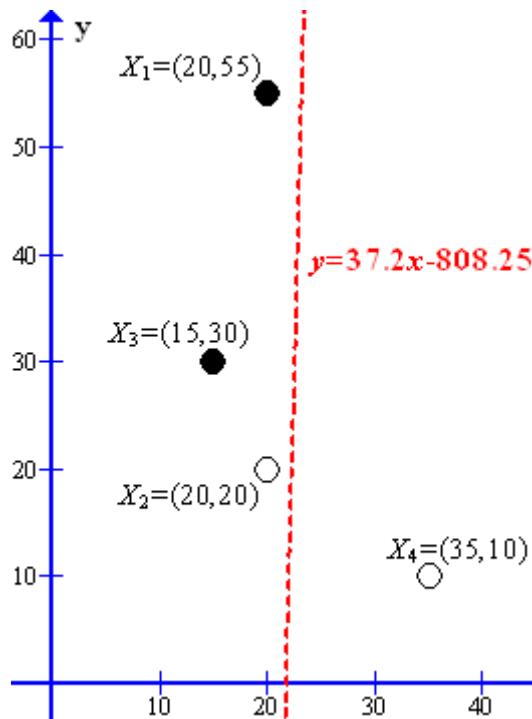
$$\begin{aligned}
 W^* &= \sum_{i=1}^4 \frac{1}{X_i \cdot (\sum_{j=1}^4 y_j X_j)} X_i = \sum_{i=1}^4 \frac{1}{X_i \cdot (-20,55)} X_i \\
 &= \frac{1}{(20,55) \cdot (-20,55)} (20,55) + \frac{1}{(20,20) \cdot (-20,55)} (20,20) \\
 &\quad + \frac{1}{(15,30) \cdot (-20,55)} (15,30) + \frac{1}{(35,10) \cdot (-20,55)} (35,10) \\
 &= \frac{1}{2625} (20,55) + \frac{1}{700} (20,20) + \frac{1}{1350} (15,30) - \frac{1}{150} (35,10) \\
 &\approx (-0.186, 0.005)
 \end{aligned}$$

$$\begin{aligned}
 b^* &= \frac{1}{4} \sum_{i=1}^4 (W^* \cdot X_i - y_i) \\
 &= \frac{1}{4} ((-0.186, 0.005) \cdot (20,55) - 1) + \frac{1}{4} ((-0.186, 0.005) \cdot (20,20) + 1) \\
 &\quad + \frac{1}{4} ((-0.186, 0.005) \cdot (15,30) - 1) + \frac{1}{4} ((-0.186, 0.005) \cdot (35,10) + 1) \\
 &= -4.04125
 \end{aligned}$$

After  $W^*$  and  $b^*$  were determined, the maximum-margin hyperplane (SVM classifier) is totally determined as below:

$$\begin{aligned}
 W^* \cdot X - b^* = 0 &\Leftrightarrow (-0.186, 0.005) \cdot (x, y) + 4.04125 = 0 \\
 &\Leftrightarrow y = 37.2x - 808.25
 \end{aligned}$$

The SVM classifier  $y = 37.2x - 808.25$  is depicted in figure III.3.5.



**Figure III.3.5.** An example of maximum-margin hyperplane

Where the maximum-margin hyperplane is draw as dash red line - - -. Derived from the above classifier  $y = 37.2x - 808.25$ , the classification rule is:

$$\begin{aligned} R &\stackrel{\text{def}}{=} \text{sign}(-0.186x + 0.005y + 4.04125) \\ &= \begin{cases} +1 & \text{if } -0.186x + 0.005y + 4.04125 \geq 0 \\ -1 & \text{if } -0.186x + 0.005y + 4.04125 < 0 \end{cases} \end{aligned}$$

Now we apply classification rule  $R \stackrel{\text{def}}{=} \text{sign}(-0.186x + 0.005y + 4.04125)$  into document classification. Suppose the numbers of times that terms “computer” and “derivative” occur in document  $D$  are 40 and 20, respectively. We need to determine which class document  $D=(40, 20)$  is belongs to. We have:

$$\begin{aligned} R(D) &= \text{sign}(-0.186 * 40 + 0.005 * 20 + 4.04125) = \text{sign}(-3.29875) \\ &= -1 \end{aligned}$$

Hence, it is easy to infer that document  $D$  belongs to class “computer science” ( $y_i=-1$ ).

The bias  $b^*$  can be enhanced by the proposed iterative method described in table [III.3.11](#). The execution of such method is shown in following table [III.3.16](#), in which four points  $X_1, X_2, X_3, X_4$  are fed into the method. The number  $n_i$  of wrong classification cases is calculated at each iteration.

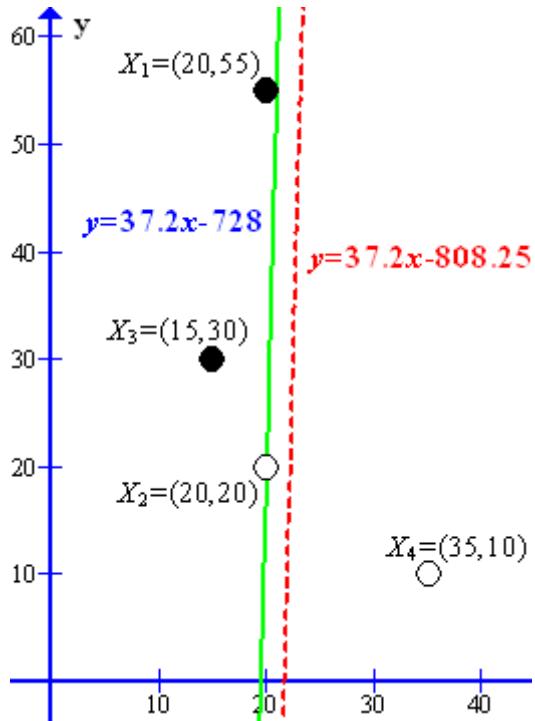
	$b_i$	$R_i$	$n_i$
$X_1=(20,55)$	$b_1 = (-0.186, 0.005) \cdot (20, 55) - 1 = -4.445$	$R_1 = \text{sign}(-0.186x + 0.005y + 4.445)$	$R_1(X_1)=+1 \text{ (true)}$ $R_1(X_2)=-1 \text{ (wrong)}$ $R_1(X_3)=+1 \text{ (true)}$ $R_1(X_4)=-1 \text{ (true)}$ $\Rightarrow n_1 = 1$
$X_2=(20,20)$	$b_2 = (-0.186, 0.005) \cdot (20, 20) + 1 = -2.62$	$R_2 = \text{sign}(-0.186x + 0.005y + 2.62)$	$R_2(X_1)=-1 \text{ (wrong)}$ $R_2(X_2)=-1 \text{ (true)}$ $R_2(X_3)=-1 \text{ (wrong)}$ $R_2(X_4)=-1 \text{ (true)}$ $\Rightarrow n_2 = 2$
$X_3=(15,30)$	$b_3 = (-0.186, 0.005) \cdot (15, 30) - 1 = -3.64$	$R_3 = \text{sign}(-0.186x + 0.005y + 3.64)$	$R_3(X_1)=+1 \text{ (true)}$ $R_3(X_2)=-1 \text{ (wrong)}$ $R_3(X_3)=+1 \text{ (true)}$ $R_3(X_4)=-1 \text{ (true)}$ $\Rightarrow n_3 = 1$
$X_4=(35,10)$	$b_4 = (-0.186, 0.005) \cdot (35, 10) + 1 = -5.46$	$R_4 = \text{sign}(-0.186x + 0.005y + 5.46)$	$R_4(X_1)=+1 \text{ (true)}$ $R_4(X_2)=-1 \text{ (wrong)}$ $R_4(X_3)=+1 \text{ (true)}$ $R_4(X_4)=-1 \text{ (true)}$ $\Rightarrow n_4 = 1$

**Table III.3.16.** Execution of proposed iterative method for enhancing the bias  $b^*$

Because the biases  $b_1=-4.445$ ,  $b_3=-3.64$ , and  $b_4=-5.46$  result out the minimum wrong cases of classification  $n_1=n_3=n_4=1$ , we can take any of them as enhanced bias. For example, we select  $b^*=b_3=-3.64$  and so, the maximum-margin hyperplane becomes:

$$\begin{aligned} W^* \cdot X - b_1 &= 0 \Leftrightarrow (-0.186, 0.005) \cdot (x, y) + 3.64 = 0 \\ &\Leftrightarrow y = 37.2x - 728 \end{aligned}$$

The maximum-margin hyperplane  $y = 37.2x - 728$  with enhance bias  $b^*=-3.64$  is drawn as solid line green line in figure [III.3.6](#).



**Figure III.3.6.** Maximum-margin hyperplane with enhanced bias  
Derived from the classifier  $y = 37.2x - 728$ , the classification rule is:

$$\begin{aligned} R &\stackrel{\text{def}}{=} \text{sign}(-0.186x + 0.005y + 3.64) \\ &= \begin{cases} +1 & \text{if } -0.186x + 0.005y + 3.64 \geq 0 \\ -1 & \text{if } -0.186x + 0.005y + 3.64 < 0 \end{cases} \end{aligned}$$

Now we apply classification rule  $R \stackrel{\text{def}}{=} \text{sign}(-0.186x + 0.005y + 3.64)$  into document classification. Suppose the numbers of times that terms “computer” and “derivative” occur in document  $D$  are 40 and 20, respectively. We need to determine which class document  $D=(40, 20)$  belongs to. We have:

$$R(D) = \text{sign}(-0.186 * 40 + 0.005 * 20 + 3.64) = \text{sign}(-3.7) = -1$$

Hence, it is easy to infer that document  $D$  belongs to class “computer science” ( $y_i=-1$ ).

Now the SVM method is described comprehensively with full of example. The next part [III.3.2.2.2](#) mentions another document classification method based on decision tree.

### [III.3.2.2.2. Document classification based on decision tree](#)

Given a set of classes  $C = \{\text{computer science}, \text{math}\}$ , a set of terms  $T = \{\text{computer}, \text{programming language}, \text{algorithm}, \text{derivative}\}$  and the corpus  $\mathcal{D} = \{\text{doc1.txt}, \text{doc2.txt}, \text{doc3.txt}, \text{doc4.txt}, \text{doc5.txt}, \text{doc6.txt}\}$ . The training corpus (training data) is shown in following table [III.3.17](#) in which cell  $(i, j)$  indicates the number of times that term  $j$  (column  $j$ ) occurs in document  $i$  (row  $i$ ); in other words, each cell represents a term frequency and each row represents a document vector. Please see sub-section [III.3.2.1](#) for more details about document vectors.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	<b>class</b>
<i>doc1.txt</i>	5	3	1	1	computer science
<i>doc2.txt</i>	5	5	40	50	math
<i>doc3.txt</i>	20	5	20	55	math
<i>doc4.txt</i>	20	55	5	20	computer science
<i>doc5.txt</i>	15	15	40	30	math
<i>doc6.txt</i>	35	10	45	10	computer science

**Table III.3.17.** Training corpus – Term frequencies of documents (decision tree)

It is required to normalize term frequencies. Let  $tf_{11}=5$ ,  $tf_{12}=3$ ,  $tf_{13}=1$ , and  $tf_{14}=1$  be the frequencies of terms “computer”, “programming language”, “algorithm”, and “derivative”, respectively of document “*doc1.txt*”, for example, these terms are normalized as follows:

$$tf_{11} = \frac{5}{5 + 3 + 1 + 1} = 0.5$$

$$tf_{12} = \frac{3}{5 + 3 + 1 + 1} \approx 0.3$$

$$tf_{13} = \frac{1}{5 + 3 + 1 + 1} = 0.1$$

$$tf_{14} = \frac{1}{5 + 3 + 1 + 1} = 0.1$$

Table [III.3.18](#) shows normalized term frequencies in corpus  $\mathcal{D}$ .

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	<b>class</b>
<i>doc1.txt</i>	0.5	0.3	0.1	0.1	computer science
<i>doc2.txt</i>	0.05	0.05	0.4	0.5	math
<i>doc3.txt</i>	0.2	0.05	0.2	0.55	math
<i>doc4.txt</i>	0.2	0.55	0.05	0.2	computer science
<i>doc5.txt</i>	0.15	0.15	0.4	0.3	math
<i>doc6.txt</i>	0.35	0.1	0.45	0.1	computer science

**Table III.3.18.** Training corpus – Normalized term frequencies (decision tree)

Because the expense of real number computation is so high, all normalized term frequencies are changed from real numbers into nominal values as following specifications:

$0 \leq tf < 0.2$	→ low
$0.2 \leq tf < 0.5$	→ medium
$0.5 \leq tf$	→ high

Table [III.3.19](#) shows nominal term frequencies according to these converted specifications.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	<b>class</b>
<i>doc1.txt</i>	high	medium	low	low	computer science
<i>doc2.txt</i>	low	low	medium	high	math
<i>doc3.txt</i>	medium	low	medium	high	math
<i>doc4.txt</i>	medium	high	low	medium	computer science
<i>doc5.txt</i>	low	low	medium	medium	math
<i>doc6.txt</i>	medium	low	medium	low	computer science

**Table III.3.19.** Training corpus – Nominal term frequencies

Decision tree induction (Han & Kamber, 2006, p. 291) is a machine learning method to build up **decision tree** whose leaf nodes are classes or labels of data (like “*computer science*” and “*math*” in table III.3.19) and non-leaf nodes are attributes of data (like “*computer*”, “*programming language*”, “*algorithm*”, and “*derivative*”). In similar to SVM method aforementioned in previous part III.3.2.2.1, classification rules are derived from decision tree where decision tree is considered as the classifier of decision tree induction. The basic idea of generating decision tree (Mitchell, 1997, pp. 52-77) is to split the tree into two sub-trees at the most informative node where each node represents an attribute. Such informative node is chosen by computing its *information gain* (Mitchell, 1997, pp. 57-58) (Han & Kamber, 2006, pp. 297-298). The more the information gain is, the more informative the node is. Training corpus is used to train (generate) decision tree is called corpus. Recall that the corpus consists of many data row and each row has many attributes. There is a so-called *class attribute* which is used to group (to classify) rows. All attributes except class attribute are represented as non-leaf nodes and class attribute is represented as leaf node in decision tree. Note that leaf node is the node having no children node and non-leaf node is the node having at least one child node. If decision tree is used to classify document then, rows represent documents and *non-class attributes* are terms; in this case, the corpus becomes a matrix  $n \times p$ , which have  $n$  rows and  $p$  columns with respect to  $n$  document vectors and  $p$  terms. Tables III.3.17, III.3.18, and III.3.19 are typical examples of corpus, in which class attribute is represented by column “*class*” and non-class attributes are terms “*computer*”, “*programming language*”, “*algorithm*”, and “*derivative*”. Now we identify attributes with both nodes and terms in decision tree.

Let  $\text{Entropy}(\mathcal{D})$  denote entropy of corpus  $\mathcal{D}$ , formula III.3.18 (Han & Kamber, 2006, p. 297) is used to calculate  $\text{Entropy}(\mathcal{D})$ .

$$\text{Entropy}(\mathcal{D}) = - \sum_{i=1}^n P_i \log_2(P_i)$$

*Formula III.3.18.* Entropy of training corpus

Where  $P_i$  is the frequency of occurrence of class  $C_i$ , for example, we have  $C_1$ =“*computer science*” and  $C_2$ =“*math*” given training corpus shown in table

**III.3.19.** The  $\log_2(P_i)$  is the logarithm with base 2 of  $P_i$ . In practice,  $P_i$  is computed as the ratio of the number of rows containing class  $C_i$  to the total number of rows inside training corpus.

$$P_i = \frac{\text{The number of rows containing class } C_i}{\text{The total number of rows inside training corpus}}$$

Given training corpus shown in table [III.3.19](#), we have:

$$\text{Entropy}(\mathcal{D}) = - \sum_{i=1}^2 P_i \log_2(P_i) = - \left( \frac{3}{6} \log_2 \left( \frac{3}{6} \right) + \frac{3}{6} \log_2 \left( \frac{3}{6} \right) \right) = 1$$

(Because there are 3 rows containing attribute “computer science” and 3 rows containing attribute “math”)

Let  $\text{Entropy}(A, \mathcal{D})$  be the entropy of given attribute  $A$  inside training corpus  $\mathcal{D}$ , formula [III.3.19](#) (Han & Kamber, 2006, p. 298) is used to calculate  $\text{Entropy}(\mathcal{D})$  with assumption that attribute  $A$  has  $v$  values  $a_1, a_2, \dots, a_v$ . For example, given training corpus shown in table [III.3.19](#), all attributes “computer”, “programming language”, “algorithm”, and “derivative” has three nominal values *low*, *medium*, *high*.

$$\text{Entropy}(A, \mathcal{D}) = \sum_{j=1}^v \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \text{Entropy}(\mathcal{D}_j)$$

*Formula III.3.19.* Entropy of given attribute inside training corpus

Where  $\mathcal{D}_j$  is a partition contacting value  $a_j$  of attribute  $A$  and we have  $\mathcal{D}_j \subseteq \mathcal{D}$ . Note that  $|\mathcal{D}_j|$  and  $|\mathcal{D}|$  denote the number of rows of  $\mathcal{D}_j$  and  $\mathcal{D}$ , respectively.

As aforementioned, the decision tree is split at the attribute (the node) whose information gain is the largest. The information gain of attribute  $A$  given training corpus  $\mathcal{D}$ , which is denoted  $\text{Gain}(A, \mathcal{D})$ , is determined by formula [III.3.20](#) as follows (Han & Kamber, 2006, p. 298):

$$\text{Gain}(A, \mathcal{D}) = \text{Entropy}(\mathcal{D}) - \text{Entropy}(A, \mathcal{D})$$

*Formula III.3.20.* Information gain of given attribute inside training corpus

Given training corpus shown in table [III.3.19](#), information gains of attributes  $A :=$ “computer”,  $A :=$ “programming language”,  $A :=$ “algorithm”, and  $A :=$ “derivative” are calculated according to formula [III.3.20](#), in turn. We have:

**$A :=$ “computer”**

$$\text{Entropy}(\mathcal{D}) = 1$$

$\mathcal{D}_1$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc2.txt</i>	low	low	medium	high	math
<i>doc5.txt</i>	low	low	medium	medium	math

### III.3. Learning history sub-model

---

$Entropy(\mathcal{D}_1) = -\frac{2}{2} \log_2 \left(\frac{2}{2}\right) = 0$					
$\mathcal{D}_2$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc3.txt</i>	medium	low	medium	high	math
<i>doc4.txt</i>	medium	high	low	medium	computer science
<i>doc6.txt</i>	medium	low	medium	low	computer science
$Entropy(\mathcal{D}_2) = -\left(\frac{2}{3} \log_2 \left(\frac{2}{3}\right) + \frac{1}{3} \log_2 \left(\frac{1}{3}\right)\right) = \log_2 3 - \frac{2}{3}$					
$\mathcal{D}_3$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc1.txt</i>	high	medium	low	low	computer science
$Entropy(\mathcal{D}_3) = -\frac{1}{1} \log_2 \left(\frac{1}{1}\right) = 0$					

$$\begin{aligned}
 Entropy(A, \mathcal{D}) &= Entropy("computer", \mathcal{D}) = \sum_{j=1}^3 \frac{|\mathcal{D}_j|}{|\mathcal{D}|} Entropy(\mathcal{D}_j) \\
 &= \frac{2}{6} * 0 + \frac{3}{6} * \left(\log_2 3 - \frac{2}{3}\right) + \frac{1}{6} * 0 = \frac{1}{2} \log_2 3 - \frac{1}{3}
 \end{aligned}$$

$$\begin{aligned}
 Gain(A, \mathcal{D}) &= Gain("computer", \mathcal{D}) \\
 &= Entropy(\mathcal{D}) - Entropy("computer", \mathcal{D}) = 1 - \frac{1}{2} \log_2 3 + \frac{1}{3} \\
 &\approx 0.54
 \end{aligned}$$

A:=“*programming language*”

$$Entropy(\mathcal{D}) = -1$$

$\mathcal{D}_1$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc2.txt</i>	low	low	medium	high	math
<i>doc3.txt</i>	medium	low	medium	high	math
<i>doc5.txt</i>	low	low	medium	medium	math
<i>doc6.txt</i>	medium	low	medium	low	computer science
$Entropy(\mathcal{D}_1) = -\left(\frac{1}{4} \log_2 \left(\frac{1}{4}\right) + \frac{3}{4} \log_2 \left(\frac{3}{4}\right)\right) = -\frac{3}{4} \log_2 3 + 2$					
$\mathcal{D}_2$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc1.txt</i>	high	medium	low	low	computer science
$Entropy(\mathcal{D}_2) = -\frac{1}{1} \log_2 \left(\frac{1}{1}\right) = 0$					
$\mathcal{D}_3$	<i>computer</i>	<i>programming</i>	<i>algorithm</i>	<i>derivative</i>	class

		<i>language</i>			
<i>doc4.txt</i>	medium	high	low	medium	computer science
$Entropy(\mathcal{D}_3) = -\frac{1}{1} \log_2 \left( \frac{1}{1} \right) = 0$					

$$\begin{aligned}
 Entropy(A, \mathcal{D}) &= Entropy("programming language", \mathcal{D}) \\
 &= \sum_{j=1}^3 \frac{|\mathcal{D}_j|}{|\mathcal{D}|} Entropy(\mathcal{D}_j) = \frac{4}{6} \left( -\frac{3}{4} \log_2 3 + 2 \right) + \frac{1}{6} * 0 + \frac{1}{6} * 0 \\
 &= -\frac{1}{2} \log_2 3 + \frac{4}{3}
 \end{aligned}$$

$$\begin{aligned}
 Gain(A, \mathcal{D}) &= Gain("programming language", \mathcal{D}) \\
 &= Entropy(\mathcal{D}) - Entropy("programming language", \mathcal{D}) \\
 &= 1 + \frac{1}{2} \log_2 3 - \frac{4}{3} \approx 0.46
 \end{aligned}$$

A:=“algorithm”

$$Entropy(\mathcal{D}) = -1$$

$\mathcal{D}_1$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc1.txt</i>	high	medium	low	low	computer science
<i>doc4.txt</i>	medium	high	low	medium	computer science
$Entropy(\mathcal{D}_1) = -\frac{2}{2} \log_2 \left( \frac{2}{2} \right) = 0$					
$\mathcal{D}_2$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc2.txt</i>	low	low	medium	high	math
<i>doc3.txt</i>	medium	low	medium	high	math
<i>doc5.txt</i>	low	low	medium	medium	math
<i>doc6.txt</i>	medium	low	medium	low	computer science
$Entropy(\mathcal{D}_2) = -\left(\frac{1}{4} \log_2 \left(\frac{1}{4}\right) + \frac{3}{4} \log_2 \left(\frac{3}{4}\right)\right) = -\frac{3}{4} \log_2 3 + 2$					

$$\begin{aligned}
 Entropy(A, \mathcal{D}) &= Entropy("algorithm", \mathcal{D}) = \sum_{j=1}^2 \frac{|\mathcal{D}_j|}{|\mathcal{D}|} Entropy(\mathcal{D}_j) \\
 &= \frac{2}{6} * 0 + \frac{4}{6} \left( -\frac{3}{4} \log_2 3 + 2 \right) = -\frac{1}{2} \log_2 3 + \frac{4}{3}
 \end{aligned}$$

$$\begin{aligned}
 Gain(A, \mathcal{D}) &= Gain("algorithm", \mathcal{D}) \\
 &= Entropy(\mathcal{D}) - Entropy("algorithm", \mathcal{D}) \\
 &= 1 + \frac{1}{2} \log_2 3 - \frac{4}{3} \approx 0.46
 \end{aligned}$$

A:=“derivative”

### III.3. Learning history sub-model

---

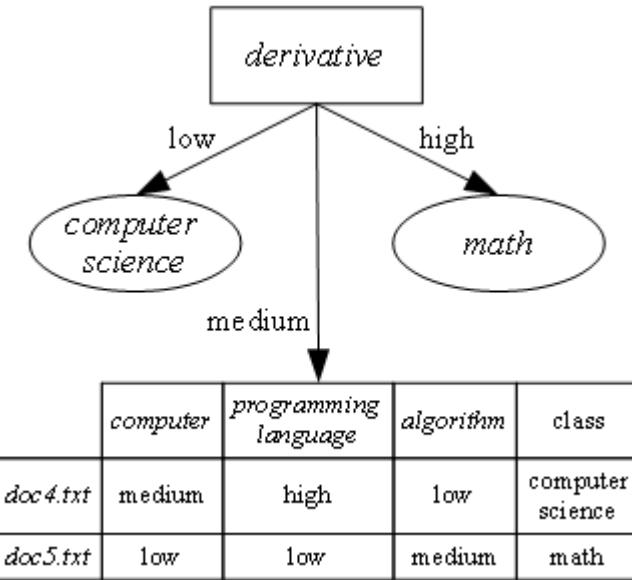
$$Entropy(\mathcal{D}) = -1$$

$\mathcal{D}_1$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc1.txt</i>	high	medium	low	low	computer science
<i>doc6.txt</i>	medium	low	medium	low	computer science
$Entropy(\mathcal{D}_1) = -\frac{2}{2} \log_2 \left(\frac{2}{2}\right) = 0$					
$\mathcal{D}_2$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc4.txt</i>	medium	high	low	medium	computer science
<i>doc5.txt</i>	low	low	medium	medium	math
$Entropy(\mathcal{D}_2) = -\left(\frac{1}{2} \log_2 \left(\frac{1}{2}\right) + \frac{1}{2} \log_2 \left(\frac{1}{2}\right)\right) = 1$					
$\mathcal{D}_3$	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<i>doc2.txt</i>	low	low	medium	high	math
<i>doc3.txt</i>	medium	low	medium	high	math
$Entropy(\mathcal{D}_3) = -\frac{2}{2} \log_2 \left(\frac{2}{2}\right) = 0$					

$$\begin{aligned} Entropy(A, \mathcal{D}) &= Entropy("derivative", \mathcal{D}) = \sum_{j=1}^3 \frac{|\mathcal{D}_j|}{|\mathcal{D}|} Entropy(\mathcal{D}_j) \\ &= \frac{2}{6} * 0 + \frac{2}{6} * 1 + \frac{2}{6} * 0 = \frac{1}{3} \end{aligned}$$

$$\begin{aligned} Gain(A, \mathcal{D}) &= Gain("derivative", \mathcal{D}) \\ &= Entropy(\mathcal{D}) - Entropy("derivative", \mathcal{D}) = 1 - \frac{1}{3} \approx 0.67 \end{aligned}$$

The attribute (term) “*derivative*” is selected to split the decision tree because its information gain,  $Gain("derivative", \mathcal{D}) = 0.67$ , is the largest. Figure III.3.7 depicts the decision constructed from nominal term frequencies shown in table III.3.19 at the first splitting.



**Figure III.3.7.** Decision tree constructed from nominal term frequencies at the first splitting

As seen in figure III.3.7, the training corpus is reduced with regard to “*derivative*=*medium*” and it contains only two document vectors such as *doc4.txt* and *doc5.txt*. For convenience, reduced training corpus  $\mathcal{D}'$  is shown again in table III.3.20.

	computer	programming language	algorithm	class
<i>doc4.txt</i>	medium	high	low	computer science
<i>doc5.txt</i>	low	low	medium	math

**Table III.3.20.** Reduced training corpus

Entropy of reduced training corpus  $\mathcal{D}'$  is:

$$\text{Entropy}(\mathcal{D}') = -\left(\frac{1}{2} \log_2 \left(\frac{1}{2}\right) + \frac{1}{2} \log_2 \left(\frac{1}{2}\right)\right) = 1$$

Given corpus in table III.3.20, the second splitting based on information gains of attributes “*computer*”, “*programming language*”, “*algorithm*” refines the decision tree. These information gains are described shortly in following table III.3.21:

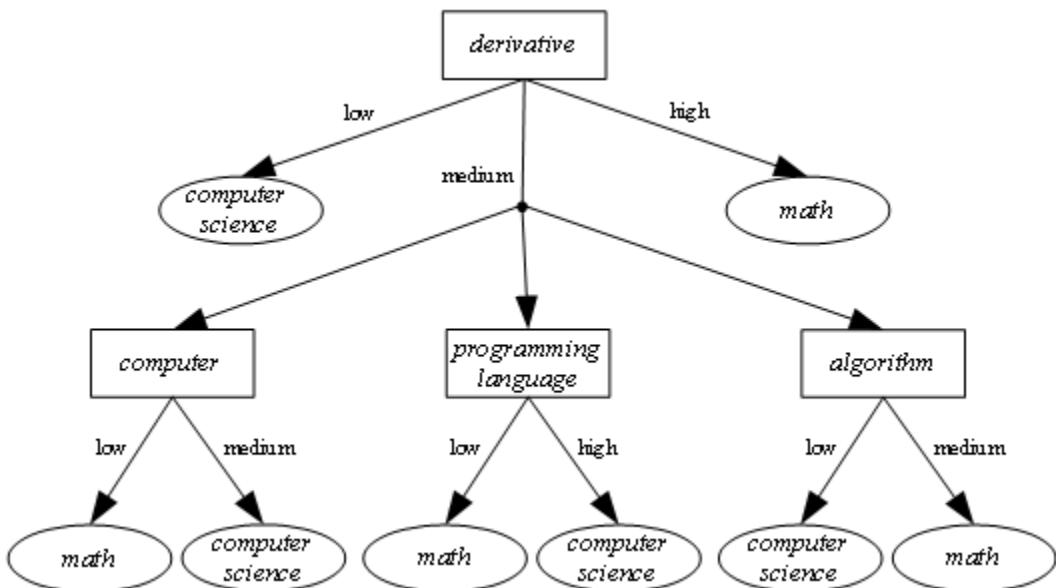
A	Partitions					Entropy(A, $\mathcal{D}'$ )	Gain(A, $\mathcal{D}'$ )
“ <i>computer</i> ”	$\mathcal{D}'_1$	computer	programming language	algorithm	class	$\text{Entropy}(\mathcal{D}'_1) = -\frac{1}{2} \log_2 \left(\frac{1}{2}\right) = 0$	$\text{Gain}(A, \mathcal{D}') = 1 - 0 = 1$
	<i>doc4.txt</i>	medium	high	low	computer science		
	$\mathcal{D}'_2$	computer	programming language	algorithm	class		
	<i>doc5.txt</i>	low	low	medium	math	$\text{Entropy}(\mathcal{D}'_2) = -\frac{1}{2} \log_2 \left(\frac{1}{2}\right) = 0$	$\text{Gain}(A, \mathcal{D}') = 1 - 0 = 1$

### III.3. Learning history sub-model

“programming language”	$\mathcal{D}'_1$	computer	programming language	algorithm	class
	doc4.txt	medium	high	low	computer science
	$Entropy(\mathcal{D}'_1) = -\frac{1}{2} \log_2 \left( \frac{1}{1} \right) = 0$				
	$\mathcal{D}'_2$	computer	programming language	algorithm	class
	doc5.txt	low	low	medium	math
	$Entropy(\mathcal{D}'_2) = -\frac{1}{2} \log_2 \left( \frac{1}{1} \right) = 0$				
					$Entropy(A, \mathcal{D}') = \frac{1}{2} * 0 + \frac{1}{2} * 0 = 0$
					$Gain(A, \mathcal{D}') = 1 - 0 = 1$
“algorithm”	$\mathcal{D}'_1$	computer	programming language	algorithm	class
	doc4.txt	medium	high	low	computer science
	$Entropy(\mathcal{D}'_1) = -\frac{1}{2} \log_2 \left( \frac{1}{1} \right) = 0$				
	$\mathcal{D}'_2$	computer	programming language	algorithm	class
	doc5.txt	low	low	medium	math
	$Entropy(\mathcal{D}'_2) = -\frac{1}{2} \log_2 \left( \frac{1}{1} \right) = 0$				
					$Entropy(A, \mathcal{D}') = \frac{1}{2} * 0 + \frac{1}{2} * 0 = 0$
					$Gain(A, \mathcal{D}') = 1 - 0 = 1$

**Table III.3.21.** Information gains at the second splitting

Because attributes “computer”, “programming language”, and “algorithm” have the same information gain  $Gain(A, \mathcal{D}')=1$ , the decision tree is refined by the second splitting at attributes “computer”, “programming language”, and “algorithm”. Following figure III.3.8 shows the final decision tree generated from our training corpus in which every document is represented by a vector of nominal term frequencies aforementioned in table III.3.19. Note that leaf nodes representing document classes are drawn as ellipses and non-leaf nodes representing attributes (terms) are drawn as rectangles.



**Figure III.3.8.** Final decision tree constructed from nominal term frequencies

The decision tree as shown in figure III.3.8 is the classifier of decision tree method. It is easy to extract classification rules from this decision tree shown in figure III.3.8. Table III.3.22 expresses these classification rules.

Rule	Description
$R_1$	If frequency of term “ <i>derivative</i> ” is <i>low</i> then document belongs to class “ <i>computer science</i> ”.
$R_2$	If frequency of term “ <i>derivative</i> ” is <i>medium</i> and frequency of term “ <i>computer</i> ” is <i>medium</i> then document belongs to class “ <i>computer science</i> ”.
$R_3$	If frequency of term “ <i>derivative</i> ” is <i>medium</i> and frequency of term “ <i>computer</i> ” is <i>low</i> then document belongs to class “ <i>math</i> ”.
$R_4$	If frequency of term “ <i>derivative</i> ” is <i>medium</i> and frequency of term “ <i>programming language</i> ” is <i>high</i> then document belongs to class “ <i>computer science</i> ”.
$R_5$	If frequency of term “ <i>derivative</i> ” is <i>medium</i> and frequency of term “ <i>programming language</i> ” is <i>low</i> then document belongs to class “ <i>math</i> ”.
$R_6$	If frequency of term “ <i>derivative</i> ” is <i>medium</i> and frequency of term “ <i>algorithm</i> ” is <i>low</i> then document belongs to class “ <i>computer science</i> ”.
$R_7$	If frequency of term “ <i>derivative</i> ” is <i>medium</i> and frequency of term “ <i>algorithm</i> ” is <i>medium</i> then document belongs to class “ <i>math</i> ”.
$R_8$	If frequency of term “ <i>derivative</i> ” is <i>high</i> then document belongs to class “ <i>math</i> ”.

**Table III.3.22.** Classification rules derived from decision tree induction

Suppose the numbers of times that terms “*computer*”, “*programming language*”, “*algorithm*” and “*derivative*” occur in document  $D$  are 40, 30, 10, and 20, respectively. We need to determine which class document  $D$  is belongs to.  $D$  is normalized as term frequency vector.

$$D = (0.4, 0.3, 0.1, 0.2)$$

Changing real number into nominal value, we have:

$$D = (\text{medium}, \text{medium}, \text{low}, \text{medium})$$

According to the rule  $R_2$  in above table III.3.22,  $D$  is *computer science* document because in document vector  $D$ , frequency of term “*derivative*” is *medium* and frequency of term “*computer*” is *medium*. In other words, document  $D$  belongs to class “*computer science*”.

Now the decision tree induction is described comprehensively with full of example. The next part III.3.2.2.3 mentions another document classification method based on neural network.

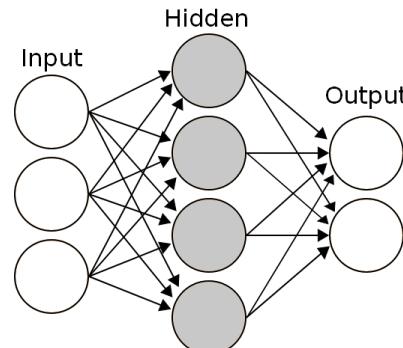
### III.3.2.2.3. Document classification based on neural network

Artificial **neural network** (ANN) is the mathematical model based on biological neural network but neural network (NN) in the research always indicates artificial neural network. It consists of a set of processing units which communicate together by sending signals to each other over a large number of

weighted connections (Kröse & Smagt, 1996, p. 15). Such processing units are also called neurons, cells, nodes, or variables. Each unit is responsible for receiving input from neighbors or external sources and using this input to compute an output signal which is propagated to other units (Kröse & Smagt, 1996, p. 15). However each unit also adjusts the weights of connections. The unit in neural network shares the same meaning with the node in Bayesian network (see sub-section III.1.1.1). There is a common scientific problem that many different terminologies have the similar concept, which leads to a little bit confusion but it is very useful for us to compare and connect them together in order to understand them thoroughly. There are three types of units (Kröse & Smagt, 1996, pp. 15-16):

- *Input units* receive data from outside the network. These units structure the *input layer*.
- *Hidden units* own input and output signals that remain within the neural network. These units structure the hidden layer. There can be one or more *hidden layers*.
- *Output units* send data out of the network. These units structure the *output layer*.

Units in neural network are considered variables. Figure III.3.9 (Wikipedia, Artificial neural network, 2009) shows the simplest structure of an artificial neural network with three layers such as input layer, hidden layer, and output layer. The structure of neural network is often called the topology.

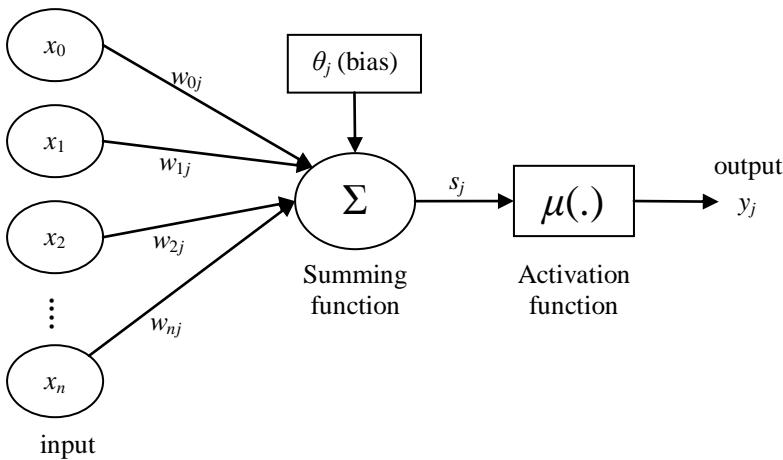


**Figure III.3.9.** Simplest topology of neural network with three layers such as input layer, hidden layer, and output layer

Each connection between unit  $i$  and unit  $j$  is defined by the weight  $w_{ij}$  determining the effect which the signal of unit  $i$  on unit  $j$ . Connection in neural network and arc in Bayesian network (see sub-section III.1.1.1) share the same meaning. Suppose input unit, hidden unit and output unit is denoted as  $x$ ,  $h$ ,  $y$  respectively. In the topology, unit  $y$  is the composition of other units  $h$  which in turn are the compositions of others units  $x$ . The composition (aggregation) of a unit is represented as a weighted sum which will be evaluated to determine the output of this unit. If such unit is the output unit, its output is the output of neural network. The process of computing the output of a unit includes two following steps (Han & Kamber, 2006, p. 331):

- An adder so-called *summing function* sums up all the inputs multiplied by their respective weights. It is essential to compute the weighted sum. This activity is referred to as linear combination.
- An *activation function* controls the amplitude of the output of the neuron. This activity aims to determine the output. Note that the output of the previous units is the input of current unit.

Figure III.3.10 (Han & Kamber, 2006, p. 331) describes the process of computing the output.



**Figure III.3.10.** Process of computing output of a unit

For example as seen in figure III.3.10, given  $n$  previous unit  $x_i$  (s) and a current unit  $y_j$ , let  $O_i$ ,  $I_j$  and  $O_j$  be the output of  $x_i$ , input of  $y_j$ , and output of  $y_j$ . The input and output are evaluated as numeric values. According to the process of computing the output of a unit, we have formula III.3.21 (Han & Kamber, 2006, p. 331) for computing the output value of a unit.

$$I_j = \sum_{i=1}^n w_{ij} O_i + \theta_j$$

$$O_j = \mu(s_j)$$

*Formula III.3.21.* Formula for computing the output of a unit

Where  $w_{ij}$  is the weight of the connection from unit  $x_i$  to unit  $y_j$  and  $\theta_j$  is the bias of unit  $y_j$ .

Note that values of units are arbitrary but they should range from 0 to 1 (sometimes -1 to 1 range). In general, every unit has following aspects:

- A set of inputs connects to it. Each connection is defined by a weight.
- Its weighted sum is computed by summing up all the inputs modified by their respective weights.
- A bias value is added to weighted sum.
- Its output is the outcome of activation function on weighted sum. Activation function is crucial factor in neural network.

### III.3. Learning history sub-model

---

Activation function  $\mu(x)$  is the squashing function which “squashes” a large weighted sum into possible smaller values ranging from 0 to 1 (sometimes -1 to 1 range). There are three types of activation function (Rios):

- *Threshold function* takes on value 0 if weighted sum is less than 0 and otherwise. The formula of threshold function is  $\mu(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$ .
- *Piecewise-linear function* takes on values according to amplification factor in a certain region of linear operation. The formula of piecewise-linear function is  $\mu(x) = \begin{cases} 0 & \text{if } x \leq -\frac{1}{2} \\ x & \text{if } -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 1 & \text{if } \frac{1}{2} \leq x \end{cases}$ .
- *Sigmoid function* or logistic function takes on values in range [0, 1] or [-1, 1]. The formula of sigmoid function is  $\mu(x) = \frac{1}{1+e^{-x}}$  where  $e^{(.)}$  or  $\exp(.)$  denotes exponent function.

There are two topologies (structures) of neural network (Rios):

- *Feed-forward neural network* is directed acyclic graphic in which flow of signal from input units to output units is one-way flow so-called feed-forward. There are no feedback connections
- *Recurrent neural network*: The graph contains cycles; so there are feedback connections in network.

It is necessary to evolve neural network by modifying the weights of connections so that they become more accurate. In other words, such weights should not be fixed by experts. The neural network should be trained by feeding it teaching patterns and letting it change its weights. This is learning process or training process. There are three types of learning methods (Rios):

- *Supervised learning*: The network is trained by providing it with input and matching output patterns (Rios). These patterns are known as classes.
- *Unsupervised learning*: The output is trained to respond to clusters of pattern within the input. There is no a priori set of categories into which the patterns are to be classified (Rios).
- *Reinforcement learning*: The learning machine does some action on the environment and gets a feedback response from the environment (Rios). The learning system grades its action rewarding or punishable based on the environmental response and accordingly adjusts weights. Weight adjustment is continued until there is no change in weights. Reinforcement learning is the intermediate form between supervised learning and unsupervised learning (Rios).

Training corpus is used to train (learn) neural network is called corpus. Recall that the corpus consists of many data row and each row has many attributes. There is a so-called *class attribute* which is used to group (classify) rows. All attributes except class attribute are often represented as input units in neural network and class attribute is often represented as output unit in neural network. If neural network is used to classify document then, rows represent

documents and non-class attributes are terms; in this case, the corpus becomes a matrix  $n \times p$ , which have  $n$  rows and  $p$  columns with respect to  $n$  document vectors and  $p$  terms. Tables III.3.17 and III.3.18 are typical examples of corpus.

We apply neural network into classifying corpus and such supervised learning algorithm used in this chapter is back-propagation algorithm. The back-propagation algorithm (Han & Kamber, 2006, pp. 330-333) is a famous supervised learning algorithm for classification, which is used in feed-forward neural network. It processes iteratively data row in training corpus and compares the network's prediction for each row to the actual class of the row. For each time it feeds a training row, the weights are modified in order to minimize the error between network's prediction and actual class. The modifications are made in backward direction, from output layer through hidden layer down to input layer. Back-propagation algorithm includes four main steps such as initializing the weights, propagating input values forward, propagating errors backward, and updating weights and biases (Han & Kamber, 2006, pp. 330-333). Table III.3.23 describes back-propagation algorithm for learning neural network like programming language.

**1. Initializing the weights:** The weights  $w_{ij}$  of connections between units are initialized as random real number which should be in space  $[0, 1]$ . Each bias  $\theta_i$  associated to each unit is also initialized.

*While terminating condition is not satisfied*

*For each data row in corpus*

**2. Propagating input values forward:** Training data row is fed to input layer.

*For each input unit  $i$ , its input value denoted  $I_i$  and its output value denoted  $O_i$  are the same.*

$$O_i = I_i$$

*End for each input unit  $i$*

*For each hidden unit  $j$  or output unit  $j$ , its input value  $I_j$  is the weighted sum of all output values of units from previous layer. The bias is also added to this weighted sum.*

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

Where  $w_{ij}$  is the weight of connection from unit  $i$  in previous layer to unit  $j$ ,  $O_i$  is the output value of unit  $i$  from previous layer and  $\theta_j$  is the bias of unit  $j$ . The output value of hidden unit or output unit  $O_j$  is computed by applying activation function to its input value (weighted sum). Suppose activation function is sigmoid function. We have:

$$O_j = \frac{1}{1 + e^{-I_j}}$$

Please see formula III.3.21 for more details of computing the output of a unit.

*End for each hidden unit  $j$  or output unit  $j$*

**3. Propagating errors backward:** The error is propagated backward by updating the weights and biases to reflect the error of network's prediction.

For each output unit  $j$ , its error  $Err_j$  is computed according to formula III.3.22 as below:

$$Err_j = O_j(1 - O_j)(V_j - O_j)$$

*Formula III.3.22. Error of output unit*

Where  $V_j$  is the real value of unit  $j$  in training corpus; in other words,  $V_j$  is the actual class.

*End for each output unit j*

For each hidden unit  $j$  from the last hidden layer to the first hidden layer, the weighted sum of the errors of other units connected to it in the next higher layer is considered when its error is computed. So the error of hidden unit  $j$  is computed according to formula III.3.23 as below:

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$

*Formula III.3.23. Error of hidden unit*

Where  $w_{jk}$  is the weight of the connection from hidden unit  $j$  to a unit  $k$  in next higher layer and  $Err_k$  is the error of unit  $k$ .

*End for each hidden unit j*

**4. Updating weights and biases** is based on the errors.

For each weight  $w_{ij}$  over the whole neural network. The weights are updated so as to minimize the errors. Given  $\Delta w_{ij}$  is the change in weight  $w_{ij}$ , the weight  $w_{ij}$  is updated according to formula III.3.24 as below:

$$\Delta w_{ij} = (l) Err_j O_i$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

*Formula III.3.24. Updating connection weight*

Where  $l$  is learning rate ranging from 0 to 1. Learning rate helps to avoid getting stuck at a local minimum in decision space and helps to approach to a global minimum (Han & Kamber, 2006, pp. 332-333).

*End for each weight  $w_{ij}$  in the whole neural network*

For each bias  $\theta_j$  over the whole neural network. The bias  $\theta_j$  of hidden or output unit  $j$  is updated according to formula III.3.25 as below:

$$\Delta \theta_j = (l) Err_j$$

$$\theta_j = \theta_j + \Delta \theta_j$$

*Formula III.3.25. Updating bias*

Where  $l$  is learning rate ranging from 0 to 1.

*End for each bias  $\theta_j$*

*End for each data row in corpus*

*End while terminating condition is not satisfied* with note that there are two common terminating conditions:

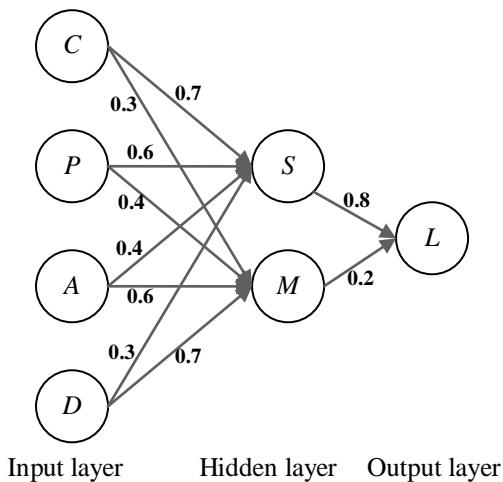
- All  $\Delta w_{ij}$  in some iteration are smaller than given threshold.
- Or, iterating through all possible training data rows.

**Table III.3.23.** Back-propagation algorithm for learning neural network

The trained (learned) neural network derived from back-propagation algorithm is the classifier of neural network. Now the **application of neural network into document classification** is described right here.

Going back the example mentioned in previous part III.3.2.2.2, given a set of classes  $C = \{\text{computer science}, \text{math}\}$ , a set of terms  $T = \{\text{computer}, \text{programming language}, \text{algorithm}, \text{derivative}\}$ . Every document (vector) is represented as a set of input variables. Each term is mapped to an input variable whose value is term frequency ( $tf$ ). So the input layer consists of four input units: “computer”, “programming language”, “algorithm” and “derivative”.

The hidden layer is constituted of two hidden units: “computer science”, “math”. Values of these hidden units range in interval  $[0, 1]$ . The output layer has only one unit named “document class” whose value also ranges in interval  $[0, 1]$  where value 1 denotes that document belongs totally to “computer science” class and value 0 denotes that document belongs totally to “math” class. The evaluation function used in network is sigmoid function. Suppose our original topology is feed-forward neural network in which the weights are initialized arbitrarily and all biases are zero. Note that such feed-forward neural network shown in figure III.3.11 is the one that has no cycle in its model.



**Figure III.3.11.** The neural network for document classification

Note that units  $C$ ,  $P$ ,  $A$  and  $D$  denote terms “computer”, “programming language”, “algorithm” and “derivative”, respectively. Units  $S$  and  $M$  denote “computer science” class and “math” class, respectively. Unit  $L$  denotes “document class”. It is easy to infer that if output value of unit  $L$  is greater than 0.5 then, it is likely that document belongs to “computer science” class.

Going back the example mentioned in previous part III.3.2.2.2, given corpus  $\mathcal{D} = \{\text{doc1.txt}, \text{doc2.txt}, \text{doc3.txt}, \text{doc4.txt}, \text{doc5.txt}, \text{doc6.txt}\}$ . The

training corpus (training data) is shown in following table [III.3.24](#) in which cell  $(i, j)$  indicates the number of times that term  $j$  (column  $j$ ) occurs in document  $i$  (row  $i$ ); in other words, each cell represents a term frequency and each row represents a document vector. Table [III.3.24](#) is the replication of table [III.3.17](#) because it is convenient for readers to keep tract main content.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	<b>class</b>
<i>doc1.txt</i>	5	3	1	1	1
<i>doc2.txt</i>	5	5	40	50	0
<i>doc3.txt</i>	20	5	20	55	0
<i>doc4.txt</i>	20	55	5	20	1
<i>doc5.txt</i>	15	15	40	30	0
<i>doc6.txt</i>	35	10	45	10	1

**Table III.3.24.** Training corpus – Term frequencies of documents (neural network)

Note that the “class” column has binary values where value 1 expresses “*computer science*” class and value 0 expresses “*math*” class.

It is required to normalize term frequencies. Let  $tf_{11}=5$ ,  $tf_{12}=3$ ,  $tf_{13}=1$ , and  $tf_{14}=1$  be the frequencies of terms “*computer*”, “*programming language*”, “*algorithm*”, and “*derivative*”, respectively of document “*doc1.txt*”, for example, these terms are normalized as follows:

$$tf_{11} = \frac{5}{5 + 3 + 1 + 1} = 0.5$$

$$tf_{12} = \frac{3}{5 + 3 + 1 + 1} \approx 0.3$$

$$tf_{13} = \frac{1}{5 + 3 + 1 + 1} = 0.1$$

$$tf_{14} = \frac{1}{5 + 3 + 1 + 1} = 0.1$$

Table [III.3.25](#) shows normalized term frequencies in corpus  $\mathcal{D}$ . Table [III.3.25](#) is the replication of table [III.3.18](#) because it is convenient for readers to keep tract main content.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	<b>class</b>
$D_1$	0.5	0.3	0.1	0.1	1
$D_2$	0.05	0.05	0.4	0.5	0
$D_3$	0.2	0.05	0.2	0.55	0
$D_4$	0.2	0.55	0.05	0.2	1
$D_5$	0.15	0.15	0.4	0.3	0
$D_6$	0.35	0.1	0.45	0.1	1

**Table III.3.25.** Training corpus – Normalized term frequencies (neural network)

Data rows in table III.3.25 representing normalized document vectors are fed to our original neural network in figure III.3.11 for supervised learning. The back-propagation algorithm is used to train network, as described in table III.3.23.

Let  $I_C, I_P, I_A, I_D, I_S, I_M$ , and  $I_L$  be input values of units  $C, P, A, D, S, M$ , and  $L$ . Let  $O_C, O_P, O_A, O_D, O_S, O_M$ , and  $O_L$  be output values of units  $C, P, A, D, S, M$ , and  $L$ . Let  $\theta_S, \theta_M$ , and  $\theta_L$  be bias of units  $S, M$ , and  $L$ . Suppose all biases are initialized by zero, we have  $\theta_S=\theta_M=\theta_L=0$ . Let  $w_{CS}, w_{CM}, w_{PS}, w_{PM}, w_{AS}, w_{AM}, w_{DS}, w_{DM}, w_{SL}$ , and  $w_{ML}$  be weights of connections (arcs) from  $C$  to  $S$ , from  $C$  to  $M$ , from  $P$  to  $S$ , from  $P$  to  $M$ , from  $A$  to  $S$ , from  $A$  to  $M$ , from  $D$  to  $S$ , from  $D$  to  $M$ , from  $S$  to  $L$ , and from  $M$  to  $L$ . According to origin neural network depicted in figure III.3.11, we have  $w_{CS}=0.7, w_{CM}=0.3, w_{PS}=0.6, w_{PM}=0.4, w_{AS}=0.4, w_{AM}=0.6, w_{DS}=0.3, w_{DM}=0.7, w_{SL}=0.8$ , and  $w_{ML}=0.2$ .

From the corpus shown in table III.3.25, the first document  $D_1=(0.5, 0.3, 0.1, 0.1)$  is fed into the back-propagation algorithm. It is required to compute output values  $O_S, O_M, O_L$  and update connection weights. For simplicity, the evaluation function is sigmoid function  $\mu(x) = \frac{1}{1+e^{-x}}$ . According to formula III.3.21 (Han & Kamber, 2006, p. 331) for computing the output value of a unit, we have:

$$O_C=I_C=0.5$$

$$O_P=I_P=0.3$$

$$O_A=I_A=0.1$$

$$O_D=I_D=0.1$$

$$\begin{aligned} I_S &= w_{CS}O_C + w_{PS}O_P + w_{AS}O_A + w_{DS}O_D + \theta_s \\ &= 0.7 * 0.5 + 0.6 * 0.3 + 0.4 * 0.1 + 0.3 * 0.1 + 0 = 0.6 \end{aligned}$$

$$O_S = \mu(I_S) = \frac{1}{1 + \exp(-I_S)} = \frac{1}{1 + \exp(-0.6)} \approx 0.65$$

$$\begin{aligned} I_M &= w_{CM}O_C + w_{PM}O_P + w_{AM}O_A + w_{DM}O_D + \theta_M \\ &= 0.3 * 0.5 + 0.4 * 0.3 + 0.6 * 0.1 + 0.7 * 0.1 + 0 = 0.4 \end{aligned}$$

$$O_M = \mu(I_M) = \frac{1}{1 + \exp(-I_M)} = \frac{1}{1 + \exp(-0.4)} \approx 0.6$$

$$I_L = w_{SL}O_S + w_{ML}O_M + \theta_L = 0.8 * 0.65 + 0.2 * 0.6 + 0 \approx 0.64$$

$$O_L = \frac{1}{1 + \exp(-I_L)} = \frac{1}{1 + \exp(-0.64)} \approx 0.65$$

Let  $V_L$  be the value of output unit  $L$ . Because  $D_1$  belongs to “computer science” class, we have:

$$V_L = 1$$

Let  $Err_L$ ,  $Err_S$ , and  $Err_M$  be the errors of units  $L, S$ , and  $M$ , respectively. According to formula III.3.22 for updating error of output unit, we have:

$$Err_L = O_L(1 - O_L)(V_L - O_L) = 0.65 * (1 - 0.65) * (1 - 0.65) \approx 0.08$$

According to formula III.3.23 for updating error of hidden units, we have:

$$Err_S = O_S(1 - O_S)Err_LW_{SL} = 0.65 * (1 - 0.65) * 0.08 * 0.8 \approx 0.01$$

$$Err_M = O_M(1 - O_M)Err_LW_{ML} = 0.6 * (1 - 0.6) * 0.08 * 0.2 \approx 0$$

According to formula III.3.24 for updating connection weights given learning rate  $l=1$ , we have:

$$w_{CS} = w_{CS} + \nabla w_{CS} = w_{CS} + 1 * Err_S O_C = 0.7 + 1 * 0.01 * 0.5 \approx 0.71$$

### III.3. Learning history sub-model

---

$$\begin{aligned}
w_{CM} &= w_{CM} + \nabla w_{CM} = w_{CM} + 1 * Err_M O_C = 0.3 + 1 * 0 * 0.5 \approx 0.3 \\
w_{PS} &= w_{PS} + \nabla w_{PS} = w_{PS} + 1 * Err_S O_P = 0.6 + 1 * 0.01 * 0.3 \approx 0.6 \\
w_{PM} &= w_{PM} + \nabla w_{PM} = w_{PM} + 1 * Err_M O_P = 0.4 + 1 * 0 * 0.3 \approx 0.4 \\
w_{AS} &= w_{AS} + \nabla w_{AS} = w_{AS} + 1 * Err_S O_A = 0.4 + 1 * 0.01 * 0.1 \approx 0.4 \\
w_{AM} &= w_{AM} + \nabla w_{AM} = w_{AM} + 1 * Err_M O_A = 0.6 + 1 * 0 * 0.1 \approx 0.6 \\
w_{DS} &= w_{DS} + \nabla w_{DS} = w_{DS} + 1 * Err_S O_D = 0.3 + 1 * 0.01 * 0.1 \approx 0.3 \\
w_{DM} &= w_{DM} + \nabla w_{DM} = w_{DM} + 1 * Err_M O_D = 0.7 + 1 * 0 * 0.1 \approx 0.7 \\
w_{SL} &= w_{SL} + \nabla w_{SL} = w_{SL} + 1 * Err_L O_S = 0.8 + 1 * 0.08 * 0.65 \approx 0.85 \\
w_{ML} &= w_{ML} + \nabla w_{ML} = w_{ML} + 1 * Err_L O_M = 0.2 + 1 * 0.08 * 0.6 \approx 0.25
\end{aligned}$$

According to formula III.3.25 for updating biases  $\theta_S$ ,  $\theta_M$ , and  $\theta_L$ , we have:

$$\theta_S = \theta_S + \Delta\theta_S = \theta_S + 1 * Err_S = 0 + 1 * 0.01 = 0.01$$

$$\theta_M = \theta_M + \Delta\theta_M = \theta_M + 1 * Err_M = 0 + 1 * 0 = 0$$

$$\theta_L = \theta_L + \Delta\theta_L = \theta_L + 1 * Err_L = 0 + 1 * 0.08 = 0.08$$

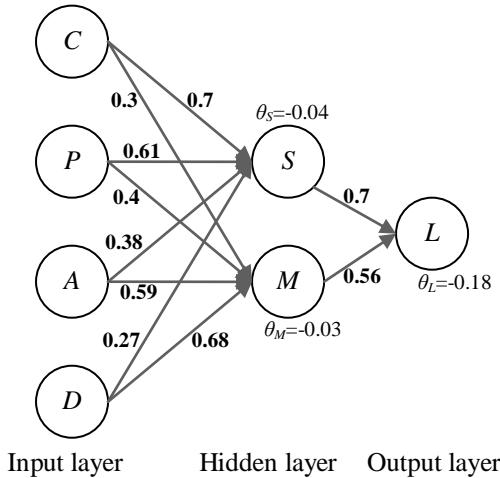
In similar way, remaining documents  $D_2=(0.05, 0.05, 0.4, 0.5)$ ,  $D_3=(0.05, 0.05, 0.4, 0.5)$ ,  $D_4=(0.2, 0.05, 0.2, 0.55)$ ,  $D_5=(0.15, 0.15, 0.4, 0.3)$ , and  $D_6=(0.35, 0.1, 0.45, 0.1)$  are fed into the back-propagation algorithm so as to calculate the final output values  $O_S$ ,  $O_M$ ,  $O_L$  and update final connection weights. Table III.3.26 shows results from this training process based on back-propagation algorithm.

	Inputs	Outputs	Weights	Biases
$D_1$	$I_C=0.5$	$O_S=0.65$	$w_{CS}=0.70$	$\theta_S=0.01$
	$I_P=0.3$	$O_M=0.60$	$w_{CM}=0.30$	$\theta_M=0.00$
	$I_A=0.1$	$O_L=0.65$	$w_{PS}=0.60$	$\theta_L=0.08$
	$I_D=0.1$		$w_{PM}=0.40$	
			$w_{AS}=0.40$	
			$w_{AM}=0.60$	
			$w_{DS}=0.30$	
			$w_{DM}=0.70$	
			$w_{SL}=0.85$	
$D_2$	$I_C=0.05$	$O_S=0.60$	$w_{CS}=0.70$	$\theta_S=-0.02$
	$I_P=0.05$	$O_M=0.65$	$w_{CM}=0.30$	$\theta_M=-0.01$
	$I_A=0.40$	$O_L=0.71$	$w_{PS}=0.60$	$\theta_L=-0.07$
	$I_D=0.50$		$w_{PM}=0.40$	
			$w_{AS}=0.39$	
			$w_{AM}=0.59$	
			$w_{DS}=0.29$	
			$w_{DM}=0.69$	
			$w_{SL}=0.76$	
$D_3$	$I_C=0.05$	$O_S=0.60$	$w_{CS}=0.70$	$\theta_S=-0.04$
	$I_P=0.05$	$O_M=0.64$	$w_{CM}=0.30$	$\theta_M=-0.03$
	$I_A=0.40$	$O_L=0.67$	$w_{PS}=0.60$	$\theta_L=-0.22$
	$I_D=0.50$		$w_{PM}=0.40$	
			$w_{AS}=0.38$	

			$w_{AM}=0.59$ $w_{DS}=0.27$ $w_{DM}=0.68$ $w_{SL}=0.68$ $w_{ML}=0.41$	
$D_4$	$I_C=0.20$ $I_P=0.05$ $I_A=0.20$ $I_D=0.55$	$O_S=0.62$ $O_M=0.60$ $O_L=0.62$	$w_{CS}=0.70$ $w_{CM}=0.30$ $w_{PS}=0.61$ $w_{PM}=0.41$ $w_{AS}=0.38$ $w_{AM}=0.59$ $w_{DS}=0.27$ $w_{DM}=0.68$ $w_{SL}=0.73$ $w_{ML}=0.55$	$\theta_S=-0.03$ $\theta_M=-0.02$ $\theta_L=-0.13$
$D_5$	$I_C=0.15$ $I_P=0.15$ $I_A=0.40$ $I_D=0.30$	$O_S=0.60$ $O_M=0.63$ $O_L=0.65$	$w_{CS}=0.70$ $w_{CM}=0.30$ $w_{PS}=0.61$ $w_{PM}=0.40$ $w_{AS}=0.37$ $w_{AM}=0.58$ $w_{DS}=0.27$ $w_{DM}=0.68$ $w_{SL}=0.64$ $w_{ML}=0.41$	$\theta_S=-0.05$ $\theta_M=-0.04$ $\theta_L=-0.28$
$D_6$	$I_C=0.35$ $I_P=0.10$ $I_A=0.45$ $I_D=0.10$	$O_S=0.61$ $O_M=0.61$ $O_L=0.60$	$w_{CS}=0.70$ $w_{CM}=0.30$ $w_{PS}=0.61$ $w_{PM}=0.40$ $w_{AS}=0.38$ $w_{AM}=0.59$ $w_{DS}=0.27$ $w_{DM}=0.68$ $w_{SL}=0.70$ $w_{ML}=0.56$	$\theta_S=-0.04$ $\theta_M=-0.03$ $\theta_L=-0.18$

**Table III.3.26.** Results from training process based on back-propagation algorithm

According to the training results shown in table III.3.26, the weights and biases of origin neural network are changed. It means that neural network is already trained. Thus, figure III.3.12 expresses the neural network learned by back-propagation algorithm.



**Figure III.3.12.** Trained neural network

The trained neural network depicted in figure III.3.12 is the typical classifier of classification method based on neural work.

Suppose the numbers of times that terms “*computer*”, “*programming language*”, “*algorithm*” and “*derivative*” occur in document *D* are 40, 30, 10, and 20, respectively. We need to determine which class document *D* belongs to. *D* is normalized as term frequency vector.

$$D = (0.4, 0.3, 0.1, 0.2)$$

Recall that the trained neural network depicted in figure III.3.12 has connection weights  $w_{CS}=0.7$ ,  $w_{CM}=0.3$ ,  $w_{PM}=0.61$ ,  $w_{PM}=0.4$ ,  $w_{AS}=0.38$ ,  $w_{AM}=0.59$ ,  $w_{DS}=0.27$ ,  $w_{DM}=0.68$ ,  $w_{SL}=0.7$ ,  $w_{ML}=0.56$  and biases  $\theta_S=-0.04$ ,  $\theta_M=-0.03$ ,  $\theta_L=-0.18$ . It is required to compute output values  $O_S$ ,  $O_M$ , and  $O_L$ . For simplicity, the evaluation function is sigmoid function  $\mu(x) = \frac{1}{1+e^{-x}}$ . According to formula III.3.21 (Han & Kamber, 2006, p. 331) for computing the output value of a unit, we have:

$$\begin{aligned} I_S &= w_{CS}O_C + w_{PS}O_P + w_{AS}O_A + w_{DS}O_D + \theta_S \\ &= 0.7 * 0.4 + 0.61 * 0.3 + 0.38 * 0.1 + 0.27 * 0.2 - 0.04 \\ &\approx 0.52 \end{aligned}$$

$$O_S = \mu(I_S) = \frac{1}{1 + exp(-I_S)} = \frac{1}{1 + exp(-0.52)} \approx 0.63$$

$$\begin{aligned} I_M &= w_{CM}O_C + w_{PM}O_P + w_{AM}O_A + w_{DM}O_D + \theta_M \\ &= 0.3 * 0.4 + 0.4 * 0.3 + 0.59 * 0.1 + 0.68 * 0.2 - 0.03 \approx 0.41 \end{aligned}$$

$$O_M = \mu(I_M) = \frac{1}{1 + exp(-I_M)} = \frac{1}{1 + exp(-0.41)} \approx 0.6$$

$$I_L = w_{SL}O_S + w_{ML}O_M + \theta_L = 0.7 * 0.63 + 0.56 * 0.6 - 0.18 \approx 0.6$$

$$O_L = \frac{1}{1 + exp(-I_L)} = \frac{1}{1 + exp(-0.6)} \approx 0.65$$

Because  $O_L$  is greater than 0.5, it is more likely that document  $D = (0.4, 0.3, 0.1, 0.2)$  belongs to class “*computer science*”.

This part ends up the introduction to classification methods and their application into document classification. The main approach to discover user interest will be described in next sub-section III.3.2.3. Recall that discovering

user interest is the second extended function of learning history sub-model. Of course, these classification methods (support vector machine, decision tree, neural network) are implemented inside mining engine (ME) because ME manage learning history sub-model. Please see section [II.2](#) for more details about ME and the user modeling system Zebra.

### **III.3.2.3. Discovering user interests based on document classification**

As aforementioned at the beginning of the section [III.3](#), discovering user interest is the second extended function of learning history sub-model. Previous sub-sections only introduce how to represent documents as numeric vectors ([III.3.2.1](#)) and supervised learning classification methods such as support vector machine, decision tree, and neural network ([III.3.2.2](#)). So this sub-section is the main one describing main tasks to discover user interests based on document classification, according to steps 3 and 4 of proposed approach for discovering user interest as described in table [III.3.10](#). Recall that previous sub-sections [III.3.2.1](#) and [III.3.2.2](#) mention step 1 and step 2 of such proposed approach.

Suppose in some library or website, user  $U$  does her/his search for her/his interesting books, documents, etc. There is demand of discovering her/his interests so that such library or website can provide adaptive documents to her/him whenever she/he visits in the next time. This is adaptation process in which system tailors documents to each individual. Given there is a set of key words or terms  $\{computer, programming\ language, algorithm, derivative\}$  that user  $U$  often looking for, her/his searching history is shown in following table [III.3.27](#):

Date	Keywords (terms) searched
Aug 28 10:20:01	<i>computer, programming language, algorithm, derivative</i>
Aug 28 13:00:00	<i>computer, programming language, derivative</i>
Aug 29 8:15:01	<i>computer, programming language</i>
Aug 30 8:15:06	<i>computer</i>

**Table III.3.27.** User's searching history

The searching history is stored in learning history sub-model; please see section [III.3](#) for more details about learning history sub-model.

This searching history is considered as training dataset for mining maximum frequent itemsets. The keywords (terms) are now considered items. A itemset is constituted of some items. The support of itemset  $x$  is defined as the fraction of total transaction which containing  $x$ . Given support threshold  $min\_sup$ , the itemset  $x$  is called *frequent itemset* if its support satisfies the support threshold ( $\geq min\_sup$ ). Moreover  $x$  is *maximum frequent itemset* if  $x$  is frequent itemset and all super-itemsets of  $x$  are not frequent. Note that  $y$  is super-itemset of  $x$  if  $x \subset y$ . The itemset that has  $k$  items is called *k-itemset*. Table [III.3.28](#) shows the supports of 1-itemsets.

1-itemset	support
<i>computer</i>	4
<i>programming language</i>	3
<i>algorithm</i>	1
<i>derivative</i>	2

**Table III.3.28.** 1-itemsets

Applying mining algorithms such as Apriori (Han & Kamber, 2006, pp. 234-239) and FP-growth (Han & Kamber, 2006, pp. 242-245), it is easy to find maximum frequent itemsets given  $min\_sup = 1$ . The maximum frequent itemset that user searches are shown in below table [III.3.29](#):

N <sub>o</sub>	itemset
1	<i>computer, programming language, algorithm, derivative</i>

**Table III.3.29.** Maximum frequent itemset that user searches

I propose the new point of view: “*The maximum frequent itemsets are considered as documents and the classes of such documents are considered as user interests*”. Such documents may be called interesting documents. Which classes such interesting documents belong to are user interests. It means that discovering user’s interests involves in classifying interesting documents. Suppose we have a set of classes  $C = \{computer science, math\}$ , a set of terms  $T = \{computer, programming language, algorithm, derivative\}$  and the set of classification rules in table [III.3.22](#). Each maximum frequent itemset that user searches is modeled as a document vector (so-called interesting document vector or user interest vector) whose elements are the support of its member items. The interesting document vector derived from maximum frequent itemset is shown in table [III.3.30](#).

N <sub>o</sub>	vector
1	( <i>computer</i> =4, <i>programming language</i> =3, <i>algorithm</i> =1, <i>derivative</i> =2)

**Table III.3.30.** Interesting document vector

The interesting document vector is normalized as in table [III.3.31](#). It is easy to recognize that supports inside interesting document vector are normalized.

N <sub>o</sub>	vector
1	( <i>computer</i> =0.4, <i>programming language</i> =0.3, <i>algorithm</i> =0.1, <i>derivative</i> =0.2)

**Table III.3.31.** Interesting document vector is normalized

Because we intend to use decision tree to classify the document. It is necessary to convert normalized numeric supports to nominal supports as following specifications:

$$0 \leq \text{support} < 0.2 \rightarrow \text{low}$$

$$\begin{aligned} 0.2 \leq \text{support} < 0.5 &\rightarrow \text{medium} \\ 0.5 \leq \text{support} &\rightarrow \text{high} \end{aligned}$$

Normalized document vector is converted into nominal interesting document vector as in table [III.3.32](#).

N <sub>o</sub>	vector
1	(computer=medium, programming language=medium, algorithm=low, derivative=medium)

**Table III.3.32.** Nominal interesting document vector

It is possible to use SVM, decision tree, or neural network to classify documents. Hence we use decision tree as sample classifier for convenience because we intend to re-use classification rules shown in table [III.3.22](#). Otherwise we must determine the weight vector  $W^*$  if applying SVM approach. SVM approach is more powerful than decision tree with regard to document classification in case of huge training data.

Applying classification rule 2 in table [III.3.22](#), the interesting document belongs to class “computer science” because the frequencies of “derivative” and “computer” are medium and medium, respectively. So we can state that user  $U$  has only one interest: *computer science*.

Note that in case of using neural network for document classification, the normalized interesting document vector is fed into the trained neural network as shown in figure [III.3.12](#). After that the output value of output unit specifies the class of the document. For instance, as seen in table [III.3.25](#), the normalized interesting document vector  $U=(0.4, 0.3, 0.1, 0.2)$  is fed into the trained neural network shown in figure [III.3.12](#). Recall that the output value  $O_L=0.65$  is greater than 0.5, it is more likely that document  $U=(0.4, 0.3, 0.1, 0.2)$  belongs to class “computer science”; please see part [III.3.2.2.3](#) for more details about how to compute the output value  $O_L$ . So the interest of user  $U$  is *computer science*.

In general, four steps of proposed approach to discover user interests based on document classification are described completely. The next sub-section [III.3.2.4](#) is the evaluation.

### III.3.2.4. Evaluation

Recall that my approach includes four following steps:

1. Documents are represented as vectors.
2. Classifying documents by using decision tree, support vector machine (SVM) or neural network.
3. Mining user’s access history to find maximum frequent itemsets. Each itemset is considered an interesting document.
4. Applying classifiers resulted from step 2 into interesting documents in order to find their suitable classes. These classes are user interests.

Two new points of view are inferred from these steps:

### III.3. Learning history sub-model

---

- The series of user access in his/her history are modeled as documents. So user is referred indirectly to as document.
- User interests are classes to which such documents are belong.

The technique of constructing vector model for representing document is not important to this approach. There are some algorithms of text segmentation for specifying all terms in documents. From this, it is easy to build up document vectors by computing term frequency and inverse document frequency. However the concerned techniques of document classification such as SVM, decision tree and neural network influence extremely on this approach. SVM and neural network is more effective than decision tree in case of huge training data set but it is not convenient for applying classifiers (weight vector  $W^*$ ) into determining the classes of documents. Otherwise it is easy to use classification rules taken out from decision tree for this task.

In general, the second extended function of learning history sub-model is described thoroughly in this sub-section III.3.2 – how to discover user interests based on document classification. The next sub-section III.3.3 mentions the last extended function of learning history sub-model – how to construct learner groups.

### III.3.3. Constructing user groups or user communities

As aforementioned at [the beginning of this section III.3](#), there are three extended functions executed in learning history sub-model such as learning concept recommendation, discovering user interests and constructing learner groups. Learning concept recommendation and discovering user interests are mentioned in previous sub-sections III.3.1 and III.3.2. This sub-section III.3.3 focuses on how to construct user groups. Note that users are learners and students in learning context.

Remind that user model is the representation of personal traits or characteristics about user such as demographic information, knowledge, learning style, goal, and interest. Learner model is defined as user model in learning context in which user is learner who profits from adaptive learning system. Note that learner model, student model, and user model are the same terms in learning context. Adaptive systems exploit valuable information in user model so as to provide adaptation effect, i.e., to behave different users in different ways. For example, the adaptive systems tune learning materials to a user in order to provide the best materials to her/him. Please see chapter I for more details about user model and adaptive learning. The usual adaptation effect is to give individually adaptation to each user, but there is a demand to provide adaptation to a group or community of users. Consequently, all users in the same group will profit from the same learning materials, teaching methods, etc. because they have the common characteristics. So there are two kinds of adaptations:

- *Individual adaptation* regards to each user.
- *Community (or group) adaptation* focuses on a community (or group) of users.

Group adaptation has more advantages than individual adaptation in some situations:

- Common features in a group which are the common information of all members in such group are relatively stable, so it is easy for adaptive systems to perform accurately adaptive tasks.
- If a new user logs in system, she/he will be classified into a group and initial information of his model is assigned to common features in such group.
- In the collaborative learning, users need to learn or discuss together. It is very useful if the collaborative learning is restricted in a group of similar users. Therefore, it is convenient for users that have common characteristics (knowledge, goal, interest, etc.) to learn together because they do not come up against an obstacle when interacting together.

The problem that needs to be solved now is how to determine user groups. This relates to clustering techniques so as to cluster user models because a group is considered as a cluster of similar user models. Sub-sections [III.3.3.1](#) and [III.3.3.2](#) discuss about user model clustering techniques, namely  $k$ -mean algorithm for vector model, overlay model, and Bayesian network. In section [III.3.3.2](#), I propose the formulas so as to compute the dissimilarity of two overlay models or two Bayesian network (Nguyen, User Model Clustering, 2014). The  $k$ -medoids algorithm and similarity measures such as cosine similarity measure and correlation coefficient are discussed in section [III.3.3.3](#). Sub-section [III.3.3.4](#) is the conclusion.

In general, these sections focus on how to construct user groups which is a extended function supported by [mining engine](#) (ME) and [learning history sub-model](#), along with other functions such as learning concept recommendation and discovering user interests aforementioned previous sections.

### **III.3.3.1. User model clustering**

Suppose user model  $U_i$  is represented as vector  $U_i = \{u_{i1}, u_{i2}, \dots, u_{ij}, \dots, u_{in}\}$  whose elements are numbers. For instance, if  $U_i$  represents user knowledge then  $U_i$  is considered as a knowledge vector in which the  $j^{th}$  component of this vector is the conditional probability expresing how user master knowledge item  $j^{th}$ . Note that  $U_i$  is called user model or user vector or user model vector.

Suppose there is a collection of users  $\{U_1, U_2, \dots, U_m\}$  and we need to find out  $k$  groups so-called  $k$  *user model clusers*. A user model cluster is a set of similar user models, so user models in the same cluster is dissimilar to ones in other clusters. The dissimilarity of two user models is defined as Euclidean distance between them, as shown in formula [III.3.26](#).

$$\begin{aligned} \text{dissim}(U_1, U_2) &= \text{distance}(U_1, U_2) \\ &= \sqrt{(u_{11} - u_{21})^2 + (u_{12} - u_{22})^2 + \dots + (u_{1n} - u_{2n})^2} \end{aligned}$$

*Formula III.3.26.* Dissimilarity of two user model vectors according to Euclidean distance

The less  $dissim(U_1, U_2)$  is, the more similar  $U_1$  and  $U_2$  are. It is easy to recognize that constructing user groups or user communities is essentially clustering user models. Applying  $k$ -mean algorithm (Han & Kamber, 2006, pp. 402-403), we partition a collection of user models into  $k$  user model clusters. The  $k$ -mean algorithm includes three following steps:

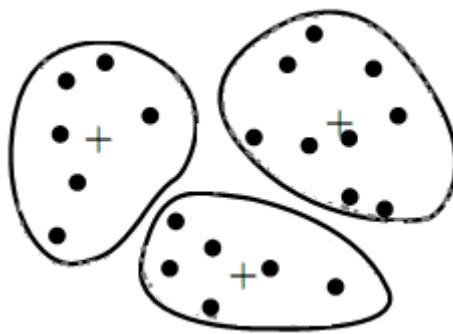
1. It randomly selects  $k$  user models, each of which initially represents a cluster mean. Of course, we have  $k$  cluster means. Each mean is considered as the “representative” of one cluster. There are  $k$  clusters.
2. For each remaining user model, the dissimilarities between it and  $k$  cluster means are computed. Such user model belongs to the cluster which it is most similar to; it means that if user model  $U_i$  belong to cluster  $C_j$ , the dissimilarity measure  $dissim(U_i, C_j)$  is minimal.
3. After that, the means of all clusters are re-computed. If stopping condition is met then algorithm is terminated, otherwise returning step 2.

This process is repeated until the stopping condition is met. For example, the stopping condition is that the error criterion is less than a pre-defined threshold. The square-error criterion is defined by formula III.3.27 as follows:

$$Err = \sum_{i=1}^k \sum_{U \in C_i} dissim(U - M_i)$$

*Formula III.3.27. Error criterion for  $k$ -mean algorithms*

Where  $C_i$  and  $M_i$  is cluster  $i$  and its mean, respectively and  $dissim(U, M_i)$  is the dissimilarity between user model  $U$  and the mean of cluster  $C_i$ . It is easy to recognize that error criterion  $Err$  is the sum of all dissimilarities between user models and means of clusters. Figure III.3.13 (Han & Kamber, 2006, p. 403) is an example of user model clusters resulted from  $k$ -mean algorithm.



**Figure III.3.13.** User model clusters (means are marked by sign “+”)

The mean  $M_i$  of cluster  $C_i$  is the center of such cluster, which is the vector whose  $t^{\text{th}}$  component is the average of  $t^{\text{th}}$  components of all user model vectors in cluster  $C_i$ . Formula III.3.28 formulate the mean  $M_i$  of cluster  $C_i$ .

$$M_i = \left( \frac{1}{n_i} \sum_{j=1}^{n_i} u_{j1}, \frac{1}{n_i} \sum_{j=1}^{n_i} u_{j2}, \dots, \frac{1}{n_i} \sum_{j=1}^{n_i} u_{jn} \right)$$

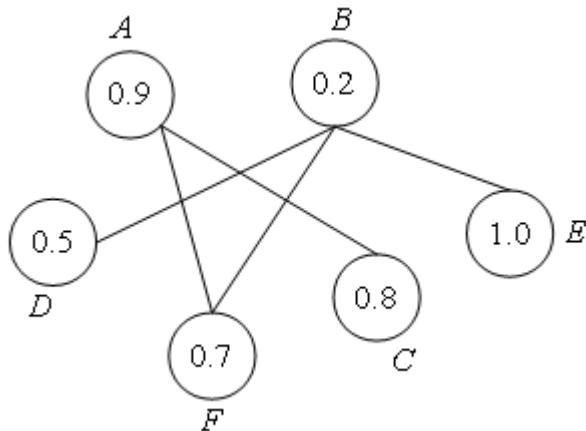
*Formula III.3.28.* The mean of a cluster

Where  $n_i$  is the number of user models in cluster  $C_i$  and  $u_{jk}$  is the  $k^{th}$  component of user model  $U_j \in C_i$ .

Clustering vectors is very easy with  $k$ -mean algorithms but it is a little bit difficult to cluster objects represented by other formats such as overlay model and Bayesian network. The next sub-sections [III.3.3.2](#) mentions solutions of clustering overlay models.

### III.3.3.2. Overlay model clustering

If user is modeled in a vector, the dissimilarity measure in  $k$ -mean algorithm is Euclidean distance. However there is a question: “how to compute such measure in case that user model is an overlay model which is in form of domain graph”. In this situation, the domain is decomposed into a set of elements and the overlay model is simply a set of masteries over those elements. So overlay model is the subset of domain model. Figure [III.3.14](#) which is a replication of figure [I.1.4](#) depicts an example of overlay model; please see sub-section [I.1.2.2](#) for more details about overlay model.



**Figure III.3.14.** Overlay model

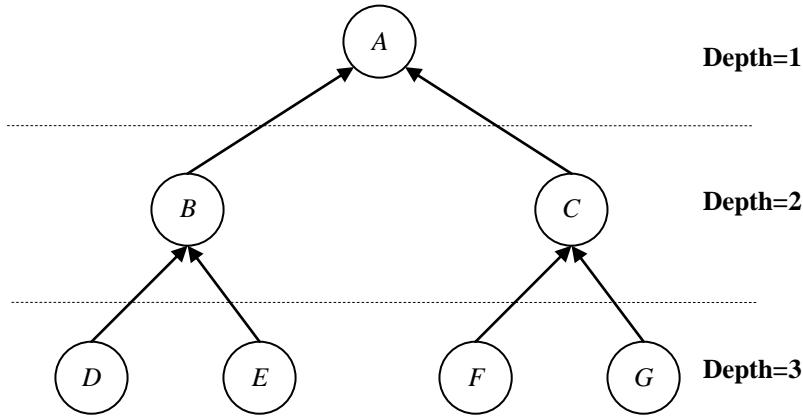
It is essential that overlay model is the graph of domain; so overlay model is also called as graph model. Each node (or vertex) in graph is a knowledge item represented by a number indicating how user masters such knowledge item. Each edge (or arc) reveals the relationship between two nodes. It is clear to say that the dissimilarity measure needs changing so as to compare two overlay models. Note that the terms “user model”, “overlay model”, “graph model” are the same in this context. Suppose two user overlay models  $U_1$  and  $U_2$  are denoted as below:

$$U_1 = G_1 = \langle V_1, E_1 \rangle \text{ and } U_2 = G_2 = \langle V_2, E_2 \rangle$$

Where  $V_i$  and  $E_i$  are set of nodes and set of arcs, respectively.  $V_i$  is also considered as a vector whose elements are numbers representing user's masteries of knowledge items.

$$V_i = (v_{i1}, v_{i2}, \dots, v_{in})$$

Each  $v_{ij}$  also denotes the  $j^{th}$  element of  $V_i$  and so, each  $v_{ij}$  is a variable representing the  $j^{th}$  node of graph  $G_i$  and we can identify variable  $v_{ij}$  with node  $v_{ij}$ . Suppose graph model is in form of tree in which each directed arc represents the relationship of two nodes. There are many relationships such as prerequisite, dependency, parent-child, cause-effect, aggregation, and diagnostic. Each relationship is specified according to application context. For example, if an arc from node  $A$  to node  $B$  represents a aggregation relationship, the mastery of node  $A$  contributes partially and exclusively to the whole mastery of node  $B$ ; please sub-sections III.1.1.2 and III.1.1.3 for more details about aggregation relationship. Figure III.3.15 depicts a graph model in form of tree.



**Figure III.3.15.** Graph model in form of tree and relationships

Let  $\text{depth}(v_{ij})$  is the depth of node  $v_{ij}$  in graph model  $G_i$ . Note that the depth of root node is 1.

$$\text{depth}(v_{root}) = 1$$

For example, given tree graph depicted in figure III.3.15, we have  $\text{depth}(A)=1$  because  $A$  is root.

Given the assumption “*the structure of graphs of all users is kept intact*”, the dissimilarity (or distance) of two graph models  $G_1$  and  $G_2$  is defined by formula III.3.29 as follows:

$$\text{dissim}(G_1, G_2) = \text{distance}(G_1, G_2) = \sum_{j=1}^n \frac{|v_{1j} - v_{2j}|}{\text{depth}(v_{1j})}$$

*Formula III.3.29.* Dissimilarity of two graph models

Note that the notation  $|.|$  denotes absolute value. In general case, the notation  $|v_{1j} - v_{2j}|$  denotes difference between node  $v_{1j}$  and node  $v_{2j}$  when node (variable)  $v_{ij}$  can be represented by any format such as real number, complex number, and structure.

The meaning of this formula III.3.29 is: “*the high level concept (node) is the aggregation of low level (basic) concepts*”. The depths of nodes  $v_{ij}$  in all graph models are the same because the structure of graphs is kept intact.

$$\forall a, b, j, \text{depth}(v_{aj}) = \text{depth}(v_{bj})$$

Where  $a$  and  $b$  are graph indices.

In general, formula [III.3.29](#) for specifying the dissimilarity of two graphs are only applied into the case that graph satisfies the condition that graph must be the multi-depth tree like the one depicted in figure [III.3.15](#).

For example, there are three graph models  $G_1$ ,  $G_2$ , and  $G_3$  whose structures are the same to the structure shown in figure [III.3.15](#) but their node values are different. The values of their nodes are shown in following table [III.3.33](#):

	A	B	C	D	E	F	G
$G_1$	2	1	1	0	3	2	1
$G_2$	1	1	0	1	4	5	4
$G_3$	2	1	1	1	4	5	4

**Table III.3.33.** Values of graph nodes

According to formula [III.3.29](#), the dissimilarities (or distances) between  $G_3$  and  $G_1$ ,  $G_2$ , respectively are computed as below:

$$dissim(G_1, G_3)$$

$$= \frac{|2 - 2|}{1} + \frac{|1 - 1|}{2} + \frac{|1 - 1|}{2} + \frac{|0 - 1|}{3} + \frac{|3 - 4|}{3} + \frac{|2 - 5|}{3} \\ + \frac{|1 - 4|}{3} = 0 + 0 + 0 + \frac{1}{3} + \frac{1}{3} + 1 + 1 \approx 2.66$$

$$dissim(G_2, G_3)$$

$$= \frac{|1 - 2|}{1} + \frac{|1 - 1|}{2} + \frac{|0 - 1|}{2} + \frac{|1 - 1|}{3} + \frac{|4 - 4|}{3} + \frac{|5 - 5|}{3} \\ + \frac{|4 - 4|}{3} = 1 + 0 + 0.5 + 0 + 0 + 0 = 1.5$$

So  $G_2$  is more similar to  $G_3$  than  $G_1$  is because  $dissim(G_2, G_3)$  is smaller than  $dissim(G_1, G_3)$ .

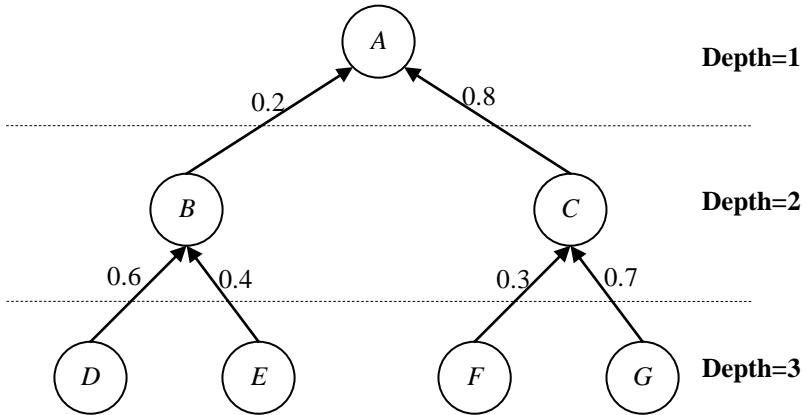
There are two common cases of graph model:

- Graph is weighted graph in which arcs are weighted.
- Graph is evolved as Bayesian network (BN). Please see sub-section [III.1.1.1](#) for more details about BN.

Dissimilarity measure relevant such two cases are mentioned in successive parts [III.3.3.2.1](#) and [III.3.3.2.2](#).

#### *III.3.3.2.1. In case that arcs in graph are weighted*

In case that each arc is assigned a weight representing the strength of relationship between parent node and child node, the figure [III.3.16](#) shows this situation:



**Figure III.3.16.** Graph model and weighted arcs

The structure of graph model in figure III.3.16 is the same to the one in figure III.3.15 except that arcs are weighted. Recall that each node in Bayesian network is a random variable.

The dissimilarity (or distance) of two weighted graph models  $G_1$  and  $G_2$  is re-defined by formula III.3.30 as follows:

$$dissim(G_1, G_2) = distance(G_1, G_2) = \sum_{j=1}^n \frac{|v_{1j} - v_{2j}|}{depth(v_{1j})} weight(v_{1j})$$

*Formula III.3.30.* Dissimilarity of two weighted graph models

Let  $weight(v_{ij})$  be the weight of arc from node  $v_{ij}$  (in graph model  $G_i$ ) to its parent; your attention please, there is an unusual convention in this part III.3.3.2.1 that given a arc from node  $X$  to node  $Y$  then,  $Y$  is the parent node of  $X$ . I consider that  $weight(v_{ij})$  is the weight at node  $v_{ij}$ . The sum of weights at nodes having the same parent equals 1.

$$\sum_{\substack{v_{ij} \\ v_{ij} \text{ has the same parent}}} weight(v_{ij}) = 1$$

The weight at root node equals 1.

$$weight(v_{root}) = 1$$

Recall that the depth of root node is 1.

$$depth(v_{root}) = 1$$

The depths and the weights at the  $j^{th}$  nodes of all graph models are the same because the structure of graphs is kept intact and the weights of arcs are kept intact too.

$$\begin{aligned} \forall a, b, j, & depth(v_{aj}) = depth(v_{bj}) \\ \forall a, b, j, & weight(v_{aj}) = weight(v_{bj}) \end{aligned}$$

Where  $a$  and  $b$  are graph indices.

If weights  $weight(v_{ij})$  (s) are focused and values  $v_{ij}$  (s) are ignored, the formula III.3.30 becomes formula III.3.31:

$$dissim(G_1, G_2) = distance(G_1, G_2) = \sum_{j=1}^n \frac{weight(v_{1j})}{depth(v_{1j})}$$

*Formula III.3.31.* Dissimilarity of two weighted graph models without node values

In general, formulas [III.3.30](#) and [III.3.31](#) for specifying the dissimilarity of two weighted graphs are only applied into the case that graph satisfies two conditions:

- Firstly, graph must be the multi-depth tree like the one depicted in figures [III.3.15](#) and [III.3.16](#).
- Secondly, the sum of weights at nodes having the same parent equals 1,  
 $\sum weight(v_{ij}) = 1$   
 $v_{ij}$  has the same parent

It is easy to infer that such graph is sigma graph mentioned in previous subsection [III.3.13](#).

For example, there are three weighted graph models  $G_1$ ,  $G_2$ , and  $G_3$  whose structures are the same to the structure shown in figures [III.3.15](#) and [III.3.16](#). The weights of their arcs are expressed in figure [III.3.16](#). The values of their nodes were shown in table [III.3.33](#), which are shows repeatedly as follows:

	A	B	C	D	E	F	G
$G_1$	2	1	1	0	3	2	1
$G_2$	1	1	0	1	4	5	4
$G_3$	2	1	1	1	4	5	4

Applying formula [III.3.30](#) into these node values, dissimilarity measures between  $G_3$  and  $G_1$ ,  $G_2$ , respectively are computed as below:

$$dissim(G_1, G_3)$$

$$\begin{aligned} &= \frac{|2 - 2|}{1} * 1 + \frac{|1 - 1|}{2} * 0.2 + \frac{|1 - 1|}{2} * 0.8 + \frac{|0 - 1|}{3} * 0.6 \\ &+ \frac{|3 - 4|}{3} * 0.4 + \frac{|2 - 5|}{3} * 0.3 + \frac{|1 - 4|}{3} * 0.7 \\ &= 0 + 0 + 0 + 0.2 + \frac{0.4}{3} + 0.3 + 0.7 \approx 1.33 \end{aligned}$$

$$dissim(G_2, G_3)$$

$$\begin{aligned} &= \frac{|1 - 2|}{1} * 1 + \frac{|1 - 1|}{2} * 0.2 + \frac{|0 - 1|}{2} * 0.8 + \frac{|1 - 1|}{3} * 0.6 \\ &+ \frac{|4 - 4|}{3} * 0.4 + \frac{|5 - 5|}{3} * 0.3 + \frac{|4 - 4|}{3} * 0.7 \\ &= 1 + 0 + 0.4 + 0 + 0 + 0 + 0 = 1.4 \end{aligned}$$

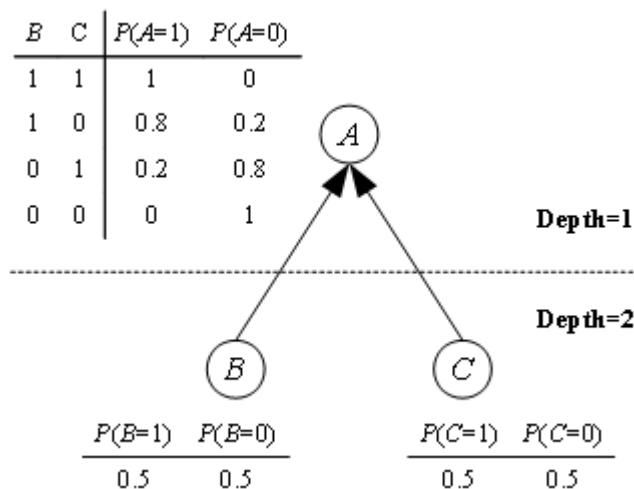
So  $G_1$  is more similar to  $G_3$  than  $G_2$  is because  $dissim(G_1, G_3)$  is smaller than  $dissim(G_2, G_3)$ .

The dissimilarity measure specified by formula [III.3.30](#) will be modified when graph model is evolved as Bayesian network. The successive part [III.3.3.2.2](#) describes how to specify dissimilarity measure in case that graph model is Bayesian network.

### III.3.3.2.2. In case that graph model is Bayesian network

Bayesian network (BN) is the directed acyclic graph (DAG) constituted of a set of nodes and a set of directed arc. Each node is referred as a random variable and each arc expresses the relationship between two nodes. The strength of dependence is quantified by Conditional Probability Table (CPT). In other words, each node owns a CPT expressing its local conditional probability distribution. Each entry in the CPT is the conditional probability of a node given its parent. The marginal probability of a node expresses user's mastery of such node. Note that marginal probability is considered as posterior probability. Please see sub-section III.1.1.1 for more details about BN.

Suppose the structure of BN is in form of tree, figure III.3.17 shows our considered BN.



**Figure III.3.17.** Bayesian network and its CPT (s)

It is easy to recognize that BN in form of tree as shown in figure III.3.17 is a sigma graph; please see sub-section III.1.1.3 for more details about sigma graph.

Let  $G_1$  and  $G_2$  be two BN (s) of user 1 and user 2, respectively.

$$G_1 = \langle V_1, E_1 \rangle \text{ and } U_2 = G_2 = \langle V_2, E_2 \rangle$$

Where  $V_i$  and  $E_i$  are a set of nodes and a set of arcs, respectively. The dissimilarity (or distance) of two Bayesian networks  $G_1$  and  $G_2$  is defined by formula III.3.32 as follows:

$$\text{dissim}(G_1, G_2) = \text{distance}(G_1, G_2) = \sum_{j=1}^n \frac{|P(v_{1j}) - P(v_{2j})|}{\text{depth}(v_{1j})}$$

**Formula III.3.32.** Dissimilarity of two Bayesian networks

Where  $P(v_{ij})$  is the posterior probability of node  $v_{ij}$  ( $=1$ ) in network  $G_i$  with attention that node  $v_{ij}$  is binary random variable. Please see sub-section III.1.1.1 for more details about BN and posterior probability.

In general, formula III.3.32 for specifying the dissimilarity of two BN (s) are only applied into the case that BN satisfies the condition that the structure of BN must be sigma graph mentioned in sub-section III.1.1.3.

For example, there are three weighted graph models  $G_1$ ,  $G_2$ , and  $G_3$  whose structures are the same to the structure shown in figure III.3.17. The values of their nodes are shown in table III.3.34 as follows:

	A	B	C
$G_1$	1	1	1
$G_2$	1	1	0
$G_3$	0	0	1

**Table III.3.34.** Values of nodes in Bayesian network

Let  $v_{11}$ ,  $v_{12}$ , and  $v_{13}$  denote nodes A, B, and C, respectively of  $G_1$ . Let  $v_{21}$ ,  $v_{22}$ , and  $v_{23}$  denote nodes A, B, and C, respectively of  $G_2$ . Let  $v_{31}$ ,  $v_{32}$ , and  $v_{33}$  denote nodes A, B, and C, respectively of  $G_3$ . We have  $\text{depth}(v_{11}) = \text{depth}(v_{21}) = \text{depth}(v_{31}) = \text{depth}(A) = 1$ ,  $\text{depth}(v_{12}) = \text{depth}(v_{22}) = \text{depth}(v_{32}) = \text{depth}(B) = 2$ , and  $\text{depth}(v_{13}) = \text{depth}(v_{23}) = \text{depth}(v_{33}) = \text{depth}(C) = 2$ . According to BN depicted in figure III.1.1.3, the joint probability distribution is:

$$P(A, B, C) = P(B) * P(C) * P(A|B, C)$$

Please see formulas III.1.2 and III.1.2' for more details about joint probability distribution. Let  $P(v_{ij})$  be the posterior probability of node  $v_{ij}$  (=1). Note that all conditional probabilities (CPT (s)) mentioned below are shown in figure III.1.1.3. We have:

$$\begin{aligned} P(A = 1) &= P(A = 1, B = 1, C = 1) + P(A = 1, B = 1, C = 0) \\ &\quad + P(A = 1, B = 0, C = 1) + P(A = 1, B = 0, C = 0) \\ &= P(B = 1) * P(C = 1) * P(A = 1|B = 1, C = 1) \\ &\quad + P(B = 1) * P(C = 0) * P(A = 1|B = 1, C = 0) \\ &\quad + P(B = 0) * P(C = 1) * P(A = 1|B = 0, C = 1) \\ &\quad + P(B = 0) * P(C = 0) * P(A = 1|B = 0, C = 0) \\ &= 0.5 * 0.5 * 1 + 0.5 * 0.5 * 0.8 + 0.5 * 0.5 * 0.2 + 0.5 * 0.5 * 0 = 0.5 \end{aligned}$$

Suppose network  $G_1$  has two evidences  $B=1$  and  $C=1$ , we have:

$$P(v_{12}) = P(B = 1) = 0.5$$

$$P(v_{13}) = P(C = 1) = 0.5$$

$$P(v_{11}) = P(A = 1|B = 1, C = 1) = \frac{P(A = 1, B = 1, C = 1)}{P(A = 1)}$$

(by applying formula III.1.3' for specifying posterior probability)

$$= \frac{P(B = 1) * P(C = 1) * P(A = 1|B = 1, C = 1)}{P(A = 1)} = \frac{0.5 * 0.5 * 1}{0.5} = 0.5$$

Suppose network  $G_2$  has two evidences  $B=1$  and  $C=0$ , we have:

$$P(v_{22}) = P(B = 1) = 0.5$$

### III.3. Learning history sub-model

---

$$\begin{aligned}
 P(v_{23}) &= P(C = 1) = 1 - P(C = 0) = 1 - 0.5 = 0.5 \\
 P(v_{21}) &= P(A = 1|B = 1, C = 0) = \frac{P(A = 1, B = 1, C = 0)}{P(A = 1)} \\
 &\quad (\text{by applying formula III.1.3' for specifying posterior probability}) \\
 &= \frac{P(B = 1) * P(C = 0) * P(A = 1|B = 1, C = 0)}{P(A = 1)} = \frac{0.5 * 0.5 * 0.8}{0.5} = 0.4
 \end{aligned}$$

Suppose network  $G_3$  has two evidences  $B=0$  and  $C=1$ , we have:

$$\begin{aligned}
 P(v_{32}) &= P(B = 0) = 1 - P(B = 1) = 1 - 0.5 = 0.5 \\
 P(v_{33}) &= P(C = 1) = 0.5 \\
 P(v_{31}) &= P(A = 1|B = 0, C = 1) = \frac{P(A = 1, B = 0, C = 1)}{P(A = 1)} \\
 &\quad (\text{by applying formula III.1.3' for specifying posterior probability}) \\
 &= \frac{P(B = 0) * P(C = 1) * P(A = 1|B = 0, C = 1)}{P(A = 1)} = \frac{0.5 * 0.5 * 0.2}{0.5} = 0.1
 \end{aligned}$$

According to formula III.3.32 into node values shown in table III.3.34, dissimilarity measures between  $G_3$  and  $G_1$ ,  $G_2$ , respectively are computed as below:

$$\begin{aligned}
 \text{dissim}(G_1, G_3) &= \frac{|P(v_{11}) - P(v_{31})|}{\text{depth}(v_{11})} + \frac{|P(v_{12}) - P(v_{32})|}{\text{depth}(v_{12})} + \frac{|P(v_{13}) - P(v_{33})|}{\text{depth}(v_{12})} \\
 &= \frac{|0.5 - 0.1|}{1} + \frac{|0.5 - 0.5|}{2} + \frac{|0.5 - 0.5|}{2} = 0.4 \\
 \text{dissim}(G_2, G_3) &= \frac{|P(v_{21}) - P(v_{31})|}{\text{depth}(v_{11})} + \frac{|P(v_{22}) - P(v_{32})|}{\text{depth}(v_{12})} + \frac{|P(v_{23}) - P(v_{33})|}{\text{depth}(v_{12})} \\
 &= \frac{|0.4 - 0.1|}{1} + \frac{|0.5 - 0.5|}{2} + \frac{|0.5 - 0.5|}{2} = 0.3
 \end{aligned}$$

So  $G_2$  is more similar to  $G_3$  than  $G_1$  is because  $\text{dissim}(G_2, G_3)$  is smaller than  $\text{dissim}(G_1, G_3)$ .

Dissimilarity measure quantifies the distance or difference between two user models but there is a demand of how to estimate similarity or likeness between two models. Although it is possible to specify the similarity as the inverse of dissimilarity measure, it is necessary to use similarity measure in many situations. Moreover, similarity measure is more effective than dissimilarity measure in some applications such as content-based filtering and collaborative filtering (Ricci, Rokach, Shapira, & Kantor, 2011, pp. 73-140). The next sub-section III.3.3.3 mentions similarity measure and how to modify  $k$ -mean algorithm so as to take advantages of similarity measure.

#### III.3.3.3. Similarity measures for clustering algorithms

Although dissimilarity measure considered as the physical distance between them are used to cluster user models, we can use another measure so-called

similarity measure so as to cluster user models. There are two typical similarity measures such as *cosine similarity measure* and *correlation coefficient*.

Suppose user model  $U_i$  is represented as vector  $U_i = \{u_{i1}, u_{i2}, \dots, u_{ij}, \dots, u_{in}\}$ , the cosine similarity measure of two user models is the cosine of the angle between two vectors, specified by formula III.3.33 as follows:

$$sim(U_i, U_j) = cosine(U_i, U_j) = \frac{U_i \cdot U_j}{|U_i| |U_j|} = \frac{\sum_{k=1}^n u_{ik} u_{jk}}{\sqrt{\sum_{k=1}^n u_{ik}^2} \sqrt{\sum_{k=1}^n u_{jk}^2}}$$

*Formula III.3.33.* Cosine similarity measure

Where the sign “.” denotes scalar product (dot product) of two vectors and the notation  $|.|$  denotes the length of a vector. Previous sub-section III.3.2.2.1 has already mentioned scalar product of two vectors and length of a vector.

The range of cosine similarity measure is from 0 to 1. If it equals 0, two users are totally different. If it equals 1, two users are identical. For example, the following table III.3.35 shows four user models.

	<i>feature<sub>1</sub></i>	<i>feature<sub>2</sub></i>	<i>feature<sub>3</sub></i>
<i>user<sub>1</sub></i>	1	2	1
<i>user<sub>2</sub></i>	2	1	2
<i>user<sub>3</sub></i>	4	1	5
<i>user<sub>4</sub></i>	1	2	0

**Table III.3.35.** Four user models

It is easy to interpret table III.3.35 that we have four user models  $user_1=(1,2,1)$ ,  $user_2=(2,1,2)$ ,  $user_3=(4,1,5)$ , and  $user_4=(1,2,0)$ .

The cosine similarity measures of user 4 and users 1, 2, 3 are computed as below:

$$sim(user_4, user_1) = \frac{1 * 1 + 2 * 2 + 0 * 1}{\sqrt{1^2 + 2^2 + 0^2} \sqrt{1^2 + 2^2 + 1^2}} \approx 0.91$$

$$sim(user_4, user_2) = \frac{1 * 2 + 2 * 1 + 0 * 2}{\sqrt{1^2 + 2^2 + 0^2} \sqrt{2^2 + 1^2 + 2^2}} \approx 0.60$$

$$sim(user_4, user_3) = \frac{1 * 4 + 2 * 1 + 0 * 5}{\sqrt{1^2 + 2^2 + 0^2} \sqrt{4^2 + 1^2 + 5^2}} \approx 0.41$$

According to cosine similarity measures, user 1 and user 2 are more similar to user 4 than user 3 is because both  $sim(user_4, user_1)$  and  $sim(user_4, user_2)$  are greater than  $sim(user_4, user_3)$ .

On other hand, correlation coefficient (Montgomery & Runger, 2003, p. 174) (Montgomery & Runger, 2003, p. 402) which is the concept in statistics is also used to specify the similarity of two vectors. Let  $\bar{U}_i$  be the mean of user model vector  $U_i$ , we have:

$$\bar{U}_i = \frac{1}{n} \sum_{k=1}^n u_{ik}$$

The correlation coefficient is defined by formula III.3.34 as follows:

$$sim(U_i, U_j) = correl(U_i, U_j) = \frac{\sum_{k=1}^n (u_{ik} - \bar{U}_i)(u_{jk} - \bar{U}_j)}{\sqrt{\sum_{k=1}^n (u_{ik} - \bar{U}_i)^2} \sqrt{\sum_{k=1}^n (u_{jk} - \bar{U}_j)^2}}$$

*Formula III.3.34.* Correlation coefficient

Where  $\bar{U}_i = \frac{1}{n} \sum_{k=1}^n u_{ik}$  and  $\bar{U}_j = \frac{1}{n} \sum_{k=1}^n u_{jk}$  are means of user vectors  $U_i$  and  $U_j$ , respectively.

The range of correlation coefficient is from  $-1$  to  $1$ . If it equals  $-1$ , two users are totally different. If it equals  $1$ , two users are identical. For example, the correlation coefficients of user 4 and users 1, 2, 3 are computed as below:

$$\bar{U}_4 = \frac{1+2+0}{3} = 1$$

$$\bar{U}_1 = \frac{1+2+1}{3} \approx 1.33$$

$$\bar{U}_2 = \frac{2+1+2}{3} \approx 1.66$$

$$\bar{U}_3 = \frac{4+1+5}{3} \approx 3.33$$

$$sim(user_4, user_1) = \frac{(1-1)(1-1.33) + (2-1)(2-1.33) + (0-1)(1-1.33)}{\sqrt{(1-1)^2 + (2-1)^2 + (0-1)^2} \sqrt{(1-1.33)^2 + (2-1.33)^2 + (1-1.33)^2}} \approx 0.86$$

$$sim(user_4, user_2) = \frac{(1-1)(2-1.66) + (2-1)(1-1.66) + (0-1)(2-1.66)}{\sqrt{(1-1)^2 + (2-1)^2 + (0-1)^2} \sqrt{(2-1.66)^2 + (1-1.66)^2 + (2-1.66)^2}} \approx -0.86$$

$$sim(user_4, user_3) = \frac{(1-1)(4-3.33) + (2-1)(1-3.33) + (0-1)(5-3.33)}{\sqrt{(1-1)^2 + (2-1)^2 + (0-1)^2} \sqrt{(4-3.33)^2 + (1-3.33)^2 + (5-3.33)^2}} \approx -0.96$$

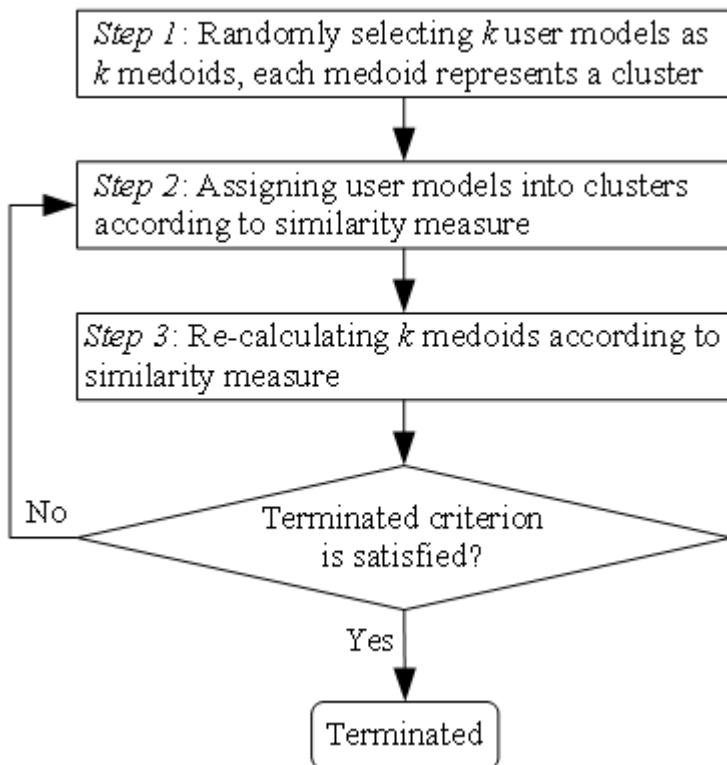
According to correlation coefficient, user 1 and user 2 are more similar to user 4 than to user 3 is because both  $sim(user_4, user_1)$  and  $sim(user_4, user_2)$  are greater than  $sim(user_4, user_3)$ .

If cosine measure and correlation coefficient are used as the similarity between two user vectors, the serious problem will occurs. That is impossible to specify the mean of each cluster because cosine measure and correlation coefficient have different semantic from Euclidean distance. Instead of using  $k$ -

mean algorithm, I apply  $k$ -medoid algorithm (Han & Kamber, 2006, pp. 404-406) into partitioning a collection of user models into  $k$  user model clusters. The “mean” is replaced by the “medoid” in  $k$ -medoid algorithm. The medoid is the actual user model and its average similarity to all other user models in the same cluster is maximal. I modify  $k$ -medoid algorithm so as to use similarity measures, as follows (Nguyen, User Model Clustering, 2014):

1. It randomly selects  $k$  user models as  $k$  medoids. Each medoid is considered as the “representative” of one cluster. There are  $k$  clusters.
2. For each remaining user model, the similarities between it and  $k$  medoids are computed. Such user model will belong to cluster  $C_i$  if the similarity measure of this user model and the medoid of  $C_i$  is the highest.
3. After that,  $k$  medoids are selected again so that the *average similarity* of each medoid to other user models in the same cluster is maximal. If  $k$  medoids are not changed or the error criterion is less than a pre-defined threshold, the algorithm is terminated; otherwise returning step 2.

Figure III.3.18 is the flow chart of  $k$ -medoid algorithm modified with similarity measure:



**Figure III.3.18.** Flow chart of  $k$ -medoid algorithm modified with similarity measure

Let  $M_i$  be the medoid of cluster  $i$  and let  $n_i$  be the number of user models inside cluster  $i$ . Recall that user model is represented as vector which is called user vector or user model vector. The *average similarity* of medoid  $M_i$  to other user models in the same cluster is:

$$a_i = \frac{1}{n_i} \sum_{U \in C_i} sim(U, M_i)$$

*Formula III.3.35.* Average similarity

The average similarity  $a_i$  of medoid  $M_i$  specified by formula III.3.35 is used in step 3 of  $k$ -medoid algorithm.

The *global average similarity* over  $k$  clusters is the sum of all similarities between user models and medoids  $M_i$  (s). Let  $A$  be global average similarity, formula III.3.36 specifies  $A$  as follows:

$$A = \frac{1}{k} \sum_{i=1}^k a_i = \frac{1}{k} \sum_{i=1}^k \left( \frac{1}{n_i} \sum_{U \in C_i} sim(U, M_i) \right)$$

*Formula III.3.36.* Global average similarity

Where  $M_i$  is the medoid of cluster  $i$  and  $sim(U, M_i)$  is similarity between user model  $U$  and medoid  $M_i$ . Note,  $sim(U, M_i)$  can be cosine measure or correlation coefficient. Please see formulas III.3.33 and III.3.34 for more details about cosine measure and correlation coefficient.

As seen in formula III.3.36, the global average similarity  $A$  can be defined via average similarities  $a_i$  (s). The average similarity  $a_i$  and the global average similarity  $A$  should be normalized in range [0, 1]. If we use correlation coefficient as similarity measure,  $a_i$  and  $A$  is normalized as follows:

$$a_i = \frac{a_i + 1}{2}$$

$$A = \frac{A + 1}{2}$$

As aforementioned, there are two alternative terminating conditions (stopping conditions) for  $k$ -medoid algorithm:

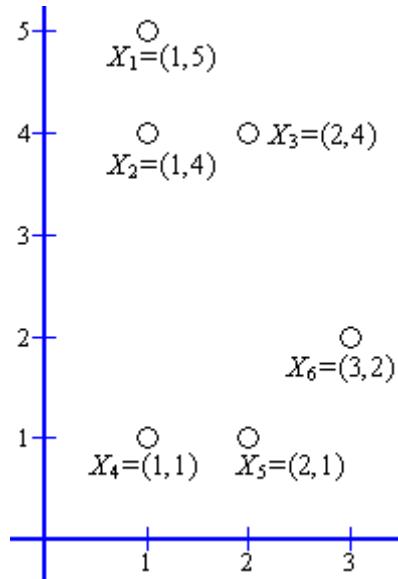
- The  $k$  medoids are not changed.
- Or, the error criterion is less than a pre-defined threshold.

The simplest way to calculate the error criterion as the inverse of global average similarity. Let  $Err$  be the error criterion, we have:

$$Err = 1 - A$$

Where  $A$  is the global average similarity specified by formula III.3.28.

It is necessary to give an example for illustrating the  $k$ -medoid algorithm modified with similarity measure. Suppose user models are 2-dimension vectors, figure III.3.19 expresses a sample of six 2-dimension user vectors:  $X_1=(1,5)$ ,  $X_2=(1,4)$ ,  $X_3=(2,4)$ ,  $X_4=(1,1)$ ,  $X_5=(2,1)$ , and  $X_6=(3,2)$ . We apply  $k$ -medoid algorithm in order to group these vectors into two clusters ( $k=2$ ).



**Figure III.3.19.** A sample of six user vectors

Two vectors  $X_1$  and  $X_2$  are selected arbitrary as two medoids  $M_1$  and  $M_2$ , respectively. We have:

$$\begin{aligned} M_1 &= X_1 = (1, 5) \\ M_2 &= X_2 = (1, 4) \end{aligned}$$

Medoids  $M_1$  and  $M_2$  represent cluster 1 and cluster 2, respectively. According to step 2 of  $k$ -medoid algorithm, it is required to determine clusters of vectors  $X_3=(2,4)$ ,  $X_4=(1,1)$ ,  $X_5=(2,1)$ , and  $X_6=(3,2)$  based on cosine similarities. Note that it is totally possible to use correlation coefficient as similarity measure. For example, according to formula III.3.33, the cosine between  $X_3$  and  $M_1$  is:

$$sim(X_3, M_1) = cosine(X_3, M_1) = \frac{2 * 1 + 4 * 5}{\sqrt{2^2 + 4^2} \sqrt{1^2 + 5^2}} \approx 0.96$$

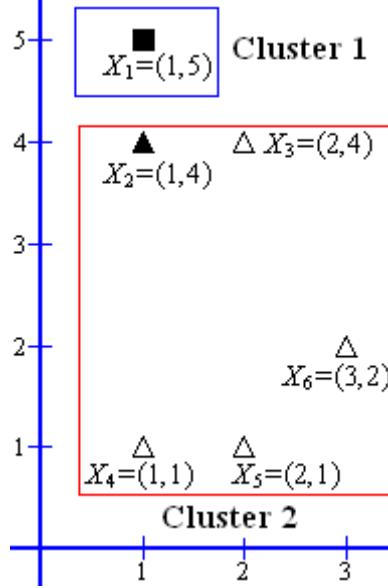
Table III.3.36 shows cosine similarities between user vectors ( $X_3$ ,  $X_4$ ,  $X_5$ ,  $X_6$ ) and medoids ( $M_1$ ,  $M_2$ ).

Vector	Cluster	Similarity	Max similarity
$X_3=(2,4)$	$M_1=(1,5)$	$cosine(X_3, M_1) \approx 0.96$	$cosine(X_3, M_2) \approx 0.98$
	$M_2=(1,4)$	$cosine(X_3, M_2) \approx 0.98$	
$X_4=(1,1)$	$M_1=(1,5)$	$cosine(X_4, M_1) \approx 0.83$	$cosine(X_4, M_2) \approx 0.86$
	$M_2=(1,4)$	$cosine(X_4, M_2) \approx 0.86$	
$X_5=(2,1)$	$M_1=(1,5)$	$cosine(X_5, M_1) \approx 0.61$	$cosine(X_5, M_2) \approx 0.65$
	$M_2=(1,4)$	$cosine(X_5, M_2) \approx 0.65$	
$X_6=(3,2)$	$M_1=(1,5)$	$cosine(X_6, M_1) \approx 0.71$	$cosine(X_6, M_2) \approx 0.74$
	$M_2=(1,4)$	$cosine(X_6, M_2) \approx 0.74$	

**Table III.3.36.** Cosine similarities between user vectors and medoids

Because  $cosine(X_3, M_2)$ ,  $cosine(X_4, M_2)$ ,  $cosine(X_5, M_2)$ , and  $cosine(X_6, M_2)$  get maximal, it is easy to infer that  $X_3$ ,  $X_4$ ,  $X_5$ , and  $X_6$  belong to cluster 2 according to step 2 of  $k$ -medoid algorithm. Of course,  $X_1$  and  $X_2$  belong to cluster 1 and 2,

respectively because they are themselves medoids  $M_1$  and  $M_2$ . Figure III.3.20 shows cluster 1 containing  $X_1$  and cluster 2 containing  $X_2, X_3, X_4, X_5, X_6$ .



**Figure III.3.20.** Two clusters resulted from  $k$ -medoid algorithm at the first running time

Where rectangles represent vectors that belong to cluster 1 and triangles represent vectors that belong to cluster 2. Medoids  $M_1$  and  $M_2$  are drawn as two solid shapes.

According to step 3 of  $k$ -medoid algorithm, medoids  $M_1$  and  $M_2$  are re-calculated. The medoid  $M_1$  is not changed because cluster 1 contains only  $M_1=X_1$ . It is necessary to calculate average similarity for each vector in cluster 2 for choosing  $M_2$ ; please see III.3.35 for details about average similarity. Let  $a_2, a_3, a_4, a_5$ , and  $a_6$  be average similarities of  $X_2, X_3, X_4, X_5$ , and  $X_6$  in cluster 2, respectively. Suppose  $X_2$  is the medoid, we have:

$$\begin{aligned} a_2 &= \frac{1}{4} (sim(X_2, X_3) + sim(X_2, X_4) + sim(X_2, X_5) + sim(X_2, X_6)) \\ &= \frac{1}{4} (cosine(X_2, X_3) + cosine(X_2, X_4) + cosine(X_2, X_5) \\ &\quad + cosine(X_2, X_6)) \\ &= \frac{1}{4} \left( \frac{1 * 2 + 4 * 4}{\sqrt{1^2 + 4^2} \sqrt{2^2 + 4^2}} + \frac{1 * 1 + 4 * 1}{\sqrt{1^2 + 4^2} \sqrt{1^2 + 1^2}} \right. \\ &\quad \left. + \frac{1 * 2 + 4 * 1}{\sqrt{1^2 + 4^2} \sqrt{2^2 + 1^2}} + \frac{1 * 3 + 4 * 2}{\sqrt{1^2 + 4^2} \sqrt{3^2 + 2^2}} \right) \approx 0.806 \end{aligned}$$

Similarly, we have:

$$\begin{aligned} a_3 &= \frac{1}{4} (sim(X_3, X_2) + sim(X_3, X_4) + sim(X_3, X_5) + sim(X_3, X_6)) \\ &= \frac{1}{4} \left( \frac{2 * 1 + 4 * 4}{\sqrt{2^2 + 4^2} \sqrt{1^2 + 4^2}} + \frac{2 * 1 + 4 * 1}{\sqrt{2^2 + 4^2} \sqrt{1^2 + 1^2}} \right. \\ &\quad \left. + \frac{2 * 2 + 4 * 1}{\sqrt{2^2 + 4^2} \sqrt{2^2 + 1^2}} + \frac{2 * 3 + 4 * 2}{\sqrt{2^2 + 4^2} \sqrt{3^2 + 2^2}} \right) \approx 0.898 \end{aligned}$$

$$\begin{aligned}
 a_4 &= \frac{1}{4} (sim(X_4, X_2) + sim(X_4, X_3) + sim(X_4, X_5) + sim(X_4, X_6)) \\
 &= \frac{1}{4} \left( \frac{1 * 1 + 1 * 4}{\sqrt{1^2 + 1^2} \sqrt{1^2 + 4^2}} + \frac{1 * 2 + 1 * 4}{\sqrt{1^2 + 1^2} \sqrt{2^2 + 4^2}} \right. \\
 &\quad \left. + \frac{1 * 2 + 1 * 1}{\sqrt{1^2 + 1^2} \sqrt{2^2 + 1^2}} + \frac{1 * 3 + 1 * 2}{\sqrt{1^2 + 1^2} \sqrt{3^2 + 2^2}} \right) \approx 0.934 \\
 a_5 &= \frac{1}{4} (sim(X_5, X_2) + sim(X_5, X_3) + sim(X_5, X_4) + sim(X_5, X_6)) \\
 &= \frac{1}{4} \left( \frac{2 * 1 + 1 * 4}{\sqrt{2^2 + 1^2} \sqrt{1^2 + 4^2}} + \frac{2 * 2 + 1 * 4}{\sqrt{2^2 + 1^2} \sqrt{2^2 + 4^2}} \right. \\
 &\quad \left. + \frac{2 * 1 + 1 * 1}{\sqrt{2^2 + 1^2} \sqrt{1^2 + 1^2}} + \frac{2 * 3 + 1 * 2}{\sqrt{2^2 + 1^2} \sqrt{3^2 + 2^2}} \right) \approx 0.848 \\
 a_6 &= \frac{1}{4} (sim(X_6, X_2) + sim(X_6, X_3) + sim(X_6, X_4) + sim(X_6, X_5)) \\
 &= \frac{1}{4} \left( \frac{3 * 1 + 2 * 4}{\sqrt{3^2 + 2^2} \sqrt{1^2 + 4^2}} + \frac{3 * 2 + 2 * 4}{\sqrt{3^2 + 2^2} \sqrt{2^2 + 4^2}} \right. \\
 &\quad \left. + \frac{3 * 1 + 2 * 1}{\sqrt{3^2 + 2^2} \sqrt{1^2 + 1^2}} + \frac{3 * 2 + 2 * 1}{\sqrt{3^2 + 2^2} \sqrt{2^2 + 1^2}} \right) \approx 0.895
 \end{aligned}$$

Because the average similarity  $a_4$  is maximum, the vector  $X_4$  is selected as medoid  $M_2$ . We have:

$$\begin{aligned}
 M_1 &= X_1 = (1, 5) \\
 M_2 &= X_4 = (1, 1)
 \end{aligned}$$

At the second running time of step 2 of  $k$ -medoid algorithm, it is required to determine which cluster each user vector belongs to, based on cosine similarities. Table III.3.37 shows cosine similarities between user vectors ( $X_2, X_3, X_5, X_6$ ) and medoids ( $M_1, M_2$ ).

Vector	Cluster	Similarity	Max similarity
$X_2=(1,4)$	$M_1=(1,5)$	$\text{cosine}(X_2, M_1) \approx 1$	$\text{cosine}(X_2, M_1) \approx 1$
	$M_2=(1,1)$	$\text{cosine}(X_2, M_2) \approx 0.86$	
$X_3=(2,4)$	$M_1=(1,5)$	$\text{cosine}(X_3, M_1) \approx 0.96$	$\text{cosine}(X_3, M_1) \approx 0.96$
	$M_2=(1,1)$	$\text{cosine}(X_3, M_2) \approx 0.95$	
$X_5=(2,1)$	$M_1=(1,5)$	$\text{cosine}(X_5, M_1) \approx 0.61$	$\text{cosine}(X_5, M_2) \approx 0.95$
	$M_2=(1,1)$	$\text{cosine}(X_5, M_2) \approx 0.95$	
$X_6=(3,2)$	$M_1=(1,5)$	$\text{cosine}(X_6, M_1) \approx 0.71$	$\text{cosine}(X_6, M_2) \approx 0.98$
	$M_2=(1,1)$	$\text{cosine}(X_6, M_2) \approx 0.98$	

**Table III.3.37.** Cosine similarities between user vectors and medoids at the second running time of  $k$ -medoid algorithm

Because  $\text{cosine}(X_2, M_1)$ ,  $\text{cosine}(X_3, M_1)$ ,  $\text{cosine}(X_5, M_2)$ , and  $\text{cosine}(X_6, M_2)$  get maximal, it is easy to infer that  $X_2, X_3$  belong to cluster 1 and  $X_5, X_6$  belong to cluster 2, respectively. Of course,  $X_1$  and  $X_4$  belong to cluster 1 and 2, respectively because they are themselves medoids  $M_1$  and  $M_2$ . So, we have:

- Cluster 1:  $X_1=(1, 5), X_2=(1, 4), X_3=(2, 4)$ .
- Cluster 2:  $X_4=(1, 1), X_5=(2, 1), X_6=(3, 2)$ .

According to step 3 of  $k$ -medoid algorithm, medoids  $M_1$  and  $M_2$  are recalculated. Let  $a_1$ ,  $a_2$ , and  $a_3$  be average similarities of  $X_1$ ,  $X_2$ , and  $X_3$  in cluster 1, respectively. Let  $a_4$ ,  $a_5$ , and  $a_6$  be average similarities of  $X_4$ ,  $X_5$ , and  $X_6$  in cluster 2, respectively. We have:

$$\begin{aligned} a_1 &= \frac{1}{2}(sim(X_1, X_2) + sim(X_1, X_3)) \\ &= \frac{1}{2}\left(\frac{1 * 1 + 5 * 4}{\sqrt{1^2 + 5^2}\sqrt{1^2 + 4^2}} + \frac{1 * 2 + 5 * 4}{\sqrt{1^2 + 5^2}\sqrt{2^2 + 4^2}}\right) \approx 0.982 \\ a_2 &= \frac{1}{2}(sim(X_2, X_1) + sim(X_2, X_3)) \\ &= \frac{1}{2}\left(\frac{1 * 1 + 4 * 5}{\sqrt{1^2 + 4^2}\sqrt{1^2 + 5^2}} + \frac{1 * 2 + 4 * 4}{\sqrt{1^2 + 4^2}\sqrt{2^2 + 4^2}}\right) \approx 0.988 \\ a_3 &= \frac{1}{2}(sim(X_3, X_1) + sim(X_3, X_2)) \\ &= \frac{1}{2}\left(\frac{2 * 1 + 4 * 5}{\sqrt{2^2 + 4^2}\sqrt{1^2 + 5^2}} + \frac{2 * 1 + 4 * 4}{\sqrt{2^2 + 4^2}\sqrt{1^2 + 4^2}}\right) \approx 0.970 \end{aligned}$$

Because the average similarity  $a_2$  is maximum, the vector  $X_2$  is selected as medoid  $M_1$ .

$$M_1 = X_2 = (1, 4)$$

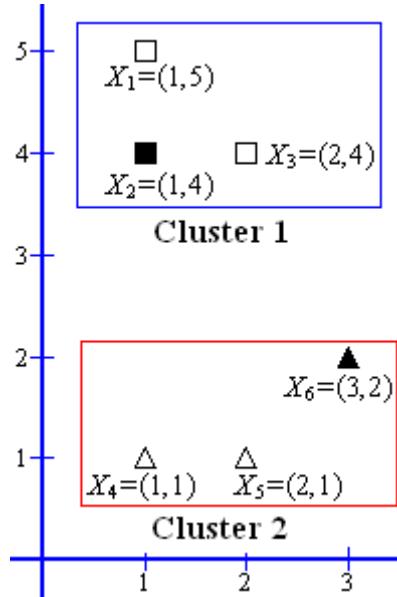
We have:

$$\begin{aligned} a_4 &= \frac{1}{2}(sim(X_4, X_5) + sim(X_4, X_6)) \\ &= \frac{1}{2}\left(\frac{1 * 2 + 1 * 1}{\sqrt{1^2 + 1^2}\sqrt{2^2 + 1^2}} + \frac{1 * 3 + 1 * 2}{\sqrt{1^2 + 1^2}\sqrt{3^2 + 2^2}}\right) \approx 0.965 \\ a_5 &= \frac{1}{2}(sim(X_5, X_4) + sim(X_5, X_6)) \\ &= \frac{1}{2}\left(\frac{2 * 1 + 1 * 1}{\sqrt{2^2 + 1^2}\sqrt{1^2 + 1^2}} + \frac{2 * 1 + 1 * 1}{\sqrt{2^2 + 1^2}\sqrt{1^2 + 1^2}}\right) \approx 0.970 \\ a_6 &= \frac{1}{2}(sim(X_6, X_4) + sim(X_6, X_5)) \\ &= \frac{1}{2}\left(\frac{3 * 1 + 2 * 1}{\sqrt{3^2 + 2^2}\sqrt{1^2 + 1^2}} + \frac{3 * 2 + 2 * 1}{\sqrt{3^2 + 2^2}\sqrt{2^2 + 1^2}}\right) \approx 0.986 \end{aligned}$$

Because the average similarity  $a_6$  is maximum, the vector  $X_6$  is selected as medoid  $M_2$ .

$$M_2 = X_6 = (3, 2)$$

Figure III.3.21 shows cluster 1 containing  $X_1$ ,  $X_2$ ,  $X_3$  and cluster 2 containing  $X_4$ ,  $X_5$ ,  $X_6$ .



**Figure III.3.21.** Two clusters resulted from  $k$ -medoid algorithm at the second running time

Where rectangles represent vectors that belong to cluster 1 and triangles represent vectors that belong to cluster 2. Medoids  $M_1$  and  $M_2$  are drawn as two solid shapes.

At the third running time of step 3 of  $k$ -medoid algorithm, it is required to determine which cluster each user vector belongs to, based on cosine similarities. Table III.3.38 shows cosine similarities between user vectors ( $X_1, X_3, X_4, X_5$ ) and medoids ( $M_1, M_2$ ).

Vector	Cluster	Similarity	Max similarity
$X_1 = (1, 5)$	$M_1 = (1, 4)$	$\text{cosine}(X_1, M_1) \approx 1$	$\text{cosine}(X_1, M_1) \approx 1$
	$M_2 = (3, 2)$	$\text{cosine}(X_1, M_2) \approx 0.71$	
$X_3 = (2, 4)$	$M_1 = (1, 4)$	$\text{cosine}(X_3, M_1) \approx 0.98$	$\text{cosine}(X_3, M_1) \approx 0.98$
	$M_2 = (3, 2)$	$\text{cosine}(X_3, M_2) \approx 0.87$	
$X_4 = (1, 1)$	$M_1 = (1, 4)$	$\text{cosine}(X_4, M_1) \approx 0.86$	$\text{cosine}(X_4, M_2) \approx 0.98$
	$M_2 = (3, 2)$	$\text{cosine}(X_4, M_2) \approx 0.98$	
$X_5 = (2, 1)$	$M_1 = (1, 4)$	$\text{cosine}(X_5, M_1) \approx 0.65$	$\text{cosine}(X_5, M_2) \approx 1$
	$M_2 = (3, 2)$	$\text{cosine}(X_5, M_2) \approx 1$	

**Table III.3.38.** Cosine similarities between user vectors and medoids at the third running time of  $k$ -medoid algorithm

Because  $\text{cosine}(X_1, M_1)$ ,  $\text{cosine}(X_3, M_1)$ ,  $\text{cosine}(X_4, M_2)$ , and  $\text{cosine}(X_5, M_2)$  get maximal, it is easy to infer that  $X_1, X_3$  belong to cluster 1 and  $X_4, X_5$  belong to cluster 2, respectively. Of course,  $X_2$  and  $X_6$  belong to cluster 1 and 2, respectively because they are themselves medoids  $M_1$  and  $M_2$ . It is easy to recognize that there is no change in clusters. Of course, medoids  $M_1$  and  $M_2$  are not changed. According to step 3, the  $k$ -medoid algorithm is stopped. Therefore, the figure III.3.21 shows the final result of  $k$ -medoid algorithm given our sample. In general, we have:

- Cluster 1:  $X_1=(1, 5), X_2=(1, 4), X_3=(2, 4)$  with medoid  $M_1=X_2=(1, 4)$ .
- Cluster 2:  $X_4=(1, 1), X_5=(2, 1), X_6=(3, 2)$  with medoid  $M_2=X_6=(3, 2)$ .

Now our example illustrating  $k$ -medoid algorithm ends up this sub-section [III.3.3.3](#) which ends up main contents of constructing user groups based on clustering user models, in turn. The successive sub-section [III.3.3.4](#) is the conclusion of the third extended function of learning history sub-model that is constructing user groups – the main subject of sub-section [III.3.3](#).

#### **III.3.3.4. Evaluation**

It is easy to infer that the essence of constructing user groups is to cluster user models and so this sub-section [III.3.3](#) focuses on clustering techniques and similarity (dissimilarity) measures. Therefore, we discussed two clustering algorithms:  $k$ -mean and  $k$ -medoid. It is concluded that the dissimilarity measure considered as distance between two user models is appropriate to  $k$ -mean algorithm and the similarity measures such as cosine similarity measure and correlation coefficient are fit to  $k$ -medoid algorithm. The reason is that the essence of specifying the mean of cluster relates to how to compute the distances among user models. Conversely, the cosine similarity measure and correlation coefficient are more effective than distance measure because they pay attention to the direction of user vector. For example, the range of correlation coefficient is from  $-1$ : “two users are totally different” to  $1$ : “two users are identical”.

However, cosine similarity measure and correlation coefficient are only used for vector model; they cannot be applied into overlay model, Bayesian network model. It is clear to say that the formulas I proposed to compute the dissimilarity of two overlay models (or Bayesian network) are the variants of distances between user models.

As a result, [Triangular Learner Model](#) (TLM) is constituted of three sub-models such as knowledge, learning styles and learning history. The third sub-model (learning history sub-model) which is the most important one was described thoroughly in section [III.3](#). Three extended functions of learning history sub-model such as learning concept recommendation, discovering user interests and constructing user groups were successively in sub-sections [III.3.1](#), [III.3.2](#), and [III.3.3](#). The last sub-section [III.3.3](#) – how to construct user groups was already introduced to you; in other words, the basic content of this research was presented to you with the full of detailed description focused on how to TLM is constructed and how to the user modeling system [Zebra](#) is built up and manipulates TLM. The next section [IV](#) is the evaluation on TLM.

## Chapter IV. Evaluation of Triangular Learner Model

Chapter I is the state-of-art of user model and user modeling system. Chapters II, III are essential chapters which focus on main works of research including how to [Triangular Learner Model](#) (TLM) is constructed and how to the user modeling system [Zebra](#) is built up and manipulates TLM. As a result, TLM is constituted of three sub-models such as knowledge (section III.1), learning styles (section III.2) and learning history (section III.3).

In general, this research is fundamental research; thus, approaches, models and mathematical formulas are proposed as a perfect whole methodology. This research is not experimental research and testing technique on testing data is not appropriate to evaluate this research. So there are three ways to evaluate this research:

- The correctness of formulas is proven by mathematical tools and logical reasoning.
- The feasibility of model and architecture is authenticated via the software associated with Zebra and TLM. Moreover, the adaptive learning system that interacts to Zebra is also implemented as e-learning web-based software. The user modeling system Zebra and the e-learning web-based software can be downloaded via link <https://sites.google.com/site/ngphloc/st/dissertations/zebra>.
- The effectiveness of adaptive learning is proven via approaches described in this chapter IV.

This chapter IV focuses on evaluating learner model TLM and user modeling system Zebra with regard to their effectiveness. Section IV.1 is the evaluation on knowledge sub-model. Section IV.2 is the evaluation on the effectiveness of adaptive learning model, especially, the whole TLM and modeling system Zebra.

### IV.1. Evaluation of knowledge model

Bayesian network is the most important component of inference mechanism in the user modeling system [Zebra](#) when it and hidden Markov model constitutes the *belief network engine* in the core engine. Note that the core of Zebra is the inference engine having two sub-engines: belief network engine (BNE) and mining engine (ME). Bayesian network is used to assess user knowledge and ME is aimed to discover new assumptions about user and to support personal recommendation. The attempt to improve Bayesian network is the same to enhancing BNE. Please see sub-section II.2.2 for more details about Zebra and its engines such as BNE and ME. Sub-sections III.1.1.1 and III.2.4 are good introductions of Bayesian network and hidden Markov model. Because knowledge sub-model inside [Triangular Learner Model](#) (TLM) and Zebra is constructed by Bayesian network, this section IV.1 includes two successive sub-sections:

- Sub-section IV.1.1 aims to evaluate Bayesian model in methodological point of view. For example, sub-section answers the question “what the aspects of proposed Bayesian model are and how effective the proposed Bayesian model are”.
- Sub-section IV.1.2 mentions how to exploit Bayesian model in practical assessment. For example, sub-section answers the question “how to applying Bayesian model into determining how much a user masters over an knowledge item”

### **IV.1.1. Evaluation of Bayesian network**

There are three methods of building up Bayesian network (BN) user model: expert centric, efficiency centric and data centric (Mayo, 2001, pp. 74-81). These methods are distinguished based on how to construct BN and how to specify conditional probabilities. In brief, the main issue of such methods relates to qualitative model (structure) and quantitative model (conditional probabilities). Each method has respective strong points and drawbacks.

- *Expert-centric method:* The structure and conditional probabilities are defined totally by experts. This is the easiest method because there is no learning algorithm which is necessary to both qualitative model and quantitative model. Expert is responsible for all modeling tasks. However the drawback of this approach is that the BN is too dependent on expert's subjective thinking to evaluate the quality of network. Maybe the network has more redundant variables or the conditional probabilities don't reflect exactly the strength of relationships between variables in real data. BN user model built up by this method is called expert-centric model.
- *Efficiency-centric method:* The structure and conditional probabilities are specified and restricted based on some restrictions. These restrictions are defined to maximize some aspects of efficiency such as the optimal number of variables, the hierarchy of domain knowledge, and the evaluation time. BN user model built up by this method is called efficiency-centric model.
- *Data-centric method:* The structure and conditional probabilities are learned directly from real-world data by machine learning algorithms (Mayo, 2001, p. 81). BN user model built up by this method is called data-centric model. This method achieves some advantages when the number of variables may be smaller than expert centric and efficiency centric method because the network is deduced from actual data and so there is no redundant variables. It is easy to evaluate the quality of network by applying network into the testing data. Both observed and hidden variables are regarded in data centric method. However there is a critical drawback of data-centric model when the complexity of learning algorithms decreases the performance of user modeling tasks in run time. It takes more time to do inference in data-centric model than expert-centric model or efficiency-centric model.

The main technique used in Bayesian model of **Zebra** can belong to efficiency centric method in which the criterion of constructing user model is to map BN to learning curriculum. The hierarchy of variables is identical to the structure of subjects, topics and lessons in the curriculum. In other words, all subjects, topics and lessons become variables (or knowledge items) of Bayesian model in the same order. The strengths of relationships among variables are set up according to the importance of these subjects, topics and lessons. For example, given subject *A* if the effect of subject *B* on *A* is more significant than the effect of subject *C* on *A* is, the weight of relationship between *B* and *A* is larger than the weight of relationship between *C* and *A*. This approach achieves the same result to data-centric method in learning context because the expert in learning context is a teacher. The teacher masters over the curriculum and so quality of structure of BN is trustworthy. There is no need to apply complex learning algorithms like data-centric method and the performance of inference tasks in run time is kept stable and fast. This approach is essentially the combination of BN and overlay model, which aims to build up knowledge sub-model as Bayesian overlay model. Please see sub-section [III.1.1.2](#) for more details about Bayesian overlay model

Moreover the research also introduces two other techniques to improve the conditional probability tables (or parameters) and the structure of BN.

The first technique is called the evolution of parameters. It is essentially parameter learning process aiming to reflect the relationships between variables more precisely and more precisely. This technique is to improve conditional probability distribution by applying inference mechanism inside beta density function. Concretely, suppose variables  $X_i$  in BN have prior conditional probabilities and there are a set of evidences, the parameters (conditional probability tables) of BN are evolved by posterior conditional probabilities. These posterior probabilities are updated according to conditional expectations of beta density function, based on given evidences. Moreover, maximization expectation (EM) algorithm is used to calculate posterior probabilities in case of missing data. EM algorithm is implemented in offline process so as not to decrease the system performance. Please see sub-section [III.1.3](#) for more details about the evolution of parameters inside Bayesian overlay model.

The second technique is to aim to improve the structure of BN. It uses the dynamic Bayesian network (DBN) to model the temporal relationships among variables when the static network cannot reflect such relationships. Thus DBN can monitor user's learning process and provide immediately new changes in user model to adaptive system. However, the number of variables in DBN becomes huge when the modeling process continues in a long time and this affects seriously the performance of inference tasks. This research suggests a new way to keep the number of variables stable by obeying the Markov property, namely, the conditional probability of current time point  $t$  is only relevant to previous time point  $t-1$ , not relevant to any further past time point ( $t-2, t-3, \dots, 0$ ). Every time the evidences occur, the DBN is re-constructed by the algorithm including six steps (see table [III.1.20](#)) such as initializing DBN, specifying transition weights, (re-)constructing DBN, normalizing weights of

relationships, (re-)defining CPT (s), and probabilistic inference. Six steps are repeated whenever evidences occur. After  $t^{\text{th}}$  iteration, the posterior marginal probability of variables in DBN will approach a certain limit; it means that DBN converge at that time. Of course, DBN is more robust than static network and I also decrease the expense of computation significantly. Moreover the research aims to solve the problem of temporary slip and lucky guess: “learner does (doesn’t) know a particular subject but there is solid evidence convincing that she/he doesn’t (does) understand it; this evidence just reflects a temporary slip (or lucky guess)” by using two additional factors *slip* and *guess* in steps 2 and 3 of the proposed six-step algorithm. Please see sub-section [III.1.4.2](#) for more details about how to use DBN to improve knowledge sub-model.

Additionally, the research also proposes an excellent method to specify prior probabilities inside BN based on maximum likelihood estimation (MLE) for beta function. Specifying prior probabilities in accuracy is also the good way to improve parameters of BN. Please see sub-section [III.1.5](#) for more details about how to specify prior probabilities.

In conclusion, the method used to build up BN at here is the hybrid of efficiency-centric and data-centric method. It takes advantages of both of them when it achieves the high-quality network (in data-centric method) and feasibility (in efficiency-centric).

The successive sub-section [IV.1.2](#) will mention how to exploit Bayesian model in practical assessment.

## **IV.1.2. Assessment of Bayesian network**

Besides providing information about users and deducing new assumptions about them, the ultimate of Bayesian (network user) model is to support the adaptive application like ITS and AEHS (see sub-sections [I.2.2](#), [I.2.3](#)) in order to give user the adaptation effects such as personal learning content, course recommendation, adaptive representation, adaptive navigation in domain space, curriculum sequence, and hint generation. So it is very necessary to assess the overall competence of user in a domain. For example, the level of knowledge user gains must be measured in Bayesian model so that it is possible to answer the question: how much user masters over such knowledge. This process is called the exploitation or assessment of Bayesian model. Firstly (sub-section [IV.1.2.1](#)), we should glance over some approaches for assessment. Secondly (sub-section [IV.1.2.2](#)), there is a future trend of the assessment in [Zebra](#), towards to the Computerized Adaptive Testing (CAT).

### **IV.1.2.1. Overview of assessment approaches**

There are three approaches to perform assessment tasks: *alternate*, *diagnostic* and *decision-theoretic* (Mayo, 2001, pp. 82-86).

**Alternate approach** (Mayo, 2001, pp. 83-85)

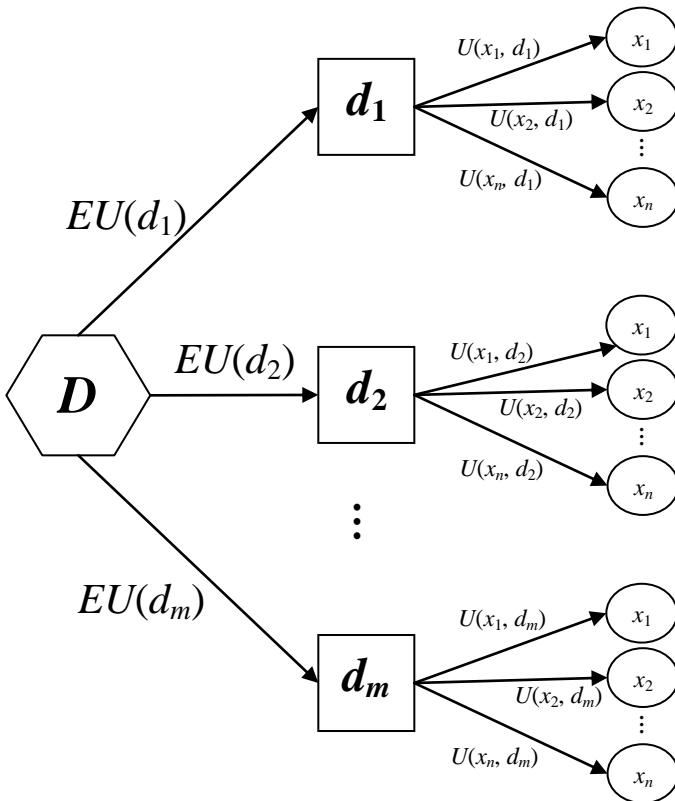
Suppose knowledge domain is decomposed into a set of knowledge elements represented as variables in Bayesian network. In this method, the mastery of a knowledge element is measured by computing the posterior probability of such knowledge element. Thus this probability is used as input to heuristic decision or adaptation rules. It means that adaptive applications will tailor this probability to learning materials via rules in order to provide user suitable learning objects like lectures, exercises, tests, curriculum sequence, and hint generation. For example, if such posterior probability is high, user will receive an advanced exercise. This is the simplest approach.

### **Diagnostic approach** (Mayo, 2001, pp. 85-86)

In this approach Bayesian network includes two kinds of variables: hidden and observed. Hidden variables are knowledge items that need to be assessed how much user masters it. Observed variables so-called evidences are questions, exercises which are used to test user's knowledge. It is possible to imagine that hidden variables represent the diseases and observed variables are symptoms. By surveying symptoms, it is able to diagnose respective diseases. It means that when evidences have been observed, the posterior probabilities of other hidden variables are computed via Bayesian updating. The conditional relationship between a hidden variable and evidence is represented as a directed arc whose direction is always from hidden variable to evidence and the reversed direction is not permitted. A typical example of diagnostic approach is computerized adaptive testing based on item response theory which will be mentioned in next sub-section [IV.1.2.2](#).

### **Decision-theoretic approach** (Mayo, 2001, pp. 42-45)

Given a set of action  $D = \{d_1, d_2, \dots, d_m\}$  and a set of possible outcomes  $X = \{x_1, x_2, \dots, x_n\}$  and a conditional probability distribution  $P(X|D)$ . Of course  $X$  is the outcome of  $D$ . Each combination of values that  $D$  and  $X$  take is defined as a value of *utility function*  $U(X, D)$ . A decision-maker needs to choose an action  $d_i$  so that the expectation of utility function  $EU(X, D)$  is maximized. Note that  $EU(X, D)$  is called expected utility and so the expected utility of an action  $d_i$  is denoted as  $EU(X, d_i)$  or  $EU(d_i)$  in brief. The overview of theory decision can be represented as the decision tree in figure [IV.1.1](#) (Mayo, 2001, p. 43).



**Figure IV.1.1.** Decision-theoretic tree

The decision-theoretic tree depicted in figure IV.1.1 is essentially a Bayesian network. The process of choosing an action is considered as a path from root to leaf in which root is the set of actions and leaf is the respective outcome of a chosen action. The intermediate  $EU(d_i)$  represents the expected utility of an action  $d_i$ . Every outcome  $x_j$  of action  $d_i$  has a conditional probability  $P(X=x_j|D=d_i)$  and a utility value  $U(X=x_j, D=d_i)$ . Of course  $P(X=x_j | D=d_i)$  and  $U(X=x_j, D=d_i)$  are values of conditional probability distribution  $P(X|D)$  and utility function  $U(X, D)$ , respectively. The *expected utility*  $EU(d_i)$  of an action is defined as the probabilistic weighted sum of utility values of all outcomes of such action, as follows:

$$EU(d_i) = \sum_{x_j \in X} P(x_j | d_i) U(x_j, d_i)$$

This formula can be generalized by formula IV.1.1 as follows (Mayo, 2001, p. 44):

$$EU(D) = \sum_X P(X|D) U(X, D)$$

*Formula IV.1.1.* Expected utility of an action

The essential idea of decision theory is to maximize the expected utility  $EU(D)$ . In other words it is to choose the action  $d_i$  that maximizes this expected utility.

If the decision theory is applied into the assessment of Bayesian network in learning context, the utility function will encode educational knowledge related

to the decision made by assigning utility to outcomes when the outcomes can be the errors users make, the grades in their examinations, etc. The actions are user's learning tasks like problem selection, doing exercise, and visiting learning web pages.

In conclusion, the *belief network engine* in the core of Zebra uses the first approach to assess the Bayesian network. It simple to compute the posterior probabilities of knowledge items (variables in network) and match such probability with adaptive rules, for example, if the posterior probability of knowledge item  $I$  learned by user  $A$  is high then, the advanced content of  $I$  is provided to user  $A$  because user  $A$  is mastered over item  $I$ . In the future, **Zebra** aims to apply the Computerized Adaptive Testing (CAT) technique into determining how users master over knowledge items when these items are tests or examinations. CAT based on the Item Response Theory (IRT) is introduced in following sub-section [IV.1.2.2](#).

#### **IV.1.2.2. Towards the Computerized Adaptive Testing**

The computer-based tests have more advantages than the traditional paper-based tests when there is the boom of internet and computer. Computer-based testing allows students to perform the tests at any time and any place and the testing environment becomes more realistic. Moreover, it is very easy to assess students' ability by using the Computerized Adaptive Testing (CAT). The CAT is considered as the branch of computer-based testing but it improves the accuracy of test core when CAT systems try to choose items which are suitable to students' abilities; such items are called adaptive items.

The important problem in CAT is how to estimate students' abilities so as to select the best items for students. There are some methods to solve this problem such as Bayesian approach (Linden & Pashley, 2002, pp. 3-7) but I propose a new method to compute ability estimates by maximization likelihood estimation (MLE). Based on this proposed MLE method for estimating examinee's ability, the new version of CAT algorithm is also invented. Such new version of CAT algorithm is called *advanced CAT algorithm*.

Moreover, I suggest the stopping criterion based on ability error, thus, the process of testing stops if the ability error is approximated to zero. The ability error measures difference of student's ability between two successive testing times. In other words, the process of testing ends only when student's knowledge becomes saturated (she/he cannot do test better or worse) and such knowledge is her/his actual knowledge. This sub-section [IV.1.2.2](#) includes three parts:

- Part [IV.1.2.2.1](#) gives an overview about CAT.
- Part [IV.1.2.2.2](#) describes the proposed MLE method for estimating examinee's ability.
- Part [IV.1.2.2.3](#) describes advanced CAT algorithm in detailed. Ability error is also mentioned in this part.

#### IV.1.2.2.1. Overview of Computerized Adaptive Testing (CAT)

**Item Response Theory** (IRT) is defined as a statistical model in which examinees can be described by a set of ability scores that are predictive. Based on mathematical models, IRT links together examinee's performance on test items, item statistics, and examinee abilities (Rudner, 1998). Note that the term "item" indicates test, exam, question, etc. Users in IRT context are examinees. Examinee's ability is often represented by variable  $\theta$ . Given an examinee and item  $i$ , IRT is modeled as a function of a true ability  $\theta$  of given examinee with three parameters of item  $i$  such as  $a_i$ ,  $b_i$ , and  $c_i$ . This function so-called Item Response Function (IRF) or Item Characteristic Curve (ICC) function computes the probability of a correct response of given examinee to item  $i$ . IRF is specified by formula IV.1.2 as follows:

$$IRF(\theta) = P_i(\theta) = c_i + \frac{1 - c_i}{1 + exp(-a_i(\theta_j - b_i))}$$

Formula IV.1.2. Item Response Function

Where  $exp^{(.)}$  or  $e^{(.)}$  denotes exponent function. Your attention please, IRF is function of examinee's ability and it is essentially the probability of a correct response of a given examinee to an item. Note that  $a_i$ ,  $b_i$ , and  $c_i$  are greater than 0.

IRF, a variant of logistic function, is plotted as the curve in following figure IV.1.2 with  $a_i=6$ ,  $b_i=0.4$ ,  $c_i=0.2$ .

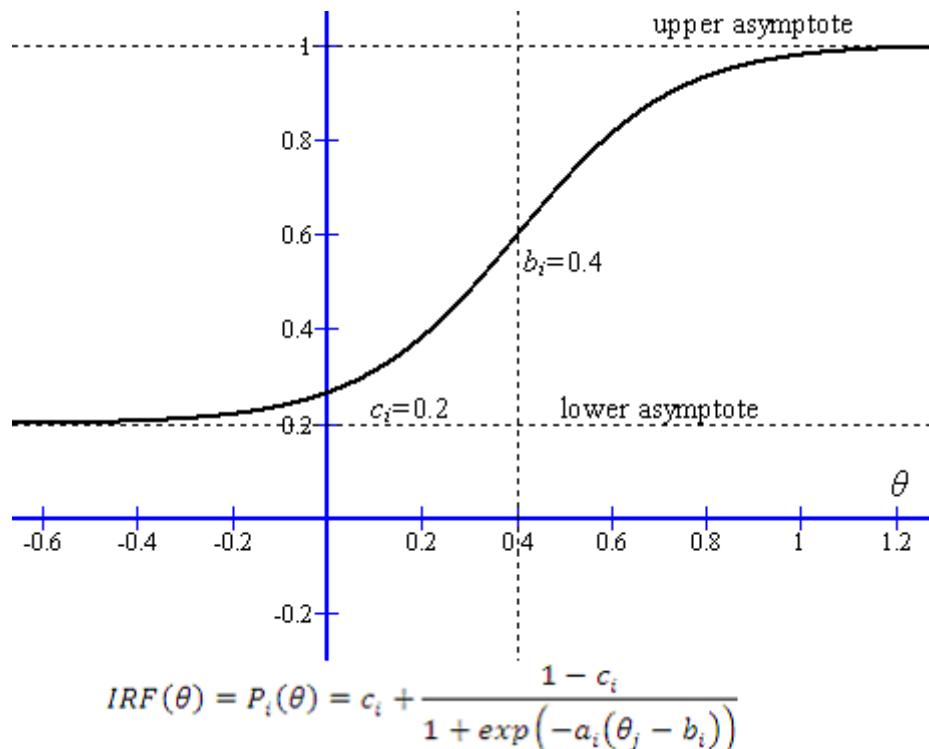


Figure IV.1.2. Item Response Function curve

The horizontal axis  $\theta$  is the scale of examinee's ability (Rudner, 1998). The vertical axis is the probability of correct response to the item specified by three parameters:  $a_i=6$ ,  $b_i=0.4$ ,  $c_i=0.2$ . As seen in figure IV.1.2, the more the IRF shifts right, the more difficult item is. The lower asymptote at  $c_i=0.2$  indicates the probability of correct response for examinee with lowest ability and otherwise for the upper asymptote at 1.

IRF measures examinee's proficiency based on her/his ability and some properties of item. Every item  $i$  has three parameters  $a_i$ ,  $b_i$ ,  $c_i$  which are specified by experts or statistical data.

- The  $a_i$  parameter so-called *discriminatory parameter* (Rudner, 1998) tells how well the item discriminates between examinees whose abilities are not different much. It defines the slope of the curve at the inflection point. The higher is the value of  $a_i$ , the steeper is the curve. In case of steep curve, there is a large difference between the probabilities of a correct response for examinees whose ability is slightly below of the inflection point and examinees whose ability is slightly above the inflection point (Rudner, 1998).
- The  $b_i$  parameter so-called *difficult parameter* (Rudner, 1998) indicates how difficult the item is. It specifies the location of inflection point of the curve along the  $\theta$  axis (examinee's ability). Higher value of  $b_i$  shifts the curve to the right and implicates that the item is more difficult.
- The  $c_i$  parameter so-called *guessing parameter* (Rudner, 1998) indicates that the probability of a correct response to item of low-ability examinees is very close to  $c_i$ . It determines the lower asymptote of the curve. This parameter is called guessing parameter because it is the random probability that low-ability examinees guess a correct response to an item when they do not master the item. The upper asymptote always goes through value 1 because the probability that high-ability examinees give right response to an item is 1 (Rudner, 1998).

In general, IRF is used by to computerized adaptive testing in order to choose the best item which is given to examinee and to estimate examinee' true ability  $\theta$ . Computerized adaptive testing is described right after.

**Computerized Adaptive Testing** (CAT) (Rudner, 1998) is the iterative algorithm which begins providing examinee an (test) item so as to be best to her/his initial ability; after that the ability is estimated again and the process of item suggestion is continued until a stopping criterion is met. This algorithm aims to make a series of (test) items which are evaluated to become chosen items that suitable to examinee's ability. The set of items from which system picks ones up is called as item *pool*. The items chosen and given to examinee compose the adaptive test. CAT includes following steps (Rudner, 1998) as shown in table IV.1.1:

1. The initial ability of examinee must be defined and items in the pool that have not yet been chosen are evaluated. The best one among these items is the most suitable to examinee's current ability estimate. Such best item will be given to examinee in the step 2. IRF is applied into

- evaluating items.
2. The best item is chosen and given to examinee and the examinee responds. Such item is moved from pool to adaptive test.
  3. A new ability estimate of examinee is computed based on responses to all of the chosen items. IRF is applied into computing the ability estimate. It is explained that ability estimate is the estimated value of true ability  $\theta$  of examinee at current time point.
  4. Steps 1 through 3 are repeated until a stopping criterion is met.

**Table IV.1.1.** Computerized adaptive testing (CAT) algorithm

Note that the chosen item is also called the *administered item* and the process of choosing best item is also called the *administration process*. The ability estimate is the value of  $\theta$  which is fit best to the model and reflects current proficiency of examinee in item but it is not imperative to define precisely the initial ability because the final ability estimate may not be closed to initial ability. The stopping criterion could be time, number of administered items, change in ability estimate, maximum-information of ability estimate, content coverage, a precision indicator (standard error), a combination of factors, etc (Rudner, 1998).

In step 1, there is the question: “how to evaluate the items so as to choose the best one”. So each item  $i$  is qualified by the amount of information or entropy at given ability  $\theta$ ; such *information function* is denoted  $I_i(\theta)$ . The best next item is the one that provides the most informative or gets highest value of  $I_i(\theta)$ . Formula IV.1.3 specifies information function for item  $i$  (Rudner, 1998).

$$I_i(\theta) = \frac{(P'_i(\theta))^2}{P_i(\theta)(1 - P_i(\theta))}$$

*Formula IV.1.3.* Information function for item  $i$

Where  $P_i(\theta)$  is the probability of a correct response to item  $i$  and so it is the IRF function specified by previous formula IV.1.2 and  $P'_i(\theta)$  is the first-order derivative of  $P_i(\theta)$ . According to formula IV.1.2, we have:

$$P_i(\theta) = IRF(\theta) = c_i + \frac{1 - c_i}{1 + \exp(-a_i(\theta - b_i))}$$

The information function  $I_i(\theta)$  reflects how much the item  $i$  matches examinee's ability. The item should not be too easy or too difficult. In the step 1 of CAT algorithm, the best item is the one that maximizes the information function  $I_i(\theta)$ . It is easy to find out such best item by brute-force technique that browses all items.

In step 3 of CAT algorithm, it is required to compute the ability estimate. Newton-Raphson method is proposed to find out the ability estimate (Rudner, 1998). Bayesian approach is also the good method for estimating examinee's ability (Nguyen, The Bayesian approach and suggested stopping criterion in Computerized Adaptive Testing, 2013). But I propose a new method to calculate ability estimate based on maximization likelihood estimation (MLE).

The successive part [IV.1.2.2.2](#) describes such MLE method for CAT. Ability error used a stopping criterion of CAT is also mentioned in this part [IV.1.2.2.2](#).

#### *IV.1.2.2.2. Maximum likelihood estimation (MLE) for CAT*

Let  $\hat{\theta}$  be the ability estimate of examinee, the goal of this part [IV.1.2.2.2](#) is to calculate  $\hat{\theta}$ . Recall that the ability estimate is very important to step 3 of CAT algorithm; please see table [IV.1.1](#) for more details about CAT algorithm.

Suppose there are  $N$  items given to an examinee; in other words, the size of item pool is  $N$ . Each item  $i$  has  $q_i$  optional responses. For example, we have  $q_i=4$  when item is question with four possible answers such as  $A, B, C$ , and  $D$ . We have  $q_i=10$  when item is an exam whose resulted grade ranges from 1 to 10. Suppose the number of *correct responses* of given examinee to item  $i$  is  $r_i$ .

$$0 \leq r_i \leq q_i \text{ and } 1 \leq q_i$$

For example, a given examinee do exam  $i$  whose grade ranges from 1 to 10 and she/he gains grade 9 then, we have  $q_i=10$  and  $r_i=9$ . Let  $P(\theta)$  be the cumulative probability of a correct response to given item. Exactly,  $P(\theta)$  is the probability that examinee's ability is less than or equal to  $\theta$ . Note that the probability  $P(\theta)$  is IRF function specified by formula [IV.1.2](#). For convenience, let guessing parameter be zero ( $c=0$ ). It means that the probability that examinee guesses correct response equals 0. Formula [IV.1.4](#) specifies  $P(\theta)$  with  $c=0$ .

$$P(\theta) = IRF(\theta) = \frac{1}{1 + \exp(-a(\theta - b))}$$

#### *Formula IV.1.4. IRF with zero guessing parameter*

Where  $a$  and  $b$  are discriminatory parameter and difficult parameter, respectively. Without loss of generality, formula [IV.1.4](#) implicates that guessing parameter is fixed.

Let  $\theta_0$  be the examinee's initial ability. When examinee's initial ability is not determined yet,  $\theta_0$  can be initialized by zero, as specified by formula [IV.1.5](#).

$$\theta_0 = 0$$

#### *Formula IV.1.5. Examinee's initial ability*

Note that it is possible to initialize  $\theta_0$  by arbitrary number.

According to Bernoulli trial (Montgomery & Runger, 2003, p. 72), the probability that examinee provides  $r_i$  correct responses for given item  $i$  is:

$$(P(\theta_0))^{q_i-r_i} (1 - P(\theta_0))^{r_i}$$

Where probability  $P(\theta_0)$  is specified by formula [IV.1.4](#) and  $\theta_0$  is examinee's initial ability specified by formula [IV.1.5](#).

The likelihood function (Czepiel, 2002, pp. 4-5) of examinee's ability when she/he responses  $N$  items in the pool is specified by formula [IV.1.6](#) as follows:

$$\begin{aligned}
 L(a, b) &= \prod_{i=1}^N (P(\theta_0))^{q_i - r_i} (1 - P(\theta_0))^{r_i} = \prod_{i=1}^N (P(\theta_0))^{q_i} \left( \frac{1 - P(\theta_0)}{P(\theta_0)} \right)^{r_i} \\
 &= \prod_{i=1}^N \left( \frac{1}{1 + \exp(ab - a\theta_0)} \right)^{q_i} (\exp(ab - a\theta_0))^{r_i}
 \end{aligned}$$

*Formula IV.1.6.* Likelihood function of examinee's ability

Note,  $a$  and  $b$  become variables of the likelihood function  $L(a, b)$ .

It is required to estimate the parameters  $a$  and  $b$  so that the likelihood function takes maximum value. Let  $\hat{a}$  and  $\hat{b}$  be parameter estimates of  $a$  and  $b$ , respectively; of course,  $L(\hat{a}, \hat{b})$  is the maximum value of likelihood function  $L(a, b)$ . Thus, this method is called maximum likelihood estimation (MLE) and the goal of MLE is to find out parameter estimates  $\hat{a}$  and  $\hat{b}$ .

$$(\hat{a}, \hat{b}) = \underset{a, b}{\operatorname{argmax}} L(a, b)$$

Please see sub-section [III.1.5.1](#) for more details about MLE. Because it is too difficult to work with the likelihood function in the form of product of condition probabilities, it is necessary to take logarithm of  $L(a, b)$  so as to transform the likelihood function from repeated multiplication into repeated addition. The natural logarithm of  $L(a, b)$  so-called log-likelihood is denoted  $\ln L(a, b)$ . We have:

$$\begin{aligned}
 \ln L(a, b) &= - \sum_{i=1}^N q_i \ln(1 + \exp(ab - a\theta_0)) + \sum_{i=1}^N r_i (ab - a\theta_0) \\
 &= (ab - a\theta_0) \sum_{i=1}^N r_i - \ln(1 + \exp(ab - a\theta_0)) \sum_{i=1}^N q_i
 \end{aligned}$$

Briefly, formula [IV.1.7](#) specifies the log-likelihood  $\ln L(a, b)$ .

$$\ln L(a, b) = (ab - a\theta_0) \sum_{i=1}^N r_i - \ln(1 + \exp(ab - a\theta_0)) \sum_{i=1}^N q_i$$

*Formula IV.1.7.* Log-likelihood function of examinee's ability

Where  $\ln(\cdot)$  denotes natural logarithm function,  $\theta_0$  is examinee's initial ability and  $r_i$  is examinee's response.

Maximizing the likelihood function is equivalent to maximizing  $\ln L(a, b)$ .

$$(\hat{a}, \hat{b}) = \underset{a, b}{\operatorname{argmax}} L(a, b) = \underset{a, b}{\operatorname{argmax}} \ln L(a, b)$$

The maximization can be done by setting first-order partial derivatives of  $\ln L(a, b)$  with respect to parameters  $a$  and  $b$  to 0 and solving these equations to find out parameter estimates  $\hat{a}$  and  $\hat{b}$ . Two first-order partial derivatives of  $\ln L(a, b)$  with respect to parameters  $a$  and  $b$  are:

$$\frac{\partial \ln L(a, b)}{\partial a} = (b - \theta_0) \left( \sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right)$$

$$\frac{\partial \ln L(a, b)}{\partial b} = a \left( \sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right)$$

Setting these first-order partial derivatives to 0, we have following equations for solving estimates  $\hat{a}$  and  $\hat{b}$ .

$$\begin{cases} \frac{\partial \ln L(a, b)}{\partial a} = 0 \\ \frac{\partial \ln L(a, b)}{\partial b} = 0 \end{cases} \Rightarrow \begin{cases} (b - \theta_0) \left( \sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right) = 0 \\ a \left( \sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right) = 0 \end{cases}$$

Given  $a > 0$ , if  $b = \theta_0$  then,

$$\begin{aligned} & a \left( \sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right) = 0 \\ \Rightarrow & \sum_{i=1}^N r_i - \frac{\exp(a\theta_0 - a\theta_0)}{1 + \exp(a\theta_0 - a\theta_0)} \sum_{i=1}^N q_i = \sum_{i=1}^N r_i - \frac{\exp(0)}{1 + \exp(0)} \sum_{i=1}^N q_i = 0 \\ \Rightarrow & \sum_{i=1}^N r_i = \frac{1}{2} \sum_{i=1}^N q_i \end{aligned}$$

Because the condition  $\sum_{i=1}^N r_i = \frac{1}{2} \sum_{i=1}^N q_i$  is not always true in all situations, it is possible to eliminate the case  $b = \theta_0$ . Without loss of generality, we have formula IV.1.8 for finding out two parameter estimates  $\hat{a}$  and  $\hat{b}$  as follows:

$$\sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i = 0$$

*Formula IV.1.8.* Equation for finding out parameter estimates

Because  $b$  is difficult parameter reflecting directly examinee's ability, it is possible to suppose that discriminatory parameter  $a$  is arbitrary. Deriving from formula IV.1.8, we have:

$$\begin{aligned} & \sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i = 0 \\ \Leftrightarrow & \sum_{i=1}^N r_i + \exp(ab - a\theta_0) \sum_{i=1}^N r_i - \exp(ab - a\theta_0) \sum_{i=1}^N q_i = 0 \end{aligned}$$

$$\begin{aligned}
 &\Leftrightarrow \exp(ab - a\theta_0) \left( \sum_{i=1}^N (q_i - r_i) \right) = \sum_{i=1}^N r_i \\
 &\Rightarrow \exp(ab - a\theta_0) = \frac{\sum_{i=1}^N r_i}{\sum_{i=1}^N (q_i - r_i)} \\
 &\Rightarrow ab - a\theta_0 = \ln \left( \sum_{i=1}^N r_i \right) - \ln \left( \sum_{i=1}^N (q_i - r_i) \right) \\
 &\Rightarrow b = \frac{\ln(\sum_{i=1}^N r_i) - \ln(\sum_{i=1}^N (q_i - r_i))}{a} + \theta_0
 \end{aligned}$$

In general, two possible parameter estimates  $\hat{a}$  and  $\hat{b}$  are specified by formula IV.1.9 as follows:

$$\begin{cases} \hat{a} \text{ is arbitrary} \\ \hat{b} = \frac{\ln(\sum_{i=1}^N r_i) - \ln(\sum_{i=1}^N (q_i - r_i))}{\hat{a}} + \theta_0 \\ \hat{a} > 0 \text{ and } \hat{b} \geq 0 \end{cases}$$

#### Formula IV.1.9. Discriminatory and difficult estimates

Where  $q_i$  is the number of possible responses of item  $i$ ,  $r_i$  is the correct response of examinee to item  $i$  and  $\theta_0$  is the examinee's initial ability. There is a convention that if examinee responds correctly all items,  $\sum_{i=1}^N r_i = \sum_{i=1}^N q_i$  then,  $\hat{b}$  gains maximum value and you can define the maximum value by positive infinity ( $+\infty$ ) or any pre-defined very large number.

The probability  $P(\theta)$  (IRF function) specified by formula IV.1.4 is essentially cumulative distribution function (CDF). Please see sub-section III.1.1.1 for more details about cumulative distribution function and probability density function. Let  $p(\theta)$  be the probability density function of ability  $\theta$ , we have:

$$p(\theta) = P'(\theta) = \frac{(a)\exp(-a(\theta - b))}{(1 + \exp(-a(\theta - b)))^2}$$

#### Formula IV.1.10. Probability density function of examinee's ability

Formula IV.1.10 indicates that the probability density function of examinee's ability is the first-order derivative of IRF function. Substituting parameter estimates  $\hat{a}$  and  $\hat{b}$  specified by formula IV.1.9 into formula IV.1.10, we get the optimal density function  $\hat{p}(\theta)$  of examinee's ability, specified by formula IV.1.11.

$$\hat{p}(\theta) = \frac{(\hat{a})\exp(-\hat{a}(\theta - \hat{b}))}{(1 + \exp(-\hat{a}(\theta - \hat{b})))^2}$$

#### Formula IV.1.11. Optimal probability density function of examinee's ability

The ability estimate  $\hat{\theta}$  is the expectation of  $\theta$  given optimal density function  $\hat{p}(\theta)$ . We have:

$$\begin{aligned}\hat{\theta} &= E(\theta|\hat{p}(\theta)) = \int_{-\infty}^{+\infty} \theta \hat{p}(\theta) d\theta = \int_{-\infty}^{+\infty} \theta \frac{(\hat{a}) \exp(-\hat{a}(\theta - \hat{b}))}{\left(1 + \exp(-\hat{a}(\theta - \hat{b}))\right)^2} d\theta \\ &= \int_{-\infty}^{+\infty} \theta d\left(\frac{1}{1 + \exp(-\hat{a}(\theta - \hat{b}))}\right) \\ &= \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \Big|_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} \frac{1}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta\end{aligned}$$

Let  $A = \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \Big|_{-\infty}^{+\infty}$  and  $B = \int_{-\infty}^{+\infty} \frac{1}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta$ , we have:

$$\begin{aligned}A &= \lim_{\theta \rightarrow +\infty} \left( \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) - \lim_{\theta \rightarrow -\infty} \left( \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) \\ &= \lim_{\theta \rightarrow +\infty} \left( \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) - \lim_{\theta \rightarrow -\infty} \left( \frac{d(\theta)}{d(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))} \right)\end{aligned}$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}&= \lim_{\theta \rightarrow +\infty} \left( \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) - \lim_{\theta \rightarrow -\infty} \left( \frac{1}{(-\hat{a}) \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) \\ &= \lim_{\theta \rightarrow +\infty} \left( \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right)\end{aligned}$$

$$\begin{aligned}B &= \int_{-\infty}^{+\infty} \frac{1}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta = \int_{-\infty}^{+\infty} d\left(\theta + \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) \\ &= \left( \theta + \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \Big|_{-\infty}^{+\infty} \\ &= \lim_{\theta \rightarrow +\infty} \theta + \lim_{\theta \rightarrow +\infty} \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) - \lim_{\theta \rightarrow -\infty} \theta \\ &\quad - \lim_{\theta \rightarrow -\infty} \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \\ &= \lim_{\theta \rightarrow +\infty} \theta - \lim_{\theta \rightarrow -\infty} \theta - \lim_{\theta \rightarrow -\infty} \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\end{aligned}$$

We have:

$$\begin{aligned}
 \hat{\theta} &= E(\theta | \hat{p}(\theta)) = A - B \\
 &= \lim_{\theta \rightarrow +\infty} \left( \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) - \lim_{\theta \rightarrow +\infty} \theta + \lim_{\theta \rightarrow -\infty} \theta \\
 &\quad + \lim_{\theta \rightarrow -\infty} \frac{1}{\hat{a}} \ln \left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right) \\
 &= \lim_{\theta \rightarrow +\infty} \left( -\frac{(\theta)\exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) + \lim_{\theta \rightarrow -\infty} \left( \frac{1}{\hat{a}} \ln \left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right) - (-\theta) \right) \\
 &= \lim_{\theta \rightarrow +\infty} \left( -\frac{\theta}{\frac{1}{\exp(\hat{a}\hat{b} - \hat{a}\theta)} + 1} \right) \\
 &\quad + \lim_{\theta \rightarrow -\infty} \ln \left( \frac{\exp \left( \frac{1}{\hat{a}} \ln \left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right) \right)}{\exp(-\theta)} \right) \\
 &= \lim_{\theta \rightarrow +\infty} \left( -\frac{d(\theta)}{d \left( \frac{1}{\exp(\hat{a}\hat{b} - \hat{a}\theta)} + 1 \right)} \right) \\
 &\quad + \lim_{\theta \rightarrow -\infty} \ln \left( \frac{\exp \left( \frac{1}{\hat{a}} \ln \left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right) \right)}{\exp(-\theta)} \right)
 \end{aligned}$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}
 &= \lim_{\theta \rightarrow +\infty} \left( \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\hat{a}} \right) + \ln \left( \lim_{\theta \rightarrow -\infty} \frac{\left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right)^{\frac{1}{\hat{a}}}}{\exp(-\theta)} \right) \\
 &= \ln \left( \lim_{\theta \rightarrow -\infty} \frac{\left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right)^{\frac{1}{\hat{a}}}}{\exp(-\theta)} \right) \\
 &= \ln \left( \lim_{\theta \rightarrow -\infty} \left( \frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(-\hat{a}\theta)} \right)^{\frac{1}{\hat{a}}} \right) = \ln \left( \left( \lim_{\theta \rightarrow -\infty} \frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(-\hat{a}\theta)} \right)^{\frac{1}{\hat{a}}} \right) \\
 &= \ln \left( \lim_{\theta \rightarrow -\infty} \left( \frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(-\hat{a}\theta)} \right)^{\frac{1}{\hat{a}}} \right)
 \end{aligned}$$

$$= \ln \left( \left( \lim_{\theta \rightarrow -\infty} \frac{d(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))}{d(\exp(-\hat{a}\theta))} \right)^{\frac{1}{\hat{a}}} \right)$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$= \ln \left( \left( \lim_{\theta \rightarrow -\infty} \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(-\hat{a}\theta)} \right)^{\frac{1}{\hat{a}}} \right)$$

$$= \ln \left( \left( \lim_{\theta \rightarrow -\infty} \exp(\hat{a}\hat{b}) \right)^{\frac{1}{\hat{a}}} \right) = \ln \left( (\exp(\hat{a}\hat{b}))^{\frac{1}{\hat{a}}} \right) = \ln(\exp(\hat{b})) = \hat{b}$$

In general, we have a very important result in which the ability estimate  $\hat{\theta}$  equals the difficult parameter estimate  $\hat{b}$ . Formula IV.1.12 represents this result, as follows:

$$\hat{\theta} = \hat{b} = \frac{\ln(\sum_{i=1}^N r_i) - \ln(\sum_{i=1}^N (q_i - r_i))}{\hat{a}} + \theta_0$$

*Formula IV.1.12.* Examinee's ability estimate

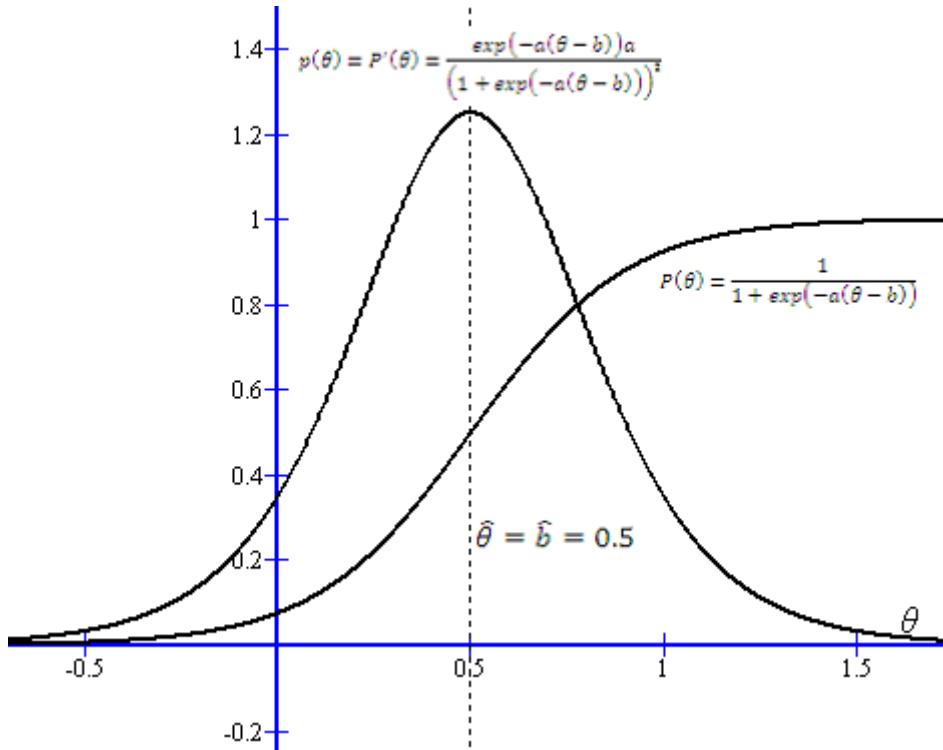
Where  $q_i$  is the number of possible responses of item  $i$ ,  $r_i$  is the correct response of examinee to item  $i$  and  $\theta_0$  is the examinee's initial ability. Recall that if examinee responds correctly all items,  $\sum_{i=1}^N r_i = \sum_{i=1}^N q_i$  then,  $\hat{\theta}$  gains maximum value and you can define the maximum value by positive infinity ( $+\infty$ ) or any pre-defined very large number.

Additionally, discriminatory parameter estimate  $\hat{a}$  is arbitrary; in practice,  $\hat{a}$  can be assigned by fixed discriminatory parameter  $a$  or by the average over all discriminatory parameters of existing items. For example, let  $a_1, a_2, \dots, a_n$  be discriminatory parameters of  $n$  existing items then, the estimate  $\hat{a}$  can be assigned by mean of such  $a_i$  (s).

$$\hat{a} = \frac{a_1 + a_2 + \dots + a_n}{n}$$

When parameter estimate  $\hat{a}$  was determined, it is easy to perform step 3 of the advanced CAT algorithm as shown in table IV.1.2. The formula IV.1.9 implicates that each examinee is modeled as an *item*; please pay attention to this interesting thing because it is the core of the advanced CAT algorithm for multi-user test.

Figure IV.1.3 is an example of IRF function  $P(\theta)$  specified by formula IV.1.4, density function of examinee's ability specified by formula IV.1.9 together with ability  $\hat{\theta} = \hat{b} = 0.5$  given discriminatory parameter  $a=5$ .



**Figure IV.1.3.** IRF function and density function of examinee's ability together with ability estimate (0.5) given discriminatory parameter  $a=5$

Based on this proposed MLE method for estimating examinee's ability (formula IV.1.11), the new version of CAT algorithm is proposed in successive part IV.1.2.2.3.

#### IV.1.2.2.3. New version of CAT algorithm based on MLE

As aforementioned in formula IV.1.12, examinee's ability estimate  $\hat{\theta}$  is equal to difficult parameter estimate  $\hat{b}$  when  $\hat{\theta}$  is essential ability mean given probability density function specified by formula IV.1.11. There is a demand of how to specify ability variance of examinee. The new version of CAT algorithm, so-called *advanced CAT algorithm*, is based on such ability variance. Let  $Var(\hat{\theta})$  be the ability variance, we have:

$$Var(\hat{\theta}) = \sigma_{\hat{\theta}}^2 = E((\theta - \hat{\theta})^2 | \hat{p}(\theta)) = E(\theta^2 | \hat{p}(\theta)) - \hat{\theta}^2 = E(\theta^2 | \hat{p}(\theta)) - \hat{b}^2$$

*Formula IV.1.13.* Examinee's ability variance

Where  $\hat{p}(\theta)$  is optimal density function of examinee's ability, specified by formula IV.1.11.

$$\hat{p}(\theta) = \frac{(\hat{a}) \exp(-\hat{a}(\theta - \hat{b}))}{(1 + \exp(-\hat{a}(\theta - \hat{b})))^2}$$

It is very easy to infer from formula IV.1.13 that the square root of  $Var(\hat{\theta})$  is sample standard error according to study of statistics (Montgomery & Runger,

2003, p. 225). The variance  $Var(\hat{\theta})$  is totally determined by calculation of  $E(\theta^2|\hat{p}(\theta))$ . We have:

$$E(\theta^2|\hat{p}(\theta)) = \int_{-\infty}^{+\infty} \theta^2 \hat{p}(\theta) d\theta = \int_{-\infty}^{+\infty} \theta^2 \frac{(\hat{a}) \exp(-\hat{a}(\theta - \hat{b}))}{(1 + \exp(-\hat{a}(\theta - \hat{b})))^2} d\theta$$

We have

$$\begin{aligned} & \int \theta^2 \frac{(\hat{a}) \exp(-\hat{a}(\theta - \hat{b}))}{(1 + \exp(-\hat{a}(\theta - \hat{b})))^2} d\theta \\ &= \int \theta^2 d\left(\frac{1}{1 + \exp(-\hat{a}(\theta - \hat{b}))}\right) \\ &= \frac{\theta^2}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \int_{-\infty}^{+\infty} \frac{2\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta \end{aligned}$$

We have

$$\begin{aligned} & \int \frac{2\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta = \int 2\theta d\left(\theta + \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) \\ &= \left(2\theta^2 + \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) \\ &\quad - 2 \int \left(\theta + \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) d\theta \\ &= \left(2\theta^2 + \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) - \theta^2 \\ &\quad - \frac{2}{\hat{a}} \int \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) d\theta \\ &= \theta^2 + \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) - \frac{2}{\hat{a}^2} \int \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) d\theta \end{aligned}$$

Given  $\hat{a} > 0$  and let  $x = \hat{a}\hat{b} - \hat{a}\theta$ , we have:

$$d\theta = -\frac{dx}{\hat{a}}$$

And

$$\frac{2}{\hat{a}} \int \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) d\theta = -\frac{2}{\hat{a}^2} \int \ln(1 + \exp(x)) dx$$

$$\begin{aligned}
 &= -\frac{2}{\hat{\alpha}^2} \int \left( \sum_{k=1}^{+\infty} \frac{(-1)^{k+1}}{k} (\exp(x))^k \right) dx = \frac{2}{\hat{\alpha}^2} \int \left( \sum_{k=1}^{+\infty} \frac{(-\exp(x))^k}{k} \right) dx \\
 &= \frac{2}{\hat{\alpha}^2} \sum_{k=1}^{+\infty} \int \frac{(-\exp(x))^k}{k} dx = \frac{2}{\hat{\alpha}^2} \sum_{k=1}^{+\infty} \frac{(-\exp(x))^k}{k^2} \\
 &= \frac{2}{\hat{\alpha}^2} \text{Li}_2(-\exp(x)) = \frac{2}{\hat{\alpha}^2} \text{Li}_2(-\exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta))
 \end{aligned}$$

Where  $\text{Li}_2(x)$  is dilogarithm function (Wikipedia, Polylogarithm, 2014). Formula IV.1.14 expresses dilogarithm function.

$$\begin{aligned}
 \text{Li}_2(x) &= \sum_{k=1}^{+\infty} \frac{x^k}{k^2} = - \int_0^x \frac{\ln(1-t)}{t} dt = \frac{\pi^2}{6} - \int_1^x \frac{\ln(1-t)}{t} dt \\
 &= - \int_0^1 \frac{\ln(1-xt)}{t} dt
 \end{aligned}$$

*Formula IV.1.14. Dilogarithm function*

Where  $\text{Li}_2(0) = 0$ .

You can also find out formula IV.1.14 in (WolframAlpha) and (Weisstein, Dilogarithm). We have

$$\begin{aligned}
 &\int \frac{2\theta}{1 + \exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta)} d\theta \\
 &= \theta^2 + \frac{2\theta}{\hat{\alpha}} \ln(1 + \exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta)) - \frac{2}{\hat{\alpha}^2} \text{Li}_2(-\exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta))
 \end{aligned}$$

It is easy to infer that

$$\begin{aligned}
 &\int \theta^2 \frac{(\hat{\alpha}) \exp(-\hat{\alpha}(\theta - \hat{b}))}{\left(1 + \exp(-\hat{\alpha}(\theta - \hat{b}))\right)^2} d\theta \\
 &= \frac{\theta^2}{1 + \exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta)} - \theta^2 - \frac{2\theta}{\hat{\alpha}} \ln(1 + \exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta)) \\
 &\quad + \frac{2}{\hat{\alpha}^2} \text{Li}_2(-\exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta)) \\
 &= -\frac{\theta^2 \exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta)}{1 + \exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta)} - \frac{2\theta}{\hat{\alpha}} \ln(1 + \exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta)) \\
 &\quad + \frac{2}{\hat{\alpha}^2} \text{Li}_2(-\exp(\hat{\alpha}\hat{b} - \hat{\alpha}\theta))
 \end{aligned}$$

It implies that

$$E(\theta^2 | \hat{p}(\theta)) = \int_{-\infty}^{+\infty} \theta^2 \hat{p}(\theta) d\theta$$

$$\begin{aligned}
 &= \lim_{\theta \rightarrow +\infty} \left( -\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\
 &\quad \left. + \frac{2}{\hat{a}^2} \text{Li}_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\
 &- \lim_{\theta \rightarrow -\infty} \left( -\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\
 &\quad \left. + \frac{2}{\hat{a}^2} \text{Li}_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \right)
 \end{aligned}$$

*Formula IV.1.15.* Expectation of square of examinee's ability

Formula IV.1.15 expresses how to calculate the expectation  $E(\theta^2 | \hat{p}(\theta))$  used to determine ability variance  $\text{Var}(\hat{\theta})$ . It is easy to get the formula similar to formula IV.1.15 for calculating  $E(\theta^2 | \hat{p}(\theta))$  by using the mathematics engine (WolframAlpha). According to formula IV.1.15, it is necessary to calculate limits of expressions  $\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}$ ,  $\frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))$ , and  $\frac{2}{\hat{a}^2} \text{Li}_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta))$  as  $\theta$  approaches  $+\infty$  and  $-\infty$ . We have:

$$\begin{aligned}
 \lim_{\theta \rightarrow +\infty} \frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} &= \frac{\lim_{\theta \rightarrow +\infty} (\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta))}{\lim_{\theta \rightarrow +\infty} (1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))} \\
 &= \lim_{\theta \rightarrow +\infty} (\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)) = \lim_{\theta \rightarrow +\infty} \frac{\theta^2}{\exp(\hat{a}\theta - \hat{a}\hat{b})} \\
 &= \lim_{\theta \rightarrow +\infty} \frac{d(\theta^2)}{d(\exp(\hat{a}\theta - \hat{a}\hat{b}))} = \lim_{\theta \rightarrow +\infty} \frac{2\theta}{(\hat{a})\exp(\hat{a}\theta - \hat{a}\hat{b})}
 \end{aligned}$$

(by using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$\lim_{\theta \rightarrow +\infty} \frac{d(2\theta)}{d((\hat{a})\exp(\hat{a}\theta - \hat{a}\hat{b}))} = \lim_{\theta \rightarrow +\infty} \frac{2}{\hat{a}^2 \exp(\hat{a}\theta - \hat{a}\hat{b})} = 0$$

We have

$$\begin{aligned}
 &\lim_{\theta \rightarrow +\infty} \left( \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\
 &= \frac{2}{\hat{a}} \lim_{\theta \rightarrow +\infty} \frac{\ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))}{\frac{1}{\theta}} = \frac{2}{\hat{a}} \lim_{\theta \rightarrow +\infty} \frac{d(\ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)))}{d(\frac{1}{\theta})}
 \end{aligned}$$

(by using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}
 & \frac{(\hat{a})\exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} = 2 \lim_{\theta \rightarrow +\infty} \frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \\
 & = 2 \frac{\lim_{\theta \rightarrow +\infty} (\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta))}{\lim_{\theta \rightarrow +\infty} (1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))} = 2 \lim_{\theta \rightarrow +\infty} (\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)) \\
 & = 2 \lim_{\theta \rightarrow +\infty} \frac{\theta^2}{\exp(-\hat{a}\hat{b} + \hat{a}\theta)} = 2 \lim_{\theta \rightarrow +\infty} \frac{d(\theta^2)}{d(\exp(-\hat{a}\hat{b} + \hat{a}\theta))}
 \end{aligned}$$

(by using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$= 2 \lim_{\theta \rightarrow +\infty} \frac{2\theta}{(\hat{a})\exp(-\hat{a}\hat{b} + \hat{a}\theta)} = 2 \lim_{\theta \rightarrow +\infty} \frac{d(2\theta)}{d((\hat{a})\exp(-\hat{a}\hat{b} + \hat{a}\theta))}$$

(by using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$= 2 \lim_{\theta \rightarrow +\infty} \frac{2}{\hat{a}^2 \exp(-\hat{a}\hat{b} + \hat{a}\theta)} = 0$$

We have

$$\begin{aligned}
 \lim_{\theta \rightarrow +\infty} \frac{2}{\hat{a}^2} Li_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) &= \frac{2}{\hat{a}^2} Li_2\left(\lim_{\theta \rightarrow +\infty} (-\exp(\hat{a}\hat{b} - \hat{a}\theta))\right) \\
 &= \frac{2}{\hat{a}^2} Li_2(0) = 0
 \end{aligned}$$

(due to  $Li_2(0) = 0$ )

It implies

$$\begin{aligned}
 & \lim_{\theta \rightarrow +\infty} \left( -\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\
 & \quad \left. + \frac{2}{\hat{a}^2} Li_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\
 &= -0 - 0 + 0 = 0
 \end{aligned}$$

We have

$$\begin{aligned}
 & \lim_{\theta \rightarrow -\infty} \frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \\
 &= \lim_{\theta \rightarrow -\infty} \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\frac{1}{\theta^2} + \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\theta^2}} = \frac{\lim_{\theta \rightarrow -\infty} \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\lim_{\theta \rightarrow -\infty} \left( \frac{1}{\theta^2} + \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\theta^2} \right)} \\
 &= \frac{\lim_{\theta \rightarrow -\infty} \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\lim_{\theta \rightarrow -\infty} \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\theta^2}} = \lim_{\theta \rightarrow -\infty} \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(\hat{a}\hat{b} - \hat{a}\theta)} = \lim_{\theta \rightarrow -\infty} \theta^2
 \end{aligned}$$

We have

$$\begin{aligned}
 & \lim_{\theta \rightarrow -\infty} \left( \frac{2\theta}{\hat{a}} \ln \left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right) \right) \\
 &= \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} \ln \left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right) - \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} (\hat{a}\hat{b} - \hat{a}\theta) \\
 &\quad + \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} (\hat{a}\hat{b} - \hat{a}\theta) \\
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} \left( \ln \left( 1 + \exp(\hat{a}\hat{b} - \hat{a}\theta) \right) - (\hat{a}\hat{b} - \hat{a}\theta) \right) \\
 &\quad + \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} (\hat{a}\hat{b} - \hat{a}\theta) \\
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} \ln \left( \frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) + \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \\
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left( \lim_{\theta \rightarrow -\infty} \ln \left( \frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left( \ln \left( \lim_{\theta \rightarrow -\infty} \left( \frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) \right) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left( \ln \left( \lim_{\theta \rightarrow -\infty} \left( \frac{d(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))}{d(\exp(\hat{a}\hat{b} - \hat{a}\theta))} \right) \right) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right)
 \end{aligned}$$

(by using L'Hôpital's rule by taking derivatives of both numerator and denominator (Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left( \ln \left( \lim_{\theta \rightarrow -\infty} \left( \frac{-(\hat{a}) \exp(\hat{a}\hat{b} - \hat{a}\theta)}{-(\hat{a}) \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) \right) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left( \ln(1) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) = \lim_{\theta \rightarrow -\infty} \left( \frac{2\theta}{\hat{a}} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\
 &= -2 \lim_{\theta \rightarrow -\infty} \theta^2 + 2\hat{b} \lim_{\theta \rightarrow -\infty} \theta
 \end{aligned}$$

According to (Wikipedia, Polygamma function, 2014), given  $x < 0$  following formula IV.1.16 is an *inversion property* of dilogarithm:

$$Li_2(x) = -Li_2\left(\frac{1}{x}\right) - \frac{(ln(-x))^2}{2} - \frac{\pi^2}{6}$$

*Formula IV.1.16.* Inversion property of dilogarithm

It implies

$$Li_2(-exp(\hat{a}\hat{b} - \hat{a}\theta)) = -Li_2(-exp(\hat{a}\theta - \hat{a}\hat{b})) - \frac{(\hat{a}\hat{b} - \hat{a}\theta)^2}{2} - \frac{\pi^2}{6}$$

We have

$$\begin{aligned} \lim_{\theta \rightarrow -\infty} \frac{2}{\hat{a}^2} Li_2(-exp(\hat{a}\hat{b} - \hat{a}\theta)) &= \frac{2}{\hat{a}^2} \lim_{\theta \rightarrow -\infty} Li_2(-exp(\hat{a}\hat{b} - \hat{a}\theta)) \\ &= -\frac{2}{\hat{a}^2} \left( \lim_{\theta \rightarrow -\infty} Li_2(-exp(\hat{a}\theta - \hat{a}\hat{b})) + \lim_{\theta \rightarrow -\infty} \frac{(\hat{a}\hat{b} - \hat{a}\theta)^2}{2} + \frac{\pi^2}{6} \right) \\ &= -\frac{2}{\hat{a}^2} \left( Li_2 \left( \lim_{\theta \rightarrow -\infty} (-exp(\hat{a}\theta - \hat{a}\hat{b})) \right) + \lim_{\theta \rightarrow -\infty} \frac{\hat{a}^2(\hat{b} - \theta)^2}{2} + \frac{\pi^2}{6} \right) \\ &= -\frac{2}{\hat{a}^2} \left( Li_2(0) + \lim_{\theta \rightarrow -\infty} \frac{\hat{a}^2(\hat{b} - \theta)^2}{2} + \frac{\pi^2}{6} \right) = -\frac{2}{\hat{a}^2} \left( \lim_{\theta \rightarrow -\infty} \frac{\hat{a}^2(\hat{b} - \theta)^2}{2} + \frac{\pi^2}{6} \right) \\ &\quad (\text{due to } Li_2(0) = 0) \\ &= -\hat{b}^2 + \lim_{\theta \rightarrow -\infty} 2\hat{b}\theta - \lim_{\theta \rightarrow -\infty} \theta^2 - \frac{\pi^2}{3\hat{a}^2} \end{aligned}$$

We have

$$\begin{aligned} \lim_{\theta \rightarrow -\infty} &\left( -\frac{\theta^2 exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} ln(1 + exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\ &\quad \left. + \frac{2}{\hat{a}^2} Li_2(-exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\ &= -\lim_{\theta \rightarrow +\infty} \theta^2 + 2 \lim_{\theta \rightarrow -\infty} \theta^2 - 2\hat{b} \lim_{\theta \rightarrow -\infty} \theta - \hat{b}^2 + \lim_{\theta \rightarrow -\infty} 2\hat{b}\theta - \lim_{\theta \rightarrow -\infty} \theta^2 - \frac{\pi^2}{3\hat{a}^2} \\ &= -\hat{b}^2 - \frac{\pi^2}{3\hat{a}^2} \end{aligned}$$

According to formula IV.1.15 for calculating the expectation  $E(\theta^2 | \hat{p}(\theta))$ , we have

$$\begin{aligned} E(\theta^2 | \hat{p}(\theta)) &= \int_{-\infty}^{+\infty} \theta^2 \hat{p}(\theta) d\theta \\ &= \lim_{\theta \rightarrow +\infty} \left( -\frac{\theta^2 exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} ln(1 + exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\ &\quad \left. + \frac{2}{\hat{a}^2} Li_2(-exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \end{aligned}$$

$$\begin{aligned}
 & -\lim_{\theta \rightarrow -\infty} \left( -\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\
 & \quad \left. + \frac{2}{\hat{a}^2} \text{Li}_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\
 & = 0 - \left( -\hat{b}^2 - \frac{\pi^2}{3\hat{a}^2} \right) = \frac{\pi^2}{3\hat{a}^2} + \hat{b}^2
 \end{aligned}$$

According to formula IV.1.13 for calculating the ability variance  $\text{Var}(\hat{\theta})$ , we have

$$\text{Var}(\hat{\theta}) = \sigma_{\hat{\theta}}^2 = E(\theta^2 | \hat{p}(\theta)) - \hat{b}^2 = \frac{\pi^2}{3\hat{a}^2} + \hat{b}^2 - \hat{b}^2 = \frac{\pi^2}{3\hat{a}^2}$$

Briefly, formula IV.1.17 indicates how to compute the variance of examinee's ability estimate. Please pay attention the important result of formula IV.1.17 because it is used to compute estimate of discriminatory parameter.

$$\text{Var}(\hat{\theta}) = \sigma_{\hat{\theta}}^2 = \frac{\pi^2}{3\hat{a}^2}$$

*Formula IV.1.17.* Examinee's explicit ability variance

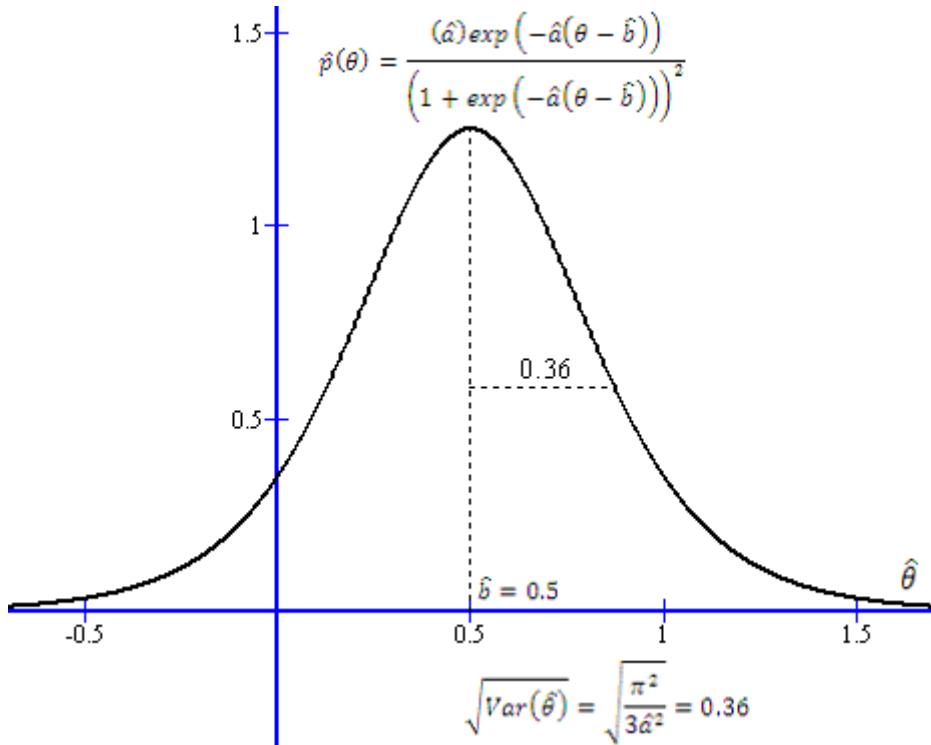
The standard deviation of  $\hat{\theta}$  that is square root of  $\text{Var}(\hat{\theta})$  is:

$$\sigma_{\hat{\theta}} = \sqrt{\text{Var}(\hat{\theta})} = \sqrt{\frac{\pi^2}{3\hat{a}^2}}$$

For example, given  $\hat{a} = 5$  and  $\hat{b} = 0.5$ , by applying formula IV.1.17, we have:

$$\sqrt{\text{Var}(\hat{\theta})} = \sqrt{\frac{\pi^2}{3\hat{a}^2}} = 0.36$$

Figure shows the example with  $\hat{a} = 5$ ,  $\hat{b} = 0.5$ , and standard deviation  $\sqrt{\text{Var}(\hat{\theta})} = 0.36$  with note that standard deviation is square root of variance and optimal density function  $\hat{p}(\theta)$  is specified by formula IV.1.11.



**Figure IV.1.4.** An example of examinee's ability variance

Suppose there are  $k$  examinees  $u_1, u_2, \dots, u_k$  who have  $k$  ability estimates  $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$  after they have done a number of test items. Let  $\bar{\hat{\theta}}$  be statistical sample mean (Montgomery & Runger, 2003, p. 190) of examinees' ability estimates, formula IV.1.18 specifies the mean  $\bar{\hat{\theta}}$ .

$$\bar{\hat{\theta}} = \frac{1}{k} \sum_{i=1}^k \hat{\theta}_i$$

*Formula IV.1.18.* Examinee's statistical ability mean

The ability variance  $Var(\hat{\theta})$  is now considered as statistical sample variance (Montgomery & Runger, 2003, p. 191), which is calculated by formula IV.1.19 as follows:

$$Var(\hat{\theta}) = \frac{1}{k-1} \sum_{i=1}^k (\hat{\theta}_i - \bar{\hat{\theta}})^2$$

*Formula IV.1.19.* Examinee's ability variance as statistical sample variance

Where  $\bar{\hat{\theta}}$  is the ability mean specified by formula IV.1.18.

Formula IV.1.19 shares the same purpose with the formula IV.1.17 but their approaches are different. The variance  $Var(\hat{\theta})$  specified by formula IV.1.17 is essentially the *local variance* focusing on individual examinee. The variance  $Var(\hat{\theta})$  specified by formula IV.1.19 is essentially the *global variance* across  $k$  examinees. The reason of using the same notation  $Var(\hat{\theta})$  for both local

variance and global variance is explained right later when the discriminatory parameter  $a$  (see previous part IV.1.2.2.2) is computed based on these ability variances.

As aforementioned in previous part IV.1.2.2.2, at the step 1 of CAT algorithm shown table IV.1.1, the best item is the one that maximizes the information function  $I_i(\theta)$  specified by formula IV.1.3. Each test item  $i$  has individual discriminatory parameter  $a_i$ , difficult parameter  $b_i$ , and guessing parameter  $c_i$ . If there are  $k$  examinees in multi-user test, it is necessary to choose items so that it is easy to discriminate examinees according to their ability estimates. In other words, such items allow system to classify examinees into distinguishable groups according to examinees' abilities. These items are called *discriminatory items*. Let  $a^*$  be the so-called *discriminatory estimate*, it is easy to recognize that  $a^*$  is the global estimated value of discriminatory parameter. In formal, discriminatory item is defined as the one whose discriminatory parameter  $a_i$  is approximated to  $a^*$ . According to formula IV.1.17 we have.

$$Var(\hat{\theta}) = \frac{\pi^2}{3(a^*)^2} \Rightarrow a^* = \frac{\pi}{\sqrt{3Var(\hat{\theta})}}$$

Applying formula IV.1.19 into determining  $Var(\hat{\theta})$ , we have:

$$a^* = \frac{\pi}{\sqrt{3Var(\hat{\theta})}} = \frac{\pi}{\sqrt{\frac{3}{k-1} \sum_{i=1}^k (\hat{\theta}_i - \bar{\hat{\theta}})^2}} = \sqrt{\frac{(k-1)\pi^2}{3 \sum_{i=1}^k (\hat{\theta}_i - \bar{\hat{\theta}})^2}}$$

Briefly, formula IV.1.20 specifies discriminatory estimate.

$$a^* = \sqrt{\frac{(k-1)\pi^2}{3 \left( \sum_{i=1}^k (\hat{\theta}_i - \bar{\hat{\theta}})^2 \right)}}$$

*Formula IV.1.20. Discriminatory estimate*

Where  $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$  are  $k$  ability estimates of  $k$  examinees  $u_1, u_2, \dots, u_k$  and  $\bar{\hat{\theta}}$  is the ability mean specified by formula IV.1.18 with assumption that there are  $k$  examinees  $u_1, u_2, \dots, u_k$  who have  $k$  ability estimates  $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$ . Based on the discriminatory estimate, the new version of CAT algorithm for multi-user test is described in table IV.1.2. Such new CAT algorithm is called *advanced CAT algorithm*.

1. Suppose there are  $k$  examinees  $u_1, u_2, \dots, u_k$  who have  $k$  ability estimates  $\theta_1, \theta_2, \dots, \theta_k$  after they have finished a number of items in previous test. Suppose there are  $n$  items in the test pool and each item  $i$  has individual parameters such as  $a_i, b_i$ , and  $c_i$ . Items available from test pool must be evaluated. Let  $\bar{\theta}$  be ability mean of such  $k$  examinees.

$$\bar{\theta} = \frac{\theta_1 + \theta_2 + \dots + \theta_k}{k}$$

Let  $a^*$  be discriminatory estimate that is calculated at step 3 in previous test. If  $a^*$  is not determined, it is initialized by the mean of discriminatory parameters  $a_i$  (s).

$$a^*(initial) = \frac{a_1 + a_2 + \dots + a_n}{n}$$

Following code is evaluation process to chose the best items recommended to examinees.

For each item  $i$  not recommended to examinee  $u_j$  yet

Let

$$\left| b_i - \frac{\bar{\theta} a^*}{a_i} \right|$$

be deviation between difficult parameter  $b_i$  of item  $i$  and the modified ability mean  $\bar{\theta}$ . Note that the mean  $\bar{\theta}$  is modified by  $\frac{a^* \bar{\theta}}{a_i}$  so that examinee's ability conforms to item's difficult parameter. Your attention please, examinee's ability shares the same meaning with difficult parameter according to formula IV.1.12.

Let  $C$  be set of items whose deviations  $\left| b_i - \frac{\bar{\theta} a^*}{a_i} \right|$  are less than a pre-defined threshold  $\delta$ . The  $\delta$  is called informative threshold and  $C$  is called informative set. Thus, items in  $C$  are called *informative items*, which are ones whose difficult parameters are approximate to the average ability  $\bar{\theta}$ . Instead of using threshold  $\delta$ , we can construct  $C$  by limiting its size. For example, given size 10, the set  $C$  will consist of 10 items whose deviations  $\left| b_i - \frac{\bar{\theta} a^*}{a_i} \right|$  are smaller than remaining items.

For each item in  $C$ , the best item is the one whose parameter  $a_i$  is nearest to discriminatory estimate  $a^*$  with note that  $a^*$  is specified by formula IV.1.20. In other words, if the best item is item  $v$  then, the deviation  $\left| b_v - \frac{\bar{\theta} a^*}{a_v} \right|$  is relatively small and the deviation  $|a_v - a^*|$  is smallest. It is easy to infer that the best item is the informative and discriminatory item.

End For

The best item are the most suitable to examinees' current ability estimates. Not like traditional CAT mentioned in table IV.1.1, it is not necessary to calculate information function specified by formula IV.1.3 for selecting the best item. For this reason, the computation cost is decreased significantly.

2. Such best items will be given to examinees and examinees responds. Following code describes process of test performance.

For each examinees  $u_j$  among  $k$  examinees

The best item  $v_j$  is given to  $u_j$ . The examinee  $u_j$  performs the test item and her/his response (result) is collected. Please see previous

part [IV.1.2.2.2](#) for more details about the example of response. For example, if item is an exam whose grade ranges from 1 to 10 then, the response is the resulted grade of examinee.

End For

3. New ability estimates of examinees are computed based on responses to all of the chosen items and current abilities  $\theta_1, \theta_2, \dots, \theta_k$ . Concretely, the  $k$  ability estimates  $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$  are re-calculated according to formula [IV.1.12](#). Moreover, discriminatory estimate  $a^*$  is re-computed based on new estimates  $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$  according to formula [IV.1.20](#). Let  $\theta_1, \theta_2, \dots, \theta_k$  are current abilities of  $k$  examinees  $u_1, u_2, \dots, u_k$ . We assign  $\theta_1 = \hat{\theta}_1, \theta_2 = \hat{\theta}_2, \dots, \theta_k = \hat{\theta}_k$  in order to update current abilities  $\theta_1, \theta_2, \dots, \theta_k$ .
4. Algorithm terminates if stopping criterion is met; otherwise going back step 1.

**Table IV.1.2.** Advanced CAT algorithm

By focusing on both information function and discriminatory parameter, the advanced CAT algorithm achieves two purposes:

- Best test items given to examinees are adaptive to examinees' abilities.
- It is easy to classify examinees according to their abilities. Note that classifying users or constructing user groups is main subject that was mentioned in previous sub-section [III.3.3](#) and so, such advanced CAT algorithm is powerful tool for clustering user models.

In normal the stopping criterion in step 4 of advanced CAT algorithm is often the number of (test) items, for example, if the test has ten items then the examinee's final estimate is specified at 10<sup>th</sup> item and the test ends. This form is appropriate to examination in certain place and certain time and user is the examinee who passes or fails such examination.

Suppose in situation that user is the learner who wants to gains knowledge about some domain as much as possible and she/he does not care about passing or failing the examination. In other words, there is no test or examination and the learners prefer to study themselves by doing exercise. There is an exercise and items are questions that belong to this exercise. It is possible to use another stopping criterion in which the exercise ends only when the learner cannot do it better or worse. At that time her/his knowledge becomes saturated and such knowledge is her/his actual knowledge. The *ability error* is used to assess the saturation of learner's knowledge. The ability error is difference between current ability estimate  $\hat{\theta}$  and previous examinee's ability  $\theta$ . Given threshold  $\xi$ , if the ability error is less than  $\xi$  then the CAT algorithm; hence this is the new stopping criterion for CAT algorithm. Formula [IV.1.21](#) specifies ability error denoted  $Err$ .

$$Err = |\hat{\theta} - \theta|$$

*Formula IV.1.21.* Ability error used as stopping criterion of CAT algorithm

If advanced CAT algorithm is applied into multi-user test, there will be  $k$  ability errors for  $k$  examinees.

$$Err_j = |\hat{\theta}_j - \theta_j|$$

In this case, the advanced CAT algorithm will terminate if all ability errors  $Err_j$  are less than given threshold  $\xi$ .

Now the advanced CAT algorithm is described comprehensively. It is necessary to give an example for illustrating such algorithm. Suppose there is a multi-user test with 5 items and 4 examinees. Each item has 10 optional responses and each examinee has zero initial ability. The test stops when every examinee finishes 5 items.

	Items			
	$a_i$	$b_i$	$c_i$	$q_i$
Item 1	$a_1=2$	$b_1=2$	$c_1=0$	$q_1=10$
Item 2	$a_2=4$	$b_2=3$	$c_2=0$	$q_2=10$
Item 3	$a_3=3$	$b_3=5$	$c_3=0$	$q_3=10$
Item 4	$a_4=3$	$b_4=2$	$c_4=0$	$q_4=10$
Item 5	$a_5=5$	$b_5=4$	$c_5=0$	$q_5=10$

	Examinees	
	$Ability (\theta_i)$	
Examinee 1	$\theta_1=0$	
Examinee 2	$\theta_2=0$	
Examinee 3	$\theta_3=0$	
Examinee 4	$\theta_4=0$	

**Table IV.1.3.** A multi-user test with 5 items and 4 examinees

The ability mean is:

$$\bar{\theta} = \frac{\theta_1 + \theta_2 + \theta_3 + \theta_4}{4} = \frac{0 + 0 + 0 + 0}{4} = 0$$

The be discriminatory estimate  $a^*$  is initialized as follows:

$$a^* = \frac{a_1 + a_2 + a_3 + a_4 + a_5}{5} = \frac{2 + 4 + 3 + 3 + 5}{5} = 3.4$$

The deviations between difficult parameters of items and the ability mean  $\bar{\theta}$  are:

$$\begin{aligned} \left| b_1 - \frac{\bar{\theta}a^*}{a_1} \right| &= |2 - 0| = 2 \\ \left| b_2 - \frac{\bar{\theta}a^*}{a_2} \right| &= |3 - 0| = 3 \\ \left| b_3 - \frac{\bar{\theta}a^*}{a_3} \right| &= |5 - 0| = 5 \\ \left| b_4 - \frac{\bar{\theta}a^*}{a_4} \right| &= |2 - 0| = 2 \\ \left| b_5 - \frac{\bar{\theta}a^*}{a_5} \right| &= |4 - 0| = 4 \end{aligned}$$

Suppose we choose two items (item 1 and item 4) whose deviations  $|b_i - \frac{\bar{a}^*}{a_i}|$  are the smallest ones according to step 1 of advanced CAT algorithm. So, the informative set  $C$  is:

$$C = \{\text{item 1, item 4}\}$$

The deviations between item 1, item 4 and discriminatory estimate  $a^*$  are:

$$\begin{aligned} |a_1 - a^*| &= |2 - 3.4| = 1.4 \\ |a_4 - a^*| &= |3 - 3.4| = 0.4 \end{aligned}$$

Because the deviation  $|a_4 - a^*|$  is the smallest one, item 4 is the best item that is given to four examinees. According to step 2 of advanced CAT algorithm, suppose that responses of examinees 1, 2, 3, and 4 to item 4 are 8, 7, 6, and 6, respectively. Of course, we have  $r_1=8$ ,  $r_2=7$ ,  $r_3=6$ , and  $r_4=6$ . The ability estimates  $\hat{\theta}_1$ ,  $\hat{\theta}_2$ ,  $\hat{\theta}_3$ ,  $\hat{\theta}_4$  and discriminatory estimate  $a^*$  will be calculated according to step 3 of advanced CAT algorithm. Given zero initial ability, according to formula IV.1.12 we have:

$$\begin{aligned} \hat{\theta}_1 &= \frac{\ln(r_1) - \ln(q_1 - r_1)}{a^*} + 0 = \frac{\ln 8 - \ln(10 - 8)}{3.4} + 0 \approx 0.41 \\ \hat{\theta}_2 &= \frac{\ln(r_2) - \ln(q_2 - r_2)}{a^*} + 0 = \frac{\ln 7 - \ln(10 - 7)}{3.4} + 0 \approx 0.25 \\ \hat{\theta}_3 &= \frac{\ln(r_3) - \ln(q_3 - r_3)}{a^*} + 0 = \frac{\ln 6 - \ln(10 - 6)}{3.4} + 0 \approx 0.12 \\ \hat{\theta}_4 &= \frac{\ln(r_4) - \ln(q_4 - r_4)}{a^*} + 0 = \frac{\ln 8 - \ln(10 - 8)}{3.4} + 0 \approx 0.12 \end{aligned}$$

The mean of ability estimates  $\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4$  is:

$$\bar{\theta} = \frac{1}{4}(0.41 + 0.25 + 0.12 + 0.12) = 0.225$$

According to formula IV.1.20, we have:

$$\begin{aligned} a^* &= \sqrt{\frac{(4-1)\pi^2}{3\left(\left(\hat{\theta}_1 - \bar{\theta}\right)^2 + \left(\hat{\theta}_2 - \bar{\theta}\right)^2 + \left(\hat{\theta}_3 - \bar{\theta}\right)^2 + \left(\hat{\theta}_4 - \bar{\theta}\right)^2\right)}} \\ &= \sqrt{\frac{\pi^2}{(0.41 - 0.225)^2 + (0.25 - 0.225)^2 + (0.12 - 0.225)^2 + (0.12 - 0.225)^2}} \approx 13.24 \end{aligned}$$

The ability estimates  $\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4$  are modified based on current discriminatory estimate 13.24 and old discriminatory estimate 3.4. We have:

$$\begin{aligned} \hat{\theta}_1 &= \frac{3.4\hat{\theta}_1}{13.24} = \frac{3.4 * 0.41}{13.24} \approx 0.1047 \\ \hat{\theta}_2 &= \frac{3.4\hat{\theta}_2}{13.24} = \frac{3.4 * 0.25}{13.24} \approx 0.0640 \\ \hat{\theta}_3 &= \frac{3.4\hat{\theta}_3}{13.24} = \frac{3.4 * 0.12}{13.24} \approx 0.0306 \\ \hat{\theta}_4 &= \frac{3.4\hat{\theta}_4}{13.24} = \frac{3.4 * 0.12}{13.24} \approx 0.0306 \end{aligned}$$

Examinees' abilities  $\theta_1, \theta_2, \theta_3$ , and  $\theta_4$  are re-assigned as follows:

$$\theta_1 = \hat{\theta}_1 \approx 0.1047$$

$$\begin{aligned}\theta_2 &= \hat{\theta}_2 \approx 0.0640 \\ \theta_3 &= \hat{\theta}_3 \approx 0.0306 \\ \theta_4 &= \hat{\theta}_4 \approx 0.0306\end{aligned}$$

The item pool now includes four items 1, 2, 3, and 5. Going back step 1 of advanced CAT algorithm and the test is repeated for the second time so as to give examinees new items. The ability mean is:

$$\bar{\theta} = \frac{\theta_1 + \theta_2 + \theta_3 + \theta_4}{4} = \frac{0.1047 + 0.064 + 0.0306 + 0.0306}{4} \approx 0.0575$$

The be discriminatory estimate  $a^*$  was determined in previous test time:

$$a^* = 13.24$$

The deviations between difficult parameters of items and the ability mean  $\bar{\theta}$  are:

$$\begin{aligned}\left| b_1 - \frac{a^* \bar{\theta}}{a_1} \right| &= \left| 2 - \frac{13.24 * 0.0575}{2} \right| \approx 1.62 \\ \left| b_2 - \frac{a^* \bar{\theta}}{a_2} \right| &= \left| 3 - \frac{13.24 * 0.0575}{4} \right| \approx 2.81 \\ \left| b_3 - \frac{a^* \bar{\theta}}{a_3} \right| &= \left| 5 - \frac{13.24 * 0.0575}{3} \right| \approx 4.75 \\ \left| b_5 - \frac{a^* \bar{\theta}}{a_5} \right| &= \left| 4 - \frac{13.24 * 0.0575}{5} \right| = 3.85\end{aligned}$$

Suppose we choose two items (item 1 and item 2) whose deviations  $\left| b_i - \frac{a^* \bar{\theta}}{a_i} \right|$  are the smallest ones according to step 1 of advanced CAT algorithm. So, the informative set  $C$  is:

$$C = \{\text{item 1, item 2}\}$$

The deviations between item 1, item 2 and discriminatory estimate  $a^*$  are:

$$\begin{aligned}|a_1 - a^*| &= |2 - 13.24| = 11.24 \\ |a_2 - a^*| &= |4 - 13.24| = 9.240\end{aligned}$$

Because the deviation  $|a_2 - a^*|$  is the smallest one, item 2 is the best item that is given to four examinees. According to step 2 of advanced CAT algorithm, suppose that responses of examinees 1, 2, 3, and 4 to item 2 are 1, 6, 2, and 9, respectively. Of course, we have  $r_1=\{8, 1\}$ ,  $r_2=\{7, 6\}$ ,  $r_3=\{6, 2\}$ , and  $r_4=\{6, 9\}$ ; recall that responses of examinees 1, 2, 3, and 4 to item 4 in previous test are 8, 7, 6, and 6, respectively. The ability estimates  $\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4$  and discriminatory estimate  $a^*$  will be calculated according to step 3 of advanced CAT algorithm. Given zero initial ability, according to formula IV.1.12 we have:

$$\begin{aligned}\hat{\theta}_1 &= \frac{\ln(8+1) - \ln((10-8)+(10-1))}{13.24} + 0 \approx -0.02 \\ \hat{\theta}_2 &= \frac{\ln(7+6) - \ln((10-7)+(10-6))}{13.24} + 0 \approx 0.05 \\ \hat{\theta}_3 &= \frac{\ln(6+2) - \ln((10-6)+(10-2))}{13.24} + 0 \approx -0.03 \\ \hat{\theta}_4 &= \frac{\ln(6+9) - \ln((10-6)+(10-9))}{13.24} + 0 \approx 0.08\end{aligned}$$

The mean of ability estimates  $\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4$  is:

$$\bar{\theta} = \frac{1}{4}(-0.02 + 0.05 - 0.03 + 0.08) \approx 0.02$$

According to formula IV.1.20, we have:

$$\begin{aligned} a^* &= \sqrt{\frac{(4-1)\pi^2}{3\left(\left(\hat{\theta}_1 - \bar{\theta}\right)^2 + \left(\hat{\theta}_2 - \bar{\theta}\right)^2 + \left(\hat{\theta}_3 - \bar{\theta}\right)^2 + \left(\hat{\theta}_4 - \bar{\theta}\right)^2\right)}} \\ &= \sqrt{\frac{\pi^2}{(-0.02 - 0.02)^2 + (0.05 - 0.02)^2 + (-0.03 - 0.02)^2 + (0.08 - 0.02)^2}} \approx 34.11 \end{aligned}$$

The ability estimates  $\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4$  are modified based on current discriminatory estimate 34.11 and old discriminatory estimate 13.24. We have:

$$\begin{aligned} \hat{\theta}_1 &= \frac{13.24\hat{\theta}_1}{34.11} = \frac{13.24 * (-0.02)}{34.11} \approx -0.0059 \\ \hat{\theta}_2 &= \frac{13.24\hat{\theta}_2}{34.11} = \frac{13.24 * 0.05}{34.11} \approx 0.0181 \\ \hat{\theta}_3 &= \frac{13.24\hat{\theta}_3}{34.11} = \frac{13.24 * (-0.03)}{34.11} \approx -0.0119 \\ \hat{\theta}_4 &= \frac{13.24\hat{\theta}_4}{34.11} = \frac{13.24 * 0.08}{34.11} \approx 0.0322 \end{aligned}$$

Examinees' abilities  $\theta_1, \theta_2, \theta_3$ , and  $\theta_4$  are re-assigned as follows:

$$\begin{aligned} \theta_1 &= \hat{\theta}_1 \approx -0.0059 \\ \theta_2 &= \hat{\theta}_2 \approx 0.0181 \\ \theta_3 &= \hat{\theta}_3 \approx -0.0119 \\ \theta_4 &= \hat{\theta}_4 \approx 0.0322 \end{aligned}$$

The item pool now includes four items 1, 3, and 5. Because examinees do not finished five items yet, the advanced CAT algorithm does not stop according to its step 4. Similarly, such four steps of advanced CAT are repeated and items 5, 3, and 1 are given to examinees in turn. Of course, choosing items 5, 3, and 1 in succession is based on deviations  $|b_i - \frac{\bar{\theta}a^*}{a_i}|$  and  $|a_i - a^*|$  according to step 1 of advanced CAT algorithm. Following table IV.1.4 shows results of our multi-user test.

		$\theta_i$	$r_i$	$\hat{\theta}_i$	$a^*$
Item 4	Examinee 1	$\theta_1=0$	$r_1=8$	$\hat{\theta}_1 \approx 0.1047$	13.24
	Examinee 2	$\theta_2=0$	$r_2=7$	$\hat{\theta}_2 \approx 0.064$	
	Examinee 3	$\theta_3=0$	$r_3=6$	$\hat{\theta}_3 \approx 0.0306$	
	Examinee 4	$\theta_4=0$	$r_4=6$	$\hat{\theta}_4 \approx 0.0306$	
Item 2	Examinee 1	$\theta_1=0.1047$	$r_1=1$	$\hat{\theta}_1 \approx -0.0059$	34.11
	Examinee 2	$\theta_2=0.064$	$r_2=6$	$\hat{\theta}_2 \approx 0.0181$	
	Examinee 3	$\theta_3=0.0306$	$r_3=2$	$\hat{\theta}_3 \approx -0.0119$	
	Examinee 4	$\theta_4=0.0306$	$r_4=9$	$\hat{\theta}_4 \approx 0.0322$	
Item 5	Examinee 1	$\theta_1=-0.0059$	$r_1=4$	$\hat{\theta}_1 \approx -0.0034$	78.22

	Examinee 2	$\theta_2=0.0181$	$r_2=6$	$\hat{\theta}_2 \approx 0.007$	
	Examinee 3	$\theta_3=-0.0119$	$r_3=5$	$\hat{\theta}_3 \approx -0.0034$	
	Examinee 4	$\theta_4=0.0322$	$r_4=9$	$\hat{\theta}_4 \approx 0.0177$	
Item 3	Examinee 1	$\theta_1=-0.0034$	$r_1=7$	$\hat{\theta}_1 \approx 0$	194.41
	Examinee 2	$\theta_2=0.007$	$r_2=3$	$\hat{\theta}_2 \approx 0.001$	
	Examinee 3	$\theta_3=-0.0034$	$r_3=8$	$\hat{\theta}_3 \approx 0.0005$	
	Examinee 4	$\theta_4=0.0177$	$r_4=9$	$\hat{\theta}_4 \approx 0.008$	
Item 1	Examinee 1	$\theta_1=0$	$r_1=2$	$\hat{\theta}_1 \approx -0.0004$	616
	Examinee 2	$\theta_2=0.001$	$r_2=7$	$\hat{\theta}_2 \approx 0.0005$	
	Examinee 3	$\theta_3=0.0005$	$r_3=9$	$\hat{\theta}_3 \approx 0.0007$	
	Examinee 4	$\theta_4=0.008$	$r_4=5$	$\hat{\theta}_4 \approx 0.002$	
	Examinee 1	$\theta_1=-0.0004$			
	Examinee 2	$\theta_2=0.0005$			
	Examinee 3	$\theta_3=0.0007$			
	Examinee 4	$\theta_4=0.002$			

**Table IV.1.4.** Results of multi-user test with 5 items and 4 examinees

After doing final test item 1, four examinees gain final abilities  $\theta_1=-0.0004$ ,  $\theta_2=0.0005$ ,  $\theta_3=0.0007$ , and  $\theta_4=0.002$ . As shown in table IV.1.4, the final discriminatory estimate  $a^*=616$  is very high while the initial value of  $a^*$  is 3.4. It implies that the variance of  $\theta_1$ ,  $\theta_2$ ,  $\theta_3$ , and  $\theta_4$  gets small; please see formula IV.1.20 for more details about inverse proportion between  $a^*$  and such variance. Therefore, it is required to use high-value discriminatory parameter to discriminate among examinees.

In general, we recognized that CAT gives us the excellent tool for assessing student's ability. The CAT algorithm includes four steps in which step 3 is the most important when student's ability estimate is determined. I propose a new method to compute the ability estimate based on maximum likelihood estimation (MLE). Thus, the ability estimate is the estimated value of difficult parameter which, in turn, is learned given student's test results; please formula IV.1.12 for more details about ability estimate. Then, the discriminatory parameter is estimated based on the variance of examinee's ability. Please see formulas IV.1.17 and IV.1.20 for more details about ability variance and discriminatory estimate. When the ability variance and the discriminatory estimate are determined, the advanced CAT algorithm for multi-user test is proposed. In other words, the advanced CAT algorithm is the result of combination of three formulas IV.1.12, IV.1.17 and IV.1.20. The strong point of advanced CAT algorithm is to achieve two purposes:

- Best test items given to examinees are adaptive to examinees' abilities.
- It is easy to classify examinees according to their abilities.

Moreover I propose the stopping criterion for CAT algorithm in which given threshold  $\xi$ , if the ability error is less than  $\xi$  then the CAT algorithm stops where the ability error is difference between current ability estimate and previous student's ability. The goal of this technique is that the exercise ends only when the student cannot do it better or worse. It means that her/his

knowledge becomes saturated and such knowledge is her/his actual knowledge. This method is only suitable to training exercises because there is no restriction for the number of (question) items in exercises. Conversely, in the formal test, the examinee must finish such test right before the decline time and the number of items in formal test is fixed.

As aforementioned at the [beginning of this section IV.1](#), knowledge sub-model in [Triangular Learner Model](#) (TLM) is evaluated by two view points:

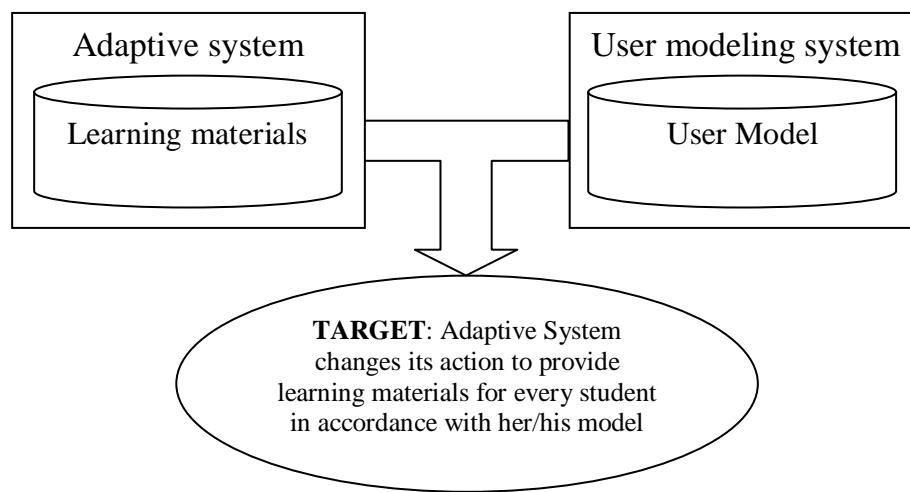
- Methodological viewpoint described in sub-section [IV.1.1](#) mentions theoretical aspects of Bayesian knowledge model.
- Practical viewpoint described in sub-section [IV.1.2](#) mentions how to exploit Bayesian model in practical assessment.

The sub-section [IV.1.2](#) focuses computerized adaptive testing (CAT) where CAT is an important assessment method for Bayesian model because it is required to assess student's knowledge via tests after they are modeled as a so-called TLM model. Finally, the subject of evaluating knowledge model is closed by this part [IV.1.2.2.3](#) in which I proposed an advanced CAT algorithm for multi-user test based on maximum likelihood estimation (MLE). The wider subject is opened by next section [IV.2](#) in which the whole of adaptive learning system and user modeling system is evaluated.

## IV.2. Evaluation of adaptive learning model

Recall that the chapter IV focuses on evaluating [Triangular Learner Model](#) (TLM) and user modeling system [Zebra](#) with regard to their effectiveness. Previous section IV.1 is the evaluation on knowledge sub-model. This section IV.2 is the wider subject which is the evaluation on the effectiveness of the whole TLM and modeling system Zebra.

As aforementioned in section I.2, adaptive learning system is defined as the system that has ability to change its actions to provide learning content and pedagogic environment/method for every student in accordance with her/his individual information/characteristics such as knowledge, learning styles, interests, goals, experiences, and backgrounds when these characteristics vary from person to person (Brusilovsky & Millán, 2007, pp. 5-14); please see section I.1 for more details about user model and user modeling. The description of learners' individual information/characteristics is learner model or user model. Adaptive learning system takes advantages of learner model to improve the quality of adaptation task but it does not build up or manipulate learner model. User modeling system is responsibility for gathering information to create and update learner model. In other words, user modeling system manages user model and provide necessary information about user to adaptive system. Following figure IV.2.1 describes the interaction between user modeling system and adaptive system.



**Figure IV.2.1.** Interaction between user modeling system and adaptive system

In this research, user modeling system is [Zebra](#), user model is [Triangular Learner Model](#) (TLM) and adaptive system is implemented as an associative learning web-based software WOW, an extension of AHA! system (De Bra, Smits, & Stash, 2006); please see sub-sections I.2.3.3 and II.2.4 for more details about AHA! system and WOW system, respectively. User modeling system is the heart of adaptive learning system. There are a lot of theories and practical methods including methodologies and approaches in this research to

build up adaptive system and user modeling system. Each method has strong points and drawbacks and so it is very useful to evaluate these methods in order to determine which model is appropriate to which situation because each method tailors to concrete conditions and contexts. For example, studying via internet website is very different from studying at a course with support of network. This section [IV.2](#) focuses on how to evaluate adaptive learning system with regard to user modeling system in e-learning or distance learning context when there is no separation between adaptive learning system and user modeling system (Nguyen, Evaluating Adaptive Learning Model, 2014). We can consider the corporation between adaptive system and user modeling system as an integrated model so-called adaptive learning model. Thus, this section has two goals:

- Firstly, research proposes criterions to evaluate adaptive learning model. Successive sub-section [IV.2.1](#) describes evaluation criterions in detailed.
- Secondly, research gives a scenario as an example that applies criterions above into performing evaluation task in concrete situations. Sub-section [IV.2.2](#) introduces an evaluation scenario.

In other words, this section [IV.2](#) gives criterions to evaluate TLM, Zebra and WOW. Note that all concepts relating to term “learning” in this research refers to learning via internet, distance learning or e-learning if there is no additional explanation.

### **IV.2.1. Evaluation criterions**

This research proposes three criterions of evaluation:

- Criterion  $\alpha$  so-called *system criterion* tells us how adaptive learning system works with/without user modeling system. For example, when modeling server applies Bayesian network into build up learner model, criterion  $\alpha$  measures the performance of adaptive system with or without the support of Bayesian network. In general, this criterion answers two following questions:
  - How adaptation is performed in adaptive system with/without the support of modeling server.
  - Whether the whole user knowledge is computed more accurately with the support of user model, for example Bayesian network.
- Criterion  $\beta$  so-called *academic criterion* tells us how well modeling server help users to study. This criterion surveys users' study result. The higher criterion  $\beta$  is, the better study result is.
- Criterion  $\gamma$  so-called *adaptation criterion* or satisfaction criterion measures the quality of adaptation function of learning system with the support of modeling server. After every student gives feedbacks or comments on adaptive system, these feedbacks are collected and analyzed; hence, criterion  $\gamma$  is calculated based on these feedbacks in order to estimate level of students' satisfaction from adaptive system. The higher criterion  $\gamma$  is, the better quality of adaptation is.

Now we discuss methods to determine these criterions. Note that in this research, the default user model is Bayesian network model if there is no additional explanation; please see sub-section III.1.1 for more details about Bayesian model. Successive sub-sections IV.2.1.1, IV.2.1.2, and IV.2.1.3 describe how to calculate criterion  $\alpha$ ,  $\beta$ , and  $\gamma$ , respectively.

#### IV.2.1.1. Calculating system criterion $\alpha$

There are two ways to calculate system criterion  $\alpha$  such as using hypothesis testing and using regression model. By using hypothesis testing, suppose the amount of knowledge that user mastered over a concept  $C$  is quantified as a measure  $k_c$ . Let  $K_U = \{k_1^U, k_2^U, \dots, k_n^U\}$  be knowledge vectors of user  $U$ , where each measure  $k_c^U$  represents the mount of knowledge  $C$  that user  $U$  mastered. Given group  $A$  and group  $B$  are groups of students learning via adaptive system with and without support of user model, respectively. Two users  $i$  and  $j$  are picked randomly in group  $A$  and group  $B$ , respectively. We have:

$$K_i = \{k_1^i, k_2^i, \dots, k_n^i\} \text{ has sample variance } s_i^2.$$

$$K_j = \{k_1^j, k_2^j, \dots, k_n^j\} \text{ has sample variance } s_j^2.$$

Let  $\bar{K}_i$  and  $\bar{K}_j$  be sample means of  $K_i$  and  $K_j$ , respectively, we have formula IV.2.1a for calculating sample mean as follows:

$$\bar{K}_i = \frac{1}{n} \sum_{c=1}^n k_c^i \text{ and } \bar{K}_j = \frac{1}{n} \sum_{c=1}^n k_c^j$$

*Formula IV.2.1a.* Sample means

Sample variances  $s_i^2$  and  $s_j^2$  are specified by following formula IV.2.1b.

$$s_i^2 = \frac{1}{n} \sum_{c=1}^n (k_c^i - \bar{K}_i)^2 \text{ and } s_j^2 = \frac{1}{n} \sum_{c=1}^n (k_c^j - \bar{K}_j)^2$$

*Formula IV.2.1b.* Sample variances

Sample standard deviation is the squared root of sample variance. Let  $s_i$  and  $s_j$  be sample standard deviations of  $K_i$  and  $K_j$ , we have formula IV.2.1c for determining sample standard deviations.

$$s_i = \sqrt{s_i^2} = \sqrt{\frac{1}{n} \sum_{c=1}^n (k_c^i - \bar{K}_i)^2}$$

$$s_j = \sqrt{s_j^2} = \sqrt{\frac{1}{n} \sum_{c=1}^n (k_c^j - \bar{K}_j)^2}$$

*Formula IV.2.1c.* Sample standard deviations

Please read (Montgomery & Runger, 2003, pp. 190-191) for more details about sample mean, and sample variance.

Criterion  $\alpha$  is represented by the statistical hypothesis testing indicates how well the Bayesian network in group  $A$  supports adaptive learning system. In other words, with the support of user model, adaptive learning system makes user knowledge around the average knowledge. Therefore, hypothesis test (Montgomery & Runger, 2003, pp. 278-288) aims to variance test instead of mean test. Null hypothesis is stated that two variances are equal. Lower-tail technique is applied when the alternative hypothesis is assumed that group  $A$  has less variance:

$$\begin{aligned} H_0 : s_i^2 &= s_j^2 \\ H_1 : s_i^2 &\neq s_j^2 \end{aligned}$$

Suppose the significant level is 0.05, F-distribution (Montgomery & Runger, 2003, p. 356) is used to test two variances. Formula IV.2.2 expresses F-distribution test for determining criterion  $\alpha$ .

$$F = \frac{s_i^2}{s_j^2}$$

*Formula IV.2.2.* System criterion  $\alpha$  determined based on F-distribution test

If  $F < f_{0.95,n-1,n-1}$  then the null hypothesis is rejected, we can include that group  $A$  with support of user modeling system improve adaptive learning system much more than group  $B$ . Note that  $f_{0.95,n-1,n-1}$  is the percentage point of F-distribution given numerator degrees of freedom  $n-1$  and denominator degrees of freedom  $n-1$  at significant level 0.05 (Montgomery & Runger, 2003, pp. 355-359). So, criterion  $\alpha$  get Boolean value *true*, indicating the preeminence of user modeling system.

The essence of  $\alpha$  is to measure the level of precision of inference methods with/without user model. By using regression technique (Montgomery & Runger, 2003, pp. 373-380), if an inference method is good, its predictive value, namely the whole knowledge user achieves, and all partial knowledge items user study at every stage on learning path will satisfy well a function or equation. In other words, this predictive value has small deviation/error. Suppose that partial user knowledge items like chapters, sessions, and concepts are represented as a set of random variable are  $X_1, X_2, X_3, \dots, X_n$ . Let  $Y$  represents the total knowledge that users gain over whole course like Java course <https://www.oracle.com/java> and Oracle database course <https://www.oracle.com/database>. We try to find out the linear function of random variables  $X_i$  (s) so that  $Y$  is the expected value of such function. Formula IV.2.3 represents linear regression function of user's total knowledge.

$$Y = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + \dots + a_nX_n$$

*Formula IV.2.3.* Linear regression function of user's total knowledge

Let  $a_i$  (s) be regression coefficients. Therefore linear function is determined, it is applied back to each user in both group  $A$  and  $B$  so as to predict her/his

knowledge so-called *estimated knowledge*. Such knowledge is compared to *real knowledge* of users. The deviation of *estimated knowledge* and *real knowledge* is called *prediction error*. The square sum of all *prediction error* reflects the measure  $\alpha$ . In general, the process to calculate  $\alpha$  has four steps:

1. Firstly, the regression coefficients  $a_i$  (s) are computed by the method of least squares (Montgomery & Runger, 2003, pp. 376-377). Note that because we have two linear functions for group A and B, there are two sets of regression coefficients, each set for one group.
2. Secondly, let  $ek_i^A$  and  $ek_i^B$  be estimated knowledge of user  $i^{th}$  in group A and B, respectively. Note that  $ek_i^A$  and  $ek_i^B$  are calculated by applying linear function determined in the first step.

$$\text{In group } A: ek_i^A = Y^A = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + \dots + a_nX_n$$

$$\text{In group } B: ek_i^B = Y^B = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + \dots + a_nX_n$$

3. Thirdly, let  $k_i^A$  and  $k_i^B$  be the real knowledge of user  $i^{th}$  in group A and B from database, respectively. Let  $err_i^A$  and  $err_i^B$  be the prediction error of user  $i^{th}$  in group A and B, respectively. There are calculated as absolute value of deviation between estimate knowledge and real knowledge, as follows:

$$err_i^A = |ek_i^A - k_i^A|$$

$$err_i^B = |ek_i^B - k_i^B|$$

4. Finally, the measure  $\alpha$  is simple inverse of square sum of all prediction errors. Formula specifies criterion  $\alpha$  based on errors of linear regression model.

$$\alpha_A = \frac{1}{\sum_{i \in A} (err_i^A)^2}$$

$$\alpha_B = \frac{1}{\sum_{i \in B} (err_i^B)^2}$$

*Formula IV.2.4.* System criterion  $\alpha$  determined based errors of linear regression model

The larger the measure  $\alpha$  is, the better the level of precision of inference method is.

Next sub-section [IV.2.1.2](#) mentions the second measure so-called academic criterion  $\beta$ .

### IV.2.1.2. Calculating academic criterion $\beta$

Let  $K_A$ ,  $K_B$  be the knowledge vectors of group A and B, respectively where  $k_i^A$  and  $k_i^B$  is the average grades of students in group A and B over concept  $i$ , respectively.

$K_A = (k_1^A, k_2^A, \dots, k_n^A)$  has sample variance  $s_A^2$  and sample mean  $\bar{A}$ .

$K_B = (k_1^B, k_2^B, \dots, k_n^B)$  has sample variance  $s_B^2$  and sample mean  $\bar{B}$ .

Sample mean and sample variance are calculated according to formulas [IV.2.1a](#) and [IV.2.1b](#), respectively. The measure  $\beta$  for each group is computed as

accumulative probability of assumption user in such group has mastered over course. Formula IV.2.5 indicates how to calculate criterion  $\beta$ .

$$\begin{aligned}\beta_A &= 1 - \Phi\left(\frac{0.5 - \bar{A}}{s_A/\sqrt{n}}\right) \\ \beta_B &= 1 - \Phi\left(\frac{0.5 - \bar{B}}{s_B/\sqrt{n}}\right)\end{aligned}$$

*Formula IV.2.5.* Academic criterion  $\beta$  based cumulative function

Where  $\Phi$  is cumulative function for standard normal distribution (Montgomery & Runger, 2003, p. 111). Note that  $\Phi$  should be accumulative function for  $t$ -distribution with  $n-1$  degree of freedom for more accurate. Here we use standard normal distribution as example for convenience. The higher criterion  $\beta$  is, the better study result is because the academy criterion  $\alpha$  is measured as the probability of event that student's grade is higher than or equal to 0.5.

Successive sub-section IV.2.1.3 mentions the last measure so-called adaptation criterion  $\gamma$ .

#### IV.2.1.3. Calculating adaptation criterion $\gamma$

Suppose a questionnaire is built up by expert and it is composed of  $n$  questions  $Q = (q_1, q_2, \dots, q_n)$ . Users in each group rate on each question where rating value may be binary satisfied and unsatisfied. Please see sub-section II.2.4 for more details about collecting users' feedbacks (ratings). By the simplest way, adaptation criterion  $\gamma$  is defined as the ratio of the number of satisfied users to the whole number of users. Formula IV.2.6 specifies simple criterion according to the number of satisfied users.

$$\gamma = \frac{\text{The number of satisfied users}}{\text{The whole number of users}}$$

*Formula IV.2.6.* Simple adaptation criterion  $\beta$  based the number of satisfied users

In enhance method, the rating values range in an interval, for example  $[0\dots5]$ , where value 0 and 5 indicates least and most satisfied. Therefore, we have two rating matrices  $A$  and  $B$  for two groups. Each row in rating matrix is composed of rating values of a user; in other words, each cell represents a rating that a user gives to a question. Each matrix is “*compressed*” into a mean vector. Let  $\mu_A$  and  $\mu_B$  be the mean vectors of group  $A$  and  $B$ , respectively. The measure  $\gamma$  is computed as the module of mean vector. Which group has higher measure  $\gamma$  will satisfy users much more.

$$\begin{aligned}\gamma_A &= |\mu_A| \\ \gamma_B &= |\mu_B|\end{aligned}$$

There are three steps to compress rating matrix and to calculate  $\gamma$ :

1. Firstly, rating matrix is “shrunken” by projecting it onto its eigenvectors (Lay, 2012, p. 267). The number of columns of shrunk matrix is much

smaller than the number of origin questions, we have  $k << n$ . These columns represent essential questions. Table IV.2.1 shows origin rating matrix as collection of users' feedbacks.

$a_{11}$	...	$a_{1j}$	...	$a_{1n}$
...	...	...	...	...
$a_{i1}$	...	$a_{ij}$	...	$a_{in}$
...	...	...	...	...
$a_{m1}$	...	$a_{mj}$	...	$a_{mn}$

**Table IV.2.1.** Rating matrix as collection of users' feedbacks

Table IV.2.2 shows shrunk matrix created by projecting origin rating matrix shown in table IV.2.1 onto its eigenvectors. Please read document (Lay, 2012, pp. 265-294) for more details about eigenvectors and how to project a matrix onto its eigenvectors.

$a_{11}$	...	$a_{1k}$
...	...	...
$a_{i1}$	...	$a_{ik}$
...	...	...
$a_{m1}$	...	$a_{mk}$

**Table IV.2.2.** Rating matrix is shrunk by projecting it onto its eigenvectors

2. Secondly, each column of matrix corresponding to each question is assumed as a statistical distribution  $F_i$ . Thus the mean of  $F_i$  is estimated by  $\mu_i$ .

$$\mu_i = \frac{1}{m} \sum_{j=1}^m a_{ij}$$

3. Finally, the mean vector of this matrix is composed of all estimates  $\mu_i$  and the criterion  $\gamma$  is the module of such mean vector. Formula IV.2.7 expresses criterion  $\gamma$  as mean of compressed vector.

$$\begin{aligned}\mu &= (\mu_1, \mu_2, \dots, \mu_k) \\ \gamma &= |\mu| = \sqrt{\mu_1^2 + \mu_2^2 + \dots + \mu_k^2}\end{aligned}$$

*Formula IV.2.7.* Adaptation criterion  $\gamma$  as mean of compressed vector

In general, the higher criterion  $\gamma$  is, the better quality of adaptation is. Now all evaluation criterions such as  $\alpha$ ,  $\beta$ , and  $\gamma$  were determined, it is necessary to organize an evaluation scenario so as to apply these measures. The next subsection IV.2.2 mentions an proposed evaluation scenario for adaptive learning model.

#### IV.2.2. An evaluation scenario

Evaluation scenario is the example for demonstrating how to calculate and apply aforementioned criterions into evaluating the quality of adaptive learning model. E-learning cannot replace face-to-face teaching and it should exist parallel and support traditional education. Thus, this scenario makes the comparison between face-to-face learning manner and distance learning manner. This evaluation scenario is divided into three main acts in which students and teacher play the roles of actors.

1. *Study act*: Teacher teaches and students learn in both face-to-face manner and e-learning manner via website. Suppose students are classified into three groups *A*, *B*, and *C*. Groups *A* and *B* represent face-to-face manner and e-learning manner via website, respectively. Especially, group *C* represents e-learning manner with support of user model, namely Bayesian network.
2. *Feedback act*: Students give feedbacks to teacher and teacher collects and analyzes them.
3. *Evaluation act* is done by teacher; thus, criterions  $\alpha$ ,  $\beta$  and  $\gamma$  are calculated according to data collected from two above acts. The quality of adaptive learning in groups *A*, *B*, and *C* are determined based on such criterions.

Study act has 5 scenes:

1. Teacher builds up school's curriculums and set up adaptive e-learning website with/without the support of user modeling system.
2. Teacher teaches and students in groups *A*, *B*, and *C* learn by face-to-face manner.
3. Students in groups *B* and *C* go on website and study by themselves. Teacher monitors them and put up important notice.
4. Students in groups *A*, *B*, and *C* do tests and exercises via website.
5. Teacher evaluates students based on their test results.

Teacher's role in study act:

- Teaching face-to-face in traditional manner.
- Building up knowledge domain and creating web resources for this domain such as defining HTML lesions, tests, exercises, etc. Please read (W3Schools, 1999) and (W3C, XHTML™ 1.1 - Module-based XHTML - Second Edition, 2010) for more details about HTML and XHTML.
- Creating user model, for example, creating Bayesian network and its weights for knowledge domain.
- Setting up user modeling system and e-learning adaptive website.
- Monitoring students' learning process.
- Sending test results and school report to students.

Students' role in study act:

- Students in groups *A*, *B*, and *C* go to class to study in face-to-face manner.

- Students in groups *B* and *C* learn themselves on adaptive learning web sites. Note that website / learning materials are adapted to each student based on their knowledge and characteristics.
- Students in groups *A*, *B*, and *C* do tests / exercises via website.

Feedback act has 3 scenes:

1. Teacher creates the questionnaire to survey students' feeling about both adaptive learning website and curriculum such as very satisfied, satisfied and not satisfied.
2. Students answer or rate on such questions online.
3. Teacher collects students' feedbacks and analyzes them.

Evaluation act has 2 scenes:

1. Teacher calculates three criterions based on students' feedback and test results.
2. Teacher makes the decision about the quality of face-to-face teaching manner and e-learning manner with/without support of user modeling system.

For example, there are two classes *A* and *B*. Class *A* is only taught in face-to-face manner, otherwise students in class *B* study both in face-to-face manner and adaptive learning environment. Study results and students feedback are collected from both two classes. Each class has the same number of students, namely 10. Let  $G_A$  and  $G_B$  be the average study results of classes *A* and *B*, respectively.

$$G_A = \{4, 5, 3, 6, 2, 8, 3, 5, 8, 6\}$$

$$G_B = \{6, 8, 9, 10, 6, 7, 9, 6, 9\}$$

Suppose there are 4 students in class *A* and 2 students in class *B* who don't satisfy teaching curriculum. Sample mean and standard deviation of class *A* are  $\bar{G}_A=5.0$  and  $s_A=\sqrt{38/9}=2.05$ . Sample mean and standard deviation of class *B* are  $\bar{G}_B=7.0$  and  $s_B=\sqrt{38/9}=1.66$ . Note that sample mean and sample standard deviation are calculated according to formulas IV.2.1a and IV.2.1c, respectively. Let  $\alpha_A$  and  $\alpha_B$  be academy criterions of class *A* and class *B*, respectively; according to formula IV.2.5, we have:

$$\alpha_A = 1 - \Phi\left(\frac{5.0 - 5.0}{\frac{2.05}{\sqrt{10}}}\right) = 1 - \Phi(0) = 0.5$$

$$\alpha_B = 1 - \Phi\left(\frac{5.0 - 7.0}{\frac{1.66}{\sqrt{10}}}\right) = 1 - \Phi(3.8) = 0.99$$

Where  $\Phi$  is cumulative function for standard normal distribution (Montgomery & Runger, 2003, p. 111)

Let  $\beta_A$  and  $\beta_B$  be satisfaction criterions of class *A* and class *B*, respectively, we calculate  $\beta_A$  and  $\beta_B$  according to formula IV.2.6 as follows:

$$\beta_A = \frac{10 - 4}{10} = 0.6$$

$$\beta_B = \frac{10 - 2}{10} = 0.8$$

Suppose the weights of criterion  $\alpha$  and criterion  $\beta$  are 0.7 and 0.3, respectively. Let  $eval_A$  and  $eval_B$  be the total evaluations on group A and group B, respectively.

$$eval_A = 0.7*0.5 + 0.3*0.6 = 0.53$$

$$eval_B = 0.7*0.99 + 0.3*0.8 = 0.93$$

When  $eval_A$  is greater than  $eval_B$ , it is possible to conclude that the teaching method in class B is more effective than the one in class A because class B takes advantages of adaptive learning method.

This section [IV.2](#) is finished with some comments on evaluation criterions. As aforementioned, there are three criterions such as system criterion  $\alpha$ , academy criterion  $\beta$  and adaptation criterion  $\gamma$ . That two of three criterions, concretely  $\alpha$  and  $\beta$ , assessing user knowledge implicates that evaluation of adaptive learning model focuses on the effect of education which is ability to help student to improve their knowledge although adaptation and personalization is significant topic in adaptive learning. You can recognize that the education never goes beyond the main goal that increases amount of human knowledge. Evaluation scenario, an example for demonstrating how to determine these criterions, indicates that study is lifelong process for everyone and so, classes and courses are short movies in this lifelong process. Both students and teachers are actors and their roles can mutually interchange, for example, teaching is the best way to learn and student is the best teacher of teacher.

This section [IV.2](#) ends up the main contents of this research and gives you the comprehensive and detailed description about thesis and so the next chapter [V](#) is the conclusion and future trend.

# Chapter V. Conclusion and Future Trend

Recall that chapter I is the state-of-art of user model and user modeling system. Chapters II, III are essential chapters which focus on main works of the research including how to [Triangular Learner Model](#) (TLM) is constructed and how to the user modeling system [Zebra](#) is built up and manipulates TLM. Chapter IV is the evaluation of TLM and Zebra. Now main contents of this research were introduced wholly to you. Therefore, this chapter V only aims to conclude the whole research with the viewpoint of general architecture when main details and formulas are described in the most important chapter III and so, the section V.1 is the general conclusion of the research. Additionally, section V.2 mentions the main future trend of the research; thus the proposed user modeling system [Zebra](#) will support ubiquitous system.

## V.1. Conclusion

This conclusion gives you an overview of this research together with strong points and limitation, research directions from this research and how to take this research further. The conclusion is described according to viewpoint of general architecture. The higher is the standard of living, the more important are adaptive IT systems which have ability of change their behaviors so as to be in accordance users' characteristics. Note that this thesis focuses on e-learning and so the term "user" implicates "learner" or "student" in learning context. The representation of such characteristics is called user model but there is too much information about individuals to model all users' characteristics; so it is necessary to choose essential characteristics from which a solid architecture of user model is built. Each modeling method is only appropriate to respective characteristic like knowledge, learning style, learning history, interest, and goal. There is no modeling method fit all characteristics. On the other hand, some domain-independent user modeling systems are too generic to "cover" all learners' characteristics in e-learning context, which may cause unpredictable bad consequences in adaptation process. Moreover user modeling systems require effective inference techniques in their modeling tasks but this is impossible if we can't recognize which individual characteristics are important.

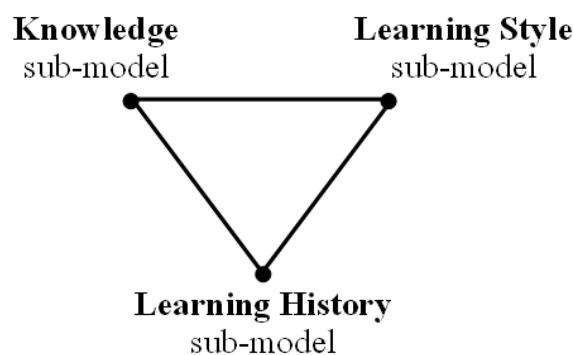
To overcome these obstacles, I propose the new learner model that contains three most important characteristics of user: knowledge (K), learning styles (LS) and learning history (LH). Such three characteristics form a triangle; so our model is called Triangular Learner Model (TLM). TLM with three underlying characteristics will cover the whole of user's information required by learning adaptation process. The reasons for such assertion are:

- Knowledge, learning styles and learning history are prerequisite for modeling learner.
- While learning history and knowledge change themselves frequently, learning styles are relatively stable. The combination of them ensures the integrity of information about learner.

- User knowledge is domain specific information and learning styles are personal traits. The combination of them supports user modeling system to take full advantages of both domain specific information and domain independent information in user model.

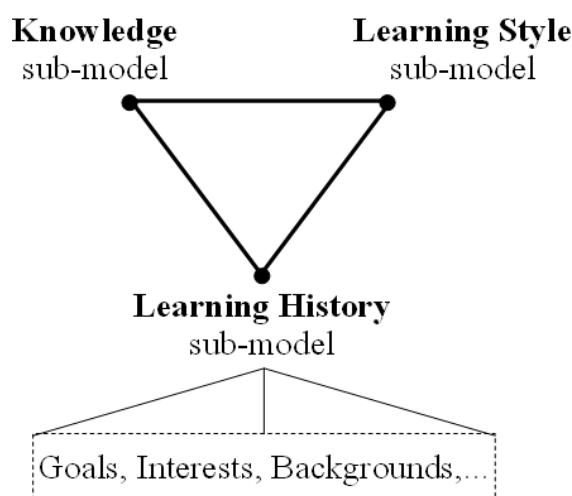
These reasons are also strong points of TLM because they reflect the sufficiency and solid of an optimal learner model. Moreover, TLM emphasizes on the inference mechanism by applying Bayesian network and Markov model into modeling user knowledge and learning style. Intelligent deduction is the best feature of TLM instead of providing user information only as normal user modeling system.

So TLM is constituted of three sub-models such as knowledge, learning style, and learning history. Following figure V.1.1 is replication of figure II.6 in previous sub-section II.2.1, which depicts TLM.



**Figure V.1.1.** Triangular Learner Model

The TLM can be extended to interpret more detailed about learner by attaching more learners' characteristics such as interests, background, and goals into the learning history sub-model. Following figure V.1.2 is replication of figure II.7 in previous sub-section II.2.1, which depicts extended TLM.

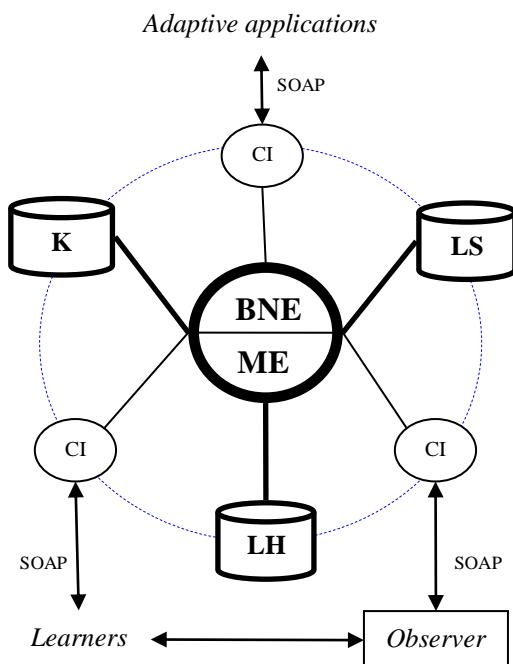


**Figure V.1.2.** extended Triangular Learner Model

I also introduce the architecture of the user modeling system so-called **Zebra** that realizes the TLM. The core of Zebra is the composition of two engines: *mining engine* (ME) and *belief network engine* (BNE).

- Mining engine (ME) is responsible for collecting learners' data, monitoring their actions, structuring and updating TLM. Mining engine also provides important information to belief network engine; it is considered as input for belief network engine. Mining engine almost always uses mining techniques. It has three other important functionalities that are to discover some other characteristics beyond knowledge and learning styles (such as interests, goals, and learning context), to support learning concept recommendation and to support collaborative learning. These functionalities were described in section [III.3](#).
- Belief network engine (BNE) is responsible for inferring new personal traits from TLM by using deduction mechanism available in belief network. This engine applies Bayesian network and hidden Markov model into inference mechanism. Two sub-models such as knowledge and learning style are managed by this engine, which were mentioned in sections [III.1](#) and [III.2](#).

Zebra provides *communication interfaces* (CI) that allows users and adaptive systems to see or modify restrictedly TLM. Adaptive applications also interact with Zebra by these interfaces. Following figure [V.1.3](#) is replication of figure [II.10](#) in previous sub-section [II.2.2](#), which depicts the general architecture of Zebra.



**Figure V.1.3.** The architecture of Zebra

This research is fundamental research, thus, the methodology to build up TLM is specified and proven via mathematic tools. The feasibility of TLM and

architecture is authenticated via the user modeling software names **Zebra** associated with TLM. Moreover, the adaptive learning system that interacts with Zebra is also implemented as e-learning web-based software. Another strong point of TLM is ability of extension, other user information such as user interests and goals can be discovered or extracted from TLM. Researchers can use the methodology, models and mathematical formulas in this research to build up their own user model and user modeling system. They can also develop TLM and Zebra by extending advanced functions such as discovering user goals, context-aware adaptation, ubiquitous modeling, and mobile service. Next section [V.2](#) discussing the future trend is an example of TLM extension; in which TLM will be integrated into ubiquitous service, which means that Zebra will support ubiquitous user modeling. This research contributes to user modeling and adaptive learning community a methodology for constructing user model and a whole perfect learner model (Nguyen, A User Modeling System for Adaptive Learning, 2014).

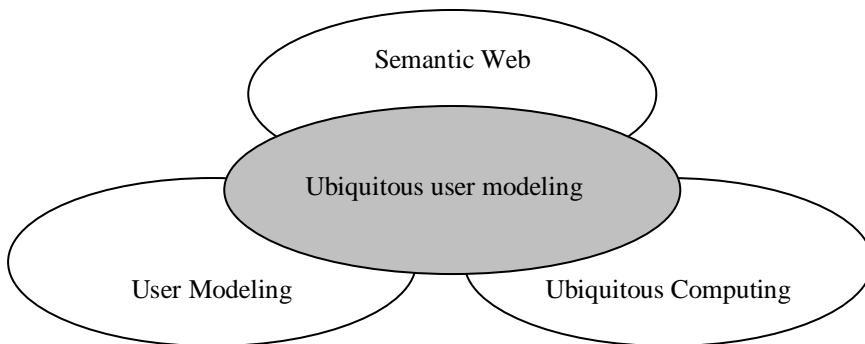
The limitation of this research is lack of privacy in learner model although external applications only access TLM via communication interface (see subsection [II.2.2](#)) but user information is not encrypted now. The solution is to plug additional encrypted/decrypted module into communication interface but this will decrease system performance and inference speed is reduced.

Zebra will be developed towards ubiquitous user modeling. This future trend is mentioned in successive section [V.2](#).

## V.2. Towards ubiquitous user modeling

Because [Triangular Learner Model](#) (TLM) and the proposed user modeling system [Zebra](#) trend to support ubiquitous user modeling, it is necessary to survey ubiquitous user modeling. Nowadays there is a need for users to interact with many information technology (IT) systems at anywhere, for examples, users withdraw cash by ATM card or book airplane ticket online or play games on mobile phones (Heckmann D. , 2005, p. 3). This tends to develop ubiquitous user modeling system which performs ongoing modeling, ongoing sharing and ongoing exploitation of user models in ubiquitous computing environment that shifts from desktop interaction to mobile interaction. Ongoing modeling, ongoing sharing and ongoing exploitation of user models are three most important aspects of ubiquitous user modeling. Ubiquitous user model is defined as the user model which is monitored at any time, at any location and in any interaction context. Ubiquitous user model can be shared or integrated when necessary.

According to (Heckmann D. , 2005, p. 6), ubiquitous user modeling describes ongoing modeling and exploitation of user behavior with a variety of systems that share their user models. Ubiquitous user modeling is considered as the intersection of three domains: user modeling, ubiquitous computing, and semantic web, as shown figure [V.2.1](#) (Heckmann D. , 2005, p. 6).



**Figure V.2.1.** Ubiquitous user modeling

All contents relevant to ubiquitous user modeling are extracted from the PhD research “Ubiquitous User Modeling” of the author (Heckmann D. , 2005). For convenience, some reference links are ignored and so readers should read this research available at <http://books.google.com.vn/books?id=e5adLEi4gYgC> for more details about ubiquitous user modeling. This section [V.2](#) includes following sub-sections:

- Sub-section [V.2.1](#) is an overview of ubiquitous user modeling.
- Ubiquitous user modeling system models all things (including learners) in real world as situation models. Hence, sub-section [V.2.2](#) mentions knowledge representation of situation model.
- Based on situation model, ubiquitous user modeling system simulates the real world as virtual world that is called ubiquitous world. Sub-section [V.2.3](#) sketches such ubiquitous world.

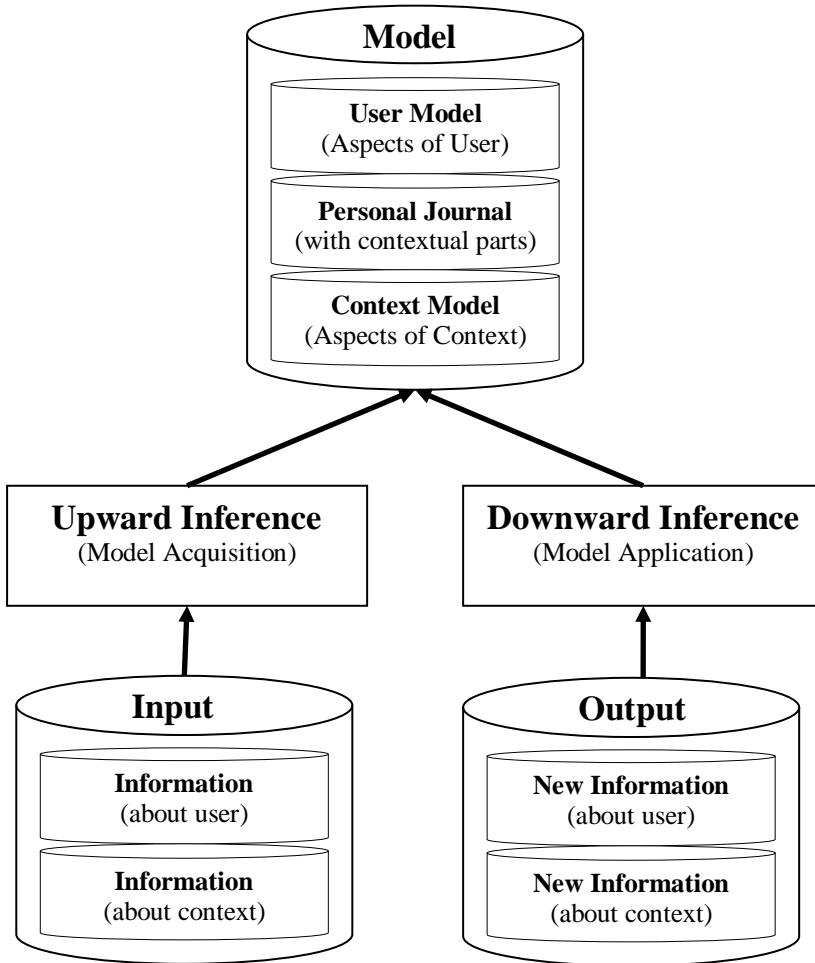
- Ubiquitous user modeling system is implemented as ubiquitous user model service. Sub-section [V.2.4](#) describes architecture and main actions of ubiquitous user model service.
- Sub-section [V.2.5](#) focuses on future trend of TLM and Zebra in which Zebra is integrated into ubiquitous user model service. In other words, Zebra will be developed towards ubiquitous user modeling.

### **V.2.1. Overview of ubiquitous user modeling**

As discussed ubiquitous user modeling is the integration of user modeling, ubiquitous computing and semantic web. It is necessary to glance over these areas. Sub-sections [V.2.1.1](#), [V.2.1.2](#), and [V.2.1.3](#) browse context-aware user modeling, ubiquitous computing, and semantic web, respectively.

#### **V.2.1.1. User modeling and context-awareness**

User modeling function is not only responsible for building up user model, maintaining the consistence of the model but also integrated with context-awareness function. It must monitor user's behaviors and changes in context and can take out new assumptions about user and context. The schema of user modeling with integrated context-awareness is shown in following figure [V.2.2](#) (Heckmann D. , 2005, p. 15). Because user modeling is surveyed carefully in previous section [I.1](#), this sub-section [V.2.1.1](#) only mentions context-aware user modeling when ubiquitous system always focuses on environment and context.



**Figure V.2.2.** User modeling with integrated context-awareness

Input data about user and context are processed in upward inference direction. This is essentially process of collecting data that adds aspects about user to user model or personal journal. The downward inference direction is responsible for taking out new assumptions about user or context (Heckmann D. , 2005, p. 15). It can use artificial intelligence techniques to improve deduction ability.

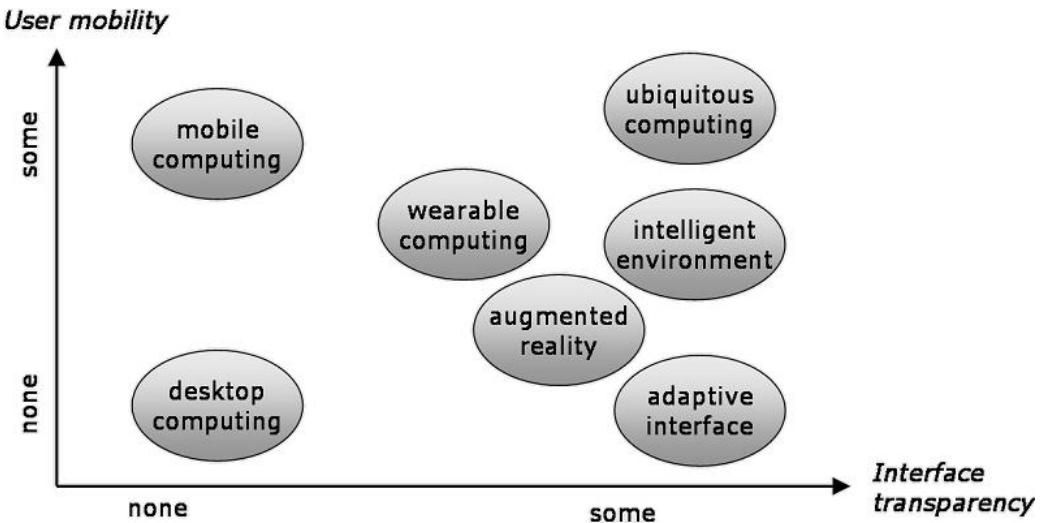
As aforementioned in figure V.2.1, ubiquitous user modeling is the intersection of user modeling, ubiquitous computing, and semantic web. The successive sub-section V.2.1.2 mentions ubiquitous computing.

### V.2.1.2. Ubiquitous computing

According to (Heckmann D. , 2005, p. 19), ubiquitous computing system is defined as a heterogeneous set of computing devices, a set of supported tasks and some infrastructure on which the devices rely on in order to carry out their tasks. There are two basic properties of ubiquitous computing system: ubiquity and transparency (Heckmann D. , 2005, p. 19).

- Ubiquity: User can interact with system at anywhere.
- Transparency: The system is non-intrusive and is integrated into everyday environment.

There are some research fields which are similar to ubiquitous computing such as mobile computing, wearable computing, intelligent environment, etc. Author (Maffioletti, 2001, p. 2) gives two dimensions: *user mobility* and *interface transparency* in order to distinguish these fields. User mobility (ubiquity) reflects the degree of freedom that user can move around, when interacting with system. Interface transparency reflects the attention that system requires of user. The boundary between these fields (or the spatial arrangement) is represented in following figure [V.2.3](#) (Heckmann D. , 2005, p. 20).



**Figure V.2.3.** Spatial arrangement of ubiquitous computing similar fields according to two dimensions: user mobility and interface transparency

As shown ubiquitous computing gains the highest level of user mobility of interface transparent. Note that intelligent environment is the environment (around user) enriched by computers embedded in everyday objects like computer, mobile devices, blackboard, tables, chairs... and enriched by sensors to catch information from context (Heckmann D. , 2005, p. 19). Ubiquitous computing system can be considered as advanced intelligent environment.

### Situation model and situated interaction

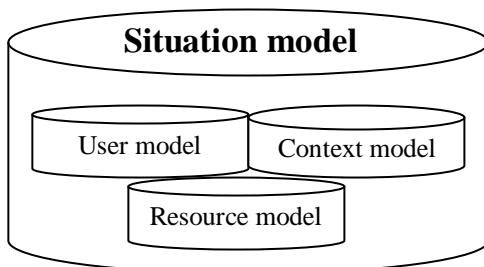
Authors (Barwise & Perry, 1981) suggests the *Situation Semantics* which describes a framework to conceptualize everyday situations. Situations are constituted of objects owning properties and standing in relationships. The real world consists of many situations. Each situation is established at a concrete time and place. Situations are categorized into various situation-types; it means that situation-types are partial functions that specify situations (Heckmann D. , 2005, p. 21).

Situations constitute the *situation model* which consists of three main sub-models: user model, context model and resource model:

- *User model* comprises user-related information such as knowledge, goals, personal traits, etc. User model was mentioned in previous section [I.1](#).

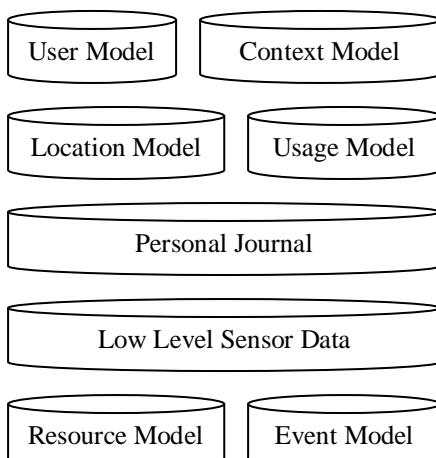
- *Context model* comprises user perception and actions which are independent of individual user when they affect equally all users in the same environment (Heckmann D. , 2005, p. 23). In other words, perception and actions are determined by environment.
- *Resource model* gives support to resource-adaptive process that is aware of what resources are available to (ubiquitous computing) system at any time and that employs a predefined adaptation strategy to perform well when faced with such a situation of varying resource availability (Heckmann D. , 2005, p. 23). Resource is defined as available means to solve a task.

Figure V.2.4 depicts situation model comprising of user model, context model, and resource model.



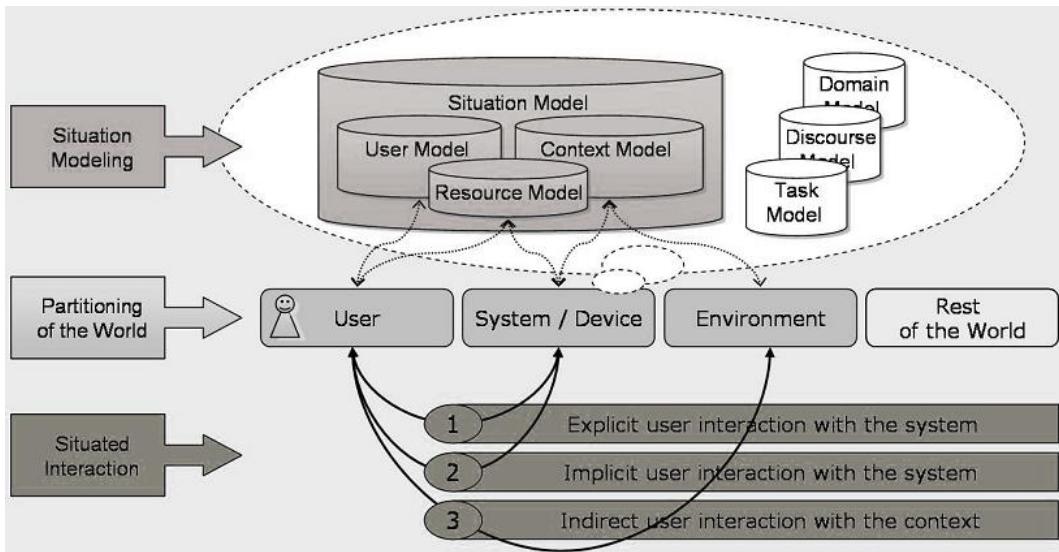
**Figure V.2.4.** Situation model

However it can be contains more models such as usage model, location model, personal journal, low-level sensor data and event log (Heckmann D. , 2005, p. 63). The following figure V.2.5 is the extended situation model (Heckmann D. , 2005, p. 63).



**Figure V.2.5.** Extended situation model

The process so-called *situation modeling* process is responsible for constructing situation model. The following figure V.2.6 (Heckmann D. , 2005, pp. 23-24) gives the overview of situated interaction for ubiquitous computing where the world is divided into four parts: *user*, *system/mobile devices*, *environment* and *rest of the world*.



**Figure V.2.6.** Situated interaction and situation modeling for ubiquitous computing

So *situated interaction* denotes the explicit, implicit and indirect interaction between a human and an intelligent environment (such as desktop, laptop, and mobile devices), that take situation-awareness into account (Heckmann D. , 2005, p. 25). Note that intelligent environment is everything around user consisting of information technology (IT) devices, tables, chairs, etc. Implicit interaction is defined as the interaction of human with environment which is aimed to achieve a goal. Within this process the system acquires implicit inputs from the user and may present implicit output to the user. Implicit input are user actions which have not chief purpose of interaction with IT devices.

As aforementioned in figure V.2.1, ubiquitous user modeling is the intersection of user modeling, ubiquitous computing, and semantic web. The successive sub-section V.2.1.3 mentions semantic web.

### V.2.1.3. Semantic web

According to W3Consortium <http://www.w3c.org>, semantic web is a web of data. It is about common formats for integration and combination of data drawn from diverse sources and about language for recording how the data relates to real world objects. The semantic web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries (W3C, W3C semantic web activity, 2001). The basic idea of semantic web is the use of ontology to annotate resource in the World Wide Web (Heckmann D. , 2005, p. 27). Semantic web is based on the Resource Description Framework (RDF). W3Consortium defines that RDF integrates a variety of applications from library catalogs and world-wide directories to syndication and aggregation of news, software, and content to personal collections of music, photos, and events using XML (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) as an interchange syntax. In brief, RDF is the mean to represent the meta-data about resources on internet

such as title, author, modification date, copyright, and license information about web document (Heckmann D. , 2005, p. 29). The resource is defined as a network data object (web document, ontology entity, etc) or service that can be identified by a URI where URI is string-format identifier used to refer to web resource on internet (Heckmann D. , 2005, p. 28).

So RDF in form of XML file is used as the mean to represent semantic web. However there is another language that is more powerful for expressing semantic web than RDF; this is Ontology Web Language (OWL) giving support to interpretability. OWL adds more vocabulary for describing properties and classes of web resources and among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes (Heckmann D. , 2005, p. 29). The General User Model Ontology (GUMO) is also based on OWL. Please read (Heckmann, Schwartz, Brandherm, Schmitz, & Wilamowitz-Moellendorff, 2005) for more details about GUMO. Because ontology is very important to semantic web, please study some basic concepts of ontology (Wikipedia, Ontology (information science), 2014).

Ubiquitous user modeling system models all things (including learners) in real world as situation models. Hence, successive sub-section [V.2.2](#) mentions knowledge representation of situation model.

### **V.2.2. Knowledge presentation of situation model**

Situation model consisting of user model, context model and resource model is the core of ubiquitous user modeling system. Situation model has many situations which are represented as situational statements. Each *situational statement* contains information about user model entries, context data, sensor data, data on mobile devices, environment, etc (Heckmann D. , 2005, p. 53). The concept “situational statement” derived from the integration of three domains: user modeling, ubiquitous computing and semantic web give us the way to represent knowledge base of ubiquitous user modeling. Now we should survey this concept.

The representation of situational statement is based on RDF and FrameNet approach. FrameNet approach aims to create an online lexical resource for English, based on frame semantic and supported by corpus evidence (Heckmann D. , 2005, p. 46). Information is organized as frames. A frame is an intuitive construct that makes it possible to formalize the links between semantics and syntax in the results of lexical analysis. Each frame identifies a set of frame elements which are frame-specific semantic roles such as participants, props, phases of a state of affairs (Heckmann D. , 2005, p. 47).

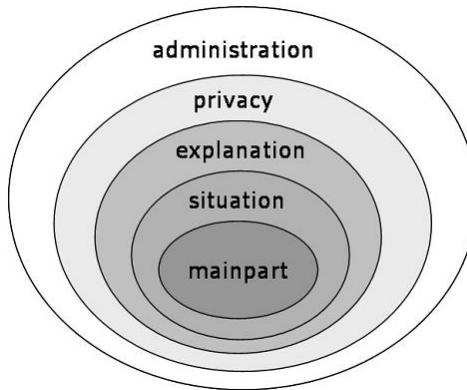
RDF is based on the triples of *subject*, *predicate* and *object* (Heckmann D. , 2005, p. 57). These elements (*subject*, *predicate*, *object*) are resources which can be refer to different ontologies such as OWL and GUMO (Heckmann, Schwartz, Brandherm, Schmitz, & Wilamowitz-Moellendorff, 2005). The *subject* is linked to the *object* denoting the value through an arc labeled with the *predicate*. It means that the *subject* has a property *predicate*, valued by

*object*. For example, the triples “*subject* = Susan; *predicate* = blood pressure; *object* = high” denotes the statement “Susan has high blood pressure”. Note that “situational statement” is called in brief “statement” and **statement model refers to situation model**. Figure V.2.7 expresses RDF triple that represents a statement.



**Figure V.2.7.** RDF triple

Each situational statement is represented as the onion model with five layers (Heckmann D., 2005, pp. 58-59), shown in figure V.2.8. Each layer containing attributes is considered a collection or box. There are five boxes: *main part*, *situation*, *explanation*, *privacy* and *administration* (Heckmann D., 2005, p. 59). The organization is the hierarchy of boxes (layers) wrapped around the main part box. The attributes in each box are arranged in form of RDF triple: *subject – predicate – object*. It is easy to recognize that the organization of situation statement is derived from FrameNet. Consequently each box is considered as a frame in FrameNet approach. Note that attributes can be linked to ontologies such as OWL and GUMO.



**Figure V.2.8.** Onion model of statement

- *Main part box* contains main information about statement in form of attribute-value pairs (Heckmann D., 2005, p. 59). Besides basic attributes derived directly from RDF triple such as *subject*, *predicate* and *object*, there are two optional attributes: *auxiliary* and *range*. For example, the main part box of statement “Susan’s mark of math test is 5” is expressed as: “*subject* = Susan”, “*predicate* = mark of math test” and “*range* = 5”. The attribute *predicate* in original RDF triple is now divided into three attributes: *auxiliary*, *predicate* and *range* in order to extend the ability of representing statement in real world. Table V.2.1 (Heckmann D., 2005, p. 59) shows the main part box.

Main part
Subject

Auxiliary
Predicate
Range
Object

**Table V.2.1.** Main part box

- *Situation box* contains information about the temporal and spatial embedding of situational statement (Heckmann D. , 2005, p. 61). This is really important in ubiquitous computing when it is necessary to observe user at any time and anywhere. Situation box has five attributes: *start*, *end*, *durability*, *location* and *position*. All of them are optional. The attribute *start* expresses the time when the statement become valid. The attribute *end* expresses the time when the statement ends. The attribute *durability* expresses the time span that the statement is still valid. The attribute *location* denotes the spatial location that the statement occurs. It is slightly different from the attribute *position* which denotes exactly the coordinates. Table [V.2.2](#) (Heckmann D. , 2005, p. 61) shows the situation box.

Situation
Start
End
Durability
Location
Position

**Table V.2.2.** Situation box

- *Explanation box* consists of attributes which are required by the user's right for explanation and by the conflict resolution mechanism inferred user modeling data (Heckmann D. , 2005, p. 62). There are five attributes: *source*, *creator*, *method*, *evidence* and *confidence*. All of them are optional. The attribute *source* declares where the statement was stored or from which sensor measurement it came from. The attribute *creator* gives the name of system or person who created statement. The attribute *method* tells us how the main part box was inferred or measured. The attribute *evidence* provides the list of evidence that supports statement. The attribute *confidence* provides the place to store the creator's expected level of confidence in statement. Table [V.2.3](#) (Heckmann D. , 2005, p. 62) shows the explanation box.

Explanation
Source
Creator
Method
Evidence
Confidence

**Table V.2.3.** Explanation box

- *Privacy box* provides critical information about privacy of user model data, especially in case of distributing sensitive data (Heckmann D. , 2005, p. 62). There are five optional attributes: *key*, *owner*, *access*, *purpose* and *retention*. The *key* attribute forms an encrypted security key. The *owner* attribute provides the person or system that is responsible for managing the distribution of user model data and editing three remaining attributes: *access*, *purpose* and *retention*. The *access* attribute which specifies access right of system or person has possible values: public, friend and private. The *purpose* attribute which put restrictions on the intention for which statement may be used has possible values: commercial, research and minimal. The *retention* attribute which expresses how long statement is kept or used has possible values: long, middle and short. Privacy is especially important in distributed environment like ubiquitous computing system. Table V.2.4 shows the privacy box.

Privacy
Key
Owner
Access
Purpose
Retention

**Table V.2.4.** Privacy box

- *Administration box* is the outermost layer in the onion model of statement (Heckmann D. , 2005, p. 63). It fastens organizational issues in huge sets of situation statements. Administration box has five attributes: *id*, *unique*, *replaces*, *group* and *notes*. The *id* attribute is the locally unique identifier of statement but the *unique* attribute is the global unique identifier of statement all over the world. *Unique* attribute is often defined as URI. Maybe the creator who creates a statement has no right to delete or replace it. So the *replaces* attribute specifies which statement can be replaced by a newer one. As discussed the situation model consists of three sub-models: user model, context model and resource model. However it can be contains more model such as usage model, location model, personal journal, low-level sensor data, event log. These models are denoted by attribute *group*. The least important attribute *notes* contains additional unstructured meta information about statement. Table V.2.5 shows the administration box.

Administrator
Id
Unique
Replaces
Group
Notes

**Table V.2.5.** Administrator box

That the representation of situation model is derived from RDF triple and OWL makes the integration of three fields: user modeling, ubiquitous computing and semantic web. The model of situational statement (onion model) is in form of XML file (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008); all attributes are translated into XML-elements. The syntax of XML-format to represent situational statement is called *SituationML* or *UserML* (Heckmann D. , 2005, pp. 67-70). The onion model with five boxes is represented by XML format as below (Heckmann D. , 2005, p. 69):

```

<statement>
  <mainpart>
    <subject>a1</subject>
    <auxiliary>a2</auxiliary>
    <predicate>a3</predicate>
    <range>a4</range>
    <object>a5</object>
  </mainpart>

  <situation>
    <start>a6</start>
    <end>a7</end>
    <durability>a8</durability>
    <location>a9</location>
    <position>a10</position>
  </situation>

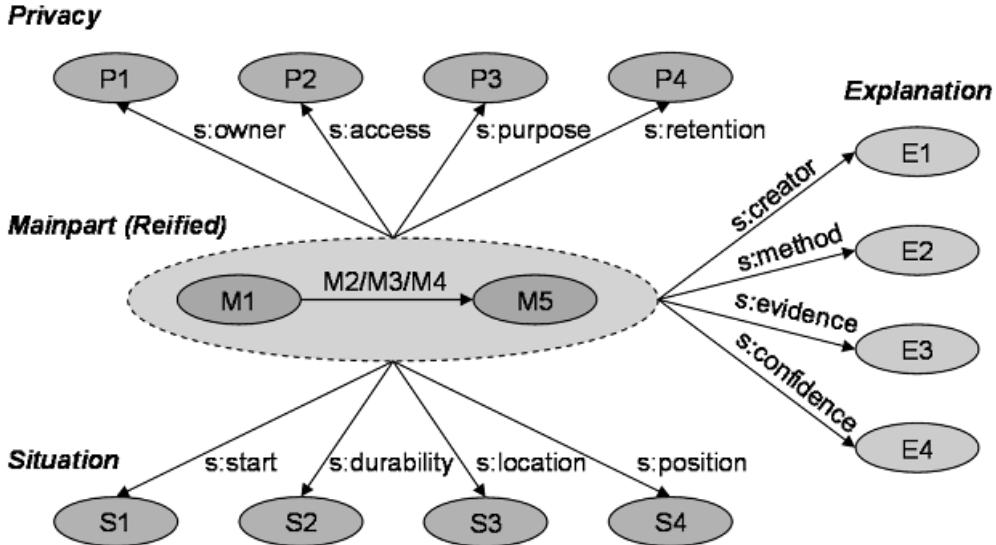
  <explanation>
    <source>a11</source>
    <creator>a12</creator>
    <method>a13</method>
    <evidence>a14</evidence>
    <confidence>a15</confidence>
  </explanation>

  <privacy>
    <key>a16</key>
    <owner>a17</owner>
    <access>a18</access>
    <purpose>a19</purpose>
    <retention>a20</retention>
  </privacy>

  <administrator>
    <id>a21</id>
    <unique>a22</unique>
    <replaces>a23</replaces>
    <group>a24</group>
    <notes>a25</notes>
  </administrator>
</statement>
```

Following figure V.2.9 shows the schema of statement model in RDF graph notation in the variation with reification (Heckmann D. , 2005, pp. 70-72). The reified main part is interpreted more with situation attributes, privacy attributes and explanation attributes. Each oval refers to a resource which may be an

ontology entity. The attributes are reified as RDF predicates such as *s:start*, *s:location*, and *s:owner* while theirs can be defined as RDF objects.



**Figure V.2.9.** The schema of statement model in RDF graph notation

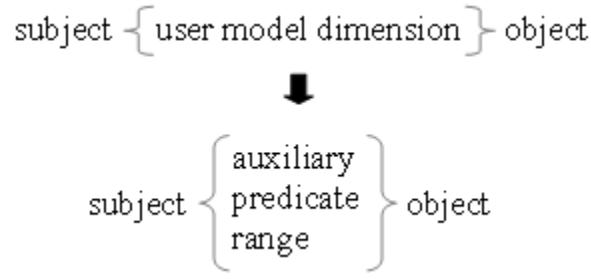
Based on situation model, ubiquitous user modeling system simulates the real world as virtual world that is called ubiquitous world. Successive sub-section [V.2.3](#) sketches such ubiquitous world.

### V.2.3. Ubiquitous World

Ubiquitous user modeling system cannot reflect totally real world; so it must perform modeling tasks in a uniform virtual world model in order to simulate the real world. Such virtual world is so-called Ubiquitous World. Moreover statements in situation model refer to various ontologies. So we should glance over user model ontologies (sub-section [V.2.3.1](#)) and ubiquitous world (sub-section [V.2.3.2](#)).

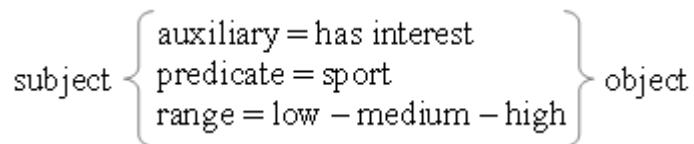
#### V.2.3.1. User model ontologies

As discussed, RDF and ontologies are main means of integrating user modeling and ubiquitous computing. The attributes in situational statement in form of RDF triple can be refer to other ontology namely GUMO (Heckmann, Schwartz, Brandherm, Schmitz, & Wilamowitz-Moellendorff, 2005). Thus the user model dimension in RDF triple is divided into three parts: *auxiliary*, *predicate*, *range* (Heckmann D. , 2005, p. 86). Figure [V.2.10](#) shows the situational statement represented by RDF triple.



**Figure V.2.10.** Situational statement represented by RDF triple

So user model dimensions such as auxiliary, predicate, range or any other information about user are entities in GUMO. The GUMO give the clear separation in the modeling between user model auxiliaries, predicate classes and special ranges (Heckmann D. , 2005, p. 86). For example the statement “*Peter likes sport very much*” is translated into the form of RDF triple, shown in figure V.2.11 as follows:



**Figure V.2.11.** An example of statement represented by RDF triple

The auxiliaries give support to specify the predicates which refer to ontology entities. There are some following auxiliaries (Heckmann D. , 2005, p. 87):

- The auxiliary “*has property*” leads to *basic user dimensions* such as personality, contact information, demographics, etc.
- The auxiliaries “*has interest*” and “*has preference*” lead to *user model interest categories* such as music categories or film genres.
- The auxiliaries “*has regularity*” and “*has done*” lead to usage data and low-level sensor data.
- The auxiliary “*has location*” leads spatial ontology.

According to GUMO, *basic user dimensions* that are basic information about user are classified into: contact information, demographics, personality, characteristics, etc and *user model interest categories* include: film, music, sports, etc. Table V.2.6 (Heckmann D. , 2005, p. 88) shows basic user dimensions in GUMO.

Basic User Dimension
Contact Information
Demographics
Ability and Proficiency
Personality
Characteristics
Emotional State
Physiological State
Mental State

Motion
Role
Nutrition
Facial Nutrition

**Table V.2.6.** Basic user dimensions in GUMO

Table [V.2.7](#) (Heckmann D. , 2005, p. 98) shows user model interest categories in GUMO.

Interest Category
Film
Music
Sports
PC-Games
Recreation
Environmental Topics
Science
Museum

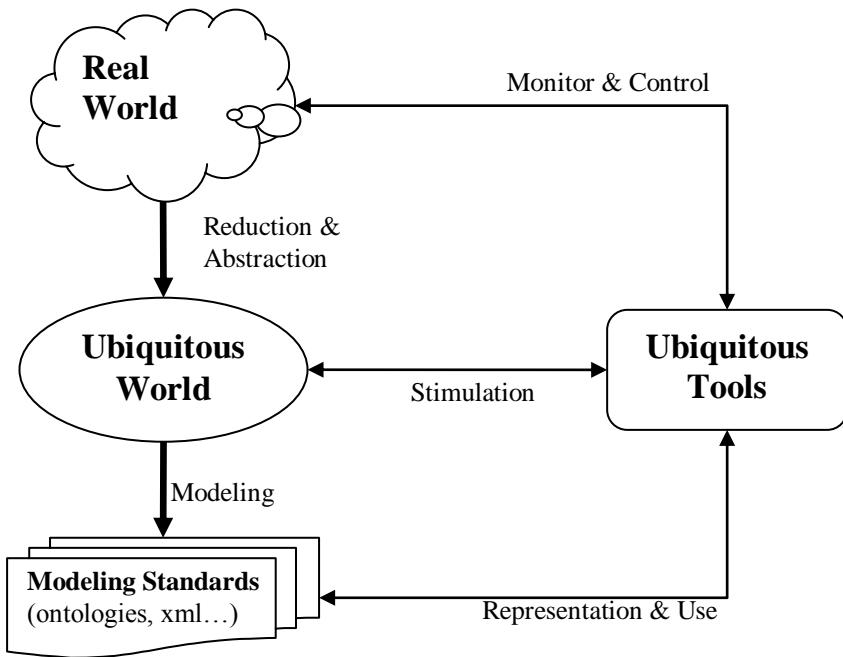
**Table V.2.7.** User model interest categories in GUMO

Note that there are not only GUMO but also other ontologies to be applied in ubiquitous user modeling. Besides GUMO – the ontology supporting situation model (situational statement) about user, we have some ontologies such as physical ontology, spatial ontology, temporal ontology, activity ontology, inference ontology, etc.

When user model ontologies are defined according to GUMO, it is easy to describe ubiquitous world in successive sub-section [V.2.3.2](#).

### V.2.3.2. Ubiquitous World

Ubiquitous World (UbisWorld) is considered as the partial copy of real world. It represents some parts of real world such as office, school, airport, etc. UbisWorld includes persons, objects, locations as well as times, events and their properties and features in real world (Heckmann D. , 2005, p. 100). So UbisWorld is also defined as the uniform virtual world used for simulation, inspection and control (Heckmann D. , 2005, p. 100). All tasks in ubiquitous user modeling are assumed to take place in UbisWorld. The correlation between UbisWorld and real world is shown in following figure [V.2.12](#) (Heckmann D. , 2005, p. 100).



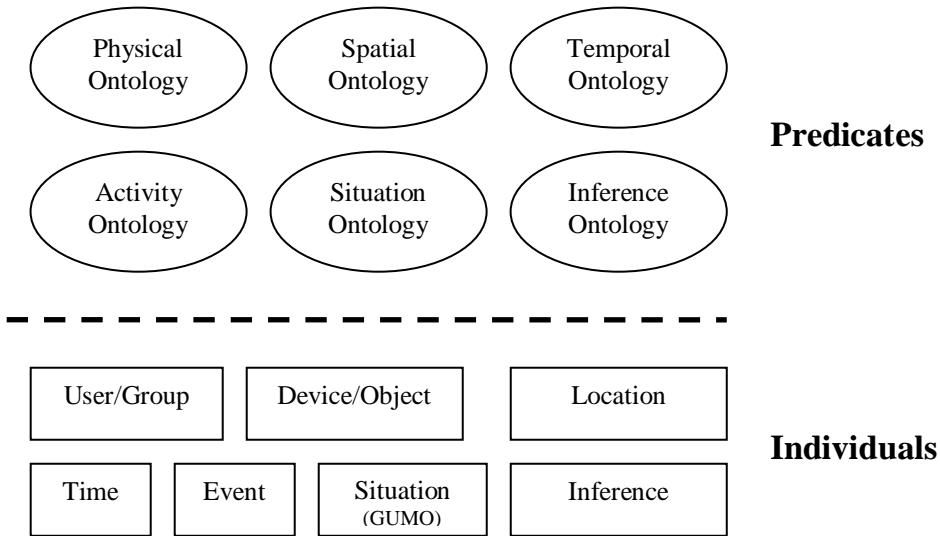
**Figure V.2.12.** The correlation between UbisWorld and real world

There are three main process relating to UbisWorld:

- UbisWorld is reduced and abstracted from real world.
- UbisWorld is modeled by modeling standards like ontologies and markup languages.
- UbisWorld is simulated and used by ubiquitous tools.

The author (Heckmann D. , 2005, p. 101) gave an example of UbisWorld in which there is a room in real world having a light switch and table. Such room is abstracted as UbisWorld room. The UbisTools can simulate the light-on, light-off behavior of the real room; thus if the virtual switch gets pressed, the virtual light in UbisWorld room will shine. The UbisWorld room can be used to monitor the real room; for example every time the real switch in real room gets pressed, the virtual light in UbisWorld room will shine. Moreover the UbisWorld room can control real room; for example that the virtual switch gets pressed causes the real light to shine.

UbisWorld can use many ontologies such as physical ontology, spatial ontology, temporal ontology, activity ontology, situation ontology and inference ontology (Heckmann D. , 2005, p. 102). The semantic web ontologies and languages like RDF and OWL support the integration of such UbisWorld ontologies. The attributes of situational statement in user model are specified in UbisWorld. Figure V.2.13 (Heckmann D. , 2005, p. 102) gives a combination of UbisWorld and ontologies.



**Figure V.2.13.** UbiWorld and ontologies

Ubiquitous user modeling system is implemented as ubiquitous user model service. Successive sub-section [V.2.4](#) describes architecture and main actions of ubiquitous user model service.

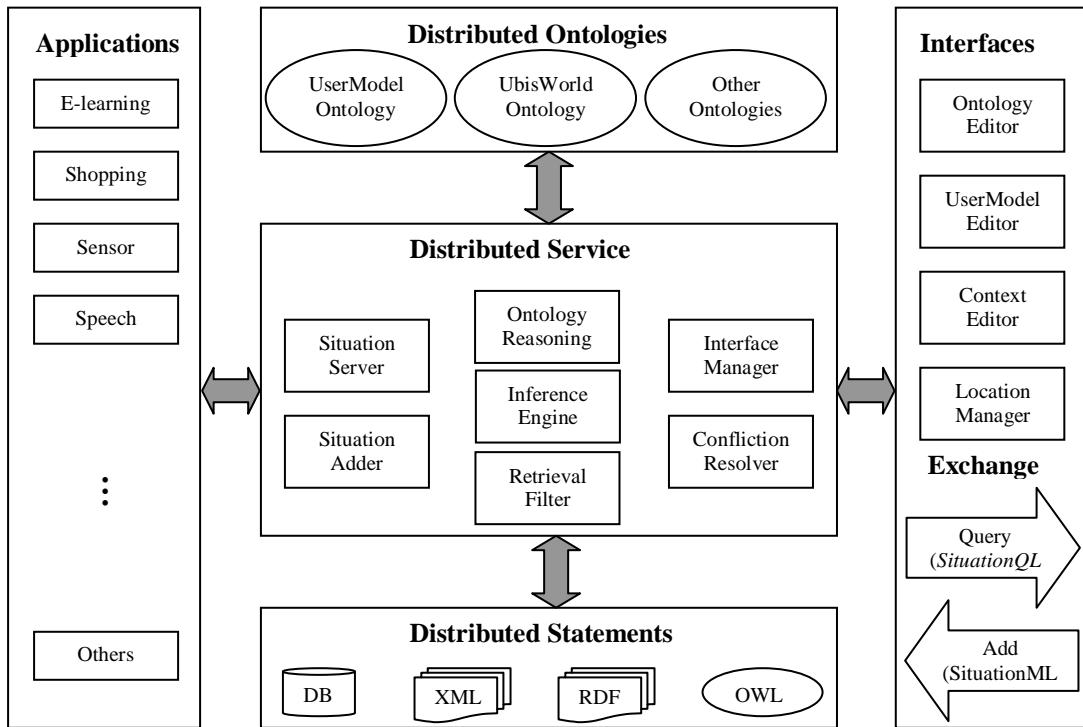
#### V.2.4. Ubiquitous user model service

*Ubiquitous user model service* is responsible for manipulating ubiquitous user model. It is essentially an implementation of ubiquitous user modeling system. According to (Heckmann D. , 2005, p. 155), ubiquitous user model service is an application independent server with a distributed approach for accessing and storing user information, the possibility to exchange and understand between different applications as well as adding privacy and transparency to the statements about user. The semantics of user information (situational statement) is mapped to the ontology GUMO. In brief, we call ubiquitous user model service as ubiquitous service in user modeling context.

The general architecture and work flow of ubiquitous user model service are mentioned in successive sub-sections [V.2.4.1](#) and [V.2.4.2](#), respectively.

##### V.2.4.1. Architecture of ubiquitous user model service

The architecture of ubiquitous user model service consists of five boxes: *Distributed Services box*, *Application box*, *Distributed Statements box*, *Distributed Ontologies box*, *Interfaces & Exchange box*. Figure [V.2.14](#) (Heckmann D. , 2005, p. 156) depicts this architecture together such five boxes.



**Figure V.2.14.** Architecture of ubiquitous user model service

**Distributed Services** box is the main box enriched by other boxes. It is constituted of internal modules (or services) such as *Situation Server* or *User Model Server*, *User Model Adder* or *Situation Adder*, *Retrieval Filter*, *Conflict Resolution*, *Inference Engine*, *Interface Engine* and *Ontology Reasoning* which are very necessary to perform ubiquitous user modeling tasks (Heckmann D. , 2005, pp. 155-156).

- Situation Server is the web server that manages the storage of statements about user.
- Situation Adder is the parser which analyzes incoming new statements and writes them to repositories.
- Retrieval Filter is responsible for controlling the retrieval of situational statements.
- Conflict Resolution is responsible for detecting and resolving possible conflicts.
- Inference Engine is the proactive engine that applies meta rules, writes new statements and triggers events.
- Interface Manager has a control mechanism that integrates user interfaces
- Ontology Reasoning is responsible for applying knowledge from various ontologies.

**Application box** lists applications that possibly cooperate with distributed services such as e-learning, sensors, airport service, and mobile phone (Heckmann D. , 2005, p. 156).

**Distributed Statements box** (Heckmann D. , 2005, pp. 156-157) plays the role of distributed database management system. Please read the book “Principles of Distributed Database Systems” (Özsu & Valduriez, 2011, pp. 1-38) for more details about distributed database management system. It is responsible for storing and managing situational statements about user. This module separates data from software. So the situation model is stored at here. It is considered that distributed statements box contains repositories of statements. These repositories are independent from the services which allow various services to operate independently on the same knowledge bases.

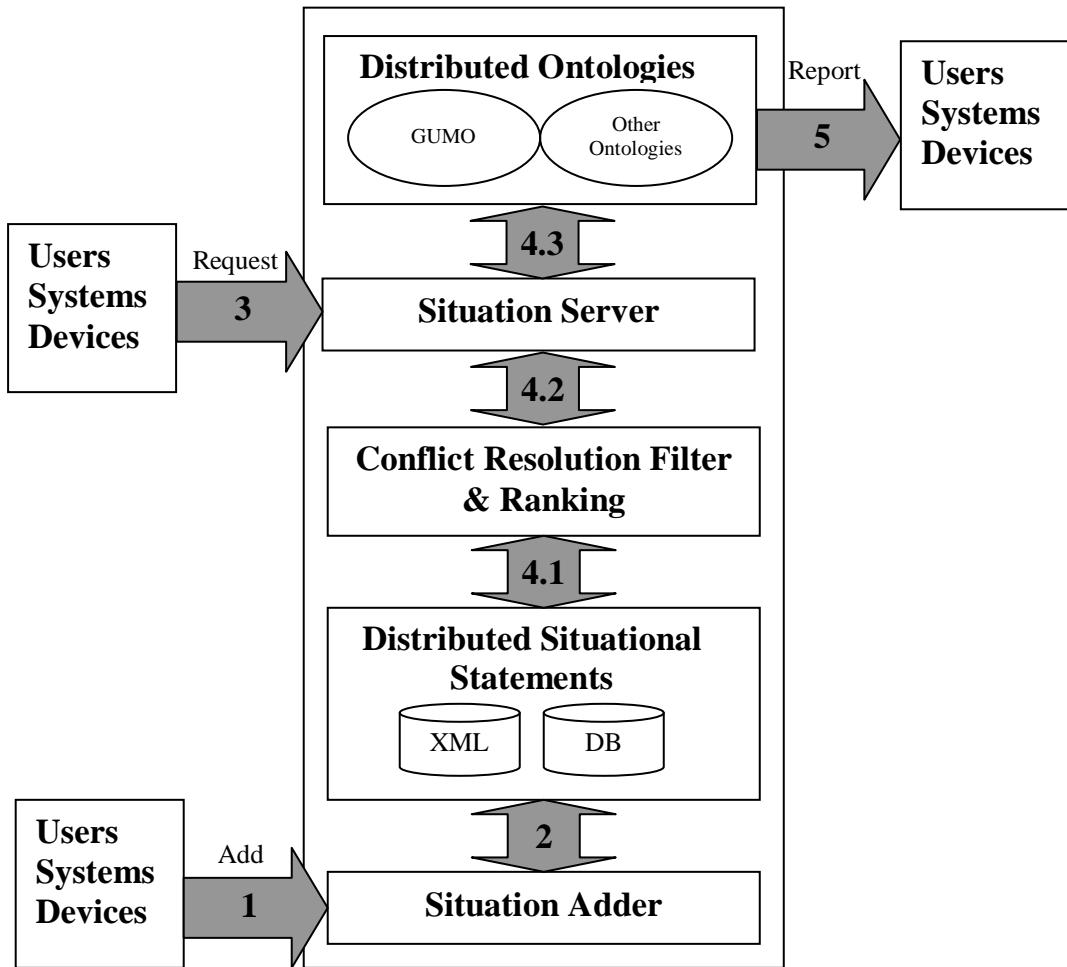
**Distributed ontologies box** (Heckmann D. , 2005, p. 157) is responsible for manipulating and integrating various ontologies into modeling service. These ontologies are used for the interpretation of statements, for the detection of conflicts and for the definition of expiry defaults and privacy defaults. This module separates the syntax of ontologies from semantics of ontologies.

**Interfaces & Exchange box** (Heckmann D. , 2005, p. 157) separates the user model service from the user interfaces and development tools. Each interface and tool can operate with different repositories, different ontologies and different services in distributed services box. It is very important for ubiquitous computing in the manner of computation at any time and at anywhere. There are some interfaces and tools such as *ontology editor*, *user model editor*, *context editor*, *UserML viewer*, *XForms viewer*, *location manager*, *strategy visualizer* are very necessary to assist user/administrator in manipulating or surveying ubiquitous user model (see figure [V.2.14](#)). The communication between user/administrator and distributed services is established through exchange tool. Thus the language *SituationQL* (or *UserQL*) is used to issue queries about user information (namely situational statements) and the language *SituationML* (or *UserML*) is used to answer such queries. Please read (Heckmann D. , 2005, pp. 124-128) for more details about *SituationQL* or *UserQL* and read (Heckmann D. , 2005, pp. 67-74) for more details about *SituationML* or *UserML*.

Derived from the general architecture shown in figure [V.2.14](#), the work flow of ubiquitous user model service is mentioned in successive sub-section [V.2.4.2](#).

#### **V.2.4.2. Main work flow of ubiquitous user model service**

Ubiquitous user model service supports three operations: *Add*, *Request* and *Report*. The work flow of them is shown in following figure [V.2.15](#):



**Figure V.2.15.** The work flow of ubiquitous user model service

The work flow includes below steps (Heckmann D. , 2005, pp. 157-158):

- The system, users or sensors add statements in form of *SituationML* (1)
- The statements are sent to module *Situation Adder* that analyzes the incoming data and distributes them to repositories in *Distributed Statements box* (2).
- The system, users or sensors request statements in form of *SituationQL* (3).
- Some repositories are chosen from *Distributed Statements box* in order to retrieve appropriate situational statements (4.1).
- The conflict resolution strategies are applied into such statements (4.2).
- Such statements are mapped to ontologies (4.3).
- Final statements are sent to users/system in form of *SituationML* (5).

Successive sub-section [V.2.5](#) focuses on future trend of [Triangular Learner Model](#) (TLM) and the proposed user modeling system [Zebra](#) in which Zebra is integrated into this ubiquitous user model service.

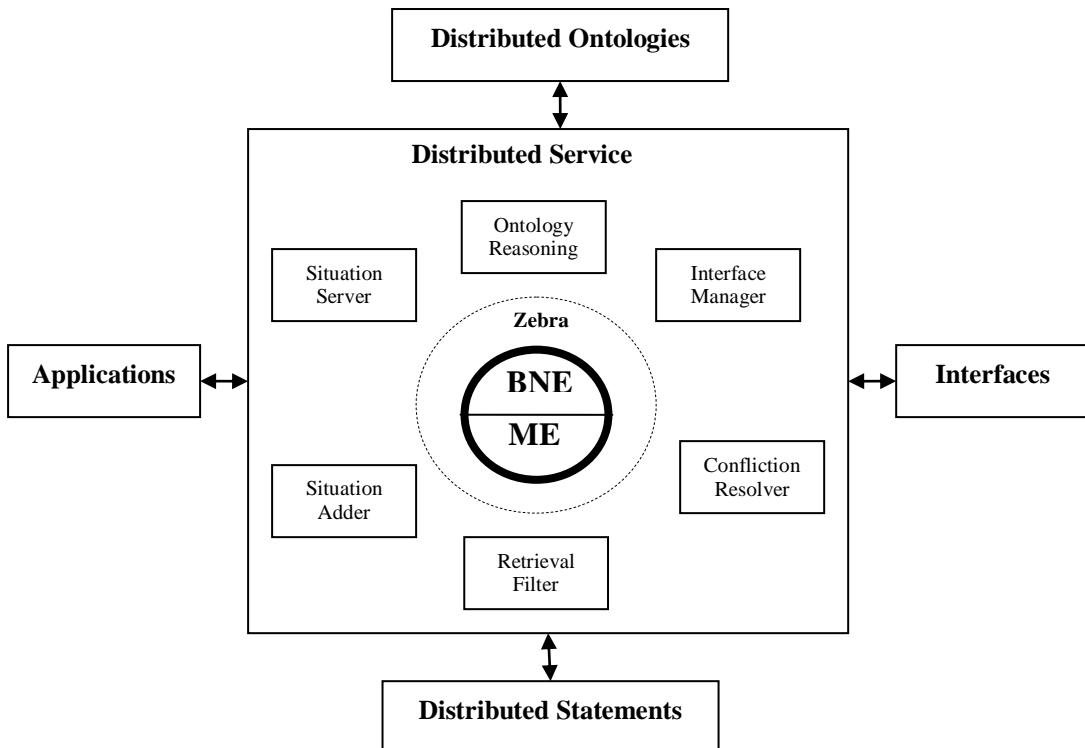
## V.2.5. Incorporating Zebra into ubiquitous user model service

When surveying the architecture of ubiquitous service, I recognize that such service lacks of the robust inference mechanism. The *Inference Engine* in

*Distributed Services box* only writes statements and triggers events. On the contrary, **Zebra** has two robust inference engines: *mining engine* and *belief network engine* with ability to infer new assumptions about user and to support personal recommendation. But Zebra don't support ubiquitous computing and user model in distributed environment. If the combination of Zebra and ubiquitous user model service is successful, there is a new user modeling system that takes advantage of both Zebra and ubiquitous computing. Thus it is possible to predict situational statement in ubiquitous environment.

So my idea is to incorporate Zebra into ubiquitous service by replacing *Inference Engine* (in *Distributed Services box*) with Zebra; figure V.2.16 expresses such incorporation. But this raises some obstacles that must be overcome.

- *Firstly*, this is how Zebra interacts with *Distributed Services box* of ubiquitous service when the database and data structures in Zebra such as Bayesian network, hidden Markov model, and sequential pattern are very different from statement repositories in ubiquitous service. Please see sub-sections III.1.1.1, III.2.4, and III.3.1.1 for more details about Bayesian network, hidden Markov model, and sequential pattern, respectively.
- *Secondly*, almost techniques in Zebra like data mining, artificial intelligence, and machine learning are unfamiliar to ubiquitous service.
- *Finally*, the most serious obstacle is that the output of inference process in Zebra such as new information about user and new personal recommendations is represented in the form which is incompatible with knowledge representation of situation model in ubiquitous service such as ontologies and RDF.

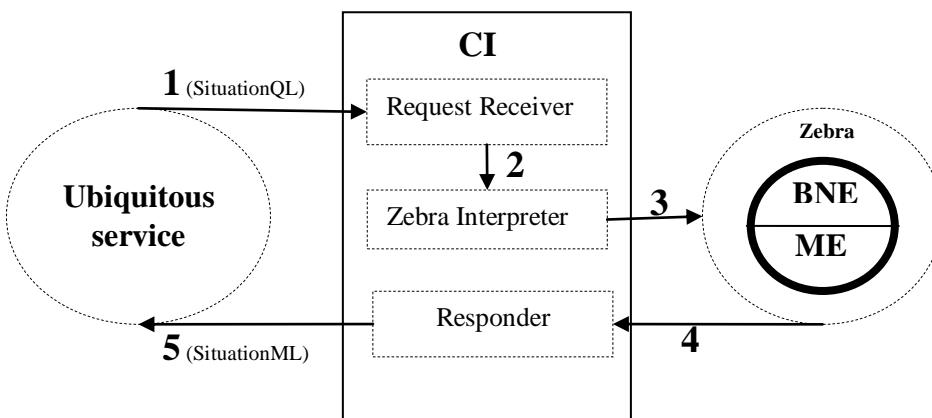


**Figure V.2.16.** Incorporating Zebra into ubiquitous service

These problems can be solved by getting the best out of the *communication interfaces* (**CI**) in architecture of Zebra. Note that each CI allows users to see or modify restrictedly their TLM. In addition, all outside applications interact with Zebra via these interfaces. So the ubiquitous service will interact with Zebra by special CI by request-response protocol including five following steps:

- (1) Ubiquitous service sends one request statement in form of *SituationQL* to CI.
- (2) CI interprets this statement into the data structure which inference engines of Zebra such as *mining engine* and *belief network engine* are aware of.
- (3) The request in form of data structure that Zebra knows is sent to Zebra.
- (4) Inference engines perform some concrete deduction tasks in order to take out some new assumptions, information, personal recommendations about user. This output is sent back CI.
- (5) CI interprets such output into a XML file (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) in form of *SituationML* which is readily understandable for ubiquitous service and sends it to ubiquitous service.

Figure V.2.17 interprets such five steps of communication protocol between Zebra and ubiquitous service.



**Figure V.2.17.** Request-Response communication protocol between Zebra and ubiquitous service

There is a big advantage of communication between Zebra and ubiquitous service when CI is implemented in web service standard; so ubiquitous service doesn't need to know what technologies inside Zebra are. The interoperation between CI and ubiquitous service is done via SOAP protocol (W3C, Simple Object Access Protocol (SOAP) 1.1, 2000). Ubiquitous service interacts with the Zebra in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP (Wikipedia, Hypertext Transfer Protocol, 2014) with an XML (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) serialization in conjunction with other web-related standards.

Now the research – “A User Modeling System for Adaptive Learning” has already been introduced to you. Thank you very much for your attention to the research.

Have a nice day!

## Related Publications

- Nguyen, L. (2014). A New Algorithm for Modeling and Inferring User's Knowledge by Using Dynamic Bayesian Network. (M. Z. Raqab, Ed.) *Statistics Research Letters (SRL)*, 3 (2).
- Nguyen, L. (2013). A New Approach for Modeling and Discovering Learning Styles by Using Hidden Markov Model. (G. Perry, Ed.) *Global Journal of Human Social Science: G - Linguistics & Education*, 13 (4 Version 1.0 Year 2013), 1-10.
- Nguyen, L. (2009). A Proposal Discovering User Interests by Support Vector Machine and Decision Tree on Document Classification. *The International Workshop on Social Networks Mining and Analysis for Business Applications (SNMABA2009) in conjunction with The 2009 IEEE International Conference on Social Computing (SocialCom)*. 4, pp. 809-814. Vancouver, Canada: Computational International Conference on Science and Engineering 2009 (CSE '09), IEEE Publisher.
- Nguyen, L. (2014). A User Modeling System for Adaptive Learning. (M. A. Chowdhury, Ed.) *Standard Scientific Research and Essays (SSRE)*, 2 (4), 65-209.
- Nguyen, L. (2010). Discovering User Interests by Document Classification. In I.-H. Ting, H.-J. Wu, T.-H. Ho, I.-H. Ting, H.-J. Wu, & T.-H. Ho (Eds.), *Mining and Analyzing Social Networks* (Vol. 288 In series "Studies in Computational Intelligence", pp. 139-159). Springer Berlin Heidelberg.
- Nguyen, L. (2014). Evaluating Adaptive Learning Model. *The 17th International Conference on Interactive Computer aided Learning (ICL2014), The 2014 World Engineering Education Forum, Engineering Education For A Global Community*. Dubai, UAE: IEEE Publisher.
- Nguyen, L. (2009). Incorporating Bayesian Inference into Adaptation Rules in AHA architecture. *Proceedings of 12th International Conference on Interactive Computer aided Learning (ICL2009)*. Villach, Austria: Kassel University Press, Kassel, Germany.
- Nguyen, L. (2013). Overview of Bayesian Network. *Science Journal of Mathematics and Statistics*, 2013, 22 pages.
- Nguyen, L. (2014). Specifying Prior Probabilities in Bayesian Network by Maximum Likelihood Estimation method. (B. N. Buszewski, Ed.) *Sylwan Journal © Copyright 2014*.

- Nguyen, L. (2009). State of the Art of Adaptive Learning. In H. R. Arabnia, A. Bahrami, & A. M. Solo (Ed.), *Proceedings of The 2009 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE 2009). The 2009 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'09)* (pp. 126-133). Las Vegas, Nevada, USA: CSREA Press USA.
- Nguyen, L. (2013). The Bayesian approach and suggested stopping criterion in Computerized Adaptive Testing. (A. T. Al-Taani, Ed.) *International Journal of Research in Engineering and Technology (IJRET)* , 2 (1-2103).
- Nguyen, L. (2014). User Model Clustering. (F. Shi, Ed.) *Journal of Data Analysis and Information Processing (JDAIP)* , 2 (2), 41-48.
- 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.
- Nguyen, L., & Do, P. (2009). Combination of Bayesian Network and Overlay Model in User Modeling. *Proceedings of 4th International Conference on Interactive Mobile and Computer Aided Learning (IMCL 2009)*. Amman, Jordan: Princess Sumaya University for Technology.
- Nguyen, L., & Do, P. (2009). Evolution of parameters in Bayesian Overlay Model. In H. R. Arabnia, D. d. Fuente, & J. A. Olivas (Ed.), *Proceedings of The 2009 International Conference on Artificial Intelligence (IC-AI'09), The 2009 World* (pp. 324-329). Monte Carlo Resort, Las Vegas, Nevada, USA: CSREA Press USA.
- Nguyen, L., & Do, P. (2008). Learner Model in Adaptive Learning. *World Academy of Science, Engineering and Technology* , 2 (2008), 395-400.
- Nguyen, L., & Do, P. (2009). Learning Concept Recommendation based on Sequential Pattern Mining. In E. Chang, F. Hussain, & E. Kayacan (Ed.), *Proceedings of The 2009 Third International Digital Ecosystems and Technologies Conference (IEEE-DEST 2009)* (pp. 66-71). Istanbul, Turkey: IEEE Publisher.

## References

- Agrawal, R., & Srikant, R. (1995). Mining Sequential Patterns. In P. S. Yu, & A. L. Chen (Ed.), *Proceedings of the Eleventh International Conference on Data Engineering (ICDE '95)* (pp. 3-14). Taipei, Taiwan: IEEE Computer Society Washington, DC, USA.
- Apache. (1999). *Apache Tomcat*. (The Apache Software Foundation) Retrieved 2008, from Apache Tomcat website: <http://tomcat.apache.org>
- Barwise, J., & Perry, J. (1981). Situations and Attitudes. *The Journal of Philosophy*, 78 (11 Seventy-Eighth Annual Meeting of the American Philosophical Association Eastern Division ), 668-691.
- Beaumont, I. H. (1994). User Modeling in the Interactive Anatomy Tutoring System ANATOM-TUTOR. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 4 (1), 21-45.
- 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.
- Boyd, S., & Vandenberghe, L. (2009). *Convex Optimization*. New York, NY, USA: Cambridge University Press.
- Brajnik, G., & Tasso, C. (1994). A Shell for Developing Non-monotonic User Modeling Systems. (B. R. Gaines, Ed.) *International Journal of Human-Computer Studies*, 40 (1), 31-62.
- Brusilovsky, P. (2001). Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11 (1-2), 87-110.
- Brusilovsky, P. (2004). KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. In S. Feldman, M. Uretsky, M. Najork, & C. Wills (Ed.), *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* (pp. 104-113). ACM New York, NY, USA.
- Brusilovsky, P. (1996). Methods and Techniques of Adaptive Hypermedia. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 6 (2-3), 87-129.
- Brusilovsky, P., & Millán, E. (2007). User Models for Adaptive Hypermedia and Adaptive Educational Systems. In P. Brusilovsky, A. Kobsa, W. Nejdl, P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The Adaptive Web* (pp. 3-53). Springer Berlin Heidelberg.

- Brusilovsky, P., Sosnovsky, S., & Shcherbinina, O. (2005). User Modeling in a Distributed E-Learning Architecture. In L. Ardissono, P. Brna, & A. Mitrovic (Ed.), *Proceedings of the 10th international conference on User Modeling* (pp. 387-391). Edinburgh, UK: Springer-Verlag Berlin, Heidelberg.
- 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). Using Bayesian Networks to Manage Uncertainty in Student Modeling. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 12 (4), 371-417.
- Czepiel, S. A. (2002). *Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation*. Czepiel's website <http://czep.net>.
- De Bra, P., & Calvi, L. (1998). AHA! An open Adaptive Hypermedia Architecture. (D. Tudhope, & D. Cunliffe, Eds.) *The New Review of Hypermedia and Multimedia*, 4 (1), 115-139.
- De Bra, P., Houben, G.-J., & Wu, H. (1999). AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In J. Westbomke, U. K. Wiil, & J. J. Leggett (Ed.), *Proceedings of 10th ACM Conference on Hypertext and hypermedia: eturning to our diverse roots: returning to our diverse roots (Hypertext '99)* (pp. 147-156). Darmstadt, Germany: ACM Press.
- De Bra, P., Smits, D., & Stash, N. (2006). The Design of AHA! In U. K. Wiil, P. J. Nürnberg, & J. Rubart (Ed.), *Proceedings of the seventeenth ACM Hypertext Conference on Hypertext and hypermedia (Hypertext '06)* (pp. 171-195). Odense, Denmark: ACM Press.
- De Bra, P., Stash, N., & Smits, D. (2005). Creating Adaptive Web-Based Applications. In L. Ardissono, P. Brna, & T. Mitrovic (Ed.), *Tutorial at the 10th International Conference on User Modeling*. Edinburgh, Scotland: Springer-Verlag Berlin, Heidelberg.
- Dunn, R., & Dunn, K. (1996). *The Dunn and Dunn Model*. (Online Learning Style Assessments & Community) Retrieved October 8, 2014, from The Official Site of Dunn and Dunn Learning Styles: <http://www.learningstyles.net/en/about-us>
- Felder, R. M., & Silverman, L. K. (1988). Learning and Teaching Styles in Engineering Education. *Journal of Engineering Education*, 78 (7), 674-681.

- Finin, T., & Drager, D. (1986, May 1). GUMS: A General User Modeling System. (R. Simpson, Ed.) *Proceedings of the workshop on Strategic computing natural language*, 224-230.
- Fink, J. (2004). *User Modeling Servers - Requirements, Design, and Evaluation* (Vol. 277 of Dissertations in Artificial Intelligence). Amsterdam, Netherlands: IOS Press.
- Fosler-Lussier, E. (1998). *Markov Models and Hidden Markov Models: A Brief Tutorial*. Technical Report TR-98-041, International Computer Science Institute, USA.
- Friedman, N., Murphy, K. P., & Russell, S. (1998). Learning the Structure of Dynamic Probabilistic Networks. In G. F. Cooper, & S. Moral (Ed.), *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* (pp. 139-147). Madison, Wisconsin, USA: Morgan Kaufmann Publishers.
- Fröschl, C. (2005). *User Modeling and User Profiling in Adaptive E-learning Systems*. Master Thesis, Graz University of Technology, Austria.
- Halasz, F. G., & Schwartz, M. (1990). The Dexter Hypertext Reference Model. In J. Moline, D. Benigni, & J. Baronas (Ed.), *Proceedings of the NIST Hypertext Standardization Workshop* (pp. 95-133). National Institute of Standards and Technology, Gaithersburg, Maryland, USA.
- Han, B. (2001). *Student Modelling and Adaptivity in web based Learning*. Master Thesis, Massey University, New Zealand.
- Han, J., & Kamber, M. (2006). *Data Mining: Concepts and Techniques* (2nd Edition ed.). (J. Gray, Ed.) San Francisco, CA, USA: Morgan Kaufmann Publishers, Elsevier.
- Han, J., Pei, J., & Yan, X. (2005). Sequential Pattern Mining by Pattern-Growth: Principles and Extensions. In W. Chu, & T. Y. Lin (Eds.), *Foundations and Advances in Data Mining* (pp. 183-220). Springer Berlin Heidelberg.
- Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., & Hsu, M.-C. (2000). FreeSpan: frequent pattern-projected sequential pattern mining. In R. Ramakrishnan, S. Stolfo, R. Bayardo, & I. Parsa (Ed.), *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '00)* (pp. 355-359). ACM New York, NY, USA.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (2nd Edition ed.). Springer New York.

- Heckmann, D. (2005). *Ubiquitous User Modeling*. Universität des Saarlandes. Künstlichen, German: Volume 297 Dissertationen zur Künstlichen Intelligenz, IOS Press.
- Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., & Wilamowitz-Moellendorff, M. v. (2005). Gumo – The General User Model Ontology. In L. Ardissono, P. Brna, A. Mitrovic, L. Ardissono, P. Brna, & A. Mitrovic (Eds.), *User Modeling 2005, Proceedings of the 10th International Conference, UM 2005* (pp. 428-432). Edinburgh, Scotland, UK: Springer Berlin Heidelberg.
- Henze, N. (2000). *Adaptive Hyperbooks: Adaptation for Project-Based Learning Resources*. PhD Thesis, University of Hannover.
- Henze, N. (2005). *Personalization for the World Wide Web*. Lecture Notes, Reasoning Web Summer School, University of Hannover, Germany.
- Henze, N., & Nejdl, W. (2004). A Logical Characterization of Adaptive Educational Hypermedia. (P. Brusilovsky, & P. De Bra, Eds.) *New Review of Hypermedia and Multimedia (NRHM)*, 10 (1), 77-113.
- Honey, P., & Mumford, A. (2000). *Learning Styles Helper's Guide*. Maidenhead, Berkshire, England: Peter Honey Publications Limited.
- Jia, Y.-B. (2013). *Lagrange Multipliers*. Lecture notes on course “Problem Solving Techniques for Applied Computer Science”, Iowa State University of Science and Technology, USA.
- Johansen, I. (2012, December 29). Graph software. *Graph version 4.4.2 build 543 (4.4.2 build 543)*. GNU General Public License.
- Karampiperis, P., & Sampson, D. (2005). Adaptive Learning Resources Sequencing in Educational Hypermedia Systems. (C.-K. Looi, & Kinshuk, Eds.) *Educational Technology & Society*, 8 (4), 128-147.
- Kay, J. (1995). The um Toolkit for Cooperative User Modelling. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 4 (3), 149-196.
- Kay, J. (2001). User Modeling for Adaptation. In C. Stephanidis, & G. Salvendy (Ed.), *User Interfaces for All: Concepts, Methods, and Tools* (Vol. IV, pp. 271-294). Mahwah, New Jersey, USA: Lawrence Erlbaum Associates, Inc.
- Kay, J., Kummerfeld, B., & Lauder, P. (2002). Personis: A Server for User Models. In P. De Bra, P. Brusilovsky, & R. Conejo (Ed.), *Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH '02)* (pp. 201-212). Springer Berlin Heidelberg.

- Kobsa, A. (1991). First experiences with the SB-ONE knowledge representation workbench in natural-language applications. (J. Doyle, Ed.) *ACM SIGART Bulletin - Special issue on implemented knowledge representation and reasoning systems*, 2 (3), 70-76.
- Kobsa, A. (2001). Generic User Modeling Systems. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 11 (1-2), 49-63.
- Kobsa, A. (2007). Generic User Modeling Systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: methods and strategies of web personalization* (Vol. I, pp. 136-154). Berlin, Heidelberg, New York: Springer Verlag.
- Kobsa, A. (1993). User Modeling: Recent Work, Prospects and Hazards. In M. Schneider-Hufschmidt, T. Kühme, & U. Malinowski, *Adaptive User Interfaces: Principles and Practice* (illustrated ed., Vol. 10, pp. 111-125). Amsterdam: North Holland Elsevier.
- Kobsa, A., & Pohl, W. (1995). The User Modeling Shell System BGP-MS. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 4 (2), 59-106.
- Kolb, A. Y., & Kolb, D. A. (2005). *The Kolb learning style inventory—version 3.1 2005 technical specifications*. Boston, MA, USA: Hay Resources Direct 200 (Hay Group).
- Kröse, B., & Smagt, P. v. (1996). *An Introduction to Neural Networks* (8th Edition ed.). Amsterdam, The Netherlands: University of Amsterdam.
- Lane, C. (2000). *Implementing Multiple Intelligences and Learning Styles in Distributed Learning/IMS Projects*. Technical Report, The Education Coalition (TEC), California, USA.
- Law, M. (2006). *A Simple Introduction to Support Vector Machines*. Lecture Notes for CSE 802 course, Michigan State University, USA, Department of Computer Science and Engineering.
- Lay, D. C. (2012). *Linear Algebra and its applications* (4th Edition ed.). (D. Lynch, Ed.) Boston, MA, USA: Pearson Education.
- Linden, W. J., & Pashley, P. J. (2002). Item Selection and Ability Estimation in Adaptive Testing. In W. J. Linden, G. A. Glas, W. J. Linden, & G. A. Glas (Eds.), *Computerized Adaptive Testing: Theory and Practice* (p. 323). Kluwer Academic Publishers.
- Lynch, S. M. (2007). *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*. Springer Berlin Heidelberg New York.

- Maffioletti, S. (2001). Requirements for an Ubiquitous Computing Infrastructure. *Cultura Narodov Prichernomoria journal*, 3, 35-40.
- Manning, C. D., Raghavan, P., & Schütze, H. (2009). *An Introduction to Information Retrieval* (Online Edition ed.). Cambridge, England, UK: Cambridge University Press.
- Martin, J., & VanLehn, K. (1995). 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.
- Medina, L. A., & Moll, V. H. (2009). The integrals in Gradshteyn and Ryzhik. Part 10: The digamma function. *Series A: Mathematical Sciences*, 17 (2009), 45-66.
- Mills, A. (1997). *Learning Dynamic Bayesian Networks*. Seminar Computational Intelligence F (708.116), Institute for Theoretical Computer Science, Graz University of Technology, Austria. Available from [http://www.igi.tugraz.at/lehre/SeminarF/WS05/CI\\_SeminarF\\_WS05.html](http://www.igi.tugraz.at/lehre/SeminarF/WS05/CI_SeminarF_WS05.html).
- Mitchell, T. M. (1997). *Machine Learning*. Burr Ridge, Illinois, USA: McGraw-Hill Science/Engineering/Math.
- Mitrovic, A. (1998). 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.
- Mödritscher, F., Garcia-Barrios, V. M., & Gütl, C. (2004). The Past, the Present and the Future of adaptive E-Learning. *Proceedings of the International Conference Interactive Computer Aided Learning (ICL2004)*.
- Montgomery, D. C., & Runger, G. C. (2003). *Applied Statistics and Probability for Engineers* (3rd Edition ed.). New York, NY, USA: John Wiley & Sons, Inc.
- Moodle, H. (2014, September 8). *About Moodle - the open source learning platform*, 2.7.2. (Moodle Headquarters) Retrieved September 18, 2014, from Moodle website: [https://docs.moodle.org/27/en/About\\_Moodle](https://docs.moodle.org/27/en/About_Moodle)

- Moore, A. W. (2001). *Support Vector Machines*. Carnegie Mellon University, USA, School of Computer Science. Available at <http://www.cs.cmu.edu/~awm/tutorials>.
- 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>
- Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD Thesis, University of California, Berkeley, USA.
- Neapolitan, R. E. (2003). *Learning Bayesian Networks*. Upper Saddle River, New Jersey, USA: Prentice Hall.
- Nguyen, L. (2014). A New Algorithm for Modeling and Inferring User's Knowledge by Using Dynamic Bayesian Network. (M. Z. Raqab, Ed.) *Statistics Research Letters (SRL)*, 3 (2).
- Nguyen, L. (2013). A New Approach for Modeling and Discovering Learning Styles by Using Hidden Markov Model. (G. Perry, Ed.) *Global Journal of Human Social Science: G - Linguistics & Education*, 13 (4 Version 1.0 Year 2013), 1-10.
- Nguyen, L. (2009). A Proposal Discovering User Interests by Support Vector Machine and Decision Tree on Document Classification. *The International Workshop on Social Networks Mining and Analysis for Business Applications (SNMABA2009) in conjunction with The 2009 IEEE International Conference on Social Computing (SocialCom)*. 4, pp. 809-814. Vancouver, Canada: Computational International Conference on Science and Engineering 2009 (CSE '09), IEEE Publisher.
- Nguyen, L. (2014). A User Modeling System for Adaptive Learning. (M. A. Chowdhury, Ed.) *Standard Scientific Research and Essays (SSRE)*, 2 (4), 65-209.
- Nguyen, L. (2010). Discovering User Interests by Document Classification. In I.-H. Ting, H.-J. Wu, T.-H. Ho, I.-H. Ting, H.-J. Wu, & T.-H. Ho (Eds.), *Mining and Analyzing Social Networks* (Vol. 288 In series "Studies in Computational Intelligence", pp. 139-159). Springer Berlin Heidelberg.
- Nguyen, L. (2014). Evaluating Adaptive Learning Model. *The 17th International Conference on Interactive Computer aided Learning (ICL2014), The 2014 World Engineering Education Forum, Engineering Education For A Global Community*. Dubai, UAE: IEEE Publisher.

- Nguyen, L. (2009). Incorporating Bayesian Inference into Adaptation Rules in AHA architecture. *Proceedings of 12th International Conference on Interactive Computer aided Learning (ICL2009)*. Villach, Austria: Kassel University Press, Kassel, Germany.
- Nguyen, L. (2013). Overview of Bayesian Network. *Science Journal of Mathematics and Statistics*, 2013, 22 pages.
- Nguyen, L. (2014). Specifying Prior Probabilities in Bayesian Network by Maximum Likelihood Estimation method. (B. N. Buszewski, Ed.) *Sylwan Journal © Copyright 2014*.
- Nguyen, L. (2009). State of the Art of Adaptive Learning. In H. R. Arabnia, A. Bahrami, & A. M. Solo (Ed.), *Proceedings of The 2009 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE 2009). The 2009 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'09)* (pp. 126-133). Las Vegas, Nevada, USA: CSREA Press USA.
- Nguyen, L. (2013). The Bayesian approach and suggested stopping criterion in Computerized Adaptive Testing. (A. T. Al-Taani, Ed.) *International Journal of Research in Engineering and Technology (IJRET)*, 2 (1-2103).
- Nguyen, L. (2014). User Model Clustering. (F. Shi, Ed.) *Journal of Data Analysis and Information Processing (JDAIP)*, 2 (2), 41-48.
- 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.
- Nguyen, L., & Do, P. (2009). Combination of Bayesian Network and Overlay Model in User Modeling. *Proceedings of 4th International Conference on Interactive Mobile and Computer Aided Learning (IMCL 2009)*. Amman, Jordan: Princess Sumaya University for Technology.
- Nguyen, L., & Do, P. (2009). Evolution of parameters in Bayesian Overlay Model. In H. R. Arabnia, D. d. Fuente, & J. A. Olivas (Ed.), *Proceedings of The 2009 International Conference on Artificial Intelligence (IC-AI'09), The 2009 World* (pp. 324-329). Monte Carlo Resort, Las Vegas, Nevada, USA: CSREA Press USA.

- Nguyen, L., & Do, P. (2008). Learner Model in Adaptive Learning. *World Academy of Science, Engineering and Technology*, 2 (2008), 395-400.
- Nguyen, L., & Do, P. (2009). Learning Concept Recommendation based on Sequential Pattern Mining. In E. Chang, F. Hussain, & E. Kayacan (Ed.), *Proceedings of The 2009 Third International Digital Ecosystems and Technologies Conference (IEEE-DEST 2009)* (pp. 66-71). Istanbul, Turkey: IEEE Publisher.
- Oracle. (n.d.). *Java language*. (Oracle Corporation) Retrieved December 25, 2014, from Java website: <https://www.oracle.com/java>
- Oracle. (2009). *The Java™ Tutorials*. Retrieved 2009, from Oracle Help Center website for Java Standard Edition: <http://docs.oracle.com/javase/tutorial>
- Orwant, J. L. (1991). *Doppelgänger: A User Modeling System*. Bachelor Thesis, Massachusetts Institute of Technology, USA.
- Orwant, J. L. (1995). Heterogeneous Learning in the Doppelgänger User Modeling System. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 4 (3), 107-130.
- Özsu, T. M., & Valduriez, P. (2011). *Principles of Distributed Database Systems* (3rd Edition ed.). Springer New York.
- Paiva, A., & Self, J. (1995). TAGUS - A User and Learner Modeling Workbench. (A. Kobsa, Ed.) *International Journal of User Modeling and User-Adapted Interaction*, 4 (3), 197-226.
- Pask, G. (1976). Styles and strategies of learning. *British journal of educational psychology*, 46 (2), 128-148.
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. (J. H. Trussell, Ed.) *Proceedings of the IEEE*, 77 (2), 257-286.
- Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems* (3rd Edition ed.). New York, USA: McGraw-Hill Higher Education.
- Reye, J. (2004). Student Modelling based on Belief Networks. *International Journal of Artificial Intelligence in Education (IJAIED)*, 14 (1), 63-96.
- Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). *Recommender Systems Handbook* (Vol. I). (F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor, Eds.) Springer New York Dordrecht Heidelberg London.
- Rich, E. (1979). User Modeling via Stereotypes. *Cognitive Science*, 3 (4), 329-354.

- Riding, R., & Rayner, S. (1998). *Cognitive Styles and Learning Strategies: Understanding Style Differences in Learning Behaviour* (illustrated, reprint ed.). London, England, UK: David Fulton Publishers.
- Rios, D. (n.d.). *Introduction to Neural Networks*. Retrieved 2009, from Neuro AI website: <http://www.learnartificialneuralnetworks.com/introduction-to-neural-networks.html>
- Rosen, K. H. (2012). *Discrete Mathematics and Its Applications* (7nd Edition ed.). (M. Lange, Ed.) McGraw-Hill Companies.
- Rudner, L. M. (1998, November). *An On-line, Interactive, Computer Adaptive Testing Tutorial*. Retrieved 2009, from Lawrence M. Rudner's website: <http://edres.org/scripts/cat>
- Russell, S. J., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach* (2nd Edition ed.). Upper Saddle River, New Jersey, USA: Pearson Education, Inc.
- Sakai, P. (2014, July 8). *About Sakai Project*, 10. (Apereo Foundation) Retrieved September 18, 2014, from Sakai Project website: <https://sakaiproject.org/about-us>
- Schmolze, J. G. (2001). *An Introduction to Hidden Markov Models*. Lecture Notes on course "COMP 232-Knowledge Based Systems", Department of Computer Science, Tufts University.
- Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In P. M. Apers, M. Bouzeghoub, & G. Gardarin (Ed.), *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '96)* (pp. 3-17). Springer-Verlag London, UK.
- Stash, N. (2007). *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*. Eindhoven University of Technology, Netherlands. ACM SIGWEB Newsletter.
- Stash, N., Cristea, A. I., & De Bra, P. (2007). Adaptation languages as vehicles of explicit intelligence in Adaptive Hypermedia. (I. Hatzilygeroudis, Ed.) *International Journal of Continuing Engineering Education and Life-Long Learning*, 17 (4), 319-336.
- Stash, N., Cristea, A., & De Bra, P. (2005). Explicit Intelligence in Adaptive Hypermedia: Generic Adaptation Languages for Learning Preferences and Styles. In I. Hatzilygeroudis (Ed.), *Proceedings International Workshop on Combining Intelligent and Adaptive Hypermedia Methods/Techniques in Web-Based Education (CIAH'05 in conjunction with HT'05)* (pp. 75-84). Patras, Greece: University of Patras.

- Urban-Lurain, M. (2002). *Intelligent Tutoring Systems: An Historic Review in the Context of the Development of Artificial Intelligence and Educational Psychology*. Lecture Notes on course CSE 101, Michigan State University, Computer Science and Engineering.
- Vergara, H. (1994). *PROTUM: A Prolog Based Tool for User Modeling*. WIS Memo 10, WG Knowledge-based Information Systems, Department of Information Science, University of Konstanz, Germany.
- Vermunt, J. D. (1996). Metacognitive, Cognitive and Affective Aspects of Learning Styles and Strategies: a Phenomenon graphic Analysis. *Higher education, special issue Individual Diversity in Effective Studying*, 31 (1), 25-50.
- W3C. (2008, November 26). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, 1.0. (T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, Editors, & World Wide Web Consortium) Retrieved September 17, 2014, from W3C website: <http://www.w3.org/TR/2008/REC-xml-20081126/>
- W3C. (2000, May 8). *Simple Object Access Protocol (SOAP) 1.1*, 1.1. (D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, et al., Editors, & World Wide Web Consortium) Retrieved September 17, 2014, from W3C website: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>
- W3C. (2001). *W3C semantic web activity*. Retrieved 2009, from W3C website: <http://www.w3.org/2001/sw>
- W3C. (2004, February 11). *Web Services Architecture*. (H. Haas, A. Brown, Editors, & World Wide Web Consortium) Retrieved September 17, 2014, from W3C website: <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>
- W3C. (2001, March 15). *Web Services Description Language (WSDL) 1.1*, 1.1. (E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Editors, & World Wide Web Consortium) Retrieved September 17, 2014, from W3C website: <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- W3C. (2010, November 23). *XHTML™ 1.1 - Module-based XHTML - Second Edition*, 1.1. (S. McCarron, M. Ishikawa, M. Altheim, S. McCarron, Editors, & World Wide Web Consortium) Retrieved September 28, 2014, from W3C website: <http://www.w3.org/TR/2010/REC-xhtml11-20101123>
- W3Schools. (1999). *HTML(5) Tutorial*. Retrieved 2014, from W3Schools website: <http://www.w3schools.com/html>

- Weisstein, E. W. (n.d.). *Dilogarithm*. (Wolfram Research) Retrieved December 17, 2014, from MathWorld--A Wolfram Web Resource: <http://mathworld.wolfram.com/Dilogarithm.html>
- Weisstein, E. W. (n.d.). *Euler-Mascheroni Constant*. (Wolfram Research) Retrieved November 13, 2014, from MathWorld--A Wolfram Web Resource: <http://mathworld.wolfram.com/Euler-MascheroniConstant.html>
- Weisstein, E. W. (n.d.). *Polygamma Function*. (Wolfram Research) Retrieved November 13, 2014, from MathWorld--A Wolfram Web Resource: <http://mathworld.wolfram.com/PolygammaFunction.html>
- Weisstein, E. W. (n.d.). *Trigamma Function*. (Wolfram Research) Retrieved November 13, 2014, from MathWorld--A Wolfram Web Resource: <http://mathworld.wolfram.com/TrigammaFunction.html>
- Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. (J. T. Ritschdorff, Ed.) San Francisco, California, USA: Morgan Kaufmann Publishers Inc.
- Wikibooks. (2008, January 1). *Support Vector Machines*. Retrieved 2008, from Wikibooks website: [http://en.wikibooks.org/wiki/Support\\_Vector\\_Machines](http://en.wikibooks.org/wiki/Support_Vector_Machines)
- Wikipedia. (2009, January 4). *Artificial neural network*. (Wikimedia Foundation) Retrieved 2009, from Wikipedia website: [http://en.wikipedia.org/wiki/Artificial\\_neural\\_network](http://en.wikipedia.org/wiki/Artificial_neural_network)
- Wikipedia. (2006). *Bayesian inference*. (Wikimedia Foundation) Retrieved 2007, from Wikipedia website: [http://en.wikipedia.org/wiki/Bayesian\\_inference](http://en.wikipedia.org/wiki/Bayesian_inference)
- Wikipedia. (2014, October 16). *Beta function*. (Wikimedia Foundation) Retrieved November 9, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Beta\\_function](http://en.wikipedia.org/wiki/Beta_function)
- Wikipedia. (2014, August 9). *Common Object Request Broker Architecture*. (Wikimedia Foundation) Retrieved September 16, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Common\\_Object\\_Request\\_Broker\\_Architecture](http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture)
- Wikipedia. (2014, December 7). *Daemon (computing)*. (Wikimedia Foundation) Retrieved December 25, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Daemon\\_\(computing\)](http://en.wikipedia.org/wiki/Daemon_(computing))

- Wikipedia. (2014, September 6). *David A. Kolb*. (Wikimedia Foundation) Retrieved October 8, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/David\\_A.\\_Kolb](http://en.wikipedia.org/wiki/David_A._Kolb)
- Wikipedia. (2014, October 12). *Digamma function*. (Wikimedia Foundation) Retrieved November 9, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Digamma\\_function](http://en.wikipedia.org/wiki/Digamma_function)
- Wikipedia. (2014, September 14). *Hypertext Transfer Protocol*. (Wikimedia Foundation) Retrieved September 15, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)
- Wikipedia. (2014, October 1). *Indeterminate form*. (Wikimedia Foundation) Retrieved November 11, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Indeterminate\\_form](http://en.wikipedia.org/wiki/Indeterminate_form)
- Wikipedia. (2014, October 3). *Internet*. (Wikimedia Foundation) Retrieved October 7, 2014, from Wikipedia website: <http://en.wikipedia.org/wiki/Internet>
- Wikipedia. (2014, September 12). *Java Servlet*. (Wikimedia Foundation) Retrieved September 15, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Java\\_Servlet](http://en.wikipedia.org/wiki/Java_Servlet)
- Wikipedia. (2014, August 4). *Karush–Kuhn–Tucker conditions*. (Wikimedia Foundation) Retrieved November 16, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Karush–Kuhn–Tucker\\_conditions](http://en.wikipedia.org/wiki/Karush–Kuhn–Tucker_conditions)
- Wikipedia. (2014, September 11). *Learning management system*. (Wikimedia Foundation) Retrieved September 18, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Learning\\_management\\_system](http://en.wikipedia.org/wiki/Learning_management_system)
- Wikipedia. (2014, September 12). *Lightweight Directory Access Protocol*. (Wikimedia Foundation) Retrieved September 16, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Lightweight\\_Directory\\_Access\\_Protocol](http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol)
- Wikipedia. (2014, September 21). *Lisp (programming language)*. (Wikimedia Foundation) Retrieved October 6, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Lisp\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Lisp_(programming_language))
- Wikipedia. (2014, September 3). *Mutual information*. (Wikimedia Foundation) Retrieved September 16, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Mutual\\_information](http://en.wikipedia.org/wiki/Mutual_information)
- Wikipedia. (2014, September 29). *Myers-Briggs Type Indicator*. (Wikimedia Foundation) Retrieved October 8, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Myers-Briggs\\_Type\\_Indicator](http://en.wikipedia.org/wiki/Myers-Briggs_Type_Indicator)

- Wikipedia. (2014, August 28). *Ontology (information science)*. (Wikimedia Foundation) Retrieved September 17, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Ontology\\_\(information\\_science\)](http://en.wikipedia.org/wiki/Ontology_(information_science))
- Wikipedia. (2014, October 12). *Polygamma function*. (Wikimedia Foundation) Retrieved November 10, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Digamma\\_function](http://en.wikipedia.org/wiki/Digamma_function)
- Wikipedia. (2014, August 22). *Polylogarithm*. (Wikimedia Foundation) Retrieved December 17, 2014, from Wikipedia website: <http://en.wikipedia.org/wiki/Polylogarithm>
- Wikipedia. (2014, September 23). *Prolog*. (Wikimedia Foundation) Retrieved September 30, 2014, from Wikipedia website: <http://en.wikipedia.org/wiki/Prolog>
- Wikipedia. (2014, October 29). *Sample (statistics)*. (Wikimedia Foundation) Retrieved October 31, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Sample\\_\(statistics\)](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\\_\(mathematics\)](http://en.wikipedia.org/wiki/Set_(mathematics))
- Wikipedia. (2014, November 4). *Support vector machine*. (Wikimedia Foundation) Retrieved November 16, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Support\\_vector\\_machine](http://en.wikipedia.org/wiki/Support_vector_machine)
- Wikipedia. (2014, October 5). *Web page*. (Wikimedia Foundation) Retrieved October 7, 2014, from Wikipedia website: [http://en.wikipedia.org/wiki/Web\\_page](http://en.wikipedia.org/wiki/Web_page)
- Wikipedia. (2014, October 6). *Zebra*. (Wikimedia Foundation) Retrieved October 7, 2014, from Wikipedia website: <http://en.wikipedia.org/wiki/Zebra>
- Witkin, H. A., Moore, C. A., Goodenough, D. R., & Cox, P. W. (1977). Field-dependent and Field-independent Cognitive Styles and Their Educational Implications. (S. Messick, Ed.) *Review of Educational Research*, 47 (1), 1-64.
- WolframAlpha. (n.d.). *Wolfram Alpha Mathematics Engine*. (Wolfram Alpha LLC-A Wolfram Research Company) Retrieved December 16, 2014, from WolframAlpha: <http://www.wolframalpha.com>
- Zaki, M. J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. (D. Fisher, Ed.) *Machine Learning*, 42 (1-2), 31-60.

- Zivot, E. (2009). *Maximum Likelihood Estimation*. Lecture Notes on course "Econometric Theory I: Estimation and Inference (first quarter, second year PhD)", University of Washington, Seattle, Washington, USA.

# Appendix

## A. List of Tables

<b>Table I.1.1.</b> Some common learning styles .....	5
<b>Table I.1.2.</b> Eight forms of aptitudes.....	5
<b>Table I.3.1.</b> CPT of a YACF cluster node .....	31
<b>Table I.3.2.</b> CPT of a child node $Y$ given cluster node $H$ .....	32
<b>Table I.3.3.</b> Conditional probability table of $relevantSS_{c,p}$ given $relevantIS_{c,p}$	44
<b>Table III.1.1.</b> Prior probabilities (CPT) of nodes $C$ , $O$ and $I$ .....	104
<b>Table III.1.2.</b> CPT of node $J$ .....	105
<b>Table III.1.3.</b> CPT (s) of evidence nodes $E$ and $Q$ .....	105
<b>Table III.1.4.</b> ECA and CA rules for Bayesian inference .....	119
<b>Table III.1.5.</b> Evidence sample corresponding to 5 trials (sample of size 5)	136
<b>Table III.1.6.</b> Posterior density functions calculated based on count numbers $s_{ij}$ and $t_{ij}$ .....	136
<b>Table III.1.7.</b> Updated CPT (s) of $X_1$ and $X_2$ .....	136
<b>Table III.1.8.</b> Evidence sample with data missing.....	138
<b>Table III.1.9.</b> New split evidences for missing data .....	138
<b>Table III.1.10.</b> Complete evidence sample in E-step of EM algorithm.....	139
<b>Table III.1.11.</b> Counters $s_{11}$ , $t_{11}$ , $s_{21}$ , $t_{21}$ , $s_{22}$ , and $t_{22}$ from estimated values (of missing values).....	139
<b>Table III.1.12.</b> Posterior density functions and posterior probabilities are updated in M-step of EM algorithm.....	140
<b>Table III.1.13.</b> All variables and their density functions, prior probabilities	143
<b>Table III.1.14.</b> All parameters of prior density functions .....	145
<b>Table III.1.15.</b> Incomplete sample with two evidences ( $E=1$ , $Q=1$ ) and ( $E=0$ , $Q=0$ ).....	146
<b>Table III.1.16.</b> New split evidences for missing values of $O$ , $I$ , and $J$ .....	146
<b>Table III.1.17.</b> Complete sample with two evidences ( $E=1$ , $Q=1$ ) and ( $E=1$ , $Q=0$ ).....	149
<b>Table III.1.18.</b> Counters $s_{ij}$ and $t_{ij}$ (s) from estimated values .....	150
<b>Table III.1.19.</b> Posterior density functions and posterior probabilities are evolved based on counters $s_{ij}$ and $t_{ij}$ .....	151
<b>Table III.1.20.</b> Six steps of new algorithm for modeling and inferring user's knowledge by using DBN .....	158
<b>Table III.1.21.</b> The weights relating $x_1[t]$ are normalized.....	163
<b>Table III.1.22.</b> CPT (s) of $X[t-1] = \{x_1[t-1], x_2[t-1], x_3[t-1]\}$ .....	165
<b>Table III.1.23.</b> CPT (s) of $X[t] = \{x_1[t], x_2[t], x_3[t], d_1[t]\}$ .....	166
<b>Table III.1.24.</b> The results of probabilistic inference – posterior probabilities .....	170
<b>Table III.1.25.</b> Iterative algorithm solving simplest equations specified by formula III.1.73 for estimating parameters $a$ and $b$ .....	190
<b>Table III.1.26.</b> The evidences corresponding to 5 trials .....	191

---

<b>Table III.1.27.</b> The values of $L_{ij}$ corresponding to beta density function $\beta_1$ , $\beta_2$ , and $\beta_3$ .....	192
<b>Table III.1.28.</b> The normal biases of $(a_1, b_1)$ with respect to $\beta_1$ .....	193
<b>Table III.1.29.</b> The normal biases of $(a_2, b_2)$ with respect to $\beta_2$ .....	193
<b>Table III.1.30.</b> The normal biases of $(a_3, b_3)$ with respect to $\beta_3$ .....	194
<b>Table III.1.31.</b> The evidences corresponding to 5 trials .....	198
<b>Table III.1.32.</b> The normal biases of $(a_1, b_1)$ with respect to $\beta_1$ .....	199
<b>Table III.1.33.</b> The normal biases of $(a_2, b_2)$ with respect to $\beta_2$ .....	200
<b>Table III.1.34.</b> The normal biases of $(a_3, b_3)$ with respect to $\beta_3$ .....	200
<b>Table III.2.1.</b> Transition probability matrix $A$ .....	214
<b>Table III.2.2.</b> Uniform initial state distribution $\Pi$ .....	214
<b>Table III.2.3.</b> Observation probability matrix $B$ .....	214
<b>Table III.2.4.</b> Transition probability matrices: $A_1, A_2, A_3$ .....	216
<b>Table III.2.5.</b> Observation probability matrices: $B_1, B_2, B_3$ .....	218
<b>Table III.2.6.</b> Learning objects selected.....	220
<b>Table III.2.7.</b> Sequence of student observations .....	220
<b>Table III.2.8.</b> Sequence of state transitions .....	220
<b>Table III.3.1.</b> Learning sessions → learning sequences .....	225
<b>Table III.3.2.</b> Large itemsets are mapped to integers .....	227
<b>Table III.3.3.</b> Transformed sequences .....	227
<b>Table III.3.4.</b> Candidate generation .....	228
<b>Table III.3.5.</b> Cracking sequence database into vertical format data sets ....	230
<b>Table III.3.6.</b> (1). frequent itemsets ( $o$ ) and ( $f$ ) are associated with pairs ( $sid$ , $eid$ ). (2). 2-sequence $\langle of \rangle$ are found by joining itemsets ( $o$ ), ( $f$ ) in (1).....	230
<b>Table III.3.7.</b> Frequent sequences mined from projected databases .....	231
<b>Table III.3.8.</b> Sequential rules .....	232
<b>Table III.3.9.</b> Recommended learning concepts constructed from sequential rules .....	233
<b>Table III.3.10.</b> Proposed approach to discover user interests .....	234
<b>Table III.3.11.</b> Proposed iterative method to determine optimal bias $b^*$ .....	245
<b>Table III.3.12.</b> $k$ couple $(W_i^*, b_i^*)$ corresponds with $k$ class $\{c_1, c_2, \dots, c_k\}$ ....	245
<b>Table III.3.13.</b> Classifying document $D$ with multi-classes.....	246
<b>Table III.3.14.</b> Term frequencies of documents (SVM) .....	246
<b>Table III.3.15.</b> Training corpus (SVM).....	247
<b>Table III.3.16.</b> Execution of proposed iterative method for enhancing the bias $b^*$ .....	249
<b>Table III.3.17.</b> Training corpus – Term frequencies of documents (decision tree).....	251
<b>Table III.3.18.</b> Training corpus – Normalized term frequencies (decision tree) .....	251
<b>Table III.3.19.</b> Training corpus – Nominal term frequencies .....	252
<b>Table III.3.20.</b> Reduced training corpus .....	257
<b>Table III.3.21.</b> Information gains at the second splitting.....	258
<b>Table III.3.22.</b> Classification rules derived from decision tree induction ....	259
<b>Table III.3.23.</b> Back-propagation algorithm for learning neural network ....	265

<b>Table III.3.24.</b> Training corpus – Term frequencies of documents (neural network).....	266
<b>Table III.3.25.</b> Training corpus – Normalized term frequencies (neural network).....	266
<b>Table III.3.26.</b> Results from training process based on back-propagation algorithm.....	269
<b>Table III.3.27.</b> User's searching history.....	271
<b>Table III.3.28.</b> 1-itemsets .....	272
<b>Table III.3.29.</b> Maximum frequent itemset that user searches.....	272
<b>Table III.3.30.</b> Interesting document vector.....	272
<b>Table III.3.31.</b> Interesting document vector is normalized .....	272
<b>Table III.3.32.</b> Nominal interesting document vector .....	273
<b>Table III.3.33.</b> Values of graph nodes .....	279
<b>Table III.3.34.</b> Values of nodes in Bayesian network .....	283
<b>Table III.3.35.</b> Four user models .....	285
<b>Table III.3.36.</b> Cosine similarities between user vectors and medoids .....	289
<b>Table III.3.37.</b> Cosine similarities between user vectors and medoids at the second running time of $k$ -medoid algorithm .....	291
<b>Table III.3.38.</b> Cosine similarities between user vectors and medoids at the third running time of $k$ -medoid algorithm.....	293
<b>Table IV.1.1.</b> Computerized adaptive testing (CAT) algorithm.....	304
<b>Table IV.1.2.</b> Advanced CAT algorithm.....	323
<b>Table IV.1.3.</b> A multi-user test with 5 items and 4 examinees .....	324
<b>Table IV.1.4.</b> Results of multi-user test with 5 items and 4 examinees .....	328
<b>Table IV.2.1.</b> Rating matrix as collection of users' feedbacks.....	336
<b>Table IV.2.2.</b> Rating matrix is shrunk by projecting it onto its eigenvectors	336
<b>Table V.2.1.</b> Main part box .....	352
<b>Table V.2.2.</b> Situation box .....	352
<b>Table V.2.3.</b> Explanation box .....	353
<b>Table V.2.4.</b> Privacy box .....	353
<b>Table V.2.5.</b> Administrator box .....	354
<b>Table V.2.6.</b> Basic user dimensions in GUMO .....	357
<b>Table V.2.7.</b> User model interest categories in GUMO .....	357

## B. List of Figures

<b>Figure I.1.1.</b> User modeling.....	2
<b>Figure I.1.2.</b> Learner profile and learner model in adaptation .....	3
<b>Figure I.1.3.</b> An example of stereotypes of Java learner .....	7
<b>Figure I.1.4.</b> An example of overlay model having six concepts .....	8
<b>Figure I.1.5.</b> Illustration of perturbation model .....	9
<b>Figure I.2.1.</b> General Architecture of ITS .....	15
<b>Figure I.2.2.</b> User modeling component in ANATOM-TUTOR .....	17
<b>Figure I.2.3.</b> Adaptive representation.....	18
<b>Figure I.2.4.</b> Adaptive navigation .....	18
<b>Figure I.2.5.</b> General architecture of AEHS .....	20
<b>Figure I.2.6.</b> AHAM – The architecture of AHA!.....	23
<b>Figure I.2.7.</b> Implemented framework of AHA! .....	23
<b>Figure I.2.8.</b> AHA! engine.....	24
<b>Figure I.3.1.</b> An example of dependency graph .....	27
<b>Figure I.3.2.</b> The levels of <i>KI</i> (s).....	28
<b>Figure I.3.3.</b> Root tree .....	28
<b>Figure I.3.4.</b> Clustering approach .....	29
<b>Figure I.3.5.</b> YACF method.....	30
<b>Figure I.3.6.</b> Cluster node .....	31
<b>Figure I.3.7.</b> Student modeling in Andes .....	33
<b>Figure I.3.8.</b> Sample problem to find normal force .....	34
<b>Figure I.3.9.</b> Solution graph in Andes .....	35
<b>Figure I.3.10.</b> Relationship between rule and context-rule nodes .....	36
<b>Figure I.3.11.</b> Relationship between nodes in task-specific part.....	38
<b>Figure I.3.12.</b> Mutually exclusive strategy .....	38
<b>Figure I.3.13.</b> Strategy node .....	39
<b>Figure I.3.14.</b> Prior/Posterior probabilities in the task-specific part .....	40
<b>Figure I.3.15.</b> Architecture of SQL-Tutor .....	42
<b>Figure I.3.16.</b> Bayesian network in SQL-Tutor .....	44
<b>Figure II.1.</b> The architecture of GUMS .....	49
<b>Figure II.2.</b> The architecture of BGP-MS .....	51
<b>Figure II.3.</b> The architecture of CUMMULATE.....	53
<b>Figure II.4.</b> The architecture of Personis .....	54
<b>Figure II.5.</b> The architecture of LDAP-UMS.....	55
<b>Figure II.6.</b> Triangular Learner Model.....	57
<b>Figure II.7.</b> extended Triangular Learner Model.....	59
<b>Figure II.8.</b> Modeling task is similar to profile mining task .....	59
<b>Figure II.9.</b> The modeling process in Zebra .....	60
<b>Figure II.10.</b> The architecture of Zebra.....	62
<b>Figure II.11.</b> The new architecture of AEHS and the interaction between AEHS and Zebra .....	63
<b>Figure II.12.</b> Steps in adaptation process with support of Zebra .....	65
<b>Figure II.13.</b> Zebra is running .....	66
<b>Figure II.14.</b> Zebra control panel.....	67

<b>Figure II.15.</b> Monitoring a learner .....	68
<b>Figure II.16.</b> TLM of given learner .....	69
<b>Figure II.17.</b> Learning style sub-model constructed by hidden Markov model .....	70
<b>Figure II.18.</b> Learning history sub-model stores coarse information in XML files .....	71
<b>Figure II.19.</b> Learning concept recommendation .....	72
<b>Figure II.20.</b> Learning path recommendation .....	73
<b>Figure II.21.</b> Discovering user interests based on document classification.....	73
<b>Figure II.22.</b> Constructing user groups .....	74
<b>Figure II.23.</b> Evaluation of learner's knowledge.....	75
<b>Figure II.24.</b> Evaluation of learner's knowledge in detailed .....	76
<b>Figure II.25.</b> Updating personal information .....	77
<b>Figure II.26.</b> Learning report .....	78
<b>Figure II.27.</b> Mailing list tool .....	79
<b>Figure II.28.</b> Feedback report .....	79
<b>Figure II.29.</b> WOW login web page .....	81
<b>Figure II.30.</b> Typical adaptive learning web page supported by WOW.....	82
<b>Figure II.31.</b> Leaner does online test via WOW.....	83
<b>Figure II.32.</b> Result of online test .....	84
<b>Figure II.33.</b> Knowledge evaluation is increased after learner finishes test successfully .....	85
<b>Figure II.34.</b> Posterior probability of given concept .....	86
<b>Figure II.35.</b> Search engine in WOW .....	87
<b>Figure II.36.</b> WOW feedback tool .....	88
<b>Figure II.37.</b> Collaborative area for collaborative learning .....	89
<b>Figure II.38.</b> Thin client connecting Zebra server .....	90
<b>Figure III.1.1.</b> Bayesian network (a classic example about “wet grass”).....	98
<b>Figure III.1.2.</b> Structure of Bayesian overlay model for Java course .....	103
<b>Figure III.1.3.</b> Bayesian network (Bayesian overlay model) of Java course with full of CPT (s) .....	106
<b>Figure III.1.4.</b> Sigma graph .....	107
<b>Figure III.1.5.</b> Sigma condition .....	108
<b>Figure III.1.6.</b> SIGMA-gate model.....	109
<b>Figure III.1.7.</b> An example of sigma graph.....	112
<b>Figure III.1.8.</b> Bayesian network transformed from sigma graph.....	113
<b>Figure III.1.9.</b> Bayesian overlay model of Java course with full of CPT (s). 118	
<b>Figure III.1.10.</b> Beta density functions with various parameters $a$ and $b$ .....	122
<b>Figure III.1.11.</b> The simple augmented BN with only one hypothesis node X .....	123
<b>Figure III.1.12.</b> The sample $\mathcal{D}=(X^{(1)}, X^{(2)}, \dots, X^{(m)})$ size of $m$ .....	125
<b>Figure III.1.13.</b> BN (a) and complex augmented BN (b) .....	127
<b>Figure III.1.14.</b> Expanded binomial BN sample of size $m$ .....	131
<b>Figure III.1.15.</b> Updated version of BN (a) and augmented BN (b) .....	137
<b>Figure III.1.16.</b> Updated version of BN (a) and augmented BN (b) in case of data missing .....	141

<b>Figure III.1.17.</b> BN structure as weighted graph (a) and augmented BN (b) of Java course .....	142
<b>Figure III.1.18.</b> Augmented BN with initial parameters in full .....	145
<b>Figure III.1.19.</b> Evolutional version of BN (a) and augmented BN (b) for Java course.....	152
<b>Figure III.1.20.</b> DBN for $t = 0, 1, 2$ .....	155
<b>Figure III.1.21.</b> The BN sample with full of weights .....	158
<b>Figure III.1.22.</b> Initial BN $G_0$ derived from BN in figure III.1.21 .....	159
<b>Figure III.1.23.</b> Transition weights .....	161
<b>Figure III.1.24.</b> DBN or expended BN with full of weights at time point $t$ ..	162
<b>Figure III.1.25.</b> DBN or expended BN (at time point $t$ ) whose weights are normalized .....	164
<b>Figure III.1.26.</b> DBN at time point $t$ with full of CPT (s) .....	167
<b>Figure III.1.27.</b> The cycle of proposed algorithm to construct DBN .....	171
<b>Figure III.1.28.</b> Bayesian network without CPT (s) .....	191
<b>Figure III.1.29.</b> Bayesian network with full of prior CPT (s).....	194
<b>Figure III.1.30.</b> Bayesian network without CPT (s) .....	197
<b>Figure III.1.31.</b> Bayesian network with full of prior CPT (s).....	201
<b>Figure III.2.1.</b> Two dimensions in Riding model.....	206
<b>Figure III.2.2.</b> Four-stage learning process of Kolb's model.....	208
<b>Figure III.2.3.</b> Learning styles in Honey and Mumford model.....	209
<b>Figure III.2.4.</b> HMM of weather forecast (hidden states are shaded) .....	214
<b>Figure III.2.5.</b> HMM (s) of learning styles $\Delta_1$ (hidden states are shaded) ....	218
<b>Figure III.2.6.</b> HMM (s) of learning styles $\Delta_2$ (hidden states are shaded) ....	219
<b>Figure III.2.7.</b> HMM (s) of learning styles $\Delta_3$ (hidden states are shaded) ....	219
<b>Figure III.3.1.</b> Separating hyperplanes .....	237
<b>Figure III.3.2.</b> Maximum-margin hyperplane, parallel hyperplanes and weight vector $W$ .....	239
<b>Figure III.3.3.</b> Support vectors .....	244
<b>Figure III.3.4.</b> Data points in training data (SVM).....	247
<b>Figure III.3.5.</b> An example of maximum-margin hyperplane.....	248
<b>Figure III.3.6.</b> Maximum-margin hyperplane with enhanced bias .....	250
<b>Figure III.3.7.</b> Decision tree constructed from nominal term frequencies at the first splitting .....	257
<b>Figure III.3.8.</b> Final decision tree constructed from nominal term frequencies .....	258
<b>Figure III.3.9.</b> Simplest topology of neural network with three layers such as input layer, hidden layer, and output layer .....	260
<b>Figure III.3.10.</b> Process of computing output of a unit .....	261
<b>Figure III.3.11.</b> The neural network for document classification .....	265
<b>Figure III.3.12.</b> Trained neural network .....	270
<b>Figure III.3.13.</b> User model clusters (means are marked by sign "+") .....	276
<b>Figure III.3.14.</b> Overlay model.....	277
<b>Figure III.3.15.</b> Graph model in form of tree and relationships.....	278
<b>Figure III.3.16.</b> Graph model and weighted arcs .....	280
<b>Figure III.3.17.</b> Bayesian network and its CPT (s).....	282

<b>Figure III.3.18.</b> Flow chart of $k$ -medoid algorithm modified with similarity measure .....	287
<b>Figure III.3.19.</b> A sample of six user vectors.....	289
<b>Figure III.3.20.</b> Two clusters resulted from $k$ -medoid algorithm at the first running time .....	290
<b>Figure III.3.21.</b> Two clusters resulted from $k$ -medoid algorithm at the second running time .....	293
<b>Figure IV.1.1.</b> Decision-theoretic tree .....	300
<b>Figure IV.1.2.</b> Item Response Function curve.....	302
<b>Figure IV.1.3.</b> IRF function and density function of examinee's ability together with ability estimate (0.5) given discriminatory parameter $a=5$ .....	312
<b>Figure IV.1.4.</b> An example of examinee's ability variance .....	320
<b>Figure IV.2.1.</b> Interaction between user modeling system and adaptive system .....	330
<b>Figure V.1.1.</b> Triangular Learner Model .....	341
<b>Figure V.1.2.</b> extended Triangular Learner Model .....	341
<b>Figure V.1.3.</b> The architecture of Zebra.....	342
<b>Figure V.2.1.</b> Ubiquitous user modeling .....	344
<b>Figure V.2.2.</b> User modeling with integrated context-awareness.....	346
<b>Figure V.2.3.</b> Spatial arrangement of ubiquitous computing similar fields according to two dimensions: user mobility and interface transparency .....	347
<b>Figure V.2.4.</b> Situation model.....	348
<b>Figure V.2.5.</b> Extended situation model.....	348
<b>Figure V.2.6.</b> Situated interaction and situation modeling for ubiquitous computing .....	349
<b>Figure V.2.7.</b> RDF triple.....	351
<b>Figure V.2.8.</b> Onion model of statement .....	351
<b>Figure V.2.9.</b> The schema of statement model in RDF graph notation .....	355
<b>Figure V.2.10.</b> Situational statement represented by RDF triple.....	356
<b>Figure V.2.11.</b> An example of statement represented by RDF triple .....	356
<b>Figure V.2.12.</b> The correlation between UbisWorld and real world .....	358
<b>Figure V.2.13.</b> UbisWorld and ontologies.....	359
<b>Figure V.2.14.</b> Architecture of ubiquitous user model service.....	360
<b>Figure V.2.15.</b> The work flow of ubiquitous user model service .....	362
<b>Figure V.2.16.</b> Incorporating Zebra into ubiquitous service .....	364
<b>Figure V.2.17.</b> Request-Response communication protocol between Zebra and ubiquitous service .....	364

## C. List of Formulas

<b>Formula I.3.1.1.</b> Condition probability of cluster node .....	31
<b>Formula I.3.1.2.</b> Conditional probability of child node $Y_i$ .....	32
<b>Formula I.3.1.3.</b> Conditional probability of child node $Y_i$ given parent nodes $X_1, \dots, X_N$ .....	32
<b>Formula I.3.3.1.</b> Posterior probability of student's mastery .....	43
<b>Formula III.1.1a.</b> Bayes' rule .....	94
<b>Formula III.1.1b.</b> Additional rule .....	95
<b>Formula III.1.1c.</b> Multiplication rule .....	95
<b>Formula III.1.1d.</b> Total probability rule .....	96
<b>Formula III.1.1d'.</b> Total probability rule in continuous case .....	96
<b>Formula III.1.1e.</b> Definition of cumulative distribution function (CDF) and probability density function (PDF) .....	97
<b>Formula III.1.2.</b> Global joint probability distribution of random vector .....	99
<b>Formula III.1.2'.</b> Reduced global joint probability distribution of random vector .....	99
<b>Formula III.1.3.</b> Posterior probability of variable $x_i$ given evidence $\mathcal{D}$ .....	99
<b>Formula III.1.3'.</b> Advanced posterior probability of variable $x_i$ given evidence $\mathcal{D}$ .....	99
<b>Formula III.1.4.</b> Global joint probability distribution of "wet grass" Bayesian network .....	100
<b>Formula III.1.5.</b> Posterior probability of rain given evidence "wet grass" ...	100
<b>Formula III.1.6.</b> Posterior probability of sprinkler given evidence "wet grass" .....	100
<b>Formula III.1.7.</b> Conditional probability of $J=1$ given $C=1$ , $O=1$ , and $I=1$ ..	104
<b>Formula III.1.8.</b> Sigma condition.....	108
<b>Formula III.1.9.</b> Probability of sigma sum .....	109
<b>Formula III.1.10.</b> Theorem of SIGMA-gate inference.....	111
<b>Formula III.1.11.</b> Theorem of SIGMA-gate inference with weighted graph	111
<b>Formula III.1.12.</b> Beta density function .....	121
<b>Formula III.1.13.</b> Gamma function .....	122
<b>Formula III.1.14.</b> Important property of gamma function with regard to factorial function .....	122
<b>Formula III.1.15.</b> Integral of product expression $x^a(1-x)^b$ .....	122
<b>Formula III.1.16.</b> Conditional probability (relative frequency) of $X$ as value of dummy variable $F$ .....	123
<b>Formula III.1.17.</b> Probability of hypothesis $X$ as expectation of beta variable $F$ .....	124
<b>Formula III.1.18.</b> Expectation of expression $F^s(1-F)^t$ .....	125
<b>Formula III.1.19.</b> Probability $P(\mathcal{D})$ of evidences $\mathcal{D}$ .....	126
<b>Formula III.1.20.</b> Posterior beta density function.....	126
<b>Formula III.1.21.</b> Posterior probability of $X$ .....	127
<b>Formula III.1.22.</b> Conditional probability (relative frequency) of $X_i$ given a parent instance $PA_{ij}$ , as value of dummy variable in multi-node BN .....	128

<b>Formula III.1.23.</b> Beta density function $\beta(F_{ij})$ corresponding to an instance of a parent of node $X_i$ .....	128
<b>Formula III.1.24.</b> Joint beta density function of variable $X_i$ having $c_i$ parent instances.....	129
<b>Formula III.1.25.</b> Global joint beta density function of $n$ independent variable $X_i$ (s).....	129
<b>Formula III.1.26.</b> Probability of variable $X_i$ with respective to its parent instance as expectation of beta variable .....	129
<b>Formula III.1.27.</b> Probability of evidences corresponding to variable $X_i$ ....	131
<b>Formula III.1.28.</b> Probability of evidence sample $\mathcal{D}$ given vectors $F_i$ .....	132
<b>Formula III.1.29.</b> Whole probability of evidence sample $\mathcal{D}$ .....	133
<b>Formula III.1.30.</b> Expectation of binomial trials .....	133
<b>Formula III.1.31.</b> Posterior beta density function in multi-node BN.....	135
<b>Formula III.1.32.</b> Posterior probability of variable $X_i$ given its parent instance $PA_{ij}$ .....	135
<b>Formula III.1.33.</b> Definition of equivalent sample size $N$ .....	143
<b>Formula III.1.34.</b> Theorem of equivalent sample size $N$ .....	143
<b>Formula III.1.35.</b> Distributed global joint probability distribution of DBN .	154
<b>Formula III.1.36.</b> Transition probability distribution.....	157
<b>Formula III.1.37.</b> Markov property according to transition probability distribution.....	157
<b>Formula III.1.38.</b> Formula of slip factor and guess factor .....	159
<b>Formula III.1.39.</b> Conditional probability $a$ .....	159
<b>Formula III.1.40.</b> The bias $b$ of user knowledge.....	160
<b>Formula III.1.41.</b> The weight $w$ as product of conditional probability $a$ and bias $b$ .....	160
<b>Formula III.1.42.</b> The weight $w$ implicates that the conditional transition probability satisfies both Markov property and stationary property .....	160
<b>Formula III.1.43.</b> The set $Y$ of all variables (nodes) of expended BN $G'[t]$ at time point $t$ .....	161
<b>Formula III.1.44.</b> Normalizing weights $w_2$ , $w_3$ .....	162
<b>Formula III.1.45.</b> Normalizing weights in general case.....	163
<b>Formula III.1.46.</b> Posterior probability of each $x_i[t-1]$ given evidences $\mathcal{D}[t-1]$ .....	164
<b>Formula III.1.47.</b> Conditional probability of each variable $x_i[t]$ at current time point $t$ .....	165
<b>Formula III.1.48.</b> Posterior probability of each $x_i[t]$ given evidences $\mathcal{D}[t]$ ...	168
<b>Formula III.1.49.</b> Beta density function $\beta(F; a, b)$ .....	173
<b>Formula III.1.50.</b> Probability of variable $X$ as expectation of beta variable $F$ .....	173
<b>Formula III.1.51.</b> Posterior probability of variable $X$ as conditional expectation of beta variable $F$ .....	174
<b>Formula III.1.52.</b> Likelihood function .....	175
<b>Formula III.1.53.</b> Optimal parameter vector .....	175
<b>Formula III.1.54.</b> Log-likelihood function and optimal parameter vector....	175
<b>Formula III.1.55.</b> Co-variance matrix of parameter vector .....	176

<b>Formula III.1.56.</b> Beta density function (beta distribution) $\beta(X; a, b)$ .....	177
<b>Formula III.1.57.</b> Definition of digamma function .....	177
<b>Formula III.1.58.</b> Integral form of digamma function .....	177
<b>Formula III.1.59.</b> Recurrence formula of digamma function .....	178
<b>Formula III.1.60.</b> Digamma function in case of positive integer variable ....	180
<b>Formula III.1.61.</b> Trigamma function .....	181
<b>Formula III.1.62.</b> Recurrence formula of trigamma function.....	182
<b>Formula III.1.63.</b> Trigamma function in case of positive integer variable ...	183
<b>Formula III.1.64.</b> Beta function $B(x, y)$ .....	183
<b>Formula III.1.65.</b> First-order partial derivative of beta function $B(x, y)$ .....	184
<b>Formula III.1.66.</b> Beta density function with regard to beta function .....	184
<b>Formula III.1.67.</b> Log-likelihood function of beta density function (beta distribution).....	184
<b>Formula III.1.68.</b> First-order partial derivative of log-likelihood function of beta density function with regard to parameter $a$ .....	185
<b>Formula III.1.69.</b> First-order partial derivative of log-likelihood function of beta density function with regard to parameter $b$ .....	185
<b>Formula III.1.70.</b> The set of differential equations for estimating parameters $a$ and $b$ .....	186
<b>Formula III.1.71.</b> Co-variance matrix of parameters of beta density function .....	187
<b>Formula III.1.72.</b> Standard errors of parameter estimates of beta distribution .....	187
<b>Formula III.1.73.</b> Two simplest equations for estimating positive integer parameters $a$ and $b$ .....	189
<b>Formula III.1.74.</b> New version of equations for estimating positive integer parameters $a$ and $b$ .....	196
<b>Formula III.3.1.</b> Term frequency .....	235
<b>Formula III.3.2.</b> Inverse document frequency .....	235
<b>Formula III.3.3.</b> The weight $w_{ij}$ of term $t_j$ in document $D_i$ .....	236
<b>Formula III.3.4.</b> Vector model of document $D_i$ .....	236
<b>Formula III.3.4'.</b> Document vector as a set of term frequencies .....	236
<b>Formula III.3.5.</b> Hyperplane equation.....	237
<b>Formula III.3.6.</b> Equations of two parallel hyperplanes .....	238
<b>Formula III.3.7.</b> Classification constraint.....	239
<b>Formula III.3.8.</b> Constrained optimization problem for constructing maximum-margin hyperplane.....	240
<b>Formula III.3.9.</b> Lagrangian function.....	240
<b>Formula III.3.10.</b> Lagrangian duality problem .....	241
<b>Formula III.3.11.</b> Formula for determining optimal weight vector $W^*$ .....	241
<b>Formula III.3.12.</b> Dual function for determining Lagrange multipliers .....	242
<b>Formula III.3.13.</b> Lagrange multipliers .....	242
<b>Formula III.3.14.</b> Optimal weight vector $W^*$ .....	242
<b>Formula III.3.15.</b> Optimal bias $b^*$ according to mean method .....	243
<b>Formula III.3.16.</b> Equation of maximum-margin hyperplane (SVM classifier) .....	243

<b>Formula III.3.17.</b> Classification rule derived from maximum-margin hyperplane (SVM classifier).....	243
<b>Formula III.3.18.</b> Entropy of training corpus .....	252
<b>Formula III.3.19.</b> Entropy of given attribute inside training corpus.....	253
<b>Formula III.3.20.</b> Information gain of given attribute inside training corpus.....	253
<b>Formula III.3.21.</b> Formula for computing the output of a unit.....	261
<b>Formula III.3.22.</b> Error of output unit.....	264
<b>Formula III.3.23.</b> Error of hidden unit .....	264
<b>Formula III.3.24.</b> Updating connection weight .....	264
<b>Formula III.3.25.</b> Updating bias.....	264
<b>Formula III.3.26.</b> Dissimilarity of two user model vectors according to Euclidean distance.....	275
<b>Formula III.3.27.</b> Error criterion for $k$ -mean algorithms.....	276
<b>Formula III.3.28.</b> The mean of a cluster.....	277
<b>Formula III.3.29.</b> Dissimilarity of two graph models .....	278
<b>Formula III.3.30.</b> Dissimilarity of two weighted graph models .....	280
<b>Formula III.3.31.</b> Dissimilarity of two weighted graph models without node values .....	281
<b>Formula III.3.32.</b> Dissimilarity of two Bayesian networks .....	282
<b>Formula III.3.33.</b> Cosine similarity measure .....	285
<b>Formula III.3.34.</b> Correlation coefficient .....	286
<b>Formula III.3.35.</b> Average similarity .....	288
<b>Formula III.3.36.</b> Global average similarity .....	288
<b>Formula IV.1.1.</b> Expected utility of an action.....	300
<b>Formula IV.1.2.</b> Item Response Function.....	302
<b>Formula IV.1.3.</b> Information function for item $i$ .....	304
<b>Formula IV.1.4.</b> IRF with zero guessing parameter .....	305
<b>Formula IV.1.5.</b> Examinee's initial ability .....	305
<b>Formula IV.1.6.</b> Likelihood function of examinee's ability .....	306
<b>Formula IV.1.7.</b> Log-likelihood function of examinee's ability .....	306
<b>Formula IV.1.8.</b> Equation for finding out parameter estimates .....	307
<b>Formula IV.1.9.</b> Discriminatory and difficult estimates.....	308
<b>Formula IV.1.10.</b> Probability density function of examinee's ability.....	308
<b>Formula IV.1.11.</b> Optimal probability density function of examinee's ability .....	308
<b>Formula IV.1.12.</b> Examinee's ability estimate .....	311
<b>Formula IV.1.13.</b> Examinee's ability variance .....	312
<b>Formula IV.1.14.</b> Dilogarithm function.....	314
<b>Formula IV.1.15.</b> Expectation of square of examinee's ability .....	315
<b>Formula IV.1.16.</b> Inversion property of dilogarithm.....	318
<b>Formula IV.1.17.</b> Examinee's explicit ability variance.....	319
<b>Formula IV.1.18.</b> Examinee's statistical ability mean .....	320
<b>Formula IV.1.19.</b> Examinee's ability variance as statistical sample variance .....	320
<b>Formula IV.1.20.</b> Discriminatory estimate .....	321

<b>Formula IV.1.21.</b> Ability error used as stopping criterion of CAT algorithm .....	323
<b>Formula IV.2.1a.</b> Sample means .....	332
<b>Formula IV.2.1b.</b> Sample variances .....	332
<b>Formula IV.2.1c.</b> Sample standard deviations .....	332
<b>Formula IV.2.2.</b> System criterion $\alpha$ determined based on F-distribution test .....	333
<b>Formula IV.2.3.</b> Linear regression function of user's total knowledge .....	333
<b>Formula IV.2.4.</b> System criterion $\alpha$ determined based errors of linear regression model .....	334
<b>Formula IV.2.5.</b> Academic criterion $\beta$ based cumulative function .....	335
<b>Formula IV.2.6.</b> Simple adaptation criterion $\beta$ based the number of satisfied users .....	335
<b>Formula IV.2.7.</b> Adaptation criterion $\gamma$ as mean of compressed vector .....	336

## D. Index

- a*, 122
- ability. *See* examinee's ability
- ability error, 323
- ability estimate, 304, 309, 311
- ability variance, 312
- Abstract Conceptualization*, 207
- AC. *See* adaptive component
- academic criterion, 331
- access attribute, 115
- Accommodating*, 208
- accountability, 108
- activation function, 261
- Active*, 58, 209
- Active Experimentation*, 207
- Activist*, 208
- adaptation criterion, 331
- adaptation model, 20, 24, 62
- adaptation process, 64
- adaptation rules, 115
- adaptive, 11
- adaptive component, 22
- Adaptive Education System, 21
- adaptive educational hypermedia system, 17
- Adaptive Educational Intelligent Web-Based System, 25
- Adaptive Educational Web-Based System, 25
- adaptive hypermedia applications, 53
- Adaptive Hypermedia for All, 22
- Adaptive Hypermedia System, 12
- adaptive learning, 1
- adaptive learning model, 331
- adaptive learning system, 1, 11
- adaptive learning web page, 82
- adaptive link hiding, 19
- adaptive link sorting, 19
- adaptive navigation, 18
- adaptive presentation, 18
- adaptive procedure, 24
- adaptive process, 24
- adaptive systems, 11
- additional rule, 95
- administered item, 304
- administration box, 353
- administration process, 304
- advanced CAT algorithm, 312
- AEHS. *See* adaptive educational hypermedia system
- AEIWBS. *See* Adaptive Educational Intelligent Web-Based System
- AES. *See* Adaptive Education System
- AEWBS. *See* Adaptive Educational Web-Based System
- agent-based approach, 52
- aggregation, 101
- aggregation inhibition, 107
- aggregation relationship, 101
- aggregative node, 107
- AHA!. *See* Adaptive Hypermedia for All
- AHA! engine, 23
- AHAM, 22
- AHS. *See* Adaptive Hypermedia System
- a*, 303
- a<sub>ij</sub>*, 128
- ALS. *See* adaptive learning system
- alternate approach, 298
- alternative hypothesis, 333
- AM. *See* adaptation model
- Analytic, 206
- Analytical, 205
- ANATOM-TUTOR, 16
- Andes, 32
- ANN. *See* artificial neural network
- Application box, 360
- Application-directed*, 209
- AprioriAll, 226
- aptitudes, 5
- aptitude-treatment interactions system, 12
- arc, 96, 277
- artificial intelligence, 6
- artificial neural network, 259
- assessment of Bayesian network, 298
- Assimilating, 208
- ATI. *See* aptitude-treatment interactions system
- attribute-value pairs, 114
- Auditory*, 205
- augmented Bayesian network (augmented BN), 123
- automatic stereotype management, 51
- auxiliary, 355
- average similarity, 287
- b*, 122
- background, 4
- back-propagation algorithm, 263
- backward reasoning, 116
- basic user dimensions, 356
- Baum Welch algorithm, 155
- Bayes' rule, 94
- Bayes' theorem, 94
- Bayesian inference, 94
- Bayesian model, 102, *See* Bayesian overlay model
- Bayesian network, 96
- Bayesian overlay model, 94, 102
- belief network engine, 61
- Belief, Goal and Plan Maintenance System, 50
- best\_grade, 31
- beta density function, 121, 173
- beta distribution. *See* beta density function
- Beta likelihood estimation, 177
- beta(x; a, b)*. *See* beta density function
- BGP-MS. *See* Belief, Goal and Plan Maintenance System
- b<sub>i</sub>*, 303

- bias, 160, 237, 263  
 $b_{ij}$ , 128  
 binary random variable, 103  
 binomial augmented Bayesian network (binomial augmented BN), 125  
 binomial sample, 125  
 binomial trials, 131  
 BN. *See* Bayesian network  
 BNE. *See* belief network engine  
 breaking sequential pattern, 231  
 CA. *See* condition-action  
 CAI. *See* Computer Assisted Instructional candidate generation, 228  
 candidate generation-and-test approach, 226  
 canned text adaptation, 19  
 CAT. *See* Computerized Adaptive Testing cause inhibition, 108  
 cause-effect, 101  
 CBM. *See* constraint-based modeling  
 CDF. *See* cumulative distribution function characteristics, 11  
 chronologic order, 160  
 $c_i$ , 303  
 CI. *See* communication interface  
 c-id, 114  
 c-info. *See* concept information class attribute, 252  
 classification constraint, 239  
 classification rule, 243, 259  
 classifier, 234, 237  
 cluster node, 29  
 clustering approach, 29  
 cognitive structure, 204  
 collaborative area, 88  
 collaborative learning, 275  
 combination taken  $k$  of  $b$ , 188  
 communication interface, 61  
 communication sub-component, 54  
 community adaptation, 274  
 Computer Assisted Instructional, 13  
 Computerized Adaptive Testing, 303  
 concept, 7  
 concept information, 114  
 concept selection process, 64  
 concept selection rules, 62  
 concept tier, 24  
*Concrete Experience*, 207  
 condition-action, 116  
 conditional approach, 29  
 conditional dependence, 102  
 conditional independence, 100  
 conditional mutual information, 45  
 conditional probability, 95  
 conditional probability distribution, 96  
 conditional probability table, 96  
 conditional text, 19  
 confidence, 232  
 Conflict Resolution, 360  
 connection (neural network connection), 260  
 connection weight, 263  
 constitutionally based learning styles and preferences, 204  
 constrained optimization problem, 239  
 constraint-based approach, 156  
 constraint-based modeling, 40  
 content selection process, 64  
 content selection rules, 62  
 context model, 348  
 context-awareness, 345  
 context-rule nodes, 37  
*Converging*, 208  
 corpus, 94, 235  
 correct knowledge, 9  
 correct response, 305  
 correlation coefficient, 286  
 cosine similarity measure, 285  
 counter, 131, 149  
 co-variance matrix, 176  
 CPD. *See* conditional probability distribution  
 CPT. *See* conditional probability table  
*Cr.* *See* relevance condition  
 Cramer-Rao lower bound, 176  
*Cs.* *See* satisfaction condition  
 CUMMULATE, 52  
 cumulative distribution function, 97  
 cumulative function. *See* cumulative distribution function  
 daemon, 65  
 DAG. *See* directed acyclic graph  
 data mining, 6  
 data point, 237  
 data sample, 94  
 data-centric method, 26, 296  
 DBN. *See* dynamic Bayesian network  
 decision base, 16  
 decision tree, 252  
 decision-theoretic approach, 299  
 default stereotype, 6  
 degrees of freedom, 333  
 demographic, 5  
 density function. *See* probability density function  
 dependency, 101  
 dependency graph, 27  
 $depth(v_{ij})$ , 278  
 Dexter Hypertext Reference Model, 22  
 DGJPD. *See* Distributed Global Joint Probability Distribution  
 diagnosis, 15  
 diagnostic, 101  
 diagnostic approach, 299  
 diagnostic relationship, 101, 103  
 didactics, 16  
 didactics module. *See* pedagogical module  
 differential model, 9  
 difficult estimate, 308  
 difficult parameter, 303  
 digamma function, 177  
 dilogarithm function, 314  
 direct guidance, 19

- directed acyclic graph, 96  
 directory component, 54  
 discriminatory estimate, 321  
 discriminatory item, 321  
 discriminatory parameter, 303  
 dissim. *See* dissimilarity  
 dissimilarity measure, 275  
 distance, 275  
 distance learning, 1  
 Distributed Global Joint Probability Distribution, 154  
 Distributed ontologies box, 361  
 Distributed Services box, 360  
 Distributed Statements box, 361  
 distribution. *See* probability distribution  
*Diverging*, 208  
 $do\$bayes\$infer$ , 118  
 DOCS, 21  
 document classification, 236  
 document vector, 236  
 domain, 8  
 domain expert, 14  
 domain independence information, 4  
 domain knowledge, 8  
 domain model, 7, 20, 23, 62  
 domain specific information, 3  
 domain-general part, 36  
 Doppelgänger, 51  
 dot product, 237  
 downward inference direction, 346  
 drafting phase, 45  
 d-separation, 45  
 dual function, 242  
 dummy attribute, 117, 118  
 dummy variable, 109, 123  
 Dunn and Dunn model, 204  
 dynamic Bayesian network, 154  
 ECA. *See* event-condition-action edge, 277  
 efficiency-centric method, 26, 296  
 eigenvectors, 335  
 e-learning, 1  
 element, 224  
 EM. *See* Expectation Maximization  
 EM algorithm, 137  
 entropy, 252  
 equivalent sample size, 143  
 error criterion, 276  
 error of hidden unit, 263  
 error of output unit, 263  
 E-step. *See* Expectation step  
 estimated knowledge, 334  
 Euler's number, 122  
 Euler-Mascheroni constant, 179  
 evaluation act, 337  
 evaluation criterions, 331  
 evaluation of Bayesian network, 296  
 evaluation scenario, 337  
 event, 96  
 event storage, 52  
 event-condition-action, 115  
 event-driven approach, 52  
 evidence, 94  
 evidence matrix, 130  
 evidence nodes, 102  
 evidence sample, 94, 130  
 evidence variables, 102  
 evidence vector, 124  
 evidences, 130  
 evolution of Bayesian network. *See* evolution of Bayesian overlay model  
 evolution of Bayesian overlay model, 120  
 evolution of CPT, 120  
 evolution of parameters, 121  
 examinee's ability, 302  
 exception independence, 108  
 exclusive aggregation, 107  
 exclusive union, 107  
 Expectation Maximization, 137  
 Expectation step, 137  
 expected knowledge, 9  
 expected utility, 300  
 expended Bayesian network (expended BN), 161  
 experience, 4  
 expert-centric method, 26, 296  
 explanation box, 352  
 explanations, 100  
 explicit ability variance, 319  
 extended functions, 222  
 external inference agents, 52  
*Extravert*, 206  
*F. See* dummy variable  
 $f\_list$ , 231  
 fact nodes, 37  
 fault model, 4  
 FD. *See* Field-dependence  
 F-distribution, 333  
 feedback act, 337  
 feedback report, 79  
 feedback tool, 87  
 feedbacks, 79, 336  
 feed-forward neural network, 262  
*Feeler*, 207  
 Felder-Silverman model, 209  
 $F_i$ , 131  
 FI. *See* Field-independency  
*Field-dependence*, 205  
*Field-independency*, 206  
 $F_{ij}$ , 128  
 first-order logic, 21  
 fix stereotype, 6  
 flexibly learning preferences, 204  
 FOL. *See* first-order logic  
 format attribute, 216  
 forward reasoning, 116  
 frame, 350  
 FrameNet, 350  
 FreeSpan. *See* Frequent pattern-projected Sequential pattern

- frequent item matrix, 231  
 frequent itemset, 226  
 Frequent pattern-projected Sequential pattern, 230  
 frequent sequence, 225  
 functional interface, 51  
 gamma function, 121  
 General User Model Ontology, 350  
 General User Modeling System, 48  
 Generalized Sequential Pattern, 228  
 generic scrutiny tools, 53  
**GJPD.** *See* Global Joint Probability Distribution  
*Global*, 205, 209  
 global average similarity, 288  
 Global Joint Probability Distribution, 98  
 global parameter independence, 129  
 global teaching knowledge, 17  
 goal nodes, 37  
 goals, 4  
 gradient, 241  
 graph model, 8, 278  
 group adaptation, 275  
 GRUNDY, 48  
 GSP. *See* Generalized Sequential Pattern  
 guess. *See* lucky guess  
 guessing parameter, 303  
 GUMO. *See* General User Model Ontology  
 GUMS. *See* General User Modeling System  
 Hessian matrix, 186  
 hidden layer, 260  
 hidden Markov model, 213  
 hidden nodes, 102  
 hidden state, 213  
 hidden unit, 260  
 hidden variables, 102  
 HMM. *See* hidden Markov model  
 Honey and Mumford model, 208  
 horizontal data format, 226  
 HTML, 67  
 human-machine interface, 2  
 hypermedia, 17  
 hyperplane, 237  
 hyperplane equation, 237  
 hypothesis, 94  
 hypothesis test, 333  
 hypothesis variable, 123  
 ICC. *See* Item Characteristic Curve  
 idf. *See* inverse document frequency  
 $I_i(\theta)$ . *See* information function  
*Imager*, 206  
*Impulsive*, 205  
 incomplete sample, 146  
 incorrect knowledge, 9  
 indicator mastered, 74  
 individual adaptation, 274  
 individual user model, 50  
 inference (stereotype), 7  
 inference agents, 52  
 Inference Engine, 360  
 inferred user model, 52  
 information function, 304  
 information gain, 253  
 information matrix, 176  
 informative item, 321  
 inhibition independence, 108  
 initial ability, 305  
 initial Bayesian network (initial BN), 154  
 initial state distribution, 212  
 input layer, 260  
 input unit, 260  
 instructional meta-strategy, 212  
 instructional strategy, 212  
 intelligent tutoring system, 13  
 interactive attribute, 216  
 interesting document, 234  
 interesting document vector, 272  
 interests. *See* user interests  
 Interface Manager, 360  
 interface transparency, 347  
 Interfaces & Exchange box, 361  
*Internal Kinesthetic*, 205  
 intersection operator, 95  
*Introvert*, 206  
*Intuitive*, 207, 209  
 inverse document frequency, 235  
 inversion property, 317  
 IRF. *See* Item Response Function  
 IRT. *See* Item Response Theory  
 item, 224, 302  
 Item Characteristic Curve, 302  
 item pool, 303  
 Item Response Function, 302  
 Item Response Theory, 302  
 itemset, 224  
 ITS. *See* intelligent tutoring system  
 Java, 6  
 Java course, 67  
 Java tutorial, 67  
*Judger*, 207  
 K. *See* knowledge sub-model  
 Karush-Kuhn-Tucker condition, 242  
 Karush-Kuhn-Tucker multiplier, 240  
 KBS hyperbook system, 26  
 KI. *See* knowledge item  
 k-mean algorithm, 276  
 k-medoid algorithm, 287  
 KN-IPCMS. *See* KoNstanz Inter-Process Communication Management System  
 knowledge attribute, 115  
 knowledge diagnosis, 102  
 knowledge element, 7  
 knowledge evaluation, 81  
 knowledge item, 7, 26  
 knowledge sub-model, 57, 92  
 knowledge vector, 332  
 Kolb Learning Style Inventory, 207  
 KoNstanz Inter-Process Communication Management System, 51  
 Lagrange multiplier, 240

- Lagrangian duality, 241  
 Lagrangian function, 240  
 large itemset. *See* frequent itemset  
 large itemset phase, 226  
 large sequence. *See* frequent sequence  
 LDAP-UMS, 54  
 leaf nodes, 252  
 leaky-OR, 37  
 learner model, 1  
 learner modeling, 1  
 learners, 2  
 learning concept recommendation, 72, 223  
 learning context, 11  
 learning history, 58  
 learning history sub-model, 57, 59, 222  
 learning management systems, 1  
 learning materials, 1, 234  
 learning object, 8, 216  
 learning path recommendation, 71  
 learning process, 2  
 learning rate, 263  
 learning report, 77  
 learning resources, 20  
 learning sequence, 225  
 learning sequence database, 223  
 learning style sub-model, 57, 203  
 learning styles, 4, 57, 203  
 learning-by-doing, 14  
 left-hand itemset, 232  
 LH. *See* learning history sub-model  
 likelihood function, 95, 175, 306  
 linear regression function, 333  
 link annotation, 19  
 link generation, 19  
 itemset. *See* frequent itemset  
 local parameter independence, 129  
 local teaching knowledge, 17  
 log tier, 24  
 logical predicates, 21  
 logistic function, 262, 302  
 log-likelihood function, 175, 306  
 LS. *See* learning history sub-model  
 l-sequence, 224  
 lucky guess, 154  
 machine learning, 6  
 macro-adaptive system, 12  
 mailing list tool, 78  
 main part box, 351  
 mal-knowledge, 9  
 map adaptation, 19  
 margin, 238  
 marginal probability, 95  
 Markov condition. *See* Markov property  
 Markov model, 212  
 Markov property, 157, 213  
 mastered, 8  
 $mastered_c$ , 42  
 mastery, 7  
 maximal frequent sequence, 225  
 Maximization step, 137  
 maximum frequent itemset, 234  
 Maximum Likelihood Estimation, 175, 306  
 maximum margin, 237  
 maximum-margin hyperplane, 237  
 ME. *See* mining engine  
*Meaning-oriented*, 209  
 media space, 20, 62  
 medoid, 287  
 micro-adaptive system, 12  
 min\_conf, 232  
 min\_sup, 224  
 minimal infrequent sequence, 225  
 mining engine, 60  
 missing data, 137  
 missing values, 138  
 MLE. *See* Maximum Likelihood Estimation,  
     *See* Maximum Likelihood Estimation  
 MM. *See* Markov model  
 M-step. *See* Maximization step  
 multi-node Bayesian network (multi-node  
     BN), 127  
 multiplication rule, 95  
 mutual information, 45  
 mutually exclusive, 96  
 mutually independent, 96  
 Myers-Briggs Type Indicator, 206  
 neural network. *See* artificial neural network  
 NN. *See* neural network  
 node, 96  
 noisy OR-gate, 111  
 noisy-AND, 37  
 nominal interesting document, 273  
 nominal term frequency, 251  
 non-class attribute, 252  
 non-evidence variable, 155  
 non-leaf nodes, 252  
 normal bias, 190  
 not mastered, 8  
 null hypothesis, 333  
 OBS, 22  
 observation, 22, 213  
 observation probability matrix, 213  
 observer, 61, 64  
 occurrences, 138  
 OLAЕ, 33  
 onion model, 351  
 online learning, 1  
 ontology, 8, 350, 357  
 Ontology Reasoning, 360  
 Ontology Web Language, 350  
 optimal bias, 241  
 optimal parameter. *See* parameter estimate  
 optimal parameter vector, 175  
 optimal weight vector, 241  
 optional response, 305  
 ordinary weight, 158  
 outcome, 124  
 output layer, 260  
 output unit, 260  
 overlay model, 7

- overlay model clustering, 277  
*OWL*. *See* Ontology Web Language  
 $PA_i$ , 131  
 $PA_{ij}$ . *See* parent instance  
 parallel hyperplanes, 238  
 parameter estimate, 175, 306  
 parameter estimator. *See* parameter estimate  
 parameter evolution, 121  
 parameter learning, 98, 120, 121, 153, 172  
 parameter vector estimate, 175  
 parameter vector estimator, 175  
 parent instance, 128  
 parent-child, 101  
 partial node, 107  
 Pask model, 209  
 Pattern-growth approach, 230  
 PDF. *See* probability density function  
 pedagogical module, 14  
*Perceiver*, 207  
 percentage point, 333  
 perception group, 210  
 $performance_{c,p}$ , 43  
 personal traits, 4  
 Personis, 53  
 perturbation model, 9  
 physics rules, 33  
 piecewise-linear function, 262  
 plan model, 9  
 plan recognition, 9  
 plans of action, 16  
 polynomial, 188  
 posterior beta density function, 126  
 posterior density function, 126  
 posterior probability, 95  
*Pragmatist*, 58, 208  
 predicate, 355  
 prediction error, 334  
 prerequisite, 101  
 prerequisite relationship, 26  
 presentation specification, 114  
 presenter, 64  
 prior density function, 124  
 prior probability, 95  
 privacy box, 353  
 probabilistic inference, 157  
 probability density function, 96  
 probability distribution, 97  
 processing unit, 259  
 profile mining, 59  
 profile tier, 24  
 projected database, 231  
 projected itemset, 230  
 PROlog based Tool for User Modeling, 49  
 proposition nodes, 37  
 PROTUM. *See* PROlog based Tool for User Modeling  
 $q_i$ . *See* optional responses  
 qualitative model, 172  
 quantitative model, 172  
 random event, 95  
 random probability, 95  
 random variable, 96  
 range, 355  
 rating matrix, 335  
 RDF. *See* Resource Description Framework  
 RDF triple, 355  
 real knowledge, 334  
 reasoning, 93  
 recurrence relation, 178  
 recurrent neural network, 262  
*Reflective*, 58, 205, 209  
*Reflective Observation*, 207  
*Reflector*, 208  
 regression coefficients, 333  
 reinforcement learning, 262  
 relationships, 8  
 relevance condition, 40  
 $relevantIS_{c,p}$ , 43  
 $relevantSS_{c,p}$ , 43  
 representation sub-component, 54  
 representation system, 51  
*Reproduction-oriented*, 209  
 Resource Description Framework, 349  
 resource model, 62, 348  
 retraction, 7  
 Retrieval Filter, 360  
 $r_i$ . *See* correct responses  
 Riding model, 206  
 right-hand itemset, 232  
 rule-application nodes, 37  
 runtime layer, 19  
 $s$ , 125  
 sample, 94, 124  
 sample mean, 332  
 sample variance, 332  
 satisfaction condition, 40  
 satisfaction criterion, 331  
 SB-ONE, 51  
 scalar product. *See* dot product  
 scheduler sub-component, 54  
 scored-based approach, 155  
 search engine, 86  
 searching history, 271  
 SEM. *See* Structural Expectation Maximization  
 semantic web, 349  
*Sensing*, 209  
*Sensor*, 206  
 sensor level, 52  
 separating hyperplane, 236  
 sequence, 224  
 sequence of anchors, 114  
 sequence of observations, 214  
 sequence of state transitions, 220  
 sequence phase, 227  
*Sequential*, 209  
 sequential pattern, 223, 225  
 Sequential PAtern Discovery using Equivalent classes, 229  
 sequential pattern mining, 223, 226

- sequential rule, 232  
*Serialist*, 209  
 server level, 52  
 sigma condition, 108  
 sigma graph, 101, 106  
 sigma sum, 104, 107  
 SIGMA-gate inference, 104, 107, 110, 111  
 sigmoid function, 262  
 significant level, 333  
 $s_{ij}$ , 131  
 similarity measure, 285  
 situated interaction, 349  
 situation, 347  
 Situation Adder, 360  
 situation box, 352  
 situation model, 347  
 situation modeling, 348  
 situation semantics, 347  
 Situation Server, 360  
 situational statement, 350  
 SituationML, 354, 361  
 SituationQL, 361  
 slip. *See* temporary slip  
 solution graph, 33  
 sorting fragments, 19  
 source node, 107  
 SPADE. *See* Sequential PAttern Discovery using Equivalent classes  
 specifying prior probabilities, 172  
 SQL-Tutor, 41  
 squashing function, 262  
 stable personality type, 204  
 standard deviation, 187, 319, 332  
 standard error, 187  
 standard normal distribution, 335  
 state, 212  
 statement, 351  
 statement model, 351  
 stationary, 157  
 statistical ability mean, 320  
 statistics, 6  
 stereotype, 6  
 stochastic approach, 29  
 stochastic process, 212  
 stopping criterion (CAT), 303, 323  
 storage layer, 19  
 strategic contexts, 16  
 strategy nodes, 38  
 stretch text, 19  
 Structural Expectation Maximization, 155  
 structure learning, 93, 98, 120, 153, 154, 172  
 student model, 2  
 students, 2  
 study act, 337  
 sub-sequence, 224  
 summing function, 261  
 super-sequence, 224  
 supervised learning, 262  
 support, 224  
 support threshold, 224  
 support vector machine, 236  
 SVM. *See* support vector machine  
 system criterion, 331  
 $t$ , 125  
*Tactile Kinesthetic*, 205  
 TAGUS, 49  
 target level, 16  
 target node, 107  
 task-specific part, 36  
 temporal dependency, 163  
 temporal random vector, 154  
 temporal relationship, 154  
 temporal variable, 156  
 temporal weight. *See* transition weight  
 temporary slip, 154  
 term, 235  
 term frequency, 235  
 tf. *See* term frequency  
 theorem of equivalent sample size, 143  
 theorem of SIGMA-gate inference, 111  
*Theorist*, 58, 208  
 thickening phase, 46  
 thin client, 89  
*Thinker*, 207  
 thinning phase, 46  
 threshold function, 262  
 $t_{ij}$ , 131  
 TLM. *See* Triangular Learner Model  
 topology, 260  
 total probability rule, 96  
 trained neural network, 270  
 training data, 94, 130  
 transformation phase, 227  
 transition Bayesian network (transition BN), 154  
 transition dependency, 155  
 transition probability, 155  
 transition probability distribution, 154, 157, 213  
 transition probability matrix, 213  
 transition weight, 157, 160  
 trials, 94, 124  
 Triangular Learner Model, 57  
 trigamma function, 181  
 trigger, 7  
 trust Bayesian network (trust BN), 129  
 tutoring module. *See* pedagogical module  
 type attribute, 216  
 ubiquitous computing, 346  
 ubiquitous user model service, 359  
 ubiquitous user modeling, 344  
 Ubiquitous World, 355, 357  
 UbisTools, 358  
 UbisWorld. *See* Ubiquitous World  
 um (a user modeling shell), 50  
 um toolkit, 50  
 UMS. *See* user modeling system  
 UMT. *See* User Modeling Tool  
 uncovering problem, 214  
 understanding group, 211

- Undirected*, 209  
uniform virtual world, 357  
union operator, 95  
unit. *See* processing unit  
unsupervised learning, 262  
upward inference direction, 346  
user communities, 73, 274  
user groups, 73, 274  
user interests, 233  
user interface, 15  
user mobility, 347  
user model, 1  
user model cluster, 275  
user model clustering, 275  
user model interest categories, 356  
user modeler, 14  
user modeling, 1  
user modeling component, 54  
user modeling server, 50  
user modeling shell, 48  
user modeling system, 1, 47  
User Modeling Tool, 49  
user profile, 2  
UserML, 354, 361  
UserQL, 361  
users, 2  
utility function, 299  
variable, 96  
vector space, 236  
*Verbal*, 58, 205, 209  
*Verbalizer*, 206  
Vermunt model, 209  
vertex, 277  
vertical data format, 226  
views, 53  
visited attribute, 115  
*Visual*, 58, 205, 209  
Viterbi algorithm, 215  
web-based learning, 1  
weight vector, 237  
 $weight(v_{ij})$ , 280  
weighted graph, 103, 141, 279, 280  
weighted sum, 260  
weights, 104  
wet grass, 98  
*Wholist*, 206, 209  
within-component layer, 23  
Witkin model, 205  
WOW, 80  
XHTML, 67  
YACF. *See* Yet Another Clustering Formalism  
Yet Another Clustering Formalism, 29  
Zebra, 56, 59, 65  
Zebra control panel, 66  
Zebra server, 65  
 $\beta(x; a, b)$ . *See* beta density function  
 $\Gamma$ . *See* gamma function

# Research History

**Author:** Loc Nguyen  
**Sponsor:** Pro. Dr. Dong Thi Bich Thuy  
**Affiliation:** University of Science – Vietnam National University, Ho Chi Minh city, Vietnam  
**Final Version:** 3.3 build 2009.11.21  
**Advanced Version:** **4.3 build 2015.01.02**

Primary tasks	
<i>Version 1.0 build 2009.04.25</i>	<ul style="list-style-type: none"><li>- <i>Designing task 1:</i> designing the architecture of user model so-called Triangular Learner Model (TLM) that constituted of three sub-models: Knowledge (K), Learning Style (LS) and Learning History (LH).</li><li>- <i>Designing task 2:</i> designing the architecture of Zebra – A user modeling system for adaptive learning. Zebra has two engines: mining engine (ME) and belief network engine (BNE). Zebra also provides communication interfaces (CI) that allows users and adaptive systems to see or modify restrictedly TLM</li><li>- <i>Writing tasks:</i> collecting all relative papers and considering them as resource to compose report</li><li>- <i>Writing bug fixing:</i> checking spelling &amp; grammar</li></ul>
<i>Version 1.1 build 2009.04.26</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> structuring report, aggregating particular parts</li><li>- <i>Writing bug fixing:</i> checking spelling &amp; grammar, repainting figures, normalizing formulas, aligning tables</li></ul>
<i>Version 1.1 build 2009.05.07</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> composing section III.3.2: discovering user interests</li></ul>
<i>Version 1.2 build 2009.06.11</i>	<ul style="list-style-type: none"><li>- <i>Designing task:</i> the architectures of both Triangular Learner Model (TLM) and Zebra are completed.</li><li>- <i>Writing tasks:</i> did finish section discussing about user interest</li><li>- <i>Programming tasks:</i> did build up Zebra server, Zebra client. So the architectures of both Triangular Learner Model (TLM) and Zebra are implemented. <b>The program was finished basically.</b></li><li>- <i>Writing bug fixing:</i> not yet</li></ul>
<i>Version 1.3 build 2009.07.18</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> composing Bayesian approach and overview of ubiquitous user modeling (in chapter: Conclusion &amp; Future Trend) but not finish yet</li></ul>
<i>Version 1.3 build 2009.07.22</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> composing Bayesian approach and overview of ubiquitous user modeling (in chapter: Conclusion &amp; Future Trend) but not finish yet</li></ul>
<i>Version 1.3 build 2009.07.25</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> composing Bayesian approach and overview of ubiquitous user modeling (in chapter: Conclusion &amp; Future Trend) but not finish yet</li></ul>
<i>Version 1.3 build 2009.07.27</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> concluding Bayesian approach (in chapter: Conclusion &amp; Future Trend) but not finish yet</li></ul>
<i>Version 1.3 build 2009.07.29</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> finish the evaluation of Bayesian network but not finish the assessment of such network yet.</li></ul>
<i>Version 1.3 build 2009.08.01</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> composing CAT in the assessment of Bayesian network.</li></ul>
<i>Version 1.4 build 2009.08.02</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> doing finish the draft of thesis but still missing many tables and figures in the last chapter.</li></ul>
<i>Version 1.4 build 2009.08.05</i>	<ul style="list-style-type: none"><li>- <i>Writing tasks:</i> All tables and figures in the last chapter are included. Doing finish sections Acknowledgement, Abstract, Reference &amp; Appendix. <b>The thesis was finished basically.</b></li></ul>

<i>Version 1.4 build 2009.08.11</i>	- <i>Writing tasks:</i> composing state of the art of Bayesian network user model (section I.3)
<i>Version 1.5 build 2009.08.14</i>	- <i>Writing tasks:</i> do finishing the state of the art of Bayesian network user model (section I.3)
<i>Version 1.5 build 2009.08.16</i>	- <i>Writing tasks:</i> adding a little content, repairing some thing
<i>Version 1.5 build 2009.08.25</i>	- <i>Writing tasks:</i> doing finish important report slides
<i>Version 1.5 build 2009.08.27</i>	- <i>Writing tasks:</i> drafting the likelihood estimation technique used to specify prior probabilities (new section III.1.5)
<i>Version 1.5 build 2009.08.30</i>	- <i>Writing tasks:</i> doing finish the likelihood estimation technique (III.1.5)
<i>Version 1.6 build 2009.08.31</i>	- <i>Writing tasks:</i> re-arranging all chapters, items and decorating the form of thesis. <b>The thesis was finished totally.</b> - <i>Programming tasks:</i> re-arranging source
<i>Version 1.6 build 2009.09.04</i>	- <i>Writing tasks:</i> fixing the example in Maximum Likelihood Estimation method - <i>Programming tasks:</i> do finishing the Maximum Likelihood Estimation algorithm and optimizing the Expectation Maximization algorithm
<i>Version 1.7 build 2009.09.09</i>	- <i>Writing tasks:</i> checking spelling and grammar - <i>Programming tasks:</i> re-arranging and optimizing source code
<i>Version 1.8 build 2009.09.14</i>	- <i>Writing tasks:</i> checking spelling and grammar - <i>Programming tasks:</i> re-arranging and optimizing source code. <b>The program was finished totally.</b>
<i>Version 1.9 build 2009.09.16</i>	- <i>Writing tasks:</i> checking grammar and decorating representation - <i>Programming tasks:</i> programming some utility services
<i>Version 2.0 build 2009.09.18</i>	- <i>Writing tasks:</i> checking grammar, composing new representation slides - <i>Programming tasks:</i> fixing bug because there are some fatal errors in the control panel GUI
<i>Version 2.0 build 2009.09.22</i>	- <i>Writing tasks:</i> improving the section IV.1.2.2: "Towards Adaptive Computerized Testing". - <i>Programming tasks:</i> fixing bug and optimizing source code
<i>Version 2.0 build 2009.09.24</i>	- <i>Learning material composing task:</i> designing the concepts & media for the course "Java tutorial" based on Sun Java Tutorial so as to test the demo - <i>Programming tasks:</i> fixing bug and optimizing source code - <i>Researching tasks:</i> researching the computer-based testing technique and integrating such technique into Zebra
<i>Version 2.0 build 2009.09.27</i>	- <i>Learning material composing task:</i> designing the concepts & media for the course "Java tutorial" based on Sun Java Tutorial so as to test the demo - <i>Researching task 1:</i> splitting overlay Bayesian network according to supervisor's advice (so difficult but more effective) - <i>Researching task 2:</i> researching the computer-based testing technique and integrating such technique into Zebra
<i>Version 2.1 build 2009.09.29</i>	- <i>Learning material composing task:</i> doing finish designing the concepts & media for the course "Java tutorial" 70% - <i>Researching task 1:</i> doing finish splitting overlay Bayesian network - <i>Researching task 2:</i> doing finish integrating the computer-based testing technique into Zebra
<i>Version 2.4 build 2009.10.07</i>	- <i>Writing task:</i> composing some representation slides about adaptive techniques - <i>Programming task:</i> test evidences now has more values (E.g: [1..10]) instead of binary variables.
<i>Version 2.6 build 2009.10.09</i>	- <i>Designing task:</i> improving the physical architecture of the adaptive learning system WOW! - <i>Programming task:</i> doing finish integrating the computer-based testing technique into Zebra
<i>Version 2.9 build 2009.10.11</i>	- <i>Designing task:</i> The physical architecture of the adaptive learning system WOW! is improved - <i>Programming task:</i> doing finish improving the physical architecture of the adaptive learning system WOW!

<i>Version 3.0 build 2009.10.12</i>	<ul style="list-style-type: none"> <li>- <i>Programming task:</i> fixing and improving Zebra</li> <li>- <i>Designing task:</i> Improving physical architecture of Zebra</li> </ul>
<i>Version 3.0 build 2009.10.14</i>	<ul style="list-style-type: none"> <li>- <i>Programming task 1:</i> fixing new fatal errors in Zebra server control panel when the physical architecture was changed</li> <li>- <i>Programming task 2:</i> enhancing EM algorithm when evaluating (assessing) user knowledge</li> <li>- <i>Programming task 3:</i> fixing Dynamic Bayesian network</li> </ul>
<i>Version 3.0 build 2009.10.16</i>	<ul style="list-style-type: none"> <li>- <i>Learning material composing task:</i> doing finish basically designing the concepts &amp; media for the course “Java tutorial”</li> <li>- <i>Programming task:</i> enhancing Dynamic Bayesian network</li> <li>- <i>Testing task:</i> testing Bayesian network, EM for determining prior probabilities, dynamic Bayesian network when evaluating user knowledge</li> </ul>
<i>Version 3.0 build 2009.10.17</i>	<ul style="list-style-type: none"> <li>- <i>Programming task:</i> fixing some bugs in Zebra server control panel</li> <li>- <i>Testing task:</i> testing Bayesian network, EM for determining prior probabilities, dynamic Bayesian network when evaluating user knowledge</li> </ul>
<i>Version 3.1 build 2009.10.21</i>	<ul style="list-style-type: none"> <li>- <i>Programming task 1:</i> improving pattern mining algorithm</li> <li>- <i>Programming task 2:</i> improving speed of Zebra. E.g: The first student doesn't need to wait for booting process of Zebra server</li> <li>- <i>Bug fixing:</i> fixing some bugs in Zebra control panel</li> </ul>
<i>Version 3.1 build 2009.10.22</i>	<ul style="list-style-type: none"> <li>- <i>Writing tasks:</i> checking spelling and grammar, adding a little content</li> </ul>
<i>Version 3.2 build 2009.10.26</i>	<ul style="list-style-type: none"> <li>- <i>Programming task:</i> add the new module “Searching Engine” into WOW! so as to allow students to search their considerable learning materials</li> </ul>
<i>Version 3.2 build 2009.10.27</i>	<ul style="list-style-type: none"> <li>- <i>Bug fixing:</i> fixing some bugs in module “Searching Engine”</li> </ul>
<i>Version 3.2 build 2009.10.29</i>	<ul style="list-style-type: none"> <li>- <i>Bug fixing:</i> fixing some bugs in module “Searching Engine”</li> </ul>
<i>Version 3.2 build 2009.10.31</i>	<ul style="list-style-type: none"> <li>- <i>Programming task:</i> enhancing the ajax GUI of module “Searching Engine”</li> </ul>
<i>Version 3.2 build 2009.11.05</i>	<ul style="list-style-type: none"> <li>- <i>Bug fixing:</i> fixing some bugs in the interface of module “Collaborative Learning”. Namely, when logging in WOW by more than one accounts (for example: userid1, userid2) at the same computer with the same browser; The userid2 can't using collaborative function (so as to chat, discuss, etc.).</li> <li>- <i>Programming tasks:</i> optimizing source code</li> </ul>
<i>Version 3.2 build 2009.11.11</i>	<ul style="list-style-type: none"> <li>- <i>Writing tasks:</i> doing finish the presentation slide “Zebra: A Modeling System for Triangular Learner Model” that summarizes thesis</li> </ul>
<i>Version 3.2 build 2009.11.14</i>	<ul style="list-style-type: none"> <li>- <i>Writing tasks:</i> composing section III.3.3 “constructing user group or user community” (user model clustering)</li> <li>- <i>Programming tasks:</i> fixing bugs and optimizing source code</li> <li>- <i>Testing task:</i> testing Zebra and WOW!</li> </ul>
<i>Version 3.3 build 2009.11.18</i>	<ul style="list-style-type: none"> <li>- <i>Writing tasks:</i> doing finish section III.3.3 “constructing user group or user community”</li> <li>- <i>Programming tasks:</i> fixing bugs and optimizing source code</li> <li>- <i>Testing task:</i> testing Zebra and WOW!</li> </ul>
<i>Final version 3.3 build 2009.11.21</i>	<ul style="list-style-type: none"> <li>- <b><i>Writing tasks: doing finish checking spelling and grammar in thesis</i></b></li> <li>- <b><i>Programming tasks: doing finish fixing bugs and optimizing source code in program</i></b></li> <li>- <b><i>Testing task: doing finish testing Zebra and WOW!</i></b></li> </ul>

## Extension 1

<i>Version 3.4 build 2009.11.26</i>	<ul style="list-style-type: none"> <li>- <i>Learning material composing task:</i> doing repair tests in Java tutorial course</li> <li>- <i>Programming tasks:</i> adding function that allows students to see their knowledge evaluation in form of hierarchical domain</li> </ul>
<i>Version 3.4 build 2009.12.21</i>	<ul style="list-style-type: none"> <li>- <i>Programming tasks:</i> Enhancing Zebra server control panel</li> </ul>
<i>Version 3.4 build 2009.12.25</i>	<ul style="list-style-type: none"> <li>- <i>Programming tasks:</i> Finished feedback module which is responsible for collecting students' feedbacks about system and performing statistical tasks so as to evaluate system</li> </ul>

<i>Version 3.4 build 2009.12.26</i>	- <i>Bug fixing and testing tasks:</i> Fixing bugs and testing feedback module
<i>Version 3.5 build 2009.12.31</i>	- <i>Programming tasks 1:</i> Did enhance the whole source code of WOW! and Zebra - <i>Programming tasks 2:</i> Adding more utilities to Zebra server control panel.
<i>Version 3.5 build 2010.01.01</i>	- <i>Programming tasks:</i> improving Zebra server control panel totally
<i>Version 3.5 build 2010.01.03</i>	- <i>Programming tasks 1:</i> discovering user interest based on document classification. This function is described in section III.3.2 in this thesis. Therefore, the series of user access in his/her history are modeled as documents; so user is referred indirectly to as document and user interests are classes that such documents are belong to. - <i>Programming tasks 2:</i> implementing the function that logs users' searching history so that each user is modeled as a user document vector. Solving some problems relating to the term frequency of user vector and the difference between user's query and keywords necessary for classification.
<i>Version 3.5 build 2010.01.04</i>	- <i>Bug fixing tasks:</i> fixing some errors occurring in classification algorithm (Decision Tree) and bugs in word segmentation. Zebra use inside word segmentation module for English language. Because Zebra doesn't use another open source tool, it has some limitation.
<i>Version 3.6 build 2010.01.15</i>	- <i>Programming tasks:</i> Supporting SOA and Web services. It means that the Communication Interface (CI) in Zebra architecture which allows users or adaptive systems to access or query TLM limitedly now supports networking protocols such as RMI, SOAP (web service), HTTP, and SOCKET. Note that CI is also called so-called TriUMQuery or Query Delegator. - <i>Bug fixing tasks:</i> fixing some errors in Zebra control panel.
<i>Version 3.6 build 2010.01.22</i>	- <i>Bug fixing tasks:</i> fixing some errors occurring in CI Web services. - <i>Programming task:</i> enhancing Zebra client that takes full advantage of communication interface (CI)
<i>Version 3.6 build 2010.01.26</i>	- <i>Programming tasks:</i> enhancing Zebra control panel.
<i>Version 3.7 build 2010.01.28</i>	- <i>Programming task 1:</i> supporting report on user's learning process. - <i>Programming task 2:</i> improving Zebra client
<i>Version 3.7 build 2010.01.30</i>	- <i>Programming task:</i> did finish the report function. There are two kinds of report: user report and course port. User report shows information and statistical data about user. Course report shows mining result, belief network inference, and statistical data in specific course. - <i>Bug fixing tasks:</i> fixing some errors occurring in report function

## Extension 2

<i>Version 3.8 build 2010.02.01</i>	- <i>Programming task:</i> enhancing Zebra control panel. Especially, the GUI that shows Triangular Learner Model (TLM) is now improved in order to run fast and is repaired so as to be friendly.
<i>Version 3.8 build 2010.02.02</i>	- <i>Review task:</i> reviewing source code and changing some method names, class names, etc. and adding some utility methods.
<i>Version 3.8 build 2010.02.03</i>	- <i>Programming task:</i> adding finding user function into Zebra control panel.
<i>Version 3.8 build 2010.02.05</i>	- <i>Programming task:</i> adding more utility methods and changing some method names, class names.
<i>Version 3.9 build 2010.02.11</i>	- <i>Programming task 1:</i> reviewing and improving source code. - <i>Programming task 2:</i> Programming sending e-mail function so as to support mailing list in future. - <i>Bug fixing task:</i> the engines (ME and BNE) of Zebra are composed physically by many threads so-called daemons or timer tasks. In case that the period of each daemon is so short (less than 1 minute), such daemon will run so fast and it consumes much more memory. There is a question: "Why does each daemon take much more memory when it runs fast?" When daemon runs fast, it makes many mining tasks. Because each mining task requires a

	lot of memory and creates many objects, it is so many waste objects that the Java garbage collection can destroy timely. In consequences, there isn't enough heap memory. Additionally, that many users access Zebra concurrently can make the daemon boom; so Java will raise <i>the out of memory error</i> . I have partially fixed this error; however you should keep the period of each daemon larger than 10 minutes by setting the variable "UPDATE_LEARNING_HISTORY_INTERVAL" in "zebraconfig.xml". Even you can set this period be daily, weekly, or monthly.
<i>Version 3.9 build 2010.02.16</i>	- <i>Bug fixing task:</i> Zebra server now run steadily and stably. The out of memory error is fixed totally.
<i>Version 3.9 build 2010.02.18</i>	- <i>Programming task:</i> sending e-mail function was finished. The sending mail SMTP host is declared in "zebraconfig.xml" as variable "MAIL_SMTP_HOST"
<i>Version 3.10 build 2010.02.20</i>	- <i>Programming task 1:</i> mailing list module was finished totally. It allows system to send report or send any information to users periodically. According to default setting, this period is daily but you can change it by modifying the variable "MAIL_MAILING_LIST_INTERVAL" in file "zebraconfig.xml". Users can submit to or withdraw from mailing list through Communication Interface (CI) so-called TriUMQuery or Query Delegator. - <i>Programming task 2:</i> enhancing source code.
<i>Version 3.11 build 2013.06.12</i>	- <i>Writing tasks:</i> Re-structuring thesis, writing additional chapter "Evaluation of Triangular Learner Model".
<i>Version 3.11 build 2013.08.03</i>	- <i>Writing tasks:</i> Checking thesis thoroughly for claims, content and grammar. It is perfect.

### Extension 3

	<ul style="list-style-type: none"> <li>- <i>Writing task 1:</i> Re-organize the whole research.</li> <li>- <i>Writing task 2:</i> Completing references over the whole research.</li> <li>- <i>Writing task 3:</i> Writing additional contents in sub-section III.1.5.2 (Beta likelihood estimation).</li> <li>- <i>Writing task 4:</i> Writing additional contents in sub-section III.3.2.2 (Methods of document classification), especially, support vector machine.</li> <li>- <i>Writing task 5:</i> Writing additional contents in sub-section IV.1.2.2 (Towards the Computerized Adaptive Testing).</li> <li>- <i>Writing task 6:</i> Restoring contents in sub-section III.3.1.1 (Approaches of sequential pattern mining) and section V.2 (Towards ubiquitous user modeling).</li> <li>- <i>Writing task 7:</i> All tables, figures, and formulas are indexed.</li> <li>- <i>Writing task 8:</i> Fixing writing errors, creating figures, adding more examples, and enhancing representation over the whole research.</li>   <li>- <i>Creative work 1:</i> Inventing new algorithm for calculating bias of support vector machine in part III.3.2.2.1 (Document classification based on support vector machine).</li> <li>- <i>Creative work 2:</i> Inventing new formula for beta likelihood estimation in sub-section III.1.5.2 (Beta likelihood estimation).</li> <li>- <i>Creative work 3:</i> Inventing new formula for estimating examinee's ability in sub-section IV.1.2.2.2 (Maximum likelihood estimation for CAT).</li> </ul>
<i>Version 4 build 2014.11.24</i>	<ul style="list-style-type: none"> <li>- <i>Writing task 1:</i> Enhancing sub-section III.3.2.2 (Methods of document classification). Composing examples and figures for support vector machine, decision tree, and neural network. Decision tree is described in detailed.</li> <li>- <i>Writing task 2:</i> Composing examples for advanced CAT algorithm</li>   <li>- <i>Creative work 1:</i> Inventing the advanced CAT algorithm for multi-user test in part IV.1.2.2.3 (New version of CAT algorithm based on MLE).</li>   <li>- <i>Programming task 1:</i> Implementing document classification based on neural</li> </ul>
<i>Version 4.1 build 2014.12.18</i>	

	<p>network for discovering user interests. Besides computer softwares such as user modeling system Zebra and adaptive learning system WOW, the new module so-called <b>Zebramples</b> is associated to this book so as to implement examples insides the book.</p> <ul style="list-style-type: none"> <li>- <i>Programming task 2:</i> Implementing the advanced CAT algorithm inside Zebramples.</li> </ul>
<i>Version 4.2 build 2014.12.27</i>	<ul style="list-style-type: none"> <li>- <i>Writing task 1:</i> Composing the new sub-section II.2.4 (Implementation of Zebra) when Zebra is implemented as computer software.</li> <li>- <i>Writing task 2:</i> Composing the new sub-section II.2.4 (New version of simple equations whose solutions are parameter estimators).</li> <li>- <i>Creative work 1:</i> Inventing the new version of simple equations whose solutions are parameter estimators. These estimators are used to specify prior probabilities of Bayesian overlay model. This invention is described in sub-section III.1.5.5 (New version of simple equations whose solutions are parameter estimators).</li> <li>- <i>Programming task 1:</i> Implementing the iterative algorithm solving new simple equations whose solutions are parameter estimators (sub-section III.1.5.5).</li> </ul>
<i>Version 4.3 build 2015.01.02</i>	<ul style="list-style-type: none"> <li>- <i>Writing task 1:</i> Indexing all terms, keywords and completing appendix D. Index.</li> <li>- <i>Writing task 2:</i> Creating more figures.</li> <li>- <i>Writing task 3:</i> Composing an example for <math>k</math>-medoid algorithm described in sub-section III.3.3.3 (Similarity measures for clustering algorithms).</li> </ul>