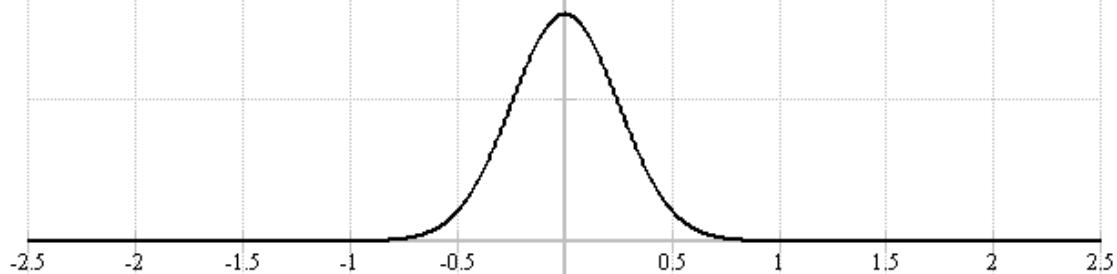


MATHEMATICAL APPROACHES



USER MODELING



First Edition by Loc Nguyen

Mathematical Approaches to User Modeling

First edition by Loc Nguyen

Email: ng_phloc@yahoo.com

Website: www.locnguyen.net

Donate: www.locnguyen.net/donate

Phone: +84-97-5250362

Research version 4.8 build 2018.04

The book is originated from a research work “A User Modeling System for Adaptive Learning” submitted for the Degree of Doctor of Philosophy in Computer Science.

The book shows my gratitude to my supervisor Prof. Dr. Dong, Bich-Thuy T.

Acknowledgement

I am very grateful to Prof. Dr. Dong, Bich-Thuy T. for providing me invaluable help, knowledge, experiences and encouragement during the time of research. She is my supervisor and sponsor. If it were not for her help, I would not complete the research 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 express my deepest gratitude for my family and friends who have supported me in difficult times. I also express my warmest gratitude to authors who provided excellent works to which I refer during doing this research. I also express my gratitude to all people who gave me the possibility to complete this research.

Acronyms and Notations

BN	Bayesian Network.
BN Engine	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.
$f'(x)$	
$d(f(x))$	First-order derivative of function f with regard to variable x . Especially, the notation $d(f(x))$ also denotes differential of $f(x)$.
$\frac{df(x)}{dx}$	
$\frac{d}{dx}f(x)$	
$f''(x)$	Second-order derivative of function f with regard to variable x .
HMM	Hidden Markov model.
idf	inverse document frequency.
itemset	Item set.
Java	An object-oriented programming language https://www.oracle.com/java
$L(\theta)$	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.
$n!$	Factorial function of non-negative integer n $n! = 1 * 2 * 3 * \dots * n$
PDF	Probability Density Function.
$P(\cdot)$	Probability.
$P(X)$	Probability of random variable or event X .
$P(Y/X)$	Conditional probability of Y given X .
sim	Similarity measure.
tf	Term frequency.
TLM	Triangular Learner Model.
UMS	User Modeling Shell/System/Server/Service.
Zebra	User modeling system Zebra.
\circ	Dot product or scalar product of two vectors, for example, $(x_1, y_1) \circ (x_2, y_2) = x_1 * x_2 + y_1 * y_2$
$\beta(x; a, b)$	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 $\Gamma(\cdot)$ is gamma function.
γ	Euler-Mascheroni constant $\gamma \approx 0.577215$.
γ	Adaptation criterion for evaluating adaptive learning model.
$\Gamma(x)$	Gamma function $\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$
ψ, ψ_1	Digamma function, trigamma function $\begin{aligned} \psi(x) &= d \left(\ln(\Gamma(x)) \right) = \frac{\Gamma'(x)}{\Gamma(x)} \\ &= \int_0^{+\infty} \left(\frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}} \right) dt \\ \psi_1(x) &= \psi'(x) = \int_0^{+\infty} \frac{te^{-xt}}{1-e^{-t}} dt \end{aligned}$ Where $\Gamma(\cdot)$ 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 .

$\frac{\partial^2 f}{\partial x \partial y}$	
∇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) . If $f(x)$ has primitive function and let $F(x)$ be primitive function of $f(x)$, we have: $\int_a^b f(x) dx = F(x) \Big _a^b = \lim_{x \rightarrow b} F(x) - \lim_{x \rightarrow a} F(x)$ Note that a and b can be infinities such as $-\infty, +\infty$.
$\int f(x) dx$	Indefinite integral of function $f(x)$. Let $F(x)$ denote primitive function of $f(x)$, we have: $\int f(x) dx = F(x)$
$\overline{a, b}$	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 $i \in \{1, 2, 3, 4, 5\}$.
$x^* = \underset{x}{\operatorname{argmax}} f(x)$ $x^* = \underset{x}{\operatorname{argmin}} f(x)$	This is optimization problem which finds out x^* that maximizes or minimizes function $f(x)$.
$f^* = \max_x f(x)$ $f^* = \min_x f(x)$	The value f^* is maximum (minimum) value of function $f(x)$.
$<< (>>)$	Much smaller (larger) than. For example, number 0 is much smaller than number 1000, we have $0 << 1000$ and $1000 >> 0$.
$ X $	Absolute value of number X , length of vector X , determinant of matrix X , cardinality of set X .
(x_1, x_2, \dots, x_n)	Row vector having n elements.
$(x_1, x_2, \dots, x_n)^T$ $= \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$	Column vector having n elements. Note that the superscript T denotes transposition operation in vector and matrix.
$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$	Matrix with m rows and n columns.
$A \Rightarrow B$	A implies B .
$A \Leftrightarrow B$	A is equivalent to B .

Preface

Nowadays modern society requires that 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, and background are the most important to adaptive system. These characteristics are organized in a structure 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 mathematical 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. In other words, the preeminence of this research is to provide mathematical approaches to user modeling study.

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 seven chapters:

- Chapter I is a survey of user model, user modeling, and adaptive learning.

- Chapter II introduces the general architecture of the proposed user modeling system Zebra and TLM.
- Chapter III, IV, V describes three sub-models of TLM such as knowledge sub-model, learning style sub-model, and learning history sub-model in full of mathematical formulas and fundamental methods. These are the most important chapters.
- Chapter VI gives some approaches to evaluate TLM and Zebra.
- Chapter VII summarizes the research and discusses future trend of Zebra.

Please pay attention to chapters II, III, IV, and V. Chapter II gives essential aspects of TLM and Zebra according to general viewpoint. Chapter III, IV, and V are the heaviest ones filled with a lot of knowledge. It is easy for you to understand TLM and Zebra if you study chapters III, IV, and V cautiously. The whole book is available at internet link:

<http://mum.locnguyen.net>

In general, the *first difference* of the book is to introduce my innovative works in mathematics, probability, user modeling, machine learning, data mining, and adaptive learning. The *second difference* is that the user modeling system Zebra is implemented as computer software associated to the book, available at internet link:

<http://zebra.locnguyen.net>

Moreover, examples, mathematical formulas, and mathematical models inside the book are implemented separately and integrated into another computer software.

Have a nice day! Do enjoy the book!

An Giang – Vietnam, January 03, 2015
Best regards,

Prof. Loc Nguyen, PhD, MD, MBA

Poet and Mathematician,

Director of International Engineering and Technology Institute (IETI),

ORCID 0000-0001-5192-8106

ResearcherID F-7019-2014

Organization International Engineering and Technology Institute (IETI), Vietnam

Home page www.locnguyen.net

Contact address 1/4B Ton Duc Thang street, My Binh ward, Long Xuyen city,
An Giang province 881092, Vietnam

Email ng_phloc@yahoo.com

Phone 84-975250362

Contents

Chapter I. User Model and Adaptive Learning	1
I.1. User Model	1
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.....	5
I.1.2.2. Overlay model.....	6
I.1.2.3. Differential model	7
I.1.2.4. Perturbation model	8
I.1.2.5. Plan model	8
I.2. Adaptive Learning	9
I.2.1. Classification of adaptive learning systems.....	10
I.2.2. Intelligent Tutoring System (ITS).....	11
I.2.2.1. Architecture of ITS	12
I.2.2.2. Characteristics of pedagogical module	13
I.2.2.3. An example of ITS: ANATOM-TUTOR.....	14
I.2.3. Adaptive Educational Hypermedia System (AEHS)	15
I.2.3.1. Architecture of AEHS	17
I.2.3.2. Characteristics of AEHS	19
I.2.3.3. An example of AEHS: Adaptive Hypermedia for All (AHA!)	20
I.2.4. Evaluation of existing ITS and AEHS	22
I.3. Bayesian network user model	23
I.3.1. KBS hyperbook system.....	23
I.3.1.1. Yet Another Clustering Formalism (YACF).....	26
I.3.1.2. How to define CPT (s) of cluster node and child nodes	27
I.3.2. Andes	29
I.3.2.1. Solution graph.....	30
I.3.2.2. Bayesian network in Andes	32
I.3.3. SQL-Tutor and constraint-based modeling	36
I.3.4. Data-centric approach	40
Chapter II. A User Modeling System for Triangular Learner Model	42
II.1. Existing user modeling systems	42
II.1.1. Early user modeling systems	43
II.1.2. User modeling shells.....	43
II.1.3. User modeling servers	45
II.2. Zebra: A User Modeling System for Triangular Learner Model	50
II.2.1. Triangular Learner Model	51
II.2.2. The architecture of Zebra	53
II.2.3. Interaction between Zebra and adaptive applications	56
II.2.4. Implementation of Zebra.....	58
II.3. Conclusion.....	87
Chapter III. Knowledge sub-model	88
III.1. Combination of Bayesian network and overlay model in building up knowledge sub-model	89
III.1.1. Bayesian network	89
III.1.2. Applying Bayesian network to overlay model	97
III.1.3. SIGMA-gate inference	102

III.1.4. Relationship conversion in Bayesian network	110
III.1.4.1. Diagnostic relationship	111
III.1.4.2. X-gate inferences.....	116
III.1.4.3. Multi-hypothesis diagnostic relationship	134
III.1.5. Evaluation	149
III.2. Incorporate Bayesian inference into adaptation rules	149
III.3. Evolution of Bayesian overlay model.....	157
III.3.1. Learning parameters in Bayesian model.....	158
III.3.2. Learning parameters in case of missing data	173
III.3.3. An example of learning parameters.....	178
III.4. Improving knowledge sub-model by using dynamic Bayesian network	187
III.4.1. Dynamic Bayesian network	188
III.4.2. Using dynamic Bayesian network to model user's knowledge.....	190
III.4.3. Evaluation	205
III.5. Specifying prior probabilities	206
III.5.1. Maximum likelihood estimation	208
III.5.2. Beta likelihood estimation	210
III.5.3. Algorithm to solve the equations whose solutions are parameter estimators ...	221
III.5.4. An example of how to specify prior probabilities	224
III.5.5. New version of the equations whose solutions are parameter estimators	228
III.5.6. Evaluation	233
Chapter IV. Learning style sub-model	235
IV.1. What learning styles are.....	235
IV.2. Learning style families	236
IV.2.1. Constitutionally based learning styles and preferences	236
IV.2.2. The cognitive structure	237
IV.2.3. Stable personal type	238
IV.2.4. Flexible stable learning preference	238
IV.3. Providing adaptation of learning materials to learning styles	241
IV.4. Hidden Markov model	243
IV.4.1. HMM evaluation problem	246
IV.4.2. HMM uncovering problem	257
IV.4.3. HMM learning problem.....	276
IV.4.3.1. EM algorithm	276
IV.4.3.2. Applying EM algorithm into solving learning problem	280
IV.5. Continuous observation hidden Markov model	310
IV.6. Applying hidden Markov model into building up learning style sub-model.....	366
IV.7. Evaluation	375
Chapter V. Learning history sub-model.....	376
V.1. Learning concept recommendation based on mining learning history ...	377
V.1.1. Approaches of sequential pattern mining	380
V.1.1.1. Candidate generation-and-test approach	380
V.1.1.2. Pattern-growth approach	385
V.1.2. A proposal of breaking sequential pattern in learning context.....	386
V.1.3. Evaluation	388
V.2. Discovering user interests by document classification.....	389
V.2.1. Vector model for representing documents	390
V.2.2. Methods of document classification	391
V.2.2.1. Document classification based on support vector machine	391
V.2.2.2. Document classification based on decision tree	417
V.2.2.3. Document classification based on neural network	425

V.2.3. Discovering user interests based on document classification	435
V.2.4. Evaluation	438
V.3. Constructing user groups or user communities.....	438
V.3.1. User model clustering	439
V.3.2. Overlay model clustering.....	444
V.3.2.1. In case that arcs in graph are weighted.....	446
V.3.2.2. In case that graph model is Bayesian network.....	448
V.3.3. Similarity measures for clustering algorithms	451
V.3.4. Evaluation	460
Chapter VI. Evaluation of Triangular Learner Model.....	461
VI.1. Evaluation of knowledge model	461
VI.1.1. Evaluation of Bayesian network	462
VI.1.2. Assessment of Bayesian network.....	464
VI.1.3. Towards the Computerized Adaptive Testing	466
VI.1.3.1. Overview of Computerized Adaptive Testing (CAT).....	467
VI.1.3.2. Maximum likelihood estimation (MLE) for CAT	469
VI.1.3.3. New version of CAT algorithm based on MLE.....	476
VI.2. Evaluation of adaptive learning model	492
VI.2.1. Evaluation criteria	493
VI.2.1.1. Calculating system criterion α	493
VI.2.1.2. Calculating academic criterion β	496
VI.2.1.3. Calculating adaptation criterion γ	497
VI.2.2. An evaluation scenario	498
Chapter VII. Future Trend of Triangular Learner Model	502
VII.1. Conclusion	502
VII.2. Towards ubiquitous user modeling	505
VII.2.1. Overview of ubiquitous user modeling	506
VII.2.1.1. User modeling and context-awareness	506
VII.2.1.2. Ubiquitous computing	507
VII.2.1.3. Semantic web	510
VII.2.2. Knowledge presentation of situation model	511
VII.2.3. Ubiquitous World	516
VII.2.3.1. User model ontologies.....	516
VII.2.3.2. Ubiquitous World.....	518
VII.2.4. Ubiquitous user model service	519
VII.2.4.1. Architecture of ubiquitous user model service	520
VII.2.4.2. Main work flow of ubiquitous user model service	521
VII.2.5. Incorporating Zebra into ubiquitous user model service	522
References	525
Appendices	539
A. List of Tables.....	539
B. List of Figures	543
C. List of Formulas.....	548
D. Index.....	555

Chapter I. User Model and Adaptive Learning

Chapter I gives us a survey of user model and adaptive learning before going into main works in chapters II, III, IV and V with assurance that the research proposes a user modeling system along with mathematical formulas and fundamental methods for adaptive learning. The preeminence of this research is to provide mathematical approaches to user modeling study. Essential concepts and methods relevant to user model and adaptive system are described in section I.1 and section I.2, respectively. Bayesian network is one important aspect of the research, so section I.3 is reserved for introducing some typical Bayesian network models. 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. Your attention please; some surveys in this book are made from many sources for a long time and hence, some references may be missed although I try my best to create a full list of references; readers are recommended to look up and read more additional documents.

I.1. User Model

Formerly, learning management systems (Wikipedia, Learning management system, 2014) such as Moodle (About Moodle - the open source learning platform, 2014) and 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 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 and Self (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.

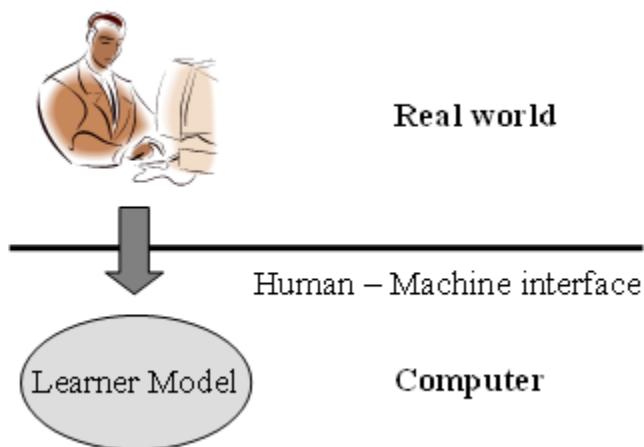


Figure I.1.1. User modeling

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.

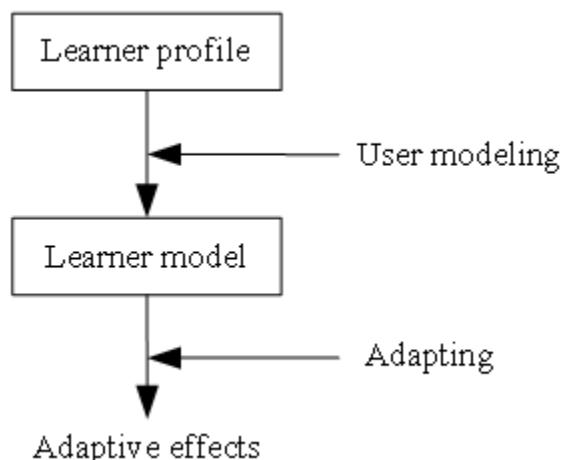


Figure I.1.2. Learner profile and learner model in adaptation

Because this research proposes a user modeling system for adaptive learning, it is relevant to both user modeling and adaptive learning. Thus, this chapter I 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. Note that the main content of sections I.1 and I.2 is extracted from the master thesis "*User Modeling and User Profiling in Adaptive E-learning Systems*" of the author Christoph Fröschl (Fröschl, 2005). I express my deep gratitude to the author Christoph Fröschl for providing her/his great research.

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. An 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 sub-section 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.* User interests are known simply as user's preferences, likes, and dislikes on something. 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.1.1.2.1 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>	Pragmatists have practical mind, prefer to try and test techniques relevant to problems (Stash, 2007, p. 106).
<i>Theorist</i>	Theorists think things through in logical steps, understand different facts into coherent theory (Stash, 2007, p. 106).

Table I.1.1.2.1. 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.1.1.2.2 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.1.1.2.2. 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 subsection 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 and Runger (Montgomery & Runger, 2003), “Artificial Intelligence: A Modern Approach” by authors Russell and Norvig (Russell & Norvig, 2003), “Machine Learning” by author Mitchell (Mitchell, 1997), and “Data Mining: Concepts and Techniques” by authors Han and Kamber (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, predefined stereotype is assigned to learner at abstract level. For example (see figure I.1.2.1.1), in Java tutorial course, students are divided into five groups, corresponding to five levels, 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 and experience, system will assign one of five levels to each learner and never change.

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

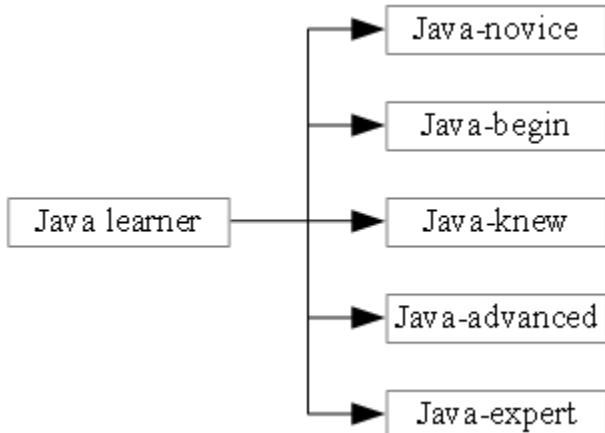


Figure I.1.2.1.1. An example of stereotypes of Java learner

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

- *Trigger* is used to activate 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’s mastery of 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 by stereotype “Java-novice” at the beginning of course but after learning process, student knew thoroughly Java, so her/his stereotype “Java-novice” is no longer suitable. Event “Learner does final Java test very well” is condition to retract her/his stereotype “Java-novice” and she/he will be assigned by 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 (Mayo, 2001, pp. 56-58). In other words, the user overlay model is a shot of comprehensive domain model. Domain model is constituted of 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 user’s knowledge about that element. This value is considered as the mastery of domain element ranging within 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.

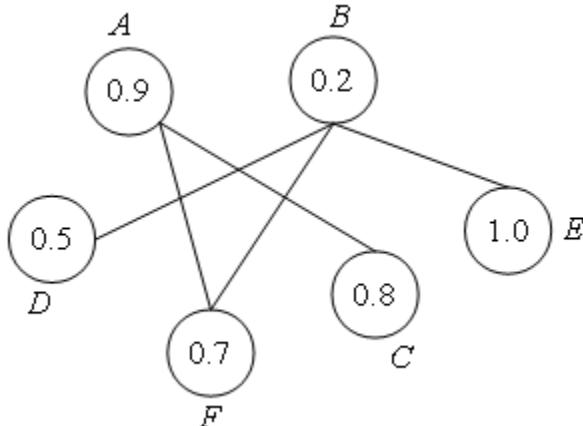


Figure I.1.2.2.1. An example of overlay model having six concepts

An example of overlay model is shown in figure I.1.2.2.1 in which there are six concepts A, B, C, D, E and F ; each concept (element) is attached by 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.1.2.2.1 depicts an 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 section III.1. Other systems applying Bayesian network into building up user model are introduced in section I.3. It is possible to say that Bayesian network is an advanced variant of overlay model.

I.1.2.3. Differential model

Overlay model is based on expert’s domain knowledge. It is requisite for tutor/teacher to suggest necessary knowledge to learner. That knowledge is called *expected knowledge*. In other words, 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. In detail, the differential model is instance of the class "fault model" (see subsection 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.1.2.4.1 (Mayo, 2001, p. 60), learner knowledge shown as shading area includes both correct knowledge and incorrect knowledge (mal-knowledge).

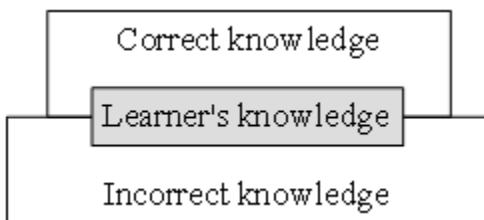


Figure I.1.2.4.1. Illustration of perturbation model

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 “Learner Model in Adaptive Learning” (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. In this section [I.2](#), I collect scientific resources to bring out an overview of adaptive learning systems along with their features (Nguyen L., State of the Art of Adaptive Learning, 2009). Main reference is the master thesis “User Modeling and User Profiling in Adaptive E-learning Systems” of author Christoph Fröschl (Fröschl, 2005). I express my deep gratitude to the author Christoph Fröschl for providing her/his great research.

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)
- 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, 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 (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 annotations 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) 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 attend 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 content taught, adaptive procedures and techniques for collecting, storing user characteristics and inferring new assumptions from them. General architecture of ITS shown in figure I.2.2.1.1 (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: goals, demography information, mastered knowledge, etc. and short-term information: whether or not students do exercises, visit a web site, etc. 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 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 raise error or do not work properly, 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.

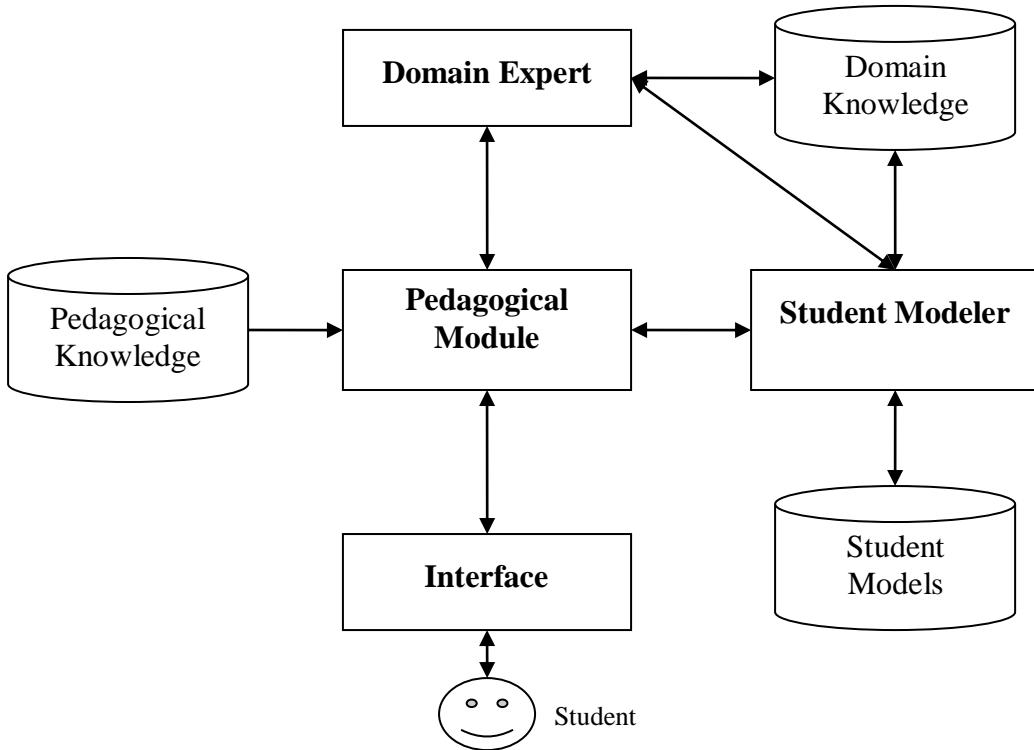


Figure I.2.2.1.1. General Architecture of ITS

I.2.2.2. Characteristics of pedagogical module

Author Wenger (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 author Wenger (Wenger, 1987), the core of ITS – pedagogical module provides two main adaptive tasks (makes decisions): diagnosis and didactics.

- *Diagnosis*: The ITS “diagnoses” students’ states at 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 (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 (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.2.2.3.1 (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.

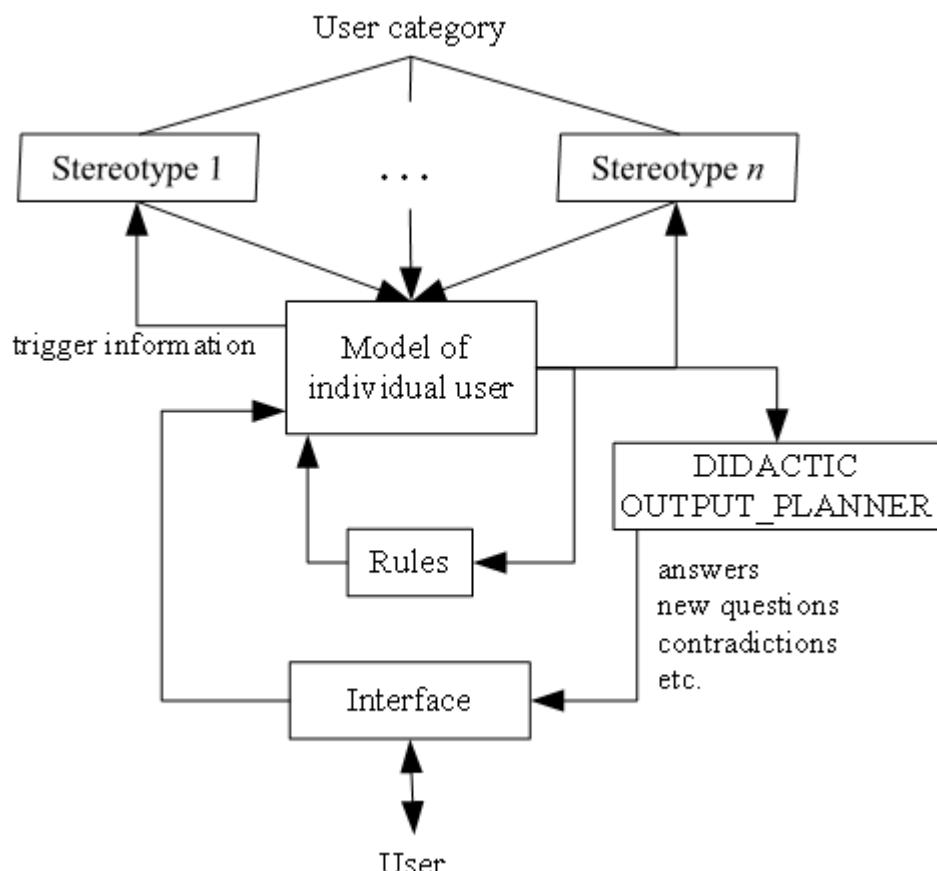


Figure I.2.2.3.1. User modeling component in ANATOM-TUTOR

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 author Brusilovsky (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.2.3.1 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.2.3.2 depicts adaptive presentation (Brusilovsky, 2001, p. 100).

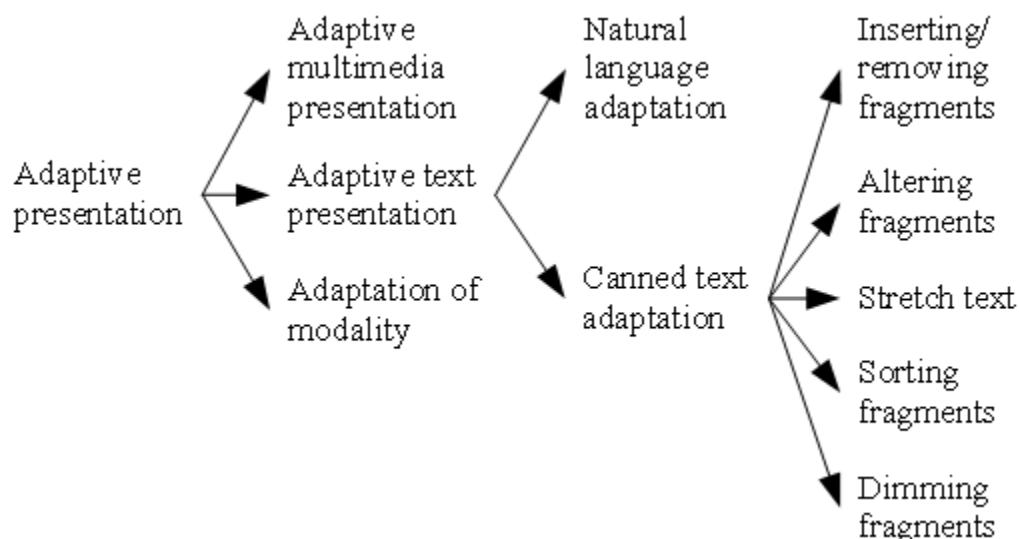


Figure I.2.3.1. Adaptive representation

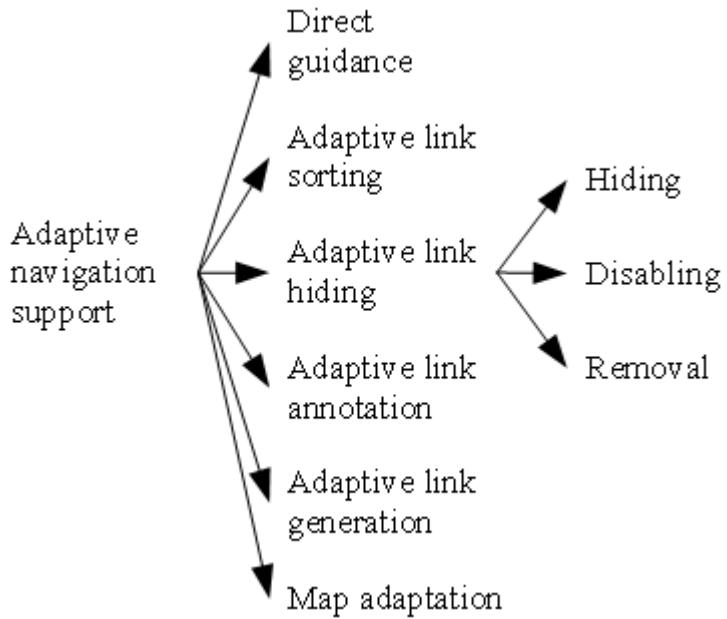


Figure I.2.3.2. Adaptive navigation

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.



Figure I.2.3.3. A typical example of adaptive navigation

The left pane of figure I.2.3.3 shows a typical example of adaptive navigation.

I.2.3.1. Architecture of AEHS

In general, the architecture of AEHS shown in figure I.2.3.1.1 (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.

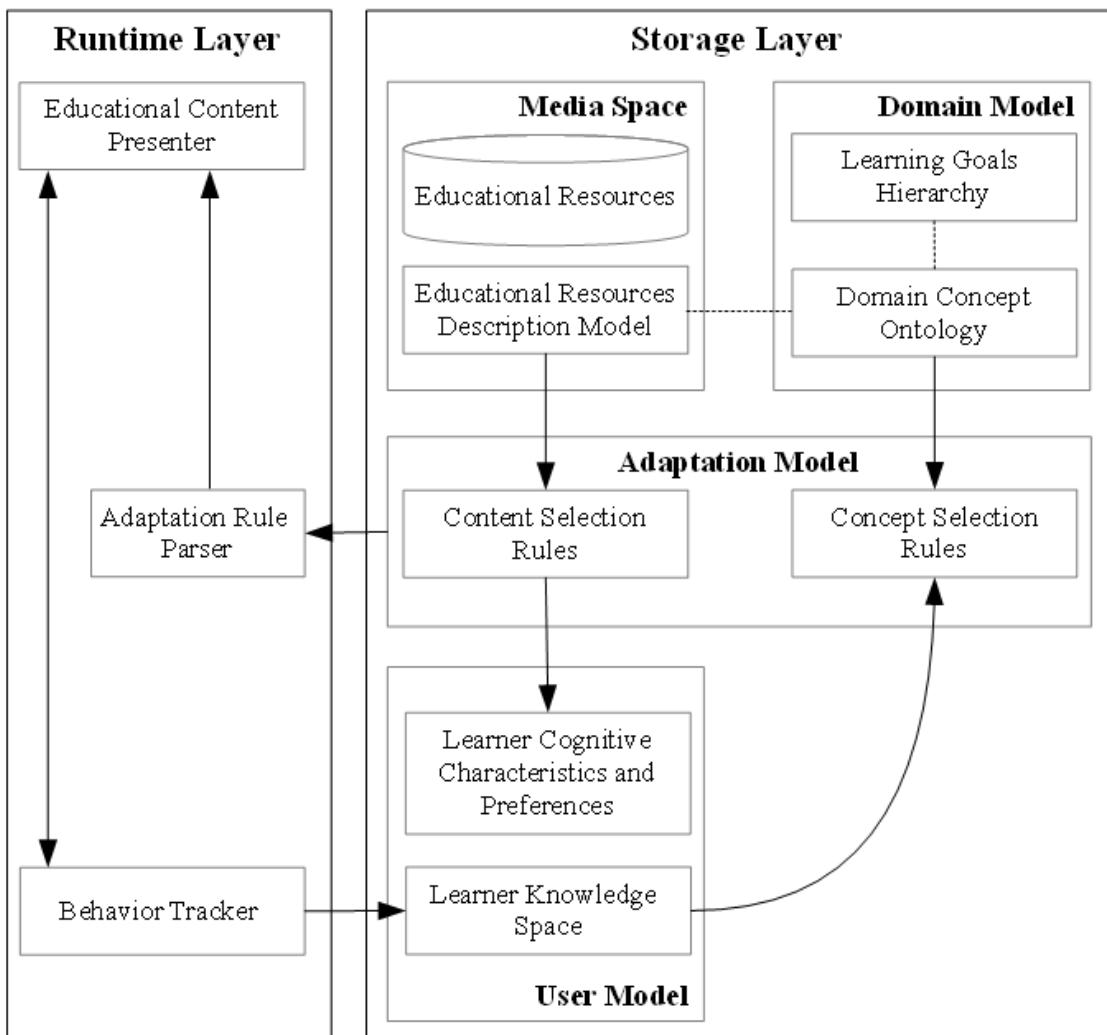


Figure I.2.3.1.1. General architecture of AEHS

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 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 concept selection rules and content selection rules. Concept selection rules are used to choose appropriate concepts from domain model. On the other hand, we apply content selection rules into choosing suitable educational resource from medial 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 and Nejdl (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 with 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_id₁, doc_id₂)* means that *doc_id₂* is a part of *doc_id₁*.
- Predicate *req(doc_id₁, doc_id₂)* means that *doc_id₂* is prerequisite of *doc_id₁*.
- Predicate *is_a(doc_id₁, doc_id₂)* expresses the taxonomy on documents.
- Predicate *is_dependent(doc_id₁, doc_id₂)* expresses the dependencies between documents.

So, every document is 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 is 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):

$$\begin{aligned} \forall \text{user_id}, \forall \text{doc_id}_1 (\forall \text{doc_id}_2, \text{preq}(\text{doc_id}_1, \text{doc_id}_2) \\ \Rightarrow \text{obs}(\text{doc_id}_2, \text{user_id}, \text{visited})) \\ \Rightarrow \text{learning_state}(\text{doc_id}_1, \text{user_id}, \text{recommended_for_reading}) \end{aligned}$$

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 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".

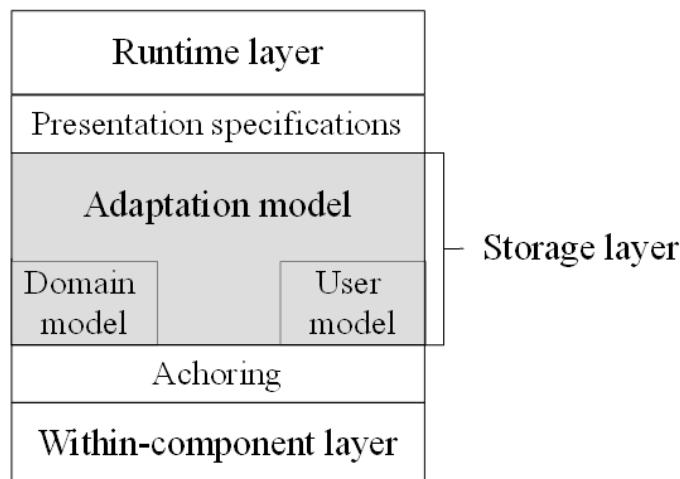


Figure I.2.3.3.1. AHAM – The architecture of AHA!

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.2.3.3.1 (De Bra, Houben, & Wu, 1999, p. 3) and figure I.2.3.3.2 (De Bra, Smits, & Stash, 2006) depict the architecture AHAM and the implemented framework of AHA!, respectively.

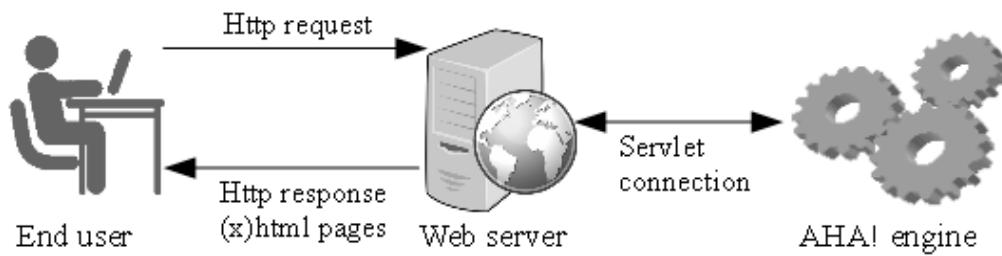


Figure I.2.3.3.2. Implemented framework of AHA!

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

- Domain model (DM) contains domain concepts associated with web learning resources. The relationships between concepts are also specified.
- 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 with 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.2.3.3.3 (De Bra, Smits, & Stash, 2006).

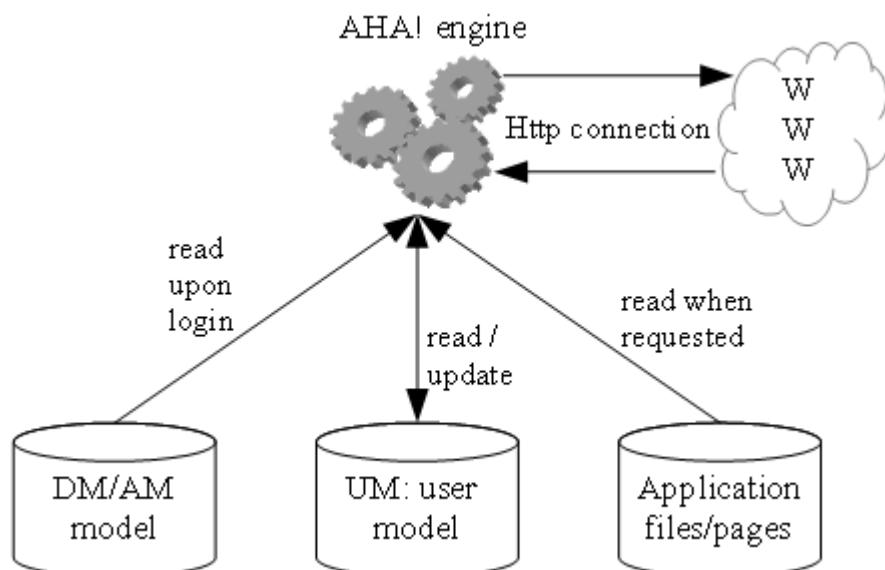


Figure I.2.3.3.3. AHA! engine

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 with updated attributes. So this process continues until no rule is triggered.
4. The resources associated with 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 a so-called Bayesian overlay model (see section III.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 L. , 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 section III.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 “Overview of Bayesian Network” (Nguyen L. , Overview of Bayesian Network, 2013) which is a good introduction to generic Bayesian network together with basic concepts, inference and learning techniques. Recall that overlay model is essential graph model whose nodes are knowledge elements, which leads to many approaches to build up overlay model from statistics to machine learning and one of them is Bayesian network method. Thus, it is possible to say that Bayesian network is an advanced variant of overlay model. Please read the excellent article “Bayesian network for student model engineering” by authors Millán, Loboda, and Pérez-de-la-Cruz (Millán, Loboda, & Pérez-de-la-Cruz, Bayesian networks for student model engineering, 2010) for comprehending applying Bayesian network user model in learning context.

I.3.1. KBS hyperbook system

KBS hyperbook system is developed by author Henze (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 (Henze, 2000, p. 52) defines the dependency graph as the neighbouring graph in which the nodes are $KI(s)$ and the arcs represent partial order among $KI(s)$. Namely, that the arc from node B to node A and there is no node Z 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 $KI(s)$ called main $KI(s)$ that have no parent. Main $KI(s)$ in the same level are assumed to be mutually independent. Figure I.3.1.1 (Henze, 2000, p. 52) depicts an example of dependency graph mentioning basic concepts of Java programming language (Oracle, Java language).

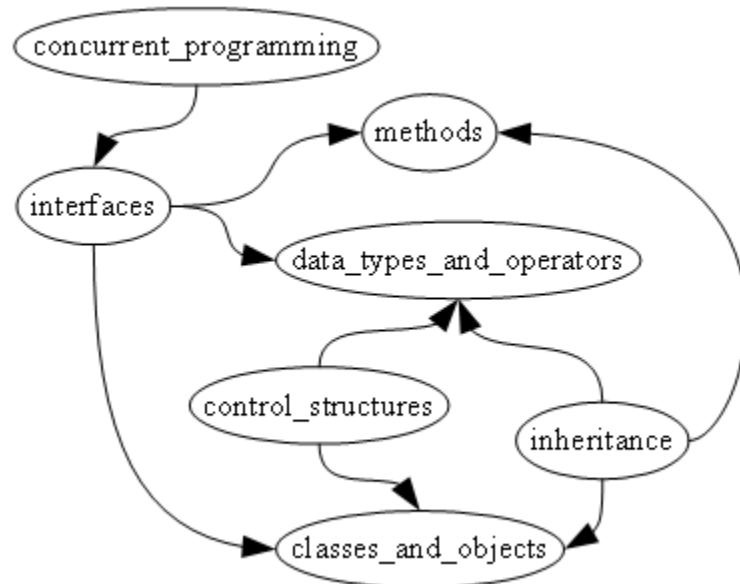


Figure I.3.1.1. An example of dependency graph

Figure I.3.1.2 (Henze, 2000, p. 53) expresses levels of $KI(s)$.

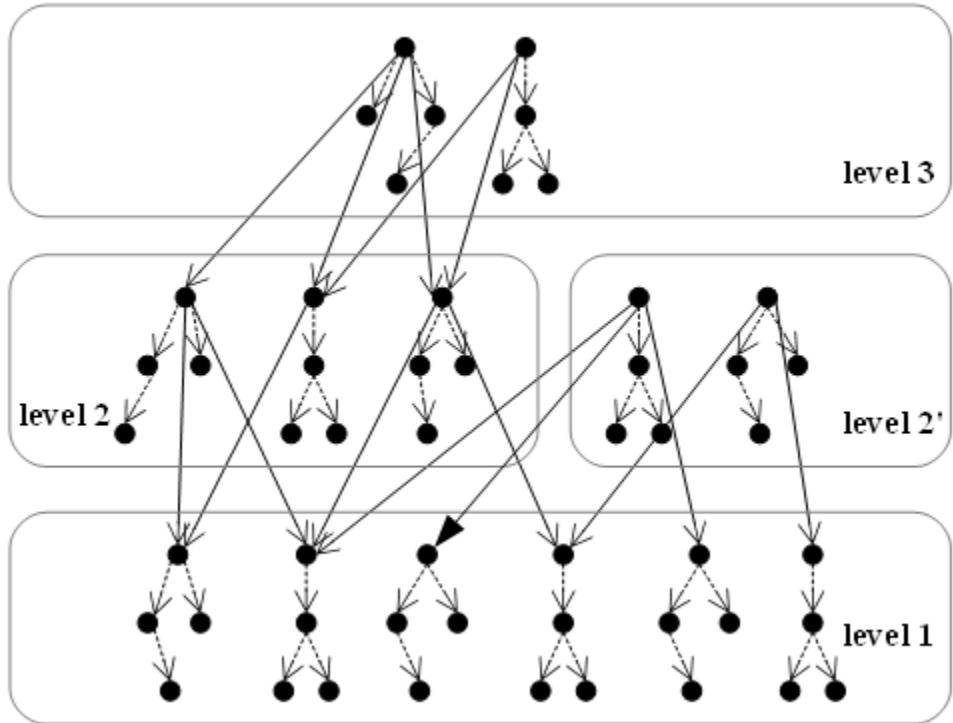


Figure I.3.1.2. The levels of KI (s)

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

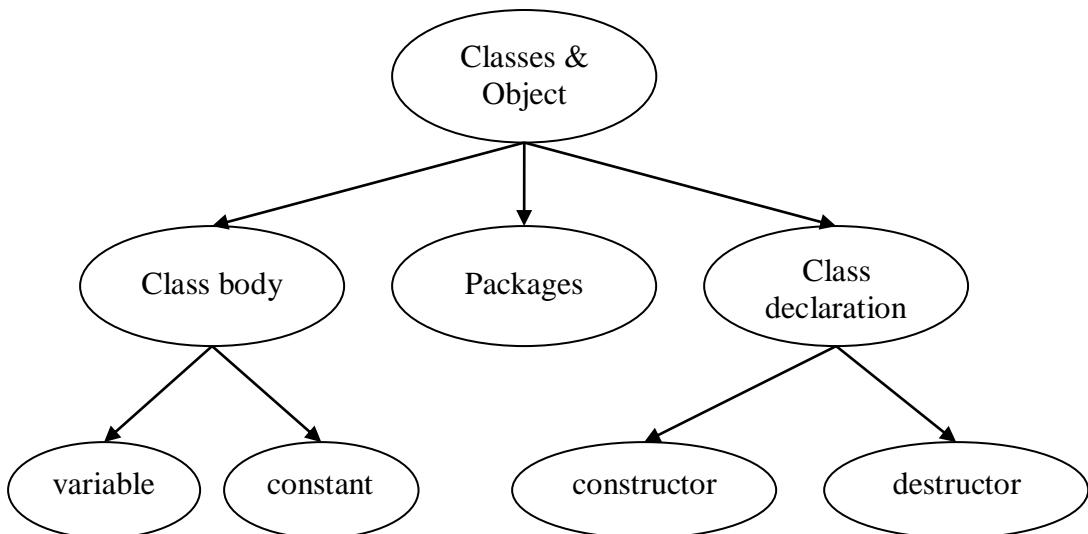


Figure I.3.1.3. Root tree

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

- E is an abbreviation of expert, which refers that user has expert's knowledge on a KI .

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

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

- *Clustering approach:* The nodes that cause directed cycle are clustered to single node, as seen in figure I.3.1.4 (Henze, 2000, p. 54).

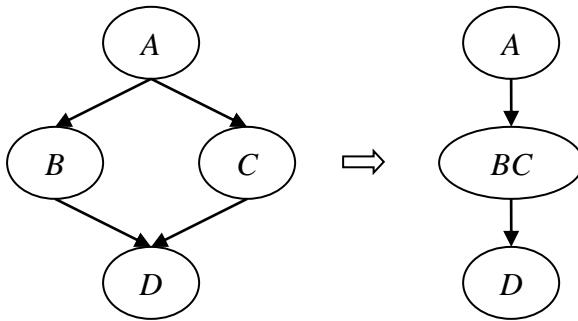


Figure I.3.1.4. Clustering approach

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

I.3.1.1. Yet Another Clustering Formalism (YACF)

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

The additional cluster node (Henze, 2000, p. 55) is the node that owns income nodes (called parent nodes) and outcome nodes (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.3.1.1.1 (Henze, 2000, p. 56) describes YACF method.

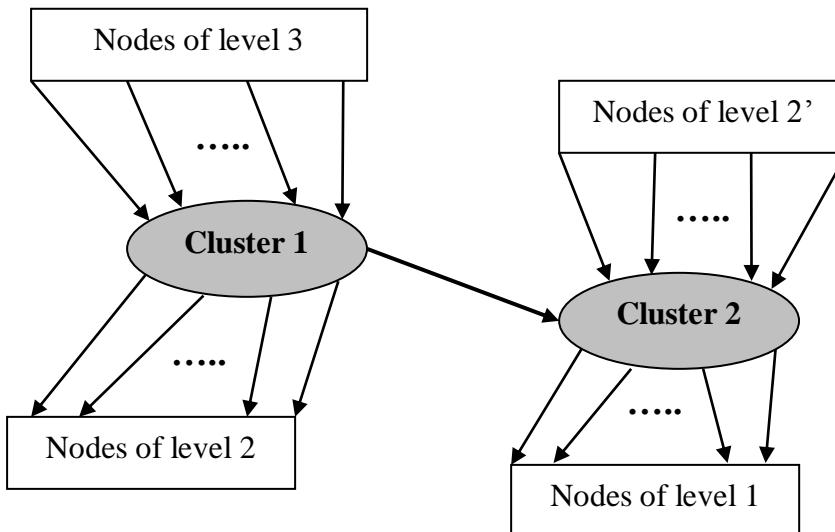


Figure I.3.1.1.1. YACF method

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.3.1.2.1) (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 .

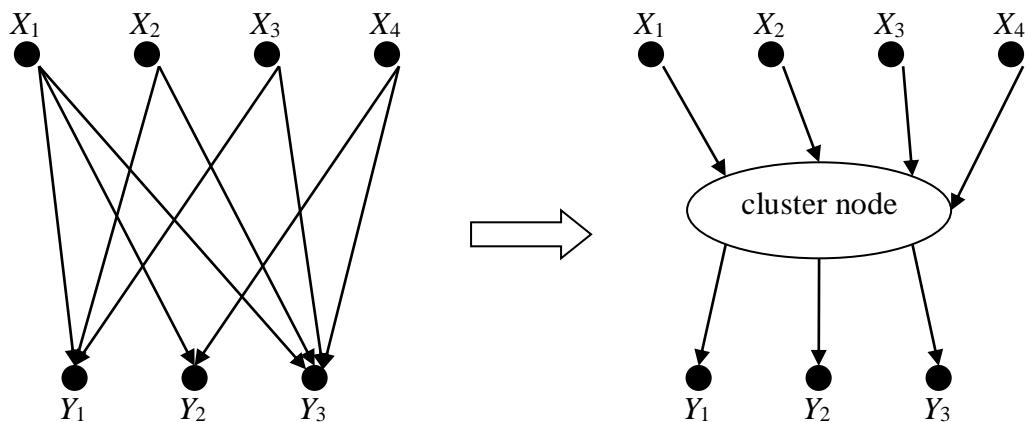


Figure I.3.1.2.1. Cluster node

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 (Henze, 2000, p. 56) proposed formula I.3.1.2.1 for calculating conditional probability of cluster node.

$$P(H = kY_i | X_1, \dots, X_N) = \begin{cases} \frac{1}{M}, & \text{if } k = \text{best_grade}(X_l \text{ (s) on which } Y_i \text{ is dependent}) \\ 0, & \text{else} \end{cases}$$

Formula I.3.1.2.1. Condition probability of cluster node

Where best_grade returns the maximum value of parent variables X_l (s) on which Y_i is dependent. Table I.3.1.2.1 (Henze, 2000, p. 57) shows the conditional probability table (CPT) of cluster node H , based on formula I.3.1.2.1.

$X_1, \dots, X_k, \dots, X_n$	$\text{range of } H \text{ holding evidence for node } Y_1$ $P(H=E_Y_1/\dots) \dots P(H=N_Y_1/\dots)$...	$\text{range of } H \text{ holding evidence for node } Y_M$ $P(H=E_Y_M/\dots) \dots P(H=N_Y_M/\dots)$
$X_1=E, \dots, X_k=E, \dots, X_N=E$. . .	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.3.1.2.1. 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.3.1.2.2. Conditional probability of child node Y_i

Table I.3.1.2.2 (Henze, 2000, p. 58) shows conditional probability of an arbitrary child node Y given cluster node H , based on formula I.3.1.2.2.

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

for Y				
$H_{i+1}=E$	0.25	0.25	0.25	0.25
.
$H_n=N$	0.25	0.25	0.25	0.25

Table I.3.1.2.2. 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 author Henze (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.3.1.2.3. 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.3.1.2.3 which is core of YACF method, the Bayesian network without cycles is totally determined.

I.3.2. Andes

Andes developed by authors Conati, Gertner, and Vanlehn (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 called *solution graph* which is then converted into Bayesian network. The solution graph is initialized right before student solves a problem. Figure I.3.2.1 depicts student modeling in intelligent tutoring system OLAE (Martin & VanLehn, 1995, p. 580) which is applied into Andes.

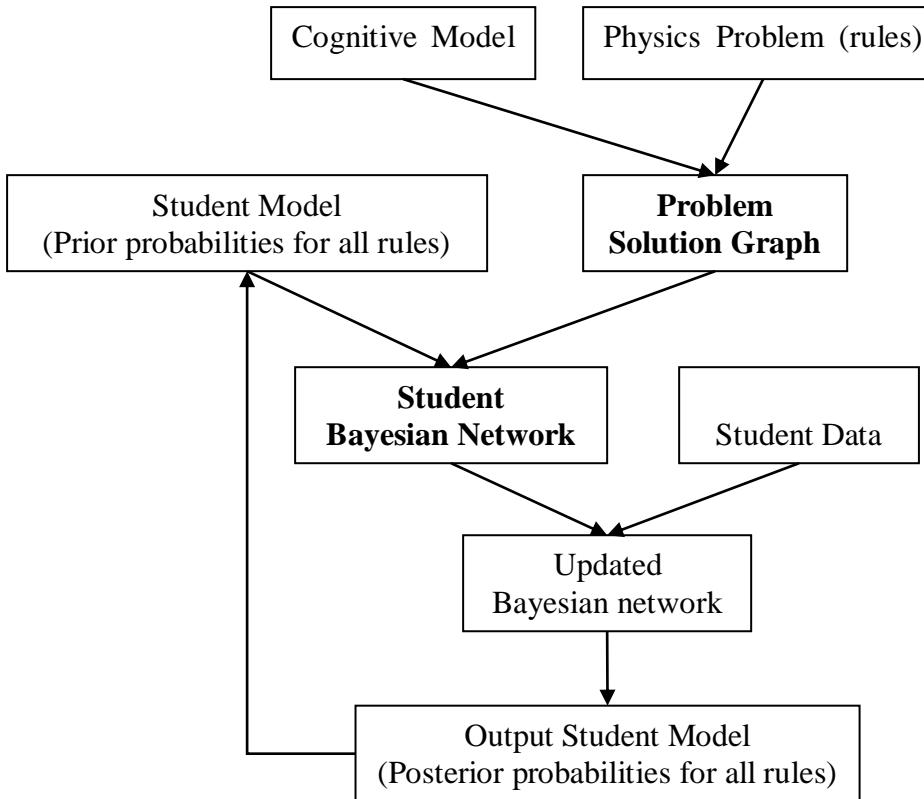


Figure I.3.2.1. Student modeling in Andes

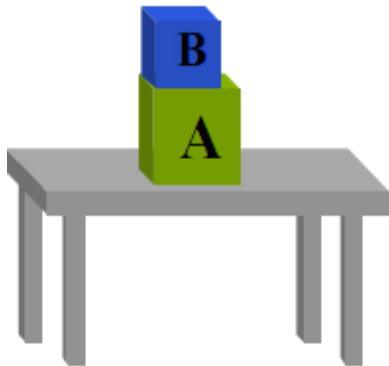
The description of Andes is mainly extracted from the article “Using Bayesian Networks to Manage Uncertainty in Student Modeling” by authors Conati, Gertner, and Vanlehn (Conati, Gertner, & Vanlehn, 2002). Readers are recommended to read this article for comprehending Andes.

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.3.2.1.1 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.3.2.1.1. Sample problem to find normal force

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

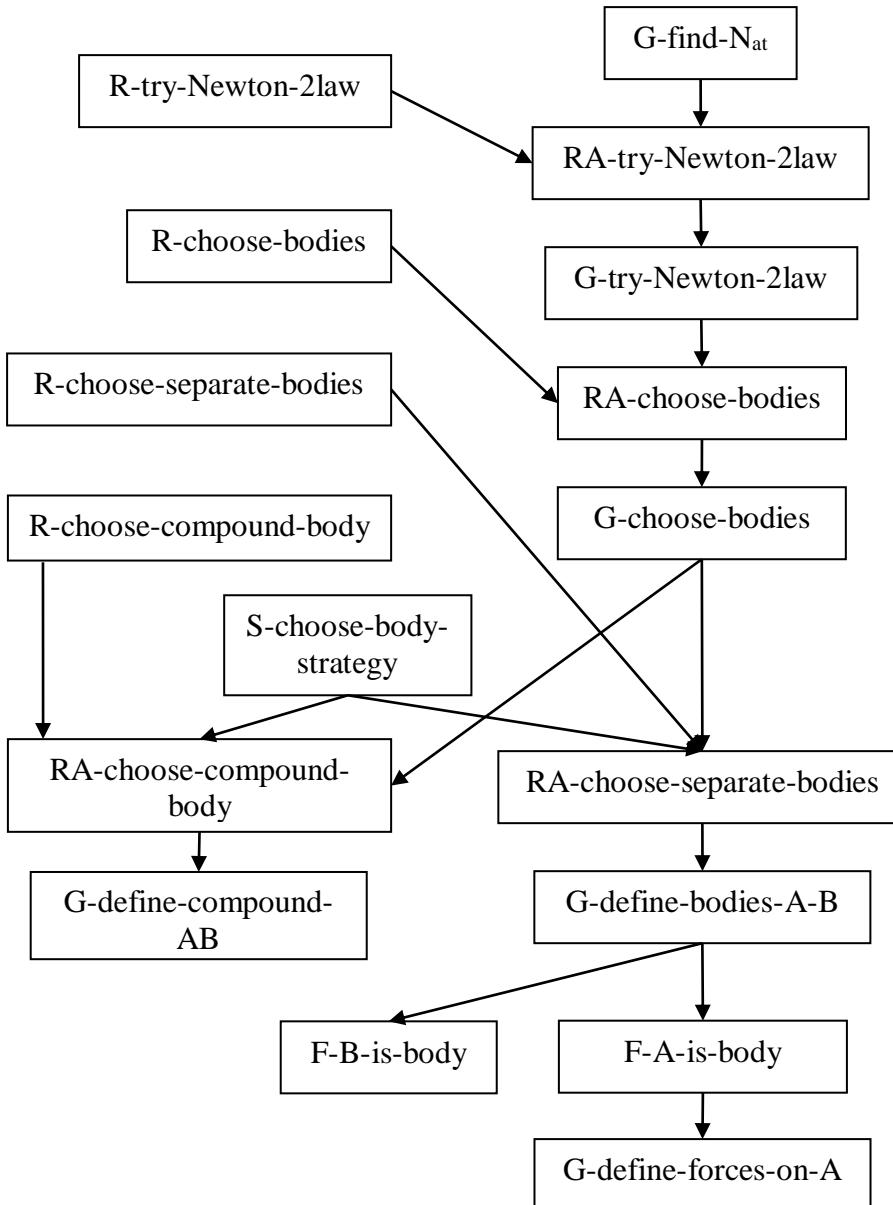


Figure I.3.2.1.2. Solution graph in Andes

Each node in the solution graph (Conati, Gertner, & Vanlehn, 2002, p. 11) denotes a particular type of information (goal, rule, rule application, strategy). For example, the nodes: *G-find-N_{at}*, *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 called *domain-general part* that encodes the domain-general knowledge and another part 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.3.2.2.1 (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.

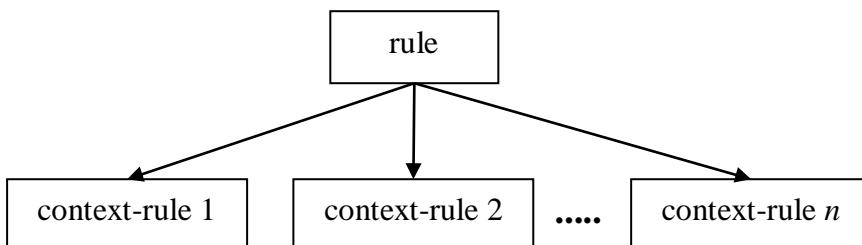


Figure I.3.2.2.1. Relationship between rule and context-rule nodes

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 β 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 α is called a “*noise*”. If at least one of its parents gets *false*, this conditional probability equals 0. It means that the student must master all context rules before she/he applies such rules into solving problem or exercise. In case that she/he even knows whole rules, it is possible to assert totally that she/he can apply perfectly rules. Figure I.3.2.2 shows relationship between nodes in task-specific part (Conati, Gertner, & Vanlehn, 2002, p. 14).

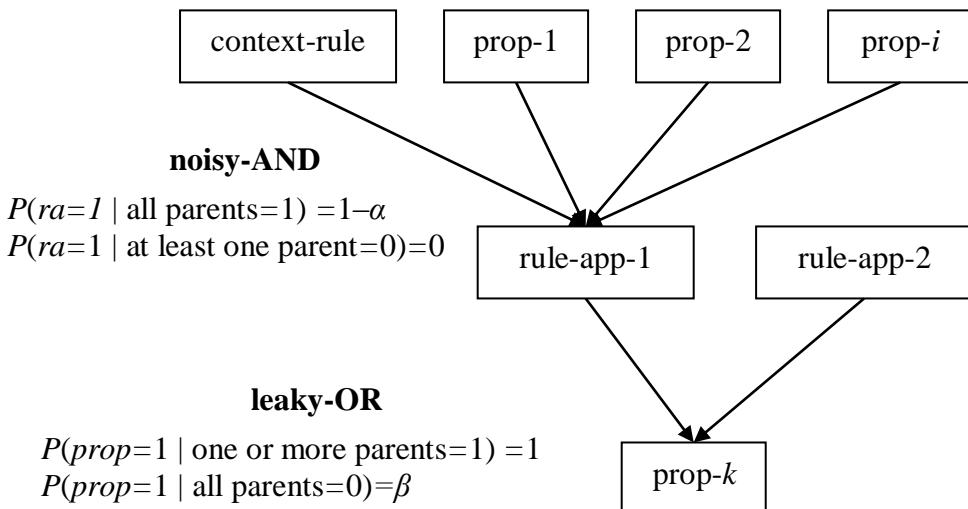


Figure I.3.2.2. Relationship between nodes in task-specific part

Strategy nodes (Conati, Gertner, & Vanlehn, 2002, p. 15) are used in situation that there are different solutions to a problem. For example, there are two application rules aiming to solve the same goal. When the posterior probability of one application rule lessens the posterior probability of another, it raises the issue called mutually exclusive strategy (Conati, Gertner, & Vanlehn, 2002, p. 15), as seen in figure I.3.2.3.

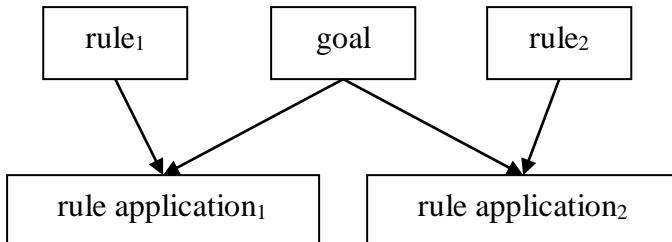


Figure I.3.2.2.3. Mutually exclusive strategy

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

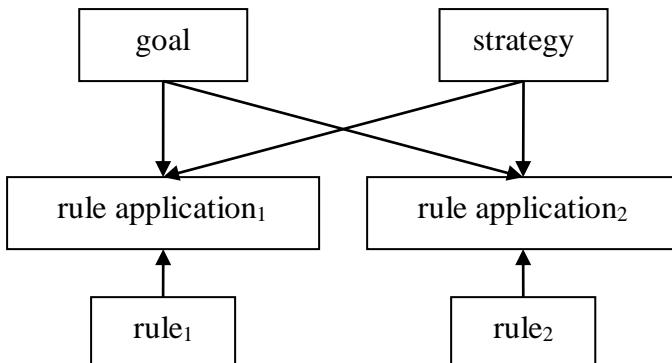


Figure I.3.2.2.4. Strategy node

Inference mechanism in Bayesian network

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

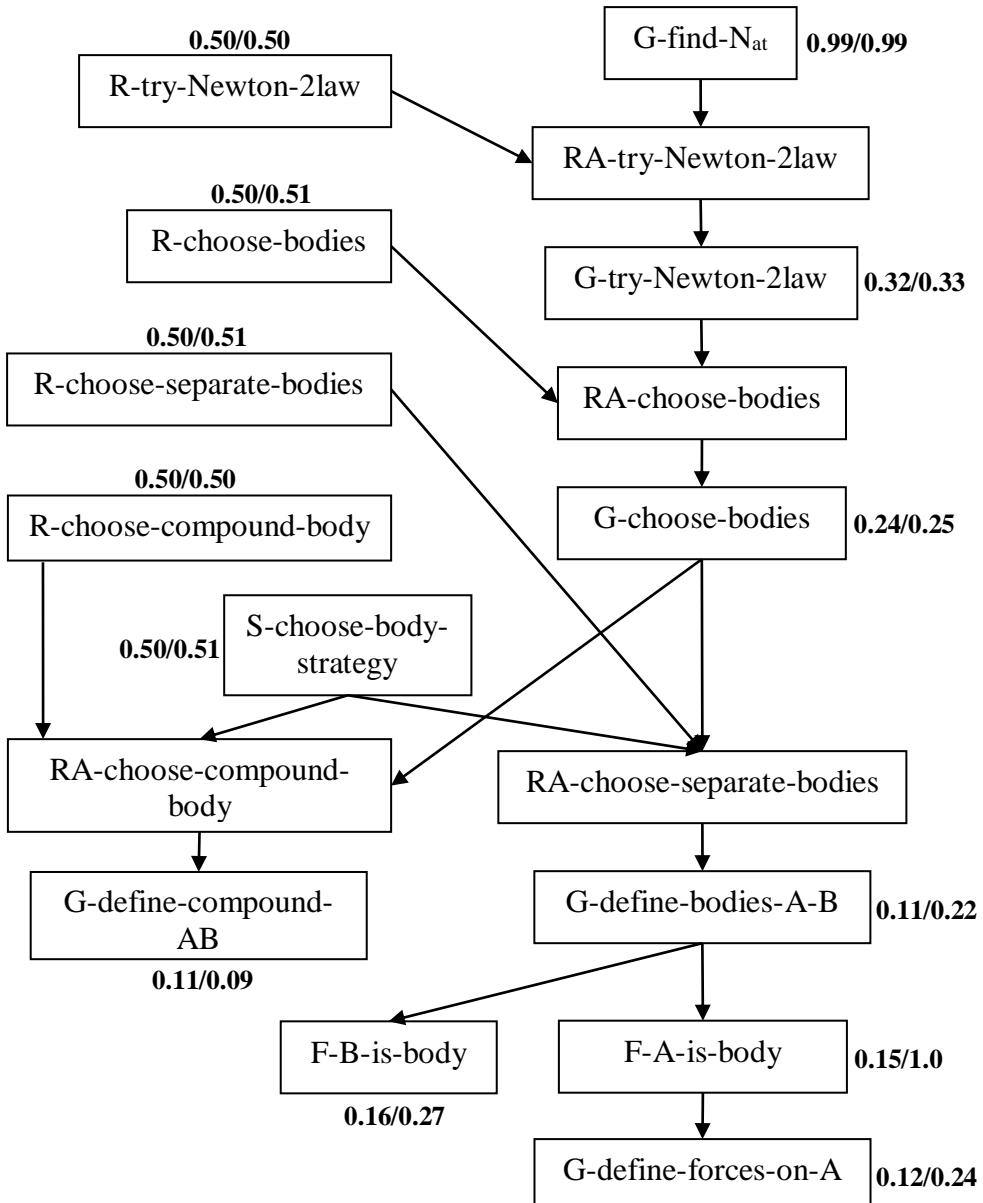


Figure I.3.2.2.5. Prior/Posterior probabilities in the task-specific part

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 Cr does not match this solution, the constraint is *violated*. Following is the matching process (Mayo, 2001, p. 71):

```

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

```

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

Architecture of SQL-Tutor

SQL-Tutor developed by author Mitrovic (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.3.3.1) has three functional models: CBM student modeler, pedagogical module and interface (Mitrovic, 1998, p. 309). As seen in figure I.3.3.1, SQL-Tutor is an Intelligent Tutoring System (ITS) aforementioned in sub-section I.2.2.

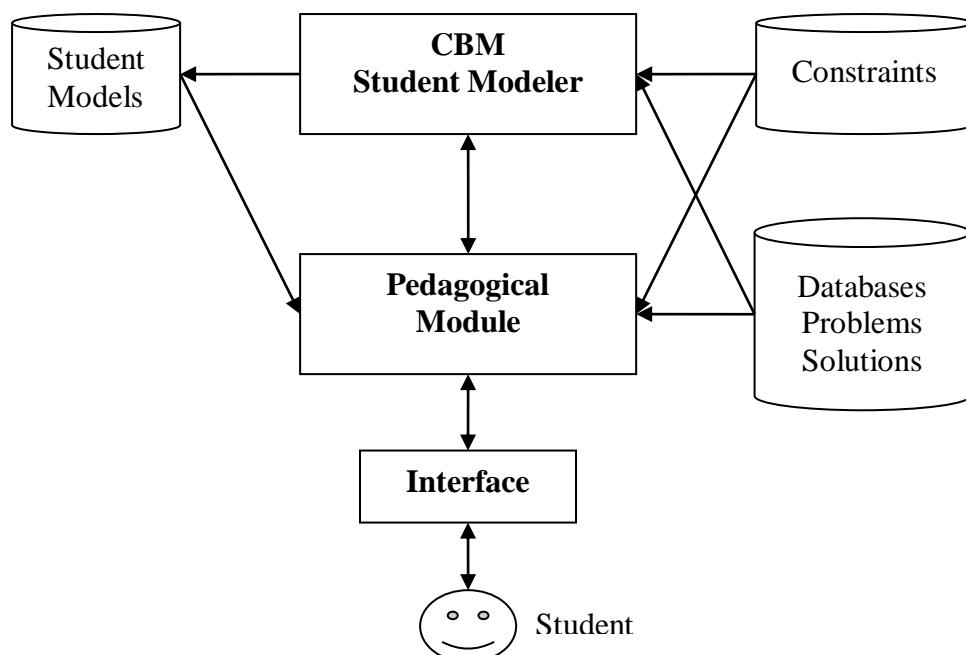


Figure I.3.3.1. Architecture of SQL-Tutor

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

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

Bayesian network in SQL-Tutor

SQL-Tutor is enhanced and extended by author Mayo (Mayo, 2001) in his PhD thesis. The author Mayo (Mayo, 2001, p. 93) added probabilistic student model into SQL-Tutor. So, please focus on how to represent student model by probabilistic approach instead of increasing or decreasing student's knowledge and how to apply Bayesian network into SQL-Tutor student model (Mayo, 2001, p. 93). The student model is constituted of a set of binary variables ($mastered_1, mastered_2, \dots, mastered_n$) 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.

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

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

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

Instead of using such rules, the author Mayo (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 (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 $mastered_c$ be the binary expressing whether student masters constraint c . The binary $relevantIS_{c,p} \in \{0, 1\}$ expresses whether constraint c is relevant to the ideal solution of problem p . The binary $relevantSS_{c,p} \in \{0, 1\}$ expresses whether constraint c is relevant to the student's solution of problem p . That variable $relevantSS_{c,p}$ depends on $relevantIS_{c,p} \in \{0, 1\}$ implicates that the student's solution must match the ideal solution. The variable $performance_{c,p}$ having three values *satisfied*, *violated* and *not-relevant* denotes the performance of student given the problem p on constraint c . The variable $performance_{c,p}$ depends on both $relevantSS_{c,p}$ and $mastered_c$. The Bayesian network representing these variables and relationships among them is shown in figure I.3.3.2 (Mayo, 2001, p. 98).

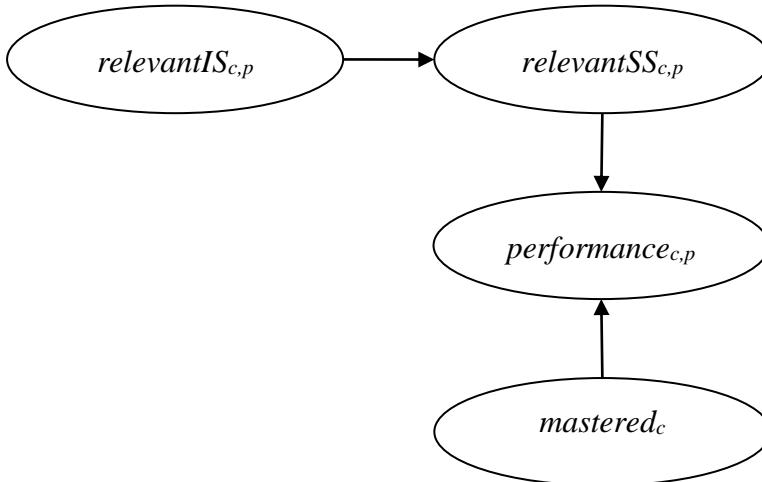


Figure I.3.3.2. Bayesian network in SQL-Tutor

As discussed, the prior probability of $mastered_c$, $P_0(mastered_c=1)$, is defined by Bayesian inference (Wikipedia, Bayesian inference, 2006) or heuristic rule. The prior probability of $relevantIS_{c,p}$ is specified by expert. According to author Mayo (Mayo, 2001, p. 98), the conditional probability table (CPT) of $relevantSS_{c,p}$ given $relevantIS_{c,p}$ is defined as in table I.3.3.1.

		$relevantIS_{c,p}$	
		yes (1)	no (0)
$relevantSS_{c,p}$	yes (1)	α_c	β_c
	no (0)	$1-\alpha_c$	$1-\beta_c$

Table I.3.3.1. Conditional probability table of $relevantSS_{c,p}$ given $relevantIS_{c,p}$

According to author Mayo (Mayo, 2001, p. 98), the parameter α_c (β_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 α_c , β_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 α_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 β_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, and Liu (Cheng, Bell, & Liu, 1997) proposed the considerable method for learning Bayesian network structure from training data. In this method, the correlation between two nodes is measured by the amount of information flow between them. Such measurement is called mutual information (Mutual information, 2014). The higher the mutual information of two nodes is, the more the correlation between them is, in other words, the more likely there is an arc connecting them. The mutual information of two nodes X and Y is defined as below (Cheng, Bell, & Liu, 1997, p. 2).

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

Note that notation $\log(\cdot)$ denotes 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 ζ , if the conditional mutual information $I(X, Y|C)$ is smaller than ζ then two nodes X, Y are d-separated by set C .

The authors Cheng, Bell, and Liu (Cheng, Bell, & Liu, 1997, pp. 2-3) proposed an algorithm for learning the structure of Bayesian network which includes three phases: *drafting*, *thickening* and *thinning*. However, before discussing about this algorithm, it is necessary to know the concept “d-separation” and “cut-set”. Give a set C and two nodes (A, B), the statements “ A is d-separated from B by C ”, “ C d-separates A from B ” or “there is a d-separation between A and B given C ” mean that there is no active

(open) undirected path between A and B . The path between A and B is active if every node in the path having head-to-head arrows (like $X \rightarrow Z \leftarrow Y$) 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 ζ , for each pair of nodes X and Y , the mutual information $I(X, Y)$ is computed by above formula. All of these pairs whose $I(X, Y)$ is larger than ζ are sorted into the set R according to their respectively $I(X, Y)$ in descending order. Starting with picking up the first pair whose $I(X, Y)$ is largest from S ; if there is no undirected path between X and Y (these two nodes are d-separated given empty set) then an undirected arc is added between X and Y . This is repeated until S contain only pairs that aren’t adjacent but are connected via a longer path. The output of this phase is the single-connected network or some unconnected single-connected networks. It means that maybe there is lack of some arcs in networks.

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

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

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

The output of *thinning phase* is the final structure of Bayesian network.

Now this chapter I 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 works, which is to propose a learner model and a user modeling system that manipulates such learner model (see section II.2). Chapters III, IV, and V will describe my works in the most detailed.

Chapter II. 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 L. , 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

Recall that 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. The main content of this section II.1 is mainly extracted from the master thesis “*User Modeling and User Profiling in Adaptive E-learning Systems*” of the author Christoph Fröschl (Fröschl, 2005) and the article “*Generic User Modeling Systems*” of the author Kobsa (Kobsa, 2007). I express my deep gratitude to the authors Fröschl and Kobsa for providing their great researches.

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 (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 uses 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 and Drager (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.1.2.1) (Finin & Drager, 1986, p. 225).

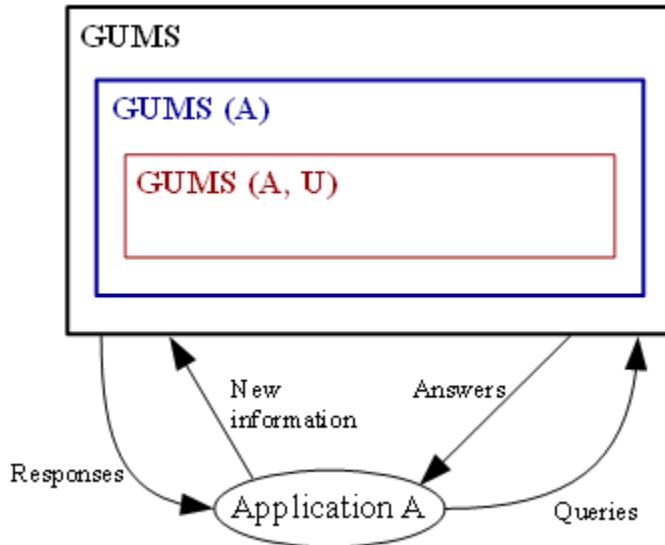


Figure II.1.2.1. The architecture of GUMS

Where A, GUMS(A) and GUMS(A, U) denote 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 and Tasso (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). At the first time user interacts with system, trigger is activated to assign her/his to 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 (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 been 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 and Self (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 (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 from 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, CUMULATE, Personis.

BGP-MS developed by authors Kobsa and Pohl (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.1.3.1 (Fröschl, 2005, p. 64).

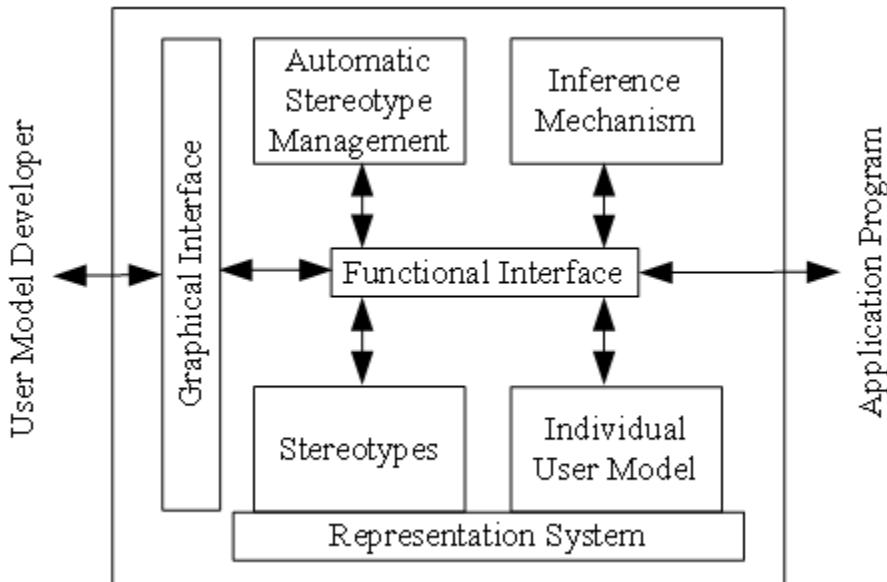


Figure II.1.3.1. The architecture of BGP-MS

As seen in figure II.1.3.1, the adaptive system communicates with main components through functional interface. 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 (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 (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).

CUMULATE developed by authors Brusilovsky, Sosnovsky, and Shcherbinina (Brusilovsky, Sosnovsky, & Shcherbinina, 2005) is a generic student-modeling server in which information about user is represented on two levels: *event storage* and *inference 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. CUMULATE 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 CUMULATE is open to a variety of *external inference agents* that receive requests from event storage and manipulate user model. CUMULATE is really a combination of event-driven approach and agent-based approach, which is prominent in distributed e-learning environment. Figure II.1.3.2 shows its architecture (Brusilovsky, Sosnovsky, & Shcherbinina, 2005, p. 3) (Brusilovsky, 2004, p. 5).

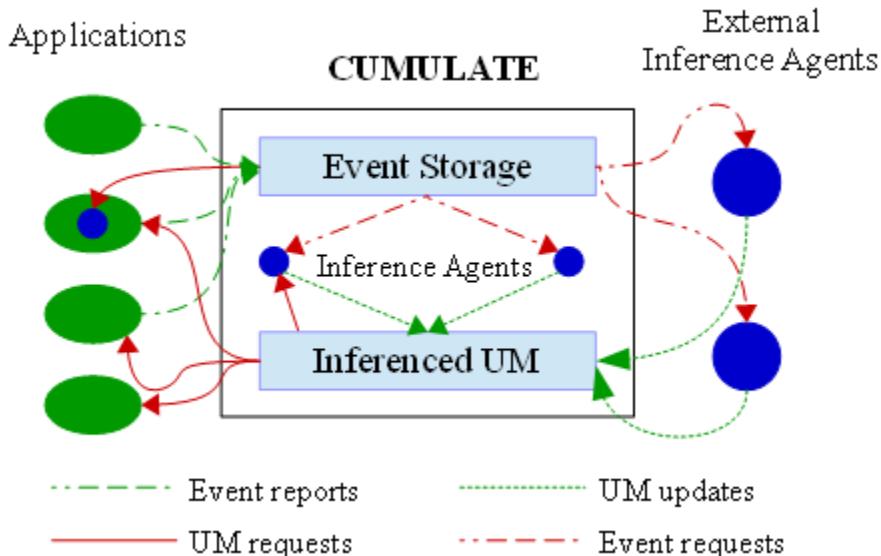


Figure II.1.3.2. The architecture of CUMULATE

Personis developed by authors Kay, Kummerfeld, and Lauder (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 with 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.1.3.3 shows the high-level architecture of Personis (Kay, Kummerfeld, & Lauder, 2002, p. 205).

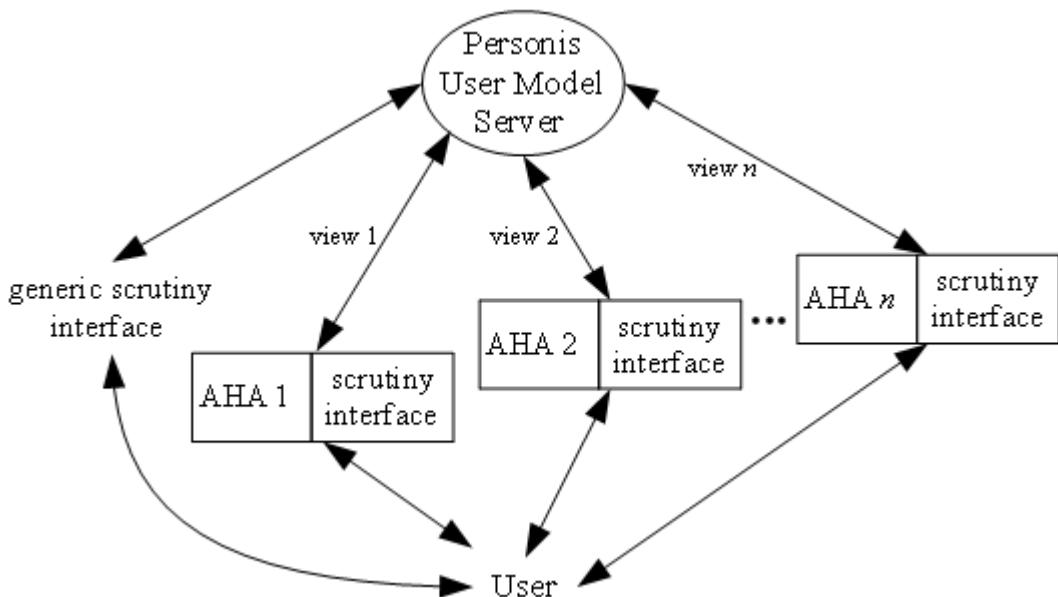


Figure II.1.3.3. The architecture of Personis

LDAP-UMS is the user modeling servers based on Lightweight Directory Access Protocol (LDAP), published by author Fink (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.1.3.4 shows the architecture of LDAP-UMS (Fink, 2004, p. 75) (Fröschl, 2005, p. 71).

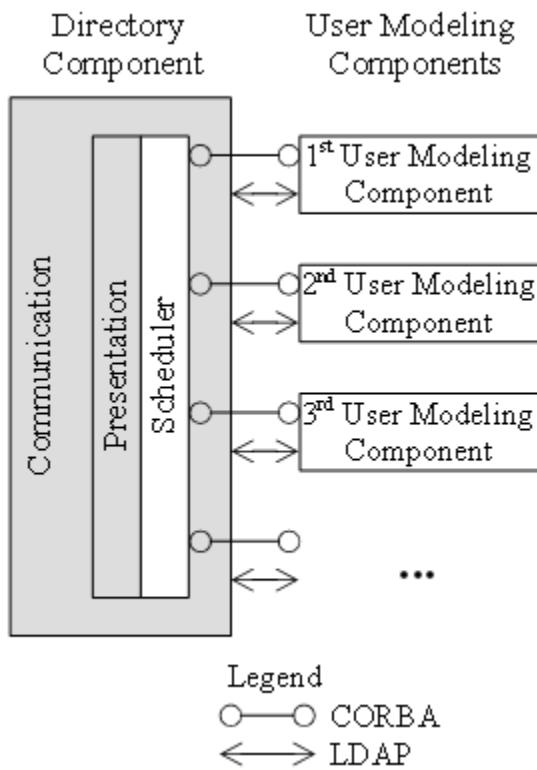


Figure II.1.3.4. The architecture of LDAP-UMS

Readers are recommended to read the PhD thesis “User Modeling Servers – Requirements, Design, and Evaluation” of the author Fink (Fink, 2004) in order to understand comprehensively the powerful system LDAP-UMS.

Now existing user modeling systems were introduced briefly in this section II.1 and most of them aim 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 like e-learning 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 is 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, most 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 architecture of Zebra.
- Sub-section [II.2.3](#) tells us interaction between Zebra and adaptive learning systems like AHS and AEHS.
- Because Zebra is implemented as computer software, sub-section [II.2.4](#) focuses on 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.2.1.1). Hence TLM has three sub-models: *knowledge sub-model*, *learning style sub-model* and *learning history sub-model*.

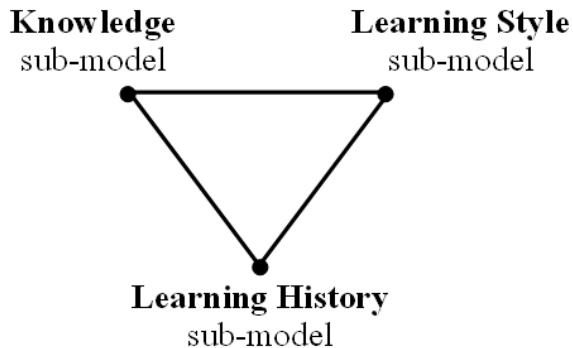


Figure II.2.1.1. Triangular Learner Model

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) is the directed acyclic graph (DAG) in which nodes are linked together by arcs; each arc expresses the relationship between two node. The strength of relationship is 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 chapter III. The proposed approach that combines overlay model and Bayesian network so as to build up Bayesian overlay sub-model is described particularly in section III.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 section IV.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 by 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 chapter IV.

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.2.1.2 shows the extended TLM whose learning history sub-model is the root to which more learners' characteristics such as interests, backgrounds, and goals to TLM are attached.

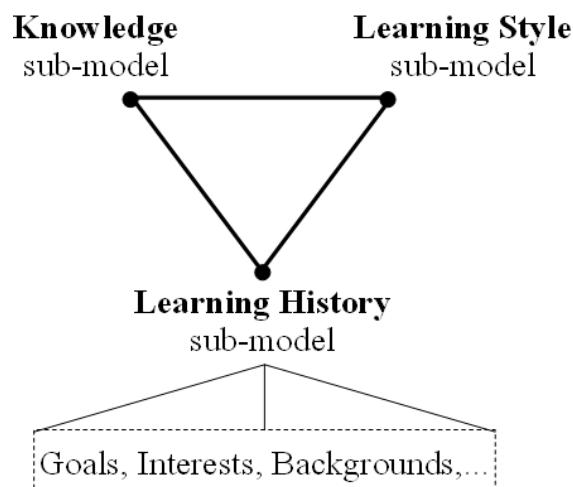


Figure II.2.1.2. extended Triangular Learner Model

Learning history sub-model is described particularly in chapter [IV](#).

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.2.2.1](#) indicates that user modeling (Brusilovsky, 1996, p. 2) is similar to profile mining.

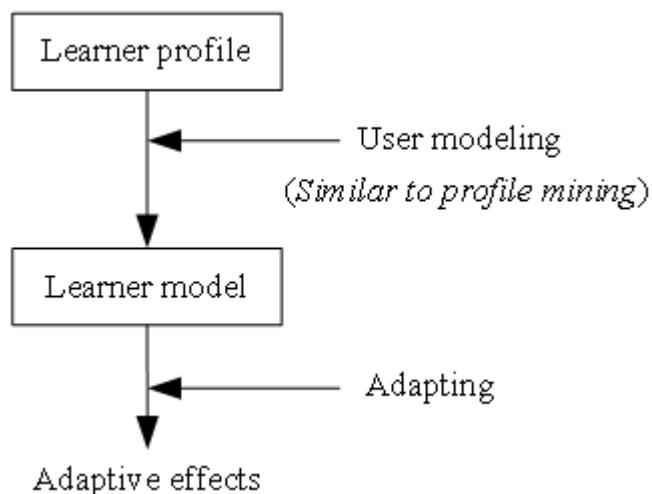


Figure II.2.2.1. Modeling task is similar to profile mining task

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.

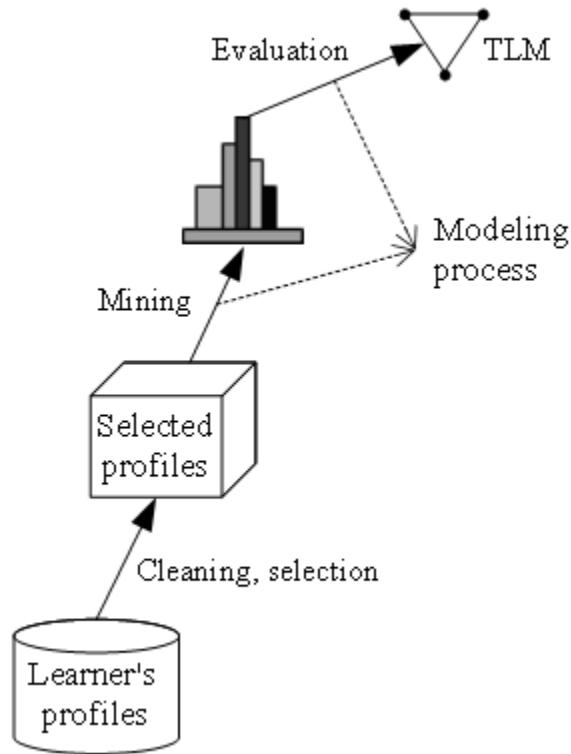


Figure II.2.2.2. The modeling process in Zebra

As seen in figure II.2.2.2 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 TLM. In next chapters III and IV, 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) and Markov model (see section IV.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 model 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 one which supports collaborative learning based on user group construction is the advanced functionality of Zebra. Please see chapter

[V](#) for more details about ME, learning history sub-model, and these functionalities.

- **Belief network engine (BNE)** is responsible for inferring new user information from TLM by using deduction mechanism available in belief network. New information mentions users' new knowledge and learning styles. 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. Please see chapters [III](#) and [IV](#) for more details about BNE, knowledge sub-model, learning style sub-model.

Zebra also provides *communication interfaces (CI)* that allow users and adaptive systems to see or modify restrictedly their TLM. Adaptive applications also interact with Zebra by these interfaces. CI (s) 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 called *observer* having responsibility for tracking learners' actions. Observer catches and delivers user observations to Zebra. Observer interacts with Zebra through CI.

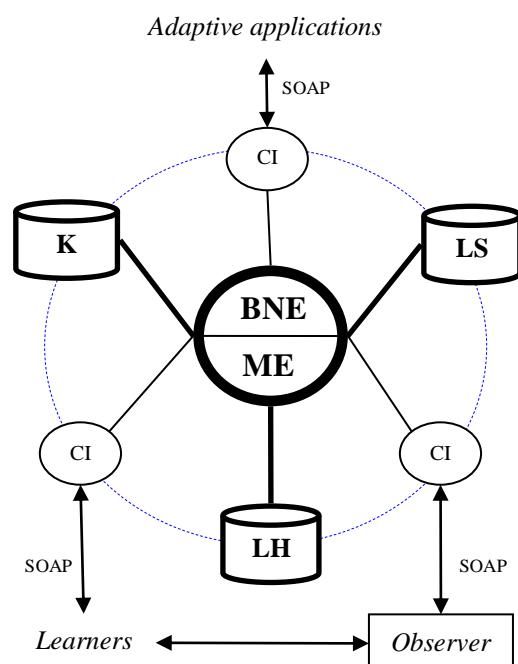


Figure II.2.2.3. The architecture of Zebra

Figure II.2.2.3 depicts the architecture of Zebra in which ME and BNE denote mining engine and belief network engine, respectively. 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 <http://zebra.locnguyen.net>. 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 is associated with descriptive information (metadata).
- *Domain model* constitutes the structure of domain knowledge, which was often represented in form of graph. Knowledge items stored in domain model are concepts, subjects, topics, etc.
- *Adaptation model* is the centric component which gives effect to adaptation. It contains *concept selection rules* and *content selection rules*. Concept selection rules are used to choose appropriate concepts from domain model. On the other hand, we apply content selection rules into choosing suitable educational resources from media space. These rules must be in accordance with user model so that the selection gets correct.
- *User model* represents 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.2.3.1 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. In the next sub-section II.2.4, the adaptive learning system **WOW** based on **AHA!** system is introduced as a sample AEHS.

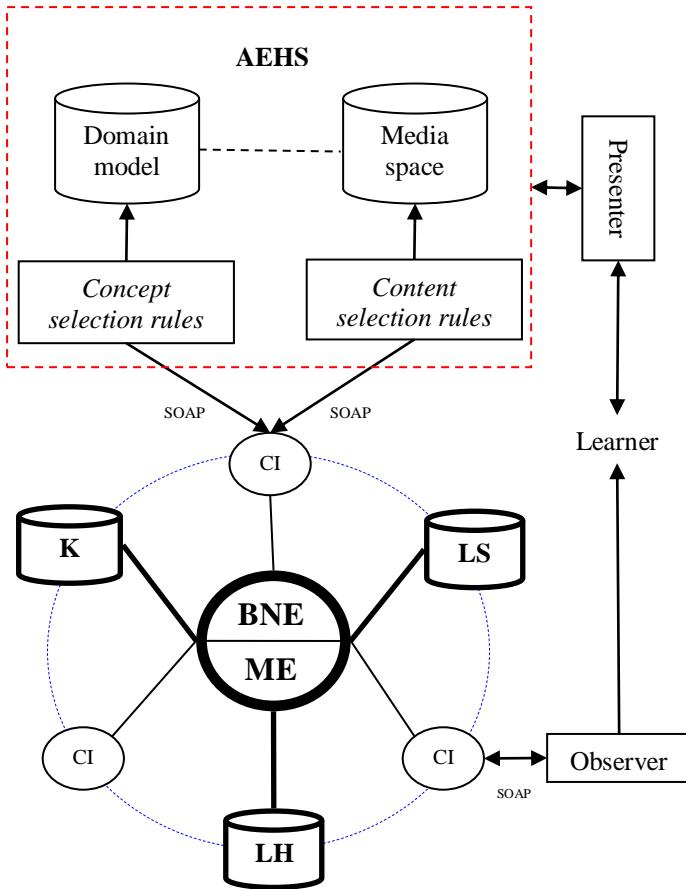


Figure II.2.3.1. The new architecture of AEHS and the interaction between AEHS and Zebra

In figure II.2.3.1, *observer* is external program that tracks learners' actions and *presenter* is any application or web site that introduces or presents learning resources to learners. Given the adaptive learning system WOW mentioned in successive subsection II.2.4, the sample presenter is e-learning web site of WOW shown in figure II.2.4.18. When WOW is associated with Zebra, the observer is itself, WOW.

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 learner's knowledge to concepts in domain model. In other words, these concepts are filtered to find ones that are necessary for learner to study in her/his course. Knowledge sub-model supports the concept selection process.
- In *content selection process*, learning resources are selected from resource model based on content selection rules that match learner's learning styles to attributes of resources in resource space. In other words, this process finds the adaptive resources that are suitable to learner. It is most likely that learner prefers such adaptive resources. Learning style sub-model and learning history sub-model support the content selection process.

The adaptation process shown in figure II.2.3.2 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 items 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 relates 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 learner in order to catch and deliver learner's observations to Zebra. Zebra uses such observations to update TLM.

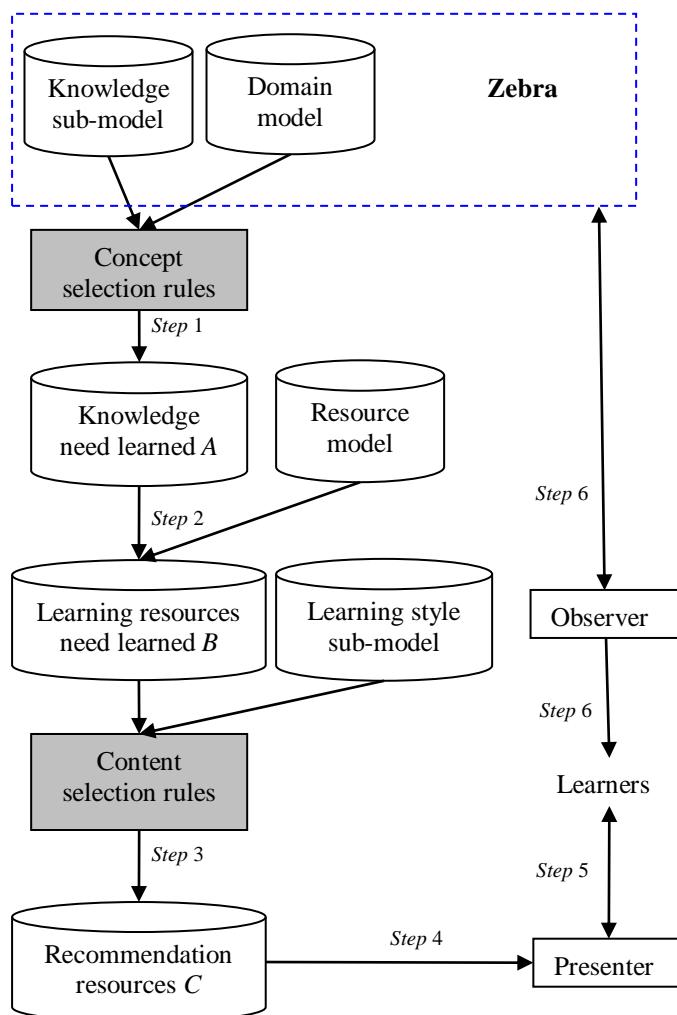


Figure II.2.3.2. Steps in adaptation process with support of Zebra

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

II.2.4. Implementation of Zebra

Zebra is implemented as a computer software working as a server (Nguyen L. , ZEBRA: A new User Modeling System for Triangular Model of Learners' Characteristics, 2009) (Nguyen L. , A User Modeling System for Adaptive Learning -

Abstract, 2014). Zebra server is written by Java language (<https://www.oracle.com/java>) which is an object-oriented programming language supported by (Oracle). 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. Table II.2.4.1 shows relationships of two engines (including BNE and ME), TLM (including knowledge sub-model, learning style sub-model, learning history sub-model), and 9 daemons.

Engine	TLM	Daemon
BNE	Knowledge sub-model	Knowledge static Bayesian network daemon
		Knowledge dynamic Bayesian network daemon
	Learning style sub-model	Learning style daemon
ME	Learning history sub-model	Learning history data daemon
		Learning concept recommendation daemon
		Learning path daemon
		Discovering user interests daemon (document classification + user searching)
		Constructing user communities daemon
		Mailing list daemon

Table II.2.4.1. Relationships of two engines, TLM, and 9 daemons

According to table II.2.4.1, two daemons: knowledge static Bayesian network daemon and knowledge dynamic Bayesian network daemon belong to knowledge sub-model which, in turn, is managed by BNE. Learning style sub-model belongs to learning style sub-model which, in turn, is managed by BNE. Five daemons: learning history data daemon, learning concept recommendation daemon, learning path daemon, discovering user interests daemon, and constructing user communities daemon belong to learning history sub-model which, in turn, is managed by ME. The mailing list daemon is utility daemon and so it does not belong to any sub-model. Table II.2.4.2 summarizes responsibilities of such 9 daemons. Of course, these responsibilities also belong to respective TLM sub-models.

<i>No</i>	<i>Daemon</i>	<i>Responsibility</i>
1	Knowledge static Bayesian network daemon	<ul style="list-style-type: none"> - Building up Bayesian model by combination of overlay model and Bayesian network; please see section III.1 for more details. - Evaluating user knowledge (see figure II.2.4.11), based on inference mechanism inside Bayesian

		network. Please see section III.2 for more details.
2	Knowledge dynamic Bayesian network daemon	Building dynamic Bayesian network to model user knowledge in chronological process; please see section III.4 for more details about dynamic Bayesian network.
3	Learning style daemon	<ul style="list-style-type: none"> - Applying hidden Markov model into building up learning style sub-model; please see section IV.6 for more details. - Inferring user's learning styles from observations of her/his study; please see section IV.6 for more details.
4	Learning history data daemon	This daemon inherits all excellent aspects of log file management function of the generic adaptive system AHA! (De Bra & Calvi, 1998) in order to organize, store, and manage learning history data (learning materials access, duration of computer use, doing exercise, taking an examination, doing test, communicating with teachers or classmates, etc.) as structured data like XML files and relation database. Please see sub-section I.2.3.3 for more details about AHA! system.
5	Learning concept recommendation daemon	This daemon mines learning history data to extract sequential patterns. After that, it breaks such patterns into concepts/learning materials which are recommended to users. This responsibility is the first extended function of learning history sub-model. Please see section V.1 for more details about learning concept recommendation based on mining learning history.
6	Learning path daemon	This daemon take advantage of sequential patterns resulted from learning concept recommendation daemon in order to recommend user optimal learning path (learning route) so that she/he is successful in her/his studying process.
7	Discovering user interests daemon (document classification + user searching)	<ul style="list-style-type: none"> - This daemon discovers user interest based on document classification. 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. This responsibility is the second extended function of learning history sub-model. Please see section V.2 for more details about discovering user interests by document classification. - This daemon also supports users to search documents relevant to their course.
8	Constructing user communities daemon	This daemon is responsible for constructing user groups or user communities, which supports group adaptation and collaborative learning. This responsibility is the third extended function of learning history sub-model. Please see section V.3 for more details about how to construct user communities.
9	Mailing list daemon	This daemon is responsible for managing mailing list of users and sending e-mails to users (see figure II.2.4.15).

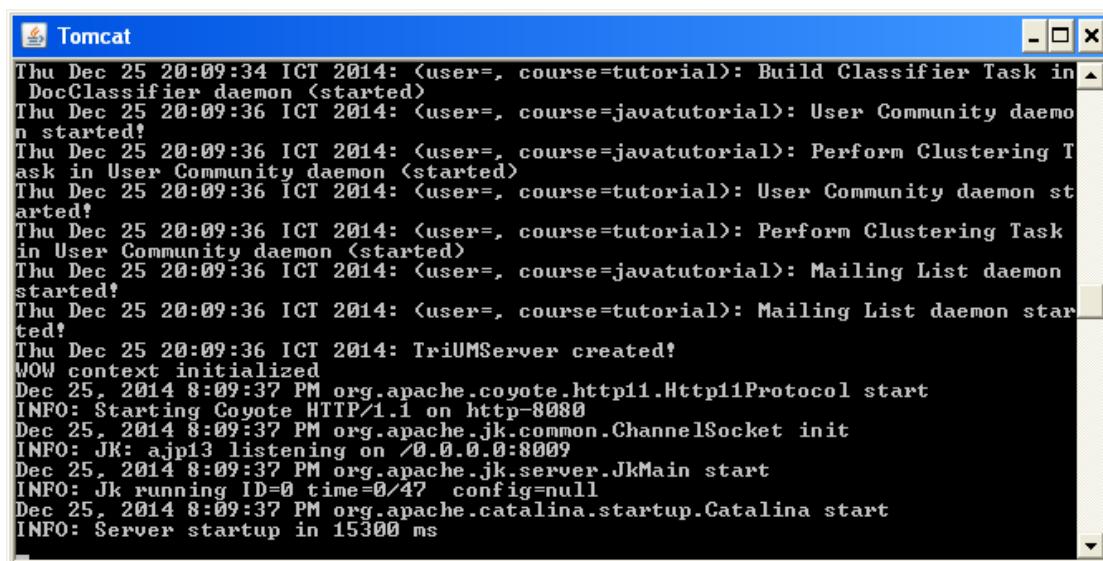
Table II.2.4.2. Responsibilities of 9 daemons

These responsibilities will be mentioned below. Zebra server, TLM, BNE, and ME have synonyms: TriUMServer, TriUM, ZebraNetworker, and ZebraMiner, respectively when you look up source code of Zebra implementation. Communication interface (CI) is implemented as so-called *Delegator* which dispatches users/applications requests to Zebra server and sends back server responses to users/applications.

Besides implementing Zebra architecture and TLM, Zebra server provides many utility tools for managing learners and supporting adaptive learning. Zebra server is available at internet link as follows:

<http://zebra.locnguyen.net>

When you startup Zebra server, it runs as underlying service as shown in figure II.2.4.1. 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.2.4.1. Zebra is running

Zebra server is often manipulated via *Zebra control panel* shown in figure II.2.4.2 (Nguyen L. , A User Modeling System for Adaptive Learning - Demonstration, 2014).

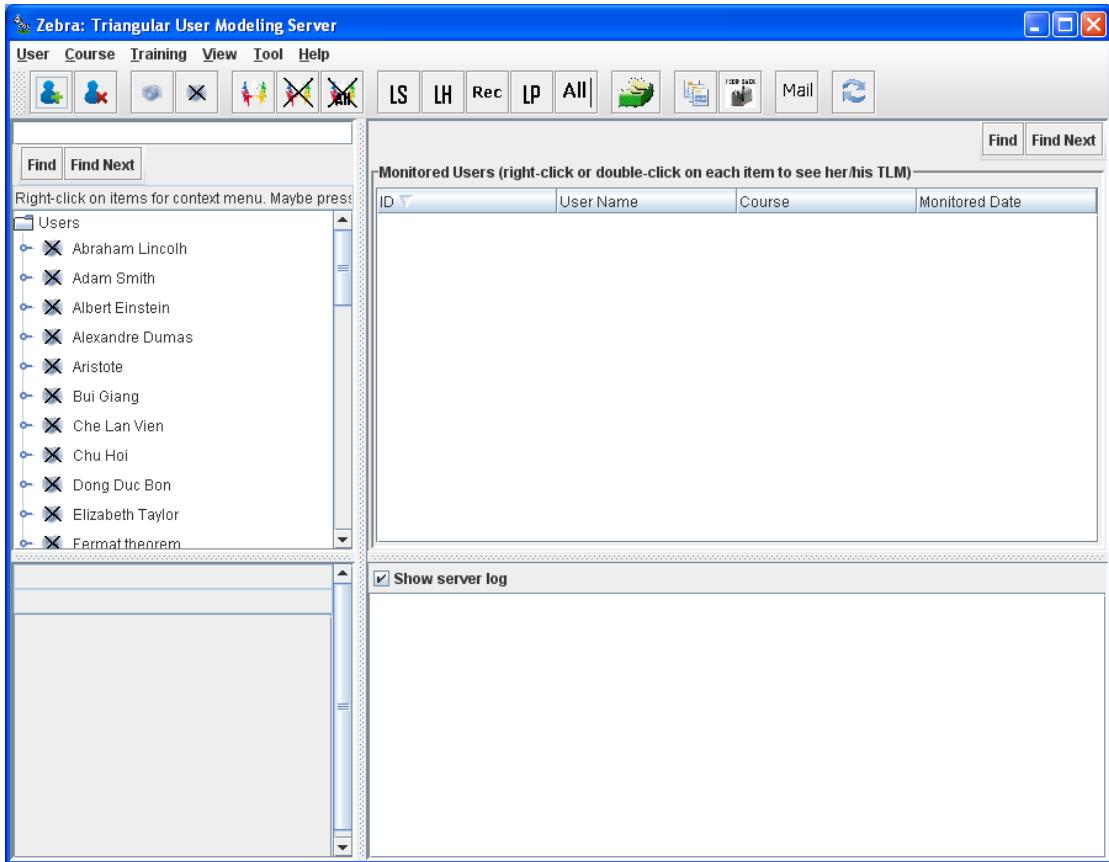


Figure II.2.4.2. Zebra control panel

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.2.4.3. 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 section III.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 identified by “*john*” will have two TLM (s) such as “*john\$javatutorial*” and “*john\$oracletutorial*”. It is concluded that each TLM always associates with a course.

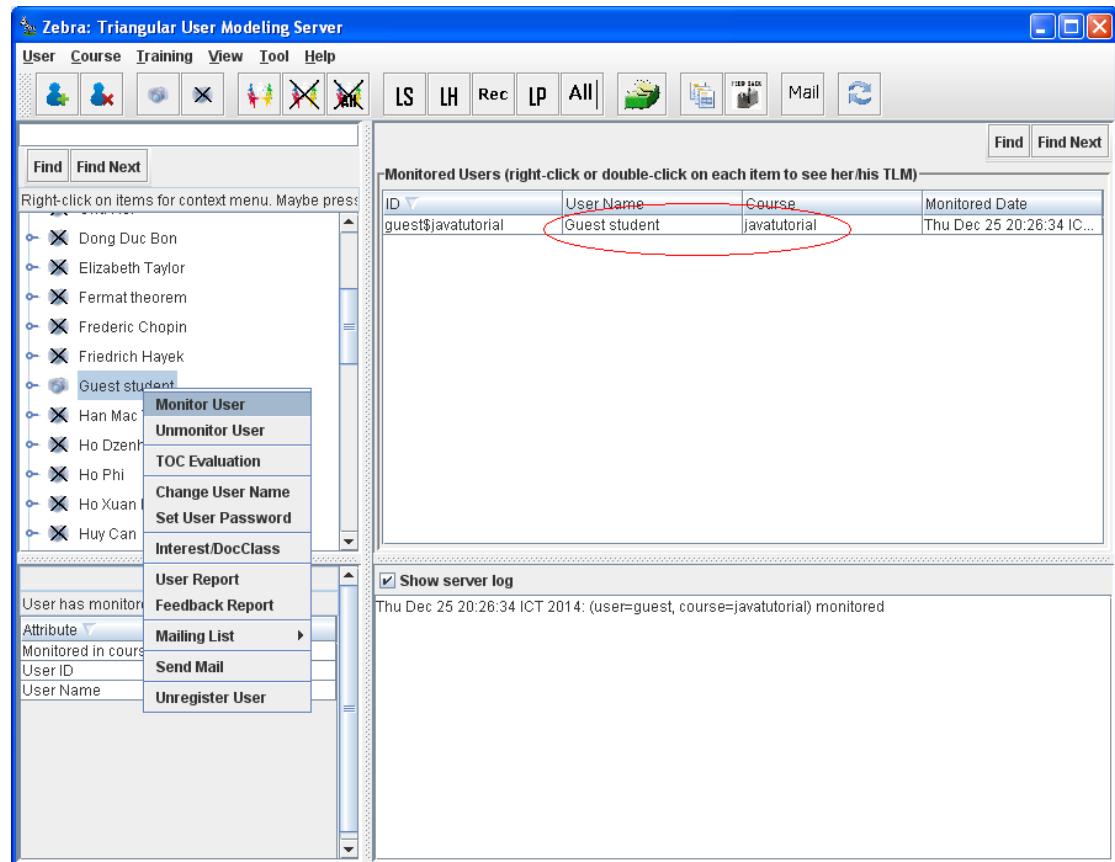


Figure II.2.4.3. Monitoring a learner

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.2.4.4.

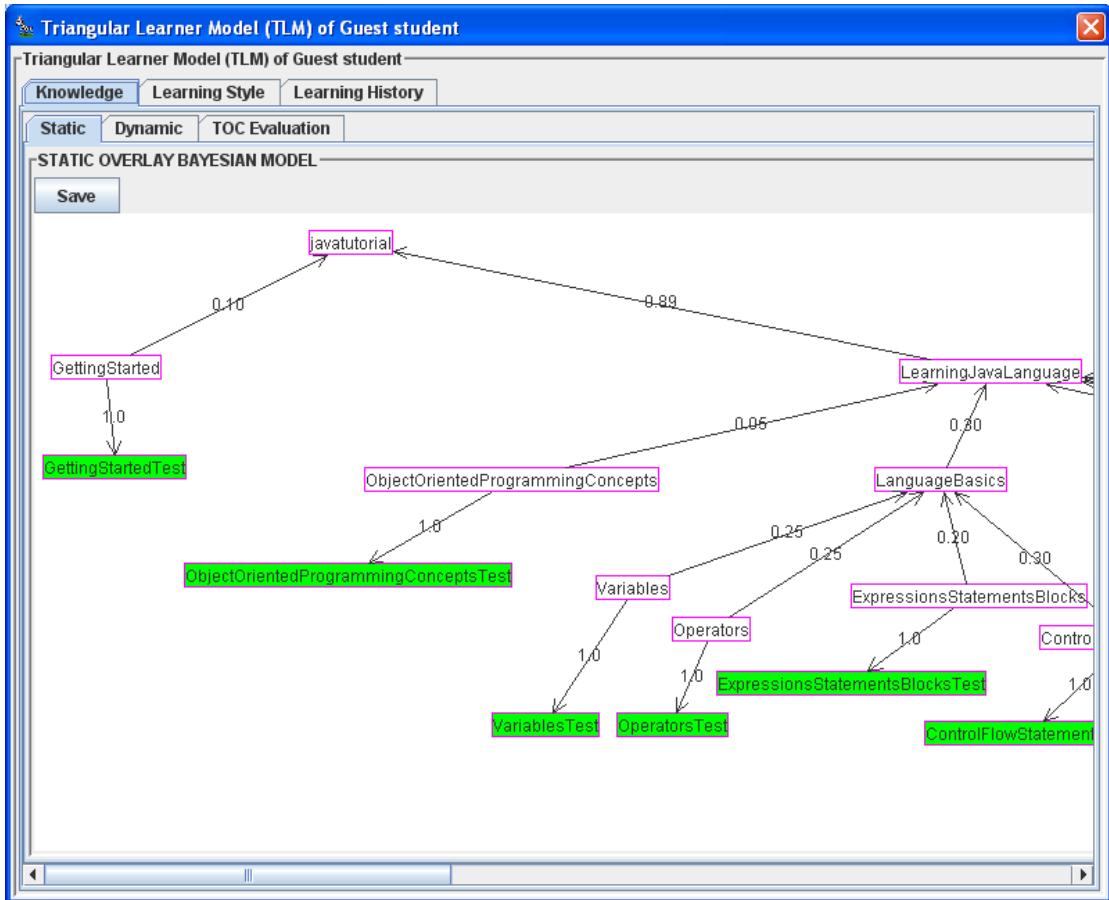


Figure II.2.4.4. TLM of given learner

There are three tabs “Knowledge”, “Learning Style”, and “Learning History” shown in figure II.2.4.4, 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. Software **JavaBayes** (<http://www.cs.cmu.edu/~javabayes/>), version 0.346, year accessed: 2007, developed by author **Fabio Gagliardi Cozman** – Escola Politécnica – University of São Paulo, is used to build up Bayesian network. I express my deep gratitude to the author Fabio Gagliardi Cozman for providing the great Bayesian network software.

Figure II.2.4.4 expresses static Bayesian overlay model in which nodes represent concepts, knowledge items and evidences. Green-shading nodes represent evidences like tests and exercises; please see section III.1 for more details about Bayesian overlay model. Software **JGraph** (<https://www.jgraph.com>) developed by **JGraph Ltd Company**, year accessed: 2008 is used to draw Bayesian network as seen in figure II.2.4.4. I express my deep gratitude to the JGraph Ltd Company for providing the great graph software.

Learning style sub-model is constructed by hidden Markov model (see chapter IV) shown in figure II.2.4.5. Hidden Markov model inside learning style sub-model is developed by authors **Kanav Kahol and Troy L. McDaniel**, year accessed: 2008. I express my deep gratitude to the authors Kanav Kahol and Troy L. McDaniel for providing their great software. BNE is responsible for manipulating knowledge sub-model and learning style sub-model. BNE uses software **Java Scientific Library** (<http://www.ee.ucl.ac.uk/~mflanaga/java>) developed by author **Michael Thomas**

[Flanagan](#), year accessed: 2008 for some mathematical tasks. I express my deep gratitude to the author Michael Thomas Flanagan for providing his great software.

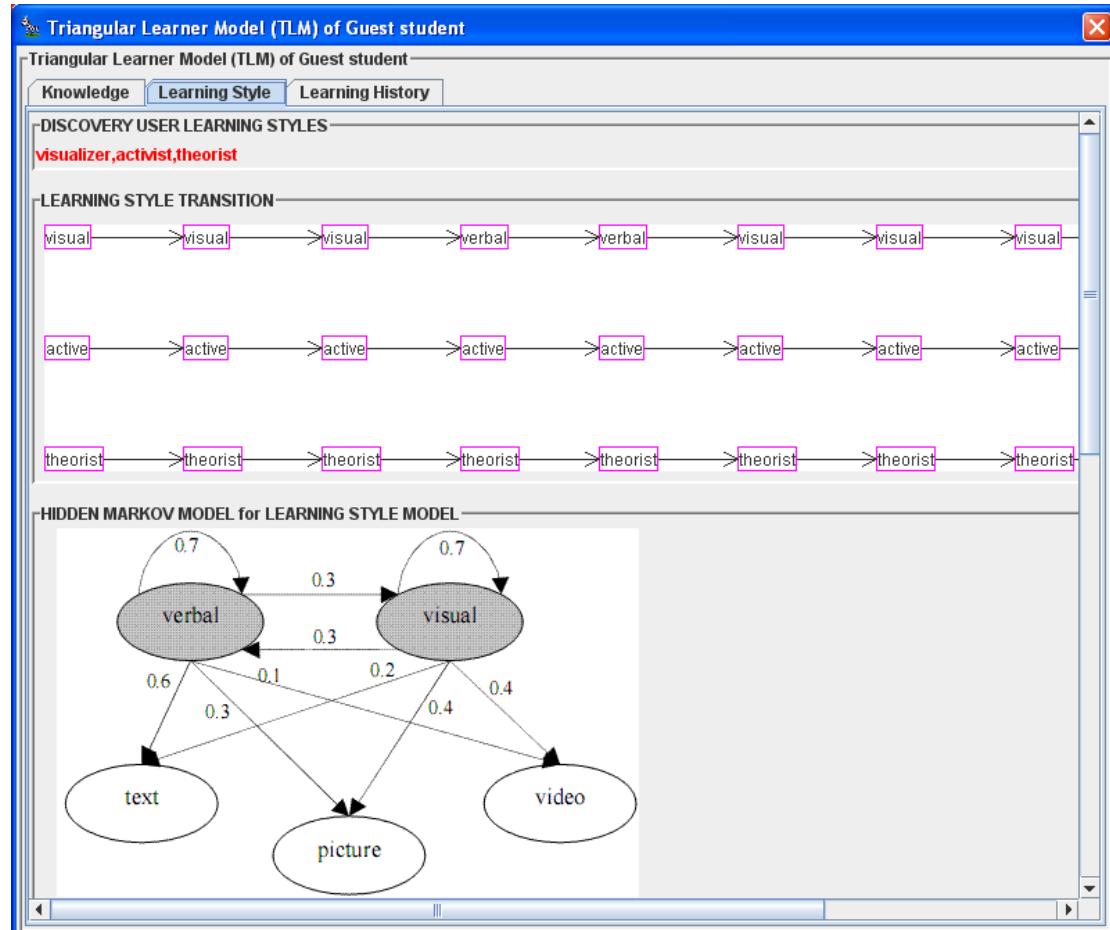


Figure II.2.4.5. Learning style sub-model constructed by hidden Markov model

Recall that learning history sub-model, the most important sub-model among three sub-models of TLM, structures and stores coarse information like learning history data in XML files (W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) or relation database (Ramakrishnan & Gehrke, 2003, pp. 25-94). Figure II.2.4.6 shows a piece of learning history data that learning history sub-model stores and manages. ME is responsible for manipulating learning history sub-model. Please see chapter V for more details about learning history sub-model.

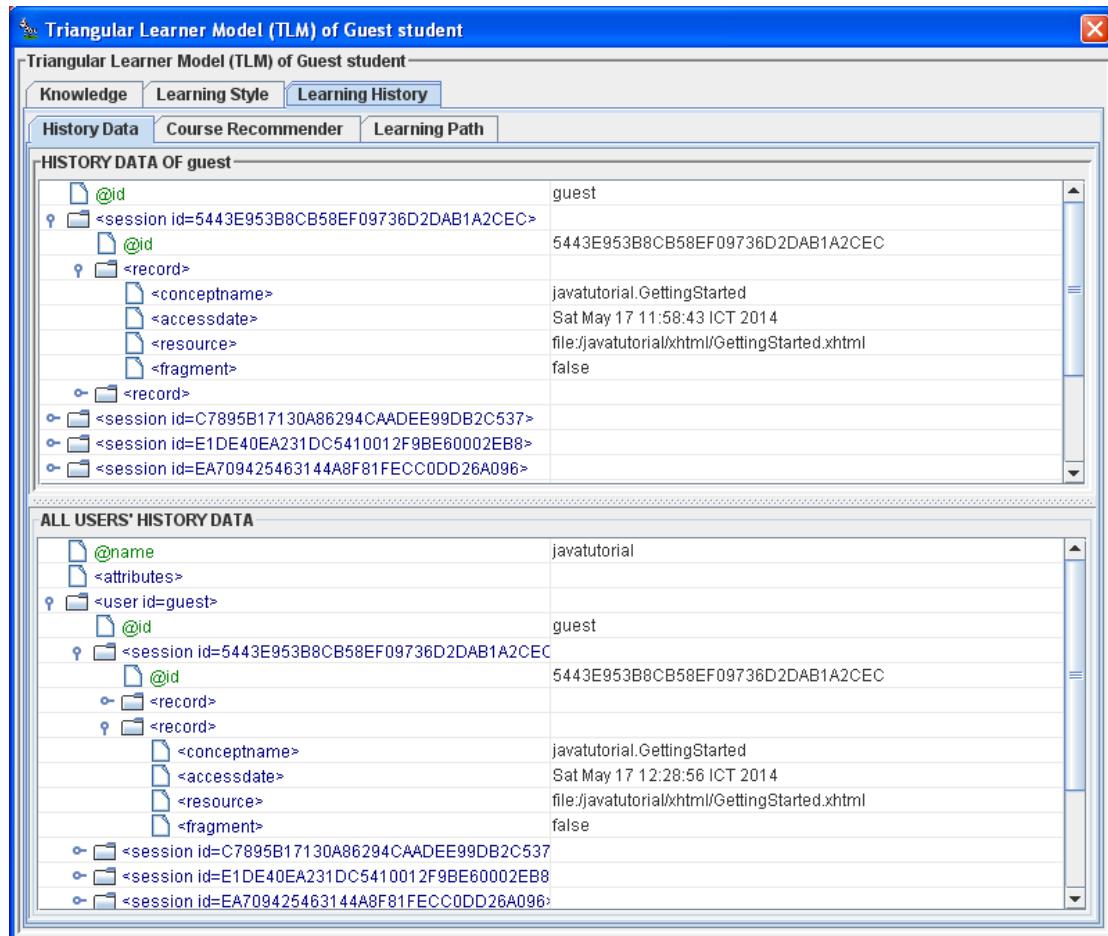


Figure II.2.4.6. Learning history sub-model stores coarse information in XML files

ME uses open source software [Weka](http://www.cs.waikato.ac.nz/ml/weka) (<http://www.cs.waikato.ac.nz/ml/weka>) developed by [Machine Learning Group](#) at the [University of Waikato](#) for mining tasks such as mining recommendation rules, mining sequential patterns, and clustering user groups, year accessed: 2008. I express my deep gratitude to the Machine Learning Group for providing their great software.

The first extended function of learning history sub-model is learning concept and learning path recommendation based on sequential pattern mining (see section [V.1](#)), which is illustrated in figures [II.2.4.7](#) and [II.2.4.8](#).

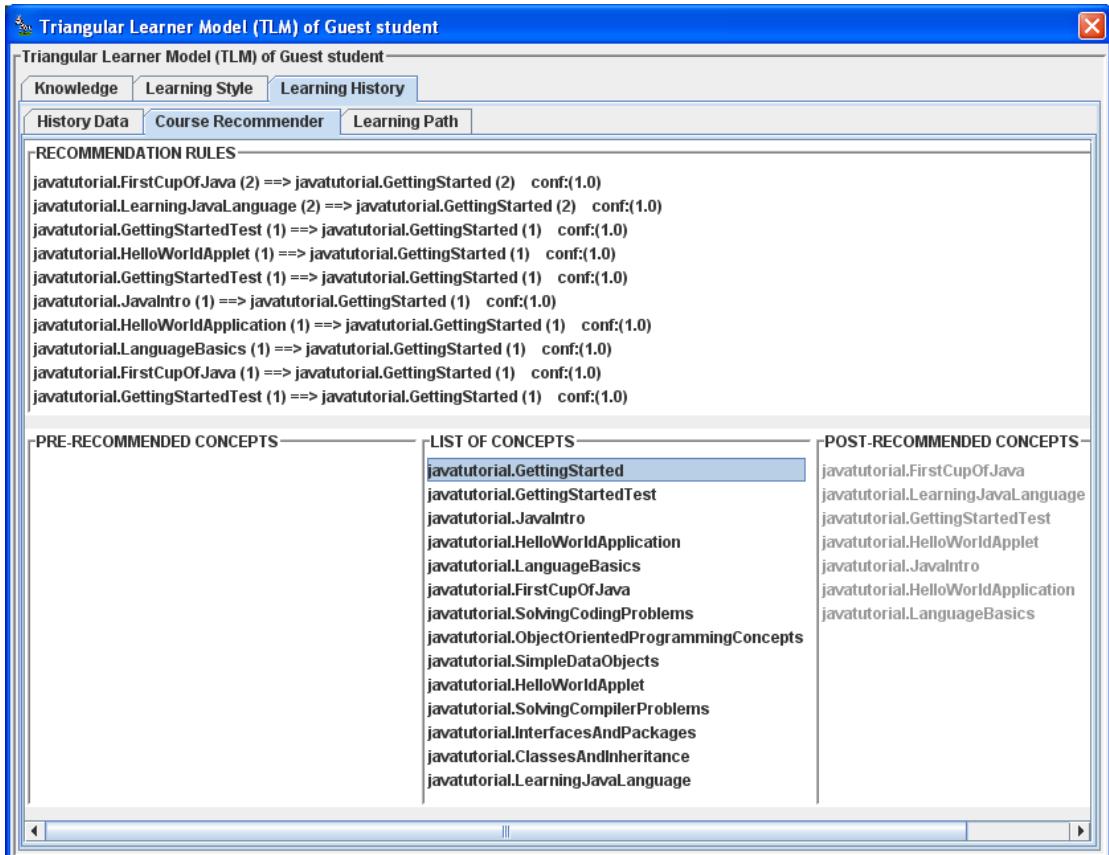


Figure II.2.4.7. Learning concept recommendation

The upper part of figure II.2.4.7 lists recommendation rules or sequential rules which are extracted from sequential patterns. The lower part of figure II.2.4.7 shows recommended concepts derived from recommendation rules. For example, given concept “javatutorial.GettingStarted”, learners are recommended to study successive concepts (post-recommended concepts): “javatutorial.FirstCupOfJava”, “javatutorial.LearningJavaLanguage”, etc. These recommended concepts are shown on e-learning website of the adaptive learning system WOW (see figure II.2.4.18).

The lower part of figure II.2.4.8 lists sequential patterns mined from learning history data. The upper part of figure II.2.4.8 shows course learning paths (learning routes); for example, learning path “javatutorial.GettingStarted” rather than learning path “javatutorial.GettingStarted→javatutorial.GettingStartedTest” is provided to learners because the support of sequential pattern <javatutorial.GettingStarted> (4.0) is higher than the support of sequential pattern <javatutorial.GettingStarted→javatutorial.GettingStartedTest> (3.0). This learning path is recommended on e-learning website of the adaptive learning system WOW (see figure II.2.4.18). Please see section V.1 for more details about sequential pattern mining and learning concept recommendation.

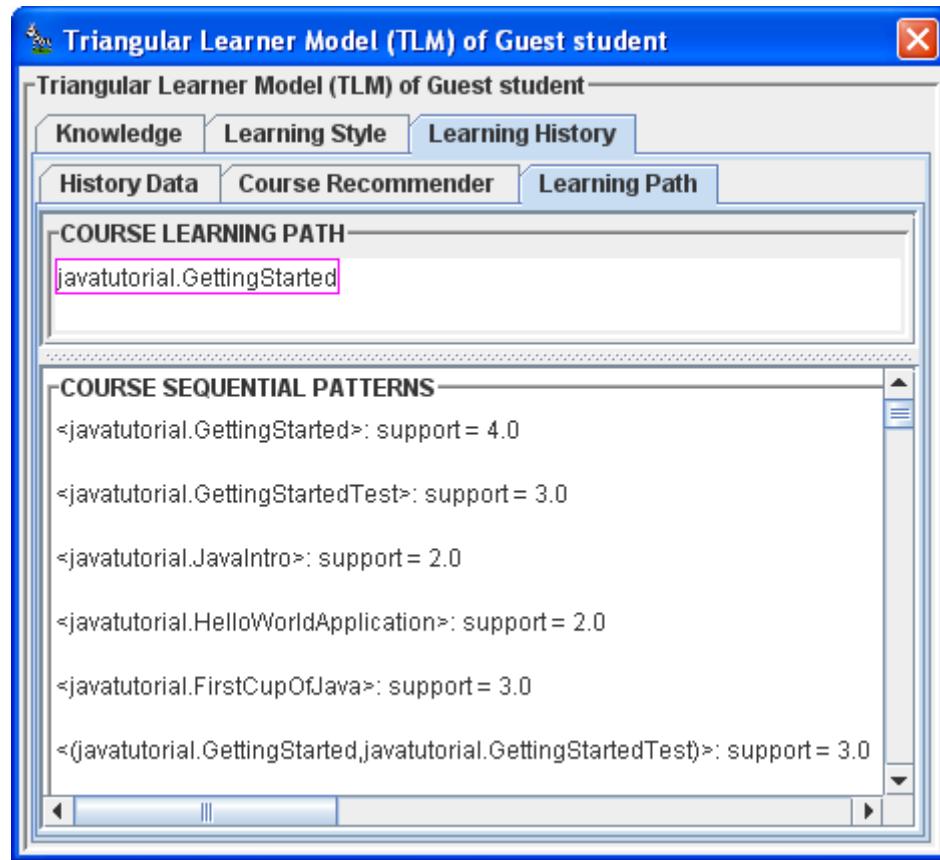


Figure II.2.4.8. Learning path recommendation

The second extended function of learning history sub-model is to discover user interests based on document classification; please see sub-section V.2 for more details about this function. Such function can be performed via Zebra control panel, as shown in figure II.2.4.9.

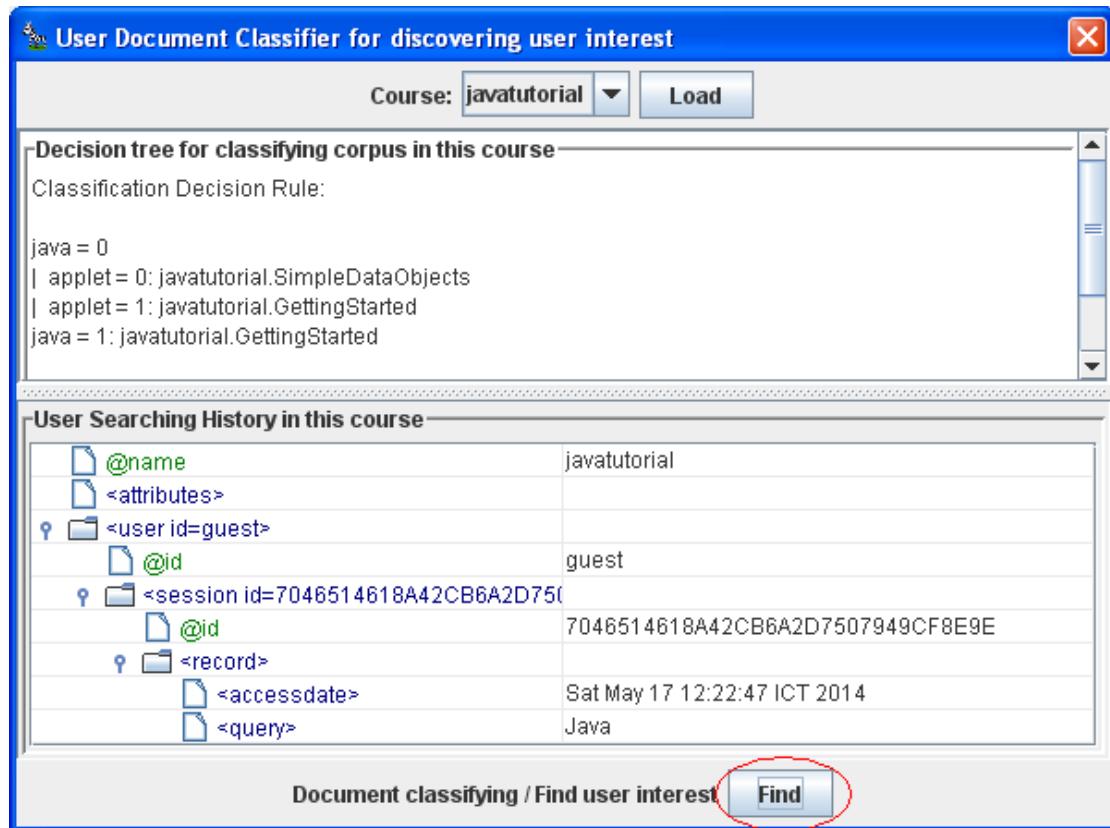


Figure II.2.4.9. Discovering user interests based on document classification

The lower part of figure II.2.4.9 displays learners' searching history and the upper part lists classification rules constructed from such searching history. When button "Find" is clicked, classification rules are applied into discovering interests of a learner.

Figure II.2.4.10 shows the third extended function of learning history sub-model which constructs user groups or user communities.

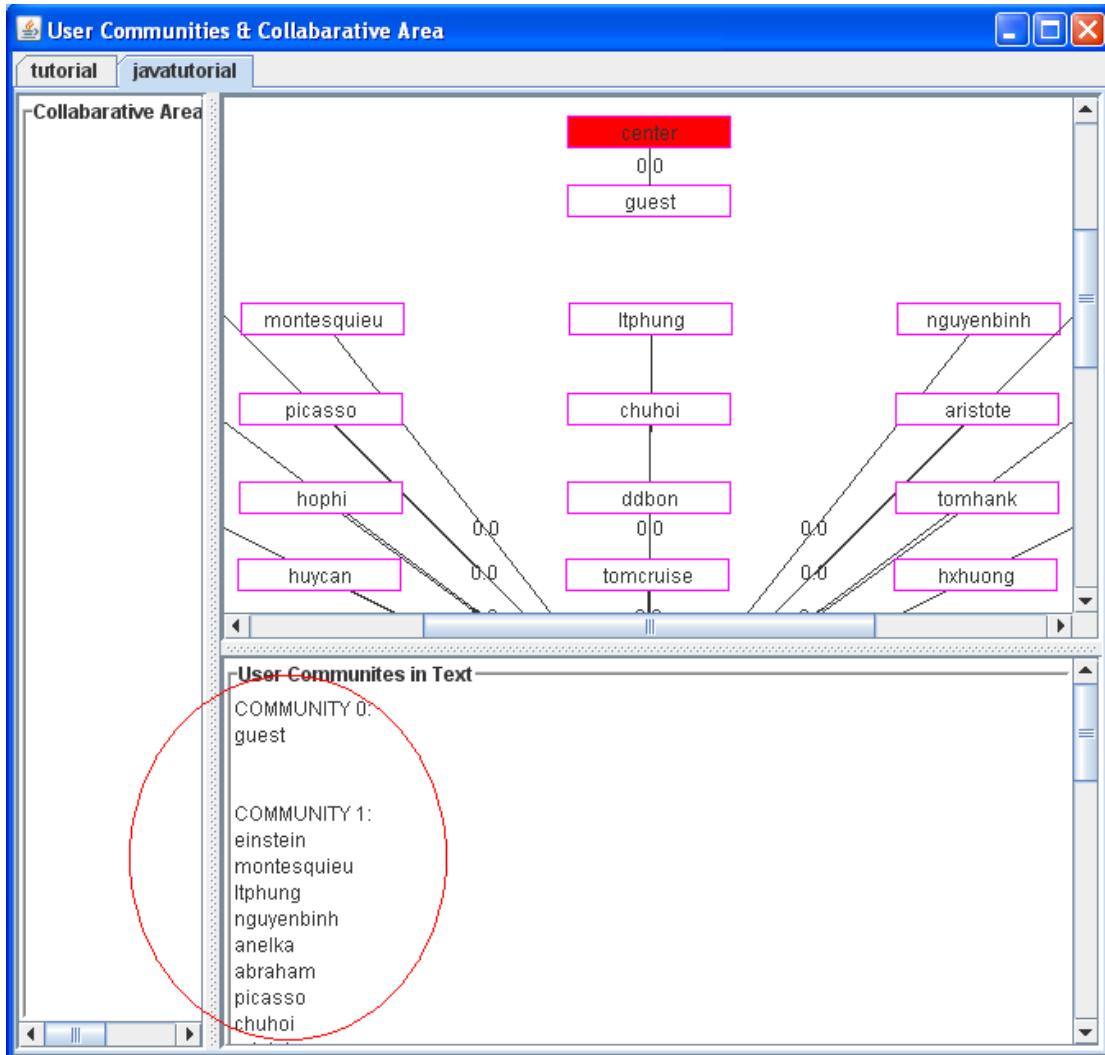


Figure II.2.4.10. Constructing user groups

As seen figure II.2.4.10, there are two communities 0 and 1. Community 0 has only user “Guest student”. These communities provide collaborative area for the adaptive learning system WOW (see figure II.2.4.25).

Recall that there are three tabs “Knowledge”, “Learning Style”, and “Learning History” shown in figure II.2.4.4, 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.2.4.11 expresses evaluation of knowledge of Guest student. According to pie chart in figure II.2.4.11, 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). Software JFreeChart (<http://www.jfree.org/jfreechart>) developed by Object Refinery Limited Company, year accessed: 2008 is used to draw evaluation chart as seen in figure II.2.4.11. I express my deep gratitude to the Object Refinery Limited Company for providing the great chart software.

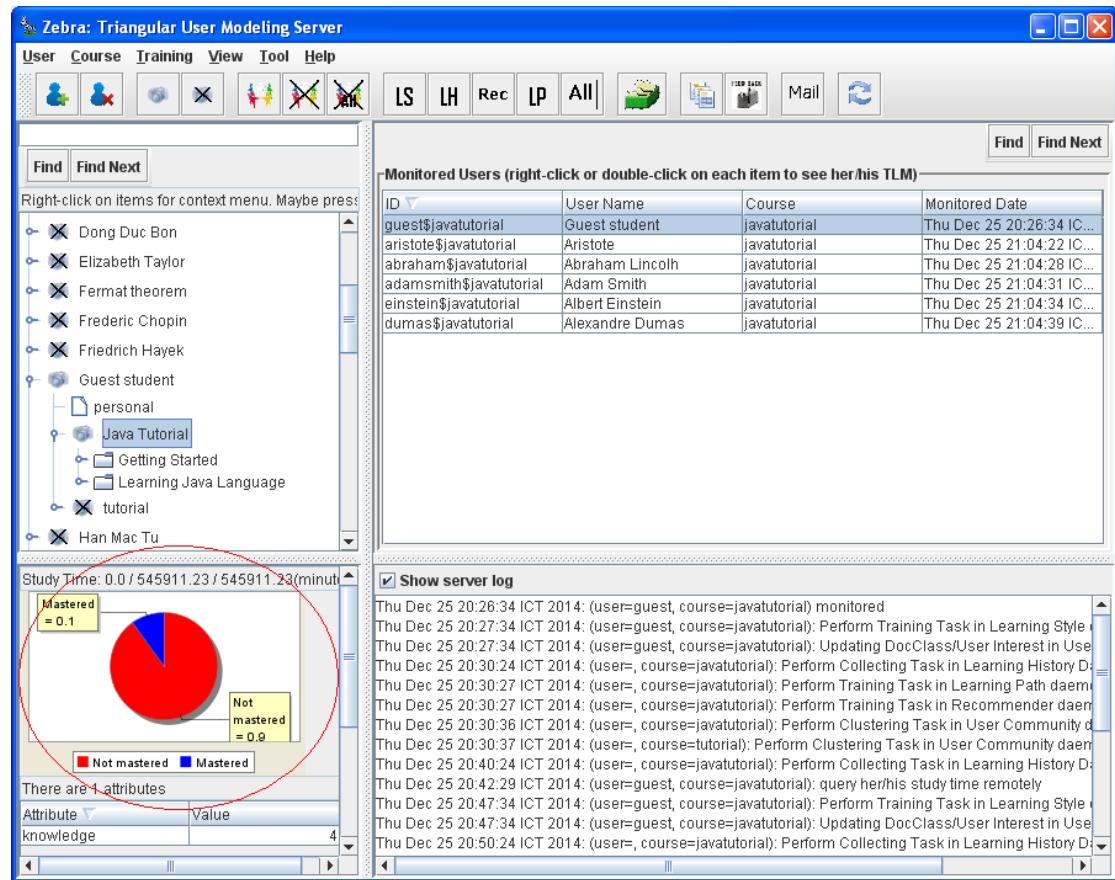


Figure II.2.4.11. Evaluation of learner's knowledge

Learner's knowledge can be evaluated in detailed as shown in figure [II.2.4.12](#).

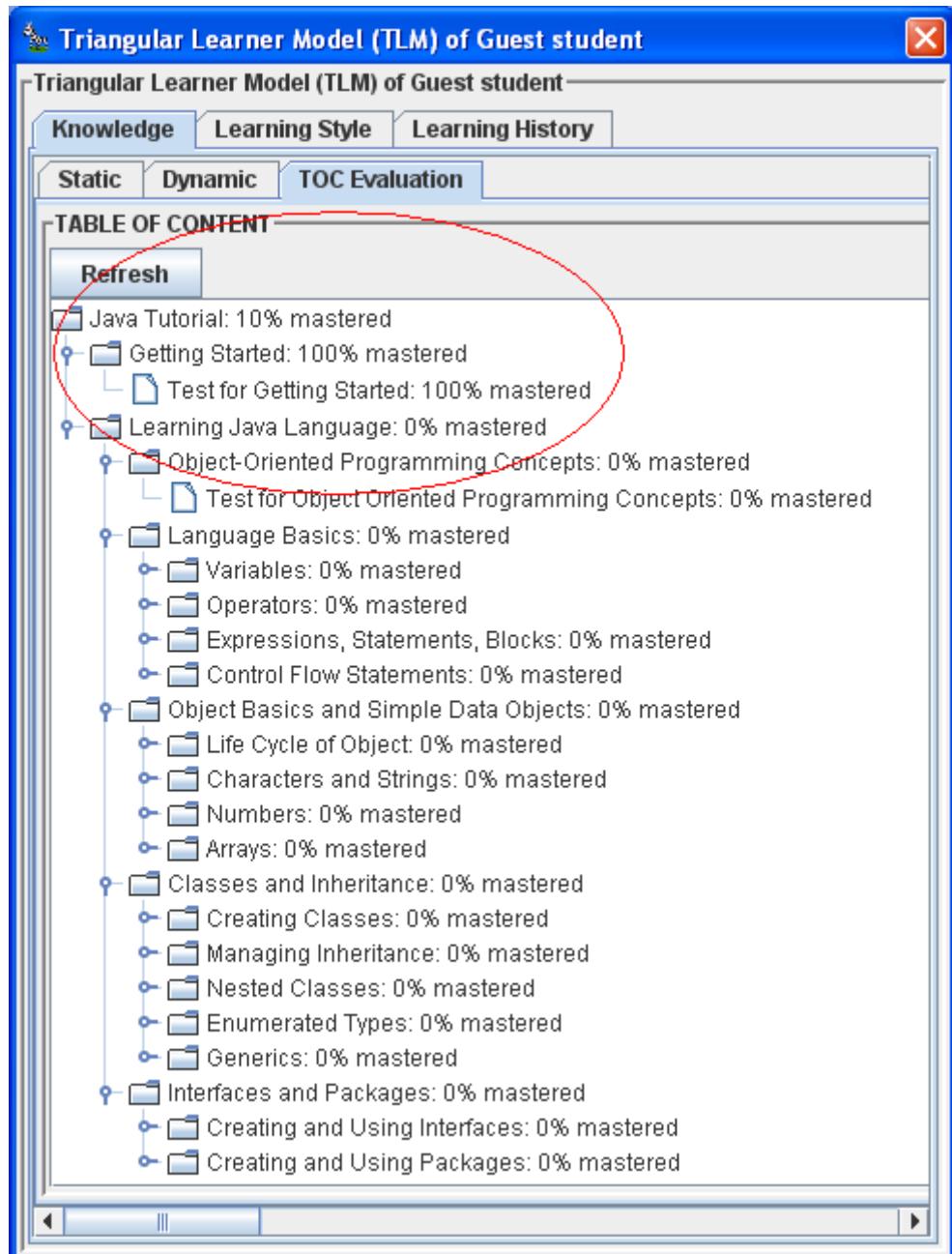


Figure II.2.4.12. Evaluation of learner's knowledge in detailed

Knowledge evaluation is also a main aspect of the adaptive learning system WOW (see figure II.2.4.21). In fact, WOW retrieves evaluation information from Zebra server.

Personal information can be added, deleted, and updated via the control panel, as shown in figure II.2.4.13.

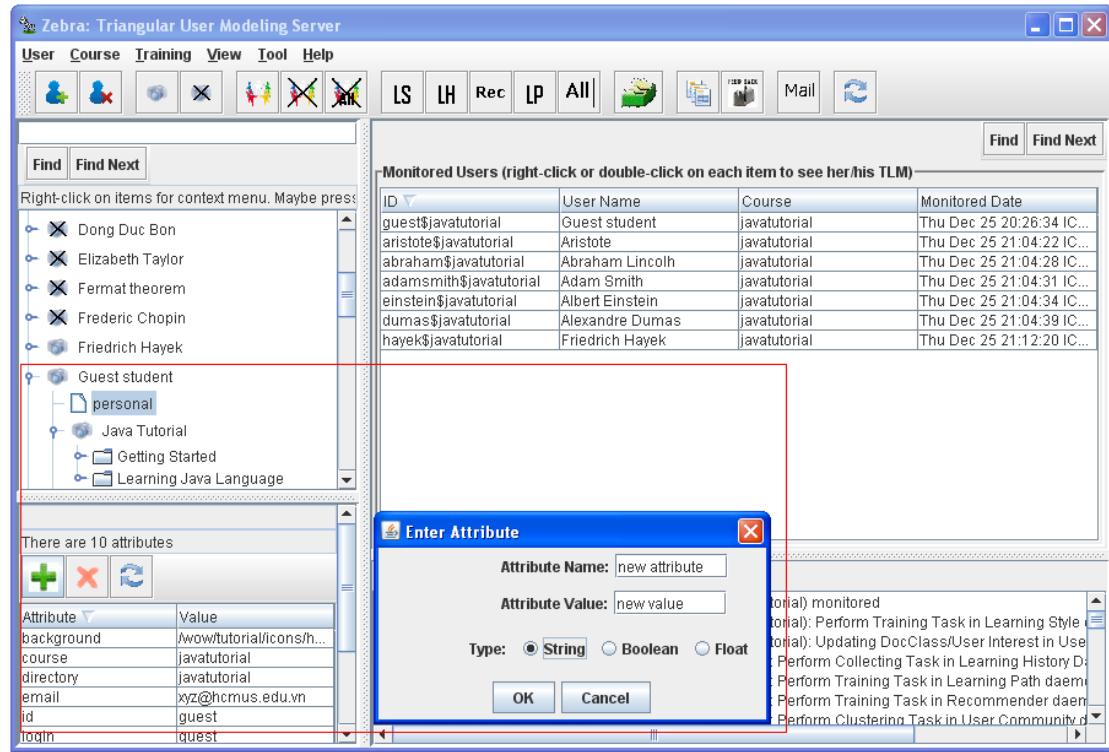


Figure II.2.4.13. Updating personal information

Learner's studying process and studying results are structured in so-called *learning report* shown in figure [II.2.4.14](#).

The screenshot shows a window titled "User Total Report" with a blue header bar. Below the title, the main content area displays the heading "Report on 'Guest student' in 'javatutorial' course". Underneath this, there is a section labeled "TOC" followed by a numbered list of 11 items, each with a blue link:

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

Figure II.2.4.14. Learning report

As seen in figure II.2.4.14, learning report contains 11 items such as Personal Information, Learning Style, Interest, Learning Path, Your Community, Course Evaluation – Study Result, Study Time, Test Log, Access Log, Search Log, and User Feedback, which 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.2.4.15 shows the mailing list tool.

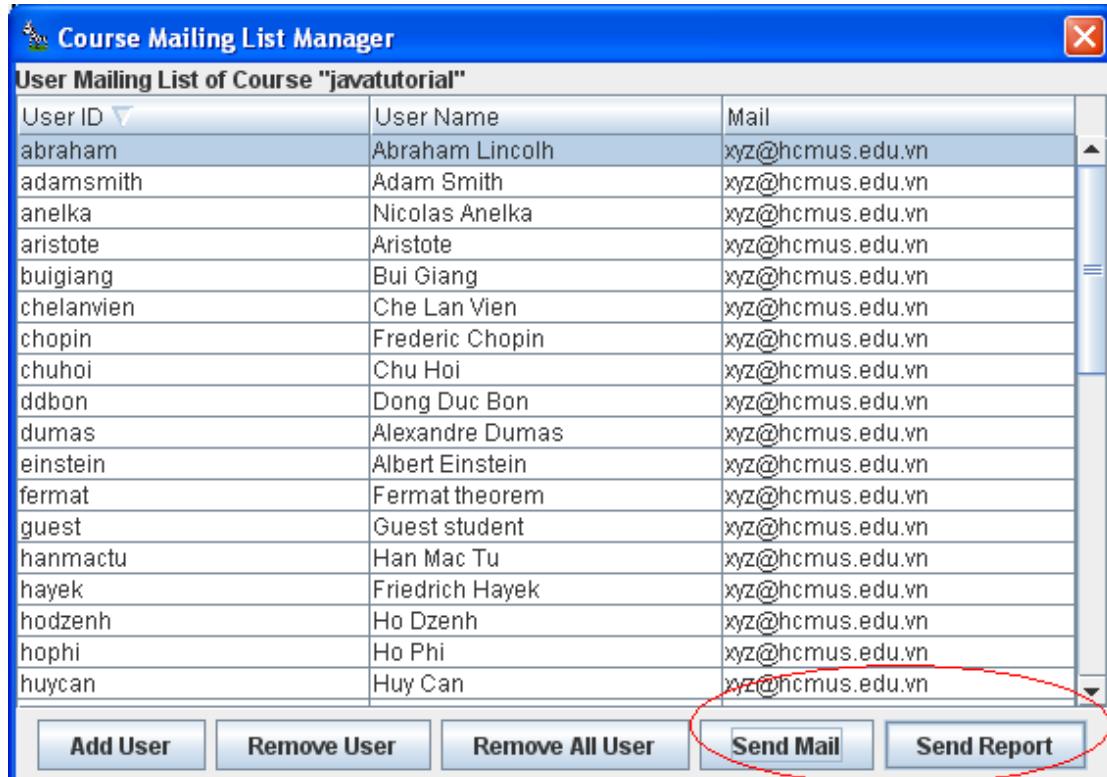


Figure II.2.4.15. Mailing list tool

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 VI.2 for more details about evaluation of user modeling system and adaptive learning system. Figure II.2.4.16 gives an example of feedback report after Zebra server collects such feedbacks.

The screenshot shows a window titled "Feedback Report of course \"javatutorial\"". The main area is a table with columns: "ID", "Question", "Summary1", "Summary2", "Summary3", and "Summary4". The table contains 11 rows of feedback data. At the bottom are two buttons: "Save Report" and "Show Report".

ID	Question	Summary1	Summary2	Summary3	Summary4
1	Do you know what Adaptive Learning is? "Yes": 0/0(0%)	"No": 0/0(0%)		Not answer: 0/0(0%)	
2	Do you know what User Modeling is? "Yes": 0/0(0%)	"No": 0/0(0%)		Not answer: 0/0(0%)	
3	Are you satisfied with the online course ... "Very satisfied": 0/0(0%)	"Satisfied": 0/0(0%)	"Normal": 0/0(0%)	Not answer: 0/0(0%)	
4	Your idea about the way to evaluate yo... "Very good": 0/0(0%)	"Good": 0/0(0%)	"Normal": 0/0(0%)	Not answer: 0/0(0%)	
5	Do you feel about that the evaluation of... "Like much": 0/0(0%)	"Like": 0/0(0%)	"Normal": 0/0(0%)	Not answer: 0/0(0%)	
6	Do you feel about concept recommend... "Like much": 0/0(0%)	"Like": 0/0(0%)	"Normal": 0/0(0%)	Not answer: 0/0(0%)	
7	How about the search engine "Very useful": 0/0(0%)	"Useful": 0/0(0%)	"Normal": 0/0(0%)	Not answer: 0/0(0%)	
8	How about the collaborative learning (c... "Very necessary": 0/0(0%)	"Necessary": 0/0(0%)	"Normal": 0/0(0%)	Not answer: 0/0(0%)	
9	How about the lesson in form of HTML... "Very good": 0/0(0%)	"Good": 0/0(0%)	"Normal": 0/0(0%)	Not answer: 0/0(0%)	
10	Have you ever used other adaptive lear... "Yes": 0/0(0%)	"No": 0/0(0%)	Not answer: 0/0(0%)		
11	What are the best points of this system? "Knowledge evaluation": ... "Many functions": 0/0(0%)	"Friendly GUI": 0/0(0%)	Not answer: 0/0(0%)		

Figure II.2.4.16. Feedback report

Now main aspects of the user modeling system Zebra were described. It is necessary to describe the adaptive learning system which interacts with Zebra according to figures II.2.3.1 and II.2.3.2. 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 AHA! system; please see sub-section [I.2.3.3](#) for more details about AHA!. The AHA! system is developed by the authors De Bra and Calvi (De Bra & Calvi, 1998) and so, the WOW system shows my deep gratitude to the author [Paul De Bra](#). Moreover, I also thank [Eindhoven University of Technology](#) where the AHA! system was developed. The AHA! system is available at internet link <http://aha.win.tue.nl>. The AHA! 3.0 tutorial is available at <http://aha.win.tue.nl:18080/aha/tutorial>.

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). WOW packed with Zebra server is available at internet link as follows:

<http://zebra.locnguyen.net>

The main aspect of WOW is to support evaluation of user knowledge based on Bayesian overlay model. According to section [III.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.2.4.1](#), 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.2.4.17](#) shows WOW login web page (Wikipedia, Web page, 2014) in which Guest student studies.

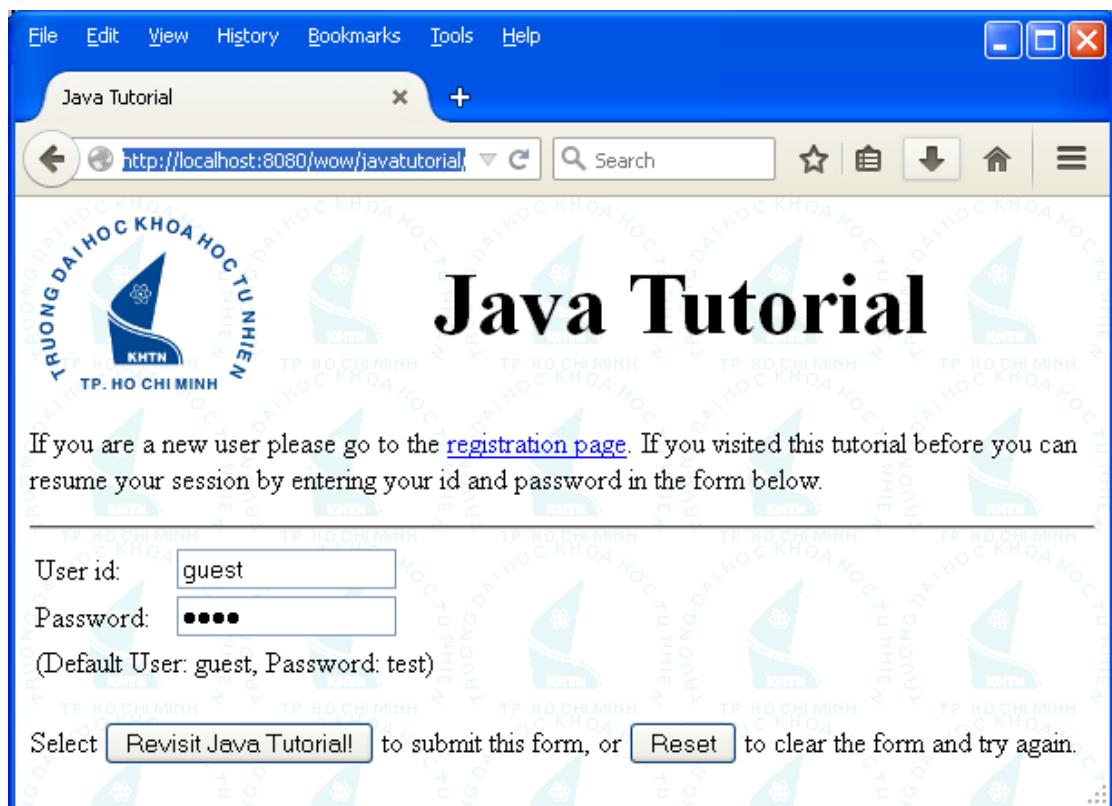


Figure II.2.4.17. WOW login web page

Figure II.2.4.18 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 with 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.2.4.18, 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 with the concept that learners study. This part supports adaptive presentation. Please see sub-section I.2.3 for more details about adaptive presentation and adaptive navigation.

Figure II.2.4.18. Typical adaptive learning web page supported by WOW

As seen in figure II.2.4.18, 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.2.4.19 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. The online test module inside AHA! is developed by Universidad de Córdoba (<http://www.uco.es>), Spain, 2008. I express my deep gratitude to the Universidad de Córdoba for providing the great online test module.

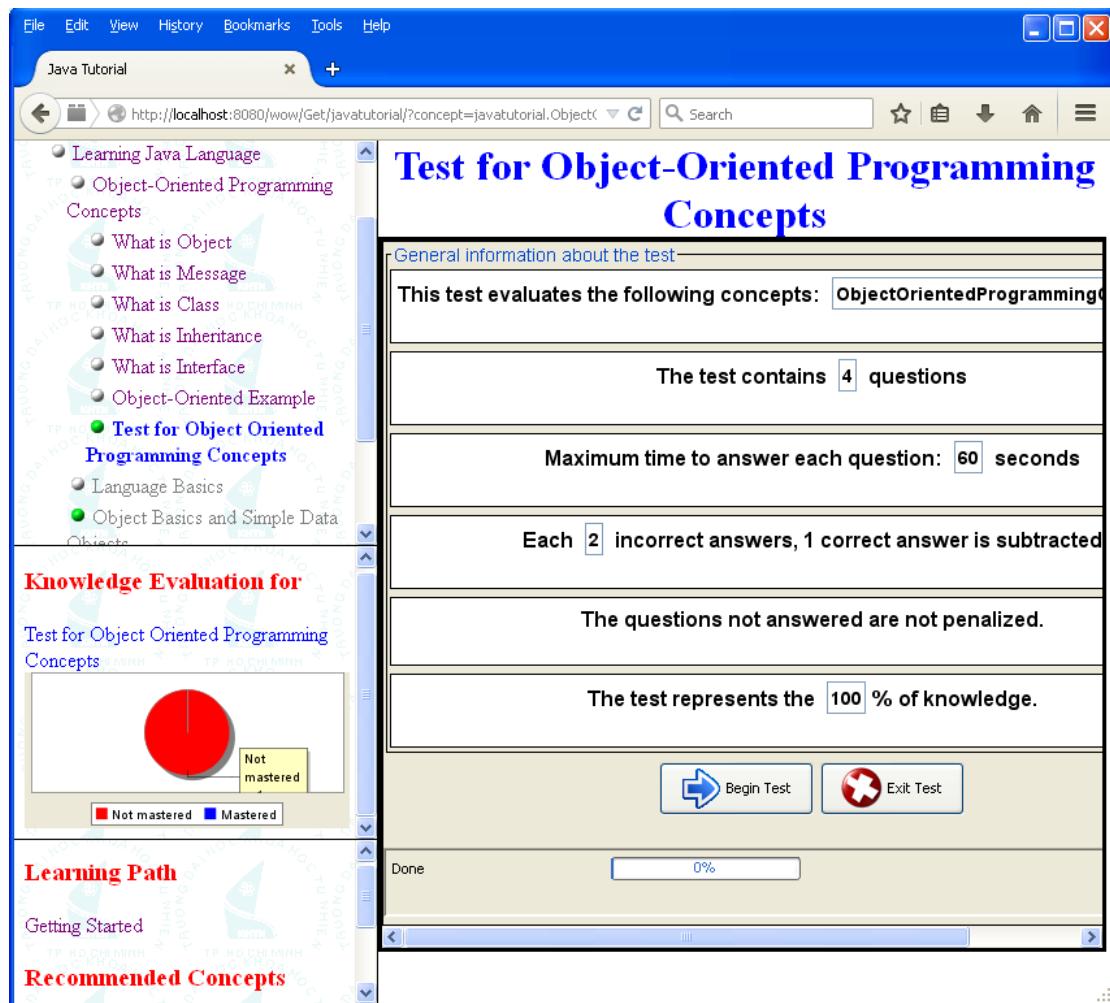


Figure II.2.4.19. Learner does online test via WOW

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.2.4.20.

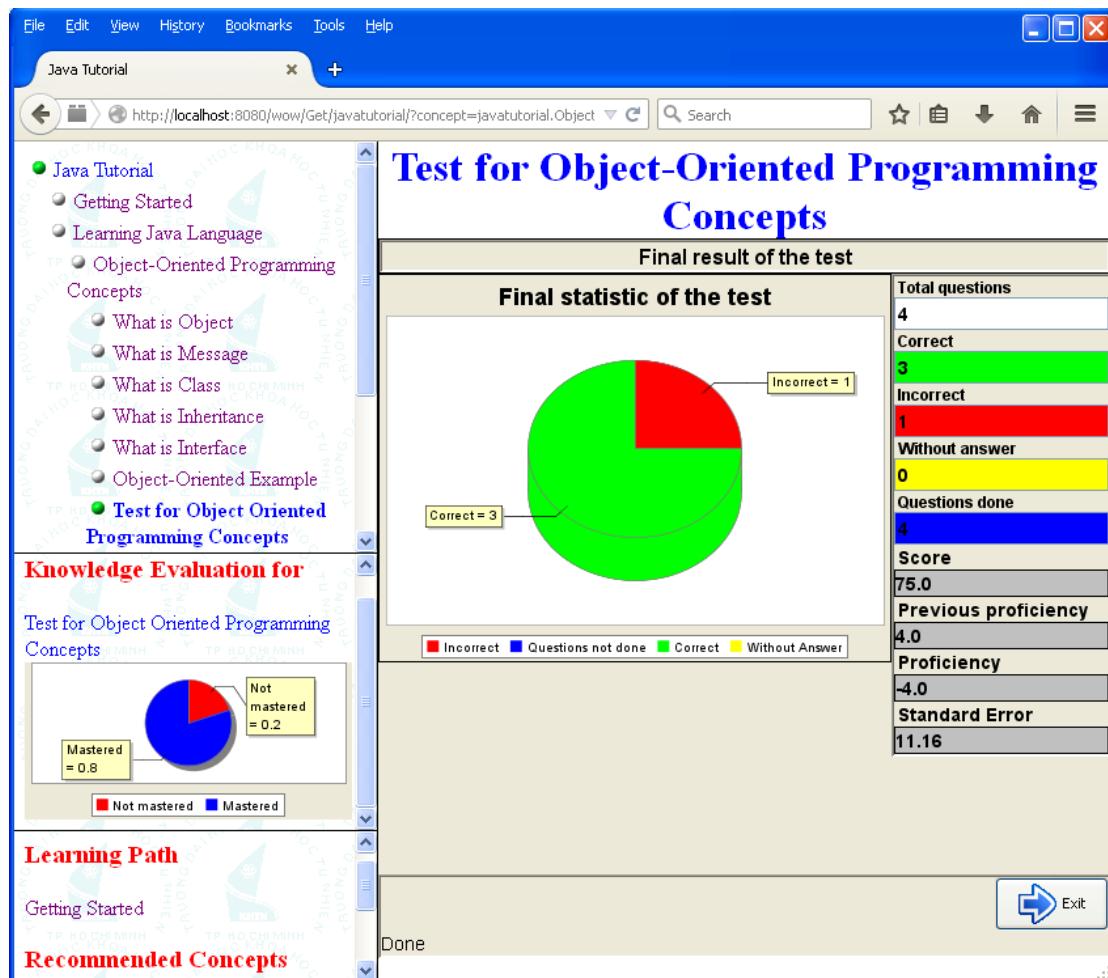


Figure II.2.4.20. Result of online test

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.2.4.21 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". A red banner at the top of the sidebar says "Knowledge Evaluation for Object-Oriented Programming Concepts". Below it is a pie chart showing the following data:

Status	Value
Mastered	= 0.821
Not mastered	= 0.179

Below the chart, there are three sections: "What Is an Object?", "What Is a Message?", and "What Is a Class?". Each section contains a brief description and a small diamond icon.

Figure II.2.4.21. 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.2.4.20), inference mechanism inside Bayesian overlay model is executed and the posterior probability represented by indicator “mastered” gets 0.821. Figure II.2.4.22 expresses the posterior probability of concept “Object Oriented Programming Concepts” and it is $0.820694791 \approx 0.821$.

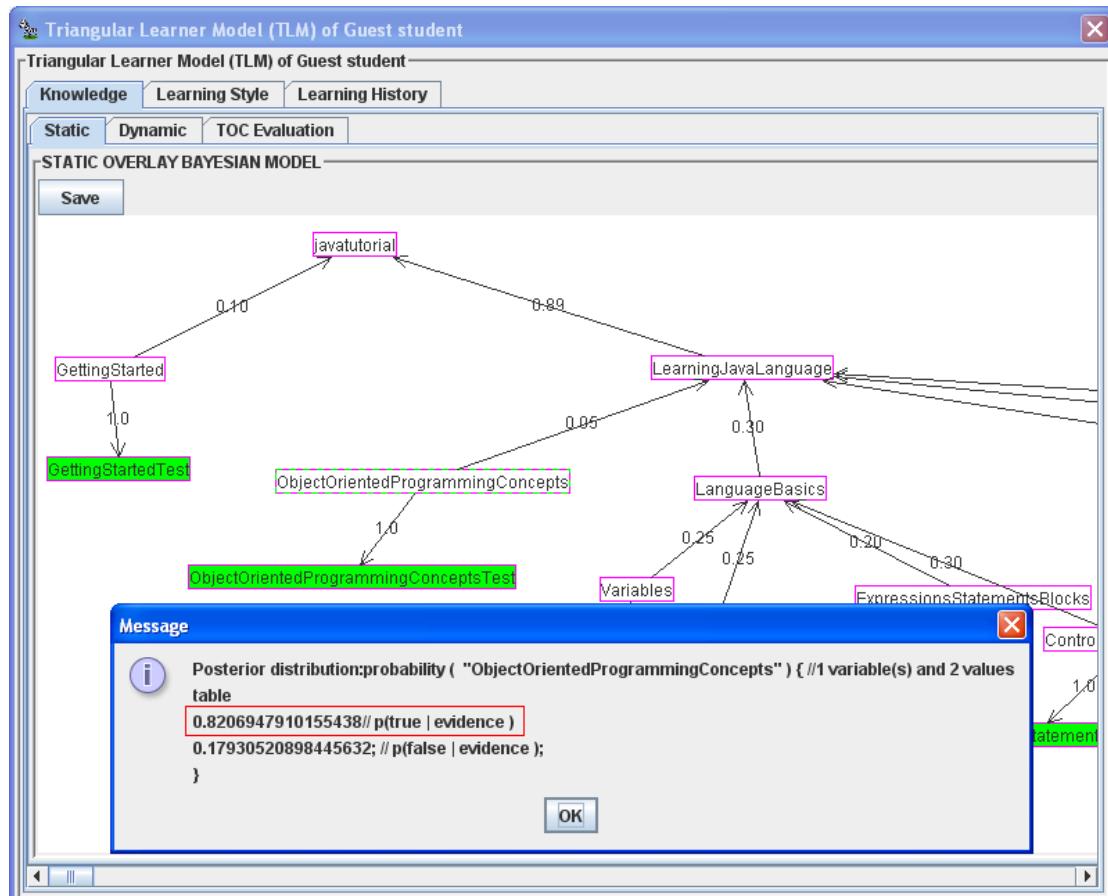


Figure II.2.4.22. Posterior probability of given concept

Note that Bayesian overlay model is built in Zebra server and so, it is easy to recognize that figure II.2.4.22 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.2.4.4 and II.2.4.22 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.2.4.23 shows this search engine.

WOW uses software Lucene Core (<https://lucene.apache.org>) developed by Lucene™/Solr™ Committers (<https://lucene.apache.org/whoweare.html>) at The Apache Software Foundation as default search engine, year accessed: 2008. I express my deep gratitude to the Lucene™/Solr™ Committers for providing their great search engine.

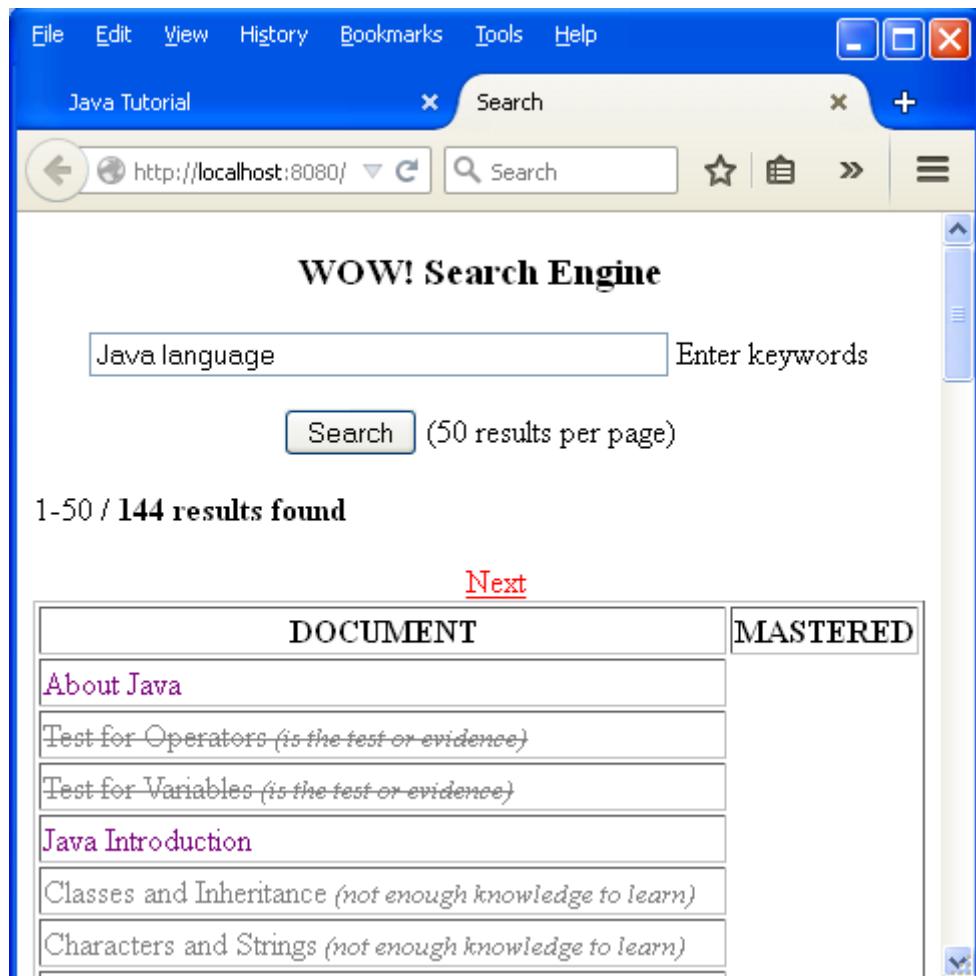


Figure II.2.4.23. Search engine in WOW

Based on aforementioned knowledge evaluation (see figure II.2.4.22), 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.2.4.23, 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.2.4.16). 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.2.4.24 shows the WOW feedback tool.

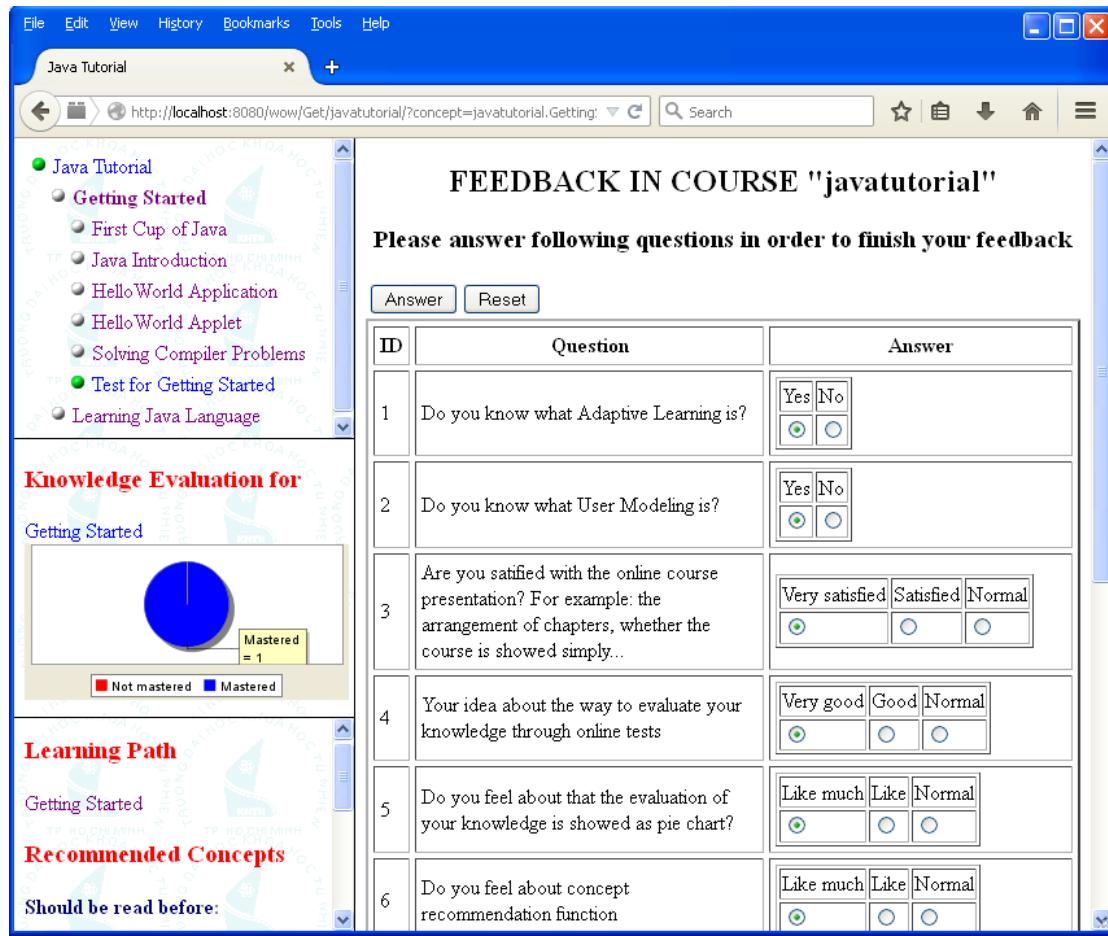


Figure II.2.4.24. WOW feedback tool

Recall that the third extended function of learning history sub-model is constructing user groups or user communities (see figure II.2.4.10). The WOW system helps learners to interact with other learners in the same community via *collaborative area* shown in figure II.2.4.25.

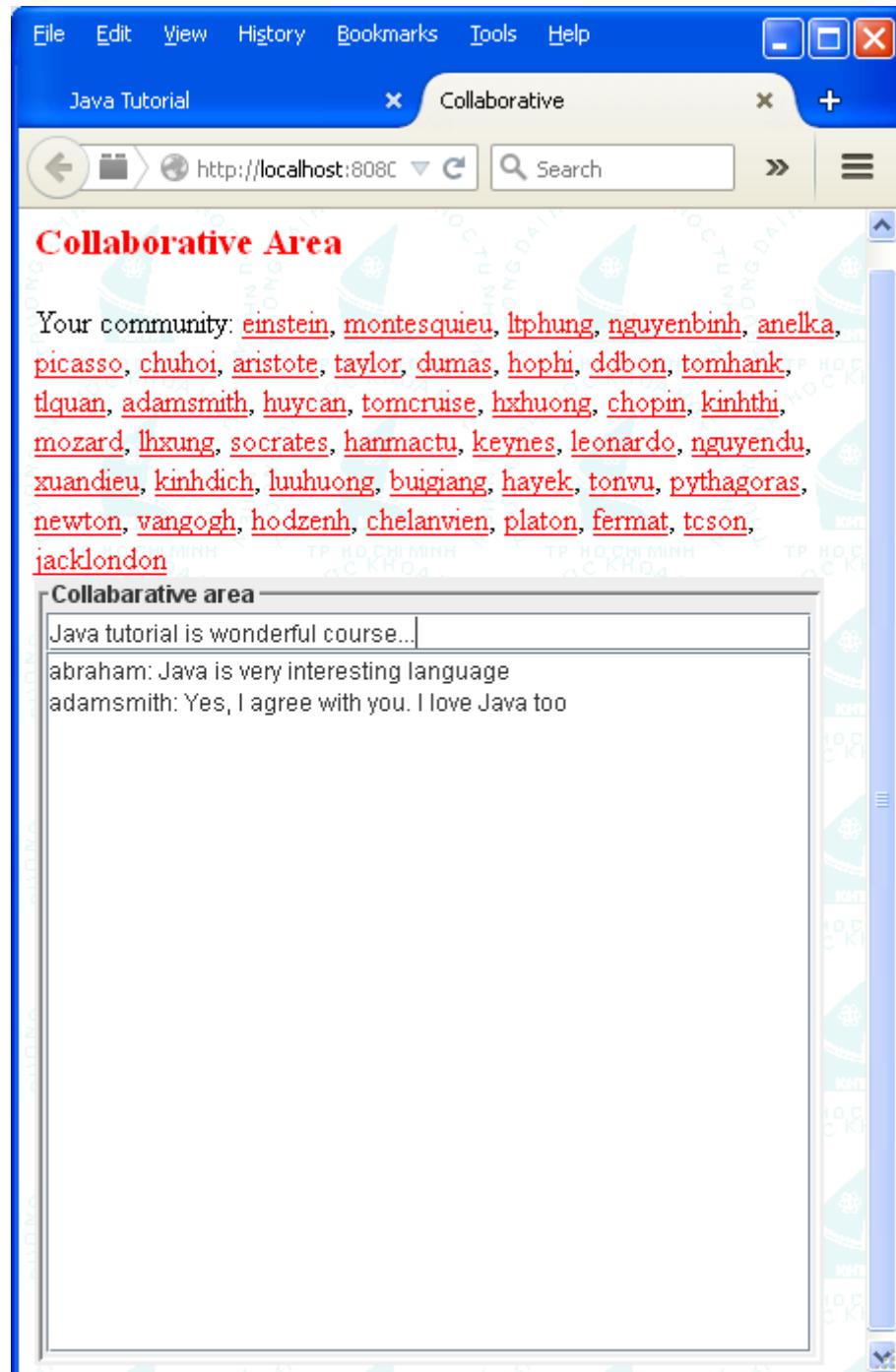


Figure II.2.4.25. Collaborative area for collaborative learning

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 an advancement that WOW is wrapped as a portlet (Abdelnur & Hepper, 2003) when WOW is deployed in portal environment (Kofman, 2009). Some typical portal environments are Liferay (<http://www.liferay.com>) and eXo Platform (<http://www.exoplatform.com>). WOW portlet uses the Ajax library Dojo Toolkit (<http://dojotoolkit.org>) developed by The Dojo Foundation, year accessed: 2008 for speeding up response of WOW. Readers

should refer to (W3Schools, AJAX Tutorial) in order to comprehend Ajax technology.

Another client that is called *thin client* also queries users' information from Zebra server. Note that this thin client interacts 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.2.4.26 shows the thin client connecting to Zebra server.

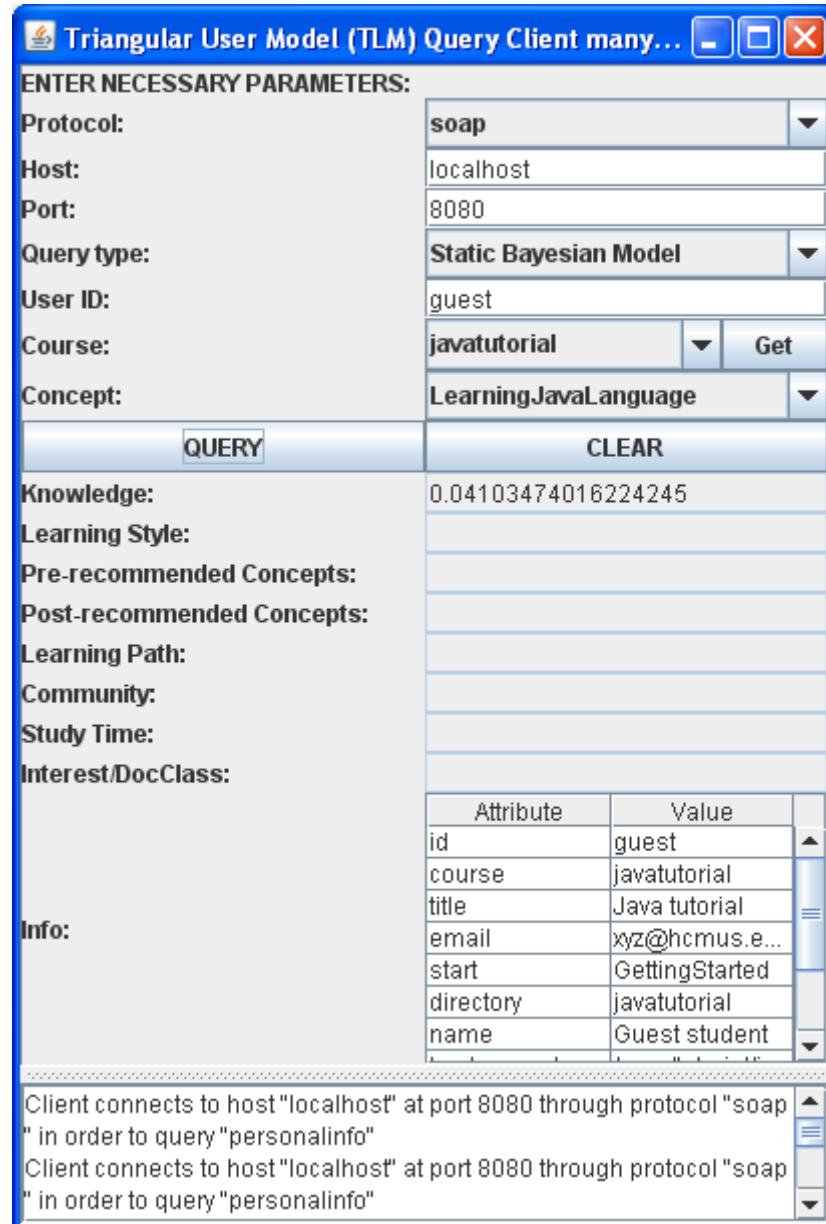


Figure II.2.4.26. Thin client connecting Zebra server

The thin client packed with Zebra server is available at internet link as follows:

<http://zebra.locnguyen.net>

Zebra server uses web services engine [Axis \(https://axis.apache.org\)](https://axis.apache.org) developed by [The Apache Software Foundation](#) to publish CI. Other softwares used by Zebra server and WOW include [JComponentPack \(http://zfqljava.com\)](http://zfqljava.com) 2006 developed by [Extreme Component Company](#), Java Mail 2009 developed by [Sun Microsystems](#), SwingX 2007 developed by [SwingLabs](#), Sortable Table by David Gilbert & Nobuo Tamemasa,

JavaScript lightweight graph [Mochikit \(<http://mochi.github.io/mochikit>\)](http://mochi.github.io/mochikit) developed by [Mochi Media, Inc.](#), and some tools supported by Java communities over the world. I express my deep gratitude to these individuals, organizations, communities, and companies for providing their great softwares.

In general, Zebra server is open source software whose first release version 3.3 launched in November 21, 2009 from which there is no significant change until the final release version 3.10 launched in February 20, 2010. Therefore, I encourage software developers to improve and extend Zebra server. This is my biggest honor and happiness. In December 5, 2014 Zebra server is honorably demonstrated at [The 17th International Conference on Interactive Computer aided Learning \(ICL2014\)](#), [The 2014 World Engineering Education Forum](#), Dubai, UAE. Recall that Zebra server is available at internet link as follows:

<http://zebra.locnguyen.net>

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. The comprehensive description of these sub-models is mentioned in successive chapters [III](#), [IV](#), and [V](#).

In general, the user modeling system that builds up and manipulates TLM is called [Zebra](#). Zebra includes the core engine and a set of communication interfaces. The core engine is the composition of two sub-engines: mining engine (ME) and belief network engine (BNE). ME structures TLM and applies data mining techniques into discovering other characteristics beyond knowledge and learning style. ME manages learning history sub-model. BNE is responsible for inferring new information from TLM by using deduction mechanism available in belief network. BNE manages knowledge sub-model and learning style sub-model. Communication interfaces (CI) allow users to see or modify restrictedly TLM. Adaptive applications also interact with Zebra through CI. 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 chapters [III](#), [IV](#), and [V](#) which will describe particularly TLM and Zebra.

Chapter III. Knowledge sub-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 which are mentioned in successive chapters [III](#), [IV](#), and [V](#), respectively. Please pay attention to the coherence of such three main chapters together with respective sub-models. Now this chapter [III](#) focuses on the knowledge sub-model.

Recall that 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 sub-model is 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 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 complex teaching curriculum, most teachers can't assess user knowledge unless they use powerful math tools which are implemented in this sub-model. This chapter [III](#) has five main sections:

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

The first section [III.1](#) is the most important; it describes in detailed how to construct knowledge sub-model by applying Bayesian network. Section [III.2](#) discusses how to take advantage of knowledge sub-model so as to perform adaptive learning tasks. Sections [III.3](#) and [III.4](#) discuss 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. Section [III.5](#) is an extra section that 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. 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 (incorrect knowledge).

Modeling user must follow three steps below:

- 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 out new information about user 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 that 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 section [III.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](#). After that main method of applying Bayesian network into overlay model, the core of my approach, is described in sub-section [III.1.2](#). Sub-section [III.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](#) if knowing Bayesian network but please pay attention to two main sub-sections [III.1.2](#) and [III.1.3](#). The advanced sub-section [III.1.4](#) discusses relationship conversion that is an extension of the combination method mentioned basically in sub-section [III.1.2](#). Evaluation of the combination between Bayesian network and overlay model is mentioned in sub-section [III.1.5](#). 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. Bayesian network

This sub-section [III.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 by an initial probability or prior probability. Note, in classical statistical theory, the random variable's probability is objective (physical) through trials. But, in Bayesian method, the probability of hypothesis is "personal" because its initial value is set subjectively by expert. When evidences were gathered enough, the hypothesis is considered trustworthy.

Bayesian inference is based on so-called Bayes' rule or Bayes' theorem (Wikipedia, Bayesian inference, 2006) specified in formula III.1.1.1 as follows:

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

Formula III.1.1.1. Bayes' rule

Where,

- H is probability variable denoting a hypothesis existing before evidence.
- D is also probabilistic variable denoting an observed evidence. It is conventional that notations d , D and \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 and Runger (Montgomery & Runger, 2003, p. 4) for more details about sample and statistics.
- $P(H)$ is *prior probability* of hypothesis H . It reflects the degree of subjective belief in hypothesis H .
- $P(H|D)$, conditional probability of H with given D , is called *posterior probability*. It tells us the changed belief in hypothesis when occurring evidence. Whether or not the hypothesis in Bayesian inference is considered trustworthy is determined based on the posterior probability. In general, posterior probability is cornerstone of Bayesian inference.
- $P(D|H)$ is conditional probability of occurring evidence D when hypothesis H was given. In fact, likelihood ratio is $P(D|H) / P(D)$ but $P(D)$ is constant value. So we can consider $P(D|H)$ as *likelihood function* of H with fixed D . Please pay attention to the conditional probability because it is mentioned over the whole research.
- $P(D)$ is probability of occurring evidence D together all mutually exclusive cases of hypothesis. If H and D are discrete, then $P(D) = \sum_H P(D|H)P(H)$, 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 expressed in formulas III.1.1.2 and III.1.1.3, respectively as follows:

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

Formula III.1.1.2. Additional rule

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

Formula III.1.1.3. 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). The probability $P(X, Y)$ is often known as joint probability.

If X and Y are mutually exclusive ($X \cap Y = \emptyset$) then, $X \cup Y$ is often denoted as $X+Y$ and we have:

$$P(X + Y) = P(X) + P(Y)$$

(Due to $P(\emptyset) = 0$)

X and Y are mutually independent if and only if one of three following conditions is satisfied:

$$\begin{aligned} P(X \cap Y) &= P(X)P(Y) \\ P(X|Y) &= P(X) \\ P(Y|X) &= P(Y) \end{aligned}$$

When X and Y are mutually independent, $X \cap Y$ are often denoted as XY and we have:

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

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.1.1.4 as follows:

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

Formula III.1.1.4. Total probability rule

Where $X_1 + X_2 + \dots + X_n = \Omega$ and $X_i \cap X_j = \emptyset, \forall i, j$.

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.1.1.5. Total probability rule in continuous case

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

Please pay attention to Bayes' rule (formula III.1.1.1) and total probability rule (formulas III.1.1.4 and III.1.1.5) because they are used frequently over the whole research.

Bayesian network (BN)

Bayesian network (BN) (Neapolitan, 2003, p. 40) (Nguyen L., Overview of Bayesian Network, 2013, p. 1) is combination of graph theory and Bayesian inference. It having a set of nodes and a set of directed arcs is the directed acyclic graph (DAG); please pay attention to the terms “DAG” and “BN” because they are used over the whole research. Each node represents a random variable which can be an evidence or hypothesis in Bayesian inference. Each arc reveals the relationship among two nodes. If there is the arc from node A to B , we call “ A causes B ” or “ A is parent of B ”, in other words, A depends conditionally on B . Otherwise there is no arc between A and B , it asserts the conditional independence. Note, in BN context, terms: *node and variable are the same*.

A node has a local Conditional Probability Distribution (CPD) with attention that conditional probability distribution is often called shortly *probability distribution* or *distribution*. If variables are discrete, CPD is simplified as Conditional Probability Table (CPT). If variables are continuous, CPD is often called conditional Probability Density Function (PDF) which will be mentioned in sub-section III.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 derivative of CDF; in other words, CDF is integral of PDF. In practical statistics, PDF is used more commonly than CDF is used and so, PDF is mentioned over the whole research. Note, notation $P(\cdot)$ 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.1.1.6 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.1.1.6. Definition of cumulative distribution function (CDF) and probability density function (PDF)

Because this sub-section III.1.1 focuses on BN, please read (Montgomery & Rungar, 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.1.1.1, event “cloudy” is cause of event “rain” which in turn is cause of “grass is wet” (Murphy, 1998). So we have three causal relationships of: 1- cloudy to rain, 2- rain to wet grass, 3- sprinkler to wet grass. This model is expressed below by BN with four nodes and three arcs corresponding to four events and three relationships. Every node has two possible values True (1) and False (0) together its CPT.

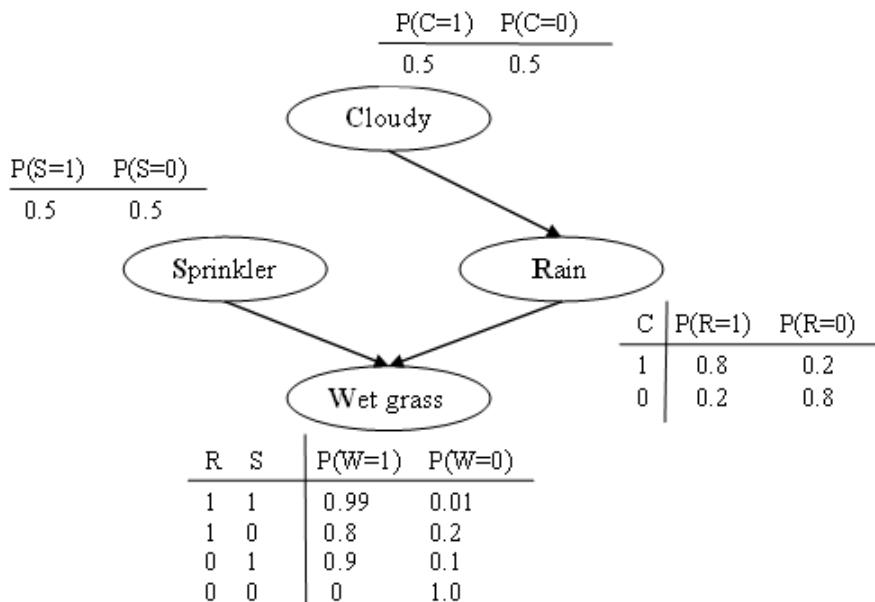


Figure III.1.1.1. Bayesian network (a classic example about wet grass)

Note that random variables C , S , R , and W denote phenomena or events such as cloudy, sprinkler, rain, and wet grass, respectively and the table next to each node expresses the CPT of such node. For instance, focusing on the CPT attached to node “Wet grass”, if it is rainy ($R=1$) and garden is sprinkled ($S=1$), it is almost certain that grass is wet ($W=1$). Such assertion can be represented mathematically by the condition probability of event “grass is wet” ($W=1$) given evident events “rain” ($R=1$) and “sprinkler” ($S=1$) is 0.99 as in the attached table, $P(W=1|R=1,S=1) = 0.99$. As seen, the conditional probability $P(W=1|R=1,S=1)$ is an entry of the CPT attached to node “Wet grass”. In general, BN consists of two models such as qualitative model and quantitative model. Qualitative model is the structure as the graph shown in figure III.1.1.1. 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 sections III.3, III.4, and III.5. This section III.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.

Let $\{X_1, X_2, \dots, X_n\}$ be the set of nodes in BN, the **Global Joint Probability Distribution (GJPD)** is defined as the probability function of event $\{X_1=x_1, X_2=x_2, \dots, X_n=x_n\}$ (Neapolitan, 2003, p. 24). Such joint probability function satisfies two conditions specified by formula III.1.1.7:

$$P(X_1, X_2, \dots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \dots P(X_n|X_1, X_2, \dots, X_{n-1})$$

Formula III.1.1.7. Two conditions of joint probability distribution

Suppose PA_i is the set of direct parent nodes such that X_i only depends on variables in PA_i . In other words, the DAG satisfies Markov condition in which there is always an arc from each variable in PA_i to X_i and no intermediate node between them. Thus, the joint probability $P(X_1, X_2, \dots, X_n)$ is specified by formula III.1.1.8.

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i|PA_i)$$

Formula III.1.1.8. Reduced global joint probability distribution of random vector

As a convention, BN satisfies Markov condition and so its joint probability specified by formula III.1.1.8 is product of all local CPT (s) with note that $P(X_i|PA_i)$ in formula III.1.1.8 is 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.1.1.9 as below:

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

Formula III.1.1.9. 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} given X_i and $P(\mathcal{D})$ is probability of occurring \mathcal{D} together all mutually exclusive cases of X . From formulas III.1.1.8 and III.1.1.9, we gain formula III.1.1.10 as follows:

$$P(X_i|\mathcal{D}) = \frac{P(X_i, \mathcal{D})}{P(\mathcal{D})} = \frac{\sum_{X \setminus (\{X_i\} \cup \mathcal{D})} P(X_1, X_2, \dots, X_n)}{\sum_{X \setminus \mathcal{D}} P(X_1, X_2, \dots, X_n)}$$

Formula III.1.1.10. 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.1.1.10 is the base for inference inside Bayesian network, which is used over the whole research. Formulas III.1.1.9 and III.1.1.10 are extensions of Bayes' rule specified by formula III.1.1.1. It is not easy to understand formula III.1.1.10 and so, please see formulas III.1.1.12 and III.1.1.13 which are advanced posterior probabilities applied into wet grass example in order to comprehend formula III.1.1.10.

From figure III.1.1.1 of wet grass example, we have:

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

Applying formula III.1.1.8, $P(S|C)=P(S)$ due to no conditional independence assertion about variables S and C . Furthermore, because S is intermediate node between C and W , we should remove C from $P(W | C, R, S)$, hence $P(W | C, R, S) = P(W | R, S)$. In short, applying formula III.1.1.8, we have formula III.1.1.11 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.1.1.11. Global joint probability distribution of wet grass Bayesian network

Inference in Bayesian network

Using Bayesian inference, we need to compute the posterior probability of each hypothesis node in network. In general, the computation based on Bayesian rule is known as the inference in BN.

Reviewing figure III.1.1.1, suppose W becomes evidence variable which is observed as the fact that the grass is wet, so, W has value 1. There is request for answering the question: how to determine which cause (sprinkler or rain) is more possible for wet grass. Hence, we will calculate two posterior probabilities of $R (=1)$ and $S (=1)$ in condition $W (=1)$. Such probabilities called *explanations* for W are simple forms of formula III.1.1.10, expended by formulas III.1.1.12 and III.1.1.13 as follows:

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

Formula III.1.1.12. Posterior probability of rain given wet grass evidence

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

Formula III.1.1.13. Posterior probability of sprinkler given wet grass evidence

Note that the numerator in the right side of formula III.1.1.12 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.1.1.11 for global joint probability distribution of “wet grass” Bayesian network, we have:

$$\begin{aligned}
 & \sum_{C,S} P(C, R = 1, S, W = 1) \\
 &= (P(C = 1) * P(S = 1) * P(R = 1|C = 1) * P(W = 1|R = 1, S = 1)) \\
 &+ (P(C = 1) * P(S = 0) * P(R = 1|C = 1) * P(W = 1|R = 1, S = 0)) \\
 &+ (P(C = 0) * P(S = 1) * P(R = 1|C = 0) * P(W = 1|R = 1, S = 1)) \\
 &+ (P(C = 0) * P(S = 0) * P(R = 1|C = 0) * P(W = 1|R = 1, S = 0)) \\
 &= (0.5 * 0.5 * 0.8 * 0.99) + (0.5 * 0.5 * 0.8 * 0.8) + (0.5 * 0.5 * 0.2 * 0.99) \\
 &\quad + (0.5 * 0.5 * 0.2 * 0.8) \\
 &= 0.4475
 \end{aligned}$$

It is easy to infer that there is the same interpretation for numerators and denominators in right sides of formulas III.1.1.12 and III.1.1.13 and the previous formula III.1.1.10 is also understood simply by this way when $\{C, S\} = \{C, R, S, W\} \setminus \{R, W\}$ and fixing $\{R, W\}$. In similar, we have:

$$\begin{aligned}
 & \sum_{C,R} P(C, R, S = 1, W = 1) \\
 &= (P(C = 1) * P(S = 1) * P(R = 1|C = 1) * P(W = 1|R = 1, S = 1)) \\
 &+ (P(C = 1) * P(S = 1) * P(R = 1|C = 1) * P(W = 1|R = 0, S = 1)) \\
 &+ (P(C = 0) * P(S = 1) * P(R = 1|C = 0) * P(W = 1|R = 1, S = 1)) \\
 &+ (P(C = 0) * P(S = 1) * P(R = 1|C = 0) * P(W = 1|R = 0, S = 1)) \\
 &= (0.5 * 0.5 * 0.8 * 0.99) + (0.5 * 0.5 * 0.8 * 0.9) + (0.5 * 0.5 * 0.2 * 0.99) \\
 &\quad + (0.5 * 0.5 * 0.2 * 0.9) \\
 &= 0.4725
 \end{aligned}$$

$$\begin{aligned}
 & \sum_{C,R,S} P(C, R, S, W = 1) \\
 &= (P(C = 1) * P(S = 1) * P(R = 1|C = 1) * P(W = 1|R = 1, S = 1)) \\
 &+ (P(C = 1) * P(S = 0) * P(R = 1|C = 1) * P(W = 1|R = 1, S = 0)) \\
 &+ (P(C = 1) * P(S = 1) * P(R = 0|C = 1) * P(W = 1|R = 0, S = 1)) \\
 &+ (P(C = 1) * P(S = 0) * P(R = 0|C = 1) * P(W = 1|R = 0, S = 0)) \\
 &+ (P(C = 0) * P(S = 1) * P(R = 1|C = 0) * P(W = 1|R = 1, S = 1)) \\
 &+ (P(C = 0) * P(S = 0) * P(R = 1|C = 0) * P(W = 1|R = 1, S = 0)) \\
 &+ (P(C = 0) * P(S = 1) * P(R = 0|C = 0) * P(W = 1|R = 0, S = 1)) \\
 &+ (P(C = 0) * P(S = 0) * P(R = 0|C = 0) * P(W = 1|R = 0, S = 0)) \\
 &= (0.5 * 0.5 * 0.8 * 0.99) + (0.5 * 0.5 * 0.8 * 0.8) + (0.5 * 0.5 * 0.2 * 0.9) \\
 &\quad + (0.5 * 0.5 * 0.2 * 0) + (0.5 * 0.5 * 0.2 * 0.99) \\
 &\quad + (0.5 * 0.5 * 0.2 * 0.8) + (0.5 * 0.5 * 0.8 * 0.9) \\
 &\quad + (0.5 * 0.5 * 0.8 * 0) \\
 &= 0.6725
 \end{aligned}$$

In fact, formulas III.1.1.12 and III.1.1.13 are expansions of formula III.1.1.10. As a result, we have:

$$P(R = 1|W = 1) = \frac{\sum_{C,S} P(C, R = 1, S, W = 1)}{\sum_{C,R,S} P(C, R, S, W = 1)} = \frac{0.4475}{0.6725} \approx 0.67$$

$$P(S = 1|W = 1) = \frac{\sum_{C,R} P(C, R, S = 1, W = 1)}{\sum_{C,R,S} P(C, R, S, W = 1)} = \frac{0.4725}{0.6725} \approx 0.70$$

Obviously, the posterior probability of event “sprinkler” ($S=1$) is larger than the posterior probability of event “rain” ($R=1$) given evidence “wet grass” ($W=1$), which leads to conclusion that sprinkler is the most likely cause of wet grass.

Now basic concepts of Bayesian network were introduced in this sub-section [III.1.1](#). The proposed approach which combines Bayesian network and overlay model in order to construct knowledge sub-model is described thoroughly in successive sub-section [III.1.2](#).

III.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.1.2.2.1](#)). 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 relationships. 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.1.3](#).

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 (tests, exams, exercises, user’s feedback, user’s activities, etc.) which are used to assess or evaluate user’s performance in learning process are considered as evidence variables. We must add these evidence variables to Bayesian network along with determining conditional

dependence relationships between them and remaining hidden variables, namely, specifying their CPT (s). Inferring user's knowledge is to compute posterior probability of hidden variables according to formula III.1.1.10 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 is an evidence node “*Exercise: set up class Human*” which is an exercise. 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. These weights were proposed by authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, p. 287), which express degrees of importance of concepts. Authors Millán and Pérez-de-la-Cruz claimed that “the weight of a concept can be computed as the number of days associated with it over the number of days associated with the topic it belongs to” (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, p. 288).

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. The aggregation relationship was mentioned in (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, p. 289). Note that the set of all parents of a hidden variable is the complete set of mutually exclusive 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.1.2.1 depicts the weighted graph representing Java course with note that such graph is the structure of Bayesian overlay model. This structure follows the granularity hierarchy proposed in (Millán & Pérez-de-la-Cruz, 2002, pp. 287-288) and (Millán, Loboda, & Pérez-de-la-Cruz, Bayesian networks for student model engineering, 2010, p. 1675)

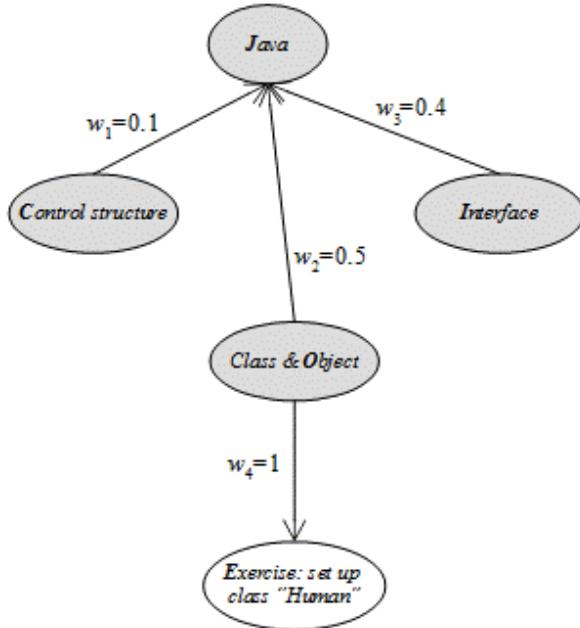


Figure III.1.2.1. Structure of Bayesian overlay model for Java course

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.1.3. 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 $w_3=0.4$.

Suppose student has mastered concepts C and I but it is not asserted if she/he has mastered concept O . Therefore, table III.1.2.1 specifies prior probabilities (CPT) of nodes C , O and I .

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

Table III.1.2.1. 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 instance, the conditional probability of $J=1$ given $C=1$, $O=1$, and $I=1$ is computed by formula III.1.2.1 as follows:

$$P(J|C, O, I) = h_1 w_1 + h_2 w_2 + h_3 w_3$$

Where $h_1 = \begin{cases} 1 & \text{if } C = J \\ 0 & \text{else} \end{cases}$, $h_2 = \begin{cases} 1 & \text{if } O = J \\ 0 & \text{else} \end{cases}$, and $h_3 = \begin{cases} 1 & \text{if } I = J \\ 0 & \text{else} \end{cases}$

Formula III.1.2.1. Conditional probability of $J=1$ given $C=1$, $O=1$, and $I=1$

For instance, we have:

$$\begin{aligned} P(J = 1 | C = 1, O = 1, I = 1) &= 1 * w_1 + 1 * w_2 + 1 * w_3 \\ &= 1 * 0.1 + 1 * 0.5 + 1 * 0.4 = 1.0 \end{aligned}$$

In fact, formula III.1.2.1 was invented by authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, pp. 287-295). Table III.1.2.2 shows the CPT of node J , which is calculated by the same way to formula III.1.2.1. Recall that the formula III.1.2.1 is based on the theorem of SIGMA-gate inference which will be introduced in next sub-section III.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.1.2.2. CPT of node J

Let $w_4=1$ be weight of diagnostic relationship from hidden node O to evidence node E (Exercise: set up class “Human”), it is required to establish the CPT of E . It is impossible to specify CPT of E by formula III.1.2.1 because diagnostic relationship is different from aggregation relationship. We should research diagnostic relationship in detail. The conditional probability of E given O is $P(E|O)$. The posterior probability of O is $P(O|E)$, which is used to evaluate student’s mastery over concept (hypothesis) O given evidence E . The weight of arc from O to E is considered 1. Formula III.1.2.2 specifies CPT of E when E is binary:

$$P(E|O) = \begin{cases} E & \text{if } O = 1 \\ 1 - E & \text{if } O = 0 \end{cases}$$

Formula III.1.2.2. Conditional probability of E given O

E is equivalent to O and it can be totally used to diagnose student's mastery of O if $P(O|E)$ is proportional to $P(E|O)$ with a fixed coefficient. In fact, we have:

$$\begin{aligned}
 P(O|E) &= \frac{P(E|O)P(O)}{P(E)} = \frac{P(E|O)P(O)}{P(E|O=0)P(O=0) + P(E|O=1)P(O=1)} \\
 &\quad (\text{due to Bayes' rule}) \\
 &= \frac{P(E|O)P(O)}{P(O)(P(E|O=0) + P(E|O=1))} \\
 &\quad (\text{due to } P(O=0) = P(O=1) = 0.5) \\
 &= \frac{P(E|O)}{P(E|O=0) + P(E|O=1)} = \frac{P(E|O)}{1 - E + E} = 1 * P(E|O)
 \end{aligned}$$

Sub-section III.1.4 will discuss diagnostic relationship in detail, in which formula III.1.2.2 is proved. The CPT of E is totally determined as in table III.1.2.3.

$P(E=1 O=1) = 1 * w_4 = 1$	$P(E=0 O=1) = 0$
$P(E=1 O=0) = 0 * w_4 = 0$	$P(E=0 O=0) = 1$

Table III.1.2.3. CPT of evidence node E

Hence, the Java course overlay model is transformed totally into Bayesian network (Bayesian overlay model) as figure III.1.2.2:

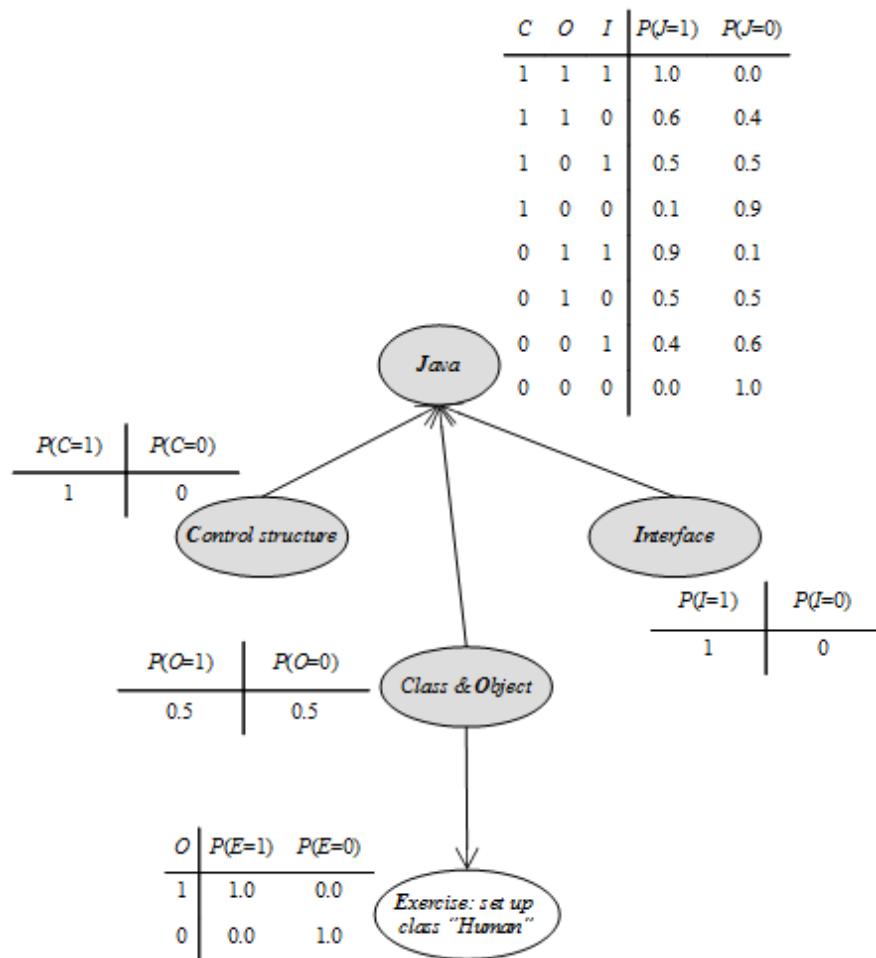


Figure III.1.2.2. Bayesian network (Bayesian overlay model) of Java course with full of CPT (s)

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](#)) inside Bayesian network.

Inferring user's knowledge

Suppose a leaner did well the exercise “Set up class *Human*”. That is to say the occurrence of an evidence, namely, $E=1$. It is necessary to answer the question: How mastered is learner over the concept “Java”? Thus, the posterior conditional probability $P(J=1 | E=1)$ of hidden variables J with fixed event $E=1$ must be computed. According to formula [III.1.1.10](#), we have:

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

Where $P(J, C, O, I, E)$ is global joint probability distribution. Applying formula [III.1.1.8](#), we have:

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

Applying all CPT (s) in table [III.1.2.1](#), [III.1.2.2](#), and [III.1.2.3](#), it is able to determined $P(J, C, O, I, E)$. After that, we compute $P(J=1 / E=1)$ to answer above question; please see section [III.2](#) to know how to calculate this posterior probability in detail.

The essence of combination between overlay model and BN is to convert relationships built in overlay model to CPT (s) of BN. We apply SIGMA-gate (formula [III.1.2.1](#)) into converting aggregation relationship to CPT but there is still an important one. That is prerequisite relationship. AND-gate and OR-gate which represent prerequisite relationship are mentioned (Carmona, Millán, Pérez-de-la-Cruz, Trella, & Conejo, 2005) and (Millán, Loboda, & Pérez-de-la-Cruz, Bayesian networks for student model engineering, 2010). Sub-section [III.1.4](#) will focus on how to convert relationships into CPT (s). In other words, relationship conversion is an extension of the combination method mentioned here.

Now the combination approach to construct Bayesian model is described basically in this sub-section [III.1.2](#). Next sub-section [III.1.3](#) is the theoretical proof for SIGMA-gate inference specified by formula [III.1.2.1](#), which is followed by the advanced sub-section [III.1.4](#).

III.1.3. SIGMA-gate inference

The main task of applying Bayesian network into overlay model mentioned in previous sub-section [III.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.1.2.1](#) specifying conditional probability of random variable is merely the learning parameter problem. I propose a theorem of SIGMA-gate inference (Nguyen L. , Theorem of SIGMA-gate Inference in Bayesian Network, 2016) in this sub-section [III.1.3](#) which is the base of formula [III.1.2.1](#), also the base of the combination of overlay model and Bayesian network mentioned in previous sub-section [III.1.2](#). Later on, we will understand that theorem of SIGMA-gate inference is only a simple and general case of works invented by authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002). I express my deep gratitude to authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002) for providing excellent works.

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 called aggregation graph in which child node is the aggregation of parent nodes; in other words, each arc expresses an aggregation relationship (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, p. 289). 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. Binary variables X_i and Y represent events. Following are some notes about probabilistic events:

- The assignment “ $X_i=1$ ” (“ $X_i=0$ ”) means event X_i does (does not) occur.
- The assignment “ $Y=1$ ” (“ $Y=0$ ”) means event Y does (does not) occur.
- The probability that X_i does occur is denoted $P(X_i=1)$. The probability that X_i does not occur is denoted $P(X_i=0)$.
- The probability that Y does occur is denoted $P(Y=1)$. The probability that Y does not occur is denoted $P(Y=0)$.

Figure III.1.3.1 depicts sigma graph.

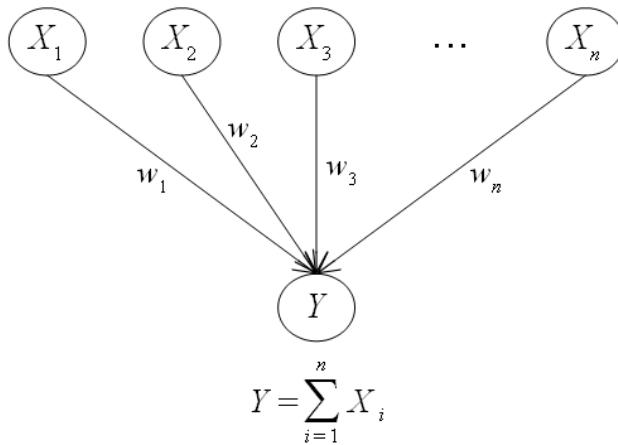


Figure III.1.3.1. Sigma graph

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.1.2.1 mentioned in previous sub-section III.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 (Neapolitan, 2003, p. 157):

- *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 .

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

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

- *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.1.3.1 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.1.3.1. 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.1.3.2 also depicts sigma condition.

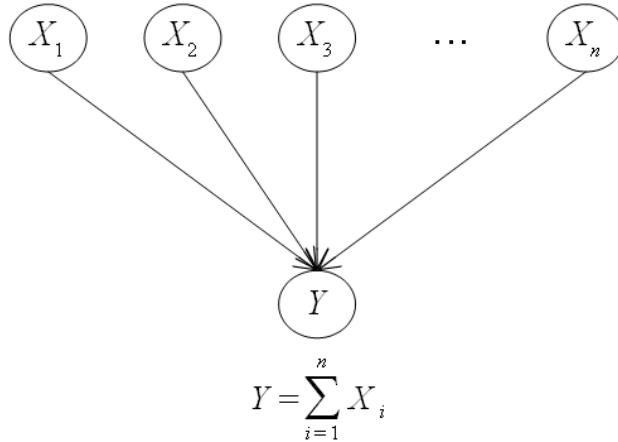


Figure III.1.3.2. Sigma condition

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 accountability variable so that A_i is *ON* ($=1$) if X_i is equal to 1 and I_i is *OFF* ($=0$), we have:

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

$$\begin{aligned} P(A_i = ON | X_i = 1, I_i = ON) &= 0 \\ P(A_i = ON | X_i = 0, I_i = OFF) &= 0 \\ P(A_i = ON | X_i = 0, I_i = ON) &= 0 \end{aligned}$$

$$\begin{aligned} P(A_i = OFF | X_i = 1, I_i = OFF) &= 0 \\ P(A_i = OFF | X_i = 1, I_i = ON) &= 1 \\ P(A_i = OFF | X_i = 0, I_i = OFF) &= 1 \\ P(A_i = OFF | X_i = 0, I_i = ON) &= 1 \end{aligned}$$

Binary variables A_i (s) and I_i (s) also represent probabilistic events. 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.1.3.2 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.1.3.2. Probability of sigma sum

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.1.3.3 depicts SIGMA-gate model with regard to accountability variables A_i (s).

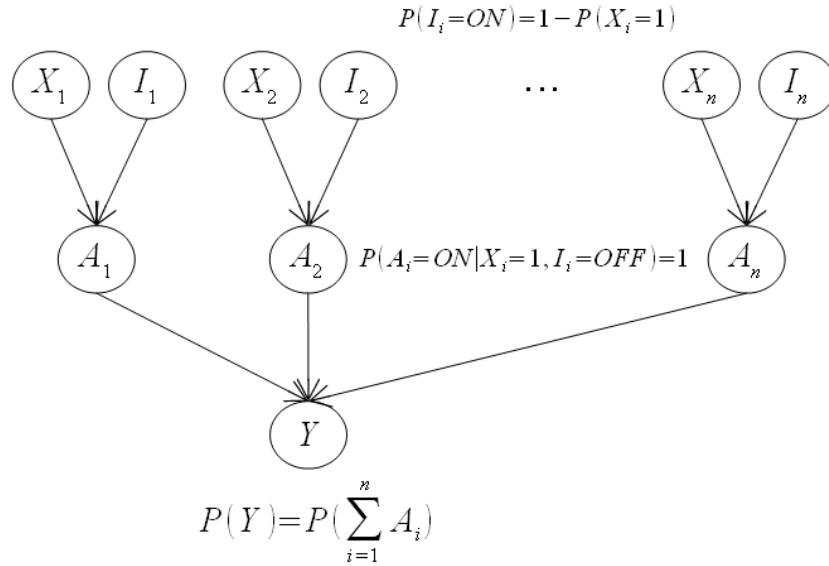


Figure III.1.3.3. SIGMA-gate model

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

$$\begin{aligned} P(I_i = ON) &= 1 - P(X_i = 1) = 1 - p_i \\ P(I_i = OFF) &= P(X_i = 1) = p_i \end{aligned}$$

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

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

Given the value set $\{X_1 = x_1, X_2 = x_2, \dots, X_n = x_n\}$, 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 $P(Y | X_1, X_2, \dots, X_n)$. We have:

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

(Due to SIGMA condition)

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

(Because A_i (s) are mutually independent)

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

(Because A_i is only dependent on X_i)

When sources $\{X_1, X_2, \dots, X_n\}$ and target Y receive values, we have:

$$P(Y = 1 | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$$

$$= \sum_{i=1}^n P(A_i = ON | X_i = x_i)$$

$$= \sum_{i=1}^n (P(A_i = ON | X_i = x_i, I_i = ON)P(I_i = ON) + P(A_i = ON | X_i = x_i, I_i = OFF)P(I_i = OFF))$$

Applying the total probability rule (see formula III.1.1.4) with regard to the set K of i (s) such that $x_i = 1$, we have:

$$P(Y = 1 | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$$

$$= \left(\sum_{i \in K} (P(A_i = ON | X_i = 1, I_i = ON)P(I_i = ON) + P(A_i = ON | X_i = 1, I_i = OFF)P(I_i = OFF)) \right)$$

$$+ \left(\sum_{i \notin K} (P(A_i = ON | X_i = 0, I_i = ON)P(I_i = ON) + P(A_i = ON | X_i = 0, I_i = OFF)P(I_i = OFF)) \right)$$

$$= \sum_{i \in K} (0 * (1 - p_i) + 1 * p_i) + \sum_{i \notin K} (0 * (1 - p_i) + 0 * p_i) = \sum_{i \in K} p_i$$

$$= \sum_{i \in K} P(X_i = 1) = \sum_{i=1}^n h_i P(X_i = 1)$$

Where,

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

Suppose target event Y does not occur ($Y=0$), we have:

$$P(Y = 0 | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$$

$$= 1 - P(Y = 1 | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$$

$$= 1 - \sum_{i \in K} P(X_i = 1)$$

Recall that the set $\{X_1, X_2, \dots, X_n\}$ is a partition of probability space, we have:

$$1 = \sum_{i=1}^n P(X_i = 1) = \sum_{i \in K} P(X_i = 1) + \sum_{i \notin K} P(X_i = 1)$$

It implies

$$\begin{aligned} P(Y = 0 | X_1 = x_1, X_2 = x_2, \dots, X_i = x_i, \dots, X_n = x_n) &= \sum_{i \notin K} P(X_i = 1) \\ &= \sum_{i=1}^n h_i P(X_i = 1) \end{aligned}$$

Where,

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

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:*

$$P(Y | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = \sum_{i=1}^n h_i P(X_i = 1)$$

Where,

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

Formula III.1.3.3. 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_n) is a partition of probability space. Essentially, formula III.1.2.1 and III.1.3.3 are the same to formulas specifying relationship between topics and subjects, invented by authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, pp. 292-295). However, formula III.1.3.3 is more general and I prove it by different way. In other words, theorem of SIGMA-gate is only a simple and general case of works that were invented by authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, 2002). I express my deep gratitude to authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, 2002) for providing excellent works.

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.1.3.3) 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.*

$$P(Y | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = \sum_{i=1}^n h_i w_i$$

Where,

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

Formula III.1.3.4. 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:

$$P(Y|X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = \sum_{i=1}^n h_i P(X_i = 1) = \sum_{i=1}^n h_i w_i$$

Where,

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

In general, the proof of SIGMA-gate inference is inspired from noisy OR-gate model (Neapolitan, 2003, p. 157) in Bayesian network. Authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, 2002) invented formula III.1.3.4 by different way. Given aggregation relationships shown as sigma graph in figure III.1.3.1, according to authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, 2002, p. 293), the conditional probability of $Y=1$ given random real variable F where $0 \leq f \leq 1$ is considered as variable F itself.

$$P(Y = 1|\{X_i: i \in K\}, F) = F$$

Where K is the set of i (s) such that $X_i=1$. In learning context, each X_i represents a concept and Y represents a topic that is composed of many concepts.

Authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, 2002, p. 292) defined that F has distribution according to formula III.1.3.5:

$$P(F|\{X_i: i \in K\}) = \begin{cases} 1 & \text{if } F = \sum_{i \in K} w_i \\ 0 & \text{otherwise} \end{cases}$$

Formula III.1.3.5. Mastery distribution

With regard to formula III.1.3.5, authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, 2002, p. 292) claimed that if the total grade of given student is equal to the normalized sum of concept masteries, $F = \sum_{i \in K} w_i$ then, it is 100% confident that such student mastered these concepts, $P(F|\{X_i: i \in K\}) = 1$.

The probability of $Y=1$ is expectation of F due to total probability law as follows:

$$\begin{aligned} E(F|\{X_i: i \in K\}) &= \int_F P(F|\{X_i: i \in K\}) F dF \\ &= \int_F P(F|\{X_i: i \in K\}) P(Y = 1|\{X_i: i \in K\}, F) dF \\ &= P(Y = 1|\{X_i: i \in K\}) \end{aligned}$$

Authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, 2002, p. 293) determined formula III.1.3.4 by calculating the expectation of F as follows:

$$P(Y = 1|\{X_i: i \in K\}) = E(F|\{X_i: i \in K\}) = 1 * \sum_{i \in K} w_i + 0 = \sum_{i \in K} w_i$$

According to formula above, if student masters some concepts $X_i (=1)$, the probability that she/he masters topic Y is equal to sum of weights of these concepts.

In next sub-section III.3.1, it is assumed that F has beta distribution and so the work of authors Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002) is very significant pioneer invention. In overlay model, each concept node is represented by an arbitrary real variable and so such work helps us to convert real variable to binary variable which is necessary to combine overlay model and BN (Millán & Pérez-de-la-Cruz, 2002, p. 293) in case of aggregation relationship.

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.1.3.4 depicts given weighted sigma graph.

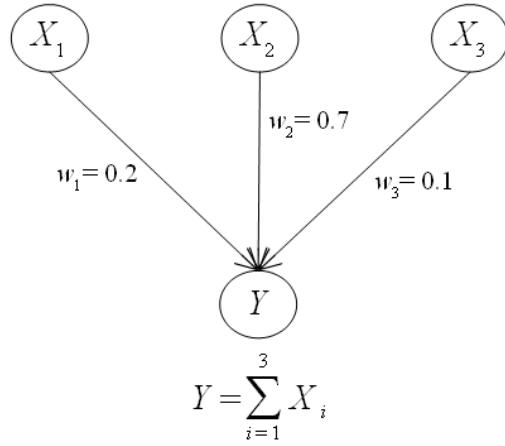


Figure III.1.3.4. An example of sigma graph

Applying theorem of SIGMA-gate inference specified in formula III.1.3.4, 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:

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

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

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

$$P(Y_1=0 | X_1=0, X_2=1, X_3=0) = 1*0.2 + 0*0.7 + 1*0.1 = 0.3$$

$$P(Y_1=0 | X_1=0, X_2=0, X_3=1) = 1*0.2 + 1*0.7 + 0*0.1 = 0.9$$

$$P(Y_1=0 | X_1=0, X_2=0, X_3=0) = 1*0.2 + 1*0.7 + 1*0.1 = 1.0$$

Figure III.1.3.5 depicts Bayesian network transformed from weighted sigma graph.

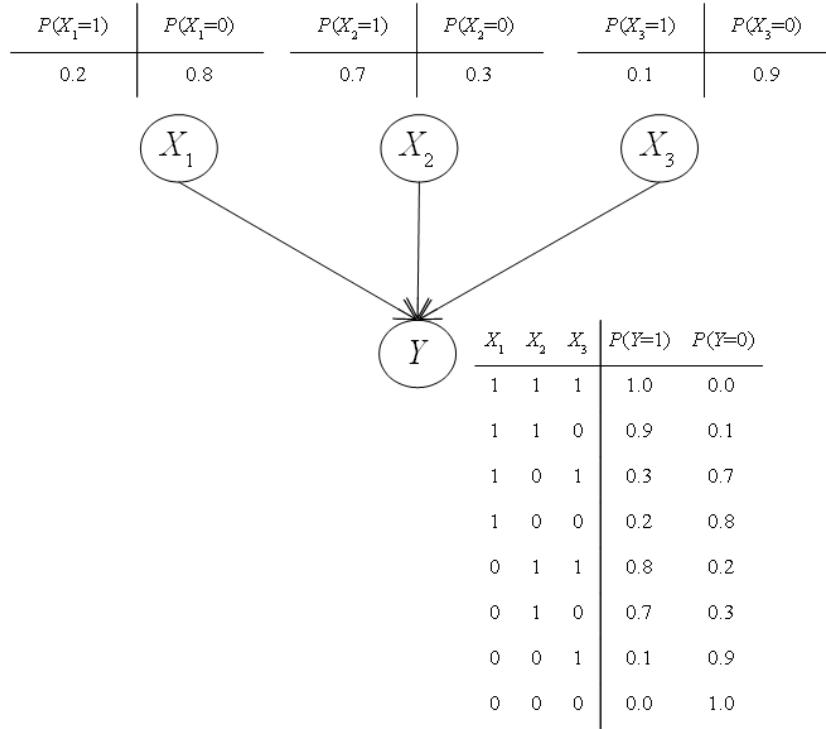


Figure III.1.3.5. Bayesian network transformed from sigma graph

The example of transforming sigma graph into Bayesian network ends up this sub-section III.1.3 with full of explanations of SIGMA-gate theorem. The successive sub-section III.1.4 discusses relationship conversion that is an extension of the combination method mentioned basically in sub-section III.1.2.

III.1.4. Relationship conversion in Bayesian network

The essence of combination of BN and overlay model is to convert relationships defined in overlay model into CPT (s) of BN. It is impossible to convert all overlay relationships but some of them such as diagnostic, aggregation, and prerequisite are mandatory ones that we must specify as computable CPT (s) of BN. These relationships are adhered to X-gates such as AND-gate, OR-gate, and SIGMA-gate. This sub-section III.1.4 focuses on X-gate relationships and diagnostic relationship.

Essentially, relationship conversion aims to determine conditional probabilities based on weights and meanings of relationships. We will have different ways to convert graphic weights into CPT (s) for different relationships. It is impossible to convert all relationships but some of them the such as diagnostic, aggregation, and prerequisite are mandatory ones that we must specify as computable CPT (s) of BN. Especially, these relationships are adhered to logic X-gates (Wikipedia, 2016) such as AND-gate, OR-gate, and SIGMA-gate. The X-gate inference in this research is derived and inspired from noisy OR-gate described in the book “Learning Bayesian Networks” by Neapolitan (Neapolitan, 2003, pp. 157-159). Díez and Druzdzel (Díez

& Druzdzel, 2007) also researched OR/MAX, AND/MIN, noisy XOR inferences but they focused on canonical models, deterministic models, and ICI models whereas I focus on logic gate and graphic relationships. So their research is different from mine but we share the same result that is AND-gate model. In general, my research focuses on applied probability adhered to Bayesian network, logic gates, and Bayesian user modeling (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002). The scientific results are shared with Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002).

Factor graph (Wikipedia, Factor graph, 2015) represents factorization of a global function into many partial functions. If joint distribution of BN is considered as the global function and CPT (s) are considered as partial functions, the sum-product algorithm (Kschischang, Frey, & Loeliger, 2001) of factor graph is applied into calculating posterior probabilities of variables in BN. Pearl's propagation algorithm (Pearl, 1986) is very successful in BN inference. The application of factor graph into BN is only realized if all CPT (s) of BN are already determined whereas this research focuses on defining such CPT (s) firstly. I did not use factor graph for constructing BN. The concept "X-gate inference" only implies how to convert simple graph into BN. However the arrange sum with a fixed variable mentioned in this research is the "*not-sum*" (Kschischang, Frey, & Loeliger, 2001, p. 499) of factor graph. Essentially, X-gate probability shown in formula 3.4 is as same as λ message in the Pearl's algorithm (Kschischang, Frey, & Loeliger, 2001, p. 518) but I use the most basic way to prove the X-gate probability.

As default, the research is applied in learning context in which BN is used to assess students' knowledge. Evidences are tests, exams, exercises, etc. and hypotheses are learning concepts, knowledge items, etc. Note that diagnostic relationship is very important to Bayesian evaluation in learning context because it is used to evaluate student's mastery of concepts (knowledge items) over entire BN. As a convention, all equations are called formulas having particular titles. Now we start relationship conversion with a research on diagnostic relationship in the next sub-section [III.1.4.1](#).

III.1.4.1. Diagnostic relationship

In some opinions like mine, the diagnostic relationship should be from hypothesis to evidence. For example, disease is hypothesis and symptom is evidence. The symptom must be conditionally dependent on disease. Given a symptom, calculating the posterior probability of disease is essentially to diagnose likelihood of such disease (Millán, Loboda, & Pérez-de-la-Cruz, Bayesian networks for student model engineering, 2010, p. 1666). Inversely, the arc from evidence to hypothesis implies prediction where evidence and hypothesis represent observation and event, respectively. Given an observation, calculating the posterior probability of the event is essentially to predict/assert such event (Millán, Loboda, & Pérez-de-la-Cruz, Bayesian networks for student model engineering, 2010, p. 1666). Figure [III.1.4.1.1](#) shows diagnosis and prediction.

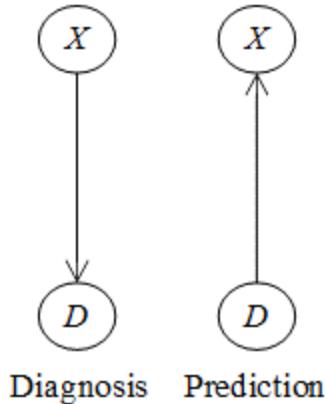


Figure III.1.4.1.1. Diagnosis and prediction with hypothesis X and evidence D

The weight w of relationship between X and D is 1. Figure III.1.4.1.1 depicts simplest graph with two random variables. We need to convert diagnostic relationship into conditional probabilities in order to construct a simplest BN from the simplest graph. Note that hypothesis is binary but evidence can be numerical. In learning context, evidence D can be test, exam, exercise, etc. The conditional probability of D given X (likelihood function) is $P(D|X)$. The posterior probability of X is $P(X|D)$, which is used to evaluate student's mastery over concept (hypothesis) X given evidence D . Formula III.1.4.1.1 specifies CPT of D when D is binary (0 and 1):

$$P(D|X) = \begin{cases} D & \text{if } X = 1 \\ 1 - D & \text{if } X = 0 \end{cases}$$

Formula III.1.4.1.1. CPT of binary evidence of diagnostic relationship

Formula III.1.4.1.1 is our first relationship conversion in this sub-section III.1.4. It implies

$$P(D|X = 0) + P(D|X = 1) = D + 1 - D = 1$$

Evidence D can be used to diagnose hypothesis X if the so-called *sufficient diagnostic proposition* is satisfied, as seen in table III.1.4.1.1.

D is equivalent to X in diagnostic relationship if $P(X|D) = kP(D|X)$ given uniform distribution of X and the *transformation coefficient* k is independent from D . In other words, k is constant with regards to D and so D is called *sufficient evidence*.

Table III.1.4.1.1. Sufficient diagnostic proposition

The concept of sufficient evidence is borrowed from the concept of sufficient statistics and it is inspired from equivalence of variables T and T' in the research (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, pp. 292-295). The proposition can be restated that evidence D is only used to assess hypotheses if it is sufficient evidence. As a convention, the proposition is called *diagnostic condition* and hypotheses have uniform distribution. The assumption of hypothetic uniform distribution ($P(X=1) = P(X=0)$) implies that we cannot assert whether or not given hypothesis is true before we observe its evidence.

In learning context, D can be totally used to assess student's mastery of X if diagnostic condition is satisfied. Derived from such condition, formula III.1.4.1.2 specifies transformation coefficient k given uniform distribution of X .

$$k = \frac{P(X|D)}{P(D|X)}$$

Formula III.1.4.1.2. Transformation coefficient given uniform distribution of X

We need to prove that formula III.1.4.1.1 satisfies diagnostic condition. Suppose the prior probability of X is uniform:

$$P(X = 0) = P(X = 1)$$

We have:

$$\begin{aligned} P(X|D) &= \frac{P(D|X)P(X)}{P(D)} = \frac{P(D|X)P(X)}{P(D|X=0)P(X=0) + P(D|X=1)P(X=1)} \\ &\quad (\text{due to Bayes' rule}) \\ &= \frac{P(D|X)P(X)}{P(X)(P(D|X=0) + P(D|X=1))} \\ &\quad (\text{due to } P(X=0) = P(X=1)) \\ &= \frac{P(D|X)}{P(D|X=0) + P(D|X=1)} = 1 * P(D|X) \\ &\quad (\text{due to } P(D|X=0) + P(D|X=1) = 1) \end{aligned}$$

It is easy to infer that the transformation coefficient k is 1 if D is binary. In practice, evidence D is often a test whose grade ranges within an interval $\{0, 1, 2, \dots, \eta\}$. Formula III.1.4.1.3 specifies CPT of D in this case:

$$P(D|X) = \begin{cases} \frac{D}{S} & \text{if } X = 1 \\ \frac{\eta}{S} - \frac{D}{S} & \text{if } X = 0 \end{cases}$$

Where,

$$\begin{aligned} D &\in \{0, 1, 2, \dots, \eta\} \\ S &= \sum_{D=0}^n D = \frac{\eta(\eta+1)}{2} \end{aligned}$$

Formula III.1.4.1.3. CPT of integer evidence ranging in $\{0, 1, 2, \dots, \eta\}$ of diagnostic relationship

As a convention, $P(D|X) = 0, \forall D \notin \{0, 1, 2, \dots, \eta\}$. Formula III.1.4.1.3 implies that if student has mastered concept ($X=1$), the probability that she/he completes the exercise/test D is proportional to her/his mark on D ($P(D|X) = \frac{D}{S}$). We also have:

$$\begin{aligned} P(D|X=0) + P(D|X=1) &= \frac{D}{S} + \frac{\eta-D}{S} = \frac{\eta}{S} = \frac{2}{(\eta+1)} \\ \sum_{D=0}^{\eta} P(D|X=1) &= \sum_{D=0}^{\eta} \frac{D}{S} = \frac{\sum_{D=0}^{\eta} D}{S} = \frac{S}{S} = 1 \\ \sum_{D=0}^{\eta} P(D|X=0) &= \sum_{D=0}^{\eta} \frac{\eta-D}{S} = \frac{\sum_{D=0}^{\eta} (\eta-D)}{S} = \frac{\sum_{D=0}^{\eta} \eta - \sum_{D=0}^{\eta} D}{S} = \frac{\eta(\eta+1) - S}{S} \\ &= \frac{2S-S}{S} = 1 \end{aligned}$$

We need to prove that formula III.1.4.1.3 satisfies diagnostic condition. Suppose the prior probability of X is uniform:

$$P(X = 0) = P(X = 1)$$

The assumption of prior uniform distribution of X implies that we do not determine if student has mastered X yet. Similarly, we have:

$$P(X|D) = \frac{P(D|X)P(X)}{P(D)} = \frac{P(D|X)}{P(D|X = 0) + P(D|X = 1)} = \frac{\eta + 1}{2} P(D|X)$$

So, the transformation coefficient k is $\frac{\eta+1}{2}$ if D ranges in $\{0, 1, 2, \dots, \eta\}$.

In the most general case, discrete evidence D ranges within an arbitrary integer interval $\{a, a + 1, a + 2, \dots, b\}$. In other words, D is bounded integer variable whose lower bound and upper bound are a and b , respectively. Formula III.1.4.1.4 specifies CPT of D where $D \in \{a, a + 1, a + 2, \dots, b\}$.

$$P(D|X) = \begin{cases} \frac{D}{S} & \text{if } X = 1 \\ \frac{b+a}{S} - \frac{D}{S} & \text{if } X = 0 \end{cases}$$

Where,

$$D \in \{a, a + 1, a + 2, \dots, b\}$$

$$S = a + (a + 1) + (a + 2) + \dots + b = \frac{(b + a)(b - a + 1)}{2}$$

Formula III.1.4.1.4. CPT of general discrete evidence of diagnostic relationship

Note, $P(D|X) = 0, \forall D \notin \{a, a + 1, a + 2, \dots, b\}$. According to the diagnostic condition, we need to prove the equality $P(X|D) = kP(D|X)$ where

$$k = \frac{b - a + 1}{2}$$

Similarly, we have:

$$P(X|D) = \frac{P(D|X)P(X)}{P(D)} = \frac{P(D|X)}{P(D|X = 0) + P(D|X = 1)} = \frac{b - a + 1}{2} P(D|X)$$

If evidence D is continuous in the real interval $[a, b]$ with note that a and b are real numbers, formula III.1.4.1.5 specifies probability density function (PDF) of continuous evidence $D \in [a, b]$. The PDF $p(D|X)$ replaces CPT in case of continuous random variable.

$$p(D|X) = \begin{cases} \frac{2D}{b^2 - a^2} & \text{if } X = 1 \\ \frac{2}{b - a} - \frac{2D}{b^2 - a^2} & \text{if } X = 0 \end{cases}$$

Where,

$$D \in [a, b] \text{ where } a \text{ and } b \text{ are real numbers}$$

$$S = \int_a^b D dD = \frac{b^2 - a^2}{2}$$

Formula III.1.4.1.5. CPT of continuous evidence of diagnostic relationship

As a convention, $[a, b]$ is called domain of continuous evidence, which can be replaced by open or half-open intervals such as (a, b) , $[a, b]$, and $(a, b]$. Of course we have $p(D|X) = 0, \forall D \notin [a, b]$. In learning context, evidence D is often a test whose grade ranges within real interval $[a, b]$.

Functions $p(D|X = 1)$ and $p(D|X = 0)$ are valid PDF (s) due to:

$$\int_D p(D|X=1)dD = \int_a^b \frac{2D}{b^2 - a^2} dD = \frac{1}{b^2 - a^2} \int_a^b 2D dD = 1$$

$$\int_D p(D|X=0)dD = \frac{2}{b-a} \int_a^b dD - \frac{1}{b^2 - a^2} \int_a^b 2D dD = 1$$

According to the diagnostic condition, we need to prove the equality

$$P(X|D) = kp(D|X)$$

Where,

$$k = \frac{b-a}{2}$$

When D is continuous, its probability is calculated in ε -vicinity where ε is very small number. As usual, ε is bias if D is measure values produced from equipment. The probability of D given X where $D+\varepsilon \in [a, b]$ and $D-\varepsilon \in [a, b]$ is:

$$P(D|X) = \int_{D-\varepsilon}^{D+\varepsilon} p(D|X)dD = \begin{cases} \int_{D-\varepsilon}^{D+\varepsilon} \frac{2D}{b^2 - a^2} dD & \text{if } X = 1 \\ \int_{D-\varepsilon}^{D+\varepsilon} \left(\frac{2}{b-a} - \frac{2D}{b^2 - a^2} \right) dD & \text{if } X = 0 \end{cases}$$

$$= \begin{cases} \frac{4\varepsilon D}{b^2 - a^2} & \text{if } X = 1 \\ \frac{4\varepsilon}{b-a} - \frac{4\varepsilon D}{b^2 - a^2} & \text{if } X = 0 \end{cases} = 2\varepsilon p(D|X)$$

In fact, we have:

$$P(X|D) = \frac{P(D|X)P(X)}{P(D|X=0)P(X=0) + P(D|X=1)P(X=1)}$$

(due to Bayes' rule)

$$= \frac{P(D|X)}{P(D|X=0) + P(D|X=1)}$$

(due to assumption $P(X=0) = P(X=1)$)

$$= \frac{b-a}{4\varepsilon} P(D|X) = kp(D|X)$$

In general, formula III.1.4.1.6 summarizes CPT of evidence of single diagnosis relationship.

$$P(D|X) = \begin{cases} \frac{D}{S} & \text{if } X = 1 \\ \frac{M}{S} - \frac{D}{S} & \text{if } X = 0 \end{cases}$$

$$k = \frac{N}{2}$$

Where,

$$N = \begin{cases} 2 & \text{if } D \in \{0,1\} \\ \eta + 1 & \text{if } D \in \{0,1,2, \dots, \eta\} \\ b - a + 1 & \text{if } D \in \{a, a+1, a+2, \dots, b\} \\ b - a & \text{if } D \text{ continuous and } D \in [a, b] \end{cases}$$

$$M = \begin{cases} 1 & \text{if } D \in \{0,1\} \\ \eta & \text{if } D \in \{0,1,2, \dots, \eta\} \\ b + a & \text{if } D \in \{a, a + 1, a + 2, \dots, b\} \\ b + a & \text{if } D \text{ continuous and } D \in [a, b] \end{cases}$$

$$S = \sum_D D = \frac{NM}{2} = \begin{cases} 1 & \text{if } D \in \{0,1\} \\ \frac{\eta(\eta + 1)}{2} & \text{if } D \in \{0,1,2, \dots, \eta\} \\ \frac{(b + a)(b - a + 1)}{2} & \text{if } D \in \{a, a + 1, a + 2, \dots, b\} \\ \frac{b^2 - a^2}{2} & \text{if } D \text{ continuous and } D \in [a, b] \end{cases}$$

Formula III.1.4.1.6. CPT of evidence of diagnostic relationship

In general, if the conditional probability $P(D/X)$ is specified by formula III.1.4.1.6, the diagnostic condition will be satisfied. Note that the CPT $P(D/X)$ is the PDF $p(D/X)$ in case of continuous evidence. The diagnostic relationship will be extended with more than one hypothesis. The next sub-section III.1.4.2 will mention how to determine CPT (s) of a simple graph with one child node and many parent nodes based on X-gate inferences.

III.1.4.2. X-gate inferences

Given a simple graph consisting of one child variable Y and n parent variables X_i , as shown in figure III.1.4.2.1. Each relationship from X_i to Y is quantified by a normalized weight w_i where $0 \leq w_i \leq 1$. A large graph is an integration of many simple graphs. Figure III.1.4.2.1 shows the DAG of a simple BN. As aforementioned, the essence of constructing simple BN is to convert graphic relationships of simple graph into CPT (s) of simple BN.

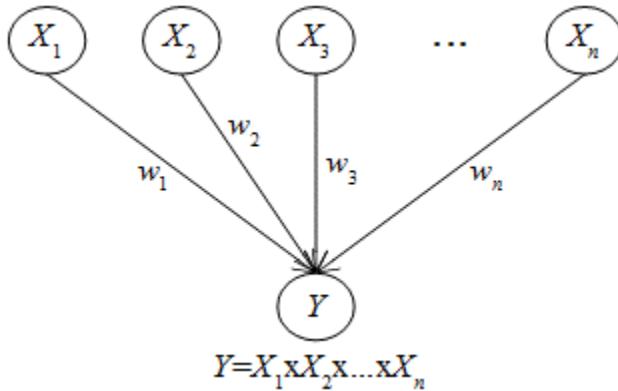


Figure III.1.4.2.1. Simple graph or simple network

Child variable Y is called target and parent variables X_i (s) are called sources. Especially, these relationships are adhere to X-gates such as AND-gate, OR-gate, and SIGMA-gate. These gates are originated from logic gate (Wikipedia, 2016). For instance, AND-gate and OR-gate represent prerequisite relationship. SIGMA-gate represents aggregation relationship. Therefore, relationship conversion is to determined X-gate inference. The simple graph shown in figure III.1.4.2.1 is also called X-gate graph or X-gate network. Please distinguish the letter "X" in the term

“X-gate inference” which implies logic operators (AND, OR, XOR, etc.) from the “variable X”.

All variables are binary and they represent events. The probability $P(X)$ indicates event X occurs. Thus, $P(X)$ implicates $P(X=1)$ and $P(\text{not}(X))$ implicates $P(X=0)$. Formula III.1.4.2.1 specifies the simple NOT-gate inference.

$$P(\text{not}(X)) = P(\bar{X}) = P(X = 0) = 1 - P(X = 1) = 1 - P(X)$$

$$P(\text{not}(\text{not}(X))) = P(X)$$

Formula III.1.4.2.1. NOT-gate inference

X-gate inference is based on three assumptions mentioned in (Neapolitan, 2003, p. 157):

- *X-gate inhibition*: Given a relationship from source X_i to target Y , there is a factor I_i that inhibits X_i from integrated into Y . Factor I_i is called inhibition of X_i . That the inhibition I_i is turned off is prerequisite of X_i integrated into Y .
- *Inhibition independence*: Inhibitions are mutually independent. For example, inhibition I_1 of X_1 is independent from inhibition I_2 of X_2 .
- *Accountability*: X-gate network is established by accountable variables A_i for X_i and I_i . Each X-gate inference owns particular combination of A_i (s).

Figure III.1.4.2.2 shows the extended X-gate network with accountable variables A_i (s) (Neapolitan, 2003, p. 158).

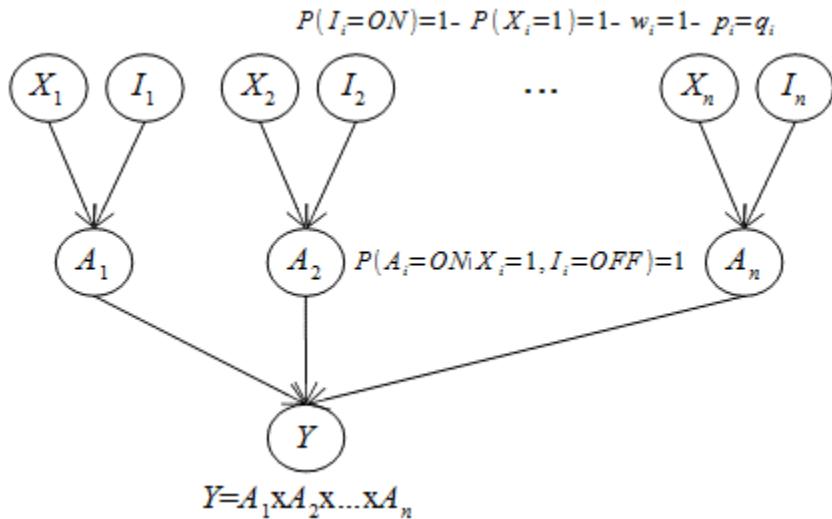


Figure III.1.4.2.2. Extended X-gate network with accountable variables A_i (s)

The strength of each relationship from source X_i to target Y is quantified by a weight $0 \leq w_i \leq 1$. According to the assumption of inhibition, probability of $I_i=OFF$ is p_i , which is set to be the weight w_i .

$$p_i = w_i$$

If notation w_i is used, we focus on strength of relationship. If notation p_i is used, we focus on probability of *OFF* inhibition. In probabilistic inference, p_i is also prior probability of $X_i=1$. However, we will assume each X_i has uniform distribution later on. Formula III.1.4.2.2 specifies probabilities of inhibitions I_i (s) and accountable variables A_i (s).

$$P(I_i = OFF) = p_i = w_i$$

$$\begin{aligned}
 P(I_i = ON) &= 1 - p_i = 1 - w_i \\
 P(A_i = ON | X_i = 1, I_i = OFF) &= 1 \\
 P(A_i = ON | X_i = 1, I_i = ON) &= 0 \\
 P(A_i = ON | X_i = 0, I_i = OFF) &= 0 \\
 P(A_i = ON | X_i = 0, I_i = ON) &= 0 \\
 P(A_i = OFF | X_i = 1, I_i = OFF) &= 0 \\
 P(A_i = OFF | X_i = 1, I_i = ON) &= 1 \\
 P(A_i = OFF | X_i = 0, I_i = OFF) &= 1 \\
 P(A_i = OFF | X_i = 0, I_i = ON) &= 1
 \end{aligned}$$

Formula III.1.4.2.2. Probabilities of inhibitions I_i (s) and accountable variables A_i (s)

According to formula III.1.4.2.2, given probability $P(A_i = ON | X_i = 1, I_i = OFF)$, it is assured 100% confident that accountable variables A_i is turned on if source X_i is 1 and inhibition I_i is turned off. Formula III.1.4.2.3 specifies conditional probability of accountable variables A_i (s) given X_i (s), which is corollary of formula III.1.4.2.2.

$$\begin{aligned}
 P(A_i = ON | X_i = 1) &= p_i = w_i \\
 P(A_i = ON | X_i = 0) &= 0 \\
 P(A_i = OFF | X_i = 1) &= 1 - p_i = 1 - w_i \\
 P(A_i = OFF | X_i = 0) &= 1
 \end{aligned}$$

Formula III.1.4.2.3. Conditional probability of accountable variables

Following is proof of formula 3.3.

$$\begin{aligned}
 P(A_i = ON | X_i) &= P(A_i = ON | X_i, I_i = ON)P(I_i = ON) + P(A_i = ON | X_i, I_i = OFF)P(I_i = OFF) \\
 &= 0 * (1 - p_i) + P(A_i = ON | X_i, I_i = OFF)p_i \\
 &\quad \text{(By applying formula III.1.4.2.2)} \\
 &= p_i P(A_i = ON | X_i, I_i = OFF)
 \end{aligned}$$

It implies

$$\begin{aligned}
 P(A_i = ON | X_i = 1) &= p_i P(A_i = ON | X_i = 1, I_i = OFF) = p_i \\
 P(A_i = ON | X_i = 0) &= p_i P(A_i = ON | X_i = 0, I_i = OFF) = 0 \\
 P(A_i = OFF | X_i = 1) &= 1 - P(A_i = ON | X_i = 1) = 1 - p_i \\
 P(A_i = OFF | X_i = 0) &= 1 - P(A_i = ON | X_i = 0) = 1
 \end{aligned}$$

As a definition, the set of all X_i (s) is complete if and only if

$$P(X_1 \cup X_2 \cup \dots \cup X_n) = P(\Omega) = \sum_{i=1}^n w_i = 1$$

The set of all X_i (s) is mutually exclusive if and only if

$$X_i \cap X_j = \emptyset, \forall i \neq j$$

For each X_i there is only one A_i and vice versa, which establishes a bijection between X_i (s) and A_i (s). Obviously, the fact that the set of all X_i (s) is complete is equivalent to the fact that the set of all A_i (s) is complete. We will prove by contradiction that “the fact that the set of all X_i (s) is mutually exclusive is equivalent to the fact that the set of all A_i (s) is mutually exclusive”. Suppose $X_i \cap X_j = \emptyset, \forall i \neq j$ but $\exists i \neq j: A_i \cap A_j = B \neq \emptyset$. Let $B^{-1} \neq \emptyset$ be preimage of B . Due to $B \subseteq A_i$ and $B \subseteq A_j$, we have $B^{-1} \subseteq X_i$ and $B^{-1} \subseteq X_j$, which causes that $X_i \cap X_j = B^{-1} \neq \emptyset$. There is a contradiction and so we have:

$$X_i \cap X_j = \emptyset, \forall i \neq j \Rightarrow A_i \cap A_j = \emptyset, \forall i \neq j$$

By similar proof, we have:

$$A_i \cap A_j = \emptyset, \forall i \neq j \Rightarrow X_i \cap X_j = \emptyset, \forall i \neq j$$

The extended X-gate network shown in figure III.1.4.2.2 is interpretation of simple network shown in figure III.1.4.2.1. Specifying CPT of the simple network is to determine the conditional probability $P(Y=1 | X_1, X_2, \dots, X_n)$ based on extended X-gate network. The X-gate inference is represented by such probability $P(Y=1 | X_1, X_2, \dots, X_n)$ specified by formula III.1.4.2.4 (Neapolitan, 2003, p. 159).

$$P(Y|X_1, X_2, \dots, X_n) = \sum_{A_1, A_2, \dots, A_n} P(Y|A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i|X_i)$$

Formula III.1.4.2.4. X-gate probability

Following is the proof of formula III.1.4.2.4.

$$\begin{aligned} P(Y|X_1, X_2, \dots, X_n) &= \frac{P(Y, X_1, X_2, \dots, X_n)}{P(X_1, X_2, \dots, X_n)} \\ &= \frac{\sum_{A_1, A_2, \dots, A_n} P(Y, X_1, X_2, \dots, X_n | A_1, A_2, \dots, A_n) * P(A_1, A_2, \dots, A_n)}{P(X_1, X_2, \dots, X_n)} \\ &\quad (\text{Due to total probability rule}) \\ &= \sum_{A_1, A_2, \dots, A_n} P(Y, X_1, X_2, \dots, X_n | A_1, A_2, \dots, A_n) * \frac{P(A_1, A_2, \dots, A_n)}{P(X_1, X_2, \dots, X_n)} \\ &= \sum_{A_1, A_2, \dots, A_n} P(Y | A_1, A_2, \dots, A_n) * P(X_1, X_2, \dots, X_n | A_1, A_2, \dots, A_n) * \frac{P(A_1, A_2, \dots, A_n)}{P(X_1, X_2, \dots, X_n)} \\ &\quad (\text{Because } Y \text{ is conditionally independent from } X_i \text{ (s) given } A_i \text{ (s)}) \\ &= \sum_{A_1, A_2, \dots, A_n} P(Y | A_1, A_2, \dots, A_n) * \frac{P(X_1, X_2, \dots, X_n, A_1, A_2, \dots, A_n)}{P(X_1, X_2, \dots, X_n)} \\ &= \sum_{A_1, A_2, \dots, A_n} P(Y | A_1, A_2, \dots, A_n) * P(A_1, A_2, \dots, A_n | X_1, X_2, \dots, X_n) \\ &\quad (\text{Due to Bayes' rule}) \\ &= \sum_{A_1, A_2, \dots, A_n} P(Y | A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i | X_1, X_2, \dots, X_n) \\ &\quad (\text{Because } A_i \text{ (s) are mutually independent}) \\ &= \sum_{A_1, A_2, \dots, A_n} P(Y | A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i | X_i) \\ &\quad (\text{Because each } A_i \text{ is only dependent on } X_i) \end{aligned}$$

It is necessary to make some mathematical notations because formula III.1.4.2.4 is complicated, which is relevant to arrangements of X_i (s). Given the set $\Omega = \{X_1, X_2, \dots, X_n\}$ where all variables are binary, table III.1.4.2.1 specifies binary arrangements of Ω .

Given $\Omega = \{X_1, X_2, \dots, X_n\}$ where $|\Omega|=n$ is cardinality of Ω .

Let $a(\Omega)$ be an arrangement of Ω , which is a set of n instances $\{X_1=x_1, X_2=x_2, \dots, X_n=x_n\}$ where x_i is 1 or 0. The number of all $a(\Omega)$ is $2^{|\Omega|}$. For instance, given $\Omega = \{X_1, X_2\}$, there are $2^2=4$ arrangements as follows:

$a(\Omega) = \{X_1=1, X_2=1\}$, $a(\Omega) = \{X_1=1, X_2=0\}$, $a(\Omega) = \{X_1=0, X_2=1\}$, $a(\Omega) = \{X_1=0, X_2=0\}$.

Let $a(\Omega: \{X_i\})$ be the arrangement of Ω with fixed X_i . The number of all $a(\Omega: \{X_i\})$ is $2^{|\Omega|-1}$. Similarly, for instance, $a(\Omega: \{X_1, X_2, X_3\})$ is an arrangement of Ω with fixed X_1, X_2, X_3 . The number of all $a(\Omega: \{X_1, X_2, X_3\})$ is $2^{|\Omega|-3}$.

Let $c(\Omega)$ and $c(\Omega: \{X_i\})$ be the number of arrangements $a(\Omega)$ and $a(\Omega: \{X_i\})$, respectively. Such $c(\Omega)$ and $c(\Omega: \{X_i\})$ are called arrangement counters. As usual, counters $c(\Omega)$ and $c(\Omega: \{X_i\})$ are equal to $2^{|\Omega|}$ and $2^{|\Omega|-1}$, respectively but they will vary according to specific cases.

Let $\sum_a F(a(\Omega))$ and $\prod_a F(a(\Omega))$ denote sum and product of values generated from function F acting on every $a(\Omega)$. The number of arrangements on which F acts is $c(\Omega)$.

Let x denote the X-gate operator, for instance, $x = \odot$ for AND-gate, $x = \oplus$ for OR-gate, $x = \text{not}\odot$ for NAND-gate, $x = \text{not}\oplus$ for NOR-gate, $x = \otimes$ for XOR-gate, $x = \text{not}\otimes$ for XNOR-gate, $x = \uplus$ for U-gate, $x = +$ for SIGMA-gate. Given an x -operator, let $s(\Omega: \{X_i\})$ and $s(\Omega)$ be sum of all $P(X_1xX_2x \dots xX_n)$ through every arrangement of Ω with and without fixed X_i , respectively.

$$s(\Omega) = \sum_a P(X_1xX_2x \dots xX_n | a(\Omega)) = \sum_a P(Y = 1 | a(\Omega))$$

$$s(\Omega: \{X_i\}) = \sum_a P(X_1xX_2x \dots xX_n | a(\Omega: \{X_i\})) = \sum_a P(Y = 1 | a(\Omega: \{X_i\}))$$

For example, $s(\Omega)$ and $s(\Omega: \{X_i\})$ for OR-gate are:

$$s(\Omega) = \sum_a P(X_1 \oplus X_2 \oplus \dots \oplus X_n | a(\Omega))$$

$$s(\Omega: \{X_i\}) = \sum_a P(X_1 \oplus X_2 \oplus \dots \oplus X_n | a(\Omega: \{X_i\}))$$

Such $s(\Omega)$ and $s(\Omega: \{X_i\})$ are called arrangement sum. They are acting function F .

Note that Ω can be any set of binary variables.

Table III.1.4.2.1. Binary arrangements

It is not easy to produce all binary arrangements of Ω . Table III.1.4.2.2 shows a code snippet written by Java programming language for producing such all arrangements.

```
public class ArrangementGenerator {
    private ArrayList<int[]> arrangements;
    private int n;
    private int r;

    private ArrangementGenerator(int n, int r) {
        this.n = n;
        this.r = r;
        this.arrangements = new ArrayList();
    }

    private void create(int[] a, int i) {
```

```

        for(int j = 0; j < n; j++) {
            a[i] = j;
            if(i < r - 1)
                create(a, i + 1);
            else if(i == r - 1) {
                int[] b = new int[a.length];
                for(int k = 0; k < a.length; k++) b[k] = a[k];
                arrangements.add(b);
            }
        }
    }

    public int[] get(int i) {
        return arrangements.get(i);
    }

    public long size() {
        return arrangements.size();
    }

    public static ArrangementGenerator parse(int n, int r) {
        ArrangementGenerator arr =
            new ArrangementGenerator(n, r);
        int[] a = new int[r];
        for(int i=0; i<r; i++) a[i] = -1;
        arr.create(a, 0);
        return arr;
    }
}
    
```

Table III.1.4.2.2. Code snippet generating all binary arrangements

Each element of the list “*arrangements*” is a binary arrangement $a(\Omega)$ presented by an array of bits (0 and 1). The method “*create(int[] a, int i)*” which is recursive method, is the main one that generates arrangements. The method call “*ArrangementGenerator.parse(2, n)*” will list all possible binary arrangements.

Formula III.1.4.2.5 specifies the connection between $s(\Omega: \{X_i=1\})$ and $s(\Omega: \{X_i=0\})$, between $c(\Omega: \{X_i=1\})$ and $c(\Omega: \{X_i=0\})$.

$$\begin{aligned} s(\Omega: \{X_i = 1\}) + s(\Omega: \{X_i = 0\}) &= s(\Omega) \\ c(\Omega: \{X_i = 1\}) + c(\Omega: \{X_i = 0\}) &= c(\Omega) \end{aligned}$$

Formula III.1.4.2.5. Connection among arrangement sum and counter

It is easy to draw formula III.1.4.2.5 when the set of all arrangements $a(\Omega: \{X_i=1\})$ is complement of the set of all arrangements $a(\Omega: \{X_i=0\})$.

Let K be a set of X_i (s) whose values are 1 and let L be a set of X_i (s) whose values are 0. K and L are mutually complementary. Formula III.1.4.2.6 determines sets K and L .

$$\begin{cases} K = \{i : X_i = 1\} \\ L = \{i : X_i = 0\} \\ K \cap L = \emptyset \\ K \cup L = \{1, 2, \dots, n\} \end{cases}$$

Formula III.1.4.2.6. Sets K and L

The **AND-gate** inference represents prerequisite relationship satisfying AND-gate condition specified by formula III.1.4.2.7.

$$P(Y = 1 | A_i = OFF \text{ for some } i) = 0$$

Formula III.1.4.2.7. AND-gate condition

From formula III.1.4.2.4, we have

$$\begin{aligned} P(Y = 1 | X_1, X_2, \dots, X_n) &= \sum_{A_1, A_2, \dots, A_n} P(Y = 1 | A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i | X_i) \\ &= \prod_{i=1}^n P(A_i = ON | X_i) \\ &\quad (\text{due to } P(Y = 1 | A_i = OFF \text{ for some } i) = 0) \\ &= \left(\prod_{i \in K} P(A_i = ON | X_i = 1) \right) \left(\prod_{i \notin K} P(A_i = ON | X_i = 0) \right) \\ &= \left(\prod_{i \in K} p_i \right) \left(\prod_{i \notin K} 0 \right) = \begin{cases} \prod_{i=1}^n p_i & \text{if all } X_i (s) \text{ are 1} \\ 0 & \text{if there exists at least one } X_i = 0 \end{cases} \\ &\quad (\text{Due to formula III.1.4.2.3}) \end{aligned}$$

In general, formula III.1.4.2.8 specifies AND-gate inference.

$$\begin{aligned} P(X_1 \odot X_2 \odot \dots \odot X_n) &= P(Y = 1 | X_1, X_2, \dots, X_n) \\ &= \begin{cases} \prod_{i=1}^n p_i & \text{if all } X_i (s) \text{ are 1} \\ 0 & \text{if there exists at least one } X_i = 0 \end{cases} \\ P(Y = 0 | X_1, X_2, \dots, X_n) &= \begin{cases} 1 - \prod_{i=1}^n p_i & \text{if all } X_i (s) \text{ are 1} \\ 1 & \text{if there exists at least one } X_i = 0 \end{cases} \end{aligned}$$

Formula III.1.4.2.8. AND-gate inference

The AND-gate formula was also described in (Díez & Druzdzel, 2007, p. 33). Formula III.1.4.2.8 varies according to two cases whose arrangement counters are listed as follows:

$$L = \emptyset$$

$$c(\Omega : \{X_i = 1\}) = 1, c(\Omega : \{X_i = 0\}) = 0, c(\Omega) = 1.$$

$$L \neq \emptyset$$

$$c(\Omega : \{X_i = 1\}) = 2^{n-1} - 1, c(\Omega : \{X_i = 0\}) = 2^{n-1}, c(\Omega) = 2^n - 1.$$

The **OR-gate** inference represents prerequisite relationship satisfying OR-gate condition specified by formula III.1.4.2.9 (Neapolitan, 2003, p. 157).

$$P(Y = 1 | A_i = ON \text{ for some } i) = 1$$

Formula III.1.4.2.9. OR-gate condition

The OR-gate condition implies

$$P(Y = 0 | A_i = ON \text{ for some } i) = 0$$

From formula III.1.4.2.4, we have (Neapolitan, 2003, p. 159):

$$\begin{aligned} P(Y = 0 | X_1, X_2, \dots, X_n) &= \sum_{A_1, A_2, \dots, A_n} P(Y = 1 | A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i | X_i) \\ &= \prod_{i=1}^n P(A_i = OFF | X_i) \\ &\quad (\text{due to } P(Y = 1 | A_i = ON \text{ for some } i) = 0) \\ &= \left(\prod_{i \in K} P(A_i = OFF | X_i = 1) \right) \left(\prod_{i \notin K} P(A_i = OFF | X_i = 0) \right) \\ &= \left(\prod_{i \in K} (1 - p_i) \right) \left(\prod_{i \notin K} 1 \right) = \begin{cases} \prod_{i \in K} (1 - p_i) & \text{if } K \neq \emptyset \\ 1 & \text{if } K = \emptyset \end{cases} \\ &\quad (\text{Due to formula III.1.4.2.3}) \end{aligned}$$

In general, formula III.1.4.2.10 specifies OR-gate inference.

$$\begin{aligned} P(X_1 \oplus X_2 \oplus \dots \oplus X_n) &= 1 - P(Y = 0 | X_1, X_2, \dots, X_n) \\ &= \begin{cases} 1 - \prod_{i \in K} (1 - p_i) & \text{if } K \neq \emptyset \\ 0 & \text{if } K = \emptyset \end{cases} \\ P(Y = 0 | X_1, X_2, \dots, X_n) &= \begin{cases} \prod_{i \in K} (1 - p_i) & \text{if } K \neq \emptyset \\ 1 & \text{if } K = \emptyset \end{cases} \end{aligned}$$

Formula III.1.4.2.10. OR-gate inference

Where K is the set of X_i (s) whose values are 1. The OR-gate formula was mentioned in (Neapolitan, 2003, p. 158) and (Díez & Druzdzel, 2007, p. 20). Formula III.1.4.2.10 varies according to two cases whose arrangement counters are listed as follows:

$$K \neq \emptyset$$

$$c(\Omega: \{X_i = 1\}) = 2^{n-1}, c(\Omega: \{X_i = 0\}) = 2^{n-1} - 1, c(\Omega) = 2^n - 1.$$

$$K = \emptyset$$

$$c(\Omega: \{X_i = 1\}) = 0, c(\Omega: \{X_i = 0\}) = 1, c(\Omega) = 1.$$

According to De Morgan's rule with regard to AND-gate and OR-gate, we have:

$$\begin{aligned} P(\text{not}(X_1 \odot X_2 \odot \dots \odot X_n)) &= P((\text{not}(X_1)) \oplus (\text{not}(X_2)) \oplus \dots \oplus (\text{not}(X_n))) \\ &= \begin{cases} 1 - \prod_{i \in L} (1 - (1 - p_i)) & \text{if } L \neq \emptyset \\ 0 & \text{if } L = \emptyset \end{cases} \\ &\quad (\text{Due to formula III.1.4.2.10}) \end{aligned}$$

According to formula III.1.4.2.8, we also have:

$$P(\text{not}(X_1 \oplus X_2 \oplus \dots \oplus X_n)) = P((\text{not}(X_1)) \odot (\text{not}(X_2)) \odot \dots \odot (\text{not}(X_n)))$$

$$= \begin{cases} \prod_{i=1}^n P(\text{not}(X_i)) & \text{if all } \text{not}(X_i) (s) \text{ are 1} \\ 0 & \text{if there exists at least one } \text{not}(X_i) = 0 \end{cases}$$

$$= \begin{cases} \prod_{i=1}^n (1 - p_i) & \text{if all } X_i (s) \text{ are 0} \\ 0 & \text{if there exists at least one } X_i = 1 \end{cases}$$

In general, formula III.1.4.2.11 specifies NAND-gate inference and NOR-gate inference derived from AND-gate and OR-gate:

$$P(\text{not}(X_1 \odot X_2 \odot \dots \odot X_n)) = \begin{cases} 1 - \prod_{i \in L} p_i & \text{if } L \neq \emptyset \\ 0 & \text{if } L = \emptyset \end{cases}$$

$$P(\text{not}(X_1 \oplus X_2 \oplus \dots \oplus X_n)) = \begin{cases} \prod_{i=1}^n (1 - p_i) & \text{if } K = \emptyset \\ 0 & \text{if } K \neq \emptyset \end{cases}$$

Formula III.1.4.2.11. NAND-gate inference and NOR-gate inference

Where K and L are the sets of X_i (s) whose values are 1 and 0, respectively.

Suppose the number of sources X_i (s) is even. Let O be the set of X_i (s) whose indices are odd. Let O_1 and O_2 be subsets of O , in which all X_i (s) are 1 and 0, respectively. Let E be the set of X_i (s) whose indices are even. Let E_1 and E_2 be subsets of E , in which all X_i (s) are 1 and 0, respectively.

$$\begin{array}{ll} \left\{ \begin{array}{l} E = \{2, 4, 6, \dots, n\} \\ E_1 \subseteq E \\ E_2 \subseteq E \\ E_1 \cup E_2 = E \\ E_1 \cap E_2 = \emptyset \\ X_i = 1, \forall i \in E_1 \\ X_i = 0, \forall i \in E_2 \end{array} \right. & \text{and} \quad \left\{ \begin{array}{l} O = \{1, 3, 5, \dots, n-1\} \\ O_1 \subseteq O \\ O_2 \subseteq O \\ O_1 \cup O_2 = O \\ O_1 \cap O_2 = \emptyset \\ X_i = 1, \forall i \in O_1 \\ X_i = 0, \forall i \in O_2 \end{array} \right. \end{array}$$

Thus, O_1 and E_1 are subsets of K . Sources X_i (s) and target Y follow **XOR-gate** if one of two XOR-gate conditions specified by formula III.1.4.2.12 is satisfied.

$$P(Y = 1 \mid \left\{ \begin{array}{l} A_i = ON \text{ for } i \in O \\ A_i = OFF \text{ for } i \notin O \end{array} \right\})$$

$$= P(Y = 1 \mid A_1 = ON, A_2 = OFF, \dots, A_{n-1} = ON, A_n = OFF) = 1$$

$$P(Y = 1 \mid \left\{ \begin{array}{l} A_i = ON \text{ for } i \in E \\ A_i = OFF \text{ for } i \notin E \end{array} \right\})$$

$$= P(Y = 1 \mid A_1 = OFF, A_2 = ON, \dots, A_{n-1} = OFF, A_n = ON) = 1$$

Formula III.1.4.2.12. Two XOR-gate conditions

From formula III.1.4.2.4, we have:

$$P(Y = 1 \mid X_1, X_2, \dots, X_n) = \sum_{A_1, A_2, \dots, A_n} P(Y = 1 \mid A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i \mid X_i)$$

If both XOR-gate conditions are not satisfied then,

$$P(Y = 1 \mid X_1, X_2, \dots, X_n) = 0$$

If the first XOR-gate condition is satisfied, we have:

$$\begin{aligned}
 & P(Y = 1 | X_1, X_2, \dots, X_n) \\
 &= P(Y = 1 | A_1 = ON, A_2 = OFF, \dots, A_{n-1} = ON, A_n = OFF) \prod_{i=1}^n P(A_i | X_i) \\
 &= \left(\prod_{i \in O} P(A_i = ON | X_i) \right) \left(\prod_{i \in E} P(A_i = OFF | X_i) \right)
 \end{aligned}$$

We have

$$\begin{aligned}
 & \prod_{i \in O} P(A_i = ON | X_i) \\
 &= \left(\prod_{i \in O_1} P(A_i = ON | X_i = 1) \right) * \left(\prod_{i \in O_2} P(A_i = ON | X_i = 0) \right) \\
 &= \left(\prod_{i \in O_1} p_i \right) * \left(\prod_{i \in O_2} 0 \right) = \begin{cases} \prod_{i \in O_1} p_i & \text{if } O_2 = \emptyset \\ 0 & \text{if } O_2 \neq \emptyset \end{cases} \\
 & \quad (\text{Due to formula III.1.4.2.3})
 \end{aligned}$$

We also have

$$\begin{aligned}
 & \prod_{i \in E} P(A_i = OFF | X_i) \\
 &= \left(\prod_{i \in E_1} P(A_i = OFF | X_i = 1) \right) * \left(\prod_{i \in E_2} P(A_i = OFF | X_i = 0) \right) \\
 &= \left(\prod_{i \in E_1} (1 - p_i) \right) \left(\prod_{i \in E_2} 1 \right) = \begin{cases} \prod_{i \in E_1} (1 - p_i) & \text{if } E_1 \neq \emptyset \\ 1 & \text{if } E_1 = \emptyset \end{cases} \\
 & \quad (\text{Due to formula III.1.4.2.3})
 \end{aligned}$$

Given the first XOR-gate condition, it implies

$$\begin{aligned}
 P(Y = 1 | X_1, X_2, \dots, X_n) &= \left(\prod_{i \in O} P(A_i = ON | X_i) \right) \left(\prod_{i \in E} P(A_i = OFF | X_i) \right) \\
 &= \begin{cases} \left(\prod_{i \in O_1} p_i \right) \left(\prod_{i \in E_1} (1 - p_i) \right) & \text{if } O_2 = \emptyset \text{ and } E_1 \neq \emptyset \\ \prod_{i \in O_1} p_i & \text{if } O_2 = \emptyset \text{ and } E_1 = \emptyset \\ 0 & \text{if } O_2 \neq \emptyset \end{cases}
 \end{aligned}$$

Similarly, given the second XOR-gate condition, we have:

$$\begin{aligned}
 P(Y = 1|X_1, X_2, \dots, X_n) &= \left(\prod_{i \in E} P(A_i = ON|X_i) \right) \left(\prod_{i \in O} P(A_i = OFF|X_i) \right) \\
 &= \begin{cases} \left(\prod_{i \in E_1} p_i \right) \left(\prod_{i \in O_1} (1 - p_i) \right) & \text{if } E_2 = \emptyset \text{ and } O_1 \neq \emptyset \\ \prod_{i \in E_1} p_i & \text{if } E_2 = \emptyset \text{ and } O_1 = \emptyset \\ 0 & \text{if } E_2 \neq \emptyset \end{cases}
 \end{aligned}$$

If one of XOR-gate conditions is satisfied then,

$$\begin{aligned}
 P(Y = 1|X_1, X_2, \dots, X_n) &= \left(\prod_{i \in O} P(A_i = ON|X_i) \right) \left(\prod_{i \in E} P(A_i = OFF|X_i) \right) \\
 &\quad + \left(\prod_{i \in E} P(A_i = ON|X_i) \right) \left(\prod_{i \in O} P(A_i = OFF|X_i) \right)
 \end{aligned}$$

This implies formula III.1.4.2.13 to specify XOR-gate inference.

$$\begin{aligned}
 P(X_1 \otimes X_2 \otimes \dots \otimes X_n) &= P(Y = 1|X_1, X_2, \dots, X_n) \\
 &= \begin{cases} \left(\prod_{i \in O_1} p_i \right) \left(\prod_{i \in E_1} (1 - p_i) \right) + \left(\prod_{i \in E_1} p_i \right) \left(\prod_{i \in O_1} (1 - p_i) \right) & \text{if } O_2 = \emptyset \text{ and } E_2 = \emptyset \\ \left(\prod_{i \in O_1} p_i \right) \left(\prod_{i \in E_1} (1 - p_i) \right) & \text{if } O_2 = \emptyset \text{ and } E_1 \neq \emptyset \text{ and } E_2 \neq \emptyset \\ \prod_{i \in O_1} p_i & \text{if } O_2 = \emptyset \text{ and } E_1 = \emptyset \\ \left(\prod_{i \in E_1} p_i \right) \left(\prod_{i \in O_1} (1 - p_i) \right) & \text{if } E_2 = \emptyset \text{ and } O_1 \neq \emptyset \text{ and } O_2 \neq \emptyset \\ \prod_{i \in E_1} p_i & \text{if } E_2 = \emptyset \text{ and } O_1 = \emptyset \\ 0 & \text{if } O_2 \neq \emptyset \text{ and } E_2 \neq \emptyset \\ 0 & \text{if } n < 2 \text{ or } n \text{ is odd} \end{cases}
 \end{aligned}$$

Where,

$$\begin{cases} O = \{1, 3, 5, \dots, n-1\} \\ O_1 \subseteq O \\ O_2 \subseteq O \\ O_1 \cup O_2 = O \\ O_1 \cap O_2 = \emptyset \\ X_i = 1, \forall i \in O_1 \\ X_i = 0, \forall i \in O_2 \end{cases} \quad \text{and} \quad \begin{cases} E = \{2, 4, 6, \dots, n\} \\ E_1 \subseteq E \\ E_2 \subseteq E \\ E_1 \cup E_2 = E \\ E_1 \cap E_2 = \emptyset \\ X_i = 1, \forall i \in E_1 \\ X_i = 0, \forall i \in E_2 \end{cases}$$

Formula III.1.4.2.13. XOR-gate inference

Given $n \geq 2$ and n is even, formula III.1.4.2.13 varies according to six cases whose arrangement counters are listed as follows:

$O_2 = \emptyset$ and $E_2 = \emptyset$

$$c(\Omega: \{X_i = 1\}) = 1, c(\Omega: \{X_i = 0\}) = 0, c(\Omega) = 1.$$

$O_2 = \emptyset$ and $E_1 \neq \emptyset$ and $E_2 \neq \emptyset$

$$c(\Omega: \{X_i = 1\}) = 2^{\frac{n}{2}} - 2, c(\Omega: \{X_i = 0\}) = 0, c(\Omega) = 2^{\frac{n}{2}} - 2.$$

$O_2 = \emptyset$ and $E_1 = \emptyset$

$$c(\Omega: \{X_i = 1\}) = 1, c(\Omega: \{X_i = 0\}) = 0, c(\Omega) = 1.$$

$E_2 = \emptyset$ and $O_1 \neq \emptyset$ and $O_2 \neq \emptyset$

$$c(\Omega: \{X_i = 1\}) = 2^{\frac{n}{2}-1} - 1, c(\Omega: \{X_i = 0\}) = 2^{\frac{n}{2}-1} - 1, c(\Omega) = 2^{\frac{n}{2}} - 2.$$

$E_2 = \emptyset$ and $O_1 = \emptyset$

$$c(\Omega: \{X_i = 1\}) = 0, c(\Omega: \{X_i = 0\}) = 1, c(\Omega) = 1.$$

$O_2 \neq \emptyset$ and $E_2 \neq \emptyset$

$$\begin{aligned} c(\Omega: \{X_i = 1\}) &= \left(2^{\frac{n}{2}-1} - 1\right)\left(2^{\frac{n}{2}} - 1\right), c(\Omega: \{X_i = 0\}) = 2^{\frac{n}{2}-1}\left(2^{\frac{n}{2}} - 1\right), c(\Omega) \\ &= \left(2^{\frac{n}{2}} - 1\right)^2. \end{aligned}$$

Suppose the number of sources X_i (s) is even. According to **XNOR-gate** inference (Wikipedia, 2016), the output is on if all inputs get the same value 1 (or 0). Sources X_i (s) and target Y follow XNOR-gate if one of two XNOR-gate conditions specified by formula III.1.4.2.14 is satisfied.

$$\begin{aligned} P(Y = 1 | A_i = ON, \forall i) &= 1 \\ P(Y = 1 | A_i = OFF, \forall i) &= 1 \end{aligned}$$

Formula III.1.4.2.14. Two XNOR-gate conditions

From formula III.1.4.2.4, we have:

$$P(Y = 1 | X_1, X_2, \dots, X_n) = \sum_{A_1, A_2, \dots, A_n} P(Y = 1 | A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i | X_i)$$

If both XNOR-gate conditions are not satisfied then,

$$P(Y = 1 | X_1, X_2, \dots, X_n) = 0$$

If $A_i = ON$ for all i , we have:

$$\begin{aligned} P(Y = 1 | X_1, X_2, \dots, X_n) &= P(Y = 1 | A_i = ON, \forall i) \prod_{i=1}^n P(A_i = ON | X_i) \\ &= \prod_{i=1}^n P(A_i = ON | X_i) = \begin{cases} \prod_{i=1}^n p_i & \text{if } L = \emptyset \\ 0 & \text{if } L \neq \emptyset \end{cases} \end{aligned}$$

(Please see similar proof in AND-gate inference)

If $A_i = OFF$ for all i , we have:

$$P(Y = 1 | X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(A_i = OFF | X_i) = \begin{cases} \prod_{i \in K} (1 - p_i) & \text{if } K \neq \emptyset \\ 1 & \text{if } K = \emptyset \end{cases}$$

(Please see similar proof in OR-gate inference)

If one of XNOR-gate conditions is satisfied then,

$$P(Y = 1 | X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(A_i = ON | X_i) + \prod_{i=1}^n P(A_i = OFF | X_i)$$

This implies formula III.1.4.2.15 to specify XNOR-gate inference.

$$P(\text{not}(X_1 \otimes X_2 \otimes \dots \otimes X_n)) = P(Y = 1 | X_1, X_2, \dots, X_n)$$

$$= \begin{cases} \prod_{i=1}^n p_i + \prod_{i=1}^n (1 - p_i) & \text{if } L = \emptyset \\ \prod_{i \in K} (1 - p_i) & \text{if } L \neq \emptyset \text{ and } K \neq \emptyset \\ 1 & \text{if } L \neq \emptyset \text{ and } K = \emptyset \end{cases}$$

Formula III.1.4.2.15. XNOR-gate inference

Where K and L are the sets of X_i (s) whose values are 1 and 0, respectively. Formula III.1.4.2.15 varies according to three cases whose arrangement counters are listed as follows:

$$L = \emptyset$$

$$c(\Omega: \{X_i = 1\}) = 1, c(\Omega: \{X_i = 0\}) = 0, c(\Omega) = 1.$$

$$L \neq \emptyset \text{ and } K \neq \emptyset$$

$$c(\Omega: \{X_i = 1\}) = 2^{n-1} - 1, c(\Omega: \{X_i = 0\}) = 2^{n-1} - 1, c(\Omega) = 2^n - 2.$$

$$L \neq \emptyset \text{ and } K = \emptyset$$

$$c(\Omega: \{X_i = 1\}) = 0, c(\Omega: \{X_i = 0\}) = 1, c(\Omega) = 1.$$

Let U be a set of indices such that $A_i = ON$ and let $\alpha \geq 0$ and $\beta \geq 0$ be predefined numbers. The U-gate inference is defined based on α, β and cardinality of U . Formula III.1.4.2.16 specifies four common U-gate conditions.

$ U =\alpha$	$P(Y = 1 A_1, A_2, \dots, A_n) = 1$ if there are exactly α variables $A_i = ON$ (s). Otherwise, $P(Y = 1 A_1, A_2, \dots, A_n) = 0$.
$ U \geq\alpha$	$P(Y = 1 A_1, A_2, \dots, A_n) = 1$ if there are at least α variables $A_i = ON$ (s). Otherwise, $P(Y = 1 A_1, A_2, \dots, A_n) = 0$.
$ U \leq\beta$	$P(Y = 1 A_1, A_2, \dots, A_n) = 1$ if there are at most β variables $A_i = ON$ (s). Otherwise, $P(Y = 1 A_1, A_2, \dots, A_n) = 0$.
$\alpha \leq U \leq \beta$	$P(Y = 1 A_1, A_2, \dots, A_n) = 1$ if the number of $A_i = ON$ (s) is from α to β . Otherwise, $P(Y = 1 A_1, A_2, \dots, A_n) = 0$.

Formula III.1.4.2.16. U-gate conditions

Your attention please, U-gate condition on $|U|$ can be arbitrary and it is only relevant to A_i (s) (*ON* or *OFF*) and the way to combine A_i (s). For example, AND-gate and OR-gate are specific cases of U-gate with $|U|=n$ and $|U|\geq 1$, respectively. XOR-gate and XNOR-gate are also specific cases of U-gate with specific conditions on A_i (s). However, it must be assured that there is at least one combination of A_i (s) satisfying the predefined U-gate condition, which causes that U-gate probability is not always equal to 0. In this research, U-gate is the most general nonlinear gate where U-gate probability contains products of weights (see formula III.1.4.2.17). Later on, we will research a so-called SIGMA-gate that contains only linear combination of weights (sum of weights, see formula III.1.4.2.19). Shortly, each X-gate is a pattern owning a particular X-gate inference that is X-gate probability $P(X_1 X_2 \dots X_n)$. Each X-gate inference is based on particular X-gate condition (s) relevant to only variables A_i (s).

From formula III.1.4.2.4, we have:

$$P(Y = 1 | X_1, X_2, \dots, X_n) = \sum_{A_1, A_2, \dots, A_n} P(Y = 1 | A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i | X_i)$$

Let \mathcal{U} be the set of all possible U (s), we have:

$$\begin{aligned}
 P(Y = 1|X_1, X_2, \dots, X_n) &= \sum_{U \in \mathcal{U}} P(Y = 1|A_1, A_2, \dots, A_n) \prod_{i=1}^n P(A_i|X_i) \\
 &= \sum_{U \in \mathcal{U}} \prod_{i \in U} P(A_i = ON|X_i) \prod_{j \notin U} P(A_j = OFF|X_j)
 \end{aligned}$$

If $X_i = 0, \forall i \in U$ then,

$$P(Y = 1|X_1, X_2, \dots, X_n) = \sum_{U \in \mathcal{U}} \prod_{i \in U} 0 \prod_{j \notin U} P(A_j = OFF|X_j) = 0$$

This implies all sets U (s) must be subsets of K . The U-gate probability is rewritten as follows:

$$\begin{aligned}
 P(Y = 1|X_1, X_2, \dots, X_n) &= \sum_{U \in \mathcal{U}} \prod_{i \in U} P(A_i = ON|X_i = 1) \prod_{j \notin U} P(A_j = OFF|X_j) \\
 &= \sum_{U \in \mathcal{U}} \prod_{i \in U} p_i \prod_{j \notin U} P(A_j = OFF|X_j) \\
 &= \sum_{U \in \mathcal{U}} \prod_{i \in U} p_i \prod_{j \in K \setminus U} P(A_j = OFF|X_j = 1) \prod_{j \notin K} P(A_j = OFF|X_j = 0) \\
 &= \sum_{U \in \mathcal{U}} \prod_{i \in U} p_i \prod_{j \in K \setminus U} (1 - p_j) \prod_{j \notin K} 1 = \sum_{U \in \mathcal{U}} \prod_{i \in U} p_i \prod_{j \in K \setminus U} (1 - p_j)
 \end{aligned}$$

(Due to formula III.1.4.2.3)

Let P_U be the U-gate probability, formula III.1.4.2.17 specifies U-gate inference and cardinality of \mathcal{U} where \mathcal{U} is the set of subsets (U) of K .

Let, $S_U = \sum_{U \in \mathcal{U}} \prod_{i \in U} p_i \prod_{j \in K \setminus U} (1 - p_j)$ $P_U = P(X_1 \sqcup X_2 \sqcup \dots \sqcup X_n) = P(Y = 1 X_1, X_2, \dots, X_n)$ As a convention, $\prod_{i \in U} p_i = 1$ if $ U = 0$ $\prod_{j \in K \setminus U} (1 - p_j) = 1$ if $ U = K $	
$ U =0$	$P_U = \begin{cases} \prod_{j=1}^n (1 - p_j) & \text{if } K > 0 \\ 1 & \text{if } K = 0 \end{cases}$ $ \mathcal{U} = 1$
$ U \geq 0$	$P_U = \begin{cases} S_U & \text{if } K > 0 \\ 1 & \text{if } K = 0 \end{cases}$ $ \mathcal{U} = 2^{ K }$ The case $ U \geq 0$ is the same to the case $ U \leq n$
$ U =n$	$P_U = \begin{cases} \prod_{i=1}^n p_i & \text{if } K = n \\ 0 & \text{if } K < n \end{cases}$ $ \mathcal{U} = \begin{cases} 1 & \text{if } K = n \\ 0 & \text{if } K < n \end{cases}$

$ U =\alpha$ $0<\alpha< n$	$P_U = \begin{cases} S_U & \text{if } K \geq \alpha \\ 0 & \text{if } K < \alpha \end{cases}$ $ U = \begin{cases} \binom{ K }{\alpha} & \text{if } K \geq \alpha \\ 0 & \text{if } K < \alpha \end{cases}$
$ U \geq\alpha$ $0<\alpha< n$	$P_U = \begin{cases} S_U & \text{if } K \geq \alpha \\ 0 & \text{if } K < \alpha \end{cases}$ $ U = \begin{cases} \sum_{j=\alpha}^{ K } \binom{ K }{j} & \text{if } K \geq \alpha \\ 0 & \text{if } K < \alpha \end{cases}$
$ U \leq\beta$ $0<\beta< n$	$P_U = \begin{cases} S_U & \text{if } K > 0 \\ 1 & \text{if } K = 0 \end{cases}$ $ U = \begin{cases} \sum_{j=0}^{\min(\beta, K)} \binom{ K }{j} & \text{if } K > 0 \\ 1 & \text{if } K = 0 \end{cases}$
$\alpha\leq U \leq\beta$ $0<\alpha< n$ $0<\beta< n$	$P_U = \begin{cases} S_U & \text{if } K \geq \alpha \\ 0 & \text{if } K < \alpha \end{cases}$ $ U = \begin{cases} \sum_{j=\alpha}^{\min(\beta, K)} \binom{ K }{j} & \text{if } K \geq \alpha \\ 0 & \text{if } K < \alpha \end{cases}$

Formula III.1.4.2.17. U-gate inference

Note that the notation $\binom{n}{j}$ denotes the number of combinations of j elements taken from n elements.

$$\binom{n}{j} = \frac{n!}{j!(n-j)!}$$

AND-gate and OR-gate are special cases of U-gate with $|U|=n$ and $|U|\geq 1$, respectively. Arrangement counters relevant to U-gate inference and the set K are listed as follows:

$$|K| = 0$$

$$c(\Omega: \{X_i = 1\}) = 0, c(\Omega: \{X_i = 0\}) = 1, c(\Omega) = 1.$$

$$|K| = 1$$

$$c(\Omega: \{X_i = 1\}) = 1, c(\Omega: \{X_i = 0\}) = 0, c(\Omega) = 1.$$

$$|K| = \alpha \text{ and } \alpha > 0$$

$$c(\Omega: \{X_i = 1\}) = \binom{n-1}{\alpha-1}, c(\Omega: \{X_i = 0\}) = \binom{n-1}{\alpha}, c(\Omega) = \binom{n}{\alpha}.$$

$$|K| \leq \alpha \text{ and } \alpha > 0$$

$$c(\Omega: \{X_i = 1\}) = \sum_{j=1}^{\alpha} \binom{n-1}{j-1}, c(\Omega: \{X_i = 0\}) = \sum_{j=0}^{\alpha} \binom{n-1}{j}, c(\Omega) = \sum_{j=0}^{\alpha} \binom{n}{j}.$$

$$|K| \geq \alpha \text{ and } \alpha > 0$$

$$c(\Omega: \{X_i = 1\}) = \sum_{j=\alpha}^n \binom{n-1}{j-1}, c(\Omega: \{X_i = 0\}) = \sum_{j=\alpha}^{n-1} \binom{n-1}{j}, c(\Omega) = \sum_{j=\alpha}^n \binom{n}{j}.$$

The **SIGMA-gate** inference (Nguyen, 2016) represents aggregation relationship satisfying SIGMA-gate condition specified by formula III.1.4.2.18.

$$P(Y) = P\left(\sum_{i=1}^n A_i\right)$$

Where the set of A_i (s) is complete and mutually exclusive.

$$\sum_{i=1}^n w_i = 1$$

$$A_i \cap A_j = \emptyset, \forall i \neq j$$

Formula III.1.4.2.18. SIGMA-gate condition

The sigma sum $\sum_{i=1}^n A_i$ indicates that Y is exclusive union of A_i (s) and here, it does not express arithmetical additions.

$$Y = \sum_{i=1}^n A_i = \bigcup_{i=1}^n A_i$$

This implies:

$$P(Y) = P\left(\sum_{i=1}^n A_i\right) = P\left(\bigcup_{i=1}^n A_i\right) = \sum_{i=1}^n P(A_i)$$

The sigma sum $\sum_{i=1}^n P(A_i)$ now expresses arithmetical additions of probabilities $P(A_i)$.

SIGMA-gate inference requires the set of A_i (s) is complete and mutually exclusive, which means that the set of X_i (s) is complete and mutually exclusive too.

The SIGMA-gate probability is (Nguyen, 2016):

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

(due to SIGMA – gate condition)

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

(because A_i (s) are mutually exclusive)

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

(because A_i is only dependent on X_i)

It implies

$$P(Y = 1 | X_1, X_2, \dots, X_n)$$

$$= \sum_{i=1}^n P(A_i = ON | X_i)$$

$$= \left(\sum_{i \in K} P(A_i = ON | X_i = 1) \right) + \left(\sum_{i \notin K} P(A_i = ON | X_i = 0) \right)$$

$$= \sum_{i \in K} w_i + \sum_{i \notin K} 0 = \sum_{i \in K} w_i$$

(Due to formula III.1.4.2.3)

In general, formula III.1.4.2.19 specifies the theorem of SIGMA-gate inference (Nguyen L. , Theorem of SIGMA-gate Inference in Bayesian Network, 2016). The base of this theorem was mentioned by Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, pp. 292-295).

$$P(X_1 + X_2 + \dots + X_n) = P\left(\sum_{i=1}^n X_i\right) = P(Y = 1|X_1, X_2, \dots, X_n) = \sum_{i \in K} w_i$$

$$P(Y = 0|X_1, X_2, \dots, X_n) = 1 - \sum_{i \in K} w_i = \sum_{i \in L} w_i$$

Where the set of X_i (s) is complete and mutually exclusive.

$$\sum_{i=1}^n w_i = 1$$

$$X_i \cap X_j = \emptyset, \forall i \neq j$$

Formula III.1.4.2.19. SIGMA-gate inference

Formula III.1.4.2.19 is the same to formula III.1.3.3, which is the theorem of SIGMA-gate inference. The arrangement counters of SIGMA-gate inference are $c(\Omega: \{X_i=1\}) = c(\Omega: \{X_i=0\}) = 2^{n-1}$, $c(\Omega) = 2^n$.

Formula III.1.4.2.3 specifies the “clockwise” strength of relationship between X_i and Y . Event $X_i=1$ causes event $A_i=ON$ with “clockwise” weight w_i . There is a question “given $X_i=0$, how likely the event $A_i=OFF$ is”. In order to solve this problem, I define a so-called “counterclockwise” strength of relationship between X_i and Y denoted ω_i . Event $X_i=0$ causes event $A_i=OFF$ with “counterclockwise” weight ω_i . In other words, each arc in simple graph is associated with a clockwise weight w_i and a counterclockwise weight ω_i . Such graph is called *bi-weight simple graph* shown in figure III.1.4.2.3.

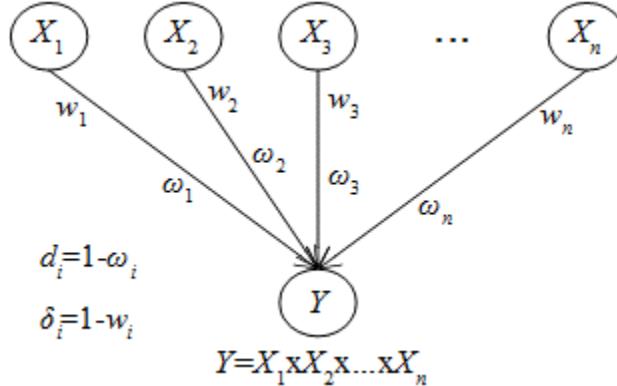


Figure III.1.4.2.3. Bi-weight simple graph

With bi-weight simple graph, all X-gate inferences are extended as so-called X-gate bi-inferences. Derived from formula III.1.4.2.3, formula III.1.4.2.20 specifies conditional probability of accountable variables with regard to bi-weight graph.

$$P(A_i = ON|X_i = 1) = p_i = w_i$$

$$P(A_i = ON|X_i = 0) = 1 - \rho_i = 1 - \omega_i$$

$$P(A_i = OFF|X_i = 1) = 1 - p_i = 1 - w_i$$

$$P(A_i = OFF|X_i = 0) = \rho_i = \omega_i$$

Formula III.1.4.2.20. Conditional probability of accountable variables with regard to bi-weight graph

The probabilities $P(A_i=ON | X_i=0)$ and $P(A_i=OFF | X_i=1)$ are called clockwise adder d_i and counterclockwise adder δ_i . As usual, d_i and δ_i are smaller than w_i and ω_i . When $d_i=0$, bi-weight graph becomes normal simple graph.

$$d_i = P(A_i = ON | X_i = 0) = 1 - \rho_i = 1 - \omega_i$$

$$\delta_i = P(A_i = OFF | X_i = 1) = 1 - p_i = 1 - w_i$$

The total clockwise weight or total counterclockwise weight is defined as sum of clockwise weight and clockwise adder or sum of counterclockwise weight and counterclockwise adder. Formula III.1.4.2.21 specifies such total weights W_i and \mathcal{W}_i . These weights are also called relationship powers.

$$W_i = w_i + d_i$$

$$\mathcal{W}_i = \omega_i + \delta_i$$

Where,

$$d_i = 1 - \rho_i = 1 - \omega_i$$

$$\delta_i = 1 - p_i = 1 - w_i$$

Formula III.1.4.2.21. Total clockwise and counterclockwise weights

Given formula III.1.4.2.21, the set of all A_i (s) is complete if and only if $\sum_{i=1}^n W_i = 1$.

By extending aforementioned X-gate inferences, we get bi-inferences for AND-gate, OR-gate, NAND-gate, NOR-gate, XOR-gate, XNOR-gate, and U-gate as shown in table III.1.4.2.3.

$$P(X_1 \odot X_2 \odot \dots \odot X_n) = \prod_{i \in K} p_i \prod_{i \in L} d_i$$

$$P(X_1 \oplus X_2 \oplus \dots \oplus X_n) = 1 - \prod_{i \in K} \delta_i \prod_{i \in L} \rho_i$$

$$P(\text{not}(X_1 \odot X_2 \odot \dots \odot X_n)) = 1 - \prod_{i \in L} \rho_i \prod_{i \in K} \delta_i$$

$$P(\text{not}(X_1 \oplus X_2 \oplus \dots \oplus X_n)) = \prod_{i \in L} d_i \prod_{i \in K} p_i$$

$$P(X_1 \otimes X_2 \otimes \dots \otimes X_n) = \prod_{i \in O_1} p_i \prod_{i \in O_2} d_i \prod_{i \in E_1} \delta_i \prod_{i \in E_2} \rho_i + \prod_{i \in E_1} p_i \prod_{i \in E_2} d_i \prod_{i \in O_1} \delta_i \prod_{i \in O_2} \rho_i$$

$$P(\text{not}(X_1 \otimes X_2 \otimes \dots \otimes X_n)) = \prod_{i \in K} p_i \prod_{i \in L} d_i + \prod_{i \in K} \delta_i \prod_{i \in L} \rho_i$$

$$P(X_1 \uplus X_2 \uplus \dots \uplus X_n) = \sum_{U \in \mathcal{U}} \left(\prod_{i \in U \cap K} p_i \prod_{i \in U \cap L} d_i \right) \left(\prod_{i \in \bar{U} \cap K} \delta_i \prod_{i \in \bar{U} \cap L} \rho_i \right)$$

There are four common conditions of U : $|U|=\alpha$, $|U|\geq\alpha$, $|U|\leq\beta$, and $\alpha\leq|U|\leq\beta$. Note that \bar{U} is the complement of U ,

$$\bar{U} = \{1, 2, \dots, n\} \setminus U$$

The largest cardinality of \mathcal{U} is:

$$|\mathcal{U}| = 2^n$$

Table III.1.4.2.3. Bi-inferences for AND-gate, OR-gate, NAND-gate, NOR-gate, XOR-gate, XNOR-gate, and U-gate

The largest cardinalities of K (L) are 2^{n-1} and 2^n with and without fixed X_i . Thus, it is possible to calculate arrangement counters. As a convention, the product of probabilities is 1 if indices set is empty.

$$\prod_{i \in I} f_i = 1 \text{ if } I = \emptyset$$

With regard to SIGMA-gate bi-inference, the sum of all total clockwise weights must be 1 as follows:

$$\sum_{i=1}^n W_i = \sum_{i=1}^n (w_i + d_i) = \sum_{i=1}^n (w_i + 1 - \omega_i) = 1$$

Derived from formula III.1.4.2.19, the SIGMA-gate probability for bi-weight graph is:

$$\begin{aligned} P(X_1 + X_2 + \dots + X_n) &= \sum_{i=1}^n P(A_i = ON | X_i) \\ &= \sum_{i \in K} P(A_i = ON | X_i = 1) + \sum_{i \in L} P(A_i = ON | X_i = 0) \\ &= \sum_{i \in K} w_i + \sum_{i \in L} d_i \end{aligned}$$

Shortly, formula III.1.4.2.22 specifies SIGMA-gate bi-inference.

$$P(X_1 + X_2 + \dots + X_n) = \sum_{i \in K} w_i + \sum_{i \in L} d_i$$

Where the set of X_i (s) is complete and mutually exclusive.

$$\begin{aligned} \sum_{i=1}^n W_i &= 1 \\ X_i \cap X_j &= \emptyset, \forall i \neq j \end{aligned}$$

Formula III.1.4.2.22. SIGMA-gate bi-inference

The next section will research diagnostic relationship which adheres to X-gate inference.

III.1.4.3. Multi-hypothesis diagnostic relationship

Given a simple graph shown in figure III.1.4.2.1, if we replace the target source Y by an evidence D , we get a so-called *multi-hypothesis diagnostic relationship* whose property adheres to X-gate inference. Maybe there are other diagnostic relationships in which X-gate inference is not concerned. However, this research focuses on X-gate inference and so multi-hypothesis diagnostic relationship is called *X-gate diagnostic relationship*. Sources X_1, X_2, \dots, X_n become hypotheses. As a convention, these hypotheses have prior uniform distribution.

According to aforementioned X-gate network shown in figures III.1.4.2.1 and III.1.4.2.2, the target variable must be binary whereas evidence D can be numeric. It is impossible to establish the evidence D as direct target variable. Thus, the solution of this problem is to add an augmented target binary variable Y and then, the evidence D is connected directly to Y . In other words, the *X-gate diagnostic network* have n sources $\{X_1, X_2, \dots, X_n\}$, one augmented hypothesis Y , and one evidence D . As a convention, X-gate diagnostic network is called *X-D network*. The CPT (s) of the entire network are determined based on combination of diagnostic relationship and X-gate inference mentioned in previous sub-sections III.1.4.1 and III.1.4.2. Figure

III.1.4.3.1 depicts the augmented X-D network. Note that variables X_1, X_2, \dots, X_n , and Y are always binary.

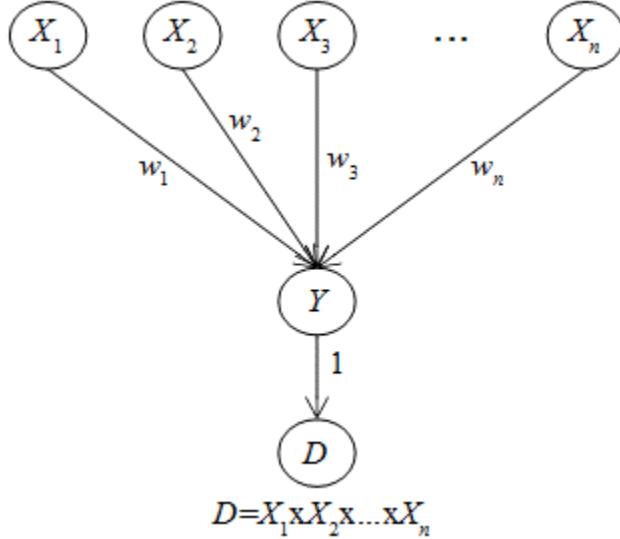


Figure III.1.4.3.1. Augmented X-D network

The joint probability of augmented X-D network shown in figure 4.1 is

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

The joint probability of X-D network is

$$P(X_1, X_2, \dots, X_n, D) = P(D|X_1, X_2, \dots, X_n) \prod_{i=1}^n P(X_i)$$

By applying total probability rule into X-D network, we have:

$$\begin{aligned} P(X_1, X_2, \dots, X_n, D) &= \frac{P(D, X_1, X_2, \dots, X_n)}{P(X_1, X_2, \dots, X_n)} \prod_{i=1}^n P(X_i) \\ &\quad (\text{Due to Bayes' rule}) \\ &= \frac{\sum_Y P(D, X_1, X_2, \dots, X_n | Y) P(Y)}{P(X_1, X_2, \dots, X_n)} \prod_{i=1}^n P(X_i) \\ &\quad (\text{Due to total probability rule}) \\ &= \frac{\sum_Y P(D, X_1, X_2, \dots, X_n | Y) P(Y)}{P(X_1, X_2, \dots, X_n)} \prod_{i=1}^n P(X_i) \\ &= \left(\sum_Y P(D, X_1, X_2, \dots, X_n | Y) * \frac{P(Y)}{P(X_1, X_2, \dots, X_n)} \right) * \prod_{i=1}^n P(X_i) \\ &= \left(\sum_Y P(D|Y) * \frac{P(X_1, X_2, \dots, X_n | Y) P(Y)}{P(X_1, X_2, \dots, X_n)} \right) * \prod_{i=1}^n P(X_i) \\ &\quad (\text{Because } D \text{ is conditionally independent from all } X_i \text{ (s) given } Y) \\ &= \left(\sum_Y P(D|Y) * \frac{P(Y, X_1, X_2, \dots, X_n)}{P(X_1, X_2, \dots, X_n)} \right) * \prod_{i=1}^n P(X_i) \end{aligned}$$

$$\begin{aligned}
 &= \sum_Y P(D|Y)P(Y|X_1, X_2, \dots, X_n) \prod_{i=1}^n P(X_i) \\
 &\quad (\text{Due to Bayes' rule}) \\
 &= \sum_Y P(X_1, X_2, \dots, X_n, Y, D)
 \end{aligned}$$

It is implied that the augmented X-D network is equivalent to X-D network with regard to variables X_1, X_2, \dots, X_n and D . As a convention, augmented X-D network is considered as same as X-D network.

The simplest case of X-D network is NOT-D network having one hypothesis X_1 and one evidence D , equipped with NOT-gate inference. NOT-D network satisfies diagnostic condition because it essentially represents the single diagnostic relationship. Inferred formulas III.1.4.1.1 and III.1.4.2.1, the conditional probability $P(D|X_1)$ and posterior probability $P(X_1|D)$ of NOT-D network are:

$$P(D|X_1) = \begin{cases} 1 - D & \text{if } X_1 = 1 \\ D & \text{if } X_1 = 0 \end{cases}$$

$$\begin{aligned}
 P(X_1|D) &= \frac{P(D|X_1)P(X_1)}{P(X_1)(P(D|X_1 = 0) + P(D|X_1 = 1))} \\
 &\quad (\text{Due to Bayes' rule and uniform distribution of } X_1) \\
 &= \frac{P(D|X_1)}{P(D|X_1 = 0) + P(D|X_1 = 1)} = 1 * P(D|X_1) \\
 &\quad (\text{due to } P(D|X_1 = 0) + P(D|X_1 = 1) = 1)
 \end{aligned}$$

It implies NOT-D network satisfies diagnostic condition. Let

$$\begin{aligned}
 \Omega &= \{X_1, X_2, \dots, X_n\} \\
 n &= |\Omega|
 \end{aligned}$$

We will validate whether the CPT of diagnostic relationship, $P(D|X)$ specified by formula III.1.4.1.6, still satisfies diagnostic condition within general case, X-D network. In other words, X-D network is general case of single diagnostic relationship.

Recall from dependencies shown in figure III.1.4.3.1, formula III.1.4.3.1 specifies the joint probability of X-D network.

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

Where $\Omega = \{X_1, X_2, \dots, X_n\}$.

Formula III.1.4.3.1. Joint probability of X-D network

Formula III.1.4.3.2 specifies the conditional probability of D given X_i (likelihood function) and the posterior probability of X_i given D .

$$\begin{aligned}
 P(D|X_i) &= \frac{P(X_i, D)}{P(X_i)} = \frac{\sum_{\{\Omega, Y, D\} \setminus \{X_i, D\}} P(\Omega, Y, D)}{\sum_{\{\Omega, Y, D\} \setminus \{X_i\}} P(\Omega, Y, D)} \\
 P(X_i|D) &= \frac{P(X_i, D)}{P(D)} = \frac{\sum_{\{\Omega, Y, D\} \setminus \{X_i, D\}} P(\Omega, Y, D)}{\sum_{\{\Omega, Y, D\} \setminus \{D\}} P(\Omega, Y, D)}
 \end{aligned}$$

Formula III.1.4.3.2. Conditional probability $P(D|X_i)$ and posterior probability $P(X_i|D)$

Where $\Omega = \{X_1, X_2, \dots, X_n\}$ and the sign “\” denotes the subtraction (excluding) operator in set theory (Wikipedia, Set (mathematics), 2014).

Given uniform distribution of X_i (s), we have:

$$P(X_1) = P(X_2) = \dots = P(X_n) = \frac{1}{2}$$

The joint probability becomes

$$P(\Omega, Y, D) = \frac{1}{2^n} P(Y|X_1, X_2, \dots, X_n) P(D|Y)$$

The joint probability of X_i and D is:

$$\begin{aligned} P(X_i, D) &= \sum_{\{\Omega, Y, D\} \setminus \{X_i, D\}} P(\Omega, Y, D) \\ &= P(X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 1, X_n = 1, Y = 1, D) \\ &\quad + P(X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 1, X_n = 0, Y = 1, D) + \dots \\ &\quad + P(X_1 = 0, X_2 = 0, \dots, X_i, \dots, X_{n-1} = 0, X_n = 1, Y = 1, D) \\ &\quad + P(X_1 = 0, X_2 = 0, \dots, X_i, \dots, X_{n-1} = 0, X_n = 0, Y = 1, D) \\ &\quad + P(X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 1, X_n = 1, Y = 0, D) \\ &\quad + P(X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 1, X_n = 0, Y = 0, D) + \dots \\ &\quad + P(X_1 = 0, X_2 = 0, \dots, X_i, \dots, X_{n-1} = 0, X_n = 1, Y = 0, D) \\ &\quad + P(X_1 = 0, X_2 = 0, \dots, X_i, \dots, X_{n-1} = 0, X_n = 0, Y = 0, D) \\ &= \frac{1}{2^n} \frac{D}{S} (P(Y = 1|X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 1, X_n = 1) \\ &\quad + P(Y = 1|X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 1, X_n = 0) + \dots \\ &\quad + P(Y = 1|X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 0, X_n = 1) \\ &\quad + P(Y = 1|X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 0, X_n = 0)) \\ &+ \frac{1}{2^n} \frac{M - D}{S} (P(Y = 0|X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 1, X_n = 1) \\ &\quad + P(Y = 0|X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 1, X_n = 0) + \dots \\ &\quad + P(Y = 0|X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 0, X_n = 1) \\ &\quad + P(Y = 0|X_1 = 1, X_2 = 1, \dots, X_i, \dots, X_{n-1} = 0, X_n = 0)) \end{aligned}$$

(Due to formula III.1.4.1.6)

The marginal probability of D is:

$$\begin{aligned} P(D) &= \sum_{\{\Omega, Y, D\} \setminus \{D\}} P(\Omega, Y, D) \\ &= P(X_1 = 1, X_2 = 1, \dots, X_n = 1, Y = 1, D) \\ &\quad + P(X_1 = 1, X_2 = 1, \dots, X_n = 0, Y = 1, D) + \dots \\ &\quad + P(X_1 = 0, X_2 = 0, \dots, X_n = 1, Y = 1, D) \\ &\quad + P(X_1 = 0, X_2 = 0, \dots, X_n = 0, Y = 1, D) \\ &\quad + P(X_1 = 1, X_2 = 1, \dots, X_n = 1, Y = 0, D) \\ &\quad + P(X_1 = 1, X_2 = 1, \dots, X_n = 0, Y = 0, D) + \dots \\ &\quad + P(X_1 = 0, X_2 = 0, \dots, X_n = 1, Y = 0, D) \\ &\quad + P(X_1 = 0, X_2 = 0, \dots, X_n = 0, Y = 0, D) \\ &= \frac{1}{2^n} \frac{D}{S} (P(Y = 1|X_1 = 1, X_2 = 1, \dots, X_n = 1) \\ &\quad + P(Y = 1|X_1 = 1, X_2 = 1, \dots, X_n = 0) + \dots \\ &\quad + P(Y = 1|X_1 = 1, X_2 = 1, \dots, X_n = 1) \\ &\quad + P(Y = 1|X_1 = 1, X_2 = 1, \dots, X_n = 0)) \end{aligned}$$

$$+ \frac{1}{2^n} \frac{M-D}{S} (P(Y=0|X_1=1, X_2=1, \dots, X_n=1) \\ + P(Y=0|X_1=1, X_2=1, \dots, X_n=0) + \dots \\ + P(Y=0|X_1=1, X_2=1, \dots, X_n=1) \\ + P(Y=0|X_1=1, X_2=1, \dots, X_n=0))$$

By applying table III.1.4.2.1, the joint probability $P(X_i, D)$ is determined as follows:

$$P(X_i, D) = \frac{1}{2^n S} \left(D \sum_a P(Y=1|a(\Omega: \{X_i\})) + (M-D) \sum_a P(Y=0|a(\Omega: \{X_i\})) \right) \\ = \frac{1}{2^n S} \left(D \sum_a P(Y=1|a(\Omega: \{X_i\})) + (M-D) \sum_a (1 - P(Y=1|a(\Omega: \{X_i\}))) \right) \\ = \frac{1}{2^n S} ((2D-M)s(\Omega: \{X_i\}) + 2^{n-1}(M-D))$$

Similarly, formula III.1.4.3.3 specifies the joint probability $P(X_i, D)$ and the marginal probability $P(D)$ given uniform distribution of all sources.

$$P(X_i, D) = \frac{1}{2^n S} ((2D-M)s(\Omega: \{X_i\}) + 2^{n-1}(M-D)) \\ P(D) = \frac{1}{2^n S} ((2D-M)s(\Omega) + 2^n(M-D))$$

Formula III.1.4.3.3. Joint probability $P(X_i, D)$ and marginal probability $P(D)$ given uniform distribution of all sources

Where $s(\Omega)$ and $s(\Omega: \{X_i\})$ are specified in table III.1.4.2.1. In general, formula III.1.4.3.4 specifies conditional probability $P(D/X_i)$, posterior probability $P(X_i/D)$, and transformation coefficient for X-gate inference.

$$P(D|X_i=1) = \frac{P(X_i=1, D)}{P(X_i=1)} = \frac{(2D-M)s(\Omega: \{X_i=1\}) + 2^{n-1}(M-D)}{2^{n-1}S} \\ P(D|X_i=0) = \frac{P(X_i=0, D)}{P(X_i=0)} = \frac{(2D-M)s(\Omega: \{X_i=0\}) + 2^{n-1}(M-D)}{2^{n-1}S} \\ P(X_i=1|D) = \frac{P(X_i=1, D)}{P(D)} = \frac{(2D-M)s(\Omega: \{X_i=1\}) + 2^{n-1}(M-D)}{(2D-M)s(\Omega) + 2^n(M-D)} \\ P(X_i=0|D) = 1 - P(X_i=1|D) = \frac{(2D-M)s(\Omega: \{X_i=0\}) + 2^{n-1}(M-D)}{(2D-M)s(\Omega) + 2^n(M-D)} \\ k = \frac{P(X_i|D)}{P(D|X_i)} = \frac{2^{n-1}S}{(2D-M)s(\Omega) + 2^n(M-D)}$$

Formula III.1.4.3.4. Conditional probability, posterior probability, and transformation coefficient of X-D network

The transformation coefficient is rewritten as follows:

$$k = \frac{2^{n-1}S}{2D(s(\Omega) - 2^{n-1}) + M(2^n - s(\Omega))}$$

Note that S, D, M are abstract symbols and there is no proportional connection between $2^{n-1}S$ and D for all D , specified by formula III.1.4.1.6. Assuming that such proportional connection $2^{n-1}S = aD^j$ exists for all D where a is arbitrary constant. Given binary case when $D=0$ and $S=1$, we have:

$$2^{n-1} = 2^{n-1} * 1 = 2^{n-1}S = aD^j = a * 0^j = 0$$

There is a contradiction, which implies that it is impossible to reduce k into the following form:

$$k_i = \frac{aD^j}{bD^i}$$

Therefore, if k is constant with regard to D then,

$$2D(s(\Omega) - 2^{n-1}) + M(2^n - s(\Omega)) = C \neq 0, \forall D$$

Where C is constant. We have:

$$\begin{aligned} \sum_D (2D(s(\Omega) - 2^{n-1}) + M(2^n - s(\Omega))) &= \sum_D C \\ \Rightarrow 2S(s(\Omega) - 2^{n-1}) + NM(2^n - s(\Omega)) &= NC \\ \Rightarrow 2^n S &= NC \end{aligned}$$

It is implied that

$$k = \frac{2^{n-1}S}{2D(s(\Omega) - 2^{n-1}) + M(2^n - s(\Omega))} = \frac{NC}{2C} = \frac{N}{2}$$

This holds

$$\begin{aligned} 2^n S &= N(2D(s(\Omega) - 2^{n-1}) + M(2^n - s(\Omega))) \\ &= 2ND(s(\Omega) - 2^{n-1}) + 2S(2^n - s(\Omega)) \\ \Rightarrow 2ND(s(\Omega) - 2^{n-1}) - 2S(s(\Omega) - 2^{n-1}) &= 0 \\ \Rightarrow (ND - S)(s(\Omega) - 2^{n-1}) &= 0 \end{aligned}$$

Assuming $ND=S$ we have:

$$ND = S = 2NM \Rightarrow D = 2M$$

There is a contradiction because M is maximum value of D . Therefore, if k is constant with regard to D then $s(\Omega) = 2^{n-1}$. Inversely, if $s(\Omega) = 2^{n-1}$ then k is:

$$k = \frac{2^{n-1}S}{2D(2^{n-1} - 2^{n-1}) + M(2^n - 2^{n-1})} = \frac{N}{2}$$

In general, the event that k is constant with regard to D is equivalent to the event $s(\Omega) = 2^{n-1}$. This implies *diagnostic theorem* stated in table III.1.4.3.1.

Given X-D network is combination of diagnostic relationship and X-gate inference:

$$P(Y = 1|X_1, X_2, \dots, X_n) = P(X_1 \times X_2 \times \dots \times X_n)$$

$$P(D|Y) = \begin{cases} \frac{D}{S} & \text{if } Y = 1 \\ \frac{M}{S} - \frac{D}{S} & \text{if } Y = 0 \end{cases}$$

The diagnostic condition of X-D network is satisfied if and only if

$$s(\Omega) = \sum_a P(Y = 1|a(\Omega)) = 2^{|\Omega|-1}, \forall \Omega \neq \emptyset$$

At that time, the transformation coefficient becomes:

$$k = \frac{N}{2}$$

Note that weights $p_i=w_i$ and $\rho_i=\omega_i$, which are inputs of $s(\Omega)$, are abstract variables. Thus, the equality $s(\Omega) = 2^{|\Omega|-1}$ implies all abstract variables are removed and so $s(\Omega)$ does not depend on weights.

Table III.1.4.3.1. Diagnostic theorem

The diagnostic theorem is the optimal way to validate the diagnostic condition.

The formula III.1.4.3.4 becomes simple with XAND-gate inference. Recall that formula III.1.4.2.8 specified AND-gate inference as follows:

$$P(X_1 \odot X_2 \odot \dots \odot X_n) = P(Y = 1 | X_1, X_2, \dots, X_n)$$

$$= \begin{cases} \prod_{i=1}^n p_i & \text{if all } X_i (s) \text{ are 1} \\ 0 & \text{if there exists at least one } X_i = 0 \end{cases}$$

Due to only one case $X_1 = X_2 = \dots = X_n = 1$, we have:

$$s(\Omega) = s(\Omega: \{X_i = 1\}) = \prod_{i=1}^n p_i$$

Due to $X_i = 0$, we have:

$$s(\Omega: \{X_i = 0\}) = 0$$

Derived from formula III.1.4.3.4, formula III.1.4.3.5 specifies conditional probability $P(D|X_i)$, posterior probability $P(X_i|D)$, and transformation coefficient according to X-D network with AND-gate reference called *AND-D network*.

$$P(D|X_i = 1) = \frac{(2D - M) \prod_{i=1}^n p_i + 2^{n-1}(M - D)}{2^{n-1}S}$$

$$P(D|X_i = 0) = \frac{M - D}{S}$$

$$P(X_i = 1|D) = \frac{(2D - M) \prod_{i=1}^n p_i + 2^{n-1}(M - D)}{(2D - M) \prod_{i=1}^n p_i + 2^n(M - D)}$$

$$P(X_i = 0|D) = \frac{2^{n-1}(M - D)}{(2D - M) \prod_{i=1}^n p_i + 2^n(M - D)}$$

$$k = \frac{2^{n-1}S}{(2D - M) \prod_{i=1}^n p_i + 2^n(M - D)}$$

Formula III.1.4.3.5. Conditional probability and posterior probability of AND-D network

For convenience, we validate diagnostic condition with a case of two sources $\Omega = \{X_1, X_2\}$, $p_1 = p_2 = w_1 = w_2 = 0.5$, $D \in \{0, 1, 2, 3\}$. According to diagnostic theorem stated in table III.1.4.3.1, if $s(\Omega) \neq 2$ for given X-gate then, such X-gate does not satisfy diagnostic condition.

Given AND-gate inference, by applying formula III.1.4.2.8, we have:

$$s(\Omega) = (0.5 * 0.5) + 0 + 0 + 0 = 0.25$$

Given OR-gate inference, by applying formula III.1.4.2.10, we have:

$$s(\Omega) = (1 - 0.5 * 0.5) + (1 - 0.5) + (1 - 0.5) + 0 = 3 - 3 * 0.5 * 0.5 = 1.75$$

Given XOR-gate inference, by applying formula III.1.4.2.13, we have:

$$s(\Omega) = (0.5 * 0.5 + 0.5 * 0.5) + 0.5 + 0.5 + 0 = 1.5$$

Given XNOR-gate inference, by applying formula III.1.4.2.15, we have:

$$s(\Omega) = (0.5 * 0.5 + 0.5 * 0.5) + 0.5 + 0.5 + 1 = 2.5$$

Given SIGMA-gate inference, by applying formula III.1.4.2.19, we have:

$$s(\Omega) = (0.5 + 0.5) + 0.5 + 0.5 + 0 = 2$$

It is asserted that AND-gate, OR-gate, XOR-gate, and XNOR-gate do not satisfy diagnostic condition and so they should not be used to assess hypotheses. However, it is not asserted if U-gate and SIGMA-gate satisfy such diagnostic condition. It is necessary to expand formula for SIGMA-gate diagnostic network (called *SIGMA-D network*) in order to validate it.

In case of SIGMA-gate inference, by applying formula III.1.4.2.19, we have:

$$\sum_i w_i = 1$$

$$s(\Omega) = 2^{n-1} \sum_i w_i = 2^{n-1}$$

$$s(\Omega: \{X_i = 1\}) = 2^{n-1}w_i + 2^{n-2} \sum_{j \neq i} w_j = 2^{n-1}w_i + 2^{n-2}(1 - w_i) = 2^{n-2}(1 + w_i)$$

$$s(\Omega: \{X_i = 0\}) = s(\Omega) - s(\Omega: \{X_i = 1\}) = 2^{n-2}(1 - w_i)$$

It is necessary to validate SIGMA-D network with SIGMA-gate bi-inference. By applying formula 3.22, we recalculate these quantities as follows:

$$s(\Omega) = 2^{n-1} \sum_i w_i + 2^{n-1} \sum_i d_i = 2^{n-1} \sum_i (w_i + d_i) = 2^{n-1}$$

$$\left(\text{due to } \sum_i (w_i + d_i) = 1 \right)$$

$$s(\Omega: \{X_i = 1\}) = 2^{n-1}w_i + 2^{n-2} \sum_{j \neq i} w_j + 2^{n-2} \sum_i d_i$$

$$= 2^{n-2}w_i + 2^{n-2} \sum_i (w_i + d_i) = 2^{n-2}(1 + w_i)$$

$$s(\Omega: \{X_i = 0\}) = s(\Omega) - s(\Omega: \{X_i = 1\}) = 2^{n-2}(1 - w_i)$$

Obviously, quantities $s(\Omega)$, $s(\Omega: \{X_i=1\})$, and $s(\Omega: \{X_i=0\})$ are kept intact. According to diagnostic theorem, we conclude that SIGMA-D network does satisfy diagnostic condition due to $s(\Omega)=2^{n-1}$. Thus, SIGMA-D network can be used to assess hypotheses.

Formula III.1.4.3.6, an immediate consequence of formula III.1.4.3.4, specifies conditional probability $P(D|X_i)$, posterior probability $P(X_i|D)$, and transformation coefficient for SIGMA-D network.

$$P(D|X_i = 1) = \frac{(2D - M)w_i + M}{2S}$$

$$P(D|X_i = 0) = \frac{(M - 2D)w_i + M}{2S}$$

$$P(X_i = 1|D) = \frac{(2D - M)w_i + M}{2M}$$

$$P(X_i = 0|D) = \frac{(M - 2D)w_i + M}{2M}$$

$$k = \frac{N}{2}$$

Formula III.1.4.3.6. Conditional probability, posterior probability, and transformation coefficient of SIGMA-D network

In case of SIGMA-gate, the augmented variable Y can be removed from X-D network. The evidence D is now established as direct target variable. Figure III.1.4.3.2 shows a so-called *direct SIGMA-gate diagnostic network* (direct SIGMA-D network).

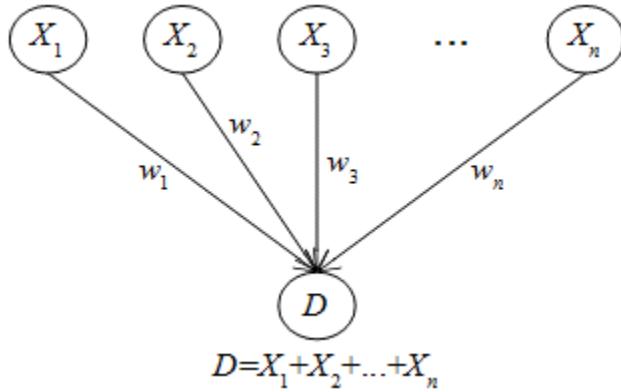


Figure III.1.4.3.2. Direct SIGMA-gate diagnostic network (direct SIGMA-D network)

Derived from formula III.1.4.2.19, the CPT of direct SIGMA-D network is determined by formula III.1.4.3.7.

$$P(D|X_1, X_2, \dots, X_n) = \sum_{i \in K} \frac{D}{S} w_i + \sum_{j \in L} \frac{M - D}{S} w_j$$

Where the set of X_i (s) is complete and mutually exclusive.

$$\sum_{i=1}^n w_i = 1$$

$$X_i \cap X_j = \emptyset, \forall i \neq j$$

Formula III.1.4.3.7. CPT of direct SIGMA-D network

The formula III.1.4.3.7 specifies valid CPT due to:

$$\begin{aligned} \sum_D P(D|X_1, X_2, \dots, X_n) &= \frac{1}{S} \sum_{i \in K} w_i \sum_D D + \frac{1}{S} \sum_{j \in L} w_j \sum_D (M - D) \\ &= \frac{1}{S} \sum_{i \in K} S w_i + \frac{1}{S} \sum_{j \in L} w_j (NM - S) = \frac{1}{S} \sum_{i \in K} S w_i + \frac{1}{S} \sum_{j \in L} S w_j = \sum_{i=1}^n w_i \\ &= 1 \end{aligned}$$

From dependencies shown in figure III.1.4.3.2, formula III.1.4.3.8 specifies the joint probability of direct SIGMA-D network.

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

Formula III.1.4.3.8. Joint probability of direct SIGMA-D network

Inferred from formula III.1.4.3.3, formula III.1.4.3.9 specifies the joint probability $P(X_i, D)$ and the marginal probability $P(D)$ of direct SIGMA-D network, given uniform distribution of all sources.

$$\begin{aligned} P(X_i, D) &= \frac{1}{2^n} s(\Omega: \{X_i\}) \\ P(D) &= \frac{1}{2^n} s(\Omega) \end{aligned}$$

Formula III.1.4.3.9. Joint probability $P(X_i, D)$ and marginal probability $P(D)$ of direct SIGMA-D network

Where $s(\Omega)$ and $s(\Omega: \{X_i\})$ are specified in table III.1.4.2.1.

By browsing all variables of direct SIGMA-D network, we have:

$$\begin{aligned} s(\Omega: \{X_i = 1\}) &= 2^{n-1} \frac{D}{S} w_i + 2^{n-2} \sum_{j \neq i} \frac{D}{S} w_j + 2^{n-2} \sum_{j \neq i} \frac{M-D}{S} w_j \\ &= \frac{2^{n-2}}{S} \left(2Dw_i + M \sum_{j \neq i} w_j \right) = \frac{2^{n-2}}{S} (2Dw_i + M(1 - w_i)) \\ &\quad \left(\text{Due to } \sum_{i=1}^n w_i = 1 \right) \\ &= \frac{2^{n-2}}{S} ((2D - M)w_i + M) \end{aligned}$$

Similarly, we have:

$$\begin{aligned} s(\Omega: \{X_i = 0\}) &= 2^{n-1} \frac{M-D}{S} w_i + 2^{n-2} \sum_{j \neq i} \frac{M-D}{S} w_j + 2^{n-2} \sum_{j \neq i} \frac{D}{S} w_j \\ &= \frac{2^{n-2}}{S} ((M - 2D)w_i + M) \\ s(\Omega) &= 2^{n-1} \sum_i \frac{D}{S} w_i + 2^{n-1} \sum_i \frac{M-D}{S} w_i = \frac{2^{n-1}M}{S} \end{aligned}$$

By applying formula III.1.4.3.9, $s(\Omega: \{X_i=0\})$, $s(\Omega: \{X_i=1\})$, and $s(\Omega)$, we get the same result with formula III.1.4.3.6.

$$\begin{aligned} P(D|X_i = 1) &= \frac{(2D - M)w_i + M}{2S} \\ P(D|X_i = 0) &= \frac{(M - 2D)w_i + M}{2S} \\ P(X_i = 1|D) &= \frac{(2D - M)w_i + M}{2M} \\ P(X_i = 0|D) &= \frac{(M - 2D)w_i + M}{2M} \\ k &= \frac{N}{2} \end{aligned}$$

Therefore, it is possible to use direct SIGMA-D network to assess hypotheses. It is asserted that SIGMA-D network satisfy diagnostic condition when single relationship, NOT-D network, direct SIGMA-D network are specific cases of SIGMA-D network. There is a question: Does an X-D network which is different from SIGMA-D network and not aforementioned exist such that it satisfies diagnostic condition?

Recall that each X-D network is a pattern owning a particular X-gate inference which in turn is based on particular X-gate condition (s) relevant to only variables A_i (s). The most general nonlinear X-D network is U-D network whereas SIGMA-D network is linear one. The U-gate inference given arbitrary condition on U is:

$$P(X_1 \sqcup X_2 \sqcup \dots \sqcup X_n) = \sum_{U \in \mathcal{U}} \left(\prod_{i \in U \cap K} p_i \prod_{i \in U \cap L} (1 - p_i) \right) \left(\prod_{i \in \bar{U} \cap K} (1 - p_i) \prod_{i \in \bar{U} \cap L} p_i \right)$$

Let f be the arrangement sum of U-gate inference.

$$f(p_i, \rho_i) = \sum_{a(\Omega)} \sum_{U \in \mathcal{U}} \left(\prod_{i \in U \cap K} p_i \prod_{i \in U \cap L} (1 - \rho_i) \right) \left(\prod_{i \in \bar{U} \cap K} (1 - p_i) \prod_{i \in \bar{U} \cap L} \rho_i \right)$$

The function f is sum of many large expressions and each expression is product of four possible sub-products (Π) as follows:

$$Expr = \prod_{i \in U \cap K} p_i \prod_{i \in U \cap L} (1 - \rho_i) \prod_{i \in \bar{U} \cap K} (1 - p_i) \prod_{i \in \bar{U} \cap L} \rho_i$$

In any case of degradation, there always exist expression $Expr$ (s) having at least 2 sub-products (Π), for example:

$$Expr = \prod_{i \in U \cap K} p_i \prod_{i \in U \cap L} (1 - \rho_i)$$

Consequently, there always exist $Expr$ (s) having at least 5 terms relevant to p_i and ρ_i if $n \geq 5$, for example:

$$Expr = p_1 p_2 p_3 (1 - \rho_4) (1 - \rho_5)$$

Thus, degree of f will be larger than or equal to 5 given $n \geq 5$. According to diagnostic theorem, U-gate network satisfies diagnostic condition if and only if $f(p_i, \rho_i) = 2^{n-1}$ for all $n \geq 1$ and for all abstract variables p_i and ρ_i . Without loss of generality, each p_i or ρ_i is sum of variable x and a variable a_i or b_i , respectively. Note that all p_i, ρ_i, a_i are b_i are abstract variables.

$$\begin{aligned} p_i &= x + a_i \\ \rho_i &= x + b_i \end{aligned}$$

The equation $f - 2^{n-1} = 0$ becomes equation $g(x) = 0$ whose degree is $m \geq 5$ if $n \geq 5$.

$$g(x) = \pm x^m + C_1 x^{m-1} + \dots + C_{m-1} x + C_m - 2^{n-1} = 0$$

Where coefficients C_i (s) are functions of a_i and b_i (s). According to Abel-Ruffini theorem (Wikipedia, Abel-Ruffini theorem, 2016), equation $g(x) = 0$ has no algebraic solution when $m \geq 5$. Thus, abstract variables p_i and ρ_i cannot be eliminated entirely from $g(x)=0$, which causes that there is no specification of U-gate inference $P(X_1 x X_2 x \dots x X_n)$ so that diagnostic condition is satisfied.

It is concluded that there is no nonlinear X-D network satisfying diagnostic condition but a new question is raised: Does there exist the general linear X-D network satisfying diagnostic condition? Such linear network is called GL-D network and SIGMA-D network is specific case of GL-D network. The GL-gate probability must be linear combination of weights.

$$P(X_1 x X_2 x \dots x X_n) = C + \sum_{i=1}^n \alpha_i w_i + \sum_{i=1}^n \beta_i d_i$$

Where C is arbitrary constant.

The GL-gate inference is singular if α_i and β_i are functions of only X_i as follows:

$$P(X_1 x X_2 x \dots x X_n) = C + \sum_{i=1}^n h_i(X_i) w_i + \sum_{i=1}^n g_i(X_i) d_i$$

The functions h_i and g_i are not relevant to A_i because the final formula of GL-gate inference is only relevant to X_i (s) and weights (s). Because GL-D network is a pattern, we only survey singular GL-gate. Mentioned GL-gate is singular by default and it is dependent on how to define functions h_i and g_i . The arrangement sum with regard to GL-gate is:

$$s(\Omega) = \sum_a \left(C + \sum_{i=1}^n h_i(X_i) w_i + \sum_{i=1}^n g_i(X_i) d_i \right)$$

$$= 2^n C + 2^{n-1} \sum_{i=1}^n (h_i(X_i = 1) + h_i(X_i = 0)) w_i \\ + 2^{n-1} \sum_{i=1}^n (g_i(X_i = 1) + g_i(X_i = 0)) d_i$$

Suppose h_i and g_i are probability mass functions with regard to X_i . For all i , we have:

$$\begin{aligned} 0 &\leq h_i(X_i) \leq 1 \\ 0 &\leq g_i(X_i) \leq 1 \\ h_i(X_i = 1) + h_i(X_i = 0) &= 1 \\ g_i(X_i = 1) + g_i(X_i = 0) &= 1 \end{aligned}$$

The arrangement sum becomes:

$$s(\Omega) = 2^n C + 2^{n-1} \sum_{i=1}^n (w_i + d_i)$$

GL-D network satisfies diagnostic condition if

$$\begin{aligned} s(\Omega) &= 2^n C + 2^{n-1} \sum_{i=1}^n (w_i + d_i) = 2^{n-1} \\ \Rightarrow 2C + \sum_{i=1}^n (w_i + d_i) &= 1 \end{aligned}$$

Suppose the set of X_i (s) is complete.

$$\sum_{i=1}^n (w_i + d_i) = 1$$

This implies $C=0$. Shortly, formula III.1.4.3.10 specifies the singular GL-gate inference so that GL-D network satisfies diagnostic condition.

$$P(X_1 \times X_2 \times \dots \times X_n) = \sum_{i=1}^n h_i(X_i) w_i + \sum_{i=1}^n g_i(X_i) d_i$$

Where h_i and g_i are probability mass functions and the set of X_i (s) is complete.

$$\sum_{i=1}^n W_i = 1$$

Formula III.1.4.3.10. Singular GL-gate inference

Functions $h_i(X_i)$ and $g_i(X_i)$ are always linear due to $X_i^m = X_i$ for all $m \geq 1$ when X_i is binary. It is easy to infer that SIGMA-D network is GL-D network with following definition of functions h_i and g_i .

$$h_i(X_i) = 1 - g_i(X_i) = X_i, \forall i$$

According to Millán and Pérez-de-la-Cruz (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002), a hypothesis can have multiple evidences as seen in figure III.1.4.3.3. This is *multi-evidence diagnostic relationship* opposite to aforementioned multi-hypothesis diagnostic relationship.

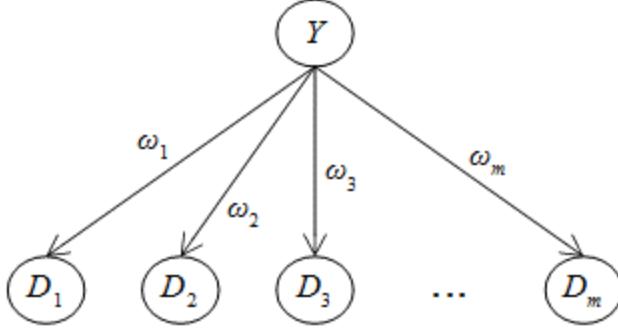


Figure III.1.4.3.3. Diagnostic relationship with multiple evidences (M-E-D network)

Figure III.1.4.3.3 depicts the multi-evidence diagnostic network called M-E-D network in which there are m evidences D_1, D_2, \dots, D_m and one hypothesis Y . Note that Y has uniform distribution.

In simplest case where all evidences are binary, the joint probability of M-E-D network is:

$$P(Y, D_1, D_2, \dots, D_m) = P(Y) \prod_{j=1}^m P(D_j|Y) = P(Y)P(D_1, D_2, \dots, D_m|Y)$$

The product $\prod_{j=1}^m P(D_j|Y)$ is denoted as likelihood function as follows:

$$P(D_1, D_2, \dots, D_m|Y) = \prod_{j=1}^m P(D_j|Y)$$

The posterior probability $P(Y | D_1, D_2, \dots, D_m)$ given uniform distribution of Y is:

$$\begin{aligned} P(Y|D_1, D_2, \dots, D_m) &= \frac{P(Y, D_1, D_2, \dots, D_m)}{P(Y = 1, D_1, D_2, \dots, D_m) + P(Y = 0, D_1, D_2, \dots, D_m)} \\ &= \frac{1}{\prod_{j=1}^m P(D_j|Y = 1) + \prod_{j=1}^m P(D_j|Y = 0)} * P(D_1, D_2, \dots, D_m|Y) \end{aligned}$$

The possible transformation coefficient is:

$$\frac{1}{k} = \prod_{j=1}^m P(D_j|Y = 1) + \prod_{j=1}^m P(D_j|Y = 0)$$

M-E-D network will satisfy diagnostic condition if $k = 1$ because all hypotheses and evidence are binary, which leads that following equation specified by formula III.1.4.3.11 has $2m$ real roots $P(D_j|Y)$ for all $m \geq 2$.

$$\prod_{j=1}^m P(D_j|Y = 1) + \prod_{j=1}^m P(D_j|Y = 0) = 1$$

Formula III.1.4.3.11. Diagnostic condition equation of binary M-E-D network

The equation III.1.4.3.11 has no real root given $m=2$ according to following proof. Suppose equation III.1.4.3.11 has 4 real roots as follows:

$$\begin{aligned} a_1 &= P(D_1 = 1|Y = 1) \\ a_2 &= P(D_2 = 1|Y = 1) \\ b_1 &= P(D_1 = 1|Y = 0) \\ b_2 &= P(D_2 = 1|Y = 0) \end{aligned}$$

From equation III.1.4.3.11, it holds:

$$\begin{cases} a_1 a_2 + b_1 b_2 = 1 \\ a_1(1 - a_2) + b_1 b_2 = 1 \\ (1 - a_1)a_2 + b_1 b_2 = 1 \\ a_1 a_2 + b_1(1 - b_2) = 1 \\ a_1 a_2 + (1 - b_1)b_2 = 1 \end{cases} \Rightarrow \begin{cases} a_1 = a_2 \\ b_1 = b_2 \\ a_1^2 + b_1^2 = 1 \\ a_1 + 2b_1^2 = 2 \\ b_1 + 2a_1^2 = 2 \end{cases} \Leftrightarrow \begin{cases} a_1 = a_2 = 0 \\ b_1 = b_2 \\ a_1^2 + b_1^2 = 1 \\ b_1 = 2 \end{cases} \text{ or } \begin{cases} a_1 = a_2 = 0.5 \\ b_1 = b_2 \\ a_1^2 + b_1^2 = 1 \\ b_1 = 1.5 \end{cases}$$

The final equation leads a contradiction ($b_1=2$ or $b_1=1.5$) and so it is impossible to apply the sufficient diagnostic proposition into M-E-D network. Such proposition is only used for one-evidence network. Moreover, X-gate inference absorbs many sources and then produces out of one targeted result whereas the M-E-D network essentially splits one source into many results. It is impossible to model M-E-D network by X-gates. The potential solution for this problem is to group many evidences D_1, D_2, \dots, D_m into one representative evidence D which in turn is dependent on hypothesis Y but this solution will be inaccurate in specifying conditional probabilities because directions of dependencies become inconsistent (relationships from D_j to D and from Y to D) except that all D_j (s) are removed and D becomes a vector. However evidence vector does not simplify the hazardous problem and it changes the current problem into a new problem.

Another solution is to reverse the direction of relationship, in which the hypothesis is dependent on evidences so as to take advantages of X-gate inference as usual. However, the reversion method violates the viewpoint in this research where diagnostic relationship must be from hypothesis to evidence. In other words, we should change the viewpoint.

Another solution is based on a so-called *partial diagnostic condition* that is a loose case of diagnostic condition for M-E-D network, which is defined as follows:

$$P(Y|D_j) = kP(D_j|Y)$$

Where k is constant with regard to D_j . The joint probability is:

$$P(Y, D_1, D_2, \dots, D_m) = P(Y) \prod_{j=1}^m P(D_j|Y)$$

M-E-D network satisfies partial diagnostic condition. In fact, given all variables are binary, we have:

$$\begin{aligned} P(Y|D_j) &= \frac{\sum_{\Psi \setminus \{Y, D_j\}} P(Y, D_1, D_2, \dots, D_m)}{\sum_{\Psi \setminus \{D_j\}} P(Y, D_1, D_2, \dots, D_m)} \\ &\quad (\text{Let } \Psi = \{D_1, D_2, \dots, D_m\}) \\ &= \frac{P(D_j|Y) \prod_{k=1, k \neq j}^m (\sum_{D_k} P(D_k|Y))}{\prod_{k=1, k \neq j}^m (\sum_{D_k} P(D_k|Y = 1)) + \prod_{k=1, k \neq j}^m (\sum_{D_k} P(D_k|Y = 0))} \\ &\quad (\text{Due to uniform distribution of } Y) \\ &= \frac{P(D_j|Y) \prod_{k=1, k \neq j}^m 1}{\prod_{k=1, k \neq j}^m 1 + \prod_{k=1, k \neq j}^m 1} = \frac{1}{2} P(D_j|Y) \\ &\quad \left(\text{Due to } \sum_{D_k} P(D_k|Y) = P(D_k = 0|Y) + P(D_k = 1|Y) = 1 \right) \end{aligned}$$

Partial diagnostic condition expresses a different viewpoint. It is not an optimal solution because we cannot test a disease based on only one symptom while ignoring other obvious symptoms, for example. The equality $P(Y/D_j) = 0.5P(D_j/Y)$ indicates the accuracy is decreased two times. However, Bayesian network provides inference

mechanism based on personal belief. It is subjective. You can use partial diagnostic condition if you think that such condition is appropriate to your application.

If we are successful in specifying conditional probabilities of M-E-D network, it is possible to define an extended network which is constituted of n hypotheses X_1, X_2, \dots, X_n and m evidences D_1, D_2, \dots, D_m . Such extended network represents *multi-hypothesis multi-evidence diagnostic relationship*, called M-HE-D network. Figure III.1.4.3.4 depicts M-HE-D network.

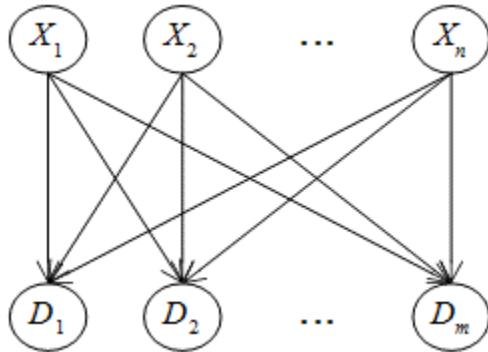


Figure III.1.4.3.4. M-HE-D network

The M-HE-D network is the most general case of diagnostic network, which is also mentioned in (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation, 2002, p. 297). We can construct any large diagnostic BN from M-HE-D networks and so the research is still open.

In short, relationship conversion is to determine conditional probabilities based on logic gates that is adhered to semantics of relationships. The weak point of logic gates is to require that all variables must be binary. For example, in learning context, it is inconvenient for expert to create an assessment BN with studying exercises (evidences) whose marks are only 0 and 1. In order to lessen the impact of such weak point, I use numeric evidence for extending capacity of simple Bayesian network. However, combination of binary hypothesis and numeric evidence leads to errors or biases in inference. For example, given a student gets maximum grade for an exercise but the built-in inference results out that she/he has not mastered fully the associated learning concept (hypothesis). Therefore, I propose the sufficient diagnostic proposition so as to confirm that numeric evidence is adequate to make complicated inference tasks in BN. The probabilistic reasoning based on evidence is always accurate. Application of the research can go beyond learning context whenever probabilistic deduction relevant to constraints of semantic relationships is required. A large BN can be constituted of many simple BN (s). Inference in large BN is hazardous problem and there are many optimal algorithms for solving such problem. In future, I will research effective inference methods for the special BN that is constituted of X-gate BN (s) mentioned in this research because X-gate BN (s) have precise and useful features of which we should take advantages. For instance, their CPT (s) are simple in some cases and the meanings of their relationships are mandatory in many applications. Moreover, I will try my best to research deeply M-E-D network and M-HE-D network whose problems I cannot solve absolutely now.

Two main documents I referred to do the sub-section III.1.4 are the book “Learning Bayesian Networks” (Neapolitan, 2003) by Richard E. Neapolitan and the article “A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation” (Millán & Pérez-de-la-Cruz, A Bayesian Diagnostic Algorithm for Student Modeling

and its Evaluation, 2002) by Eva Millán and José Luis Pérez-de-la-Cruz. Especially, the SIGMA-gate inference is based on and derived from the work of Eva Millán and José Luis Pérez-de-la-Cruz. The sub-section III.1.4 is originated from my PhD research “A User Modeling System for Adaptive Learning” (Nguyen, A User Modeling System for Adaptive Learning, 2014). Other references relevant to user modeling, overlay model, and Bayesian network are (Fröschl, 2005), (De Bra, Smits, & Stash, 2006), (Murphy, 1998), and (Heckerman, 1995). Please concern these references.

The sub-section III.1.4 ends up the proposed method of combination between overlay model equipped semantic relationships and BN equipped with CPT (s). Next sub-section III.1.5 is the evaluation of proposed method.

III.1.5. 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 section III.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 sections III.4 and III.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 section III.2.

III.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 authors Paul De Bra and Licia Calvi (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 or not user visited the page associated with concept.

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.2, user model is implicated as overlay model.

Adaptation model (AM) is responsible for associating user model with domain model in order to generate adaptation. AM contains a set of adaptation rules used to tailor learning concepts and materials to user characteristics in user model. Rules are categorized into two classes associated with 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 to 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 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:	<i>access(C)</i>
Condition:	<i>C.visited = true</i>
Action:	<i>C.knowledge = 100%</i>

The adaptation rules whose purpose is top defined 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:	<i>C.knowledge > 50</i>
Action:	True status: providing advanced subjects about <i>C</i>
	False status: providing basic explanations about <i>C</i>

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” become *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 by new value (“*true*”). This technique is called forward reasoning because the

“*recommended*” attribute is determined as soon as possible 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 forward 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.1.2](#) is constituted of three concepts considered as knowledge variables whose links are aggregation relationships. Additionally, there is an evidence variable: “*Exercise: set up class Human*”. The evidence “*Exercise: set up class Human*” proves whether or not she/he understands concept “*Class & Object*”. The number (in range 0...1) that measures the relative importance of each aggregation or evidence is defined as the weight of arc from parent node to child node. All weights concerning the child variable are normalized and used to build up its CPT. Figure [III.2.1](#) 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.1.2](#). In order to be convenient for reader to keep tract important problems, figure [III.2.1](#) is a repetition of figure [III.1.2.2](#).

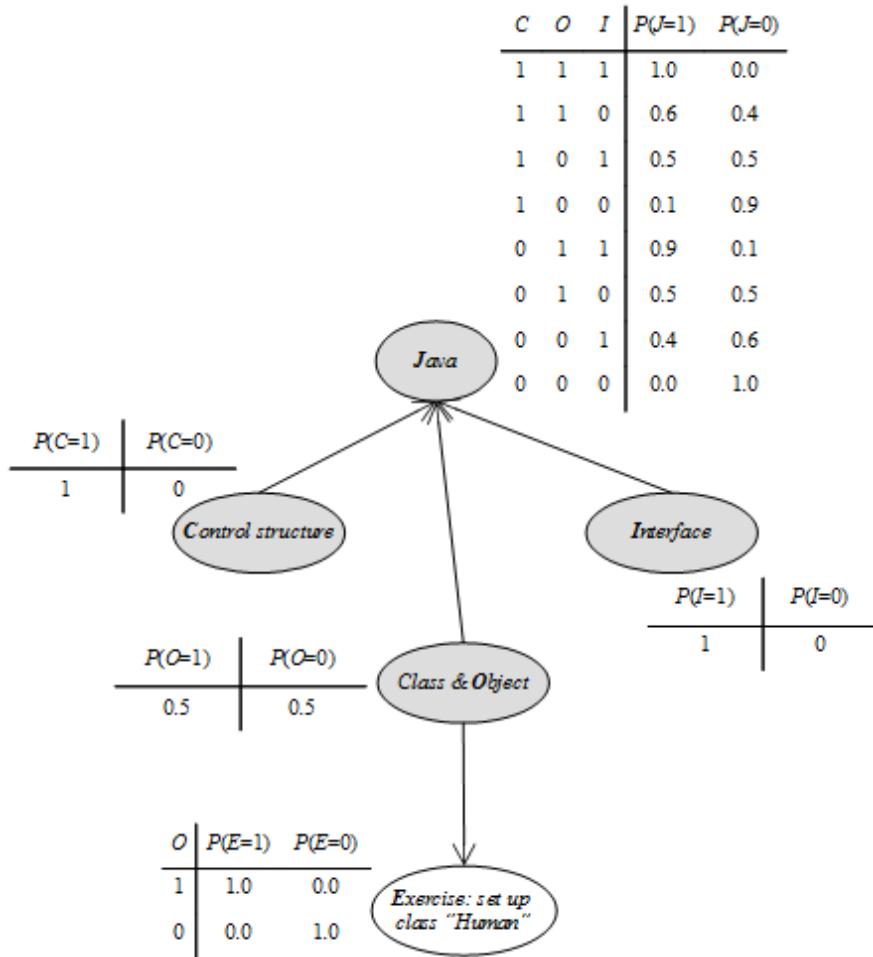


Figure III.2.1. Bayesian overlay model of Java course with full of CPT (s)

Note that five concepts (nodes) O, I, J, E denote knowledge variables and evidence: *Class & Object*, *Java*, *Interface*, “*Exercise: set up class Human*”, respectively. Concept E has a new attribute *is_evidence* indicating whether E is fit to become evidences or not. Concepts O, I, J have the new dummy attribute *do\$bayes\$infer*. It is 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 that tells us how mastered leaner is over the concept J . Please see sub-section III.1.1 for knowing posterior probability.

I defined one ECA rule for updating evidence “*Exercise: set up class Human*” and one CA rule for setting presentation specification.

- The 1st ECA rule: If user does exercise “*Exercise: set up class Human*” the attribute *E.is_evidence* is set to be *true*.
- The 2nd 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.2.1 as follows:

	Event	Condition	Action
Rule 1 st	$\text{access}(E)$	$E.\text{visited} \geq 1$	$E.\text{is_evidence} = \text{true}$

Rule 2 nd		$J.do$bayes$infer \geq 0.5$	<ul style="list-style-type: none"> - <i>True status</i>: providing advanced subjects about Java. - <i>False status</i>: providing basic explanations about Java.
----------------------	--	-----------------------------	--

Table III.2.1. ECA and CA rules for Bayesian inference

The 2nd rule is translated in XML form as follows:

```
<if expr="J.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 2nd rule, attribute $J.do$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 1st rule is executed, concept E becomes actual evidences. At that time, random variable E is equal to 1; we have $E=1$. According to the formula III.1.1.10 for calculating posterior probability in Bayesian network, the attribute $J.do$bayes$infer$ that represents user's knowledge about concept J is determined as follows:

$$J.do$bayes$infer = P(J = 1|E = 1) = \frac{\sum_{C,O,I} P(J = 1, C, O, I, E = 1)}{\sum_{J,C,O,I} P(J, C, O, I, E = 1)}$$

Formula III.2.1. Attribute $J.do$bayes$infer$ represents posterior probability

Where $P(J, C, O, I, E)$ is global joint probability distribution. Note that variables J , C , O , I , and E are binary variables whose values are 0 or 1. According to formula III.1.1.8 for specifying global joint probability distribution in effective way, we have:

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

The numerator of formula III.2.1 is:

$$\begin{aligned}
 & \sum_{C,O,I} P(J = 1, C, O, I, E = 1) \\
 &= P(J = 1, C = 1, O = 1, I = 1, E = 1) \\
 &+ P(J = 1, C = 1, O = 1, I = 0, E = 1) \\
 &+ P(J = 1, C = 1, O = 0, I = 1, E = 1) \\
 &+ P(J = 1, C = 1, O = 0, I = 0, E = 1) \\
 &+ P(J = 1, C = 0, O = 1, I = 1, E = 1) \\
 &+ P(J = 1, C = 0, O = 1, I = 0, E = 1) \\
 &+ P(J = 1, C = 0, O = 0, I = 1, E = 1) \\
 &+ P(J = 1, C = 0, O = 0, I = 0, E = 1) \\
 &= P(C = 1) * P(O = 1) * P(I = 1) * P(E = 1|O = 1) \\
 &\quad * P(J = 1|C = 1, O = 1, I = 1) \\
 &+ P(C = 1) * P(O = 1) * P(I = 0) * P(E = 1|O = 1) \\
 &\quad * P(J = 1|C = 1, O = 1, I = 0)
 \end{aligned}$$

$$\begin{aligned}
 & +P(C = 1) * P(O = 0) * P(I = 1) * P(E = 1|O = 0) \\
 & \quad * P(J = 1|C = 1, O = 0, I = 1) \\
 & +P(C = 1) * P(O = 0) * P(I = 0) * P(E = 1|O = 0) \\
 & \quad * P(J = 1|C = 1, O = 0, I = 0) \\
 & +P(C = 0) * P(O = 1) * P(I = 1) * P(E = 1|O = 1) \\
 & \quad * P(J = 1|C = 1, O = 1, I = 1) \\
 & +P(C = 0) * P(O = 1) * P(I = 0) * P(E = 1|O = 1) \\
 & \quad * P(J = 1|C = 1, O = 1, I = 0) \\
 & +P(C = 0) * P(O = 0) * P(I = 1) * P(E = 1|O = 0) \\
 & \quad * P(J = 1|C = 1, O = 0, I = 1) \\
 & +P(C = 0) * P(O = 0) * P(I = 0) * P(E = 1|O = 0) \\
 & \quad * P(J = 1|C = 1, O = 0, I = 0) \\
 & = 1 * 0.5 * 1 * 1 * 1 \\
 & +1 * 0.5 * 0 * 1 * 0.6 \\
 & +1 * 0.5 * 1 * 0 * 0.5 \\
 & +1 * 0.5 * 0 * 0 * 0.1 \\
 & +0 * 0.5 * 1 * 1 * 0.9 \\
 & +0 * 0.5 * 0 * 1 * 0.5 \\
 & +0 * 0.5 * 1 * 0 * 0.4 \\
 & +0 * 0.5 * 0 * 0 * 0 \\
 & = 0.5
 \end{aligned}$$

The denominator of formula III.2.1 is:

$$\begin{aligned}
 & \sum_{J,C,O,I} P(J, C, O, I, E = 1) \\
 & = P(J = 1, C = 1, O = 1, I = 1, E = 1) \\
 & +P(J = 1, C = 1, O = 1, I = 0, E = 1) \\
 & +P(J = 1, C = 1, O = 0, I = 1, E = 1) \\
 & +P(J = 1, C = 1, O = 0, I = 0, E = 1) \\
 & +P(J = 1, C = 0, O = 1, I = 1, E = 1) \\
 & +P(J = 1, C = 0, O = 1, I = 0, E = 1) \\
 & +P(J = 1, C = 0, O = 0, I = 1, E = 1) \\
 & +P(J = 1, C = 0, O = 0, I = 0, E = 1) \\
 & +P(J = 0, C = 1, O = 1, I = 1, E = 1) \\
 & +P(J = 0, C = 1, O = 1, I = 0, E = 1) \\
 & +P(J = 0, C = 1, O = 0, I = 1, E = 1) \\
 & +P(J = 0, C = 1, O = 0, I = 0, E = 1) \\
 & +P(J = 0, C = 0, O = 1, I = 1, E = 1) \\
 & +P(J = 0, C = 0, O = 1, I = 0, E = 1) \\
 & +P(J = 0, C = 0, O = 0, I = 1, E = 1) \\
 & +P(J = 0, C = 0, O = 0, I = 0, E = 1) \\
 & = P(C = 1) * P(O = 1) * P(I = 1) * P(E = 1|O = 1) \\
 & \quad * P(J = 1|C = 1, O = 1, I = 1) \\
 & +P(C = 1) * P(O = 1) * P(I = 0) * P(E = 1|O = 1) \\
 & \quad * P(J = 1|C = 1, O = 1, I = 0) \\
 & +P(C = 1) * P(O = 0) * P(I = 1) * P(E = 1|O = 0) \\
 & \quad * P(J = 1|C = 1, O = 0, I = 1) \\
 & +P(C = 1) * P(O = 0) * P(I = 0) * P(E = 1|O = 0) \\
 & \quad * P(J = 1|C = 1, O = 0, I = 0)
 \end{aligned}$$

$$\begin{aligned}
 & +P(C = 0) * P(O = 1) * P(I = 1) * P(E = 1|O = 1) \\
 & \quad * P(J = 1|C = 1, O = 1, I = 1) \\
 & +P(C = 0) * P(O = 1) * P(I = 0) * P(E = 1|O = 1) \\
 & \quad * P(J = 1|C = 1, O = 1, I = 0) \\
 & +P(C = 0) * P(O = 0) * P(I = 1) * P(E = 1|O = 0) \\
 & \quad * P(J = 1|C = 1, O = 0, I = 1) \\
 & +P(C = 0) * P(O = 0) * P(I = 0) * P(E = 1|O = 0) \\
 & \quad * P(J = 1|C = 1, O = 0, I = 0) \\
 & +P(C = 1) * P(O = 1) * P(I = 1) * P(E = 1|O = 1) \\
 & \quad * P(J = 0|C = 1, O = 1, I = 1) \\
 & +P(C = 1) * P(O = 1) * P(I = 0) * P(E = 1|O = 1) \\
 & \quad * P(J = 0|C = 1, O = 1, I = 0) \\
 & +P(C = 1) * P(O = 0) * P(I = 1) * P(E = 1|O = 0) \\
 & \quad * P(J = 0|C = 1, O = 0, I = 1) \\
 & +P(C = 1) * P(O = 0) * P(I = 0) * P(E = 1|O = 0) \\
 & \quad * P(J = 0|C = 1, O = 0, I = 0) \\
 & +P(C = 0) * P(O = 1) * P(I = 1) * P(E = 1|O = 1) \\
 & \quad * P(J = 0|C = 1, O = 1, I = 1) \\
 & +P(C = 0) * P(O = 1) * P(I = 0) * P(E = 1|O = 1) \\
 & \quad * P(J = 0|C = 1, O = 1, I = 0) \\
 & +P(C = 0) * P(O = 0) * P(I = 1) * P(E = 1|O = 0) \\
 & \quad * P(J = 0|C = 1, O = 0, I = 1) \\
 & +P(C = 0) * P(O = 0) * P(I = 0) * P(E = 1|O = 0) \\
 & \quad * P(J = 0|C = 1, O = 0, I = 0) \\
 & = 1 * 0.5 * 1 * 1 * 1 \\
 & +1 * 0.5 * 0 * 1 * 0.6 \\
 & +1 * 0.5 * 1 * 0 * 0.5 \\
 & +1 * 0.5 * 0 * 0 * 0.1 \\
 & +0 * 0.5 * 1 * 1 * 0.9 \\
 & +0 * 0.5 * 0 * 1 * 0.5 \\
 & +0 * 0.5 * 1 * 0 * 0.4 \\
 & +0 * 0.5 * 0 * 0 * 0 \\
 & 1 * 0.5 * 1 * 1 * 0 \\
 & +1 * 0.5 * 0 * 1 * 0.4 \\
 & +1 * 0.5 * 1 * 0 * 0.5 \\
 & +1 * 0.5 * 0 * 0 * 0.9 \\
 & +0 * 0.5 * 1 * 1 * 0.1 \\
 & +0 * 0.5 * 0 * 1 * 0.5 \\
 & +0 * 0.5 * 1 * 0 * 0.6 \\
 & +0 * 0.5 * 0 * 0 * 1 \\
 & = 0.5
 \end{aligned}$$

The attribute $J.\text{do\$bayes\$infer}$ is determined as follows:

$$J.\text{do\$bayes\$infer} = \frac{\sum_{C,O,I} P(J = 1, C, O, I, E = 1)}{\sum_{J,C,O,I} P(J, C, O, I, E = 1)} = \frac{0.5}{0.5} = 1$$

Due to $J.\text{do\$bayes\$infer} \geq 0.5$, learner is provided advanced subjects about Java according to the 2nd CA rule shown in table III.2.1. The adaptive learning system WOW, a modified version of AHA! system, mainly uses Bayesian network inference for making adaptation by taking advantage of the dummy attribute $J.\text{do\$bayes\$infer}$. Please see sub-section II.2.4 for more details about WOW.

In general, my method (Nguyen L., 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 IV.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.1.2, proved in sub-section III.1.3 and applied in section III.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 section III.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 section III.4.

III.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 section III.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 section III.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 most adaptive learning systems. There are some effective modeling methods such as stereotype, overlay, plan recognition but overlay model is proven soundness due to two its properties: flexible graphic structure and reflecting comprehensibly the domain knowledge in education course. The basic ideology of overlay model is to represent user knowledge as subset of domain model. The combination between overlay model and BN (see section III.1) will make use of each method's strong points and restrain drawbacks.

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

Such model is called Bayesian overlay model, Bayesian model, or Bayesian network in brief with note that knowledge sub-model inside [Triangular Learner Model](#) (TLM) is represented as such Bayesian model. 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.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.3.2](#) mentions how to learn parameters in case of missing data. Sub-section [III.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.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.3.1](#) are found in the book “Learning Bayesian Networks” by the author Neapolitan (Neapolitan, 2003) from page 293 to page 373 but I rearrange them and prove them again by myself with a few changes according to the purpose of this sub-section [III.3.1](#) – the evolution of parameters in BN. Readers are recommended to read the book “Learning Bayesian Networks” in order to understand comprehensively Bayesian network; this is an excellent book to which I referred. I express my deep gratitude to the author Richard E. Neapolitan for providing the great book.

It is conventional that definitions, theorems, corollaries and lemmas are noted as formulas so that it is easy for readers to follow and look up mathematical formulas. The [appendix C](#) lists all formulas in this research.

Parameter 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](#) 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 are positive numbers. Beta density function with two parameters a and b (Neapolitan, 2003, p. 300) is defined in formula [III.3.1.1](#).

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

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

Formula III.3.1.1. 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.3.1.2. Gamma function

It is conventional that $e^{(.)}$ and $\exp(.)$ denote exponent function and $e \approx 2.71828$ is Euler's number. If x is positive integer, gamma function in formula III.3.1.2 is equivalent to factorial function,

$$\Gamma(x) = (x - 1)!$$

There is an important property of gamma function which is expressed in formula III.3.1.3 (Neapolitan, 2003, p. 298).

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

Formula III.3.1.3. Important property of gamma function with regard to factorial function

Figure III.3.1.1 shows beta density function with various parameters a and b . Beta functions $\beta(x;2,2)$, $\beta(x;4,2)$, and $\beta(x;2,4)$ are drawn as black line, green line, and red line, respectively.

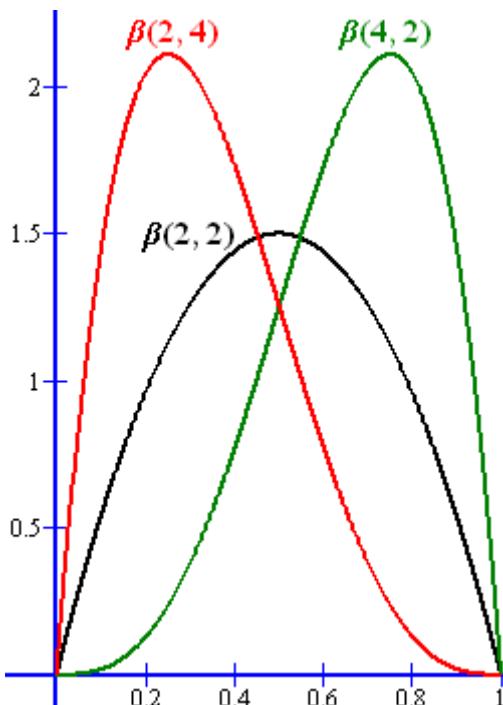


Figure III.3.1.1. Beta density functions with various parameters a and b

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

The integral in interval $[0, 1]$ of the expression $x^{a-1}(1-x)^{b-1}$ inside definition of beta function specified by formula III.3.1.1 is determined by formula III.3.1.4 as follow:

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

Formula III.3.1.4. Integral of product expression $x^a(1-x)^b$

Proof,

$$\begin{aligned}
 \int_0^1 x^a(1-x)^b dx &= \int_0^1 \frac{\Gamma(a+1+b+1)}{\Gamma(a+1)\Gamma(b+1)} x^a(1-x)^b \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+1+b+1)} dx \\
 &= \int_0^1 \frac{\Gamma(a+b+2)}{\Gamma(a+1)\Gamma(b+1)} x^a(1-x)^b \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} dx \\
 &= \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \int_0^1 \frac{\Gamma(a+b+2)}{\Gamma(a+1)\Gamma(b+1)} x^a(1-x)^b dx \\
 &= \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \int_0^1 \beta(x; a+1, b+1) dx \\
 &\quad \text{(due to beta density function specified by formula III.3.1.1)} \\
 &= \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} P(0 \leq x \leq 1) = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \\
 &\quad \text{(due to } P(0 \leq x \leq 1) = 1\text{)}
 \end{aligned}$$

The formula III.3.1.4 is lemma 6.2 in (Neapolitan, 2003, p. 300).

Suppose there is one binary variable X in network and the probability distribution of X is considered as relative frequency having values in space $[0, 1]$ which is the range of variable F . A parameter variable F (whose space is $[0, 1]$, 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.3.1.5. Conditional probability (relative frequency) of X as value of parameter 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.3.1.5, $P(X=1/F) = F$ 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). Figure III.3.1.2 shows the simplest augmented BN. We use binomial sample to learn BN and variable F is essentially the parameter of binomial sampling when a parameter is considered as random variable in Bayesian approach.

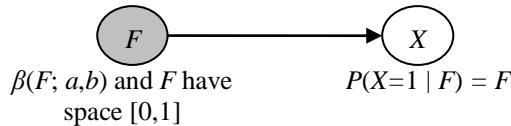


Figure III.3.1.2. The simple augmented BN with only one hypothesis node X

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

Proof, owing to the total probability rule in continuous case (see formula III.1.1.5), we have:

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

Because F is beta function, its expectation is $E(F) = \frac{a}{N}$, 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.3.1.6. Probability of hypothesis X as expectation of beta variable F

Proof,

$$\begin{aligned} P(X = 1) &= E(F) = \int_0^1 F\beta(F)dF = \int_0^1 F \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} F^{a-1}(1-F)^{b-1} df \\ &= \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \int_0^1 F^a(1-F)^{b-1} = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+1)\Gamma(b)}{\Gamma(N+1)} \\ &\quad (\text{due to formula III.3.1.4}) \\ &= \frac{\Gamma(N)}{\Gamma(N+1)} \frac{\Gamma(a+1)}{\Gamma(a)} = \frac{a}{N} \\ &\quad (\text{due to formula III.3.1.3}) \end{aligned}$$

Please pay attention to formula III.3.1.6, it is the most essential formula used over the whole sub-section III.3.1. The formula III.3.1.6 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 prior density function, and $P(X^{(u)} = 1) = a/N$ (due to formula III.3.1.6) 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.3.1.3 depicts this sample $\mathcal{D} = (X^{(1)}, X^{(2)}, \dots, X^{(m)})$.

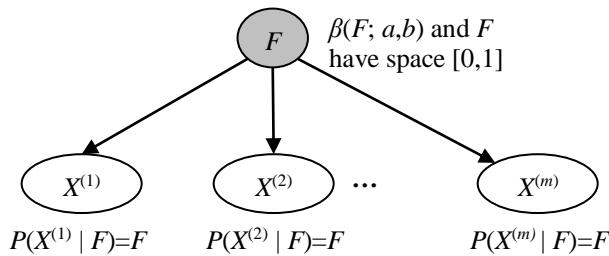


Figure III.3.1.3. The sample $\mathcal{D} = (X^{(1)}, X^{(2)}, \dots, X^{(m)})$ of size m

We only survey the case of binomial sample. Thus, \mathcal{D} having binomial distribution is called binomial sample and the network in figure III.3.1.2 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.1.1.5), we have

$$\begin{aligned}
 E(F^s(1-F)^t) &= \int_0^1 F^s(1-F)^t \beta(F) dF \\
 &= \int_0^1 F^s(1-F)^t \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} F^{a-1}(1-F)^{b-1} dF \\
 &\quad (\text{by applying definition of beta function specified by formula III.3.1.1}) \\
 &= \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \int_0^1 F^{a+s-1}(1-F)^{b+t-1} dF = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a+b+s+t)} \\
 &\quad (\text{due to formula III.3.1.4}) \\
 &= \frac{\Gamma(N)}{\Gamma(N+M)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)} \\
 &\quad (\text{due to } N = a + b \text{ and } M = s + t)
 \end{aligned}$$

In brief, we have formula III.3.1.7 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.3.1.7. Expectation of expression $F^s(1-F)^t$

The same proof for formula III.3.1.7 is found in (Neapolitan, 2003, p. 306) and formula III.3.1.7 is 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.3.1.8 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.3.1.8. Probability $P(\mathcal{D})$ of evidences \mathcal{D}

The similar proof for formula III.3.1.8 is found in (Neapolitan, 2003, p. 307) and formula III.3.1.8 is 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).

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:

$$\begin{aligned}\beta(F|\mathcal{D}) &= \frac{P(\mathcal{D}|F)\beta(F)}{P(\mathcal{D})} \\ &\quad (\text{According to Bayes' rule shown in formula III.1.1.1}) \\ &= \frac{F^s(1-F)^t\beta(F)}{E(F^s(1-F)^t)} \\ &\quad \left(\text{due to } P(\mathcal{D}|F) = \prod_{i=1}^m P(X^{(i)}|F) = F^s(1-F)^t \text{ and } P(\mathcal{D}) \right. \\ &\quad \left. = E(F^s(1-F)^t) \text{ specified in formula III.3.1.8} \right) \\ &= \frac{F^s(1-F)^t \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} F^{a-1}(1-F)^{b-1}}{\frac{\Gamma(N)}{\Gamma(N+M)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)}} \\ &\quad (\text{By applying formulas III.3.1.1 and III.3.1.7}) \\ &= \frac{\Gamma(N+M)}{\Gamma(a+s)\Gamma(b+t)} F^{a+s-1}(1-F)^{b+t-1} = \beta(F; a+s, b+t)\end{aligned}$$

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.3.1.9.

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

Formula III.3.1.9. Posterior beta density function

The similar proof formula III.3.1.9 is found in (Neapolitan, 2003, pp. 306-308) and formula III.3.1.9 is corollary 6.3 in (Neapolitan, 2003, p. 308). According to formula III.3.1.6, 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.3.1.10. Posterior probability of X

Formula III.3.1.10 is theorem 6.4 in (Neapolitan, 2003, p. 309). In general, you should merely remember the formulas III.3.1.1, III.3.1.6, III.3.1.9, III.3.1.10 and the way to recognize prior density function, prior probability of X , posterior density function, and posterior probability of X , respectively. Additionally, formula III.3.1.5 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 $\beta(F; a+s, b+t)$.

$b+t$). This is the process of parameter learning or the evolution of Bayesian model aforementioned in [beginning of this sub-section III.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.3.1.4](#) (Neapolitan, 2003, p. 329):

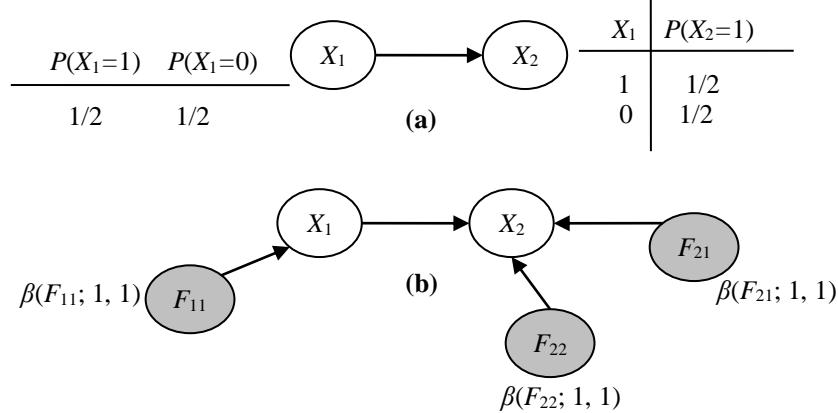


Figure III.3.1.4. BN (a) and complex augmented BN (b)

In figure [III.3.1.4](#), the BN (a) having no attached parameter 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 parameter parent nodes to X_i , obeying two principles below:

1. If X_i has no parent (not conditionally dependent on any others, X_i is a root), we add only one parameter variable denoted F_{i1} having the probability density function $\beta(F_{i1}; a_{i1}, b_{i1})$ so as to $P(X_i=1|F_{i1}) = F_{i1}$.
2. If X_i has a set of k_i parent nodes and each parent node is binary, we add a set of $c_i=2k_i$ parameter 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 parameter 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.3.1.4](#), node X_2 has one parent node X_1 , which causes that X_2 has two parent instances represented by two parameter 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.3.1.11](#) for connecting CPT of variable X_i with beta density function of parameter 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.3.1.11. Conditional probability (relative frequency) of X_i given a parent instance PA_{ij} , as value of parameter variable in multi-node BN

Formula III.3.1.11 is an extension of formula III.3.1.5 in multi-node BN and formula III.3.1.11 degenerates to formula III.3.1.5 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.3.1.4, 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.3.1.12 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.3.1.12. Beta density function $\beta(F_{ij})$ corresponding to an instance of a parent of node X_i

Note that formulas III.3.1.1 and III.3.1.12 have the same meaning for representing beta function except that formula III.3.1.12 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.3.1.13. Joint beta density function of variable X_i having c_i parent instances

Besides the local parameter independence expressed in formula III.3.1.13, 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.3.1.14 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.3.1.14. Global joint beta density function of n independent variable X_i (s)

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

All variables X_i and their parameter variables form the complex augmented BN representing the trust BN in figure III.3.1.4. 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.3.1.15. Probability of variable X_i with respect to its parent instance as expectation of beta variable

The formula III.3.1.15 is extension of formula III.3.1.6 when variable X_i has parent and both of these formulas express prior probability of variable X_i . Following is proof of formula III.3.1.15.

$$\begin{aligned} & P(X_i = 1 | PA_{ij}) \\ &= \int_0^1 \dots \int_0^1 P(X_i = 1 | PA_{ij}, F_{i1}, \dots, F_{ij}, \dots, F_{ic_i}) \beta(F_{i1}, \dots, F_{ij}, \dots, F_{ic_i}) \\ &\quad dF_{i1} \dots dF_{ij} \dots dF_{ic_i} \\ &= \int_0^1 \dots \int_0^1 P(X_i = 1 | PA_{ij}, F_{i1}, \dots, F_{ij}, \dots, F_{ic_i}) \beta(F_{i1}) \dots \beta(F_{ij}) \dots \beta(F_{ic_i}) \\ &\quad dF_{i1} \dots dF_{ij} \dots dF_{ic_i} \\ &\quad (\text{due to local parameter independence specified in formula III.3.1.13 when } F_{ij} \text{ (s) are mutually independent}) \\ &= \int_0^1 \dots \int_0^1 F_{ij} \beta(F_{i1}) \dots \beta(F_{ij}) \dots \beta(F_{ic_i}) dF_{i1} \dots dF_{ij} \dots dF_{ic_i} \\ &\quad (\text{due to } P(X_i = 1 | PA_{ij}, F_{i1}, \dots, F_{ij}, \dots, F_{ic_i}) = F_{ij} \text{ specified in formula III.3.1.11}) \\ &= \left(\int_0^1 \beta(F_{i1}) dF_{i1} \right) * \dots * \left(\int_0^1 F_{ij} \beta(F_{ij}) dF_{ij} \right) * \dots * \left(\int_0^1 \beta(F_{ic_i}) dF_{ic_i} \right) \\ &= 1 * \dots * \left(\int_0^1 F_{ij} \beta(F_{ij}) dF_{ij} \right) * \dots * 1 \\ &= \int_0^1 F_{ij} \beta(F_{ij}) dF_{ij} = E(F_{ij}) = \frac{a_{ij}}{N_{ij}} \end{aligned}$$

The formula III.3.1.15 is theorem 6.7 proved by the similar way in (Neapolitan, 2003, pp. 334-335) to which I referred. For illustrating formulas III.3.1.11 and III.3.1.15, 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.3.1.4, and so, the CPT (s) in figure III.3.1.4 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.3.1.4 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.3.1.4, $n = 2$. 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 only review the 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.3.1.4 is depicted by figure III.3.1.5 as below (Neapolitan, 2003, p. 337):

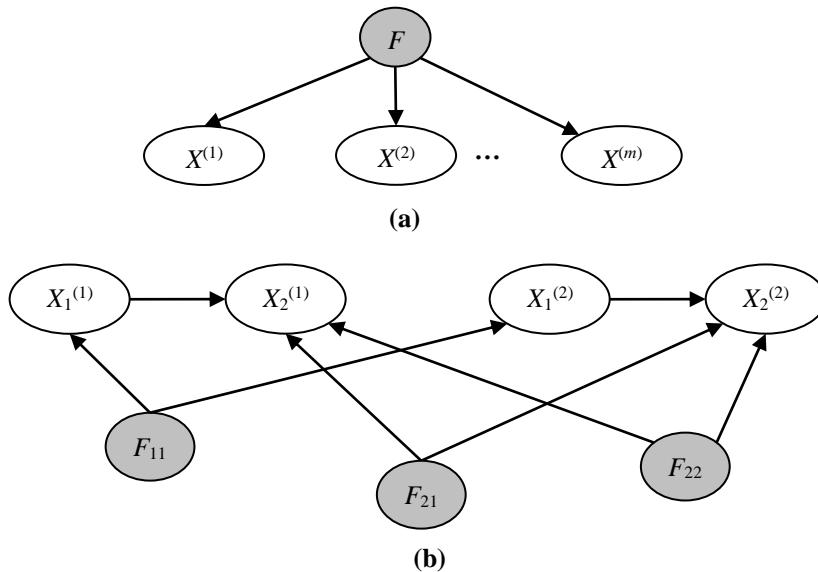


Figure III.3.1.5. Expanded binomial BN sample of size m

After m trials are performed, the augmented BN are updated and so, parameter variables' density functions and hypothesis variables' conditional probabilities are changed. We need to compute the posterior density function $\beta(F_{ij}/\mathcal{D})$ of each parameter 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 given all 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, formula III.3.1.16 is used 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.3.1.16. 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.3.1.16, 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:

$$\begin{aligned}
 P(\mathcal{D}|F_1, F_2, \dots, F_n) &= P(X^{(1)}, X^{(2)}, \dots, X^{(m)}|F_1, F_2, \dots, F_n) = \prod_{u=1}^m P(X^{(u)}|F_1, F_2, \dots, F_n) \\
 &\quad (\text{because evidence vectors } X^{(u)} \text{ (s) are mutually independent}) \\
 &= \prod_{u=1}^m \frac{P(X^{(u)}, F_1, F_2, \dots, F_n)}{P(F_1, F_2, \dots, F_n)} \\
 &\quad (\text{due to Bayes' rule specified in formula III.1.1.1}) \\
 &= \prod_{u=1}^m \frac{P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)}, F_1, F_2, \dots, F_n)}{P(F_1, F_2, \dots, F_n)} \\
 &= \prod_{u=1}^m \frac{P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)}|F_1, F_2, \dots, F_n)P(F_1, F_2, \dots, F_n)}{P(F_1, F_2, \dots, F_n)} \\
 &\quad (\text{applying multiplication rule specified by formula III.1.1.3 into the numerator}) \\
 &= \prod_{u=1}^m P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)}|F_1, F_2, \dots, F_n) \\
 &= \prod_{u=1}^m \prod_{i=1}^n P(X_i^{(u)}|PA_i, F_i) \\
 &\quad (\text{because } X_i^{(u)} \text{ (s) are mutually independent given } F_i \text{ (s) and each } X_i \text{ depends only on } PA_i \text{ and } F_i) \\
 &= \prod_{i=1}^n \prod_{u=1}^m P(X_i^{(u)}|PA_i, F_i) = \prod_{i=1}^n \prod_{j=1}^{c_i} (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}} \\
 &\quad \left(\text{due to formula III.3.1.16: } \prod_{u=1}^m P(X_i^{(u)}|PA_i, F_i) = \prod_{j=1}^{c_i} (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}} \right)
 \end{aligned}$$

In brief, we have formula III.3.1.17 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.3.1.17. Probability of evidence sample \mathcal{D} given vectors F_i

The formula III.3.1.17 is lemma 6.8 proved by similar way in (Neapolitan, 2003, pp. 338-339) to which I referred. It is necessary to calculate the whole probability $P(\mathcal{D})$ of evidence sample \mathcal{D} , we have:

$$\begin{aligned}
 P(\mathcal{D}) &= P(X^{(1)}, X^{(2)}, \dots, X^{(m)}) = \prod_{u=1}^m P(X^{(u)}) = \prod_{u=1}^m P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)}) \\
 &\quad (\text{due evidence vectors } X^{(u)} \text{ (s) are independent}) \\
 &= \prod_{u=1}^m \int_{F_1} \dots \int_{F_n} P(X_1^{(u)}, X_2^{(u)}, \dots, X_n^{(u)} | F_1, F_2, \dots, F_n) \beta(F_1, F_2, \dots, F_n) \\
 &\quad dF_1 dF_2 \dots dF_n \\
 &\quad (\text{due to total probability rule in continuous case, please see formula III.1.1.5}) \\
 &= \prod_{u=1}^m \int_{F_1} \dots \int_{F_n} \prod_{i=1}^n P(X_i^{(u)} | PA_i, F_i) \prod_{i=1}^n \beta(F_i) \\
 &\quad dF_1 dF_2 \dots dF_n \\
 &\quad (\text{Because } X_i^{(u)} \text{ (s) are mutually independent given } F_i \text{ (s) and each } X_i \text{ depends only on } PA_i \text{ and } F_i. \text{ Moreover, all } F_i \text{ (s) are mutually independent}) \\
 &= \prod_{u=1}^m \int_{F_1} \dots \int_{F_n} \left(\prod_{i=1}^n P(X_i^{(u)} | PA_i, F_i) \beta(F_i) \right) dF_1 dF_2 \dots dF_n \\
 &= \prod_{u=1}^m \prod_{i=1}^n \int_{F_i} P(X_i^{(u)} | PA_i, F_i) \beta(F_i) dF_i = \prod_{i=1}^n \prod_{u=1}^m \int_{F_i} P(X_i^{(u)} | PA_i, F_i) \beta(F_i) dF_i \\
 &= \prod_{i=1}^n \prod_{j=1}^{c_i} \int_0^1 (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}} \beta(F_{ij}) dF_{ij} \\
 &\quad \left(\text{due to binomial trials: } \prod_{u=1}^m \int_{F_i} P(X_i^{(u)} | PA_i, F_i) \beta(F_i) dF_i \right. \\
 &\quad \left. = \prod_{j=1}^{c_i} \int_0^1 (F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}} \beta(F_{ij}) dF_{ij} \right) \\
 &= \prod_{i=1}^n \prod_{j=1}^{c_i} E((F_{ij})^{s_{ij}} (1 - F_{ij})^{t_{ij}})
 \end{aligned}$$

In brief, we have formula III.3.1.18 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.3.1.18. Whole probability of evidence sample \mathcal{D}

Formula III.3.1.18 is 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.3.1.18” and so we have formula III.3.1.19 for calculating this expectation by extending formula III.3.1.7, as follows:

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

Formula III.3.1.19. 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 $P(X_i=1/P_{A_{ij}}, \mathcal{D})$ for each variable X_i in BN. In fact, we have:

$$\beta(F_{ij}|\mathcal{D}) = \frac{P(\mathcal{D}|F_{ij})\beta(F_{ij})}{P(\mathcal{D})}$$

(due to Bayes' rule specified in formula III.1.1.1)

$$= \frac{\left(\int_0^1 \dots \int_0^1 P(\mathcal{D}|F_1, F_2, \dots, F_n) \prod_{kl \neq ij} \beta(F_{kl}) dF_{kl} \right) \beta(F_{ij})}{P(\mathcal{D})}$$

(Due to total probability rule in continuous case, specified by formula III.1.1.5. Note that $F_i = \{F_{i1}, F_{i2}, \dots, F_{ic_i}\}$)

$$= \frac{\left(\int_0^1 \dots \int_0^1 \left(\prod_{u=1}^n \prod_{v=1}^{c_u} (F_{uv})^{s_{uv}} (1 - F_{uv})^{t_{uv}} \right) \left(\prod_{kl \neq ij} \beta(F_{kl}) dF_{kl} \right) \right) \beta(F_{ij})}{P(\mathcal{D})}$$

(due to formula III.3.1.17)

$$= \frac{\left(F_{ij} \right)^{s_{ij}} \left(1 - F_{ij} \right)^{t_{ij}} \left(\int_0^1 \dots \int_0^1 \prod_{kl \neq ij} (F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}} \beta(F_{kl}) dF_{kl} \right) \beta(F_{ij})}{P(\mathcal{D})}$$

$$= \frac{\left(F_{ij} \right)^{s_{ij}} \left(1 - F_{ij} \right)^{t_{ij}} \left(\prod_{kl \neq ij} \int_0^1 (F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}} \beta(F_{kl}) dF_{kl} \right) \beta(F_{ij})}{P(\mathcal{D})}$$

$$= \frac{\left(F_{ij} \right)^{s_{ij}} \left(1 - F_{ij} \right)^{t_{ij}} \left(\prod_{kl \neq ij} E((F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}}) \right) \beta(F_{ij})}{\prod_{k=1}^n \prod_{l=1}^{c_k} E((F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}})}$$

(applying formula III.3.1.18 into denominator)

$$= \frac{\left(F_{ij} \right)^{s_{ij}} \left(1 - F_{ij} \right)^{t_{ij}} \left(\prod_{kl \neq ij} E((F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}}) \right) \beta(F_{ij})}{\prod_{kl} E((F_{kl})^{s_{kl}} (1 - F_{kl})^{t_{kl}})}$$

$$= \frac{\left(F_{ij} \right)^{s_{ij}} \left(1 - F_{ij} \right)^{t_{ij}} \beta(F_{ij})}{E \left(\left(F_{ij} \right)^{s_{ij}} \left(1 - F_{ij} \right)^{t_{ij}} \right)}$$

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

(applying definition of beta density function specified by formula III.3.1.1 into numerator and applying formula III.3.1.19 into denominator, note that $N_{ij} = a_{ij} + b_{ij}$ and $M_{ij} = s_{ij} + t_{ij}$)

$$\begin{aligned}
 &= \frac{\Gamma(N_{ij} + M_{ij})}{\Gamma(a_{ij} + s_{ij})\Gamma(b_{ij} + t_{ij})} (F_{ij})^{a_{ij} + s_{ij} - 1} (1 - F_{ij})^{b_{ij} + t_{ij} - 1} \\
 &= \text{beta}(F_{ij}; a_{ij} + s_{ij}, b_{ij} + t_{ij})
 \end{aligned}$$

(due to definition of beta density function specified in formula III.3.1.1)

In brief, we have formula III.3.1.20 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.3.1.20. Posterior beta density function in multi-node BN

Note that formula III.3.1.20 is an extension of formula III.3.1.9 in case of multi-node BN. Formula III.3.1.20 is corollary 6.7 proved by similar way in (Neapolitan, 2003, p. 347) to which I referred. Applying formulas III.3.1.15 and III.3.1.20, 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.3.1.21. 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.3.1.21 is an extension of formula III.3.1.10 in case of multi-node BN. In general, in case of binomial distribution, if we have the real/trust BN embedded in the expanded augmented network like figure III.3.1.4 and each parameter 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.3.1.10) 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.3.1.4. Evidence sample (evidence matrix) \mathcal{D} is shown in table III.3.1.1 (Neapolitan, 2003, p. 358).

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

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

Table III.3.1.1. Evidence sample corresponding to 5 trials (sample of size 5)

In order to interpret evidence sample \mathcal{D} in table III.3.1.1, for instance, the first evidence (vector) $X^{(1)} = \begin{pmatrix} X_1^{(1)} = 1 \\ X_2^{(1)} = 1 \end{pmatrix}$ 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.3.1.2 shows counters s_{ij} , t_{ij} (s) and posterior density functions calculated based on these counters; please see formula III.3.1.20 for more details about updating posterior density functions. For instance, the number of rows (evidences) in table III.3.1.1 such that $X_2=1$ given $X_1=1$ is 3, which causes $s_{21} = 3$ in table III.3.1.2.

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

Table III.3.1.2. 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.3.1.21. Table III.3.1.3 expresses such posterior conditional probabilities as evolutional CPT (s) of X_1 and X_2 .

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

Table III.3.1.3. 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.3.1.4 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.3.1.6 as follows:

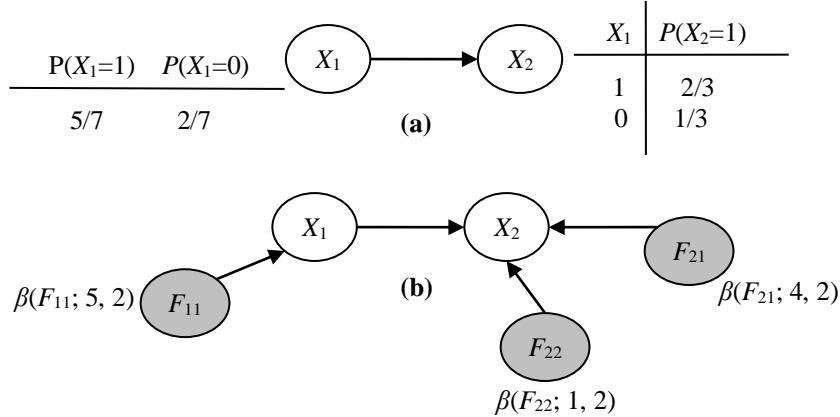


Figure III.3.1.6. Updated version of BN (a) and augmented BN (b)

It is easy to perform learning parameters or the evolution of Bayesian by counting numbers s_{ij} and t_{ij} among sample according to expectation of beta density function as in formulas III.3.1.10 and III.3.1.21 but a problem occurs when data in sample is missing. This problem is solved by expectation maximization (EM) algorithm mentioned in next sub-section III.3.2.

III.3.2. Learning parameters in case of missing data

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.3.1.1, 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.3.1.4 but the evidences $X^{(2)}$ and $X^{(5)}$ have not data yet. Table III.3.2.1 shows such missing data (Neapolitan, 2003, p. 359).

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

Table III.3.2.1. Evidence sample with missing data

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.3.2.2 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.3.2.2. New split evidences for missing data

The number $\#n_{11}$ ($\#n_{10}$) of occurrences of $v_1=1$ ($v_1=0$) is estimated by the probability of $X_2 = 1$ given $X_1 = 1$ ($X_2 = 0$ given $X_1 = 1$) with assumption that $a_{21} = 1$ and $b_{21} = 1$ as in figure III.3.1.4.

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

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

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.3.1.4.

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

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

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.3.2.3.

	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.3.2.3. Complete evidence sample in E-step of EM algorithm

In general, the essence of this task – estimating missing values by *expectations* of F_{21} and F_{22} based on previous parameters a_{21} , b_{21} , a_{22} , and b_{22} of beta density functions is E-step in EM algorithm. Of course, in E-step, when missing values are estimated, it is easy to determine counters s_{11} , t_{11} , s_{21} , t_{21} , s_{22} , and t_{22} . Recall that counters s_{11} and t_{11} are numbers of evidences such that $X_1 = 1$ and $X_1 = 0$, respectively. Counters s_{21} and t_{21} (s_{22} and t_{22}) are numbers of evidences such that $X_2 = 1$ and $X_2 = 0$ given $X_1 = 1$ ($X_2 = 1$ and $X_2 = 0$ given $X_1 = 0$), respectively. In fact, these counters are ultimate results of E-step. From complete sample \mathcal{D} in table III.3.2.3, we have table III.3.2.4 showing such ultimate results of E-step:

$$s_{11} = 1 + \frac{1}{2} + \frac{1}{2} + 1 + 1 = 4 \quad t_{11} = \frac{1}{2} + \frac{1}{2} = 1$$

$$\begin{array}{ll}
 s_{21} = 1 + \frac{1}{2} + 1 = \frac{5}{2} & t_{21} = \frac{1}{2} + 1 = \frac{3}{2} \\
 s_{22} = \frac{1}{2} & t_{22} = \frac{1}{2}
 \end{array}$$

Table III.3.2.4. Counters s_{11} , t_{11} , s_{21} , t_{21} , s_{22} , and t_{22} from estimated values (of missing values)

The next step of EM algorithm, M-step is responsible for updating posterior density functions $\beta(F_{11}|\mathcal{D})$, $\beta(F_{21}|\mathcal{D})$, 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.3.2.4). Table III.3.2.5 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 $P(X_2=1|X_1=0, \mathcal{D})$.

$$\begin{aligned}
 \beta(F_{11}|\mathcal{D}) &= \beta(F_{11}; a_{11} + s_{11}, b_{11} + t_{11}) = \beta(F_{11}; 1 + 4, 1 + 1) = \beta(F_{11}; 5, 2) \\
 \beta(F_{21}|\mathcal{D}) &= \beta(F_{21}; a_{21} + s_{21}, b_{21} + t_{21}) = \beta\left(F_{21}; 1 + \frac{5}{2}, 1 + \frac{3}{2}\right) = \beta\left(F_{21}; \frac{7}{2}, \frac{5}{2}\right) \\
 \beta(F_{22}|\mathcal{D}) &= \beta(F_{22}; a_{22} + s_{22}, b_{22} + t_{22}) = \beta\left(F_{22}; 1 + \frac{1}{2}, 1 + \frac{1}{2}\right) = \beta\left(F_{22}; \frac{3}{2}, \frac{3}{2}\right) \\
 P(X_1 = 1|\mathcal{D}) &= E(F_{11}|\mathcal{D}) = \frac{5}{5+2} = \frac{5}{7} \\
 P(X_2 = 1|X_1 = 1, \mathcal{D}) &= E(F_{21}|\mathcal{D}) = \frac{7/2}{7/2 + 5/2} = \frac{7}{12} \\
 P(X_2 = 1|X_1 = 0, \mathcal{D}) &= E(F_{22}|\mathcal{D}) = \frac{3/2}{3/2 + 3/2} = \frac{1}{2}
 \end{aligned}$$

Table III.3.2.5. Posterior density functions and posterior probabilities are updated in M-step of EM algorithm

Note that origin parameters such as $a_{11}=1$, $b_{11}=1$, $a_{21}=1$, $b_{21}=1$, $a_{22}=1$, and $b_{22}=1$ (see figure III.3.1.4) are kept intact in the task of updating posterior density functions $\beta(F_{11}|\mathcal{D})$, $\beta(F_{21}|\mathcal{D})$, and $\beta(F_{22}|\mathcal{D})$. For example, $\beta(F_{11}|\mathcal{D}) = \beta(F_{11}; a_{11}+s_{11}, b_{11}+t_{11}) = \beta(F_{11}; 1+4, 1+1) = \beta(F_{11}; 5, 2)$. After the updating task, these parameters are changed into new values; concretely, $a_{11}=5$, $b_{11}=2$, $a_{21}=7/2$, $b_{21}=5/2$, $a_{22}=3/2$, and $b_{22}=3/2$. These parameters updated with new values, which are called posterior parameters, are in turn used for the new iteration of EM algorithm.

The process of such two steps (E-step and M-step) repeated more and more brings out the EM algorithm. In general, EM algorithm is the iterative algorithm having many iterations and each iteration has two steps: E-step and M-step. Given the 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). Table III.3.2.4 shows such current counters which are ultimate results of E-step.
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. Table

III.3.2.5 shows results of M-step. Terminating algorithm if stop condition becomes true, otherwise, reiterating step 1. The stop condition may be “posterior density functions and posterior probabilities are not changed significantly”, “the number of iterations approaches k times” or “there is no missing value”.

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 $P(X_i=1|PA_{ij}, \mathcal{D}) = \lim_{k \rightarrow +\infty} E(F_{ij} | \mathcal{D})^{(k)}$ 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).

Go back to the example of missing data, the results of EM algorithm at the first iteration are summarized from table **III.3.2.5**, as follows:

$$\begin{aligned} a_{11} &= 5, b_{11} = 2, a_{21} = \frac{7}{2}, b_{21} = \frac{5}{2}, a_{22} = \frac{3}{2}, b_{22} = \frac{3}{2} \\ P(X_1 = 1) &= \frac{5}{7} \approx 0.71, P(X_2 = 1|X_1 = 1) = \frac{7}{12} \approx 0.58, P(X_2 = 1|X_1 = 0) = \frac{1}{2} \\ &= 0.5 \end{aligned}$$

When compared with the original probabilities

$$P(X_1 = 1) = \frac{1}{2} = 0.5, P(X_2 = 1|X_1 = 1) = \frac{1}{2} = 0.5, P(X_2 = 1|X_1 = 0) = \frac{1}{2} = 0.5$$

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

At the second iteration, the E-step starts calculating the number $\#n_{11}$ ($\#n_{10}$) of occurrences of $v_1=1$ ($v_1=0$) and the number $\#n_{21}$ ($\#n_{20}$) of occurrences of $v_2=1$ ($v_2=0$) again:

$$\begin{aligned} \#n_{11} &= P(X_2 = 1|X_1 = 1) = E(F_{21}) = \frac{a_{21}}{a_{21} + b_{21}} = \frac{7/2}{7/2 + 5/2} = \frac{7}{12} \\ \#n_{10} &= P(X_2 = 0|X_1 = 1) = 1 - P(X_2 = 1|X_1 = 1) = 1 - \frac{7}{12} = \frac{5}{12} \\ \#n_{21} &= P(X_2 = 1|X_1 = 0) = E(F_{22}) = \frac{a_{22}}{a_{22} + b_{22}} = \frac{3/2}{3/2 + 3/2} = \frac{1}{2} \\ \#n_{20} &= P(X_2 = 0|X_1 = 0) = 1 - P(X_2 = 1|X_1 = 0) = 1 - \frac{1}{2} = \frac{1}{2} \end{aligned}$$

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 follows:

	X_1	X_2	#Occurrences
$X^{(1)}$	$X_1^{(1)} = 1$	$X_2^{(1)} = 1$	1
$X^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = 1$	7/12
$X^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = 0$	5/12
$X^{(3)}$	$X_1^{(3)} = 1$	$X_2^{(3)} = 1$	1
$X^{(4)}$	$X_1^{(4)} = 1$	$X_2^{(4)} = 0$	1
$X^{(5)}$	$X_1^{(5)} = 0$	$X_2^{(5)} = 1$	1/2
$X^{(5)}$	$X_1^{(5)} = 0$	$X_2^{(5)} = 0$	1/2

Recall that counters s_{11} and t_{11} are numbers of evidences such that $X_1 = 1$ and $X_1 = 0$, respectively. Counters s_{21} and t_{21} (s_{22} and t_{22}) are numbers of evidences such that $X_2 = 1$ and $X_2 = 0$ given $X_1 = 1$ ($X_2 = 1$ and $X_2 = 0$ given $X_1 = 0$), respectively. These counters which are ultimate results of E-step are calculated as follows:

$$\boxed{\begin{aligned} s_{11} &= 1 + \frac{7}{12} + \frac{5}{12} + 1 + 1 = 4 & t_{11} &= \frac{1}{2} + \frac{1}{2} = 1 \\ s_{21} &= 1 + \frac{7}{12} + 1 = \frac{31}{12} & t_{21} &= \frac{5}{12} + 1 = \frac{17}{12} \\ s_{22} &= \frac{1}{2} & t_{22} &= \frac{1}{2} \end{aligned}}$$

Posterior density functions $\beta(F_{11}|\mathcal{D})$, $\beta(F_{21}|\mathcal{D})$, and $\beta(F_{22}|\mathcal{D})$, posterior probabilities $P(X_1=1|\mathcal{D})$, $P(X_2=1|X_1=1,\mathcal{D})$, and $P(X_2=1|X_1=0,\mathcal{D})$ are updated at M-step as follows:

$$\beta(F_{11}|\mathcal{D}) = \beta(F_{11}; a_{11} + s_{11}, b_{11} + t_{11}) = \beta(F_{11}; 5 + 4, 2 + 1) = \beta(F_{11}; 9, 3)$$

$$\begin{aligned} \beta(F_{21}|\mathcal{D}) &= \beta(F_{21}; a_{21} + s_{21}, b_{21} + t_{21}) = \beta\left(F_{21}; \frac{7}{2} + \frac{31}{12}, \frac{5}{2} + \frac{17}{12}\right) \\ &= \beta\left(F_{21}; \frac{73}{12}, \frac{47}{12}\right) \end{aligned}$$

$$\beta(F_{22}|\mathcal{D}) = \beta(F_{22}; a_{22} + s_{22}, b_{22} + t_{22}) = \beta\left(F_{22}; \frac{3}{2} + \frac{1}{2}, \frac{3}{2} + \frac{1}{2}\right) = \beta(F_{22}; 2, 2)$$

$$P(X_1 = 1|\mathcal{D}) = E(F_{11}|\mathcal{D}) = \frac{9}{9+3} = \frac{3}{4} = 0.75$$

$$P(X_2 = 1|X_1 = 1, \mathcal{D}) = E(F_{21}|\mathcal{D}) = \frac{73/12}{73/12 + 47/12} = \frac{73}{120} \approx 0.61$$

$$P(X_2 = 1|X_1 = 0, \mathcal{D}) = E(F_{22}|\mathcal{D}) = \frac{2}{2+2} = \frac{1}{2} = 0.5$$

When compared with the previous probabilities

$$\begin{aligned} P(X_1 = 1) &= \frac{5}{7} \approx 0.71, P(X_2 = 1|X_1 = 1) = \frac{7}{12} \approx 0.58, P(X_2 = 1|X_1 = 0) = \frac{1}{2} \\ &= 0.5 \end{aligned}$$

There is no significant change in these probabilities if the maximum deviation is pre-defined 0.05. It is easy for us to verify this assertion, concretely, $|0.75 - 0.71| = 0.04 < 0.05$, $|0.61 - 0.58| = 0.03 < 0.05$, and $|0.5 - 0.5| = 0 < 0.05$. So the EM algorithm is stopped with note that we can execute more iterations for EM algorithm in order to receive more precise results that posterior probabilities are stable ($\lim_{k \rightarrow +\infty} E(F_{ij}|\mathcal{D})^{(k)}$ approaches certain limit). Consequently, the Bayesian overlay model in figure III.3.1.4 is converted into the evolutional version specified in figure III.3.2.1.

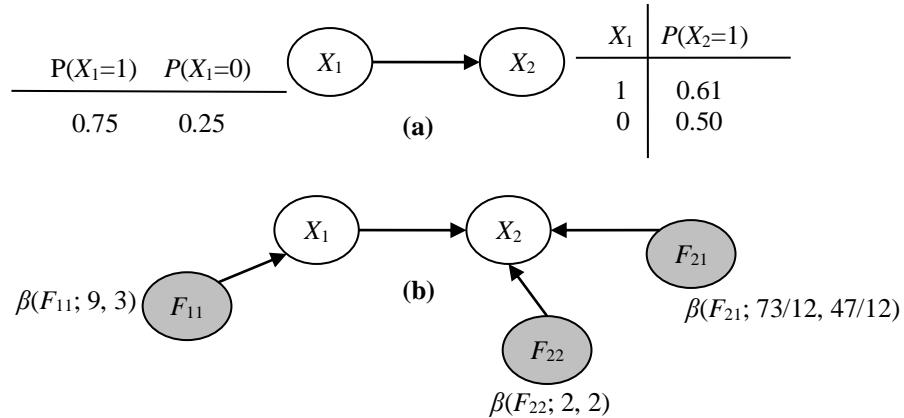


Figure III.3.2.1. Updated version of BN (a) and augmented BN (b) in case of missing data

In general, parameter learning or evolution of Bayesian overlay model is described thoroughly in this sub-section III.3.2 and previous sub-section III.3.1. The next sub-section III.3.3 is a full example illustrating main aspects of parameter learning.

III.3.3. An example of learning parameters

Suppose a short Java course (see sub-section III.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 an evidence variable: “Exercise: set up class Human”. The evidence “Exercise: set up class Human” proves whether or not she/he understands concept “Class & Object”. The number (in range 0...1) that measures the relative importance of each aggregation or evidence is defined as the weight of arc from parent node to child node. All weights concerning the child variable are normalized and used to build up its CPT implied by beta density function. It is logical to initialize weights by uniform distribution. Our work is to define prior density functions and enhance them based on evidences, namely, specifying appropriate posterior density functions. As known, this process is parameter evolution (evolution of Bayesian overlay model) or parameter learning described totally in previous sub-sections III.3.2 and III.3.1. Figure III.3.3.1 depicts network structure as weighted graph (a) and augmented BN (b) of Java course.

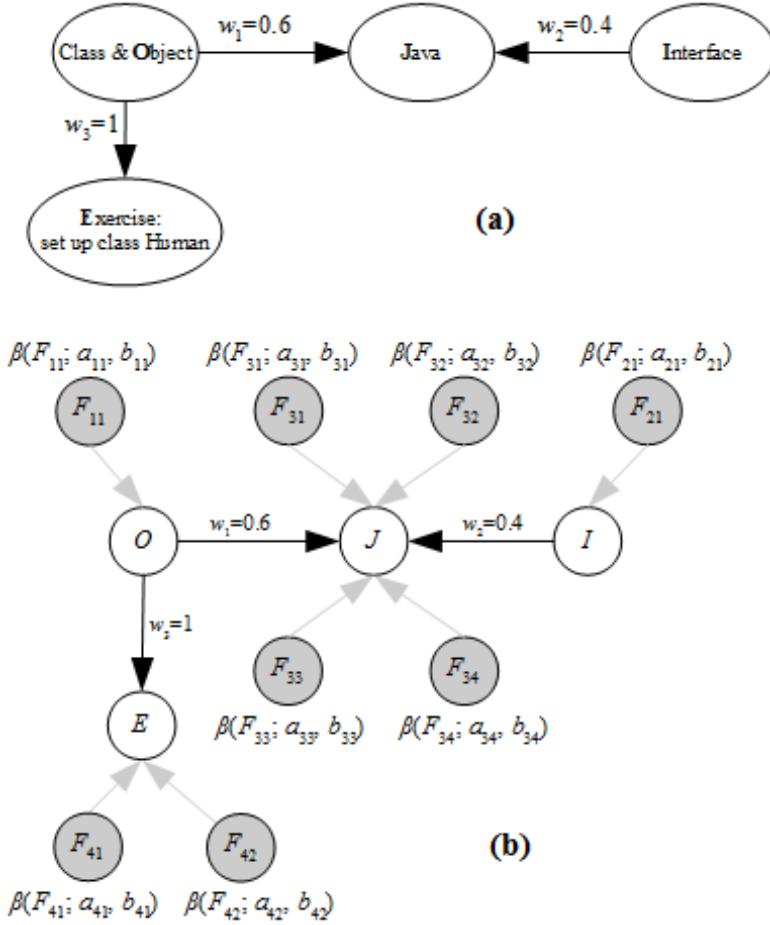


Figure III.3.3.1. BN structure as weighted graph (a) and augmented BN (b) of Java course

Nodes J , O , I denote knowledge variables (hypothesis variables) “Java”, “Class & Object”, “Interface”, respectively. Node E denotes evidence variable “Exercise: set up class Human”. In this example, node J has two parents: O and I which in turn are corresponding to two weights of aggregation relationship: $w_1=0.6$, $w_2=0.4$. The default weight of diagnostic relationship between E and O is $w_3=1$. Prior conditional probabilities (CPT (s)) of variables J , O , I , and E are specified based on these weights according to theorem of SIGMA-gate inference specified in formula III.1.3.4 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.1.3). For instance, the prior conditional probability of $J=1$ given $O=1$, and $I=1$ is:

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

Prior conditional probability of evidence E is specified by formula III.1.2.2. Shortly, it is easy to determine all CPT (s) of variables J , O , I , and E as shown in table III.3.3.1.

Real Variable	Parameter 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}	$\beta(F_{21}; a_{21}, b_{21})$	$P(I=1) = a_{21}/(a_{21}+b_{21}) = 1 * 1 = 1$ $P(I=0) = 1 - P(I=1) = 0$
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}	$\beta(F_{42}; a_{42}, b_{42})$	$P(E=1 O=0) = 0 * 1 = 0 = a_{42}/(a_{42}+b_{42})$ $P(E=0 O=0) = 1 - P(E=1 O=0) = 1$

Table III.3.3.1. 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.3.3.1. 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.3.3.1 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.3.3.1. 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.3.3.1 is equation 6.13 in (Neapolitan, 2003, p. 351). For example, reviewing BN in figure III.3.1.4, given $\beta(F_{11}; 2, 2), \beta(F_{21}; 1, 1), \beta(F_{22}; 1, 1)$, we have:

$$4 = a_{11} + b_{11} = 1 * 4 = 4 \quad (P(PA_{11})=1 \text{ because } X_1 \text{ has no parent})$$

$$2 = a_{21} + b_{21} = P(X_1=1) * 4 = \frac{1}{2} * 4 = 2$$

$$2 = a_{22} + b_{22} = P(X_1=0) * 4 = \frac{1}{2} * 4 = 2$$

So, this network has equivalent sample size 4. The theorem of “equivalent sample size” is stated that if there exists the number N so that all parameters a_{ij} and b_{ij} (s) are satisfied formula III.3.3.2 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.3.3.2. Theorem of equivalent sample size N

It is easy to prove this theorem, we have:

$$\begin{aligned} \text{For all } i \text{ and } j, N_{ij} &= a_{ij} + b_{ij} \\ &= P(X_i = 1 | PA_{ij}) * P(PA_{ij}) * N + P(X_i = 0 | PA_{ij}) * P(PA_{ij}) * N \\ &= P(PA_{ij}) * N * (P(X_i = 1 | PA_{ij}) + P(X_i = 0 | PA_{ij})) \\ &= P(PA_{ij}) * N * (P(X_i = 1 | PA_{ij}) + 1 - P(X_i = 1 | PA_{ij})) \\ &= P(PA_{ij}) * N \end{aligned}$$

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.3.3.1, we suppose that the augmented BN has equivalent sample size $N = 100$. Applying the theorem of equivalent sample size specified in formula III.3.3.2, all prior parameters a_{ij} and b_{ij} (s) are calculated as in table III.3.3.2 as follows:

Density Functions	Parameters
$\beta(F_{11}; a_{11}, b_{11})$	$a_{11} = P(O=1)*1*10 = 0.5*100 = \mathbf{50}$ $b_{11} = P(O=0)*1*10 = 0.5*100 = \mathbf{50}$
$\beta(F_{21}; a_{21}, b_{21})$	$a_{21} = P(I=1)*1*10 = 1*100 = \mathbf{100}$ $b_{21} = P(I=0)*1*10 = 0*100 = \mathbf{0}$
$\beta(F_{31}; a_{31}, b_{31})$	$a_{31} = P(J=1 O=1, I=1)*P(O=1, I=1)*100$ = $P(J=1 O=1, I=1)*P(O=1)*P(I=1)*100$ = $1*0.5*0.5*100 = \mathbf{25}$ $b_{31} = P(J=0 O=1, I=1)*P(O=1, I=1)*100$ = $P(J=0 O=1, I=1)*P(O=1)*P(I=1)*100$ = $0*0.5*0.5*100 = \mathbf{0}$
$\beta(F_{32}; a_{32}, b_{32})$	$a_{32} = P(J=1 O=1, I=0)*P(O=1, I=0)*100$ = $P(J=1 O=1, I=0)*P(O=1)*P(I=0)*100$ = $0.6*0.5*0.5*100 = \mathbf{15}$ $b_{32} = P(J=0 O=1, I=0)*P(O=1, I=0)*100$ = $P(J=0 O=1, I=0)*P(O=1)*P(I=0)*100$ = $0.4*0.5*0.5*100 = \mathbf{10}$
$\beta(F_{33}; a_{33}, b_{33})$	$a_{33} = P(J=1 O=0, I=1)*P(O=0, I=1)*100$ = $P(J=1 O=0, I=1)*P(O=0)*P(I=1)*100$ = $0.4*0.5*0.5*100 = \mathbf{10}$ $b_{33} = P(J=0 O=0, I=1)*P(O=0, I=1)*100$ = $P(J=0 O=0, I=1)*P(O=0)*P(I=1)*100$ = $0.6*0.5*0.5*100 = \mathbf{15}$
$\beta(F_{34}; a_{34}, b_{34})$	$a_{34} = P(J=1 O=0, I=0)*P(O=0, I=0)*100$ = $P(J=1 O=0, I=0)*P(O=0)*P(I=0)*100$ = $0*0.5*0.5*100 = \mathbf{0}$ $b_{34} = P(J=0 O=0, I=0)*P(O=0, I=0)*100$ = $P(J=0 O=0, I=0)*P(O=0)*P(I=0)*100$ = $1*0.5*0.5*100 = \mathbf{25}$
$\beta(F_{41}; a_{41}, b_{41})$	$a_{41} = P(E=1 O=1)*P(O=1)*100$ = $1*0.5*100 = \mathbf{50}$ $b_{41} = P(E=0 O=1)*P(O=1)*100$ = $0*0.5*100 = \mathbf{0}$
$\beta(F_{42}; a_{42}, b_{42})$	$a_{42} = P(E=1 O=0)*P(O=0)*100$ = $0*0.5*100 = \mathbf{0}$ $b_{42} = P(E=0 O=0)*P(O=0)*100$ = $1*0.5*100 = \mathbf{50}$

Table III.3.3.2. All parameters of prior density functions

Note that prior conditional probabilities $P(X_i/PA_{ij})$ are shown in table III.3.3.1. The augmented BN with parameters in full is shown in figure III.3.3.2.

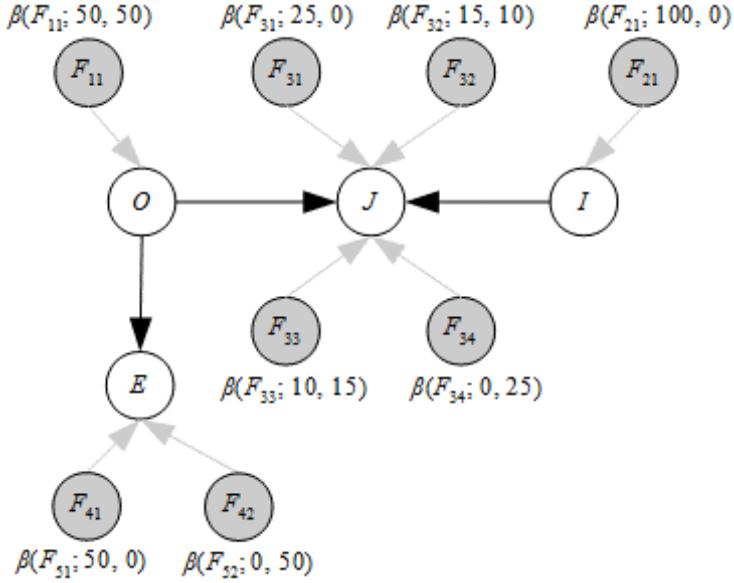


Figure III.3.3.2. Augmented BN with initial parameters in full

When prior CPT (s) (see table III.3.3.1) and prior beta density functions (see table III.3.3.2) are specified, the example Java course in this sub-section III.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.3.1 and III.3.2. Suppose there is an observation: User does well the exercise “*Set up class Human*”. Therefore, we have an evidence $D = (E=1)$. Because it lacks information about other concepts such as O, J, I , this is case of missing data, aforementioned in previous sub-section III.3.2. Table III.3.3.3 shows incomplete sample \mathcal{D} with the evidence $D = (E=1)$, in which question mark (?) denotes missing values of variables O, I , and J . EM algorithm mentioned in previous sub-section III.3.2 will be used to estimate these missing values.

	O	I	J	E
D	?	?	?	1

Table III.3.3.3. Incomplete sample with evidence $D = (E=1)$

Note that missing values (?) are binary and so the evidence D is split into many D' (s) according two possible values 0 and 1 of missing values (?). Table III.3.3.4 shows new split evidences for missing values.

	O	I	J	E	#Occurrences
D'	1	1	1	1	# n_1
D'	1	1	0	1	# n_2
D'	1	0	1	1	# n_3
D'	1	0	0	1	# n_4
D'	0	1	1	1	# n_5
D'	0	1	0	1	# n_6
D'	0	0	1	1	# n_7
D'	0	0	0	1	# n_8

Table III.3.3.4. New split evidences for missing values of O, I , and J

Where $\#n_i$ are numbers of possible combinations (occurrences) of binary variables O , I , and J ; please see tables III.3.2.2 and III.3.2.3 for knowing more about $\#n_i$.

It is required to estimate $\#n_i$, which is the most important task in E-step of EM algorithm. For instance, the $\#n_1$ is estimated by the probability of $O=1$, $I=1$, and $J=1$ given $E=1$. We have:

$$\begin{aligned}
 & P(E = 1) \\
 & = P(O = 1, I = 1, J = 1, E = 1) \\
 & + P(O = 1, I = 1, J = 0, E = 1) \\
 & + P(O = 1, I = 0, J = 1, E = 1) \\
 & + P(O = 1, I = 0, J = 0, E = 1) \\
 & + P(O = 0, I = 1, J = 1, E = 1) \\
 & + P(O = 0, I = 1, J = 0, E = 1) \\
 & + P(O = 0, I = 0, J = 1, E = 1) \\
 & + P(O = 0, I = 0, J = 0, E = 1) \\
 & = P(O = 1) * P(I = 1) * P(J = 1 | O = 1, I = 1) * P(E = 1 | O = 1) \\
 & + P(O = 1) * P(I = 1) * P(J = 0 | O = 1, I = 1) * P(E = 1 | O = 1) \\
 & + P(O = 1) * P(I = 0) * P(J = 1 | O = 1, I = 0) * P(E = 1 | O = 1) \\
 & + P(O = 1) * P(I = 0) * P(J = 0 | O = 1, I = 0) * P(E = 1 | O = 1) \\
 & + P(O = 0) * P(I = 1) * P(J = 1 | O = 0, I = 1) * P(E = 1 | O = 0) \\
 & + P(O = 0) * P(I = 1) * P(J = 0 | O = 0, I = 1) * P(E = 1 | O = 0) \\
 & + P(O = 0) * P(I = 0) * P(J = 1 | O = 0, I = 0) * P(E = 1 | O = 0) \\
 & + P(O = 0) * P(I = 0) * P(J = 0 | O = 0, I = 0) * P(E = 1 | O = 0) \\
 & \quad (\text{by applying formula III.1.1.8 into joint probability distribution } P(O, I, J, E, Q)) \\
 & = 0.5 * 1 * 1 * 1 * 1 \\
 & + 0.5 * 1 * 0 * 1 * 1 \\
 & + 0.5 * 0 * 0.6 * 1 * 1 \\
 & + 0.5 * 0 * 0.4 * 1 * 1 \\
 & + 0.5 * 1 * 0.4 * 0 * 0 \\
 & + 0.5 * 1 * 0.6 * 0 * 0 \\
 & + 0.5 * 0 * 0 * 0 * 0 \\
 & + 0.5 * 0 * 1 * 0 * 0 \\
 & \quad (\text{prior probabilities } P(\cdot) \text{ are specified in table III.3.3.1}) \\
 & = 0.5 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0.5
 \end{aligned}$$

$$\begin{aligned}
 \#n_1 & = P(O = 1, I = 1, J = 1 | E = 1) = \frac{P(O = 1, I = 1, J = 1, E = 1)}{P(E = 1)} \\
 & \quad (\text{due to Bayes' rule specified in formula III.1.1.1}) \\
 & = (P(O = 1) * P(I = 1) * P(J = 1 | O = 1, I = 1) * P(E = 1 | O = 1)) / (P(E = 1)) \\
 & \quad (\text{by applying formula III.1.1.8 into joint probability distribution } P(O, I, J, E, Q)) \\
 & = \frac{0.5}{0.5} = 1
 \end{aligned}$$

Similarly, we have:

$$\begin{aligned}
 \#n_2 & = \frac{P(O = 1, I = 1, J = 0, E = 1)}{P(E = 1)} = \frac{0}{0.5} = 0 \\
 \#n_3 & = \frac{P(O = 1, I = 0, J = 1, E = 1)}{P(E = 1)} = \frac{0}{0.5} = 0 \\
 \#n_4 & = \frac{P(O = 1, I = 0, J = 0, E = 1)}{P(E = 1)} = \frac{0}{0.5} = 0
 \end{aligned}$$

$$\#n_5 = \frac{P(O=0, I=1, J=1, E=1)}{P(E=1)} = \frac{0}{0.5} = 0$$

$$\#n_6 = \frac{P(O=0, I=1, J=0, E=1)}{P(E=1)} = \frac{0}{0.5} = 0$$

$$\#n_7 = \frac{P(O=0, I=0, J=1, E=1)}{P(E=1)} = \frac{0}{0.5} = 0$$

$$\#n_8 = \frac{P(O=0, I=0, J=0, E=1)}{P(E=1)} = \frac{0}{0.5} = 0$$

When occurrence numbers $\#n_i$ (s) are determined, missing data is filled fully and evidence sample \mathcal{D} shown in table III.3.3.4 is completed as follows:

	O	I	J	E	#Occurrences
D'	1	1	1	1	$\#n_1=1$
D'	1	1	0	1	$\#n_2=0$
D'	1	0	1	1	$\#n_3=0$
D'	1	0	0	1	$\#n_4=0$
D'	0	1	1	1	$\#n_5=0$
D'	0	1	0	1	$\#n_6=0$
D'	0	0	1	1	$\#n_7=0$
D'	0	0	0	1	$\#n_8=0$

Table III.3.3.5. Complete sample with evidence $D = (E=1)$

When missing values are estimated as in table III.3.3.5, 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}) is the number of evidences such that $O=1$ ($O=0$), which corresponds to variable F_{11} .
- The counter s_{21} (t_{11}) is the number of evidences such that $I=1$ ($I=0$), which corresponds to variable F_{21} .
- The counter s_{31} (t_{31}) is the number of evidences such that $J=1$ ($J=0$) given $O=1$ and $I=1$, which corresponds to variable F_{31} .
- The counter s_{32} (t_{32}) is the number of evidences such that $J=1$ ($J=0$) given $O=1$ and $I=0$, which corresponds to variable F_{32} .
- The counter s_{33} (t_{33}) is the number of evidences such that $J=1$ ($J=0$) given $O=0$ and $I=1$, which corresponds to variable F_{33} .
- The counter s_{34} (t_{34}) is the number of evidences such that $J=1$ ($J=0$) given $O=0$ and $I=0$, which corresponds to variable F_{34} .
- The counter s_{41} (t_{41}) is the number of evidences such that $E=1$ ($E=0$) given $O=1$, which corresponds to variable F_{41} .
- The counter s_{42} (t_{42}) is the number of evidences such that $E=1$ ($E=0$) given $O=0$, which corresponds to variable F_{42} .

From complete sample \mathcal{D} in table III.3.3.5, we have:

$s_{11}=\#n_1=1$	$t_{11}=0$
$s_{21}=\#n_1=1$	$t_{21}=0$
$s_{31}=\#n_1=1$	$t_{31}=0$
$s_{32}=0$	$t_{32}=0$
$s_{33}=0$	$t_{33}=0$
$s_{34}=0$	$t_{34}=0$
$s_{41}=\#n_1=1$	$t_{41}=0$

s ₄₂ =0	t ₄₂ =0
--------------------	--------------------

Table III.3.3.6. 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 $P(O=1|\mathcal{D})$, $P(I=1|\mathcal{D})$, $P(J=1|O=1, I=1, \mathcal{D})$, $P(J=1|O=1, I=0, \mathcal{D})$, $P(J=1|O=0, I=1, \mathcal{D})$, $P(J=1|O=0, I=0, \mathcal{D})$, $P(E=1|O=1, \mathcal{D})$, $P(E=1|O=0, \mathcal{D})$, based on current counters s_{ij} and t_{ij} from complete evidence sample \mathcal{D} (table III.3.3.6). Table III.3.3.7 shows results of M-step which are these posterior density functions and posterior probabilities.

$\beta(F_{11} \mathcal{D}) = \beta(F_{11}; a_{11}+s_{11}, b_{11}+t_{11}) = \beta(F_{11}; 50+1, 50+0) = \beta(F_{11}; 51, 50)$
$\beta(F_{21} \mathcal{D}) = \beta(F_{21}; a_{21}+s_{21}, b_{21}+t_{21}) = \beta(F_{21}; 100+1, 0+0) = \beta(F_{21}; 101, 0)$
$\beta(F_{31} \mathcal{D}) = \beta(F_{31}; a_{31}+s_{31}, b_{31}+t_{31}) = \beta(F_{31}; 25+1, 0+0) = \beta(F_{31}; 26, 0)$
$\beta(F_{32} \mathcal{D}) = \beta(F_{32}; a_{32}+s_{32}, b_{32}+t_{32}) = \beta(F_{32}; 15+0, 10+0) = \beta(F_{32}; 15, 10)$
$\beta(F_{33} \mathcal{D}) = \beta(F_{33}; a_{33}+s_{33}, b_{33}+t_{33}) = \beta(F_{33}; 10+0, 15+0) = \beta(F_{33}; 10, 15)$
$\beta(F_{34} \mathcal{D}) = \beta(F_{34}; a_{34}+s_{34}, b_{34}+t_{34}) = \beta(F_{34}; 0+0, 25+0) = \beta(F_{34}; 0, 25)$
$\beta(F_{41} \mathcal{D}) = \beta(F_{41}; a_{41}+s_{41}, b_{41}+t_{41}) = \beta(F_{41}; 50+1, 0+0) = \beta(F_{41}; 51, 0)$
$\beta(F_{42} \mathcal{D}) = \beta(F_{42}; a_{42}+s_{42}, b_{42}+t_{42}) = \beta(F_{42}; 0+0, 50+0) = \beta(F_{42}; 0, 50)$
$P(O=1 \mathcal{D}) = E(F_{11} \mathcal{D}) = 51/(51+50) \approx 0.505$
$P(I=1 \mathcal{D}) = E(F_{21} \mathcal{D}) = 101/(101+0) = 1$
$P(J=1 O=1, I=1, \mathcal{D}) = E(F_{31} \mathcal{D}) = 26/(26+0) = 1$
$P(J=1 O=1, I=0, \mathcal{D}) = E(F_{32} \mathcal{D}) = 15/(15+10) = 0.6$
$P(J=1 O=0, I=1, \mathcal{D}) = E(F_{33} \mathcal{D}) = 10/(10+15) = 0.4$
$P(J=1 O=0, I=0, \mathcal{D}) = E(F_{34} \mathcal{D}) = 0/(0+25) = 0$
$P(E=1 O=1, \mathcal{D}) = E(F_{41} \mathcal{D}) = 51/(51+0) = 1$
$P(E=1 O=0, \mathcal{D}) = E(F_{42} \mathcal{D}) = 0/(0+50) = 0$
$P(O=0 \mathcal{D}) = 1 - P(O=1 \mathcal{D}) \approx 0.495$
$P(I=0 \mathcal{D}) = 1 - P(I=1 \mathcal{D}) = 0$
$P(J=0 O=1, I=1, \mathcal{D}) = 1 - P(J=1 O=1, I=1, \mathcal{D}) = 0$
$P(J=0 O=1, I=0, \mathcal{D}) = 1 - P(J=1 O=1, I=0, \mathcal{D}) = 0.4$
$P(J=0 O=0, I=1, \mathcal{D}) = 1 - P(J=1 O=0, I=1, \mathcal{D}) = 0.6$
$P(J=0 O=0, I=0, \mathcal{D}) = 1 - P(J=1 O=0, I=0, \mathcal{D}) = 1$
$P(E=0 O=1, \mathcal{D}) = 1 - P(E=1 O=1, \mathcal{D}) = 0$
$P(E=0 O=0, \mathcal{D}) = 1 - P(E=1 O=0, \mathcal{D}) = 1$

Table III.3.3.7. Posterior density functions and posterior probabilities are evolved based on counters s_{ij} and t_{ij}

Note that origin parameters such as $a_{11}=50$, $b_{11}=50$, $a_{21}=100$, $b_{21}=0$, $a_{31}=25$, $b_{31}=0$, $a_{32}=15$, $b_{32}=10$, $a_{33}=10$, $b_{33}=15$, $a_{34}=0$, $b_{34}=25$, $a_{41}=50$, $b_{41}=0$, $a_{42}=0$, and $b_{42}=50$ (see figure III.3.3.2) are kept intact in the task of updating posterior density functions $\beta(F_{11}|\mathcal{D})$, $\beta(F_{21}|\mathcal{D})$, $\beta(F_{31}|\mathcal{D})$, $\beta(F_{32}|\mathcal{D})$, $\beta(F_{33}|\mathcal{D})$, $\beta(F_{34}|\mathcal{D})$, $\beta(F_{41}|\mathcal{D})$, $\beta(F_{42}|\mathcal{D})$, $\beta(F_{51}|\mathcal{D})$, and $\beta(F_{52}|\mathcal{D})$. For example, $\beta(F_{11}|\mathcal{D}) = \beta(F_{11}; a_{11}+s_{11}, b_{11}+t_{11}) = \beta(F_{11}; 50+1, 50+1) = \beta(F_{11}; 51, 51)$. After the updating task, these parameters are changed into new values; concretely, $a_{11}=51$, $b_{11}=50$, $a_{21}=101$, $b_{21}=0$, $a_{31}=26$, $b_{31}=0$, $a_{41}=51$, and $b_{41}=0$. These parameters updated with new values, which are called posterior parameters, are in turn used for the new iteration of EM

algorithm. Please pay attention that such posterior parameters a_{ij} and b_{ij} (s) are calculated based on counters s_{ij} and t_{ij} (s).

By posterior CPT (s) shown in table III.3.3.7 which is the ultimate result of EM algorithm, the Bayesian overlay model of Java course in figure III.3.3.2 is converted into the evolutional version specified in figure III.3.3.3.

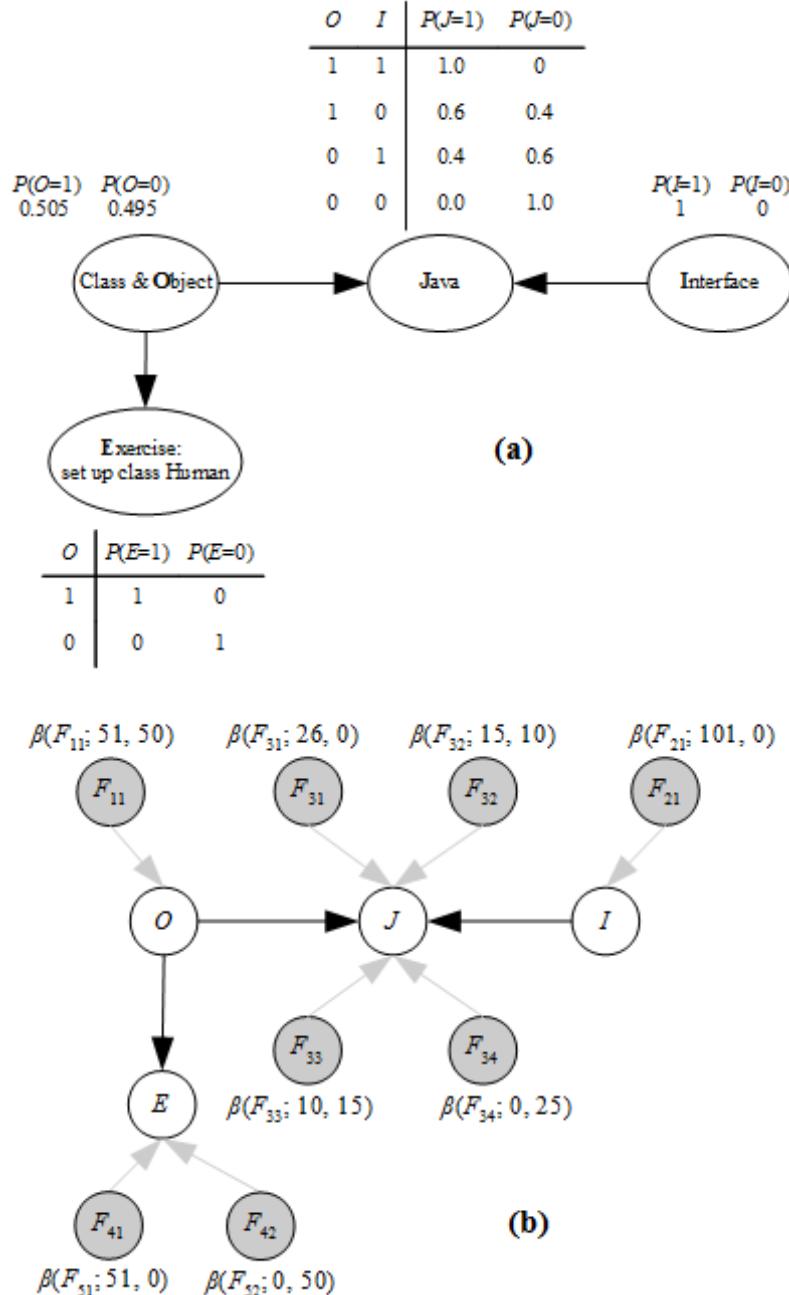


Figure III.3.3.3. Evolutional version of BN (a) and augmented BN (b) for Java course

It is possible to run more iterations for EM algorithm so that the posterior density functions are updated and become more accurate after many iterations because the limit

$\lim_{k \rightarrow +\infty} \frac{a_{ij}^{(k)} + s_{ij}^{(k)}}{a_{ij}^{(k)} + s_{ij}^{(k)} + b_{ij}^{(k)} + t_{ij}^{(k)}}$ will gain certain value; please see previous sub-section III.3.2.

In general, this Java course example is an extension of example in previous

sub-section III.3.2, which help us to know clearly combination of Bayesian network and overlay model (sub-section III.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 missing data.

Section III.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 section III.3 suggests the approach to parameter evolution that uses the EM algorithm for beta functions. Note that particular features of beta function make this suggestion feasible because it is possible to compute the expectation of beta function which is the conditional probability in BN. Whether the EM converges quickly or not depends on how to pre-define the parameters. So, I specify the initial parameters (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.1.2 and III.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 section III.4. Your attention please, as aforementioned there are two ways to improve BN such as parameter learning which was mentioned in section III.3 and structure learning which will be described in next section III.4. Hence, section III.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.4. Improving knowledge sub-model by using dynamic Bayesian network

Normal Bayesian network (BN) described in previous sub-section III.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 is known as static BN opposite to DBN mention in this section III.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 L., 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 documents (Reye, 2004) and (Millán & Pérez-de-la-Cruz, 2002, p. 282) is the good description of concepts “temporal slip” and “lucky guess”.

As aforementioned in section III.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 section III.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.4.2, we should glance over what dynamic Bayesian network is in sub-section III.4.1. Sub-section III.4.3 is the evaluation.

III.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 called DGJPD is product of conditional probability distribution of G_0 and product of all $P_{\rightarrow}(s)$ evaluated at all time points, which is specified in formula III.4.1.1 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.4.1.1. 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 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.4.1.1.

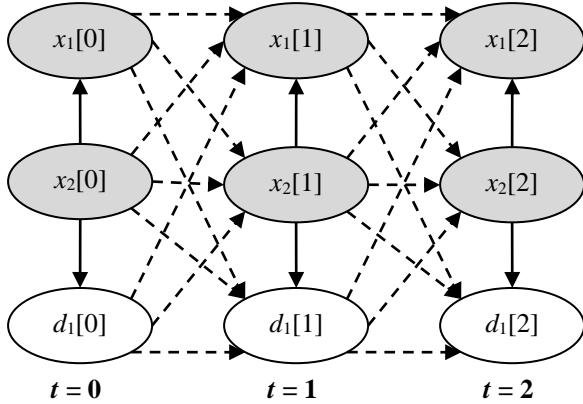


Figure III.4.1.1. DBN for $t = 0, 1, 2$

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.4.1.1, 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 section III.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 criteria; 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, and Russell (Friedman, Murphy, & Russell, 1998) propose criteria 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 δ assigned to every BN, which BN gains highest δ is the best BN. This criterion δ 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.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.4.1.1 is considered as extension of formula III.1.1.8; so, the posterior probability of each temporal variable is now computed by using Distributed Global Joint Probability Distribution (DGJPD) in formula III.4.1.1 which is much more complex than normal Global Joint Probability Distribution (GJPD) in formula III.1.1.8. Whenever the posterior probability of a variable evaluated at 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 for 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.4.1.1. Thus, the proposed algorithm described in next sub-section III.4.2 overcomes the drawbacks of DBN.

III.4.2. Using dynamic Bayesian network to model user's knowledge

In order 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.4.2.1. 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.4.2.1 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.4.2.2. Markov property according to transition probability distribution

If the DBN meets fully formula III.4.2.2, 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 ($t-2, t-3, \dots, 0$). The equivalent statement of Markov property is “the conditional

probability of next time point $t+1$ is only dependent on the current time point t , as follows:

$$P_{\rightarrow}(X[t+1]|X[t], X[t-1], \dots, X[0]) = P_{\rightarrow}(X[t+1]|X[t])$$

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.4.2.1. Six steps of new algorithm for modeling and inferring user's knowledge by using DBN

As seen in table III.4.2.1, 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 ; for example, d_1 is a test or exam. 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.4.2.1. 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.1.2).

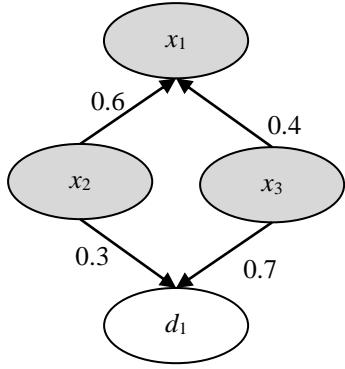


Figure III.4.2.1. The BN sample with full of weights

In figure III.4.2.1, 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.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.4.2.2 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.

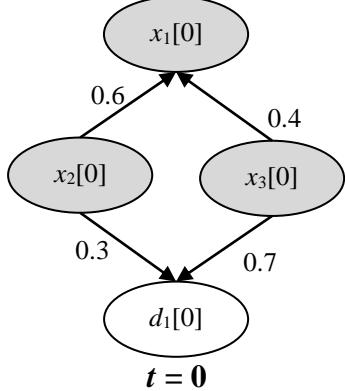


Figure III.4.2.2. Initial BN G_0 derived from BN in figure III.4.2.1

In figure III.4.2.2, 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.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 has not known concept/subject x before but it seems that she/he knows x .

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.4.2.3 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.4.2.3. Formula of slip factor and guess factor

Where $0 \leq \text{slip} \leq 1, 0 \leq \text{guess} \leq 1$ and the sign “*not*” denotes negation operator, for example, $(\text{not } x)=0$ if $x=1$.

So the conditional probability (named a) of event that user knows $x[t]$ given event that she/he has already known $x[t-1]$ has value $1 - \text{slip}$. The conditional probability a is proved and expressed in formula III.4.2.4.

$$a = P(x[t] | x[t-1]) = 1 - P(\text{not } x[t] | x[t-1]) = 1 - \text{slip}$$

Formula III.4.2.4. 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.4.2.5 represents the bias b of user knowledge.

$$b = \frac{1}{1 + P(x[t] | \text{not } x[t-1])} = \frac{1}{1 + \text{guess}}$$

Formula III.4.2.5. 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.1.2). The weight w is specified by formula III.4.2.6.

$$w = a * b = (1 - \text{slip}) \frac{1}{1 + \text{guess}}$$

Formula III.4.2.6. 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.4.2.7 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.4.2.7. 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.4.2.6. Figure III.4.2.3 expresses transition weights between two points in time $t-1$ and t with regard to variables in figures III.4.2.1 and III.4.2.2.

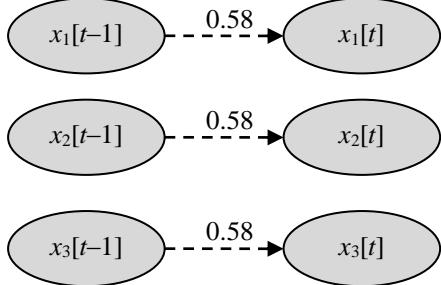


Figure III.4.2.3. Transition weights

In figure III.4.2.3, 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.4.2.1 and III.4.2.7), 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.4.2.8 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.4.2.8. 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.4.2.6. 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.4.2.4. The key of

proposed algorithm in this sub-section III.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 .

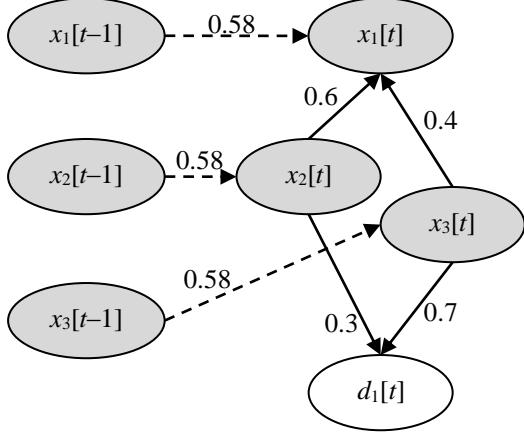


Figure III.4.2.4. DBN or expended BN with full of weights at time point t

In figure III.4.2.4, non-evidence variables (hypothesis variables) are shaded; otherwise, evidence variable ($d_1[t]$) is not shaded. Dash lines - - - denote 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.4.2.6.

$$w_1 = a * b = (1 - \text{slip}) \frac{1}{1 + \text{guess}}$$

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.4.2.9 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.4.2.9. Normalizing weights w_2 , w_3

Suppose S is the sum of w_1 , w_2 and w_3 , we have:

$$\begin{aligned} S &= w_1 + w_2(1 - w_1) + w_3(1 - w_1) = w_1 + (w_2 + w_3)(1 - w_1) = w_1 + (1 - w_1) \\ &= 1 \end{aligned}$$

Expend formula III.4.2.9 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.4.2.10 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.4.2.10. Normalizing weights in general case

After normalizing weights following formula III.4.2.10, transition weight w_{i1} is kept intact but other weights w_{ij} ($j > 1$) get smaller. So the meaning of formula III.4.2.10 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.4.2.4), 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.4.2.10 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.4.2.2 lists weights and normalized weights relating to $x_1[t]$, $x_2[t]$, and $x_3[t]$.

$x_1[t]$	$w_{11}=0.58$	$w_{12}=0.6$	$w_{13}=0.4$
$x_1[t]$ (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.4.2.2. The weights relating $x_1[t]$, $x_2[t]$, and $x_3[t]$ are normalized

The following figure III.4.2.5 shows the variant of expended BN (or DBN) at time point t (shown in figure III.4.2.4) whose weights are normalized.

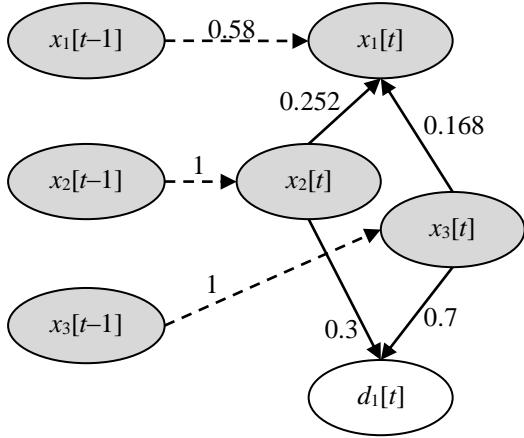


Figure III.4.2.5. DBN or expended BN (at time point t) whose weights are normalized

In figure III.4.2.5, non-evidence variables (hypothesis variables) are shaded; otherwise, evidence variable ($d_1[t]$) is not shaded. Dash lines - - - denote transition probabilities or transition dependencies between consecutive points in time. Now structure of DBN is determined as in figure III.4.2.5 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, as below.

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 $x_i[t] \in X[t]$.

- 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.4.2.11. Posterior probability of each $x_i[t-1]$ given evidences $\mathcal{D}[t-1]$

Formula III.4.2.11 is the same to formula III.4.2.13 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.4.2.11 which are not explained here yet except that formula III.4.2.11 is derived from formula III.1.1.10 described in sub-section III.1.1.

Going back our example of DBN shown in figure III.4.2.5, table III.4.2.3 expresses CPT (s) of $x_1[t-1]$, $x_2[t-1]$, and $x_3[t-1]$ as follows:

$P(x_1[t-1]=1)$	α_1
-----------------	------------

$P(x_1[t-1]=0)$	$1-\alpha_1$
$P(x_2[t-1]=1)$	α_2
$P(x_2[t-1]=0)$	$1-\alpha_2$
$P(x_3[t-1]=1)$	α_3
$P(x_3[t-1]=0)$	$1-\alpha_3$

Table III.4.2.3. 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.4.2.3 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.4.2.13 in step 6. So prior probabilities α_1 , α_2 , and α_3 are posterior probabilities of x_1 , x_2 , and x_3 , respectively computed at previous iteration. Evaluation of α_1 , α_2 , and α_3 will be illustrated in step 6.

- ii. *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.1.3). Applying theorem of SIGMA-gate inference specified in formula III.1.3.4, we have formula III.4.2.12 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.4.2.12. 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.4.2.5, 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 $pa_1[t] = \{x_1[t-1], x_2[t]$ and $x_3[t]\}$. Applying formula III.4.2.12, 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:

$$\begin{aligned} P(x_1[t] = 1 | x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\ = 1 * 0.58 + 1 * 0.252 + 1 * 0.168 = 1 \end{aligned}$$

$$\begin{aligned} P(x_1[t] = 0 | x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\ = 1 - P(x_1[t] = 1 | x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) = 0 \end{aligned}$$

Similarly, it is easy to calculate conditional probabilities of $x_2[t]$ and $x_3[t]$. Table III.4.2.4 shows CPT (s) of $X[t] = x_1[t], x_2[t], x_3[t]$, and $d_1[t]$.

$P(x_1[t]=1 x_1[t-1]=1, x_2[t]=1, x_3[t]=1) = 1*0.58+1*0.252+1*0.168 = 1$
$P(x_1[t]=1 x_1[t-1]=1, x_2[t]=1, x_3[t]=0) = 1*0.58+1*0.252+0*0.168 = 0.832$
$P(x_1[t]=1 x_1[t-1]=1, x_2[t]=0, x_3[t]=1) = 1*0.58+0*0.252+1*0.168 = 0.748$

$P(x_1[t]=1 x_1[t-1]=1, x_2[t]=0, x_3[t]=0) = 1*0.58+0*0.252+0*0.168 = 0.58$
$P(x_1[t]=1 x_1[t-1]=0, x_2[t]=1, x_3[t]=1) = 0*0.58+1*0.252+1*0.168 = 0.42$
$P(x_1[t]=1 x_1[t-1]=0, x_2[t]=1, x_3[t]=0) = 0*0.58+1*0.252+0*0.168 = 0.252$
$P(x_1[t]=1 x_1[t-1]=0, x_2[t]=1, x_3[t]=0) = 0*0.58+0*0.252+1*0.168 = 0.168$
$P(x_1[t]=1 x_1[t-1]=0, x_2[t]=0, x_3[t]=0) = 0*0.58+0*0.252+0*0.168 = 0$
$P(x_1[t]=0 x_1[t-1]=1, x_2[t]=1, x_3[t]=1) = 0$
$P(x_1[t]=0 x_1[t-1]=1, x_2[t]=1, x_3[t]=0) = 0.168$
$P(x_1[t]=0 x_1[t-1]=1, x_2[t]=0, x_3[t]=1) = 0.252$
$P(x_1[t]=0 x_1[t-1]=1, x_2[t]=0, x_3[t]=0) = 0.42$
$P(x_1[t]=0 x_1[t-1]=0, x_2[t]=1, x_3[t]=1) = 0.58$
$P(x_1[t]=0 x_1[t-1]=0, x_2[t]=1, x_3[t]=0) = 0.748$
$P(x_1[t]=0 x_1[t-1]=0, x_2[t]=1, x_3[t]=0) = 0.832$
$P(x_1[t]=0 x_1[t-1]=0, x_2[t]=0, x_3[t]=0) = 1$
$P(x_2[t]=1 x_2[t-1]=1) = 1*1 = 1$
$P(x_2[t]=1 x_2[t-1]=0) = 0*0 = 0$
$P(x_2[t]=0 x_2[t-1]=1) = 0$
$P(x_2[t]=0 x_2[t-1]=0) = 1$
$P(x_3[t]=1 x_3[t-1]=1) = 1*1 = 1$
$P(x_3[t]=1 x_3[t-1]=0) = 0*0 = 0$
$P(x_3[t]=0 x_3[t-1]=1) = 0$
$P(x_3[t]=0 x_3[t-1]=0) = 1$
$P(d_1[t]=1 x_2[t]=1, x_3[t]=1) = 1*0.3+1*0.7 = 1$
$P(d_1[t]=1 x_2[t]=1, x_3[t]=0) = 1*0.3+0*0.7 = 0.3$
$P(d_1[t]=1 x_2[t]=0, x_3[t]=1) = 0*0.3+1*0.7 = 0.7$
$P(d_1[t]=1 x_2[t]=0, x_3[t]=0) = 0*0.3+0*0.7 = 0$
$P(d_1[t]=0 x_2[t]=1, x_3[t]=1) = 0$
$P(d_1[t]=0 x_2[t]=1, x_3[t]=0) = 0.7$
$P(d_1[t]=0 x_2[t]=0, x_3[t]=1) = 0.3$
$P(d_1[t]=0 x_2[t]=0, x_3[t]=0) = 1$

Table III.4.2.4. 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.4.2.5 is totally determined and so, figure III.4.2.6 shows the DBN or expended BN at time point t with full of CPT (s).

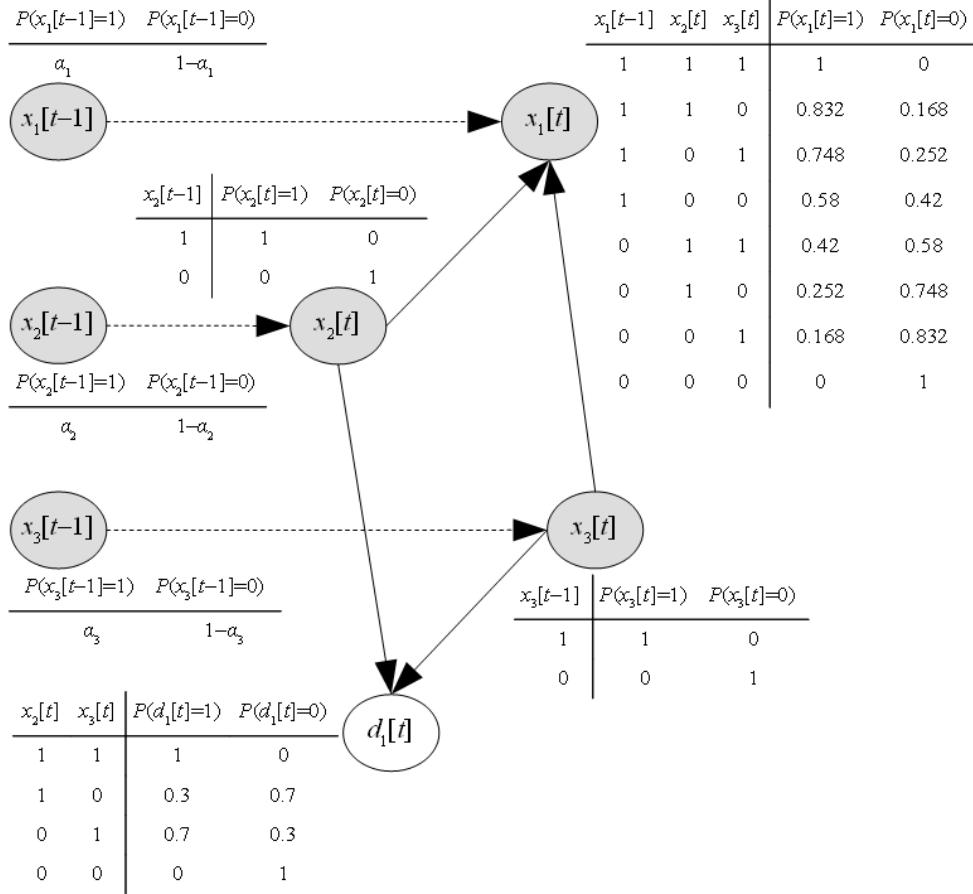


Figure III.4.2.6. DBN at time point t with full of CPT (s)

As seen in figure III.4.2.6, the CPT (s) of $X[t-1] = \{x_1[t-1], x_2[t-1], x_3[t-1]\}$ at current time point t include posterior probabilities α_1, α_2 , and α_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, which is mentioned below.

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.1.1.10. 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 $n*(t+1)$ 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.4.2.13 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.4.2.13. 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.4.2.6, posterior probabilities of $x_1[t]$, $x_2[t]$ and $x_3[t]$ are α_1 , α_2 and α_3 , respectively, which are mentioned in step 5 and shown in table III.4.2.3. 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, which implicates user's true knowledge.

Going back our example of DBN shown in figure III.4.2.6, 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 $\alpha_3=P(x_3[t]=1|\mathcal{D}[t])$ at step 6. 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.1.1.8, we have global joint probability distribution as follows:

$$\begin{aligned} P(x_1[t-1], x_2[t-1], x_3[t-1], x_1[t], x_2[t], x_3[t], d_1[t]) \\ = P(x_1[t-1]) * P(x_2[t-1]) * P(x_3[t-1]) * P(x_2[t]|x_2[t-1]) \\ * P(x_3[t]|x_3[t-1]) * P(x_1[t]|x_1[t-1], x_2[t], x_3[t]) \\ * P(d_1[t]|x_2[t], x_3[t]) \end{aligned}$$

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 α_1 , α_2 , and α_3 calculated at step 6 in previous time point $t-1$. 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:

$$\begin{aligned} P(x_1[t], x_2[t], x_3[t], d_1[t]) \\ = 0.125 * P(x_2[t]|x_2[t-1] = 1) * P(x_3[t]|x_3[t-1] = 1) \\ * P(x_1[t]|x_1[t-1] = 1, x_2[t], x_3[t]) * P(d_1[t] = 1|x_2[t], x_3[t]) \end{aligned}$$

Formula III.4.2.14. An example of DBN global joint probability

Probabilities $P(x_1[t] | x_1[t-1]=1)$, $P(x_2[t] | x_2[t-1]=1)$, $P(x_3[t] | x_3[t-1]=1)$, and $P(d_1[t]=1 | x_2[t], x_3[t])$ are determined in step 5. Please see table III.4.2.4 and figure III.4.2.6 for reviewing CPT (s) of DBN. Therefore, the global joint probability $P(x_1[t], x_2[t], x_3[t], d_1[t])$ is totally determined.

The posterior probability of evidence sample $\mathcal{D}[t] = \{d_1[t]=1\}$ called marginal probability is:

$$\begin{aligned} P(\mathcal{D}[t]) \\ = P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\ + P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\ + P(x_1[t] = 1, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\ + P(x_1[t] = 1, x_2[t] = 0, x_3[t] = 0, d_1[t] = 1) \\ + P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\ + P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\ + P(x_1[t] = 0, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\ + P(x_1[t] = 0, x_2[t] = 0, x_3[t] = 0, d_1[t] = 1) \end{aligned}$$

$$\begin{aligned}
 &= 0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 1) \\
 &+ 0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 0|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 0) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 0) \\
 &+ 0.125 * P(x_2[t] = 0|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 0, x_3[t] = 1) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 0, x_3[t] = 1) \\
 &+ 0.125 * P(x_2[t] = 0|x_2[t-1] = 1) * P(x_3[t] = 0|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 0, x_3[t] = 0) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 0, x_3[t] = 0) \\
 &+ 0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 0|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 1) \\
 &+ 0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 0|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 0|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 0) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 0) \\
 &+ 0.125 * P(x_2[t] = 0|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 0|x_1[t-1] = 1, x_2[t] = 0, x_3[t] = 1) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 0, x_3[t] = 1) \\
 &+ 0.125 * P(x_2[t] = 0|x_2[t-1] = 1) * P(x_3[t] = 0|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 0|x_1[t-1] = 1, x_2[t] = 0, x_3[t] = 0) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 0, x_3[t] = 0) \\
 &\quad \text{(due to formula III.4.2.14)} \\
 &= 0.125 * 1 * 1 * 1 * 1 \\
 &+ 0.125 * 1 * 0 * 0.832 * 0.3 \\
 &+ 0.125 * 0 * 1 * 0.748 * 0.7 \\
 &+ 0.125 * 0 * 0 * 0.58 * 0 \\
 &+ 0.125 * 1 * 1 * 0 * 1 \\
 &+ 0.125 * 1 * 0 * 0.168 * 0.3 \\
 &+ 0.125 * 0 * 1 * 0.252 * 0.7 \\
 &+ 0.125 * 0 * 0 * 0.42 * 0 \\
 &\quad \text{(please see CPT (s) of DBN in table III.4.2.4)} \\
 &= 0.125 + 0 + 0 + 0 + 0 + 0 + 0 = 0.125
 \end{aligned}$$

Similarly, we also have:

$$\begin{aligned}
 &P(x_1[t] = 1, \mathcal{D}[t]) \\
 &= P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 &+ P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\
 &+ P(x_1[t] = 1, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\
 &+ P(x_1[t] = 1, x_2[t] = 0, x_3[t] = 0, d_1[t] = 1) \\
 &= 0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 1) \\
 &+ 0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 0|x_3[t-1] = 1) \\
 &\quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 0) \\
 &\quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 0)
 \end{aligned}$$

$$\begin{aligned}
 & +0.125 * P(x_2[t] = 0|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 0, x_3[t] = 1) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 0, x_3[t] = 1) \\
 & +0.125 * P(x_2[t] = 0|x_2[t-1] = 1) * P(x_3[t] = 0|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 0, x_3[t] = 0) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 0, x_3[t] = 0) \\
 & \quad \text{(due to formula III.4.2.14)} \\
 & = 0.125 * 1 * 1 * 1 * 1 \\
 & +0.125 * 1 * 0 * 0.832 * 0.3 \\
 & +0.125 * 0 * 1 * 0.748 * 0.7 \\
 & +0.125 * 0 * 0 * 0.58 * 0 \\
 & \quad \text{(please see CPT (s) of DBN in table III.4.2.4)} \\
 & = 0.125 + 0 + 0 + 0 = 0.125
 \end{aligned}$$

$$\begin{aligned}
 & P(x_2[t] = 1, \mathcal{D}[t]) \\
 & = P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 0, d_1[t] = 1) \\
 & = 0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 1) \\
 & +0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 0|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 0) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 0) \\
 & +0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 0|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 1) \\
 & +0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 0|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 0|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 0) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 0) \\
 & \quad \text{(due to formula III.4.2.14)} \\
 & = 0.125 * 1 * 1 * 1 * 1 \\
 & +0.125 * 1 * 0 * 0.832 * 0.3 \\
 & +0.125 * 1 * 1 * 0 * 1 \\
 & +0.125 * 1 * 0 * 0.168 * 0.3 \\
 & \quad \text{(please see CPT (s) of DBN in table III.4.2.4)} \\
 & = 0.125 + 0 + 0 + 0 = 0.125
 \end{aligned}$$

$$\begin{aligned}
 & P(x_3[t] = 1, \mathcal{D}[t]) \\
 & = P(x_1[t] = 1, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 1, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 1, x_3[t] = 1, d_1[t] = 1) \\
 & +P(x_1[t] = 0, x_2[t] = 0, x_3[t] = 1, d_1[t] = 1) \\
 & = 0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 1)
 \end{aligned}$$

$$\begin{aligned}
 & +0.125 * P(x_2[t] = 0|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 1|x_1[t-1] = 1, x_2[t] = 0, x_3[t] = 1) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 0, x_3[t] = 1) \\
 & +0.125 * P(x_2[t] = 1|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 0|x_1[t-1] = 1, x_2[t] = 1, x_3[t] = 1) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 1, x_3[t] = 1) \\
 & +0.125 * P(x_2[t] = 0|x_2[t-1] = 1) * P(x_3[t] = 1|x_3[t-1] = 1) \\
 & \quad * P(x_1[t] = 0|x_1[t-1] = 1, x_2[t] = 0, x_3[t] = 1) \\
 & \quad * P(d_1[t] = 1|x_2[t] = 0, x_3[t] = 1) \\
 & \quad \text{(due to formula III.4.2.14)} \\
 & = 0.125 * 1 * 1 * 1 * 1 \\
 & +0.125 * 0 * 1 * 0.748 * 0.7 \\
 & +0.125 * 0 * 0 * 0.58 * 0 \\
 & +0.125 * 1 * 1 * 0 * 1 \\
 & +0.125 * 0 * 1 * 0.252 * 0.7 \\
 & \quad \text{(please see CPT (s) of DBN in table III.4.2.4)} \\
 & = 0.125 + 0 + 0 + 0 = 0.125
 \end{aligned}$$

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 $\alpha_3 = P(x_3[t]=1|\mathcal{D}[t])$.

$$\begin{aligned}
 \alpha_1 &= P(x_1[t] = 1|\mathcal{D}[t]) = \frac{P(x_1[t] = 1, \mathcal{D}[t])}{P(\mathcal{D}[t])} = \frac{0.125}{0.125} = 1 \\
 \alpha_2 &= P(x_2[t] = 1|\mathcal{D}[t]) = \frac{P(x_2[t] = 1, \mathcal{D}[t])}{P(\mathcal{D}[t])} = \frac{0.125}{0.125} = 1 \\
 \alpha_3 &= P(x_3[t] = 1|\mathcal{D}[t]) = \frac{P(x_3[t] = 1, \mathcal{D}[t])}{P(\mathcal{D}[t])} = \frac{0.125}{0.125} = 1
 \end{aligned}$$

(due to Bayes' rule specified by formula III.1.1.1)

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.4.2.5 summarizes the posterior probabilities α_1 , α_2 , and α_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.4.2.5. The results of probabilistic inference – posterior probabilities

After the step 6 is finished, if new evidences occur, there are two options for executing a new iteration as follows:

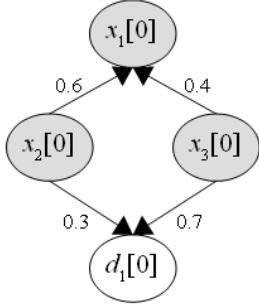
- Turning back step 5 if the structure of DBN is kept intact. This option is more effective because the algorithm will run faster. This option indicates that the algorithm leans towards parameter learning instead of structure learning.
- Turning back step 3 if the structure of DBN needs re-constructed for some improvements. It is even possible to turn back step 2 if it is necessary to change slip factor and guess factor.

As usual, the algorithm runs non-stop in order to monitor chronologically user's process of gaining knowledge.

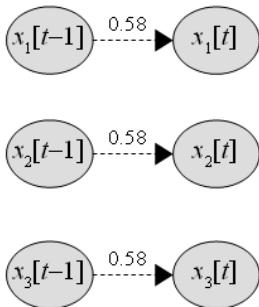
In general, the proposed algorithm to construct DBN based on Markov property and stationary property is iterative process, which has many iterations. Each iteration

includes at most six aforementioned steps creating a closed cycle in which the posterior probabilities at step 6 of previous iteration become prior probabilities at step 5 of current iteration so that the DBN (expended BN) is improved continuously. The closed cycle is executed continuously whenever evidences occur. Figure III.4.2.7 shows such cycle.

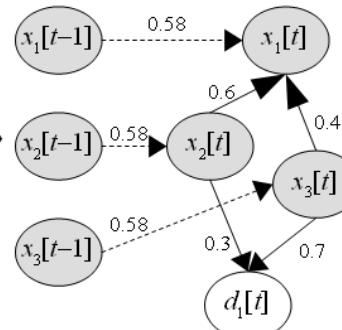
1. Initializing DBN



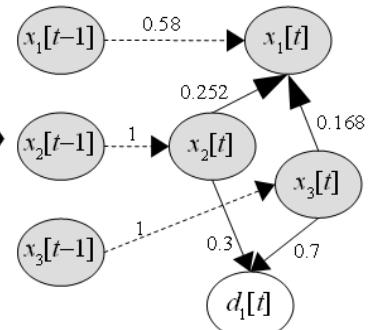
2. Specifying transition weights



3. Re-constructing

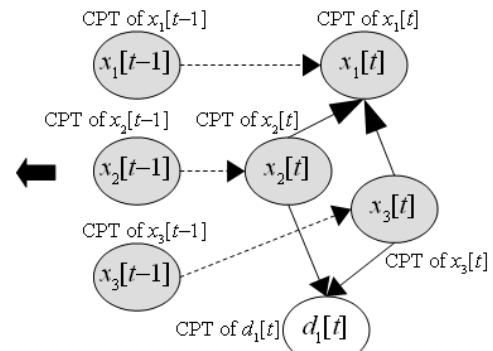


4. Normalizing weights



$$\begin{aligned}
 a_1 &= P(x_1[t]=1 | D[t]) = 1 \\
 a_2 &= P(x_2[t]=1 | D[t]) = 1 \\
 a_3 &= P(x_3[t]=1 | D[t]) = 1
 \end{aligned}$$

6. Probabilistic inference



5. Re-defining CPT (s)

Figure III.4.2.7. The cycle of proposed algorithm to construct DBN

III.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 computational formulas of two factors: *slip* and *guess* in order 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 $n*(t+1)$ 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 at most six steps is done in succession through many iterations, the result of current iteration will be input for next iteration. After the t^{th} iteration, the posterior probabilities of all variables $x_i[t]$ gain stable values and the DBN converges.

Instead of using DBN to re-construct network, another approach described in next section [III.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 section [III.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.3.1](#) and [III.3.2](#) are parameter learning techniques (opposite to structure learning technique based on DBN mentioned in this section [III.4](#)) but their difference will be made clear. Please read next section [III.5](#) in order to release your curiousness.

III.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 and 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](#) 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 this 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, a simple equation was constructed so that differential equations are eliminated and it becomes much easier to estimate parameters in case that such parameters are positive integer numbers (Nguyen L. , Specifying Prior Probabilities in Bayesian Network by Maximum Likelihood Estimation method, 2016). The algorithm to find out the approximate solutions of these simple equations is also proposed in this study. Recall that there are two ways to improve BN, parameter learning, which is also known as evolution of [Bayesian overlay model](#) mentioned in previous section [III.3](#) and structure learning, which is based on dynamic Bayesian network described in previous section [III.4](#). The proposed algorithm specifying prior probabilities introduced in this section [III.5](#) is essentially a parameter learning technique like evolution of Bayesian network mentioned in previous section [III.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 no 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 for more details about BN. The strengths of relationships 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 probabilities.

Every node X in BN is a binary random variable. As aforementioned in sub-section III.3.1, each variable X is attached by a parameter variable F so that the probability density function (PDF) of such variable F represents CPT of X . Please see sub-section III.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.5.1.

$$\beta(F; a, b) = \text{beta}(F; a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} F^{a-1} (1 - F)^{b-1}$$

Formula III.5.1. 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.3.1.2. Please see sub-section III.3.1 for more details about parameter variable F and beta density function. Because it is convenient for readers to keep track main content in this section III.5, formula III.5.1 is replication of formula III.3.1.1 for specifying beta density function. The expectation $E(F)$ and the variance $Var(F)$ of parameter 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.5.2. Probability of variable X as expectation of beta variable F

Because it is convenient for readers to keep track main content in this section III.5, formula III.5.2 is the same to formula III.3.1.6. Note that the equation $E(F) = E(\beta(a, b))$ implicates that parameter 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.5.3 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.5.3. 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.5.3 is the replication of formula III.3.1.10 is to give readers convenience for keeping track main content in this section III.5. Note that the equation $E(F|\mathcal{D}) = E(\beta(a, b)|\mathcal{D})$ implicates that parameter variable F is identified with its beta distribution $\beta(a, b)$.

It is recognized 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.5.1 discusses some basic concepts of maximum likelihood estimation (MLE) technique. Sub-section III.5.2 considers applying MLE into beta distribution (beta density function). Sub-section III.5.3 – the main sub-section describes the proposed algorithm to estimate two parameters a and b of beta distribution. Sub-section III.5.3 also proposes the simple equations whose solutions are estimates of positive parameters a and b . Sub-section III.5.4 illustrates the proposed algorithm by example. Sub-section III.5.5 introduces the new version of simple equations mentioned in sub-section III.5.4. Sub-section III.5.6 is evaluation of the algorithm proposed in the main sub-section III.5.3.

III.5.1. Maximum likelihood estimation

Let Θ 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. This means that x_i (s) and X have the same probability distribution or the same probability density function (PDF). Please see formula III.1.1.6 for more details about PDF and probability distribution. According multiplication rule (see formula III.1.1.3) 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 Θ (Lynch, 2007, p. 36). Formula III.5.1.1 expresses the likelihood function $L(\Theta)$ with regard to variable Θ .

$$L(\Theta) = P(X | \Theta) = \prod_{i=1}^n P(x_i | \Theta)$$

Formula III.5.1.1. Likelihood function

Where $P(x_i | \Theta)$ is the conditional probability of instance x_i given the hypothesis Θ . 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.5.1.2. 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” exchangeably. 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.5.1.2. 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, as shown in formula III.5.1.1, into form of repeated addition, as shown in formula III.5.1.3 (Lynch, 2007, p. 38). The natural logarithm of $L(\Theta)$, which is called log-likelihood, is denoted $LnL(\Theta)$, as shown in formula III.5.1.3.

$$LnL(\Theta) = \ln \left(\prod_{i=1}^n P(x_i | \Theta) \right) = \sum_{i=1}^n \ln(P(x_i | \Theta))$$

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}} LnL(\Theta) = \underset{\Theta}{\operatorname{argmax}} \left(\sum_{i=1}^n \ln(P(x_i | \Theta)) \right)$$

Formula III.5.1.3. 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 θ_i to 0 and solving this equation to find out parameter θ_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” exchangeably.

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 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.5.1.4 specifies the co-variance matrix of parameter vector Θ .

$$Var(\Theta) = \left(-E \left(\frac{\partial^2 LnL}{\partial \Theta \partial \Theta^T} \right) \right)^{-1}$$

Formula III.5.1.4. 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. Exactly, $\left(-E \left(\frac{\partial^2 LnL}{\partial \Theta \partial \Theta^T} \right) \right)^{-1}$ is a so-called Cramer-Rao lower bound of co-variance matrix. However, $\left(-E \left(\frac{\partial^2 LnL}{\partial \Theta \partial \Theta^T} \right) \right)^{-1}$ is equal to co-variance matrix if $\hat{\Theta}$ 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.5.2 – “Beta likelihood estimation” discusses how to apply MLE into beta distribution (beta density function). This sub-section is also summarized in (Nguyen L. , Beta Likelihood Estimation and Its Application to Specify Prior Probabilities in Bayesian Network, 2016).

III.5.2. Beta likelihood estimation

As discussed, each variable X within BN is attached by a parameter variable F and variable F , in turn, has beta distribution $\beta(F; a, b)$ specified in the formula III.5.1. For convenience, the beta density function specified by previous formula III.5.1 is re-written as formula III.5.2.1.

$$f(F; a, b) = \beta(F; a, b) = beta(F; a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} F^{a-1} (1 - F)^{b-1}$$

Formula III.5.2.1. Beta density function (beta distribution) $\beta(F; a, b)$

Where $\Gamma(.)$ denotes gamma function which is expressed by formula III.3.1.2.

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

Note that $e^{(.)}$ and $exp(.)$ 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.5.2.2 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.5.2.2. Definition of digamma function

Note that $\ln(.)$ denotes natural logarithm function. According to formula III.5.2.2, digamma function is the derivative of natural logarithm of gamma function.

The integral form of digamma function is specified by formula III.5.2.3 (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.5.2.3. 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.5.2.4 (Wikipedia, Digamma function, 2014).

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

Formula III.5.2.4. Recurrence formula of digamma function

Formula III.5.2.4 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 a positive integer. Thus, it is necessary to calculate $\psi(1)$ which is the evaluation of digamma function at starting positive point 1, we have:

$$\begin{aligned} \psi(1) &= \int_0^{+\infty} \left(\frac{e^{-t}}{t} - \frac{e^{-t}}{1-e^{-t}} \right) dt = \int_0^{+\infty} \frac{e^{-t}}{t} dt - \int_0^{+\infty} \frac{e^{-t}}{1-e^{-t}} dt \\ &= \int_0^{+\infty} e^{-t} d(\ln t) - \int_0^{+\infty} d(\ln(1-e^{-t})) \\ &= e^{-t} \ln t \Big|_0^{+\infty} + \int_0^{+\infty} e^{-t} \ln t dt - \ln(1-e^{-t}) \Big|_0^{+\infty} \\ &= \lim_{t \rightarrow +\infty} (e^{-t} \ln t) - \lim_{t \rightarrow 0} (e^{-t} \ln t) - \gamma - \lim_{t \rightarrow +\infty} \ln(1-e^{-t}) + \lim_{t \rightarrow 0} \ln(1-e^{-t}) \\ &\quad (\text{due to Euler-Mascheroni constant } \gamma = -\int_0^{+\infty} e^{-t} \ln t dt) \\ &= -\gamma + \lim_{t \rightarrow +\infty} (e^{-t} \ln t) - \lim_{t \rightarrow 0} (e^{-t} \ln t) + \lim_{t \rightarrow 0} \ln(1-e^{-t}) \\ &= -\gamma + \lim_{t \rightarrow +\infty} \frac{\ln t}{e^t} + \lim_{t \rightarrow 0} (\ln(1-e^{-t}) - e^{-t} \ln t) \\ &= -\gamma + \lim_{t \rightarrow +\infty} \frac{\ln t}{e^t} + \lim_{t \rightarrow 0} \ln \frac{e^{\ln(1-e^{-t})}}{e^{e^{-t} \ln t}} \\ &\quad (\text{due to transformation with regard to indeterminate form (Wikipedia, Indeterminate form, 2014)}) \\ &= -\gamma + \lim_{t \rightarrow +\infty} \frac{\ln t}{e^t} + \lim_{t \rightarrow 0} \ln \frac{1-e^{-t}}{t^{e^{-t}}} \end{aligned}$$

$$= -\gamma + \lim_{t \rightarrow +\infty} \frac{\ln t}{e^t} + \ln \left(\lim_{t \rightarrow 0} \frac{1 - e^{-t}}{t^{e^{-t}}} \right)$$

$$= -\gamma + \lim_{t \rightarrow +\infty} \frac{d(\ln t)}{d(e^t)} + \ln \left(\lim_{t \rightarrow 0} \frac{1 - e^{-t}}{t^{e^{-t}}} \right)$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator
(Wikipedia, Indeterminate form, 2014))

$$= -\gamma + \lim_{t \rightarrow +\infty} \frac{1}{t e^t} + \ln \left(\lim_{t \rightarrow 0} \frac{1 - e^{-t}}{t^{e^{-t}}} \right)$$

$$= -\gamma + \ln \left(\lim_{t \rightarrow 0} \frac{1 - e^{-t}}{t^{e^{-t}}} \right)$$

$$= -\gamma + \ln \left(\lim_{t \rightarrow 0} \frac{e^{-t}}{g'(t)} \right) \text{ where } g(t) = t^{e^{-t}}$$

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:

$$\lim_{t \rightarrow 0} (t^{e^{-t}-1}) = \lim_{t \rightarrow 0} \exp \left(\frac{e^{-t} - 1}{1/\ln t} \right) = \exp \left(\lim_{t \rightarrow 0} \frac{e^{-t} - 1}{1/\ln t} \right)$$

(due to transformation with regard to indeterminate form (Wikipedia, Indeterminate form, 2014))

$$= \exp \left(\lim_{t \rightarrow 0} \frac{d(e^{-t} - 1)}{d(1/\ln t)} \right) = \exp \left(\lim_{t \rightarrow 0} \frac{-e^{-t}}{-\frac{1}{t(\ln t)^2}} \right)$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator
(Wikipedia, Indeterminate form, 2014))

$$= \exp \left(\lim_{t \rightarrow 0} e^{-t} t (\ln t)^2 \right) = \exp \left(\lim_{t \rightarrow 0} t (\ln t)^2 \right) = \exp \left(\lim_{t \rightarrow 0} \frac{(\ln t)^2}{1/t} \right)$$

$$= \exp \left(\lim_{t \rightarrow 0} \frac{d((\ln t)^2)}{d(1/t)} \right) = \exp \left(\lim_{t \rightarrow 0} \frac{2 \ln t / t}{-1/t^2} \right)$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator
(Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}
 &= \exp\left(-\lim_{t \rightarrow 0} 2t \ln t\right) = \exp\left(-\lim_{t \rightarrow 0} \frac{2 \ln t}{1/t}\right) \\
 &= \exp\left(-\lim_{t \rightarrow 0} \frac{d(2 \ln t)}{d(1/t)}\right) = \exp\left(\lim_{t \rightarrow 0} \frac{2/t}{1/t^2}\right) \\
 &\quad (\text{using L'Hôpital's rule by taking derivatives of both numerator and denominator}) \\
 &\quad (\text{Wikipedia, Indeterminate form, 2014}) \\
 &= \exp\left(\lim_{t \rightarrow 0} 2t\right) = \exp(0) = 1
 \end{aligned}$$

We also have:

$$\begin{aligned}
 \lim_{t \rightarrow 0}(1 - t \ln t) &= 1 - \lim_{t \rightarrow 0}(t \ln t) = 1 - \lim_{t \rightarrow 0} \frac{\ln t}{1/t} \\
 &= 1 - \lim_{t \rightarrow 0} \frac{d(\ln t)}{d(1/t)} = 1 + \lim_{t \rightarrow 0} \frac{1/t}{1/t^2} \\
 &\quad (\text{using L'Hôpital's rule by taking derivatives of both numerator and denominator}) \\
 &\quad (\text{Wikipedia, Indeterminate form, 2014}) \\
 &= 1 + \lim_{t \rightarrow 0} t = 1 + 0 = 1
 \end{aligned}$$

Therefore, it implies that:

$$\psi(1) = -\gamma + \ln\left(\frac{1}{\lim_{t \rightarrow 0}(t^{e^{-t}-1}) \lim_{t \rightarrow 0}(1 - t \ln t)}\right) = -\gamma + \ln\left(\frac{1}{1 * 1}\right) = -\gamma$$

Briefly, the value $\psi(1)$ is always equal to $-\gamma$ (Wikipedia, Digamma function, 2014). Given x is positive integer, formula III.5.2.3 is replaced by formula III.5.2.5 for calculating digamma function in case of positive integer number.

$$\begin{aligned}
 \psi(1) &= -\gamma \\
 \psi(x) &= -\gamma + \sum_{k=1}^{x-1} \frac{1}{k} \\
 &\quad (x \text{ positive integer and } x \geq 2)
 \end{aligned}$$

Formula III.5.2.5. Digamma function in case of positive integer variable

Proof,

$$\begin{aligned}
 \forall x \geq 2, \psi(x) &= \psi(x-1) + \frac{1}{x-1} = \psi(x-2) + \frac{1}{x-2} + \frac{1}{x-1} \\
 &\quad (\text{by applying formula III.5.2.4}) \\
 &= \psi(x-3) - \frac{1}{x+3} - \frac{1}{x+2} - \frac{1}{x+1} \\
 &= \dots = \psi(x-(x-1)) + \underbrace{\frac{1}{x-(x-1)} + \frac{1}{x-(x-2)} + \dots + \frac{1}{x-1}}_{x-1} \\
 &= \psi(x-(x-1)) + \underbrace{\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{x-1}}_{x-1} \\
 &= \psi(1) + \sum_{k=1}^{x-1} \frac{1}{k} = -\gamma + \sum_{k=1}^{x-1} \frac{1}{k}
 \end{aligned}$$

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), (Wikipedia, Polygamma function, 2014), and (Weisstein, Trigamma Function) for more details about trigamma function. Briefly, formula III.5.2.6 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.5.2.6. Trigamma function

Suppose variable x is non-zero, we have:

$$\begin{aligned} \psi_1(x+1) &= \int_0^{+\infty} \frac{te^{-(x+1)t}}{1-e^{-t}} dt = \int_0^{+\infty} \left(\frac{te^{-xt}}{1-e^{-t}} - te^{-xt} \right) dt \\ &= \int_0^{+\infty} \frac{te^{-xt}}{1-e^{-t}} dt - \int_0^{+\infty} te^{-xt} dt = \psi_1(x) + \frac{1}{x} \int_0^{+\infty} t d(e^{-xt}) dt \\ &= \psi_1(x) + \frac{1}{x} te^{-xt} \Big|_0^{+\infty} - \frac{1}{x} \int_0^{+\infty} e^{-xt} dt \\ &= \psi_1(x) + \frac{1}{x} te^{-xt} \Big|_0^{+\infty} + \frac{1}{x^2} e^{-xt} \Big|_0^{+\infty} \\ &= \psi_1(x) + \frac{1}{x} \lim_{t \rightarrow +\infty} (te^{-xt}) - \frac{1}{x} \lim_{t \rightarrow 0} (te^{-xt}) + \frac{1}{x^2} \lim_{t \rightarrow +\infty} e^{-xt} - \frac{1}{x^2} \lim_{t \rightarrow 0} e^{-xt} \\ &= \psi_1(x) + \frac{1}{x} \lim_{t \rightarrow +\infty} (te^{-xt}) - \frac{1}{x^2} = \psi_1(x) - \frac{1}{x^2} + \frac{1}{x} \lim_{t \rightarrow +\infty} \frac{t}{e^{xt}} \\ &= \psi_1(x) - \frac{1}{x^2} + \frac{1}{x} \lim_{t \rightarrow +\infty} \frac{d(t)}{d(e^{xt})} = \psi_1(x) - \frac{1}{x^2} + \frac{1}{x} \lim_{t \rightarrow +\infty} \frac{1}{xe^{xt}} \end{aligned}$$

(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.5.2.7
 (Wikipedia, Trigamma function, 2015).

$$\psi_1(x+1) = \psi_1(x) - \frac{1}{x^2}$$

Formula III.5.2.7. Recurrence formula of trigamma function

Formula III.5.2.7 shows recurrence relation (Wikipedia, Trigamma function, 2015) 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 $\psi_1(1)$ which is the evaluation of trigamma function at starting positive point 1, we have:

$$\begin{aligned} \psi_1(x) &= \psi'(x) = \frac{d}{dx} \left(\frac{\Gamma'(x)}{\Gamma(x)} \right) \\ &= \frac{\Gamma''(x)\Gamma(x) - \Gamma'(x)\Gamma'(x)}{\Gamma(x)\Gamma(x)} = \frac{\Gamma''(x)}{\Gamma(x)} - \left(\frac{\Gamma'(x)}{\Gamma(x)} \right)^2 \\ &= \frac{\Gamma''(x)}{\Gamma(x)} - (\psi(x))^2 \\ \Rightarrow \psi_1(1) &= \frac{\Gamma''(1)}{\Gamma(1)} - (\psi(1))^2 = \frac{\Gamma''(1)}{\Gamma(1)} - \gamma^2 \end{aligned}$$

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 $\int_0^{+\infty} e^{-t} (lnt)^2 dt = \gamma^2 + \frac{\pi^2}{6}$ is found out in (Weisstein, Euler-Mascheroni Constant).

It implies that

$$\psi_1(1) = \frac{\Gamma''(1)}{\Gamma(1)} - \gamma^2 = \frac{\gamma^2 + \frac{\pi^2}{6}}{1} - \gamma^2 = \frac{\pi^2}{6}$$

Briefly, the value $\psi_1(1)$ is always equal to $\frac{\pi^2}{6}$ (Wikipedia, Trigamma function, 2015). Given x is positive integer, formula III.5.2.6 is replaced by formula III.5.2.8 for calculating trigamma function in case of positive integer number, as follows:

$$\begin{aligned}\psi_1(1) &= \frac{\pi^2}{6} \\ \psi_1(x) &= \frac{\pi^2}{6} - \sum_{k=1}^{x-1} \frac{1}{k^2} \\ (x \text{ positive integer and } x &\geq 2)\end{aligned}$$

Formula III.5.2.8. Trigamma function in case of positive integer variable

Proof,

$$\begin{aligned}\forall x \geq 2, \psi_1(x) &= \psi_1(x-1) - \frac{1}{(x-1)^2} = \psi_1(x-2) - \left(\frac{1}{(x-2)^2} + \frac{1}{(x-1)^2} \right) \\ &\quad (\text{by applying formula III.5.2.7}) \\ &= \psi_1(x-3) - \left(\frac{1}{(x-3)^2} + \frac{1}{(x-2)^2} + \frac{1}{(x-1)^2} \right) \\ &= \dots \\ &= \psi_1(x-(x-1)) - \underbrace{\left(\frac{1}{(x-(x-1))^2} + \frac{1}{(x-(x-2))^2} + \dots + \frac{1}{(x-1)^2} \right)}_{x-1} \\ &== \psi_1(x-(x-1)) - \underbrace{\left(\frac{1}{1^2} + \frac{1}{2^2} + \dots + \frac{1}{(x-1)^2} \right)}_{x-1} \\ &= \psi_1(1) - \sum_{k=1}^{x-1} \frac{1}{k^2} = \frac{\pi^2}{6} - \sum_{k=1}^{x-1} \frac{1}{k^2}\end{aligned}$$

In general, the two formulas III.5.2.5 and III.5.2.8 have been derived by this method. They will be used to calculate digamma function and trigamma function with regard to positive variable.

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.5.2.9. Beta function $B(x, y)$

Please distinguish beta density function $\beta(X; a, b)$ specified in formulas III.5.2.1, III.5.1, and III.3.1.1 known as probability density function (PDF) from beta function $B(x, y)$ specified by formula III.5.2.9.

The first-order partial derivative of $B(x, y)$ is determined as follows:

$$\begin{aligned}
 \frac{\partial B(x, y)}{\partial x} &= \Gamma(y) \left(\frac{\Gamma'(x)\Gamma(x+y) - \Gamma(x)\Gamma'(x+y)}{(\Gamma(x+y))^2} \right) \\
 &= \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \frac{\Gamma'(x)\Gamma(x+y) - \Gamma(x)\Gamma'(x+y)}{\Gamma(x)\Gamma(x+y)} \\
 &= B(x, y) \frac{\Gamma'(x)\Gamma(x+y) - \Gamma(x)\Gamma'(x+y)}{\Gamma(x)\Gamma(x+y)} \\
 &= B(x, y) \left(\frac{\Gamma'(x)}{\Gamma(x)} - \frac{\Gamma'(x+y)}{\Gamma(x+y)} \right)
 \end{aligned}$$

Due to digamma function $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$, we have formula III.5.2.10 to specify the first-order partial derivative of beta function:

$$\frac{\partial B(x, y)}{\partial x} = B(x, y)(\psi(x) - \psi(x+y))$$

Formula III.5.2.10. First-order partial derivative of beta function $B(x, y)$

The digamma function is always determined by formulas III.5.2.3 and III.5.2.5. Substituting beta function $B(x, y)$ specified formula III.5.2.9 into formula III.5.2.1, the beta density function is re-written:

$$f(F; a, b) = \beta(F; a, b) = \frac{1}{B(a, b)} F^{a-1} (1-F)^{b-1}$$

Formula III.5.2.11. Beta density function with regard to beta function

Now we specify the likelihood function of beta distribution by applying formula III.5.2.11 as below:

$$\begin{aligned}
 L(a, b) &= \prod_{i=1}^n \beta(f_i; a, b) = \prod_{i=1}^n \frac{1}{B(a, b)} f_i^{a-1} (1-f_i)^{b-1} \\
 &= \frac{1}{B^n(a, b)} \left(\prod_{i=1}^n f_i^{a-1} \right) \left(\prod_{i=1}^n (1-f_i)^{b-1} \right)
 \end{aligned}$$

Where f_i is physical frequency of a specified event in sample \mathcal{D}_i with note that $0 < f_i < 1$ and so this likelihood can use n samples. Taking the logarithm of $L(a, b)$, we have the log-likelihood function for beta distribution as follows (Wikipedia, Beta distribution, 2018):

$$\begin{aligned}
 LnL(a, b) &= \ln(L(a, b)) \\
 &= n \ln\left(\frac{1}{B(a, b)}\right) + \sum_{i=1}^n ((a-1) \ln(f_i)) + \sum_{i=1}^n ((b-1) \ln(1-f_i)) \\
 &= -n \ln(B(a, b)) + (a-1) \sum_{i=1}^n \ln(f_i) + (b-1) \sum_{i=1}^n \ln(1-f_i)
 \end{aligned}$$

Formula III.5.2.12. Log-likelihood function of beta density function (beta distribution)

Where $\ln(\cdot)$ denotes natural logarithm function.

Figure III.5.2.1 (Wolfram) shows the graph of log-likelihood function specified by formula III.5.2.12 with regard to variables a and b given $f_1=0.1$ and $f_2=0.2$.

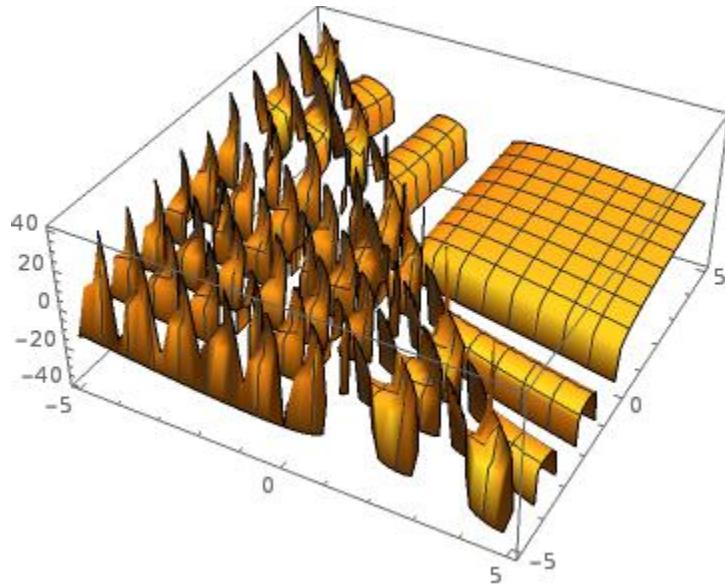


Figure III.5.2.1. Log-likelihood function with regard to variables a and b

Please pay attention to formula III.5.2.12 because formula III.5.2.12 is specific case of formula III.5.1.3 mentioned in previous sub-section III.5.1; thus, MLE is applied into beta distribution.

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.5.2.12 corresponding to two parameters and by applying formula III.5.2.10, we have:

$$\begin{aligned}\frac{\partial \ln L(a, b)}{\partial a} &= -n \frac{1}{B(a, b)} \frac{\partial B(a, b)}{\partial a} + \sum_{i=1}^n \ln(f_i) \\ &= -n(\psi(a) - \psi(a + b)) + \sum_{i=1}^n \ln(f_i)\end{aligned}$$

Formula III.5.2.13. First-order partial derivative of log-likelihood function of beta density function with regard to parameter a

$$\begin{aligned}\frac{\partial \ln L(a, b)}{\partial b} &= -n \frac{1}{B(a, b)} \frac{\partial B(a, b)}{\partial b} + \sum_{i=1}^n \ln(1 - f_i) \\ &= -n(\psi(b) - \psi(a + b)) + \sum_{i=1}^n \ln(1 - f_i)\end{aligned}$$

Formula III.5.2.14. 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 specified by formulas III.5.2.3 and III.5.2.5. Note that notation $\frac{\partial g}{\partial x}$ denotes first-order partial derivative of multi-variable function g with regard to variable x .

Please pay attention to formulas III.5.2.13 and III.5.2.14 for determining two first-order partial derivatives of log-likelihood function of beta distribution. By 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 :

$$\begin{aligned} \left\{ \begin{array}{l} \frac{\partial \ln L(a, b)}{\partial a} = 0 \\ \frac{\partial \ln L(a, b)}{\partial b} = 0 \end{array} \right. &\Leftrightarrow \left\{ \begin{array}{l} -n(\psi(a) - \psi(a + b)) + \sum_{i=1}^n \ln(f_i) = 0 \\ -n(\psi(b) - \psi(a + b)) + \sum_{i=1}^n \ln(1 - f_i) = 0 \end{array} \right. \\ &\Leftrightarrow \left\{ \begin{array}{l} \psi(a) - \psi(a + b) = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ \psi(b) - \psi(a + b) = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i) \end{array} \right. \end{aligned}$$

Formula III.5.2.15. The set of differential equations for estimating parameters a and b

Formula III.5.2.15 shows the set of differential equations for estimating parameters a and b . In the next sub-section III.5.3, the simple form of such equations is invented when parameters a and b are positive integer numbers. Furthermore, the algorithm to find out the approximate solutions is derived.

Before going into the main sub-section III.5.3, it is necessary to glance over the co-variance matrix $Var(a, b)$ of parameters of beta density function mentioned in previous sub-section III.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 called Hessian matrix, we have:

$$H(a, b) = \begin{pmatrix} \frac{\partial^2 \ln L}{\partial a^2} & \frac{\partial^2 \ln L}{\partial a \partial b} \\ \frac{\partial^2 \ln L}{\partial b \partial a} & \frac{\partial^2 \ln L}{\partial b^2} \end{pmatrix}$$

Note, the bracket (.) denotes matrix.

Basing on formulas III.5.2.13 and III.5.2.14, we can determine four second-order partial derivatives of log-likelihood function as follows:

$$\frac{\partial^2 \ln L}{\partial a^2} = \frac{\partial \psi(a + b)}{\partial a} - n \frac{\partial \psi(a)}{\partial a} = \psi_1(a + b) - n\psi_1(a)$$

$$\frac{\partial^2 \ln L}{\partial a \partial b} = \frac{\partial \psi(a + b)}{\partial b} = \psi_1(a + b)$$

$$\frac{\partial^2 \ln L}{\partial b \partial a} = \frac{\partial \psi(a + b)}{\partial a} = \psi_1(a + b)$$

$$\frac{\partial^2 \ln L}{\partial b^2} = \frac{\partial \psi(a+b)}{\partial b} - n \frac{\partial \psi(b)}{\partial b} = \psi_1(a+b) - n\psi_1(b)$$

Where $\psi_1(\cdot)$ denotes trigamma function specified by formulas III.5.2.6 and III.5.2.8. According to formula III.5.1.4, the co-variance matrix $Var(a, b)$ is the inversion of negative expectation of Hessian matrix. Please read the book “Linear Algebra” by author Viet-Hung Huu Nguyen (Nguyen V. H., 1999, p. 134) and the book “Linear Algebra and Its Applications” by author Lay (Lay, 2012, pp. 102-109) for more details of how to take inversion of a given matrix. We have:

$$\begin{aligned} Var(a, b) &= (-E(H(a, b)))^{-1} \\ &= \left(\begin{array}{cc} -E(\psi_1(a+b) - n\psi_1(a)) & \psi_1(a+b) \\ \psi_1(a+b) & \psi_1(a+b) - n\psi_1(b) \end{array} \right)^{-1} \\ &= \left(\begin{array}{cc} -(\psi_1(a+b) - n\psi_1(a)) & \psi_1(a+b) \\ \psi_1(a+b) & \psi_1(a+b) - n\psi_1(b) \end{array} \right)^{-1} \end{aligned}$$

(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 $H(a, b)$)

$$\begin{aligned} &= \left(\begin{array}{cc} n\psi_1(a) - \psi_1(a+b) & -\psi_1(a+b) \\ -\psi_1(a+b) & n\psi_1(b) - \psi_1(a+b) \end{array} \right)^{-1} \\ &= \frac{1}{n^2\psi_1(a)\psi_1(b) - n\psi_1(a+b)(\psi_1(a) + \psi_1(b))} \\ &\quad * \left(\begin{array}{cc} n\psi_1(b) - \psi_1(a+b) & \psi_1(a+b) \\ \psi_1(a+b) & n\psi_1(a) - \psi_1(a+b) \end{array} \right) \end{aligned}$$

Briefly, formula III.5.2.16 specifies the co-variance matrix of parameters of beta density function as follows:

$$Var(a, b) = A \left(\begin{array}{cc} n\psi_1(b) - \psi_1(a+b) & \psi_1(a+b) \\ \psi_1(a+b) & n\psi_1(a) - \psi_1(a+b) \end{array} \right)$$

Formula III.5.2.16. 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.5.2.16 is a concrete case of formula III.5.1.4 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.5.2.8; this is the ultimate purpose of this sub-section III.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.5.2.15, 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.5.2.17. 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.5.2.17 specifying standard errors of parameter estimates ends up this sub-section III.5.2 mentioning applying MLE technique into beta distribution. Now the next sub-section III.5.3 will mention the proposed approach to solve the set of differential equations specified in formula III.5.2.15.

III.5.3. Algorithm to solve the equations whose solutions are parameter estimators

As specified by formula III.5.2.15, the parameter estimators \hat{a} and \hat{b} are solutions of two equations:

$$\begin{cases} \psi(a) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ \psi(b) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(1-f_i) \end{cases}$$

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, a simple form of them is discovered with parameters a and b being 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:

$$\begin{aligned} \psi(a) - \psi(a+b) &= \int_0^{+\infty} \left(\frac{e^{-t}}{t} - \frac{e^{-at}}{1-e^{-t}} \right) dt - \int_0^{+\infty} \left(\frac{e^{-t}}{t} - \frac{e^{-(a+b)t}}{1-e^{-t}} \right) dt \\ &= \int_0^{+\infty} \left(\frac{e^{-(a+b)t}}{1-e^{-t}} - \frac{e^{-at}}{1-e^{-t}} \right) dt = \int_0^{+\infty} \frac{e^{-at}(e^{-bt}-1)}{1-e^{-t}} dt \\ &= \int_0^{+\infty} e^{t(1-a)}(e^{-bt}-1) \frac{e^{-t}}{1-e^{-t}} dt = \int_{-\infty}^0 (1-e^x)^{a-1}((1-e^x)^b - 1) dx \\ &= \int_{-\infty}^0 ((1-e^x)^{a+b-1} - (1-e^x)^{a-1}) dx \end{aligned}$$

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 polynomials $(1-e^x)^{a+b-1}$ and $(1-e^x)^{a-1}$ with note that a and b are positive integer numbers, we also have

$$(1 - e^x)^{a+b-1} = \sum_{k=0}^{a+b-1} (-1)^k C_{a+b-1}^k e^{kx}$$

$$(1 - e^x)^{a-1} = \sum_{k=0}^{a-1} (-1)^k C_{a-1}^k e^{kx}$$

Where C_{a+b-1}^k is the combination taken k of $a + b - 1$ elements and C_{a-1}^k is the combination taken k of $a - 1$ elements,

$$C_{a+b-1}^k = \binom{a+b-1}{k} = \frac{(a+b-1)!}{k!(a+b-1-k)!}$$

$$C_{a-1}^k = \binom{a-1}{k} = \frac{(a-1)!}{k!(a-1-k)!}$$

Hence, we have

$$\psi(a) - \psi(a+b) = \int_{-\infty}^0 \left(\sum_{k=0}^{a+b-1} (-1)^k C_{a+b-1}^k e^{kx} - \sum_{k=0}^{a-1} (-1)^k C_{a-1}^k e^{kx} \right) dx$$

Without loss of generality, suppose $a \geq 2$, we have:

$$\begin{aligned} \psi(a) - \psi(a+b) &= \int_{-\infty}^0 \left(\sum_{k=1}^{a+b-1} (-1)^k C_{a+b-1}^k e^{kx} - \sum_{k=1}^{a-1} (-1)^k C_{a-1}^k e^{kx} \right) dx \\ &= \int_{-\infty}^0 \left(\sum_{k=1}^{a+b-1} (-1)^k C_{a+b-1}^k e^{kx} \right) dx - \int_{-\infty}^0 \left(\sum_{k=1}^{a-1} (-1)^k C_{a-1}^k e^{kx} \right) dx \\ &= \sum_{k=1}^{a+b-1} (-1)^k C_{a+b-1}^k \int_{-\infty}^0 e^{kx} dx - \sum_{k=1}^{a-1} (-1)^k C_{a-1}^k \int_{-\infty}^0 e^{kx} dx \\ &= \sum_{k=1}^{a+b-1} (-1)^k C_{a+b-1}^k \left(\frac{e^{kx}}{k} \Big|_{-\infty}^0 \right) - \sum_{k=1}^{a-1} (-1)^k C_{a-1}^k \left(\frac{e^{kx}}{k} \Big|_{-\infty}^0 \right) \\ &= \sum_{k=1}^{a+b-1} \frac{(-1)^k}{k} C_{a+b-1}^k - \sum_{k=1}^{a-1} \frac{(-1)^k}{k} C_{a-1}^k \end{aligned}$$

In the similar way, given $b \geq 2$ we have:

$$\psi(b) - \psi(a+b) = \sum_{k=1}^{a+b-1} \frac{(-1)^k}{k} C_{a+b-1}^k - \sum_{k=1}^{b-1} \frac{(-1)^k}{k} C_{b-1}^k$$

The equations whose solutions are parameter estimators becomes two following equations:

$$\begin{cases} \sum_{k=1}^{a+b-1} \frac{(-1)^k}{k} C_{a+b-1}^k - \sum_{k=1}^{a-1} \frac{(-1)^k}{k} C_{a-1}^k = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ \sum_{k=1}^{a+b-1} \frac{(-1)^k}{k} C_{a+b-1}^k - \sum_{k=1}^{b-1} \frac{(-1)^k}{k} C_{b-1}^k = \frac{1}{n} \sum_{i=1}^n \ln(1-f_i) \end{cases} \Leftrightarrow \begin{cases} G_1(a, b) = L_1 \\ G_2(a, b) = L_2 \end{cases}$$

Formula III.5.3.1. Two simplest equations for estimating positive integer parameters a and b

Where,

$$a \geq 2, b \geq 2$$

$$G_1(a, b) = \sum_{k=1}^{a+b-1} \frac{(-1)^k}{k} C_{a+b-1}^k - \sum_{k=1}^{a-1} \frac{(-1)^k}{k} C_{a-1}^k$$

$$G_2(a, b) = \sum_{k=1}^{a+b-1} \frac{(-1)^k}{k} C_{a+b-1}^k - \sum_{k=1}^{b-1} \frac{(-1)^k}{k} C_{b-1}^k$$

$$L_1 = \frac{1}{n} \sum_{i=1}^n \ln(f_i)$$

$$L_2 = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i)$$

Obviously, formula III.5.3.1 is the simplest form from which all differential functions are removed. However, the number of solutions of equations in formula III.5.3.1 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 $\Delta_2=G_2(a_i, b_i)-L_2$ are computed. The normal bias is the root of sum of the second power Δ_1 and the second power of Δ_2 and so we have $\Delta=\sqrt{\Delta_1^2 + \Delta_2^2}$. The pair (\hat{a}, \hat{b}) whose normal bias Δ is minimum are chosen as the parameter estimators. The algorithm is described in table III.5.3.1 as below:

$\min\Delta = +\infty$ $\hat{a} = \hat{b} = 1$ (uniform distribution) For $a=1$ to n do For $b=1$ to n do $\Delta_1=G_1(a, b)-L_1$ $\Delta_2=G_2(a, b)-L_2$ $\Delta=\sqrt{\Delta_1^2 + \Delta_2^2}$ If $\Delta < \min\Delta$ then $\min\Delta=\Delta$ $\hat{a}=a$ $\hat{b}=b$ End If End For a End For b (\hat{a} and \hat{b} are optimal parameters)

Table III.5.3.1. Iterative algorithm solving simplest equations specified by formula III.5.3.1 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.5.2 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.5.4 describes an example illustrating proposed algorithm in table III.5.3.1.

III.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.5.4.1 depicts the example of BN including such variables X_1 and X_2 without CPT (s).

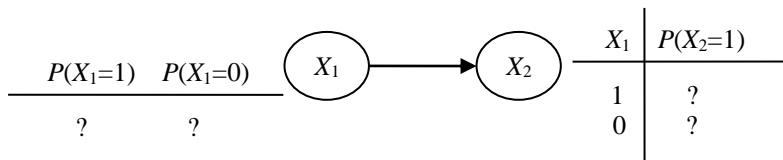


Figure III.5.4.1. Bayesian network without CPT (s)

In figure III.5.4.1, 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.5.2 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 β_1 , β_2 and β_3 , respectively. Suppose we perform 5 trials of a random process. Outcome of the i^{th} trial denoted $D^{(i)}$ is considered as an evidence in which X_1 and X_2 obtain value 0 or 1. So we have one sample $\mathcal{D} = (D^{(1)}, D^{(2)}, D^{(2)}, D^{(3)}, D^{(4)}, D^{(5)})$. Table III.5.4.1 shows these five evidences.

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

Table III.5.4.1. The evidences corresponding to 5 trials

According to the proposed algorithm described in table III.5.3.1, let L_{ij} , G_{ij} , Δ_{ij} , Δ_i be the values of L_j , G_j , Δ_j , Δ with respect to β_i where $i = \overline{1,3}$, $j = \overline{1,2}$, and $n = 1$. Because there is only one sample ($n=1$), we have:

- $L_{11} = \ln(f_i)$ and $L_{12} = \ln(1-f_i)$ where f_i is the frequency of event $X_1=1$.
- $L_{21} = \ln(f_i)$ and $L_{22} = \ln(1-f_i)$ where f_i is the frequency of event $X_2=1$ given $X_1=1$.
- $L_{31} = \ln(f_i)$ and $L_{32} = \ln(1-f_i)$ where f_i is the frequency of event $X_2=1$ given $X_1=0$.
- $G_{11}(a_1, b_1) = \sum_{k=1}^{a_1+b_1-1} \frac{(-1)^k}{k} C_{a_1+b_1-1}^k - \sum_{k=1}^{a_1-1} \frac{(-1)^k}{k} C_{a_1-1}^k$ and $G_{12}(a_1, b_1) = \sum_{k=1}^{a_1+b_1-1} \frac{(-1)^k}{k} C_{a_1+b_1-1}^k - \sum_{k=1}^{b_1-1} \frac{(-1)^k}{k} C_{b_1-1}^k$.
- $G_{21}(a_2, b_2) = \sum_{k=1}^{a_2+b_2-1} \frac{(-1)^k}{k} C_{a_2+b_2-1}^k - \sum_{k=1}^{a_2-1} \frac{(-1)^k}{k} C_{a_2-1}^k$ and $G_{22}(a_2, b_2) = \sum_{k=1}^{a_2+b_2-1} \frac{(-1)^k}{k} C_{a_2+b_2-1}^k - \sum_{k=1}^{b_2-1} \frac{(-1)^k}{k} C_{b_2-1}^k$.
- $G_{31}(a_3, b_3) = \sum_{k=1}^{a_3+b_3-1} \frac{(-1)^k}{k} C_{a_3+b_3-1}^k - \sum_{k=1}^{a_3-1} \frac{(-1)^k}{k} C_{a_3-1}^k$ and $G_{32}(a_3, b_3) = \sum_{k=1}^{a_3+b_3-1} \frac{(-1)^k}{k} C_{a_3+b_3-1}^k - \sum_{k=1}^{b_3-1} \frac{(-1)^k}{k} C_{b_3-1}^k$.
- $\Delta_{11} = G_{11} - L_{11}$, $\Delta_{12} = G_{12} - L_{12}$ 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 $\Delta_3 = \sqrt{\Delta_{31}^2 + \Delta_{32}^2}$

From above evidences shown in table III.5.4.1, it is easy to compute L_{ij} . For example, because there are 3 evidences $D^{(1)}, D^{(2)}, D^{(3)}$ among 5 evidences in which $X_1=1$, we have $L_{11} = \ln(3/5) \approx -0.51$ and $L_{12} = \ln(1-3/5) \approx -0.92$. In the similar way, all L_{ij} are determined. Table III.5.4.2 shows values of L_{ij} corresponding to beta density functions β_1, β_2 , and β_3 .

β_1	β_2	β_3
$L_{11} = -0.51$	$L_{21} = -0.41$	$L_{31} = -0.69$
$L_{12} = -0.92$	$L_{22} = -1.10$	$L_{32} = -0.69$

Table III.5.4.2. The values of L_{ij} corresponding to beta density function β_1, β_2 , and β_3

Suppose the range of all parameters is from 1 to 4. By applying the proposed algorithm described in table III.5.3.1, it is easy to compute the normal biases. For example, given $a_1 = 1$ and $b_1 = 1$, we have:

$$\begin{aligned} G_{11}(a_1 = 2, b_1 = 2) &= \sum_{k=1}^3 \frac{(-1)^k}{k} C_3^k - \sum_{k=1}^1 \frac{(-1)^k}{k} C_1^k \\ &= -C_3^1 + \frac{1}{2} C_3^2 - \frac{1}{3} C_3^3 + C_1^1 = -3 + \frac{3}{2} - \frac{1}{3} + 1 \approx -0.83 \end{aligned}$$

$$G_{12}(a_1 = 2, b_1 = 2) = \sum_{k=1}^3 \frac{(-1)^k}{k} C_3^k - \sum_{k=1}^1 \frac{(-1)^k}{k} C_1^k \approx -0.83$$

$$\Delta_{11} = G_{11} - L_{11} = -0.83 + 0.51 \approx -0.32$$

$$\Delta_{12} = G_{12} - L_{12} = -0.83 + 0.92 \approx 0.08$$

$$\Delta_1 = \sqrt{\Delta_{11}^2 + \Delta_{12}^2} = \sqrt{(-0.32)^2 + (0.08)^2} \approx 0.33$$

Following table III.5.4.3 shows normal biases of all possible values of (a_1, b_1) .

a_1	b_1	L_{11}	L_{12}	G_{11}	G_{12}	Δ_{11}	Δ_{12}	Δ_1
2.0	2.0	-0.51	-0.92	-0.83	-0.83	-0.32	0.08	0.33
2.0	3.0	-0.51	-0.92	-1.08	-0.58	-0.57	0.33	0.66
2.0	4.0	-0.51	-0.92	-1.28	-0.45	-0.77	0.47	0.90

3.0	2.0	-0.51	-0.92	-0.58	-1.08	-0.07	-0.17	0.18
3.0	3.0	-0.51	-0.92	-0.78	-0.78	-0.27	0.13	0.30
3.0	4.0	-0.51	-0.92	-0.95	-0.62	-0.44	0.30	0.53
4.0	2.0	-0.51	-0.92	-0.45	-1.28	0.06	-0.37	0.37
4.0	3.0	-0.51	-0.92	-0.62	-0.95	-0.11	-0.03	0.11
4.0	4.0	-0.51	-0.92	-0.76	-0.76	-0.25	0.16	0.29

Table III.5.4.3. The normal biases of (a_1, b_1) with respect to β_1

The normal biases of all possible values of (a_2, b_2) with respect to β_2 are shown in following table [III.5.4.4](#).

a_2	b_2	L_{21}	L_{22}	G_{21}	G_{22}	Δ_{21}	Δ_{22}	Δ_2
2.0	2.0	-0.41	-1.1	-0.83	-0.83	-0.43	0.27	0.50
2.0	3.0	-0.41	-1.1	-1.08	-0.58	-0.68	0.52	0.85
2.0	4.0	-0.41	-1.1	-1.28	-0.45	-0.88	0.65	1.09
3.0	2.0	-0.41	-1.1	-0.58	-1.08	-0.18	0.02	0.18
3.0	3.0	-0.41	-1.1	-0.78	-0.78	-0.38	0.32	0.49
3.0	4.0	-0.41	-1.1	-0.95	-0.62	-0.54	0.48	0.73
4.0	2.0	-0.41	-1.1	-0.45	-1.28	-0.04	-0.18	0.19
4.0	3.0	-0.41	-1.1	-0.62	-0.95	-0.21	0.15	0.26
4.0	4.0	-0.41	-1.1	-0.76	-0.76	-0.35	0.34	0.49

Table III.5.4.4. The normal biases of (a_2, b_2) with respect to β_2

The normal biases of all possible values of (a_3, b_3) with respect to β_3 are shown in following table [III.5.4.5](#).

a_3	b_3	L_{31}	L_{32}	G_{31}	G_{32}	Δ_{31}	Δ_{32}	Δ_3
2.0	2.0	-0.69	-0.69	-0.83	-0.83	-0.14	-0.14	0.20
2.0	3.0	-0.69	-0.69	-1.08	-0.58	-0.39	0.11	0.41
2.0	4.0	-0.69	-0.69	-1.28	-0.45	-0.59	0.24	0.64
3.0	2.0	-0.69	-0.69	-0.58	-1.08	0.11	-0.39	0.41
3.0	3.0	-0.69	-0.69	-0.78	-0.78	-0.09	-0.09	0.13
3.0	4.0	-0.69	-0.69	-0.95	-0.62	-0.26	0.08	0.27
4.0	2.0	-0.69	-0.69	-0.45	-1.28	0.24	-0.59	0.64
4.0	3.0	-0.69	-0.69	-0.62	-0.95	0.08	-0.26	0.27
4.0	4.0	-0.69	-0.69	-0.76	-0.76	-0.07	-0.07	0.09

Table III.5.4.5. The normal biases of (a_3, b_3) with respect to β_3

From above tables [III.5.4.3](#), [III.5.4.4](#), and [III.5.4.5](#), we recognize that when $(a_1, b_1)=(4,3)$, $(a_2, b_2)=(3,2)$ and $(a_3, b_3)=(4,4)$, the normal biases of distributions β_1 , β_2 and β_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 β_1 , β_2 and β_3 are $(4,3)$, $(3,2)$ and $(4,4)$, 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:

$$P(X_1 = 1) = \frac{\hat{a}_1}{\hat{a}_1 + \hat{b}_1} = \frac{4}{4 + 3} \approx 0.57$$

$$P(X_2 = 1 | X_1 = 1) = \frac{\hat{a}_2}{\hat{a}_2 + \hat{b}_2} = \frac{3}{3 + 2} = 0.60$$

$$P(X_2 = 1 | X_1 = 0) = \frac{\hat{a}_3}{\hat{a}_3 + \hat{b}_3} = \frac{4}{4 + 4} = 0.50$$

When these prior probabilities were calculated, the Bayesian network is totally determined with full of prior CPT (s) as in figure III.5.4.2.

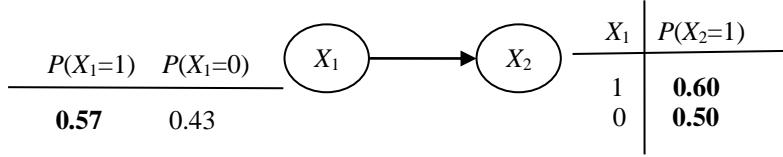


Figure III.5.4.2. Bayesian network with full of prior CPT (s)

The figure III.5.4.2 shows the ultimate result of the interesting algorithm mentioned mainly in sub-section III.5.3, which is used for specifying prior CPT (s) of Bayesian network.

Let \$\sigma(\hat{a}_1)\$, \$\sigma(\hat{b}_1)\$, \$\sigma(\hat{a}_2)\$, \$\sigma(\hat{b}_2)\$, \$\sigma(\hat{a}_3)\$, and \$\sigma(\hat{b}_3)\$ be standard errors of \$\hat{a}_1\$, \$\hat{b}_1\$, \$\hat{a}_2\$, \$\hat{b}_2\$, \$\hat{a}_3\$, and \$\hat{b}_3\$. By applying formula III.5.2.17, it is easy to determine these standard errors as follows:

$$\begin{aligned} A_1 &= \frac{1}{1^2 \psi_1(\hat{a}_1) \psi_1(\hat{b}_1) - 1 \psi_1(\hat{a}_1 + \hat{b}_1) (\psi_1(\hat{a}_1) + \psi_1(\hat{b}_1))} = 127.0410 \\ \sigma(\hat{a}_1) &= \sqrt{A_1 (1 \psi_1(\hat{b}_1) - \psi_1(\hat{a}_1 + \hat{b}_1))} = 5.5377 \\ \sigma(\hat{b}_1) &= \sqrt{A_1 (1 \psi_1(\hat{a}_1) - \psi_1(\hat{a}_1 + \hat{b}_1))} = 4.0682 \\ A_2 &= \frac{1}{1^2 \psi_1(\hat{a}_2) \psi_1(\hat{b}_2) - 1 \psi_1(\hat{a}_2 + \hat{b}_2) (\psi_1(\hat{a}_2) + \psi_1(\hat{b}_2))} = 40.7170 \\ \sigma(\hat{a}_2) &= \sqrt{A_2 (1 \psi_1(\hat{b}_2) - \psi_1(\hat{a}_2 + \hat{b}_2))} = 4.1531 \\ \sigma(\hat{b}_2) &= \sqrt{A_2 (1 \psi_1(\hat{a}_2) - \psi_1(\hat{a}_2 + \hat{b}_2))} = 2.6588 \\ A_3 &= \frac{1}{1^2 \psi_1(\hat{a}_3) \psi_1(\hat{b}_3) - 1 \psi_1(\hat{a}_3 + \hat{b}_3) (\psi_1(\hat{a}_3) + \psi_1(\hat{b}_3))} = 200.7710 \\ \sigma(\hat{a}_3) &= \sqrt{A_3 (1 \psi_1(\hat{b}_3) - \psi_1(\hat{a}_3 + \hat{b}_3))} = 5.5003 \\ \sigma(\hat{b}_3) &= \sqrt{A_3 (1 \psi_1(\hat{a}_3) - \psi_1(\hat{a}_3 + \hat{b}_3))} = 5.5003 \end{aligned}$$

The errors \$\sigma(\hat{a}_2)\$ and \$\sigma(\hat{b}_2)\$ are minimum, which implies that \$\hat{a}_2\$ and \$\hat{b}_2\$ are best estimates.

In general, the iterative algorithm takes advantages of simple equations whose solutions are estimates of positive parameter \$a\$ and \$b\$. The successive sub-section III.5.5 gives an enjoyable subject that is to recommend a new version of these equations.

III.5.5. New version of the equations whose solutions are parameter estimators

According to formula III.5.3.1 aforementioned in sub-section III.5.3, the parameter estimators \hat{a} and \hat{b} are solutions of two equations as follows:

$$\begin{cases} \sum_{k=1}^{a+b-1} \frac{(-1)^k}{k} C_{a+b-1}^k - \sum_{k=1}^{a-1} \frac{(-1)^k}{k} C_{a-1}^k = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ \sum_{k=1}^{a+b-1} \frac{(-1)^k}{k} C_{a+b-1}^k - \sum_{k=1}^{b-1} \frac{(-1)^k}{k} C_{b-1}^k = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i) \end{cases}$$

Although these equations are very simple, a question is issued “Are there another version of these equations?”. This sub-section III.5.5 proves that the new version of these equations exists.

As specified by formula III.5.2.15, the parameter estimators \hat{a} and \hat{b} are solutions of two equations:

$$\begin{cases} \psi(a) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ \psi(b) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i) \end{cases}$$

According to formula III.5.2.5, 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:

$$\begin{aligned} & \begin{cases} \psi(a) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ \psi(b) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i) \end{cases} \Leftrightarrow \begin{cases} \psi(a) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ \psi(b) - \psi(a+b) = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i) \end{cases} \\ & \Leftrightarrow \begin{cases} -\gamma + \sum_{k=1}^{a-1} \frac{1}{k} + \gamma - \sum_{k=1}^{a+b-1} \frac{1}{k} = \sum_{i=1}^n \ln(f_i) \\ -\gamma + \sum_{k=1}^{b-1} \frac{1}{k} + \gamma - \sum_{k=1}^{a+b-1} \frac{1}{k} = \sum_{i=1}^n \ln(1 - f_i) \end{cases} \\ & \Leftrightarrow \begin{cases} - \sum_{k=a}^{a+b-1} \frac{1}{k} = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ - \sum_{k=b}^{a+b-1} \frac{1}{k} = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i) \end{cases} \end{aligned}$$

Briefly, the parameter estimators \hat{a} and \hat{b} are solutions of two following equations specified by formula III.5.5.1.

$$\begin{cases} - \sum_{k=a}^{a+b-1} \frac{1}{k} = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \\ - \sum_{k=b}^{a+b-1} \frac{1}{k} = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i) \end{cases} \Leftrightarrow \begin{cases} G_1(a, b) = L_1 \\ G_2(a, b) = L_2 \end{cases}$$

Formula III.5.5.1. New version of equations for estimating positive integer parameters a and b

Where,

$$G_1(a, b) = - \sum_{k=a}^{a+b-1} \frac{1}{k} \text{ and } G_2(a, b) = - \sum_{k=b}^{a+b-1} \frac{1}{k}$$

$$L_1 = \frac{1}{n} \sum_{i=1}^n \ln(f_i) \text{ and } L_2 = \frac{1}{n} \sum_{i=1}^n \ln(1 - f_i)$$

Formula III.5.5.1 is slightly simpler than formula III.5.3.1. The iterative algorithm described in table III.5.3.1 is applied into formula III.5.5.1 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.5.5.1. Going back to previous example shown in figure III.5.4.1, 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.5.5.1 is the replication of figure III.5.4.1.

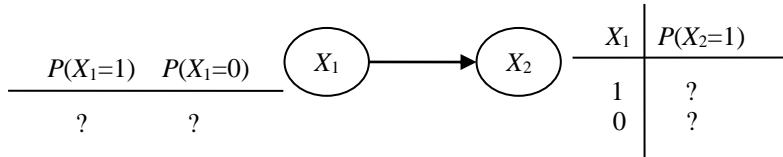


Figure III.5.5.1. Bayesian network without CPT (s)

In figure III.5.5.1, 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.5.2 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 β_1 , β_2 and β_3 , respectively. Suppose we perform 5 trials of a random process. Outcome of the i^{th} trial denoted $D^{(i)}$ is considered as an evidence in which X_1 and X_2 obtain value 0 or 1. So we have one sample $\mathcal{D} = (D^{(1)}, D^{(2)}, D^{(2)}, D^{(3)}, D^{(4)}, D^{(5)})$. For convenience, table III.5.5.1 is replication of table III.5.4.1 showing these five evidences.

	X_1	X_2
--	-------	-------

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

Table III.5.5.1. The evidences corresponding to 5 trials

According to the proposed algorithm described in table III.5.3.1, let L_{ij} , G_{ij} , Δ_{ij} , Δ_i be the values of L_j , G_j , Δ_j , Δ with respect to β_i where $i = \overline{1,3}$, $j = \overline{1,2}$, and $n = 1$. Because there is only one sample ($n = 1$), we have:

- $L_{11} = \ln(f_i)$ and $L_{12} = \ln(1-f_i)$ where f_i is frequency of event $X_1=1$.
- $L_{21} = \ln(f_i)$ and $L_{22} = \ln(1-f_i)$ where f_i is frequency of event $X_2=1$ given $X_1=1$.
- $L_{31} = \ln(f_i)$ and $L_{32} = \ln(1-f_i)$ where f_i is frequency of event $X_2=1$ given $X_1=0$.
- $G_{11}(a_1, b_1) = -\sum_{k=a_1}^{a_1+b_1-1} \frac{1}{k}$ and $G_{12}(a_1, b_1) = -\sum_{k=b_1}^{a_1+b_1-1} \frac{1}{k}$
- $G_{21}(a_2, b_2) = -\sum_{k=a_2}^{a_2+b_2-1} \frac{1}{k}$ and $G_{22}(a_2, b_2) = -\sum_{k=b_2}^{a_2+b_2-1} \frac{1}{k}$
- $G_{31}(a_3, b_3) = -\sum_{k=a_3}^{a_3+b_3-1} \frac{1}{k}$ and $G_{32}(a_3, b_3) = -\sum_{k=b_3}^{a_3+b_3-1} \frac{1}{k}$
- $\Delta_{11} = G_{11} - L_{11}$, $\Delta_{12} = G_{12} - L_{12}$ 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 $\Delta_3 = \sqrt{\Delta_{31}^2 + \Delta_{32}^2}$

From above evidences shown in table III.5.5.1, it is easy to compute L_{ij} . For example, because there are 3 evidences $D^{(1)}, D^{(2)}, D^{(3)}$ among 5 evidences in which $X_1=1$, we have $L_{11} = \ln(3/5) \approx -0.51$ and $L_{12} = \ln(1-3/5) \approx -0.92$. In the similar way, all L_{ij} are determined.

Suppose the range of all parameters is from 1 to 4. By applying the proposed algorithm described in table III.5.3.1, it is easy to compute the normal biases. For example, given $a_1 = 1$ and $b_1 = 1$, we have:

$$L_{11}(a_1 = 1, b_1 = 1) = \ln(3/5) \approx -0.51$$

$$L_{12}(a_1 = 1, b_1 = 1) = \ln(1 - 3/5) \approx -0.92$$

$$G_{11}(a_1 = 1, b_1 = 1) = -\sum_{k=1}^{1+1-1} \frac{1}{k} = -1$$

$$G_{12}(a_1 = 1, b_1 = 1) = -\sum_{k=1}^{1+1-1} \frac{1}{k} = -1$$

$$\Delta_{11} = G_{11} - L_{11} = -1 + 0.51 = -0.49$$

$$\Delta_{12} = G_{12} - L_{12} = -1 + 0.92 = -0.08$$

$$\Delta_1 = \sqrt{\Delta_{11}^2 + \Delta_{12}^2} = \sqrt{(-0.49)^2 + (-0.08)^2} \approx 0.5$$

Following table III.5.5.2 shows normal biases of all possible values of (a_1, b_1) .

a_1	b_1	L_{11}	L_{12}	G_{11}	G_{12}	Δ_{11}	Δ_{12}	Δ_1
1	1	-0.51	-0.92	-1.00	-1.00	-0.49	-0.08	0.50
1	2	-0.51	-0.92	-1.50	-0.50	-0.99	0.42	1.07
1	3	-0.51	-0.92	-1.83	-0.33	-1.32	0.58	1.45
1	4	-0.51	-0.92	-2.08	-0.25	-1.57	0.67	1.71
2	1	-0.51	-0.92	-0.50	-1.50	0.01	-0.58	0.58
2	2	-0.51	-0.92	-0.83	-0.83	-0.32	0.08	0.33

2	3	-0.51	-0.92	-1.08	-0.58	-0.57	0.33	0.66
2	4	-0.51	-0.92	-1.28	-0.45	-0.77	0.47	0.90
3	1	-0.51	-0.92	-0.33	-1.83	0.18	-0.92	0.93
3	2	-0.51	-0.92	-0.58	-1.08	-0.07	-0.17	0.18
3	3	-0.51	-0.92	-0.78	-0.78	-0.27	0.13	0.30
3	4	-0.51	-0.92	-0.95	-0.62	-0.44	0.30	0.53
4	1	-0.51	-0.92	-0.25	-2.08	0.26	-1.17	1.20
4	2	-0.51	-0.92	-0.45	-1.28	0.06	-0.37	0.37
4	3	-0.51	-0.92	-0.62	-0.95	-0.11	-0.03	0.11
4	4	-0.51	-0.92	-0.76	-0.76	-0.25	0.16	0.29

Table III.5.5.2. The normal biases of (a_1, b_1) with respect to β_1

The normal biases of all possible values of (a_2, b_2) with respect to β_2 are shown in following table [III.5.5.3](#).

a_2	b_2	L_{21}	L_{22}	G_{21}	G_{22}	Δ_{21}	Δ_{22}	Δ_2
1	1	-0.41	-1.1	-1.00	-1.00	-0.59	0.10	0.60
1	2	-0.41	-1.1	-1.50	-0.50	-1.09	0.60	1.25
1	3	-0.41	-1.1	-1.83	-0.33	-1.43	0.77	1.62
1	4	-0.41	-1.1	-2.08	-0.25	-1.68	0.85	1.88
2	1	-0.41	-1.1	-0.50	-1.50	-0.09	-0.40	0.41
2	2	-0.41	-1.1	-0.83	-0.83	-0.43	0.27	0.50
2	3	-0.41	-1.1	-1.08	-0.58	-0.68	0.52	0.85
2	4	-0.41	-1.1	-1.28	-0.45	-0.88	0.65	1.09
3	1	-0.41	-1.1	-0.33	-1.83	0.07	-0.73	0.74
3	2	-0.41	-1.1	-0.58	-1.08	-0.18	0.02	0.18
3	3	-0.41	-1.1	-0.78	-0.78	-0.38	0.32	0.49
3	4	-0.41	-1.1	-0.95	-0.62	-0.54	0.48	0.73
4	1	-0.41	-1.1	-0.25	-2.08	0.16	-0.98	1.00
4	2	-0.41	-1.1	-0.45	-1.28	-0.04	-0.18	0.19
4	3	-0.41	-1.1	-0.62	-0.95	-0.21	0.15	0.26
4	4	-0.41	-1.1	-0.76	-0.76	-0.35	0.34	0.49

Table III.5.5.3. The normal biases of (a_2, b_2) with respect to β_2

The normal biases of all possible values of (a_3, b_3) with respect to β_3 are shown in following table [III.5.5.4](#).

a_3	b_3	L_{31}	L_{32}	G_{31}	G_{32}	Δ_{31}	Δ_{32}	Δ_3
1	1	-0.69	-0.69	-1.00	-1.00	-0.31	-0.31	0.43
1	2	-0.69	-0.69	-1.50	-0.50	-0.81	0.19	0.83
1	3	-0.69	-0.69	-1.83	-0.33	-1.14	0.36	1.20
1	4	-0.69	-0.69	-2.08	-0.25	-1.39	0.44	1.46
2	1	-0.69	-0.69	-0.50	-1.50	0.19	-0.81	0.83
2	2	-0.69	-0.69	-0.83	-0.83	-0.14	-0.14	0.20
2	3	-0.69	-0.69	-1.08	-0.58	-0.39	0.11	0.41
2	4	-0.69	-0.69	-1.28	-0.45	-0.59	0.24	0.64
3	1	-0.69	-0.69	-0.33	-1.83	0.36	-1.14	1.20
3	2	-0.69	-0.69	-0.58	-1.08	0.11	-0.39	0.41

3	3	-0.69	-0.69	-0.78	-0.78	-0.09	-0.09	0.13
3	4	-0.69	-0.69	-0.95	-0.62	-0.26	0.08	0.27
4	1	-0.69	-0.69	-0.25	-2.08	0.44	-1.39	1.46
4	2	-0.69	-0.69	-0.45	-1.28	0.24	-0.59	0.64
4	3	-0.69	-0.69	-0.62	-0.95	0.08	-0.26	0.27
4	4	-0.69	-0.69	-0.76	-0.76	-0.07	-0.07	0.09

Table III.5.5.4. The normal biases of (a_3, b_3) with respect to β_3

From above tables III.5.5.2, III.5.5.3, and III.5.5.4, we recognize that when $(a_1, b_1) = (4, 3)$, $(a_2, b_2) = (3, 2)$ and $(a_3, b_3) = (4, 4)$, the normal biases of distributions β_1 , β_2 and β_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 β_1 , β_2 and β_3 are $(4, 3)$, $(3, 2)$ and $(4, 4)$, 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:

$$P(X_1 = 1) = \frac{\hat{a}_1}{\hat{a}_1 + \hat{b}_1} = \frac{4}{4+3} \approx 0.57$$

$$P(X_2 = 1 | X_1 = 1) = \frac{\hat{a}_2}{\hat{a}_2 + \hat{b}_2} = \frac{3}{3+2} = 0.60$$

$$P(X_2 = 1 | X_1 = 0) = \frac{\hat{a}_3}{\hat{a}_3 + \hat{b}_3} = \frac{4}{4+4} = 0.50$$

When these prior probabilities were calculated, the Bayesian network is totally determined with full of prior CPT (s) as in figure III.5.5.2.

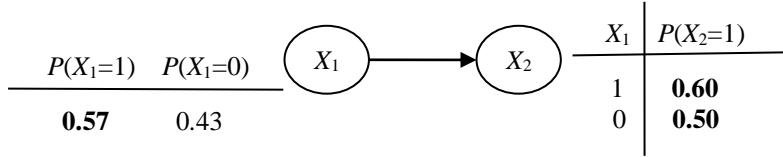


Figure III.5.5.2. Bayesian network with full of prior CPT (s)

The figure III.5.5.2 shows the ultimate result of applying the iterative algorithm mentioned in sub-section III.5.3 into the simple equations specified by formula III.5.5.1 in this sub-section III.5.5. It is easy to recognize that this result is the same to the one (shown in figure III.5.4.2) that is produced from equations specified by formula III.5.3.1.

Let $\sigma(\hat{a}_1)$, $\sigma(\hat{b}_1)$, $\sigma(\hat{a}_2)$, $\sigma(\hat{b}_2)$, $\sigma(\hat{a}_3)$, and $\sigma(\hat{b}_3)$ be standard errors of \hat{a}_1 , \hat{b}_1 , \hat{a}_2 , \hat{b}_2 , \hat{a}_3 , and \hat{b}_3 . By applying formula III.5.2.17, it is easy to determine these standard errors as follows:

$$A_1 = \frac{1}{1^2 \psi_1(\hat{a}_1) \psi_1(\hat{b}_1) - 1 \psi_1(\hat{a}_1 + \hat{b}_1) (\psi_1(\hat{a}_1) + \psi_1(\hat{b}_1))} = 127.0410$$

$$\sigma(\hat{a}_1) = \sqrt{A_1 (1 \psi_1(\hat{b}_1) - \psi_1(\hat{a}_1 + \hat{b}_1))} = 5.5377$$

$$\sigma(\hat{b}_1) = \sqrt{A_1 (1 \psi_1(\hat{a}_1) - \psi_1(\hat{a}_1 + \hat{b}_1))} = 4.0682$$

$$A_2 = \frac{1}{1^2 \psi_1(\hat{a}_2) \psi_1(\hat{b}_2) - 1 \psi_1(\hat{a}_2 + \hat{b}_2) (\psi_1(\hat{a}_2) + \psi_1(\hat{b}_2))} = 40.7170$$

$$\sigma(\hat{a}_2) = \sqrt{A_2 \left(1\psi_1(\hat{b}_2) - \psi_1(\hat{a}_2 + \hat{b}_2) \right)} = 4.1531$$

$$\sigma(\hat{b}_2) = \sqrt{A_2 \left(1\psi_1(\hat{a}_2) - \psi_1(\hat{a}_2 + \hat{b}_2) \right)} = 2.6588$$

$$A_3 = \frac{1}{1^2\psi_1(\hat{a}_3)\psi_1(\hat{b}_3) - 1\psi_1(\hat{a}_3 + \hat{b}_3)(\psi_1(\hat{a}_3) + \psi_1(\hat{b}_3))} = 200.7710$$

$$\sigma(\hat{a}_3) = \sqrt{A_3 \left(1\psi_1(\hat{b}_3) - \psi_1(\hat{a}_3 + \hat{b}_3) \right)} = 5.5003$$

$$\sigma(\hat{b}_3) = \sqrt{A_3 \left(1\psi_1(\hat{a}_3) - \psi_1(\hat{a}_3 + \hat{b}_3) \right)} = 5.5003$$

The errors $\sigma(\hat{a}_2)$ and $\sigma(\hat{b}_2)$ are minimum, which implies that \hat{a}_2 and \hat{b}_2 are best estimates.

In general, the iterative algorithm for solving simple equations specified by formulas III.5.3.1 and III.5.5.1 is the result of applying MLE method into beta density function. Sub-section III.5.6 is evaluation of the iterative algorithm which is considered as an implementation of MLE method for specifying prior probabilities of BN.

III.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. MLE is applied into beta distribution so as to determine the equations for computing two parameters of beta distribution. Due to the complexity involved with the beta distribution in estimating its parameters when compared to other distributions such as the binomial distribution and normal distribution, a simple form of MLE equations was formulated and utilized in the case that parameters are positive integer numbers. Moreover, an algorithm that calculates the approximate solutions of these equations was derived. 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 pinpoint the precise solutions but it is easy and feasible to implement the proposed 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 can focus on finding out discrete parameters instead of making effort to solve differential equations. The proposed iterative algorithm is only appropriate to such ideology. Additionally, the new version of these simple equations is also proposed, which gives convenience for solving such equations. In other words, the new version digs deeply into mathematical functions relevant to gamma function, digamma function, and trigamma function.

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 section III.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 chapter [IV](#) 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](#)) and hidden Markov model (see section [IV.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 chapter [IV](#).

Chapter IV. Learning style sub-model

The 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 which are mentioned in successive chapters III, IV, and V, respectively. Please pay attention to the coherence of such three main chapters together with respective sub-models. Now this chapter IV focuses on the 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 sub-model described in previous chapter III 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 chapter IV, 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 section IV.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 mentioned 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 section IV.1 as an introduction of some basic concepts of learning styles.

IV.1. What learning styles are

People have different views upon the same situation, the way they perceive and estimate the world is different (Stash, 2007, p. 91). 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 section IV.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 some drawbacks of question-and-answer technique, i.e., not questions enough, confusing questions, and users’ wrong answers. Hence, such technique is not an optimal 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 section IV.5. Section IV.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 section IV.3. Additionally, it is necessary to read section IV.4 which is the brief survey of hidden Markov model (HMM).

IV.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.

IV.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*: focuses 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'.

IV.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.
- *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 IV.2.2.1 (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.

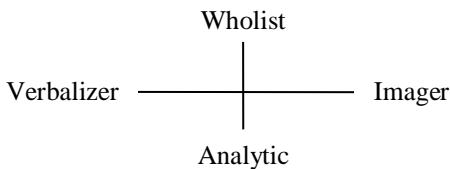


Figure IV.2.2.1. Two dimensions in Riding model

IV.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.

IV.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, 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 IV.2.4.1 (Stash, 2007, p. 103) depicts this four-stage cycle in learning process.

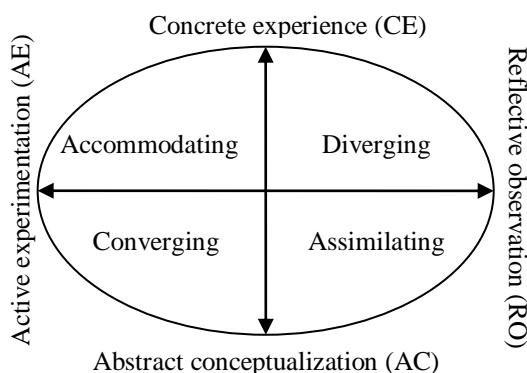


Figure IV.2.4.1. Four-stage learning process of Kolb's model

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 IV.2.4.2 depicts learning styles in Honey and Mumford model.

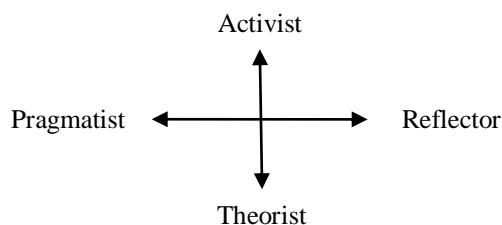


Figure IV.2.4.2. Learning styles in Honey and Mumford model

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 students 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 section [IV.3](#).

IV.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 (Stash, 2007, pp. 117-118). Readers are recommended to read the PhD thesis “*Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*” by the author [Natalia Stash](#) in order to understand comprehensively learning styles and learning strategies; this is an excellent research to which I referred and so, I express my deep gratitude to the author Stash (Stash, 2007). Although there are many learning style models but they share some common features, for example, 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 (Stash, Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System, 2007, pp. 118-123):

Perception group: This group related learners’ perception includes (Stash, Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System, 2007, pp. 122-123):

- 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 (Stash, Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System, 2007, pp. 118-122):

- 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. Figures I.2.3.1 and I.2.3.2 list adaptation techniques.
- *Ordering information or providing different navigation paths:* The order in which learning materials is suggested to users is tuned with learning styles. Following is a sample of ordering information. For active learners, learning materials are presented in the order "activity→example→theory→exercise". For reflective learner, this order is changed into "example→theory→exercise→activity". This form is corresponding to link adaptation techniques: direct guidance, link sorting, link hiding, link annotation. Figures I.2.3.1 and I.2.3.2 list adaptation techniques.
- *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 and 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 section IV.5, it is necessary to glance over hidden Markov model (HMM) in successive section IV.4. Although the purpose of description of HMM is to introduce how to apply HMM into representing learning style model, the next section IV.4 can be read as separated report for HMM (Nguyen L. , Tutorial on Hidden Markov Model, 2016).

IV.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). Suppose you are in the room and do not know the weather outside but you are notified observations such as wind speed, atmospheric pressure, humidity, and temperature from someone else. Basing on these observations, it is possible for you 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 Π 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 π_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 time points. This stochastic process is defined as a finite vector $X=(x_1, x_2, \dots, x_T)$ whose element x_t is a state at time point t . The process X is called *state stochastic process* and $x_t \in S$ equals some state $s_i \in S$. Note that X is also called *state sequence*. Time point can be in terms of second, minute, hour, day, month, year, etc. It is easy to infer that the initial probability $\pi_i = P(x_1=s_i)$ where x_1 is the first state of the stochastic process. The state stochastic process X must meet fully the *Markov property*, namely, given previous state x_{t-1} of process X , the conditional probability of current state x_t is only dependent on the previous state x_{t-1} , not relevant to any further past state ($x_{t-2}, x_{t-3}, \dots, x_1$). In other words, $P(x_t / x_{t-1}, x_{t-2}, x_{t-3}, \dots, x_1) = P(x_t / x_{t-1})$ 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 sub-section III.4.2.
- At each time point, the process changes to the next state based on the *transition probability distribution* a_{ij} , which depends only on the previous state. So a_{ij} is the probability that the stochastic process changes current state s_i to next state s_j . It means that $a_{ij} = P(x_t=s_j | x_{t-1}=s_i) = P(x_{t+1}=s_j | x_t=s_i)$. 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 . Note that A is n by n matrix because there are n distinct states. It is easy to infer that matrix A represents state stochastic process X . It is possible to understand that the initial probability matrix Π is degradation case of matrix A .

Briefly, MM is the triple $\langle S, A, \Pi \rangle$. In typical MM, states are observed directly by users and transition probabilities (A and Π) are 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):

- Suppose there is a finite set of possible observations $\Phi = \{\varphi_1, \varphi_2, \dots, \varphi_m\}$ whose cardinality is m . There is the second stochastic process which produces *observations* correlating with hidden states. This process is called *observable stochastic process*, which is defined as a finite vector $O = (o_1, o_2, \dots, o_T)$ whose element o_t is an observation at time point t . Note that $o_t \in \Phi$ equals some φ_k . The process O is often known as *observation sequence*.
- There is a probability distribution of producing a given observation in each state. Let $b_i(k)$ be the probability of observation φ_k when the state stochastic process is in state s_i . It means that $b_i(k) = b_i(o_t=\varphi_k) = P(o_t=\varphi_k | x_t=s_i)$. The sum of probabilities of all observations which observed in a certain state is 1, we have $\forall s_i \in S, \sum_{\varphi_k \in \Phi} b_i(k) = 1$. All probabilities of observations $b_i(k)$

constitute the *observation probability matrix* B . It is convenient for us to use notation b_{ik} instead of notation $b_i(k)$. Note that B is n by m matrix because there are n distinct states and m distinct observations. While matrix A represents state stochastic process X , matrix B represents observable stochastic process O .

Thus, HMM is the 5-tuple $\Delta = \langle S, \Phi, A, B, \Pi \rangle$. Note that components S, Φ, A, B , and Π are often called parameters of HMM in which A, B , and Π are essential parameters. Going back weather example, suppose you need to predict how weather tomorrow is: *sunny*, *cloudy* or *rainy* since you know only observations about the humidity: *dry*, *dryish*, *damp*, *soggy*. The HMM is totally determined based on its parameters S, Φ, A, B , and Π according to weather example. We have $S = \{s_1=\text{sunny}, s_2=\text{cloudy}, s_3=\text{rainy}\}$, $\Phi = \{\varphi_1=\text{dry}, \varphi_2=\text{dryish}, \varphi_3=\text{damp}, \varphi_4=\text{soggy}\}$. Transition probability matrix A is shown in table IV.4.1.

		Weather current day (Time point t)		
		<i>sunny</i>	<i>cloudy</i>	<i>rainy</i>
Weather previous day (Time point $t-1$)	<i>sunny</i>	$a_{11}=0.50$	$a_{12}=0.25$	$a_{13}=0.25$
	<i>cloudy</i>	$a_{21}=0.30$	$a_{22}=0.40$	$a_{23}=0.30$
	<i>rainy</i>	$a_{31}=0.25$	$a_{32}=0.25$	$a_{33}=0.50$

Table IV.4.1. Transition probability matrix A

From table IV.4.1, we have $a_{11}+a_{12}+a_{13}=1$, $a_{21}+a_{22}+a_{23}=1$, $a_{31}+a_{32}+a_{33}=1$. Initial state distribution specified as uniform distribution is shown in table IV.4.2.

	<i>sunny</i>	<i>cloudy</i>	<i>rainy</i>
	$\pi_1=0.33$	$\pi_2=0.33$	$\pi_3=0.33$

Table IV.4.2. Uniform initial state distribution Π

From table IV.4.2, we have $\pi_1+\pi_2+\pi_3=1$. Observation probability matrix B is shown in table IV.4.3.

		Humidity			
		<i>dry</i>	<i>dryish</i>	<i>damp</i>	<i>soggy</i>
Weather	<i>sunny</i>	$b_{11}=0.60$	$b_{12}=0.20$	$b_{13}=0.15$	$b_{14}=0.05$
	<i>cloudy</i>	$b_{21}=0.25$	$b_{22}=0.25$	$b_{23}=0.25$	$b_{24}=0.25$
	<i>rainy</i>	$b_{31}=0.05$	$b_{32}=0.10$	$b_{33}=0.35$	$b_{34}=0.50$

Table IV.4.3. Observation probability matrix B

From table IV.4.3, we have $b_{11}+b_{12}+b_{13}+b_{14}=1$, $b_{21}+b_{22}+b_{23}+b_{24}=1$, $b_{31}+b_{32}+b_{33}+b_{34}=1$.

The whole weather HMM is depicted in figure IV.4.1.

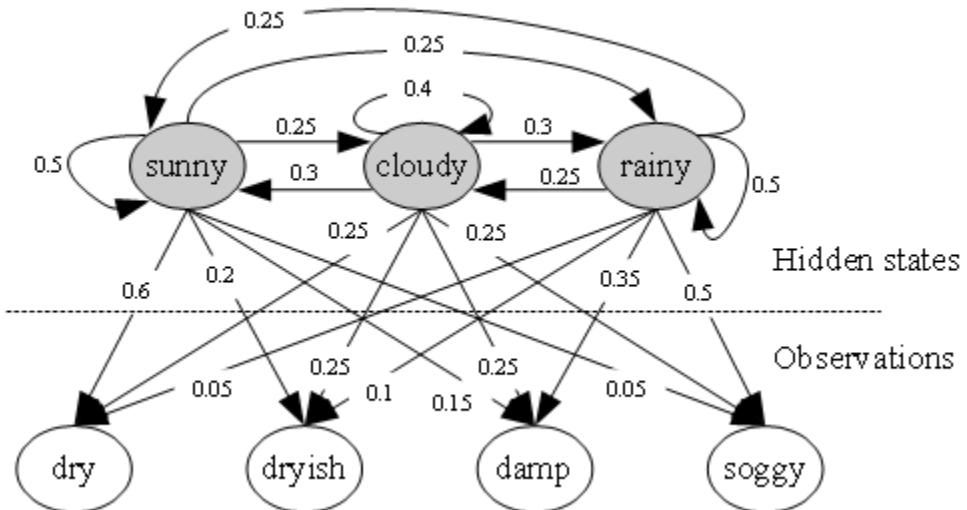


Figure IV.4.1. HMM of weather forecast (hidden states are shaded)

There are three problems of HMM (Schmolze, 2001) (Rabiner, 1989, pp. 262-266):

- Given HMM Δ and an observation sequence $O = \{o_1, o_2, \dots, o_T\}$ where $o_t \in \Phi$, how to calculate the probability $P(O|\Delta)$ of this observation sequence. Such probability $P(O|\Delta)$ indicates how much the HMM Δ affects on sequence O . This is *evaluation problem* or *explanation problem*. Note that it is possible to denote $O = \{o_1 \rightarrow o_2 \rightarrow \dots \rightarrow o_T\}$ and the sequence O is aforementioned observable stochastic process.
- Given HMM Δ and an observation sequence $O = \{o_1, o_2, \dots, o_T\}$ where $o_t \in \Phi$, how to find the sequence of states $X = \{x_1, x_2, \dots, x_T\}$ where $x_t \in S$ so that X is most likely to have produced the observation sequence O . This is *uncovering problem*. Note that the sequence X is aforementioned state stochastic process.
- Given HMM Δ and an observation sequence $O = \{o_1, o_2, \dots, o_T\}$ where $o_t \in \Phi$, how to adjust parameters of Δ such as initial state distribution \prod , transition probability matrix A , and observation probability matrix B so that the quality of HMM Δ is enhanced. This is *learning problem*.

These problems will be mentioned in sub-sections IV.4.1, IV.4.2, and IV.4.3, in turn.

IV.4.1. HMM evaluation problem

The essence of evaluation problem is to find out the way to compute the probability $P(O|\Delta)$ most effectively given the observation sequence $O = \{o_1, o_2, \dots, o_T\}$. For example, given HMM Δ whose parameters A , B , and \prod specified in tables IV.4.1, IV.4.2, and IV.4.3, which is designed for weather forecast. Suppose we need to calculate the probability of event that humidity is *soggy*, *dry*, and *dryish* in days 1, 2, and 3, respectively. This is evaluation problem with sequence of observations $O = \{o_1=\varphi_4=\text{soggy}, o_2=\varphi_1=\text{dry}, o_3=\varphi_2=\text{dryish}\}$. There is a complete set of $3^3=27$ mutually exclusive cases of weather states for three days as follows:

- Weather states in days 1, 2, and 3 are *sunny*, *sunny*, and *sunny*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_1=\text{sunny}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *sunny*, *sunny*, and *cloudy*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_1=\text{sunny}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *sunny*, *sunny*, and *rainy*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_1=\text{sunny}, x_3=s_3=\text{rainy}\}$.

- Weather states in days 1, 2, and 3 are *sunny*, *cloudy*, and *sunny*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_2=\text{cloudy}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *sunny*, *cloudy*, and *cloudy*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_2=\text{cloudy}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *sunny*, *cloudy*, and *rainy*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_2=\text{cloudy}, x_3=s_3=\text{rainy}\}$.
- Weather states in days 1, 2, and 3 are *sunny*, *rainy*, and *sunny*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_3=\text{rainy}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *sunny*, *rainy*, and *cloudy*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_3=\text{rainy}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *sunny*, *rainy*, and *rainy*. State stochastic process is $X = \{x_1=s_1=\text{sunny}, x_2=s_3=\text{rainy}, x_3=s_3=\text{rainy}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *sunny*, and *sunny*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_1=\text{sunny}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *sunny*, and *cloudy*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_1=\text{sunny}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *sunny*, and *rainy*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_1=\text{sunny}, x_3=s_3=\text{rainy}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *cloudy*, and *sunny*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_2=\text{cloudy}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *cloudy*, and *cloudy*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_2=\text{cloudy}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *cloudy*, and *rainy*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_2=\text{cloudy}, x_3=s_3=\text{rainy}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *rainy*, and *sunny*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_3=\text{rainy}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *rainy*, and *cloudy*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_3=\text{rainy}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *cloudy*, *rainy*, and *rainy*. State stochastic process is $X = \{x_1=s_2=\text{cloudy}, x_2=s_3=\text{rainy}, x_3=s_3=\text{rainy}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *sunny*, and *sunny*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_1=\text{sunny}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *sunny*, and *cloudy*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_1=\text{sunny}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *sunny*, and *rainy*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_1=\text{sunny}, x_3=s_3=\text{rainy}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *cloudy*, and *sunny*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_2=\text{cloudy}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *cloudy*, and *cloudy*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_2=\text{cloudy}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *cloudy*, and *rainy*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_2=\text{cloudy}, x_3=s_3=\text{rainy}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *rainy*, and *sunny*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_3=\text{rainy}, x_3=s_1=\text{sunny}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *rainy*, and *cloudy*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_3=\text{rainy}, x_3=s_2=\text{cloudy}\}$.
- Weather states in days 1, 2, and 3 are *rainy*, *rainy*, and *rainy*. State stochastic process is $X = \{x_1=s_3=\text{rainy}, x_2=s_3=\text{rainy}, x_3=s_3=\text{rainy}\}$.

According to total probability rule specified by formula III.1.1.4, the probability $P(O|\Delta)$ is:

$$\begin{aligned}
 & + P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_3, x_3 = s_1) \\
 & \quad * P(x_1 = s_3, x_2 = s_3, x_3 = s_1) \\
 & + P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_3, x_3 = s_2) \\
 & \quad * P(x_1 = s_3, x_2 = s_3, x_3 = s_2) \\
 & + P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_3, x_3 = s_3) \\
 & \quad * P(x_1 = s_3, x_2 = s_3, x_3 = s_3)
 \end{aligned}$$

We have:

$$\begin{aligned}
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_1, x_3 = s_1) \\
 & \quad * P(x_1 = s_1, x_2 = s_1, x_3 = s_1) \\
 & = P(o_1 = \varphi_4 | x_1 = s_1, x_2 = s_1, x_3 = s_1) * P(o_2 = \varphi_1 | x_1 = s_1, x_2 = s_1, x_3 = s_1) \\
 & \quad * P(o_3 = \varphi_2 | x_1 = s_1, x_2 = s_1, x_3 = s_1) * P(x_1 = s_1, x_2 = s_1, x_3 = s_1) \\
 & \quad (\text{Because observations } o_1, o_2, \text{ and } o_3 \text{ are mutually independent}) \\
 & = P(o_1 = \varphi_4 | x_1 = s_1) * P(o_2 = \varphi_1 | x_2 = s_1) * P(o_3 = \varphi_2 | x_3 = s_1) \\
 & \quad * P(x_1 = s_1, x_2 = s_1, x_3 = s_1) \\
 & \quad (\text{Because an observation is only dependent on the day when it is observed}) \\
 & = P(o_1 = \varphi_4 | x_1 = s_1) * P(o_2 = \varphi_1 | x_2 = s_1) * P(o_3 = \varphi_2 | x_3 = s_1) \\
 & \quad * P(x_3 = s_1 | x_1 = s_1, x_2 = s_1) * P(x_1 = s_1, x_2 = s_1) \\
 & \quad (\text{Due to multiplication rule specified by formula III.1.1.3}) \\
 & = P(o_1 = \varphi_4 | x_1 = s_1) * P(o_2 = \varphi_1 | x_2 = s_1) * P(o_3 = \varphi_2 | x_3 = s_1) \\
 & \quad * P(x_3 = s_1 | x_2 = s_1) * P(x_1 = s_1, x_2 = s_1) \\
 & \quad (\text{Due to Markov property, current state is only dependent on right previous state}) \\
 & = P(o_1 = \varphi_4 | x_1 = s_1) * P(o_2 = \varphi_1 | x_2 = s_1) * P(o_3 = \varphi_2 | x_3 = s_1) \\
 & \quad * P(x_3 = s_1 | x_2 = s_1) * P(x_2 = s_1 | x_1 = s_1) * P(x_1 = s_1) \\
 & \quad (\text{Due to multiplication rule specified by formula III.1.1.3}) \\
 & = b_{14} b_{11} b_{12} a_{11} a_{11} \pi_1
 \end{aligned}$$

(According to parameters A , B , and \prod specified in tables IV.4.1, IV.4.2, and IV.4.3)

Similarly, we have:

$$\begin{aligned}
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_1, x_3 = s_2) \\
 & \quad * P(x_1 = s_1, x_2 = s_1, x_3 = s_2) = b_{14} b_{11} b_{22} a_{12} a_{11} \pi_1 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_1, x_3 = s_3) \\
 & \quad * P(x_1 = s_1, x_2 = s_1, x_3 = s_3) = b_{14} b_{11} b_{32} a_{13} a_{11} \pi_1 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_2, x_3 = s_1) \\
 & \quad * P(x_1 = s_1, x_2 = s_2, x_3 = s_1) = b_{14} b_{21} b_{12} a_{21} a_{12} \pi_1 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_2, x_3 = s_2) \\
 & \quad * P(x_1 = s_1, x_2 = s_2, x_3 = s_2) = b_{14} b_{21} b_{22} a_{22} a_{12} \pi_1 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_2, x_3 = s_3) \\
 & \quad * P(x_1 = s_1, x_2 = s_2, x_3 = s_3) = b_{14} b_{21} b_{32} a_{23} a_{12} \pi_1 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_3, x_3 = s_1) \\
 & \quad * P(x_1 = s_1, x_2 = s_3, x_3 = s_1) = b_{14} b_{31} b_{12} a_{31} a_{13} \pi_1 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_3, x_3 = s_2) \\
 & \quad * P(x_1 = s_1, x_2 = s_3, x_3 = s_2) = b_{14} b_{31} b_{22} a_{32} a_{13} \pi_1 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_1, x_2 = s_3, x_3 = s_3) \\
 & \quad * P(x_1 = s_1, x_2 = s_3, x_3 = s_3) = b_{14} b_{31} b_{32} a_{33} a_{13} \pi_1 \\
 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_1, x_3 = s_1) \\
 & \quad * P(x_1 = s_2, x_2 = s_1, x_3 = s_1) = b_{24} b_{11} b_{12} a_{11} a_{21} \pi_2
 \end{aligned}$$

$$\begin{aligned}
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_1, x_3 = s_2) \\
 & \quad * P(x_1 = s_2, x_2 = s_1, x_3 = s_2) = b_{24}b_{11}b_{22}a_{12}a_{21}\pi_2 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_1, x_3 = s_3) \\
 & \quad * P(x_1 = s_2, x_2 = s_1, x_3 = s_3) = b_{24}b_{11}b_{32}a_{13}a_{21}\pi_2 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_2, x_3 = s_1) \\
 & \quad * P(x_1 = s_2, x_2 = s_2, x_3 = s_1) = b_{24}b_{21}b_{12}a_{21}a_{22}\pi_2 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_2, x_3 = s_2) \\
 & \quad * P(x_1 = s_2, x_2 = s_2, x_3 = s_2) = b_{24}b_{21}b_{22}a_{22}a_{22}\pi_2 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_2, x_3 = s_3) \\
 & \quad * P(x_1 = s_2, x_2 = s_2, x_3 = s_3) = b_{24}b_{21}b_{32}a_{23}a_{22}\pi_2 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_3, x_3 = s_1) \\
 & \quad * P(x_1 = s_2, x_2 = s_3, x_3 = s_1) = b_{24}b_{31}b_{12}a_{31}a_{23}\pi_2 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_3, x_3 = s_2) \\
 & \quad * P(x_1 = s_2, x_2 = s_3, x_3 = s_2) = b_{24}b_{31}b_{22}a_{32}a_{23}\pi_2 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_2, x_2 = s_3, x_3 = s_3) \\
 & \quad * P(x_1 = s_2, x_2 = s_3, x_3 = s_3) = b_{24}b_{31}b_{32}a_{33}a_{23}\pi_2 \\
 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_1, x_3 = s_1) \\
 & \quad * P(x_1 = s_3, x_2 = s_1, x_3 = s_1) = b_{34}b_{11}b_{12}a_{11}a_{31}\pi_3 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_1, x_3 = s_2) \\
 & \quad * P(x_1 = s_3, x_2 = s_1, x_3 = s_2) = b_{34}b_{11}b_{22}a_{12}a_{31}\pi_3 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_1, x_3 = s_3) \\
 & \quad * P(x_1 = s_3, x_2 = s_1, x_3 = s_3) = b_{34}b_{11}b_{32}a_{13}a_{31}\pi_3 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_2, x_3 = s_1) \\
 & \quad * P(x_1 = s_3, x_2 = s_2, x_3 = s_1) = b_{34}b_{21}b_{12}a_{21}a_{32}\pi_3 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_2, x_3 = s_2) \\
 & \quad * P(x_1 = s_3, x_2 = s_2, x_3 = s_2) = b_{34}b_{21}b_{22}a_{22}a_{32}\pi_3 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_2, x_3 = s_3) \\
 & \quad * P(x_1 = s_3, x_2 = s_2, x_3 = s_3) = b_{34}b_{21}b_{32}a_{23}a_{32}\pi_3 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_3, x_3 = s_1) \\
 & \quad * P(x_1 = s_3, x_2 = s_3, x_3 = s_1) = b_{34}b_{31}b_{12}a_{31}a_{33}\pi_3 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_3, x_3 = s_2) \\
 & \quad * P(x_1 = s_3, x_2 = s_3, x_3 = s_2) = b_{34}b_{31}b_{22}a_{32}a_{33}\pi_3 \\
 & P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2 | x_1 = s_3, x_2 = s_3, x_3 = s_3) \\
 & \quad * P(x_1 = s_3, x_2 = s_3, x_3 = s_3) = b_{34}b_{31}b_{32}a_{33}a_{33}\pi_3
 \end{aligned}$$

It implies

$$\begin{aligned}
 P(O|\Delta) &= P(o_1 = \varphi_4, o_2 = \varphi_1, o_3 = \varphi_2) \\
 &= b_{14}b_{11}b_{12}a_{11}a_{11}\pi_1 + b_{14}b_{11}b_{22}a_{12}a_{11}\pi_1 + b_{14}b_{11}b_{32}a_{13}a_{11}\pi_1 \\
 &\quad + b_{14}b_{21}b_{12}a_{21}a_{12}\pi_1 + b_{14}b_{21}b_{22}a_{22}a_{12}\pi_1 + b_{14}b_{21}b_{32}a_{23}a_{12}\pi_1 \\
 &\quad + b_{14}b_{31}b_{12}a_{31}a_{13}\pi_1 + b_{14}b_{31}b_{22}a_{32}a_{13}\pi_1 + b_{14}b_{31}b_{32}a_{33}a_{13}\pi_1 \\
 &\quad + b_{24}b_{11}b_{12}a_{11}a_{21}\pi_2 + b_{24}b_{11}b_{22}a_{12}a_{21}\pi_2 + b_{24}b_{11}b_{32}a_{13}a_{21}\pi_2 \\
 &\quad + b_{24}b_{21}b_{12}a_{21}a_{22}\pi_2 + b_{24}b_{21}b_{22}a_{22}a_{22}\pi_2 + b_{24}b_{21}b_{32}a_{23}a_{22}\pi_2 \\
 &\quad + b_{24}b_{31}b_{12}a_{31}a_{23}\pi_2 + b_{24}b_{31}b_{22}a_{32}a_{23}\pi_2 + b_{24}b_{31}b_{32}a_{33}a_{23}\pi_2 \\
 &\quad + b_{34}b_{11}b_{12}a_{11}a_{31}\pi_3 + b_{34}b_{11}b_{22}a_{12}a_{31}\pi_3 + b_{34}b_{11}b_{32}a_{13}a_{31}\pi_3 \\
 &\quad + b_{34}b_{21}b_{12}a_{21}a_{32}\pi_3 + b_{34}b_{21}b_{22}a_{22}a_{32}\pi_3 + b_{34}b_{21}b_{32}a_{23}a_{32}\pi_3 \\
 &\quad + b_{34}b_{31}b_{12}a_{31}a_{33}\pi_3 + b_{34}b_{31}b_{22}a_{32}a_{33}\pi_3 + b_{34}b_{31}b_{32}a_{33}a_{33}\pi_3 \\
 &= 0.05 * 0.6 * 0.2 * 0.5 * 0.5 * 0.33 \\
 &+ 0.05 * 0.6 * 0.25 * 0.25 * 0.5 * 0.33
 \end{aligned}$$

$$\begin{aligned}
 & +0.05 * 0.6 * 0.1 * 0.25 * 0.5 * 0.33 \\
 & +0.05 * 0.25 * 0.2 * 0.3 * 0.25 * 0.33 \\
 & +0.05 * 0.25 * 0.25 * 0.4 * 0.25 * 0.33 \\
 & +0.05 * 0.25 * 0.1 * 0.3 * 0.25 * 0.33 \\
 & +0.05 * 0.05 * 0.2 * 0.25 * 0.25 * 0.33 \\
 & +0.05 * 0.05 * 0.25 * 0.25 * 0.25 * 0.33 \\
 & +0.05 * 0.05 * 0.1 * 0.5 * 0.25 * 0.33 \\
 & +0.25 * 0.6 * 0.2 * 0.5 * 0.3 * 0.33 \\
 & +0.25 * 0.6 * 0.25 * 0.25 * 0.3 * 0.33 \\
 & +0.25 * 0.6 * 0.1 * 0.25 * 0.3 * 0.33 \\
 & +0.25 * 0.25 * 0.2 * 0.3 * 0.4 * 0.33 \\
 & +0.25 * 0.25 * 0.25 * 0.4 * 0.4 * 0.33 \\
 & +0.25 * 0.25 * 0.1 * 0.3 * 0.4 * 0.33 \\
 & +0.25 * 0.05 * 0.2 * 0.25 * 0.3 * 0.33 \\
 & +0.25 * 0.05 * 0.25 * 0.25 * 0.3 * 0.33 \\
 & +0.25 * 0.05 * 0.1 * 0.5 * 0.3 * 0.33 \\
 & +0.5 * 0.6 * 0.2 * 0.5 * 0.25 * 0.33 \\
 & +0.5 * 0.6 * 0.25 * 0.25 * 0.25 * 0.33 \\
 & +0.5 * 0.6 * 0.1 * 0.25 * 0.25 * 0.33 \\
 & +0.5 * 0.25 * 0.2 * 0.3 * 0.25 * 0.33 \\
 & +0.5 * 0.25 * 0.25 * 0.4 * 0.25 * 0.33 \\
 & +0.5 * 0.25 * 0.1 * 0.3 * 0.25 * 0.33 \\
 & +0.5 * 0.05 * 0.2 * 0.25 * 0.5 * 0.33 \\
 & +0.5 * 0.05 * 0.25 * 0.25 * 0.5 * 0.33 \\
 & +0.5 * 0.05 * 0.1 * 0.5 * 0.5 * 0.33 \\
 & = 0.012981
 \end{aligned}$$

It is easy to explain that given weather HMM modeled by parameters A , B , and Π specified in tables IV.4.1, IV.4.2, and IV.4.3, the event that it is *soggy*, *dry*, and *dryish* in three successive days is rare because the probability of such event $P(O/\Delta)$ is low ($\approx 1.3\%$). It is easy to recognize that it is impossible to browse all combinational cases of given observation sequence $O = \{o_1, o_2, \dots, o_T\}$ as we knew that it is necessary to survey $3^3=27$ mutually exclusive cases of weather states with a tiny number of observations *{soggy, dry, dryish}*. Exactly, given n states and T observations, it takes extremely expensive cost to survey n^T cases. According to (Rabiner, 1989, pp. 262-263), there is a so-called *forward-backward procedure* to decrease computational cost for determining the probability $P(O/\Delta)$. Let $\alpha_t(i)$ be the joint probability of partial observation sequence $\{o_1, o_2, \dots, o_t\}$ and state $x_t=s_i$ where $1 \leq t \leq T$, specified by formula IV.4.1.1.

$$\alpha_t(i) = P(o_1, o_2, \dots, o_t, x_t = s_i | \Delta)$$

Formula IV.4.1.1. Forward variable

The joint probability $\alpha_t(i)$ is also called *forward variable* at time point t and state s_i . The product $\alpha_t(i)a_{ij}$ where a_{ij} is the transition probability from state i to state j counts for probability of joint event that partial observation sequence $\{o_1, o_2, \dots, o_t\}$ exists and the state s_i at time point t is changed to s_j at time point $t+1$.

$$\begin{aligned}
 \alpha_t(i)a_{ij} &= P(o_1, o_2, \dots, o_t, x_t = s_i | \Delta)P(x_{t+1} = s_j | x_t = s_i) \\
 &= P(o_1, o_2, \dots, o_t | x_t = s_i)P(x_t = s_i)P(x_{t+1} = s_j | x_t = s_i)
 \end{aligned}$$

(Due to multiplication rule specified by formula III.1.1.3)

$$= P(o_1, o_2, \dots, o_t | x_t = s_i) P(x_{t+1} = s_j | x_t = s_i) P(x_t = s_i)$$

$$= P(o_1, o_2, \dots, o_t, x_{t+1} = s_j | x_t = s_i) P(x_t = s_i)$$

(Because the partial observation sequence $\{o_1, o_2, \dots, o_t\}$ is independent from next state x_{t+1} given current state x_t)

$$= P(o_1, o_2, \dots, o_t, x_t = s_i, x_{t+1} = s_j)$$

(Due to multiplication rule specified by formula III.1.1.3)

Summing product $\alpha_t(i)a_{ij}$ over all n possible states of x_t produces probability of joint event that partial observation sequence $\{o_1, o_2, \dots, o_t\}$ exists and the next state is $x_{t+1}=s_j$ regardless of the state x_t .

$$\sum_{i=1}^n \alpha_t(i)a_{ij} = \sum_{i=1}^n P(o_1, o_2, \dots, o_t, x_t = s_i, x_{t+1} = s_j) = P(o_1, o_2, \dots, o_t, x_{t+1} = s_j)$$

The forward variable at time point $t+1$ and state s_j is calculated on $\alpha_t(i)$ as follows:

$$\alpha_{t+1}(j) = P(o_1, o_2, \dots, o_t, o_{t+1}, x_{t+1} = s_j | \Delta)$$

$$= P(o_{t+1} | o_1, o_2, \dots, o_t, x_{t+1} = s_j) P(o_1, o_2, \dots, o_t, x_{t+1} = s_j)$$

(Due to multiplication rule specified by formula III.1.1.3)

$$= P(o_{t+1} | x_{t+1} = s_j) P(o_1, o_2, \dots, o_t, x_{t+1} = s_j)$$

(Due to observations are mutually independent)

$$= b_j(o_{t+1}) \sum_{i=1}^n \alpha_t(i)a_{ij}$$

Where $b_j(o_{t+1})$ is the probability of observation o_{t+1} when the state stochastic process is in state s_j , please see an example of observation probability matrix shown in table IV.4.3. In brief, please pay attention to recurrence property of forward variable specified by formula IV.4.1.2.

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^n \alpha_t(i)a_{ij} \right) b_j(o_{t+1})$$

Formula IV.4.1.2. Recurrence property of forward variable

The aforementioned construction of forward recurrence formula IV.4.1.2 is essentially to build up Markov chain, illustrated by figure IV.4.1.1 (Rabiner, 1989, p. 262).

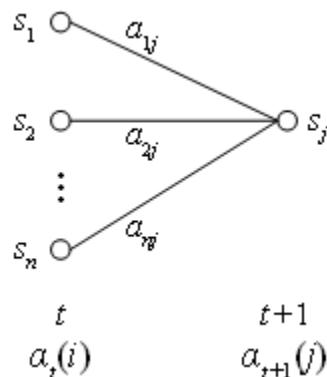


Figure IV.4.1.1. Construction of recurrence formula for forward variable

According to the forward recurrence formula IV.4.1.2, given observation sequence $O = \{o_1, o_2, \dots, o_T\}$, we have:

$$\alpha_T(i) = P(o_1, o_2, \dots, o_T, x_T = s_i | \Delta)$$

The probability $P(O|\Delta)$ is sum of $\alpha_T(i)$ over all n possible states of x_T , specified by formula IV.4.1.3.

$$P(O|\Delta) = P(o_1, o_2, \dots, o_T) = \sum_{i=1}^n P(o_1, o_2, \dots, o_T, x_T = s_i | \Delta) = \sum_{i=1}^n \alpha_T(i)$$

Formula IV.4.1.3. Probability $P(O|\Delta)$ based on forward variable

The forward-backward procedure to calculate the probability $P(O|\Delta)$, based on forward formulas IV.4.1.2 and IV.4.1.3, includes three steps as shown in table IV.4.1.1 (Rabiner, 1989, p. 262).

1. Initialization step: Initializing $\alpha_1(i) = b_i(o_1)\pi_i$ for all $1 \leq i \leq n$
2. Recurrence step: Calculating all $\alpha_{t+1}(j)$ for all $1 \leq j \leq n$ and $1 \leq t \leq T - 1$ according to formula IV.4.1.2.

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^n \alpha_t(i) a_{ij} \right) b_j(o_{t+1})$$

3. Evaluation step: Calculating the probability $P(O|\Delta) = \sum_{i=1}^n \alpha_T(i)$

Table IV.4.1.1. Forward-backward procedure based on forward variable to calculate the probability $P(O|\Delta)$

It is required to execute $n + 2n^2(T-1) + n - 1 = 2n^2(T-1) + 2n - 1$ operations for forward-backward procedure based on forward variable due to:

- There are n multiplications at initialization step.
- There are n multiplications, $n-1$ additions, and 1 multiplication over the expression $(\sum_{i=1}^n \alpha_t(i) a_{ij}) b_j(o_{t+1})$. There are n cases of values $\alpha_{t+1}(j)$ for all $1 \leq j \leq n$ at time point $t+1$. So, there are $(n+n-1+1)n = 2n^2$ operations over values $\alpha_{t+1}(j)$ for all $1 \leq j \leq n$ at time point t . The recurrence step runs over $T-1$ times and so, there are $2n^2(T-1)$ operations at recurrence step.
- There are $n-1$ additions at evaluation step.

Inside $2n^2(T-1) + 2n - 1$ operations, there are $n + (n+1)n(T-1) = n + (n^2+n)(T-1)$ multiplications and $(n-1)n(T-1) + n - 1 = (n^2+n)(T-1) + n - 1$ additions.

Going back example with weather HMM whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3. We need to re-calculate the probability of observation sequence $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$ by forward-backward procedure shown in table IV.4.1.1 (based on forward variable). According to initialization step of forward-backward procedure based on forward variable, we have:

$$\alpha_1(1) = b_1(o_1 = \varphi_4)\pi_1 = b_{14}\pi_1 = 0.05 * 0.33 = 0.0165$$

$$\alpha_1(2) = b_2(o_1 = \varphi_4)\pi_2 = b_{24}\pi_2 = 0.25 * 0.33 = 0.0825$$

$$\alpha_1(3) = b_3(o_1 = \varphi_4)\pi_3 = b_{34}\pi_3 = 0.5 * 0.33 = 0.165$$

According to recurrence step of forward-backward procedure based on forward variable, we have:

$$\begin{aligned} \alpha_2(1) &= \left(\sum_{i=1}^3 \alpha_1(i) a_{i1} \right) b_1(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i) a_{i1} \right) b_{11} \\ &= (\alpha_1(1)a_{11} + \alpha_1(2)a_{21} + \alpha_1(3)a_{31})b_{11} \\ &= (0.0165 * 0.5 + 0.0825 * 0.3 + 0.165 * 0.25) * 0.6 = 0.04455 \end{aligned}$$

$$\begin{aligned}
 \alpha_2(2) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_2(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_{21} \\
 &= (\alpha_1(1)a_{12} + \alpha_1(2)a_{22} + \alpha_1(3)a_{32})b_{21} \\
 &= (0.0165 * 0.25 + 0.0825 * 0.4 + 0.165 * 0.25) * 0.25 = 0.019594 \\
 \alpha_2(3) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_3(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_{31} \\
 &= (\alpha_1(1)a_{13} + \alpha_1(2)a_{23} + \alpha_1(3)a_{33})b_{31} \\
 &= (0.0165 * 0.25 + 0.0825 * 0.3 + 0.165 * 0.5) * 0.05 = 0.005569 \\
 \alpha_3(1) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_1(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_{12} \\
 &= (\alpha_2(1)a_{11} + \alpha_2(2)a_{21} + \alpha_2(3)a_{31})b_{12} \\
 &= (0.04455 * 0.5 + 0.019594 * 0.3 + 0.005569 * 0.25) * 0.2 \\
 &= 0.005909 \\
 \alpha_3(2) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_2(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_{22} \\
 &= (\alpha_2(1)a_{12} + \alpha_2(2)a_{22} + \alpha_2(3)a_{32})b_{22} \\
 &= (0.04455 * 0.25 + 0.019594 * 0.4 + 0.005569 * 0.25) * 0.25 \\
 &= 0.005092 \\
 \alpha_3(3) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i3} \right) b_3(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i3} \right) b_{32} \\
 &= (\alpha_2(1)a_{13} + \alpha_2(2)a_{23} + \alpha_2(3)a_{33})b_{32} \\
 &= (0.04455 * 0.25 + 0.019594 * 0.3 + 0.005569 * 0.5) * 0.1 \\
 &= 0.00198
 \end{aligned}$$

According to evaluation step of forward-backward procedure based on forward variable, the probability of observation sequence $O = \{o_1=s_4=\text{soggy}, o_2=s_1=\text{dry}, o_3=s_2=\text{dryish}\}$ is:

$$P(O|\Delta) = 0.005909 + 0.005092 + 0.00198 = 0.012981$$

The result from the forward-backward procedure based on forward variable is the same to the one from aforementioned brute-force method that browses all $3^3=27$ mutually exclusive cases of weather states.

There is interesting thing that the forward-backward procedure can be implemented based on so-called *backward variable*. Let $\beta_t(i)$ be the backward variable which is conditional probability of partial observation sequence $\{o_t, o_{t+1}, \dots, o_T\}$ given state $x_t=s_i$ where $1 \leq t \leq T$, specified by formula [IV.4.1.4](#).

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | x_t = s_i, \Delta)$$

Formula IV.4.1.4. Backward variable

We have

$$\begin{aligned}
 &a_{ij}b_j(o_{t+1})\beta_{t+1}(j) \\
 &= P(x_{t+1} = s_j | x_t = s_i)P(o_{t+1} | x_{t+1} = s_j)P(o_{t+2}, o_{t+3}, \dots, o_T | x_{t+1} = s_j, \Delta) \\
 &= P(x_{t+1} = s_j | x_t = s_i)P(o_{t+1}, o_{t+2}, o_{t+3}, \dots, o_T | x_{t+1} = s_j, \Delta) \\
 &\quad (\text{Because observations } o_{t+1}, o_{t+2}, \dots, o_T \text{ are mutually independent}) \\
 &= P(x_{t+1} = s_j | x_t = s_i)P(o_{t+1}, o_{t+2}, o_{t+3}, \dots, o_T | x_t = s_i, x_{t+1} = s_j, \Delta)
 \end{aligned}$$

(Because partial observation sequence $o_{t+1}, o_{t+2}, \dots, o_T$ is independent from state x_t at time point t)

$$= P(o_{t+1}, o_{t+2}, o_{t+3}, \dots, o_T, x_{t+1} = s_j | x_t = s_i, \Delta)$$

(Due to multiplication rule specified by formula III.1.1.3)

Summing the product $a_{ij}b_j(o_{t+1})\beta_{t+1}(j)$ over all n possible states of $x_{t+1}=s_j$, we have:

$$\begin{aligned} & \sum_{j=1}^n a_{ij}b_j(o_{t+1})\beta_{t+1}(j) = \sum_{j=1}^n P(o_{t+1}, o_{t+2}, o_{t+3}, \dots, o_T, x_{t+1} = s_j | x_t = s_i, \Delta) \\ & = P(o_{t+1}, o_{t+2}, o_{t+3}, \dots, o_T | x_t = s_i, \Delta) \\ & \quad (\text{Due to the total probability rule specified by formula III.1.1.4}) \\ & = \beta_t(i) \end{aligned}$$

In brief, the recurrence property of backward variable specified by formula IV.4.1.5.

$$\beta_t(i) = \sum_{j=1}^n a_{ij}b_j(o_{t+1})\beta_{t+1}(j)$$

Formula IV.4.1.5. Recurrence property of backward variable

Where $b_j(o_{t+1})$ is the probability of observation o_{t+1} when the state stochastic process is in state s_j , please see an example of observation probability matrix shown in table IV.4.3. The construction of backward recurrence formula IV.4.1.5 is essentially to build up Markov chain, illustrated by figure IV.4.1.2 (Rabiner, 1989, p. 263).

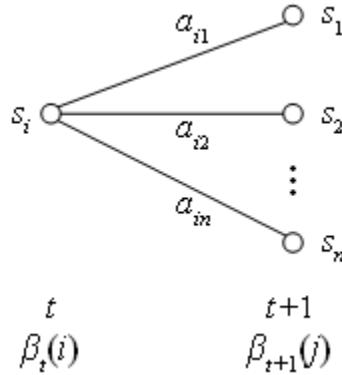


Figure IV.4.1.2. Construction of recurrence formula for backward variable

According to the backward recurrence formula IV.4.1.5, given observation sequence $O = \{o_1, o_2, \dots, o_T\}$, we have:

$$\beta_1(i) = P(o_2, o_3, \dots, o_T | x_1 = s_i, \Delta)$$

The product $\pi_i b_i(o_1)\beta_1(i)$ is:

$$\begin{aligned} \pi_i b_i(o_1)\beta_1(i) &= P(x_1 = s_i)P(o_1 | x_1 = s_i)P(o_2, o_3, \dots, o_T | x_1 = s_i, \Delta) \\ &= P(x_1 = s_i)P(o_1, o_2, o_3, \dots, o_T | x_1 = s_i, \Delta) \\ &\quad (\text{Because observations } o_1, o_2, \dots, o_T \text{ are mutually independent}) \end{aligned}$$

$$= P(o_1, o_2, o_3, \dots, o_T, x_1 = s_i | \Delta)$$

It implies that the probability $P(O|\Delta)$ is:

$$P(O|\Delta) = P(o_1, o_2, \dots, o_T)$$

$$= \sum_{i=1}^n P(o_1, o_2, \dots, o_T, x_1 = s_i | \Delta)$$

(Due to the total probability rule specified by formula III.1.1.4)

$$= \sum_{i=1}^n \pi_i b_i(o_1) \beta_1(i)$$

Shortly, the probability $P(O|\Delta)$ is sum of product $\pi_i b_i(o_1) \beta_1(i)$ over all n possible states of $x_1=s_i$, specified by formula IV.4.1.6.

$$P(O|\Delta) = \sum_{i=1}^n \pi_i b_i(o_1) \beta_1(i)$$

Formula IV.4.1.6. Probability $P(O|\Delta)$ based on backward variable

The forward-backward procedure to calculate the probability $P(O|\Delta)$, based on backward formulas IV.4.1.5 and IV.4.1.6, includes three steps as shown in table IV.4.1.2 (Rabiner, 1989, p. 263).

1. Initialization step: Initializing $\beta_T(i) = 1$ for all $1 \leq i \leq n$
 2. Recurrence step: Calculating all $\beta_t(i)$ for all $1 \leq i \leq n$ and $t=T-1, t=T-2, \dots, t=1$, according to formula IV.4.1.5.
- $$\beta_t(i) = \sum_{j=1}^n a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$
3. Evaluation step: Calculating the probability $P(O|\Delta)$ according to formula IV.4.1.6, $P(O|\Delta) = \sum_{i=1}^n \pi_i b_i(o_1) \beta_1(i)$

Table IV.4.1.2. Forward-backward procedure based on backward variable to calculate the probability $P(O|\Delta)$

It is required to execute $(3n-1)n(T-1)+2n+n-1 = 3n^2(T-1)-n(T-4)-1$ operations for forward-backward procedure based on forward variable due to:

- There are $2n$ multiplications and $n-1$ additions over the sum $\sum_{j=1}^n a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$. So, there are $(2n+n-1)n = (3n-1)n$ operations over values $\beta_t(i)$ for all $1 \leq i \leq n$ at time point t . The recurrence step runs over $T-1$ times and so, there are $(3n-1)n(T-1)$ operations at recurrence step.
- There are $2n$ multiplications and $n-1$ additions over the sum $\sum_{i=1}^n \pi_i b_i(o_1) \beta_1(i)$ at evaluation step.

Inside $3n^2(T-1)-n(T-4)-1$ operations, there are $2n^2(T-1)+2n$ multiplications and $(n-1)n(T-1)+n-1 = n^2(T-1)-n(T-2)-1$ additions.

Going back example with weather HMM whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3. We need to re-calculate the probability of observation sequence $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$ by forward-backward procedure shown in table IV.4.1.2 (based on backward variable). According to initialization step of forward-backward procedure based on backward variable, we have:

$$\beta_3(1) = \beta_3(2) = \beta_3(3) = 1$$

According to recurrence step of forward-backward procedure based on backward variable, we have:

$$\begin{aligned} \beta_2(1) &= \sum_{j=1}^n a_{1j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{1j} b_{j2} \beta_3(j) \\ &= a_{11} b_{12} \beta_3(1) + a_{12} b_{22} \beta_3(2) + a_{13} b_{32} \beta_3(3) \\ &= 0.5 * 0.2 * 1 + 0.25 * 0.25 * 1 + 0.25 * 0.1 * 1 = 0.1875 \end{aligned}$$

$$\begin{aligned}
 \beta_2(2) &= \sum_{j=1}^n a_{2j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{2j} b_{j2} \beta_3(j) \\
 &= a_{21} b_{12} \beta_3(1) + a_{22} b_{22} \beta_3(2) + a_{23} b_{32} \beta_3(3) \\
 &= 0.3 * 0.2 * 1 + 0.4 * 0.25 * 1 + 0.3 * 0.1 * 1 = 0.19 \\
 \beta_2(3) &= \sum_{j=1}^n a_{3j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{3j} b_{j2} \beta_3(j) \\
 &= a_{31} b_{12} \beta_3(1) + a_{32} b_{22} \beta_3(2) + a_{33} b_{32} \beta_3(3) \\
 &= 0.25 * 0.2 * 1 + 0.25 * 0.25 * 1 + 0.5 * 0.1 * 1 = 0.1625 \\
 \beta_1(1) &= \sum_{j=1}^n a_{1j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{1j} b_{j1} \beta_2(j) \\
 &= a_{11} b_{11} \beta_2(1) + a_{12} b_{21} \beta_2(2) + a_{13} b_{31} \beta_2(3) \\
 &= 0.5 * 0.6 * 0.1875 + 0.25 * 0.25 * 0.19 + 0.25 * 0.05 * 0.1625 \\
 &= 0.070156 \\
 \beta_1(2) &= \sum_{j=1}^n a_{2j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{2j} b_{j1} \beta_2(j) \\
 &= a_{21} b_{11} \beta_2(1) + a_{22} b_{21} \beta_2(2) + a_{23} b_{31} \beta_2(3) \\
 &= 0.3 * 0.6 * 0.1875 + 0.4 * 0.25 * 0.19 + 0.3 * 0.05 * 0.1625 \\
 &= 0.055188 \\
 \beta_1(3) &= \sum_{j=1}^n a_{3j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{3j} b_{j1} \beta_2(j) \\
 &= a_{31} b_{11} \beta_2(1) + a_{32} b_{21} \beta_2(2) + a_{33} b_{31} \beta_2(3) \\
 &= 0.25 * 0.6 * 0.1875 + 0.25 * 0.25 * 0.19 + 0.5 * 0.05 * 0.1625 \\
 &= 0.044063
 \end{aligned}$$

According to evaluation step of forward-backward procedure based on backward variable, the probability of observation sequence $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$ is:

$$\begin{aligned}
 P(O|\Delta) &= \sum_{i=1}^3 \pi_i b_i(o_1 = \varphi_4) \beta_1(i) = \sum_{i=1}^3 \pi_i b_{i4} \beta_1(i) \\
 &= \pi_1 b_{14} \beta_1(1) + \pi_2 b_{24} \beta_1(2) + \pi_3 b_{34} \beta_1(3) \\
 &= 0.33 * 0.05 * 0.070156 + 0.33 * 0.25 * 0.055188 + 0.33 * 0.5 * 0.044063 = 0.012981
 \end{aligned}$$

The result from the forward-backward procedure based on backward variable is the same to the one from aforementioned brute-force method that browses all $3^3=27$ mutually exclusive cases of weather states and the one from forward-backward procedure based on forward variable.

The research applies solution of uncovering problem into building up learning style model. Thus, the uncovering problem is mentioned particularly in successive sub-section IV.4.2.

IV.4.2. HMM uncovering problem

Recall that given HMM Δ and observation sequence $O = \{o_1, o_2, \dots, o_T\}$ where $o_t \in \Phi$, how to find out a state sequence $X = \{x_1, x_2, \dots, x_T\}$ where $x_t \in S$ so that X 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 observation sequence. In other words, it is required to establish an *optimal criterion* so that the state sequence X leads to maximizing such criterion. The simple criterion is the conditional probability of sequence X with respect to sequence O and model Δ , denoted $P(X|O,\Delta)$. We can apply brute-force strategy: “go through all possible such X and pick the one leading to maximizing the criterion $P(X|O,\Delta)$ ”.

$$X = \operatorname{argmax}_X (P(X|O, \Delta))$$

This strategy is impossible if the number of states and observations is huge. Another popular way is to establish a so-called *individually optimal criterion* (Rabiner, 1989, p. 263) which is described right later.

Let $\gamma_t(i)$ be joint probability that the stochastic process is in state s_i at time point t with observation sequence $O = \{o_1, o_2, \dots, o_T\}$, formula IV.4.2.1 specifies this probability based on forward variable α_t and backward variable β_t .

$$\gamma_t(i) = P(o_1, o_2, \dots, o_T, x_t = s_i | \Delta) = \alpha_t(i)\beta_t(i)$$

Formula IV.4.2.1. Joint probability of being in state s_i at time point t with observation sequence O

The variable $\gamma_t(i)$ is also called *individually optimal criterion* with note that forward variable α_t and backward variable β_t are calculated according to recurrence formulas IV.4.1.2 and IV.4.1.5, respectively.

Following is proof of formula IV.4.2.1.

$$\begin{aligned} \gamma_t(i) &= P(o_1, o_2, \dots, o_T, x_t = s_i | \Delta) \\ &= P(x_t = s_i, o_1, o_2, \dots, o_T | \Delta) \\ &\quad (\text{Due to Bayes' rule specified by formula III.1.1.1}) \\ &= P(o_1, o_2, \dots, o_t, x_t = s_i, o_{t+1}, o_{t+2}, \dots, o_T | \Delta) \\ &= P(o_1, o_2, \dots, o_t, x_t = s_i | \Delta)P(o_{t+1}, o_{t+2}, \dots, o_T | o_1, o_2, \dots, o_t, x_t = s_i, \Delta) \\ &\quad (\text{Due to multiplication rule specified by formula III.1.1.3}) \\ &= P(o_1, o_2, \dots, o_t, x_t = s_i | \Delta)P(o_{t+1}, o_{t+2}, \dots, o_T | x_t = s_i, \Delta) \\ &\quad (\text{Because observations } o_1, o_2, \dots, o_T \text{ are observed independently}) \\ &= \alpha_t(i)\beta_t(i) \\ &\quad (\text{According to formulas IV.4.1.1 and IV.4.1.4 for determining forward variable and backward variable}) \end{aligned}$$

The state sequence $X = \{x_1, x_2, \dots, x_T\}$ is determined by selecting each state $x_t \in S$ so that it maximizes $\gamma_t(i)$.

$$\begin{aligned} x_t &= \operatorname{argmax}_i P(x_t = s_i | o_1, o_2, \dots, o_T, \Delta) = \operatorname{argmax}_i \frac{\alpha_t(i)\beta_t(i)}{P(o_1, o_2, \dots, o_T | \Delta)} \\ &= \operatorname{argmax}_i \frac{P(o_1, o_2, \dots, o_T, x_t = s_i | \Delta)}{P(o_1, o_2, \dots, o_T | \Delta)} \\ &\quad (\text{Due to Bayes' rule specified by formula III.1.1.1}) \\ &= \operatorname{argmax}_i \frac{\gamma_t(i)}{P(o_1, o_2, \dots, o_T | \Delta)} \\ &\quad (\text{Due to formula IV.4.2.1}) \end{aligned}$$

Because the probability $P(o_1, o_2, \dots, o_T | \Delta)$ is not relevant to state sequence X , it is possible to remove it from the optimization criterion. Thus, formula IV.4.2.2 specifies how to find out the optimal state x_t of X at time point t .

$$x_t = \operatorname{argmax}_i \gamma_t(i) = \operatorname{argmax}_i \alpha_t(i)\beta_t(i)$$

Formula IV.4.2.2. Optimal state at time point t

Note that index i is identified with state $s_i \in S$ according to formula IV.4.2.2. The optimal state x_t of X at time point t is the one that maximizes product $\alpha_t(i) \beta_t(i)$ over all values s_i . The procedure to find out state sequence $X = \{x_1, x_2, \dots, x_T\}$ based on individually optimal criterion is called *individually optimal procedure* that includes three steps, shown in table IV.4.2.1.

1. Initialization step:
 - Initializing $\alpha_1(i) = b_i(o_1)\pi_i$ for all $1 \leq i \leq n$
 - Initializing $\beta_T(i) = 1$ for all $1 \leq i \leq n$
2. Recurrence step:
 - Calculating all $\alpha_{t+1}(i)$ for all $1 \leq i \leq n$ and $1 \leq t \leq T - 1$ according to formula IV.4.1.2.
 - Calculating all $\beta_t(i)$ for all $1 \leq i \leq n$ and $t=T-1, t=T-2, \dots, t=1$, according to formula IV.4.1.5.
 - Calculating all $\gamma_t(i) = \alpha_t(i)\beta_t(i)$ for all $1 \leq i \leq n$ and $1 \leq t \leq T$ according to formula IV.4.2.1.
 - Determining optimal state x_t of X at time point t is the one that maximizes $\gamma_t(i)$ over all values s_i .
$$x_t = \operatorname{argmax}_i \gamma_t(i)$$
3. Final step: The state sequence $X = \{x_1, x_2, \dots, x_T\}$ is totally determined when its partial states x_t (s) where $1 \leq t \leq T$ are found in recurrence step.

Table IV.4.2.1. Individually optimal procedure to solve uncovering problem

It is required to execute $n + (5n^2 - n)(T-1) + 2nT$ operations for individually optimal procedure due to:

- There are n multiplications for calculating $\alpha_1(i)$ (s).
- The recurrence step runs over $T-1$ times. There are $2n^2(T-1)$ operations for determining $\alpha_{t+1}(i)$ (s) over all $1 \leq i \leq n$ and $1 \leq t \leq T - 1$. There are $(3n-1)n(T-1)$ operations for determining $\beta_t(i)$ (s) over all $1 \leq i \leq n$ and $t=T-1, t=T-2, \dots, t=1$. There are nT multiplications for determining $\gamma_t(i) = \alpha_t(i)\beta_t(i)$ over all $1 \leq i \leq n$ and $1 \leq t \leq T$. There are nT comparisons for determining optimal state $x_t = \operatorname{argmax}_i \gamma_t(i)$ over all $1 \leq i \leq n$ and $1 \leq t \leq T$. In general, there are $2n^2(T-1) + (3n-1)n(T-1) + nT + nT = (5n^2 - n)(T-1) + 2nT$ operations at the recurrence step.

Inside $n + (5n^2 - n)(T-1) + 2nT$ operations, there are $n + (n+1)n(T-1) + 2n^2(T-1) + nT = (3n^2 + n)(T-1) + nT + n$ multiplications and $(n-1)n(T-1) + (n-1)n(T-1) = 2(n^2 - n)(T-1)$ additions and nT comparisons.

For example, given HMM Δ whose parameters A , B , and Π specified in tables IV.4.1, IV.4.2, and IV.4.3, which is designed for weather forecast. Suppose humidity is *soggy* and *dry* in days 1 and 2, respectively. We apply individual optimal procedure into solving the uncovering problem that finding out the optimal state sequence $X = \{x_1, x_2\}$ with regard to observation sequence $O = \{o_1=\varphi_4=\text{soggy}, o_2=\varphi_1=\text{dry}, o_3=\varphi_2=\text{dryish}\}$. According to formulas IV.4.1.2 and IV.4.1.5, forward variable and backward variable are calculated as follows:

$$\alpha_1(1) = b_1(o_1 = \varphi_4)\pi_1 = b_{14}\pi_1 = 0.05 * 0.33 = 0.0165$$

$$\alpha_1(2) = b_2(o_1 = \varphi_4)\pi_2 = b_{24}\pi_2 = 0.25 * 0.33 = 0.0825$$

$$\alpha_1(3) = b_3(o_1 = \varphi_4)\pi_3 = b_{34}\pi_3 = 0.5 * 0.33 = 0.165$$

$$\begin{aligned}
 \alpha_2(1) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_1(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_{11} \\
 &= (\alpha_1(1)a_{11} + \alpha_1(2)a_{21} + \alpha_1(3)a_{31})b_{11} \\
 &= (0.0165 * 0.5 + 0.0825 * 0.3 + 0.165 * 0.25) * 0.6 = 0.04455 \\
 \alpha_2(2) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_2(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_{21} \\
 &= (\alpha_1(1)a_{12} + \alpha_1(2)a_{22} + \alpha_1(3)a_{32})b_{21} \\
 &= (0.0165 * 0.25 + 0.0825 * 0.4 + 0.165 * 0.25) * 0.25 = 0.019594 \\
 \alpha_2(3) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_3(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_{31} \\
 &= (\alpha_1(1)a_{13} + \alpha_1(2)a_{23} + \alpha_1(3)a_{33})b_{31} \\
 &= (0.0165 * 0.25 + 0.0825 * 0.3 + 0.165 * 0.5) * 0.05 = 0.005569 \\
 \alpha_3(1) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_1(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_{12} \\
 &= (\alpha_2(1)a_{11} + \alpha_2(2)a_{21} + \alpha_2(3)a_{31})b_{12} \\
 &= (0.04455 * 0.5 + 0.019594 * 0.3 + 0.005569 * 0.25) * 0.2 \\
 &= 0.005909 \\
 \alpha_3(2) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_2(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_{22} \\
 &= (\alpha_2(1)a_{12} + \alpha_2(2)a_{22} + \alpha_2(3)a_{32})b_{22} \\
 &= (0.04455 * 0.25 + 0.019594 * 0.4 + 0.005569 * 0.25) * 0.25 \\
 &= 0.005092 \\
 \alpha_3(3) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i3} \right) b_3(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i3} \right) b_{32} \\
 &= (\alpha_2(1)a_{13} + \alpha_2(2)a_{23} + \alpha_2(3)a_{33})b_{32} \\
 &= (0.04455 * 0.25 + 0.019594 * 0.3 + 0.005569 * 0.5) * 0.1 \\
 &= 0.00198 \\
 \beta_3(1) &= \beta_3(2) = \beta_3(3) = 1 \\
 \beta_2(1) &= \sum_{j=1}^n a_{1j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{1j} b_{j2} \beta_3(j) \\
 &= a_{11} b_{12} \beta_3(1) + a_{12} b_{22} \beta_3(2) + a_{13} b_{32} \beta_3(3) \\
 &= 0.5 * 0.2 * 1 + 0.25 * 0.25 * 1 + 0.25 * 0.1 * 1 = 0.1875 \\
 \beta_2(2) &= \sum_{j=1}^n a_{2j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{2j} b_{j2} \beta_3(j) \\
 &= a_{21} b_{12} \beta_3(1) + a_{22} b_{22} \beta_3(2) + a_{23} b_{32} \beta_3(3) \\
 &= 0.3 * 0.2 * 1 + 0.4 * 0.25 * 1 + 0.3 * 0.1 * 1 = 0.19 \\
 \beta_2(3) &= \sum_{j=1}^n a_{3j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{3j} b_{j2} \beta_3(j) \\
 &= a_{31} b_{12} \beta_3(1) + a_{32} b_{22} \beta_3(2) + a_{33} b_{32} \beta_3(3) \\
 &= 0.25 * 0.2 * 1 + 0.25 * 0.25 * 1 + 0.5 * 0.1 * 1 = 0.1625
 \end{aligned}$$

$$\begin{aligned}
 \beta_1(1) &= \sum_{j=1}^n a_{1j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{1j} b_{j1} \beta_2(j) \\
 &= a_{11} b_{11} \beta_2(1) + a_{12} b_{21} \beta_2(2) + a_{13} b_{31} \beta_2(3) \\
 &= 0.5 * 0.6 * 0.1875 + 0.25 * 0.25 * 0.19 + 0.25 * 0.05 * 0.1625 \\
 &= 0.070156 \\
 \beta_1(2) &= \sum_{j=1}^n a_{2j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{2j} b_{j1} \beta_2(j) \\
 &= a_{21} b_{11} \beta_2(1) + a_{22} b_{21} \beta_2(2) + a_{23} b_{31} \beta_2(3) \\
 &= 0.3 * 0.6 * 0.1875 + 0.4 * 0.25 * 0.19 + 0.3 * 0.05 * 0.1625 \\
 &= 0.055188 \\
 \beta_1(3) &= \sum_{j=1}^n a_{3j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{3j} b_{j1} \beta_2(j) \\
 &= a_{31} b_{11} \beta_2(1) + a_{32} b_{21} \beta_2(2) + a_{33} b_{31} \beta_2(3) \\
 &= 0.25 * 0.6 * 0.1875 + 0.25 * 0.25 * 0.19 + 0.5 * 0.05 * 0.1625 \\
 &= 0.044063
 \end{aligned}$$

According to recurrence step of individually optimal procedure, individually optimal criterion $\gamma_t(i)$ and optimal state x_t are calculated as follows:

$$\begin{aligned}
 \gamma_1(1) &= \alpha_1(1)\beta_1(1) = 0.0165 * 0.070156 = 0.001158 \\
 \gamma_1(2) &= \alpha_1(2)\beta_1(2) = 0.0825 * 0.055188 = 0.004553 \\
 \gamma_1(3) &= \alpha_1(3)\beta_1(3) = 0.165 * 0.044063 = 0.00727 \\
 x_1 &= \operatorname{argmax}_i \{\gamma_1(i)\} = \operatorname{argmax}_i \{\gamma_1(1), \gamma_1(2), \gamma_1(3)\} = s_3 = \text{rainy} \\
 \gamma_2(1) &= \alpha_2(1)\beta_2(1) = 0.04455 * 0.1875 = 0.008353 \\
 \gamma_2(2) &= \alpha_2(2)\beta_2(2) = 0.019594 * 0.19 = 0.003723 \\
 \gamma_2(3) &= \alpha_2(3)\beta_2(3) = 0.005569 * 0.1625 = 0.000905 \\
 x_2 &= \operatorname{argmax}_i \{\gamma_2(i)\} = \operatorname{argmax}_i \{\gamma_2(1), \gamma_2(2), \gamma_2(3)\} = s_1 = \text{sunny} \\
 \gamma_3(1) &= \alpha_3(1)\beta_3(1) = 0.005909 * 1 = 0.005909 \\
 \gamma_3(2) &= \alpha_3(2)\beta_3(2) = 0.005092 * 1 = 0.005092 \\
 \gamma_3(3) &= \alpha_3(3)\beta_3(3) = 0.00198 * 1 = 0.00198 \\
 x_3 &= \operatorname{argmax}_i \{\gamma_3(i)\} = \operatorname{argmax}_i \{\gamma_3(1), \gamma_3(2), \gamma_3(3)\} = s_1 = \text{sunny}
 \end{aligned}$$

As a result, the optimal state sequence is $X = \{x_1=\text{rainy}, x_2=\text{sunny}, x_3=\text{sunny}\}$.

The individually optimal criterion $\gamma_t(i)$ does not reflect the whole probability of state sequence X given observation sequence O because it focuses only on how to find out each partially optimal state x_t at each time point t . Thus, the individually optimal procedure is heuristic method. Viterbi algorithm (Rabiner, 1989, p. 264) is alternative method that takes interest in the whole state sequence X by using joint probability $P(X, O | \Delta)$ of state sequence and observation sequence as optimal criterion for determining state sequence X . Let $\delta_t(i)$ be the maximum joint probability of observation sequence O and state $x_t=s_i$ over $t-1$ previous states. The quantity $\delta_t(i)$ is called *joint optimal criterion* at time point t , which is specified by formula IV.4.2.3.

$$\delta_t(i) = \max_{x_1, x_2, \dots, x_{t-1}} (P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_t = s_i | \Delta))$$

Formula IV.4.2.3. Joint optimal criterion at time point t

The recurrence property of *joint optimal criterion* is specified by formula IV.4.2.4.

$$\delta_{t+1}(j) = \left(\max_i (\delta_t(i) a_{ij}) \right) b_j(o_{t+1})$$

Formula IV.4.2.4. Recurrence property of joint optimal criterion

The semantic content of joint optimal criterion δ_t is similar to the forward variable a_t . Following is the proof of formula [IV.4.2.4](#).

$$\begin{aligned}
 \delta_{t+1}(j) &= \max_{x_1, x_2, \dots, x_t} \left(P(o_1, o_2, \dots, o_t, o_{t+1}, x_1, x_2, \dots, x_t, x_{t+1} = s_j | \Delta) \right) \\
 &= \max_{x_1, x_2, \dots, x_t} \left(P(o_{t+1} | o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_t, x_{t+1} = s_j) \right. \\
 &\quad \left. * P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_t, x_{t+1} = s_j) \right) \\
 &\quad \text{(Due to multiplication rule specified by formula [III.1.1.3](#))} \\
 &= \max_{x_1, x_2, \dots, x_t} \left(P(o_{t+1} | x_{t+1} = s_j) * P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_t, x_{t+1} = s_j) \right) \\
 &\quad \text{(Due to observations are mutually independent)} \\
 &= \max_{x_1, x_2, \dots, x_t} \left(b_j(o_{t+1}) * P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_t, x_{t+1} = s_j) \right) \\
 &= \max_{x_1, x_2, \dots, x_t} \left(P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_t, x_{t+1} = s_j) \right) * b_j(o_{t+1}) \\
 &\quad \text{(The probability } b_j(o_{t+1}) \text{ is moved out of the maximum operation because it is} \\
 &\quad \text{independent from states } x_1, x_2, \dots, x_t) \\
 &= \max_{x_1, x_2, \dots, x_t} \left(P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1}, x_{t+1} = s_j | x_t) * P(x_t) \right) * b_j(o_{t+1}) \\
 &\quad \text{(Due to multiplication rule specified by formula [III.1.1.3](#))} \\
 &= \max_{x_1, x_2, \dots, x_t} \left(P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1} | x_{t+1} = s_j, x_t) * P(x_{t+1} = s_j | x_t) * P(x_t) \right) \\
 &\quad * b_j(o_{t+1}) \\
 &\quad \text{(Due to multiplication rule specified by formula [III.1.1.3](#))} \\
 &= \max_{x_1, x_2, \dots, x_t} \left(P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1} | x_t) * P(x_{t+1} = s_j | x_t) * P(x_t) \right) * b_j(o_{t+1}) \\
 &\quad \text{(Because observation } x_{t+1} \text{ is dependent from } o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1}) \\
 &= \max_{x_1, x_2, \dots, x_t} \left(P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1} | x_t) * P(x_t) * P(x_{t+1} = s_j | x_t) \right) * b_j(o_{t+1}) \\
 &= \max_{x_1, x_2, \dots, x_t} \left(P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1}, x_t) * P(x_{t+1} = s_j | x_t) \right) * b_j(o_{t+1}) \\
 &\quad \text{(Due to multiplication rule specified by formula [III.1.1.3](#))} \\
 &= \max_{x_t} \left(\max_{x_1, x_2, \dots, x_{t-1}} \left(P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1}, x_t) * P(x_{t+1} = s_j | x_t) \right) \right) \\
 &\quad * b_j(o_{t+1}) \\
 &= \max_{x_t} \left(\left(\max_{x_1, x_2, \dots, x_{t-1}} P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1}, x_t) \right) * P(x_{t+1} = s_j | x_t) \right) \\
 &\quad * b_j(o_{t+1}) \\
 &= \max_i \left(\left(\max_{x_1, x_2, \dots, x_{t-1}} P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1}, x_t = s_i) \right) \right. \\
 &\quad \left. * P(x_{t+1} = s_j | x_t = s_i) \right) * b_j(o_{t+1}) \\
 &= \max_i \left(\left(\max_{x_1, x_2, \dots, x_{t-1}} P(o_1, o_2, \dots, o_t, x_1, x_2, \dots, x_{t-1}, x_t = s_i) \right) * a_{ij} \right) * b_j(o_{t+1}) \\
 &= \max_i (\delta_t(i) * a_{ij}) * b_j(o_{t+1})
 \end{aligned}$$

$$= \left(\max_i (\delta_t(i) a_{ij}) \right) b_j(o_{t+1})$$

Given criterion $\delta_{t+1}(j)$, the state $x_{t+1}=s_j$ that maximizes $\delta_{t+1}(j)$ is stored in the backtracking state $q_{t+1}(j)$ that is specified by formula IV.4.2.5.

$$q_{t+1}(j) = \operatorname{argmax}_i (\delta_t(i) a_{ij})$$

Formula IV.4.2.5. Backtracking state

Note that index i is identified with state $s_i \in S$ according to formula IV.4.2.5. The Viterbi algorithm based on joint optimal criterion $\delta_t(i)$ includes three steps described in table IV.4.2.2.

1. Initialization step:
 - Initializing $\delta_1(i) = b_i(o_1)\pi_i$ for all $1 \leq i \leq n$
 - Initializing $q_1(i) = 0$ for all $1 \leq i \leq n$
2. Recurrence step:
 - Calculating all $\delta_{t+1}(j) = \left(\max_i (\delta_t(i) a_{ij}) \right) b_j(o_{t+1})$ for all $1 \leq i, j \leq n$ and $1 \leq t \leq T - 1$ according to formula IV.4.2.4.
 - Keeping tracking optimal states $q_{t+1}(j) = \operatorname{argmax}_i (\delta_t(i) a_{ij})$ for all $1 \leq j \leq n$ and $1 \leq t \leq T - 1$ according to formula IV.4.2.5.
3. State sequence backtracking step: The resulted state sequence $X = \{x_1, x_2, \dots, x_T\}$ is determined as follows:
 - The last state $x_T = \operatorname{argmax}_j (\delta_T(j))$
 - Previous states are determined by backtracking: $x_t = q_{t+1}(x_{t+1})$ for $t=T-1, t=T-2, \dots, t=1$.

Table IV.4.2.2. Viterbi algorithm to solve uncovering problem

The total number of operations inside the Viterbi algorithm is $2n + (2n^2 + n)(T-1)$ as follows:

- There are n multiplications for initializing n values $\delta_1(i)$ when each $\delta_1(i)$ requires 1 multiplication.
- There are $(2n^2 + n)(T-1)$ operations over the recurrence step because there are $n(T-1)$ values $\delta_{t+1}(j)$ and each $\delta_{t+1}(j)$ requires n multiplications and n comparisons for maximizing $\max_i (\delta_t(i) a_{ij})$ plus 1 multiplication.
- There are n comparisons for constructing the state sequence X , $x_T = \operatorname{argmax}_j (\delta_T(j))$.

Inside $2n + (2n^2 + n)(T-1)$ operations, there are $n + (n^2 + n)(T-1)$ multiplications and $n^2(T-1) + n$ comparisons. The number of operations with regard to Viterbi algorithm is smaller than the number of operations with regard to individually optimal procedure when individually optimal procedure requires $(5n^2 - n)(T-1) + 2nT + n$ operations. Therefore, Viterbi algorithm is more effective than individually optimal procedure. Besides, individually optimal procedure does not reflect the whole probability of state sequence X given observation sequence O .

Going back to the weather HMM Δ whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3. Suppose humidity is *soggy* and *dry* in days 1 and

2, respectively. We apply Viterbi algorithm into solving the uncovering problem that finding out the optimal state sequence $X = \{x_1, x_2, x_3\}$ with regard to observation sequence $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$. According to initialization step of Viterbi algorithm, we have:

$$\delta_1(1) = b_1(o_1 = \varphi_4)\pi_1 = b_{14}\pi_1 = 0.05 * 0.33 = 0.0165$$

$$\delta_1(2) = b_2(o_1 = \varphi_4)\pi_2 = b_{24}\pi_2 = 0.25 * 0.33 = 0.0825$$

$$\delta_1(3) = b_3(o_1 = \varphi_4)\pi_3 = b_{34}\pi_3 = 0.5 * 0.33 = 0.165$$

$$q_1(1) = q_1(2) = q_1(3) = 0$$

According to recurrence step of Viterbi algorithm, we have:

$$\delta_1(1)a_{11} = 0.0165 * 0.5 = 0.00825$$

$$\delta_1(2)a_{21} = 0.0825 * 0.3 = 0.02475$$

$$\delta_1(3)a_{31} = 0.165 * 0.25 = 0.04125$$

$$\begin{aligned} \delta_2(1) &= \left(\max_i \{\delta_1(i)a_{i1}\} \right) b_1(o_2 = \varphi_1) = \left(\max_i \{\delta_1(i)a_{i1}\} \right) b_{11} \\ &= \left(\max_i \{\delta_1(1)a_{11}, \delta_1(2)a_{21}, \delta_1(3)a_{31}\} \right) b_{11} = 0.04125 * 0.6 \\ &= 0.02475 \end{aligned}$$

$$q_2(1) = \operatorname{argmax}_i \{\delta_1(i)a_{i1}\} = \operatorname{argmax}_i \{\delta_1(1)a_{11}, \delta_1(2)a_{21}, \delta_1(3)a_{31}\} = s_3 = rainy$$

$$\delta_1(1)a_{12} = 0.0165 * 0.25 = 0.004125$$

$$\delta_1(2)a_{22} = 0.0825 * 0.4 = 0.033$$

$$\delta_1(3)a_{32} = 0.165 * 0.25 = 0.04125$$

$$\begin{aligned} \delta_2(2) &= \left(\max_i \{\delta_1(i)a_{i2}\} \right) b_2(o_2 = \varphi_1) = \left(\max_i \{\delta_1(i)a_{i2}\} \right) b_{21} \\ &= \left(\max_i \{\delta_1(1)a_{12}, \delta_1(2)a_{22}, \delta_1(3)a_{32}\} \right) b_{21} = 0.04125 * 0.25 \\ &= 0.010313 \end{aligned}$$

$$q_2(2) = \operatorname{argmax}_i \{\delta_1(i)a_{i2}\} = \operatorname{argmax}_i \{\delta_1(1)a_{12}, \delta_1(2)a_{22}, \delta_1(3)a_{32}\} = s_3 = rainy$$

$$\delta_1(1)a_{13} = 0.0165 * 0.25 = 0.004125$$

$$\delta_1(2)a_{23} = 0.0825 * 0.3 = 0.02475$$

$$\delta_1(3)a_{33} = 0.165 * 0.5 = 0.0825$$

$$\begin{aligned} \delta_2(3) &= \left(\max_i \{\delta_1(i)a_{i3}\} \right) b_3(o_2 = \varphi_1) = \left(\max_i \{\delta_1(i)a_{i3}\} \right) b_{31} \\ &= \left(\max_i \{\delta_1(1)a_{13}, \delta_1(2)a_{23}, \delta_1(3)a_{33}\} \right) b_{31} = 0.0825 * 0.05 \\ &= 0.004125 \end{aligned}$$

$$q_2(3) = \operatorname{argmax}_i \{\delta_1(i)a_{i3}\} = \operatorname{argmax}_i \{\delta_1(1)a_{13}, \delta_1(2)a_{23}, \delta_1(3)a_{33}\} = s_3 = rainy$$

$$\delta_2(1)a_{11} = 0.02475 * 0.5 = 0.012375$$

$$\delta_2(2)a_{21} = 0.010313 * 0.3 = 0.003094$$

$$\delta_2(3)a_{31} = 0.004125 * 0.25 = 0.001031$$

$$\begin{aligned} \delta_3(1) &= \left(\max_i \{\delta_2(i)a_{i1}\} \right) b_1(o_3 = \varphi_2) = \left(\max_i \{\delta_2(i)a_{i1}\} \right) b_{12} \\ &= \left(\max_i \{\delta_2(1)a_{11}, \delta_2(2)a_{21}, \delta_2(3)a_{31}\} \right) b_{12} = 0.012375 * 0.2 \\ &= 0.002475 \end{aligned}$$

$$q_3(1) = \operatorname{argmax}_i \{\delta_2(i)a_{i1}\} = \operatorname{argmax}_i \{\delta_2(1)a_{11}, \delta_2(2)a_{21}, \delta_2(3)a_{31}\} = s_1 = sunny$$

$$\delta_2(1)a_{12} = 0.02475 * 0.25 = 0.006188$$

$$\delta_2(2)a_{22} = 0.010313 * 0.4 = 0.004125$$

$$\delta_2(3)a_{32} = 0.004125 * 0.25 = 0.001031$$

$$\begin{aligned}\delta_3(2) &= \left(\max_i \{\delta_2(i)a_{i2}\} \right) b_2(o_3 = \varphi_2) = \left(\max_i \{\delta_2(i)a_{i2}\} \right) b_{22} \\ &= \left(\max_i \{\delta_2(1)a_{12}, \delta_2(2)a_{22}, \delta_2(3)a_{32}\} \right) b_{22} = 0.006188 * 0.25 \\ &= 0.001547\end{aligned}$$

$$q_3(2) = \operatorname{argmax}_i \{\delta_2(i)a_{i2}\} = \operatorname{argmax}_i \{\delta_2(1)a_{12}, \delta_2(2)a_{22}, \delta_2(3)a_{32}\} = s_1 = \text{sunny}$$

$$\delta_2(1)a_{13} = 0.02475 * 0.25 = 0.006188$$

$$\delta_2(2)a_{23} = 0.010313 * 0.3 = 0.003094$$

$$\delta_2(3)a_{33} = 0.004125 * 0.5 = 0.002063$$

$$\begin{aligned}\delta_3(3) &= \left(\max_i \{\delta_2(i)a_{i3}\} \right) b_3(o_3 = \varphi_2) = \left(\max_i \{\delta_2(i)a_{i3}\} \right) b_{32} \\ &= \left(\max_i \{\delta_2(1)a_{13}, \delta_2(2)a_{23}, \delta_2(3)a_{33}\} \right) b_{32} = 0.006188 * 0.1 \\ &\approx 0.000618\end{aligned}$$

$$q_3(3) = \operatorname{argmax}_i \{\delta_2(i)a_{i3}\} = \operatorname{argmax}_i \{\delta_2(1)a_{13}, \delta_2(2)a_{23}, \delta_2(3)a_{33}\} = s_1 = \text{sunny}$$

According to state sequence backtracking of Viterbi algorithm, we have:

$$x_3 = \operatorname{argmax}_j \{\delta_3(j)\} = \operatorname{argmax}_j \{\delta_3(1), \delta_3(2), \delta_3(3)\} = s_1 = \text{sunny}$$

$$x_2 = q_3(x_3 = s_1) = q_3(1) = s_1 = \text{sunny}$$

$$x_1 = q_2(x_2 = s_1) = q_2(1) = s_3 = \text{rainy}$$

As a result, the optimal state sequence is $X = \{x_1=\text{rainy}, x_2=\text{sunny}, x_3=\text{sunny}\}$. The result from the Viterbi algorithm is the same to the one from aforementioned individually optimal procedure described in table IV.4.2.1.

Essentially, Viterbi algorithm maximizes the joint probability $P(X, O|\Delta)$ instead of maximizing the conditional probability $P(X|O, \Delta)$. I propose so-called *longest-path algorithm* based on longest path of graph for solving uncovering problem. This algorithm that maintains using the conditional probability $P(X|O, \Delta)$ as optimal criterion gives a viewpoint different from the viewpoint of Viterbi algorithm although it is easy for you to recognize that the ideology of the longest-path algorithm does not go beyond the ideology of Viterbi algorithm after you comprehend the longest-path algorithm. Following is description of longest-path algorithm.

The optimal criterion $P(X|O, \Delta)$ of graphic method is:

$$\begin{aligned}P(X|O, \Delta) &= P(x_1, x_2, \dots, x_T | o_1, o_2, \dots, o_T) \\ &= P(x_1, x_2, \dots, x_{T-1}, x_T | o_1, o_2, \dots, o_{T-1}, o_T) \\ &= P(x_T | x_1, x_2, \dots, x_{T-1}, o_1, o_2, \dots, o_{T-1}, o_T) * P(x_1, x_2, \dots, x_{T-1} | o_1, o_2, \dots, o_{T-1}, o_T) \\ &\quad (\text{Due to multiplication rule specified by formula III.1.1.3}) \\ &= P(x_T | x_{T-1}, o_1, o_2, \dots, o_{T-1}, o_T) * P(x_1, x_2, \dots, x_{T-1} | o_1, o_2, \dots, o_{T-1}, o_T)\end{aligned}$$

(Due to Markov property: the probability of current state is only dependent on the probability of right previous state)

$$= P(x_T | x_{T-1}, o_T) * P(x_1, x_2, \dots, x_{T-1} | o_1, o_2, \dots, o_{T-1})$$

(Because an observation is only dependent on the time point when it is observed)

By recurrence calculation on probability $P(x_1, x_2, \dots, x_{T-1} | o_1, o_2, \dots, o_{T-1})$, we have:

$$\begin{aligned}P(X|O, \Delta) &= P(x_1, x_2, \dots, x_T | o_1, o_2, \dots, o_T) \\ &= P(x_1 | o_1) P(x_2 | x_1, o_2) \dots P(x_t | x_{t-1}, o_t) \dots P(x_T | x_{T-1}, o_T)\end{aligned}$$

Applying Bayes' rule (formula III.1.1.1) into the probability $P(x_{t-1}, x_t | o_t)$, we have:

$$\begin{aligned}
 P(x_{t-1}, x_t | o_t) &= \frac{P(x_t | x_{t-1}, o_t) P(x_{t-1} | o_t)}{P(x_t | o_t)} = P(x_t | x_{t-1}, o_t) \frac{1}{P(x_t | o_t)} P(x_{t-1} | o_t) \\
 &= P(x_t | x_{t-1}, o_t) \frac{1}{P(x_t | o_t)} \frac{P(o_t | x_{t-1}) P(x_{t-1})}{P(o_t)} \\
 &\quad (\text{Applying Bayes' rule into the probability } P(x_{t-1} | o_t)) \\
 &= P(x_t | x_{t-1}, o_t) \frac{P(o_t)}{P(o_t | x_t) P(x_t)} \frac{P(o_t | x_{t-1}) P(x_{t-1})}{P(o_t)} \\
 &\quad (\text{Applying Bayes' rule into the probability } P(x_t | o_t)) \\
 &= \frac{P(x_t | x_{t-1}, o_t) P(o_t | x_{t-1}) P(x_{t-1})}{P(o_t | x_t) P(x_t)} = \frac{P(x_t | x_{t-1}, o_t) P(o_t) P(x_{t-1})}{P(o_t | x_t) P(x_t)}
 \end{aligned}$$

(Because an observation is only dependent on the time point when it is observed,
 $P(o_t | x_{t-1}) = P(o_t)$)

Applying Bayes' rule into the probability $P(x_{t-1}, x_t | o_t)$ by another way, we have:

$$\begin{aligned}
 P(x_{t-1}, x_t | o_t) &= \frac{P(o_t | x_{t-1}, x_t) P(x_{t-1}, x_t)}{P(o_t)} \\
 &= \frac{P(o_t | x_t) P(x_{t-1}, x_t)}{P(o_t)} \\
 &\quad (\text{Because an observation is only dependent on the time point when it is observed, } \\
 &\quad P(o_t | x_{t-1}, x_t) = P(o_t | x_t)) \\
 &= \frac{P(o_t | x_t) P(x_t | x_{t-1}) P(x_{t-1})}{P(o_t)} \\
 &\quad (\text{Applying multiplication rule into the probability } P(x_t | o_t))
 \end{aligned}$$

Because we had $P(x_{t-1}, x_t | o_t) = \frac{P(x_t | x_{t-1}, o_t) P(o_t) P(x_{t-1})}{P(o_t | x_t) P(x_t)}$, it implies that

$$\begin{aligned}
 \frac{P(x_t | x_{t-1}, o_t) P(o_t) P(x_{t-1})}{P(o_t | x_t) P(x_t)} &= \frac{P(o_t | x_t) P(x_t | x_{t-1}) P(x_{t-1})}{P(o_t)} \\
 \Rightarrow P(x_t | x_{t-1}, o_t) &= \frac{(P(o_t | x_t))^2 P(x_t | x_{t-1}) P(x_t)}{(P(o_t))^2}
 \end{aligned}$$

We have

$$\begin{aligned}
 P(X | O, \Delta) &= P(x_1, x_2, \dots, x_T | o_1, o_2, \dots, o_T) \\
 &= P(x_1 | o_1) P(x_2 | x_1, o_2) \dots P(x_{t-1}, x_t | o_t) \dots P(x_T | x_{T-1}, o_T) \\
 &= \frac{P(o_1 | x_1) P(x_1)}{P(o_1)} * \frac{(P(o_2 | x_2))^2 P(x_2 | x_1) P(x_2)}{(P(o_2))^2} * \dots * \frac{(P(o_t | x_t))^2 P(x_t | x_{t-1}) P(x_t)}{(P(o_t))^2} \\
 &\quad * \dots * \frac{(P(o_T | x_T))^2 P(x_T | x_{T-1}) P(x_T)}{(P(o_T))^2} \\
 &= \frac{P(o_1)}{(P(o_1))^2 (P(o_2))^2 \dots (P(o_t))^2 \dots (P(o_T))^2} * P(o_1 | x_1) P(x_1) \\
 &\quad * (P(o_2 | x_2))^2 P(x_2 | x_1) P(x_2) * \dots * (P(o_t | x_t))^2 P(x_t | x_{t-1}) P(x_t) * \dots \\
 &\quad * (P(o_T | x_T))^2 P(x_T | x_{T-1}) P(x_T) \\
 &= c w_1 w_2 \dots w_t \dots w_T
 \end{aligned}$$

Where,

$$\begin{cases} c \text{ is constant, } c = \frac{P(o_1)}{(P(o_1))^2 (P(o_2))^2 \cdots (P(o_t))^2 \cdots (P(o_T))^2} \\ w_1 = P(o_1|x_1)P(x_1) \text{ when } t = 1 \\ w_t = (P(o_t|x_t))^2 P(x_t|x_{t-1})P(x_t), \forall 1 < t \leq T \end{cases}$$

Because the constant c is independent from state transitions, maximizing the criterion $P(X|O, \Delta)$ with regard to state transitions is the same to maximizing the product $w_1 w_2 \dots w_t \dots w_T$. Let ρ be this product and so, ρ is the optimal criterion of longest-path algorithm, re-written by formula IV.4.2.6.

$$\rho = w_1 w_2 \dots w_t \dots w_T$$

Formula IV.4.2.6. Optimal criterion of longest-path algorithm

Where,

$$\begin{cases} w_1 = P(o_1|x_1)P(x_1) \text{ when } t = 1 \\ w_t = (P(o_t|x_t))^2 P(x_t|x_{t-1})P(x_t), \forall 1 < t \leq T \end{cases}$$

The essence of longest-path algorithm is to construct a graph and then, the algorithm finds out the longest path inside such graph with attention that the optimal criterion ρ represents length of every path inside the graph. There is an interesting thing that such length ρ is product of weights instead of sequence of additions as usual. The criterion ρ is function of state transitions and the longest-path algorithm aims to maximize ρ . Following is description of how to build up the graph.

Each w_t representing the influence of state x_t on the observation sequence $O = \{o_1, o_2, \dots, o_T\}$ at time point t is dependent on states x_{t-1} and x_t . We will create a graph from these w_t (s). Because there are n possible values of x_t , the state x_t is decomposed into n nodes $X_{t1}, X_{t2}, \dots, X_{tn}$. There are T time points, we have nT time nodes. Let $X = \{X_0, X_{11}, X_{12}, \dots, X_{1n}, X_{21}, X_{22}, \dots, X_{2n}, \dots, X_{T1}, X_{T2}, \dots, X_{Tn}\}$ be a set of $1+nT$ nodes where X_0 is null node. Firstly, we create n weighted arcs from node X_0 to n nodes $X_{11}, X_{12}, \dots, X_{1n}$ at the first time point. These directed arcs are denoted $W_{0111}, W_{0112}, \dots, W_{011n}$ and their weights are also denoted $W_{0111}, W_{0112}, \dots, W_{011n}$. These weights W_{011j} (s) at the first time point are calculated according to w_1 (see formula IV.4.2.6). Formula IV.4.2.7 determines W_{1j} (s).

$$W_{011j} = P(o_1|x_1 = s_j)P(x_1 = s_j) = b_j(o_1)\pi_j$$

$$\forall j = \overline{1, n}$$

Formula IV.4.2.7. Weights at the first time point

Your attention please, it is conventional that W_{0i1j} is equal to W_{011j} , $\forall i = \overline{1, n}$ because the null node X_0 has no state.

$$W_{0i1j} = W_{011j}, \forall i = \overline{1, n}$$

Moreover, these weights W_{011j} (s) are depicted by figure IV.4.2.1.

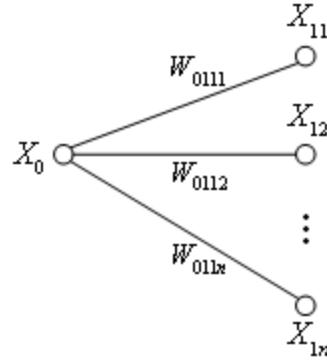


Figure IV.4.2.1. Weighted arcs from null node X_0 to n nodes $X_{11}, X_{12}, \dots, X_{1n}$

For example, given weather HMM Δ whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3, suppose observation sequence is $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$, we have 3 weights at the initial time point as follows:

$$W_{0111} = b_1(o_1 = \varphi_4)\pi_1 = b_{14}\pi_1$$

$$W_{0112} = b_2(o_1 = \varphi_4)\pi_2 = b_{24}\pi_2$$

$$W_{0113} = b_3(o_1 = \varphi_4)\pi_3 = b_{34}\pi_3$$

For each node $X_{(t-1)i}$ where $t > 1$, we create n weighted arcs from node $X_{(t-1)i}$ to n nodes $X_{t1}, X_{t2}, \dots, X_{tn}$ at the time point t . These directed arcs are denoted $W_{(t-1)it1}, W_{(t-1)it2}, \dots, W_{(t-1)itm}$ and their weights are also denoted $W_{(t-1)it1}, W_{(t-1)it2}, \dots, W_{(t-1)itm}$. These weights $W_{(t-1)itj}$ at the time point t are calculated according to w_t (see formula IV.4.2.6). Formula IV.4.2.8 determines $W_{(t-1)itj}$.

$$W_{(t-1)itj} = \left(P(o_t | x_t = s_j) \right)^2 P(x_t = s_j | x_{t-1} = s_i) P(x_t = s_j) = \left(b_j(o_t) \right)^2 a_{ij} \pi_j \quad \forall i, j = \overline{1, n}$$

Formula IV.4.2.8. Weights at the time point t

Moreover, these weights $W_{(t-1)itj}$ (s) are depicted by figure IV.4.2.2.

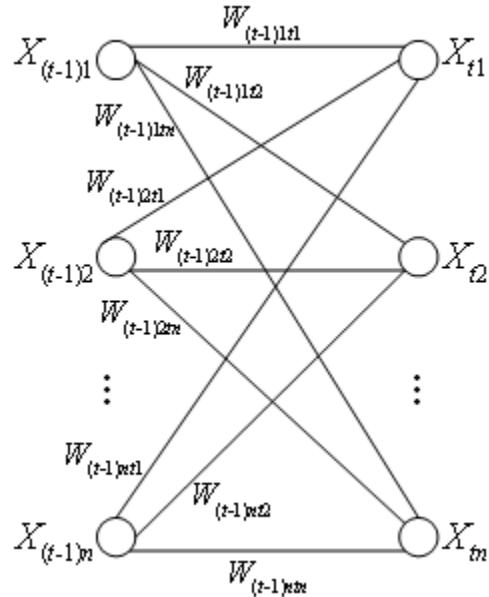


Figure IV.4.2.2. Weighted arcs from n nodes $X_{(t-1)i}$ to n nodes X_{tj} at time point t

Going back given weather HMM Δ whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3, suppose observation sequence is $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$, we have 18 weights from time point 1 to time point 3 as follows:

$$W_{1121} = (b_1(o_2 = \varphi_1))^2 a_{11}\pi_1 = (b_{11})^2 a_{11}\pi_1$$

$$W_{1122} = (b_2(o_2 = \varphi_1))^2 a_{12}\pi_2 = (b_{21})^2 a_{12}\pi_2$$

$$W_{1123} = (b_3(o_2 = \varphi_1))^2 a_{13}\pi_3 = (b_{31})^2 a_{13}\pi_3$$

$$W_{1221} = (b_1(o_2 = \varphi_1))^2 a_{21}\pi_1 = (b_{11})^2 a_{21}\pi_1$$

$$W_{1222} = (b_2(o_2 = \varphi_1))^2 a_{22}\pi_2 = (b_{21})^2 a_{22}\pi_2$$

$$W_{1223} = (b_3(o_2 = \varphi_1))^2 a_{23}\pi_3 = (b_{31})^2 a_{23}\pi_3$$

$$W_{1321} = (b_1(o_2 = \varphi_1))^2 a_{31}\pi_1 = (b_{11})^2 a_{31}\pi_1$$

$$W_{1322} = (b_2(o_2 = \varphi_1))^2 a_{32}\pi_2 = (b_{21})^2 a_{32}\pi_2$$

$$W_{1323} = (b_3(o_2 = \varphi_1))^2 a_{33}\pi_3 = (b_{31})^2 a_{33}\pi_3$$

$$W_{2131} = (b_1(o_3 = \varphi_2))^2 a_{11}\pi_1 = (b_{12})^2 a_{11}\pi_1$$

$$W_{2132} = (b_2(o_3 = \varphi_2))^2 a_{12}\pi_2 = (b_{22})^2 a_{12}\pi_2$$

$$W_{2133} = (b_3(o_3 = \varphi_2))^2 a_{13}\pi_3 = (b_{32})^2 a_{13}\pi_3$$

$$W_{2231} = (b_1(o_3 = \varphi_2))^2 a_{21}\pi_1 = (b_{12})^2 a_{21}\pi_1$$

$$W_{2232} = (b_2(o_3 = \varphi_2))^2 a_{22}\pi_2 = (b_{22})^2 a_{22}\pi_2$$

$$W_{2233} = (b_3(o_3 = \varphi_2))^2 a_{23}\pi_3 = (b_{32})^2 a_{23}\pi_3$$

$$W_{2331} = (b_1(o_3 = \varphi_2))^2 a_{31}\pi_1 = (b_{12})^2 a_{31}\pi_1$$

$$W_{2332} = (b_2(o_3 = \varphi_2))^2 a_{32}\pi_2 = (b_{22})^2 a_{32}\pi_2$$

$$W_{2333} = (b_3(o_3 = \varphi_2))^2 a_{33}\pi_3 = (b_{32})^2 a_{33}\pi_3$$

In general, there are $(T-1)n^2$ weights from time point 1 to time point T . Moreover, there are n weights derived from null node X_0 at time point 1. Let W be set of $n+(T-1)n^2$ weights from null node X_0 to nodes $X_{T1}, X_{T2}, \dots, X_{Tn}$ at the last time point T . Let $G = \langle X, W \rangle$ be the graph consisting of the set of nodes $X = \{X_0, X_{11}, X_{12}, \dots, X_{1n}, X_{21}, X_{22}, \dots, X_{2n}, \dots, X_{T1}, X_{T2}, \dots, X_{Tn}\}$ be a set of $n+(T-1)n^2$ weights W . The graph G is called *state transition graph* shown in figure IV.4.2.3.

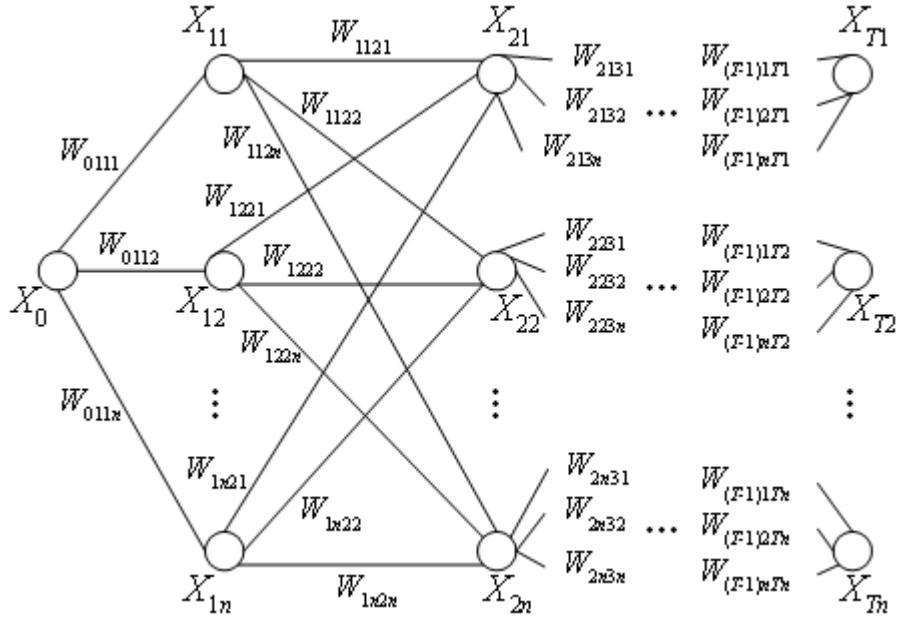


Figure IV.4.2.3. State transition graph

Please pay attention to a very important thing that both graph G and its weights are not determined before the longest-path algorithm is executed because there are a huge number of nodes and arcs. State transition graph shown in figure IV.4.2.3 is only illustrative example. Going back given weather HMM Δ whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3, suppose observation sequence is $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$, the state transition graph of this weather example is shown in figure IV.4.2.4.

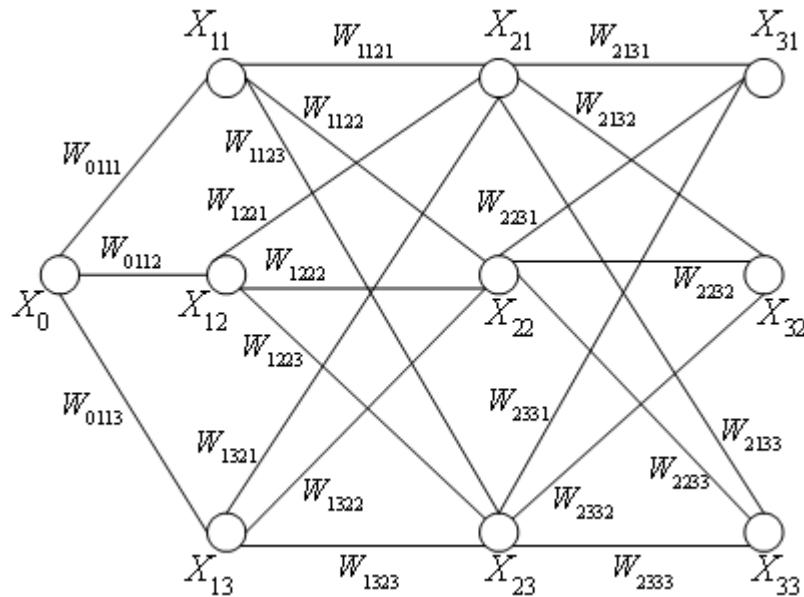


Figure IV.4.2.4. State transition graph of weather example

The ideology of the longest-path algorithm is to solve uncovering problem by finding the longest path of state transition graph where the whole length of every path is represented by the optimal criterion ρ (see formula IV.4.2.6). In other words, the longest-path algorithm maximizes the optimal criterion ρ by finding the longest path. Let $X = \{x_1, x_2, \dots, x_T\}$ be the longest path of state transition graph and so, length of X

is maximum value of the path length ρ . The path length ρ is calculated as product of weights $W_{(t-1)itj}$ (s). By heuristic assumption, ρ is maximized locally by maximizing weights $W_{(t-1)itj}$ (s) at each time point. The longest-path algorithm is described by pseudo-code shown in table IV.4.2.3 with note that X is state sequence that is ultimate result of the longest-path algorithm.

X is initialized to be empty, $X = \emptyset$.

Calculating initial weights $W_{0111}, W_{0112}, \dots, W_{011n}$ according to formula IV.4.2.7.

$j = \underset{k}{\operatorname{argmax}}\{W_{011k}\}$ where $k = \overline{1, n}$

Adding state $x_1=s_j$ to the longest path: $X = X \cup \{x_1 = s_j\}$

For $t = 2$ to T

Calculating n weights $W_{(t-1)jt1}, W_{(t-1)jt2}, \dots, W_{(t-1)jtn}$ according to formula IV.4.2.8.

$j = \underset{k}{\operatorname{argmax}}\{W_{(t-1)jtk}\}$ where $k = \overline{1, n}$

Adding state $x_t=s_j$ to the longest path: $X = X \cup \{x_t = s_j\}$

End for

Table IV.4.2.3. Longest-path algorithm

The total number of operations inside the longest-path algorithm is $2n+4n(T-1)$ as follows:

- There are n multiplications for initializing n weights $W_{0111}, W_{0112}, \dots, W_{011n}$ when each weight W_{011j} requires 1 multiplication. There are n comparisons due to finding maximum weight index $j = \underset{k}{\operatorname{argmax}}\{W_{011k}\}$.
- There are $3n(T-1)$ multiplications over the loop inside the algorithm because there are $n(T-1)$ weights $W_{(t-1)jtk}$ over the loop and each $W_{(t-1)jtk}$ requires 3 multiplications. There are $n(T-1)$ comparisons over the loop inside the algorithm due to finding maximum weight indices: $j = \underset{k}{\operatorname{argmax}}\{W_{(t-1)jtk}\}$.

Inside $2n+4n(T-1)$ operations, there are $n+3n(T-1)$ multiplications and $n+n(T-1)$ comparisons.

The longest-path algorithm is similar to Viterbi algorithm (see table IV.4.2.2) with regard to the aspect that the path length ρ is calculated accumulatively but computational formulas and viewpoints of longest-path algorithm and Viterbi algorithm are different. The longest-path algorithm is more effective than Viterbi algorithm because it requires $2n+4n(T-1)$ operations while Viterbi algorithm executes $2n+(2n^2+n)(T-1)$ operations. However, longest-path algorithm does not produce the most accurate result because the path length ρ is maximized locally by maximizing weights $W_{(t-1)itj}$ (s) at each time point, which leads that the resulted sequence X may not be global longest path. In general, the longest-path algorithm is heuristic algorithm that gives a new viewpoint of uncovering problem when applying graphic approach into solving uncovering problem.

Going back given weather HMM Δ whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3, suppose observation sequence is $O = \{o_1=\varphi_4=\text{soggy}, o_2=\varphi_1=\text{dry}, o_3=\varphi_2=\text{dryish}\}$, the longest-path algorithm is applied to find out the optimal state sequence $X = \{x_1, x_2, x_3\}$ as below.

At the first time point, we have:

$$W_{0111} = b_{14}\pi_1 = 0.05 * 0.33 = 0.0165$$

$$W_{0112} = b_{24}\pi_2 = 0.25 * 0.33 = 0.0825$$

$$W_{0113} = b_{34}\pi_3 = 0.5 * 0.33 = 0.165$$

$$j = \underset{k}{\operatorname{argmax}}\{W_{011k}\} = \underset{k}{\operatorname{argmax}}\{W_{0111}, W_{0112}, W_{0113}\} = 3$$

$$X = X \cup \{x_1 = s_j\} = X \cup \{x_1 = s_3\} = \{x_1 = \text{rainy}\}$$

At the second time point, we have:

$$W_{1j21} = W_{1321} = (b_{11})^2 a_{31}\pi_1 = 0.6^2 * 0.25 * 0.33 = 0.0297$$

$$W_{1j22} = W_{1322} = (b_{21})^2 a_{32}\pi_2 = 0.25^2 * 0.25 * 0.33 = 0.005156$$

$$W_{1j23} = W_{1323} = (b_{31})^2 a_{33}\pi_3 = 0.05^2 * 0.5 * 0.33 = 0.000413$$

$$j = \underset{k}{\operatorname{argmax}}\{W_{132k}\} = \underset{k}{\operatorname{argmax}}\{W_{1321}, W_{1322}, W_{1323}\} = 1$$

$$X = X \cup \{x_2 = s_j\} = X \cup \{x_2 = s_1\} = \{x_1 = \text{rainy}, x_2 = \text{sunny}\}$$

At the third time point, we have:

$$W_{2j31} = W_{2131} = (b_{12})^2 a_{11}\pi_1 = 0.2^2 * 0.5 * 0.33 = 0.0066$$

$$W_{2j32} = W_{2132} = (b_{22})^2 a_{12}\pi_2 = 0.25^2 * 0.25 * 0.33 = 0.005156$$

$$W_{2j33} = W_{2133} = (b_{32})^2 a_{13}\pi_3 = 0.1^2 * 0.25 * 0.33 = 0.000825$$

$$j = \underset{k}{\operatorname{argmax}}\{W_{213k}\} = \underset{k}{\operatorname{argmax}}\{W_{2131}, W_{2132}, W_{2133}\} = 1$$

$$X = X \cup \{x_3 = s_j\} = X \cup \{x_3 = s_1\} = \{x_1 = \text{rainy}, x_2 = \text{sunny}, x_3 = \text{sunny}\}$$

As a result, the optimal state sequence is $X = \{x_1=\text{rainy}, x_2=\text{sunny}, x_3=\text{sunny}\}$. The result from the longest-path algorithm in this example is the same to the one from individually optimal procedure (see table IV.4.2.1) and Viterbi algorithm (see table IV.4.2.2).

The longest-path algorithm does not result out accurate state sequence X because it assumes that two successive nodes $X_{(t-1)i}$ and X_{tj} are mutually independent, which leads that the path length ρ is maximized locally by maximizing weight $W_{(t-1)itj}$ at each time point, while formula IV.4.2.6 indicates that the former node $X_{(t-1)i}$ is dependent on the prior node X_{tj} . However, according to Markov property, two intermittent nodes $X_{(t-1)i}$ and $X_{(t+1)k}$ are conditional independent given the middle node X_{tj} . This observation is very important, which helps us to enhance the accuracy of longest-path algorithm. The advanced longest-path algorithm divides the path represented by ρ into a set of 2-weight intervals. Each 2-weight interval includes two successive weights $W_{(t-1)itj}$ and $W_{ti(t+1)j}$ corresponding three nodes $X_{(t-1)i}$, X_{tj} , and $X_{(t+1)k}$ where the middle node X_{tj} is also called the *midpoint* of 2-weight interval. The advanced longest-path algorithm maximizes the path ρ by maximizing every 2-weight interval. Each 2-weight interval has $2n^2$ connections (sub-paths) because each weight $W_{(t-1)itj}$ or $W_{ti(t+1)j}$ has n^2 values. Figure IV.4.2.5 depicts an example of 2-weight interval.

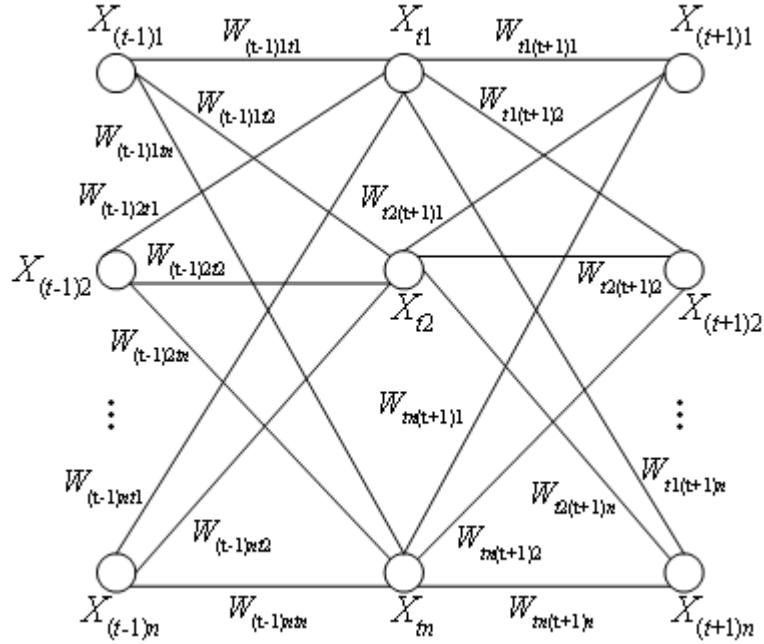


Figure IV.4.2.5. 2-weight interval

The advanced longest-path algorithm is described by pseudo-code shown in table IV.4.2.4.

X is initialized to be empty, $X = \emptyset$.

$i = 1$

For $t = 1$ to T step 2

// Note that time point t is increased by 2 as follows: 1, 3, 5, ...

Calculating n weights $W_{(t-1)it1}, W_{(t-1)it2}, \dots, W_{(t-1)itn}$ according to formulas IV.4.2.7 and IV.4.2.8.

For $j = 1$ to n

Calculating n weights $W_{tj(t+1)1}, W_{tj(t+1)2}, \dots, W_{tj(t+1)n}$ according to formula IV.4.2.8.

$$k_j = \operatorname{argmax}_l \{W_{tj(t+1)l}\}$$

End for

$$u = \operatorname{argmax}_j (W_{(t-1)itj} W_{tj(t+1)k_j})$$

Adding two states $x_t = s_u$ and $x_{t+1} = s_{k_u}$ to the longest path: $X = X \cup \{x_t = s_u\} \cup \{x_{t+1} = s_{k_u}\}$

$$i = k_u$$

End for

Table IV.4.2.4. Advanced longest-path algorithm

Because two intermittent nodes $X_{(t-1)i}$ and $X_{(t+1)k}$ that are two end-points of a 2-weight interval are conditional independent given the midpoint X_j , the essence of advanced longest-path algorithm is to adjust the midpoint of 2-weight interval so as to maximize such 2-weight interval.

The total number of operations inside the longest-path algorithm is $(2n^2+1.5n)T$ as follows:

- There are n multiplications for determining weights $W_{(t-1)it1}, W_{(t-1)it2}, \dots, W_{(t-1)itn}$. Shortly, there are $nT/2 = 0.5nT$ multiplications over the whole algorithm because time point is increased by 2.
- There are $3n^2$ multiplications for determining n^2 weights $W_{tj(t+1)l}$ (s) at each time point when each weight requires 3 multiplications. There are n multiplications for determining product $W_{(t-1)itj}W_{tj(t+1)k_j}$. Shortly, there are $(3n^2+n)T/2 = (1.5n^2+0.5n)T$ multiplications over the whole algorithm because time point is increased by 2.
- There are n^2+n comparisons for maximizing: $\underset{l}{\operatorname{argmax}}\{W_{tj(t+1)l}\}$ and $\underset{j,k}{\operatorname{argmax}}(W_{(t-1)itj}W_k)$. Shortly, there are $(n^2+n)T/2 = (0.5n^2+0.5n)T$ multiplications over the whole algorithm because time point is increased by 2.

Inside $(2n^2+1.5n)T$ operations, there are $(1.5n^2+n)T$ multiplications and $(0.5n^2+0.5n)T$ comparisons. The advanced longest-path algorithm is not more effective than Viterbi algorithm because it requires $(2n^2+1.5n)T$ operations while Viterbi algorithm executes $2n+(2n^2+n)(T-1)$ operations but it is more accurate than normal longest-path algorithm aforementioned in table IV.4.2.3.

Going back given weather HMM Δ whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3, suppose observation sequence is $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$, the advanced longest-path algorithm is applied to find out the optimal state sequence $X = \{x_1, x_2, x_3\}$ as follows:

At $t=1$, we have:

$$W_{0111} = b_{14}\pi_1 = 0.05 * 0.33 = 0.0165$$

$$W_{0112} = b_{24}\pi_2 = 0.25 * 0.33 = 0.0825$$

$$W_{0113} = b_{34}\pi_3 = 0.5 * 0.33 = 0.165$$

$$W_{1121} = (b_{11})^2 a_{11}\pi_1 = 0.6^2 * 0.5 * 0.33 = 0.0594$$

$$W_{1122} = (b_{21})^2 a_{12}\pi_2 = 0.25^2 * 0.25 * 0.33 = 0.005156$$

$$W_{1123} = (b_{31})^2 a_{13}\pi_3 = 0.05^2 * 0.25 * 0.33 = 0.000206$$

$$k_1 = \underset{l}{\operatorname{argmax}}\{W_{112l}\} = \underset{l}{\operatorname{argmax}}\{W_{1121}, W_{1122}, W_{1123}\} = 1$$

$$W_{0111}W_{112k_1} = W_{0111}W_{1121} = 0.0165 * 0.0594 = 0.00098$$

$$W_{1221} = (b_{11})^2 a_{21}\pi_1 = 0.6^2 * 0.3 * 0.33 = 0.03564$$

$$W_{1222} = (b_{21})^2 a_{22}\pi_2 = 0.25^2 * 0.4 * 0.33 = 0.00825$$

$$W_{1223} = (b_{31})^2 a_{23}\pi_3 = 0.05^2 * 0.3 * 0.33 = 0.000248$$

$$k_2 = \underset{l}{\operatorname{argmax}}\{W_{122l}\} = \underset{l}{\operatorname{argmax}}\{W_{1221}, W_{1222}, W_{1223}\} = 1$$

$$W_{0112}W_{122k_2} = W_{0112}W_{1221} = 0.0825 * 0.03564 = 0.00294$$

$$W_{1321} = (b_{11})^2 a_{31}\pi_1 = 0.6^2 * 0.25 * 0.33 = 0.0297$$

$$W_{1322} = (b_{21})^2 a_{32}\pi_2 = 0.25^2 * 0.25 * 0.33 = 0.005156$$

$$W_{1323} = (b_{31})^2 a_{33}\pi_3 = 0.05^2 * 0.5 * 0.33 = 0.000413$$

$$k_3 = \underset{l}{\operatorname{argmax}}\{W_{132l}\} = \underset{l}{\operatorname{argmax}}\{W_{1321}, W_{1322}, W_{1323}\} = 1$$

$$W_{0113}W_{132k_3} = W_{0113}W_{1321} = 0.165 * 0.0297 = 0.0049$$

$$u = \operatorname{argmax}_j \{W_{011j} W_{112k_j}\} = \operatorname{argmax}_j \{W_{0111} W_{112k_1}, W_{0112} W_{112k_2}, W_{0113} W_{112k_3}\}$$

$$= 3$$

$$X = X \cup \{x_1 = s_u\} \cup \{x_2 = s_{k_u}\} = X \cup \{x_1 = s_3\} \cup \{x_2 = s_{k_3}\} \\ = X \cup \{x_1 = s_3\} \cup \{x_2 = s_1\} = \{x_1 = \text{rainy}, x_2 = \text{sunny}\}$$

At $t=3$, we have:

$$W_{2k_331} = W_{2131} = (b_{12})^2 a_{11} \pi_1 = 0.2^2 * 0.5 * 0.33 = 0.0066$$

$$W_{2k_332} = W_{2132} = (b_{22})^2 a_{12} \pi_2 = 0.25^2 * 0.25 * 0.33 = 0.005156$$

$$W_{2k_333} = W_{2133} = (b_{32})^2 a_{13} \pi_3 = 0.1^2 * 0.25 * 0.33 = 0.000825$$

$$u = \operatorname{argmax}_j \{W_{213j}\} = \operatorname{argmax}_j \{W_{2131}, W_{2132}, W_{2133}\} = 1$$

$$X = X \cup \{x_3 = s_u\} = X \cup \{x_3 = s_1\} = \{x_1 = \text{rainy}, x_2 = \text{sunny}, x_3 = \text{sunny}\}$$

As a result, the optimal state sequence is $X = \{x_1=\text{rainy}, x_2=\text{sunny}, x_3=\text{sunny}\}$, which is the same to the one from individually optimal procedure (see table IV.4.2.1), Viterbi algorithm (see table IV.4.2.2), and normal longest-path algorithm (see table IV.4.2.3). The resulted sequence $X = \{x_1=\text{rainy}, x_2=\text{sunny}, x_3=\text{sunny}\}$ that is the longest path is drawn as bold line from node X_0 to node X_{13} to node X_{21} to node X_{31} inside the state transition graph, as seen in following figure IV.4.2.6.

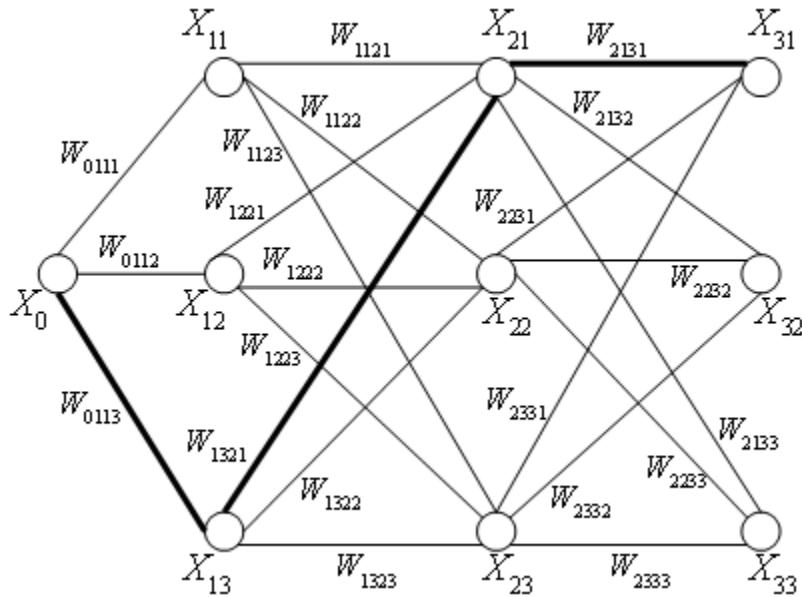


Figure IV.4.2.6. Longest path drawn as bold line inside state transition graph

The uncovering problem is described thoroughly in this sub-section IV.4.2 because it is very important in the research when I apply algorithms such as individually optimal procedure, Viterbi algorithm, longest-path algorithm, and advanced longest-path algorithm into solving such problem in order to predict learner's styles given observations about her/him. The longest-path algorithm is also published in (Nguyen L., Longest-path Algorithm to Solve Uncovering Problem of Hidden Markov Model, 2016). Successive sub-section IV.4.3 will mention the last problem of HMM that is the learning problem.

IV.4.3. HMM learning problem

The learning problem is to adjust parameters such as initial state distribution Π , transition probability matrix A , and observation probability matrix B so that given HMM Δ gets more appropriate to an observation sequence $O = \{o_1, o_2, \dots, o_T\}$ with note that Δ is represented by these parameters. In other words, the learning problem is to adjust parameters by maximizing probability of observation sequence O , as follows:

$$(A, B, \Pi) = \underset{A, B, \Pi}{\operatorname{argmax}} P(O|\Delta)$$

The Expectation Maximization (EM) algorithm is applied successfully into solving HMM learning problem, which is equivalently well-known Baum-Welch algorithm (Rabiner, 1989). Although EM algorithm is aforementioned in previous sub-section III.3.2, successive sub-section IV.4.3.1 describes EM algorithm in detailed before going into Baum-Welch algorithm.

IV.4.3.1. EM algorithm

Expectation Maximization (EM) is effective parameter estimator in case that incomplete data is composed of two parts: observed part and hidden part (missing part). EM is iterative algorithm that improves parameters after iterations until reaching optimal parameters. Each iteration includes two steps: E(xpectation) step and M(aximization) step. In E-step the hidden data is estimated based on observed data and current estimate of parameters; so the lower-bound of likelihood function is computed by the expectation of complete data. In M-step new estimates of parameters are determined by maximizing the lower-bound. Please see document (Sean, 2009) for short tutorial of EM. This sub-section IV.4.3.1 focuses on practice general EM algorithm; the theory of EM algorithm is described comprehensively in article “Maximum Likelihood from Incomplete Data via the EM algorithm” by authors Dempster, Laird, and Rubin (Dempster, Laird, & Rubin, 1977).

Suppose O and X are observed data and hidden data, respectively. Note O and X can be represented in any form such as discrete values, scalar, integer number, real number, vector, list, sequence, sample, and matrix. Let Θ represent parameters of probability distribution. Concretely, Θ includes initial state distribution Π , transition probability matrix A , and observation probability matrix B inside HMM. In other words, Θ represents HMM Δ itself. EM algorithm aims to estimate Θ by finding out which $\widehat{\Theta}$ maximizes the likelihood function $L(\Theta) = P(O|\Theta)$.

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}} L(\Theta) = \underset{\Theta}{\operatorname{argmax}} P(O|\Theta)$$

Where $\widehat{\Theta}$ is the optimal estimate of parameters which is called usually *parameter estimate*. Because the likelihood function is product of factors, it is replaced by the log-likelihood function $LnL(\Theta)$ that is natural logarithm of the likelihood function $L(\Theta)$, for convenience. We have:

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}} LnL(\Theta) = \underset{\Theta}{\operatorname{argmax}} \ln(L(\Theta)) = \underset{\Theta}{\operatorname{argmax}} \ln(P(O|\Theta))$$

Where,

$$LnL(\Theta) = \ln(L(\Theta)) = \ln(P(O|\Theta))$$

The method finding out the parameter estimate $\widehat{\Theta}$ by maximizing the log-likelihood function is called maximum likelihood estimation (MLE) aforementioned in previous sub-section III.5.1. Of course, EM algorithm is based on MLE.

Suppose the current parameter is Θ_t after the t^{th} iteration. Next we must find out the new estimate $\widehat{\Theta}$ that maximizes the next log-likelihood function $\text{LnL}(\Theta)$. In other words it maximizes the deviation between current log-likelihood $\text{LnL}(\Theta_t)$ and next log-likelihood $\text{LnL}(\Theta)$ with regard to Θ .

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}} (\text{LnL}(\Theta) - \text{LnL}(\Theta_t)) = \underset{\Theta}{\operatorname{argmax}} (Q(\Theta, \Theta_t))$$

Where $Q(\Theta, \Theta_t) = \text{LnL}(\Theta) - \text{LnL}(\Theta_t)$ is the deviation between current log-likelihood $\text{LnL}(\Theta_t)$ and next log-likelihood $\text{LnL}(\Theta)$ with note that $Q(\Theta, \Theta_t)$ is function of Θ when Θ_t was determined.

Suppose the total probability of observed data can be determined by marginalizing over hidden data:

$$P(O|\Theta) = \sum_X P(O|X, \Theta)P(X|\Theta)$$

The expansion of $P(O|\Theta)$ is total probability rule specified formula III.1.1.4. The deviation $Q(\Theta, \Theta_t)$ is re-written:

$$\begin{aligned} Q(\Theta, \Theta_t) &= \text{LnL}(\Theta) - \text{LnL}(\Theta_t) = \ln(P(O|\Theta)) - \ln(P(O|\Theta_t)) \\ &= \ln\left(\sum_X P(O|X, \Theta)P(X|\Theta)\right) - \ln(P(O|\Theta_t)) \\ &= \ln\left(\sum_X P(O, X|\Theta)\right) - \ln(P(O|\Theta_t)) \\ &\quad \text{(Due to multiplication rule specified by formula III.1.1.3)} \\ &= \ln\left(\sum_X P(X|O, \Theta_t) \frac{P(O, X|\Theta)}{P(X|O, \Theta_t)}\right) - \ln(P(O|\Theta_t)) \end{aligned}$$

Because hidden X is the complete set of mutually exclusive variables, the sum of conditional probabilities of X is equal to 1 given O and Θ_t .

$$\sum_X P(X|O, \Theta_t) = 1$$

Applying Jensen's inequality (Sean, 2009, pp. 3-4)

$$\ln\left(\sum_i \lambda_i x_i\right) \geq \sum_i \lambda_i \ln(x_i) \text{ where } \sum_i \lambda_i = 1$$

into deviation $Q(\Theta, \Theta_t)$, we have:

$$\begin{aligned} Q(\Theta, \Theta_t) &\geq \left(\sum_X P(X|O, \Theta_t) \ln\left(\frac{P(O, X|\Theta)}{P(X|O, \Theta_t)}\right) \right) - \ln(P(O|\Theta_t)) \\ &= \left(\sum_X P(X|O, \Theta_t) \left(\ln(P(O, X|\Theta)) - \ln(P(X|O, \Theta_t)) \right) \right) - \ln(P(O|\Theta_t)) \\ &= \left(\sum_X P(X|O, \Theta_t) \ln(P(O, X|\Theta)) \right) - \left(\sum_X P(X|O, \Theta_t) \ln(P(X|O, \Theta_t)) \right) \\ &\quad - \ln(P(O|\Theta_t)) \\ &= \sum_X P(X|O, \Theta_t) \ln(P(O, X|\Theta)) + C \end{aligned}$$

Where,

$$C = -\left(\sum_X P(X|O, \Theta_t) \ln(P(X|O, \Theta_t)) \right) - \ln(P(O|\Theta_t))$$

Because C is constant with regard to Θ , it is possible to eliminate C in order to simplify the optimization criterion as follows:

$$\begin{aligned}\widehat{\Theta} &= \underset{\Theta}{\operatorname{argmax}}(Q(\Theta, \Theta_t)) \approx \underset{\Theta}{\operatorname{argmax}} \left(\sum_X P(X|O, \Theta_t) \ln(P(O, X|\Theta)) - C \right) \\ &= \underset{\Theta}{\operatorname{argmax}} \sum_X P(X|O, \Theta_t) \ln(P(O, X|\Theta))\end{aligned}$$

The expression $\sum_X P(X|O, \Theta_t) \ln(P(O, X|\Theta))$ is essentially expectation of $\ln(P(O, X|\Theta))$ given conditional probability distribution $P(X|O, \Theta_t)$ when $P(X|O, \Theta_t)$ is totally determined. Let $E_{X|O, \Theta_t}\{\ln(P(O, X|\Theta))\}$ denote this conditional expectation, formula IV.4.3.1.1 specifies EM optimization criterion for determining the parameter estimate, which is the most important aspect of EM algorithm.

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}} E_{X|O, \Theta_t}\{\ln(P(O, X|\Theta))\}$$

Where,

$$E_{X|O, \Theta_t}\{\ln(P(O, X|\Theta))\} = \sum_X P(X|O, \Theta_t) \ln(P(O, X|\Theta))$$

Formula IV.4.3.1.1. EM optimization criterion based on conditional expectation

If $P(X|O, \Theta_t)$ is continuous density function, the continuous version of this conditional expectation is:

$$E_{X|O, \Theta_t}\{\ln(P(O, X|\Theta))\} = \int_X P(X|O, \Theta_t) \ln(P(O, X|\Theta))$$

Finally, the EM algorithm is described in table IV.4.3.1.1.

Starting with initial parameter Θ_0 , each iteration in EM algorithm has two steps:

1. *E-step*: computing the conditional expectation $E_{X|O, \Theta_t}\{\ln(P(O, X|\Theta))\}$ based on the current parameter Θ_t according to formula IV.4.3.1.1.
2. *M-step*: finding out the estimate $\widehat{\Theta}$ that maximizes such conditional expectation. The next parameter Θ_{t+1} is assigned by the estimate $\widehat{\Theta}$, we have:

$$\Theta_{t+1} = \widehat{\Theta}$$

Of course Θ_{t+1} becomes current parameter for next iteration. How to maximize the conditional expectation is optimization problem which is dependent on applications. For example, the popular method to solve optimization problem is Lagrangian duality (Jia, 2013, p. 8).

EM algorithm stops when it meets the terminating condition, for example, the difference of current parameter Θ_t and next parameter Θ_{t+1} is smaller than some pre-defined threshold ε .

$$|\Theta_{t+1} - \Theta_t| < \varepsilon$$

In addition, it is possible to define a custom terminating condition.

Table IV.4.3.1.1. General EM algorithm

General EM algorithm is simple but please pay attention to the concept of lower-bound and what the essence of EM is. Recall that the next log-likelihood function $LnL(\Theta)$ is current likelihood function $LnL(\Theta_t)$ plus the deviation $Q(\Theta, \Theta_t)$. We have:

$$LnL(\Theta) = LnL(\Theta_t) + Q(\Theta, \Theta_t) \geq LnL(\Theta_t) + E_{X|O, \Theta_t}\{\ln(P(O, X|\Theta))\} + C$$

Where,

$$C = - \left(\sum_X P(X|O, \Theta_t) \ln(P(X|O, \Theta_t)) \right) - \ln(P(O|\Theta_t))$$

Let $lb(\Theta, \Theta_t)$ denote the lower-bound of the log-likelihood function $LnL(\Theta)$ given current parameter Θ_t (Sean, 2009, pp. 7-8). The lower-bound $lb(\Theta, \Theta_t)$ is the function of Θ as specified by formula IV.4.3.1.2:

$$lb(\Theta, \Theta_t) = LnL(\Theta_t) + E_{X|O, \Theta_t} \{ \ln(P(O, X|\Theta)) \} + C$$

Formula IV.4.3.1.2. Lower-bound of log-likelihood function

Determining $lb(\Theta, \Theta_t)$ is to calculate the EM conditional expectation $E_{X|O, \Theta_t} \{ \ln(P(O, X|\Theta)) \}$ because terms $LnL(\Theta_t)$ and C were totally determined. The lower-bound $lb(\Theta, \Theta_t)$ has a feature where its evaluation at $\Theta = \Theta_t$ equals the log-likelihood function $LnL(\Theta)$.

$$lb(\Theta_t, \Theta_t) = LnL(\Theta_t)$$

In fact,

$$\begin{aligned} lb(\Theta, \Theta_t) &= LnL(\Theta_t) + E_{X|O, \Theta_t} \{ \ln(P(O, X|\Theta)) \} + C \\ &= LnL(\Theta_t) + \left(\sum_X P(X|O, \Theta_t) \ln(P(O, X|\Theta)) \right) - \left(\sum_X P(X|O, \Theta_t) \ln(P(X|O, \Theta_t)) \right) \\ &\quad - \ln(P(O|\Theta_t)) \\ &= LnL(\Theta_t) + \left(\sum_X P(X|O, \Theta_t) \ln \left(\frac{P(O, X|\Theta)}{P(X|O, \Theta_t)} \right) \right) - \ln(P(O|\Theta_t)) \\ &= LnL(\Theta_t) + \left(\sum_X P(X|O, \Theta_t) \ln \left(\frac{P(X|O, \Theta)P(O|\Theta)}{P(X|O, \Theta_t)} \right) \right) - \ln(P(O|\Theta_t)) \end{aligned}$$

(Due to multiplication rule specified by formula III.1.1.3)

It implies

$$\begin{aligned} lb(\Theta_t, \Theta_t) &= LnL(\Theta_t) + \left(\sum_X P(X|O, \Theta_t) \ln \left(\frac{P(X|O, \Theta_t)P(O|\Theta_t)}{P(X|O, \Theta_t)} \right) \right) - \ln(P(O|\Theta_t)) \\ &= LnL(\Theta_t) + \left(\sum_X P(X|O, \Theta_t) \ln(P(O|\Theta_t)) \right) - \ln(P(O|\Theta_t)) \\ &= LnL(\Theta_t) + \ln(P(O|\Theta_t)) \sum_X P(X|O, \Theta_t) - \ln(P(O|\Theta_t)) \\ &= LnL(\Theta_t) + \ln(P(O|\Theta_t)) - \ln(P(O|\Theta_t)) \\ &\quad \left(\text{due to } \sum_X P(X|O, \Theta_t) = 1 \right) \\ &= LnL(\Theta_t) \end{aligned}$$

Figure IV.4.3.1.1 (Borman, 2004, p. 7) shows relationship between the log-likelihood function $LnL(\Theta)$ and its lower-bound $lb(\Theta, \Theta_t)$.

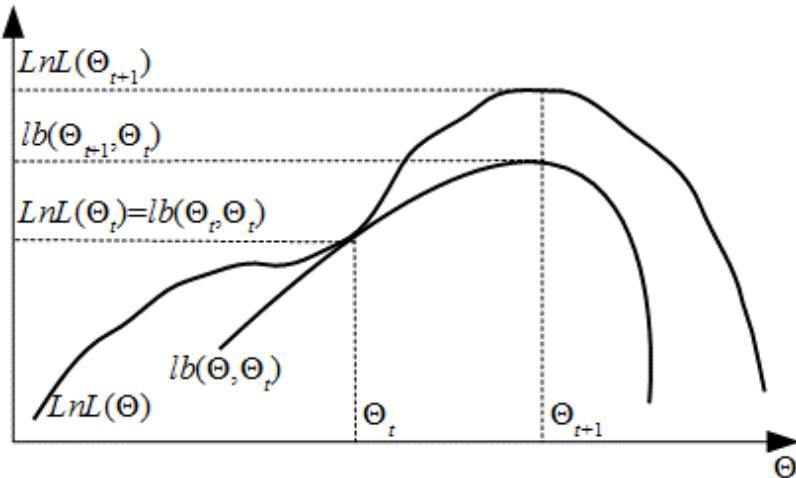


Figure IV.4.3.1.1. Relationship between the log-likelihood function and its lower-bound

The essence of maximizing the deviation $Q(\Theta, \Theta_t)$ is to maximize the lower-bound $lb(\Theta, \Theta_t)$ with respect to Θ . For each iteration the new lower-bound and its maximum are computed based on previous lower-bound. A single iteration in EM algorithm can be understood as below:

1. E-step: the new lower-bound $lb(\Theta, \Theta_t)$ is determined based on current parameter Θ_t according to formula IV.4.3.1.2. Of course, determining $lb(\Theta, \Theta_t)$ is to calculate the EM conditional expectation $E_{X|O, \Theta_t}\{ln(P(O, X|\Theta))\}$.
3. M-step: finding out the estimate $\widehat{\Theta}$ so that $lb(\Theta, \Theta_t)$ reaches maximum at $\widehat{\Theta}$. The next parameter Θ_{t+1} is assigned by the estimate $\widehat{\Theta}$, we have:

$$\Theta_{t+1} = \widehat{\Theta}$$

Of course Θ_{t+1} becomes current parameter for next iteration. Note, maximizing $lb(\Theta, \Theta_t)$ is to maximize the EM conditional expectation $E_{X|O, \Theta_t}\{ln(P(O, X|\Theta))\}$.

In general, it is easy to calculate the EM expectation $E_{X|O, \Theta_t}\{ln(P(O, X|\Theta))\}$ but finding out the estimate $\widehat{\Theta}$ based on maximizing such expectation is complicated optimization problem. It is possible to state that the essence of EM algorithm is to determine the estimate $\widehat{\Theta}$. Now the EM algorithm is introduced with full of details. How to apply it into solving HMM learning problem is described in successive subsection IV.4.3.2.

IV.4.3.2. Applying EM algorithm into solving learning problem

Now going back the HMM learning problem, the EM algorithm is applied into solving this problem, which is equivalently well-known Baum-Welch algorithm by authors Leonard E. Baum and Lloyd R. Welch (Rabiner, 1989). The parameter Θ becomes the HMM model $\Delta = (A, B, \Pi)$. Recall that the learning problem is to adjust parameters by maximizing probability of observation sequence O , as follows:

$$\widehat{\Delta} = (\widehat{A}, \widehat{B}, \widehat{\Pi}) = (\widehat{a}_{ij}, \widehat{b}_j(k), \widehat{\pi}_j) = \underset{\Delta}{\operatorname{argmax}} P(O|\Delta)$$

Where \widehat{a}_{ij} , $\widehat{b}_j(k)$, $\widehat{\pi}_j$ are parameter estimates and so, the purpose of HMM learning problem is to determine them.

The observation sequence $O = \{o_1, o_2, \dots, o_T\}$ and state sequence $X = \{x_1, x_2, \dots, x_T\}$ are observed data and hidden data within context of EM algorithm, respectively. Note O and X is now represented in sequence. According to EM algorithm, the parameter estimate $\widehat{\Delta}$ is determined as follows:

$$\widehat{\Delta} = (\widehat{a}_{ij}, \widehat{b}_j(k), \widehat{\pi}_j) = \underset{\Delta}{\operatorname{argmax}} E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}$$

Where $\Delta_r = (A_r, B_r, \Pi_r)$ is the known parameter at the current iteration. Note that we use notation Δ_r instead of popular notation Δ_t in order to distinguish iteration indices of EM algorithm from time points inside observation sequence O and state sequence X . The EM conditional expectation in accordance with HMM is:

$$\begin{aligned} E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \} &= \sum_X P(X|O, \Delta_r) \ln(P(O, X|\Delta)) \\ &= \sum_X P(X|O, \Delta_r) \ln(P(O|X, \Delta) P(X|\Delta)) \\ &= \sum_X P(X|O, \Delta_r) \ln(P(o_1, o_2, \dots, o_T | x_1, x_2, \dots, x_T, \Delta) * P(x_1, x_2, \dots, x_T|\Delta)) \\ &= \sum_X P(X|O, \Delta_r) \ln(P(o_1|x_1, x_2, \dots, x_T, \Delta) * P(o_2|x_1, x_2, \dots, x_T, \Delta) * \dots \\ &\quad * P(o_T|x_1, x_2, \dots, x_T, \Delta) * P(x_1, x_2, \dots, x_T|\Delta)) \end{aligned}$$

(Because observations o_1, o_2, \dots, o_T are mutually independent)

$$\begin{aligned} &= \sum_X P(X|O, \Delta_r) \ln(P(o_1|x_1, \Delta) * P(o_2|x_2, \Delta) * \dots * P(o_T|x_T, \Delta) \\ &\quad * P(x_1, x_2, \dots, x_T|\Delta)) \end{aligned}$$

(Because each observations o_t is only dependent on state x_t)

$$\begin{aligned} &= \sum_X P(X|O, \Delta_r) \ln \left(\left(\prod_{t=1}^T P(o_t|x_t, \Delta) \right) * P(x_1, x_2, \dots, x_T|\Delta) \right) \\ &= \sum_X P(X|O, \Delta_r) \ln \left(\left(\prod_{t=1}^T P(o_t|x_t, \Delta) \right) * P(x_T|x_1, x_2, \dots, x_{T-1}, \Delta) \right. \\ &\quad \left. * P(x_1, x_2, \dots, x_{T-1}|\Delta) \right) \\ &= \sum_X P(X|O, \Delta_r) \ln \left(\left(\prod_{t=1}^T P(o_t|x_t, \Delta) \right) * P(x_T|x_{T-1}, \Delta) * P(x_1, x_2, \dots, x_{T-1}|\Delta) \right) \end{aligned}$$

(Because each state x_t is only dependent on previous state x_{t-1})

$$\begin{aligned} &= \sum_X P(X|O, \Delta_r) \ln \left(\left(\prod_{t=1}^T P(o_t|x_t, \Delta) \right) * P(x_T|x_{T-1}, \Delta) * P(x_{T-1}|x_{T-2}, \Delta) * \dots \right. \\ &\quad \left. * P(x_2|x_1, \Delta) * P(x_1|\Delta) \right) \end{aligned}$$

(Due to recurrence on probability $P(x_1, x_2, \dots, x_t)$)

$$= \sum_X P(X|O, \Delta_r) \ln \left(\left(\prod_{t=1}^T P(o_t|x_t, \Delta) \right) \left(\prod_{t=2}^T P(x_t|x_{t-1}, \Delta) \right) P(x_1|\Delta) \right)$$

It is conventional that $P(x_1|x_0, \Delta) = P(x_1|\Delta)$ where x_0 is pseudo-state, formula [IV.4.3.2.1](#) specifies general EM conditional expectation for HMM:

$$\begin{aligned} E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \} &= \sum_X P(X|O, \Delta_r) \ln \left(\prod_{t=1}^T P(x_t|x_{t-1}, \Delta) P(o_t|x_t, \Delta) \right) \\ &= \sum_X P(X|O, \Delta_r) \sum_{t=1}^T (\ln(P(x_t|x_{t-1}, \Delta)) + \ln(P(o_t|x_t, \Delta))) \end{aligned}$$

Formula IV.4.3.2.1. General EM conditional expectation for HMM

Let $I(x_{t-1} = s_i, x_t = s_j)$ and $I(x_t = s_j, o_t = \varphi_k)$ are two index functions so that

$$\begin{aligned} I(x_{t-1} = s_i, x_t = s_j) &= \begin{cases} 1 & \text{if } x_{t-1} = s_i \text{ and } x_t = s_j \\ 0 & \text{otherwise} \end{cases} \\ I(x_t = s_j, o_t = \varphi_k) &= \begin{cases} 1 & \text{if } x_t = s_j \text{ and } o_t = \varphi_k \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

We have:

$$\begin{aligned} E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \} &= \sum_X P(X|O, \Delta_r) \left(\sum_{t=1}^T \ln(P(x_t|x_{t-1}, \Delta)) + \sum_{t=1}^T \ln(P(o_t|x_t, \Delta)) \right) \\ &= \sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(P(s_j|s_i, \Delta)) \right. \\ &\quad \left. + \sum_{j=1}^n \sum_{k=1}^m \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \ln(P(\varphi_k|s_j, \Delta)) \right) \\ &= \sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(a_{ij}) \right. \\ &\quad \left. + \sum_{j=1}^n \sum_{k=1}^m \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \ln(b_j(k)) \right) \end{aligned}$$

Because of the convention $P(x_1|x_0, \Delta) = P(x_1|\Delta)$, matrix \prod is degradation case of matrix A at time point $t=1$. In other words, the initial probability π_j is equal to the transition probability a_{ij} from pseudo-state x_0 to state $x_1=s_j$.

$$P(x_1 = s_j|x_0, \Delta) = P(x_1 = s_j|\Delta) = \pi_j$$

Note that $n=|S|$ is the number of possible states and $m=|\Phi|$ is the number of possible observations.

Shortly, the EM conditional expectation for HMM is specified by formula [IV.4.3.2.2](#).

$$\begin{aligned}
 & E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \} \\
 &= \sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(a_{ij}) \right. \\
 &\quad \left. + \sum_{j=1}^m \sum_{k=1}^m \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \ln(b_j(k)) \right)
 \end{aligned}$$

Formula IV.4.3.2.2. EM conditional expectation for HMM

Where,

$$\begin{aligned}
 I(x_{t-1} = s_i, x_t = s_j) &= \begin{cases} 1 & \text{if } x_{t-1} = s_i \text{ and } x_t = s_j \\ 0 & \text{otherwise} \end{cases} \\
 I(x_t = s_j, o_t = \varphi_k) &= \begin{cases} 1 & \text{if } x_t = s_j \text{ and } o_t = \varphi_k \\ 0 & \text{otherwise} \end{cases} \\
 P(x_1 = s_j | x_0, \Delta) &= P(x_1 = s_j | \Delta) = \pi_j
 \end{aligned}$$

Note that the conditional expectation $E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \}$ is function of Δ . There are two constraints for HMM as follows:

$$\begin{aligned}
 \sum_{j=1}^n a_{ij} &= 1, \forall i = \overline{1, n} \\
 \sum_{k=1}^m b_j(k) &= 1, \forall k = \overline{1, m}
 \end{aligned}$$

Maximizing $E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \}$ with subject to these constraints is optimization problem that is solved by Lagrangian duality theorem (Jia, 2013, p. 8). Original optimization problem mentions minimizing target function but it is easy to infer that maximizing target function shares the same methodology. Let $l(\Delta, \lambda, \mu)$ be Lagrangian function constructed from $E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \}$ together with these constraints (Ramage, 2007, p. 9), we have formula IV.4.3.2.3 for specifying HMM Lagrangian function as follows:

$$\begin{aligned}
 l(\Delta, \lambda, \mu) &= l(a_{ij}, b_j(k), \lambda_i, \mu_j) \\
 &= E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \} + \sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \\
 &\quad + \sum_{j=1}^m \mu_j \left(1 - \sum_{k=1}^m b_j(k) \right)
 \end{aligned}$$

Formula IV.4.3.2.3. Lagrangian function for HMM

Where λ is n -component vector $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ and μ is m -component vector $\mu = (\mu_1, \mu_2, \dots, \mu_m)$. Factors $\lambda_i \geq 0$ and $\mu_j \geq 0$ are called Lagrange multipliers or Karush-Kuhn-Tucker multipliers (Wikipedia, Karush–Kuhn–Tucker conditions, 2014) or dual variables. The expectation $E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \}$ is specified by formula IV.4.3.2.2.

The parameter estimate $\widehat{\Delta}$ is extreme point of the Lagrangian function. According to Lagrangian duality theorem (Boyd & Vandenberghe, 2009, p. 216) (Jia, 2013, p. 8), we have:

$$\widehat{\Delta} = (\widehat{A}, \widehat{B}) = (\widehat{a}_{ij}, \widehat{b}_j(k)) = \underset{A, B}{\operatorname{argmax}} l(\Delta, \lambda, \mu)$$

$$(\widehat{\lambda}, \widehat{\mu}) = \underset{\lambda, \mu}{\operatorname{argmin}} l(\Delta, \lambda, \mu)$$

The parameter estimate $\widehat{\Delta} = (\widehat{a}_{ij}, \widehat{b}_j(k))$ is determined by setting partial derivatives of $l(\Delta, \lambda, \mu)$ with respect to a_{ij} and $b_j(k)$ to be zero. The partial derivative of $l(\Delta, \lambda, \mu)$ with respect to a_{ij} is:

$$\begin{aligned} \frac{\partial l(\Delta, \lambda, \mu)}{\partial a_{ij}} &= \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial a_{ij}} + \frac{\partial}{\partial a_{ij}} \left(\sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \right) \\ &\quad + \frac{\partial}{\partial a_{ij}} \left(\sum_{j=1}^n \mu_j \left(1 - \sum_{k=1}^m b_j(k) \right) \right) \\ &= \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial a_{ij}} - \lambda_i \\ &= \frac{\partial}{\partial a_{ij}} \left(\sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^n \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(a_{ij}) \right. \right. \\ &\quad \left. \left. + \sum_{j=1}^n \sum_{k=1}^m \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \ln(b_j(k)) \right) \right) - \lambda_i \\ &= \left(\sum_X P(X|O, \Delta_r) \sum_{i=1}^n \sum_{j=1}^n \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \frac{\partial \ln(a_{ij})}{\partial a_{ij}} \right) - \lambda_i \\ &= \left(\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \frac{1}{a_{ij}} \right) - \lambda_i \\ &= \frac{1}{a_{ij}} \left(\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \right) - \lambda_i \end{aligned}$$

Setting the partial derivative $\frac{\partial l(\Delta, \lambda, \mu)}{\partial a_{ij}}$ to be zero:

$$\frac{\partial l(\Delta, \lambda, \mu)}{\partial a_{ij}} = 0 \Leftrightarrow \frac{1}{a_{ij}} \left(\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \right) - \lambda_i = 0$$

The parameter estimate \widehat{a}_{ij} is solution of equation $\frac{\partial l(\Delta, \lambda, \mu)}{\partial a_{ij}} = 0$, we have:

$$\widehat{a}_{ij} = \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j)}{\lambda_i}$$

It is required to estimate the Lagrange multiplier λ_i . The multiplier estimate $\widehat{\lambda}_i$ is determined by setting the partial derivative of $l(\Delta, \lambda, \mu)$ with respect to λ_i to be zero as follows:

$$\frac{\partial l(\Delta, \lambda, \mu)}{\partial \lambda_i} = 0$$

$$\begin{aligned} & \Rightarrow \frac{\partial E_{X|O,\Delta_r}\{ln(P(O,X|\Delta))\}}{\partial \lambda_i} + \frac{\partial}{\partial \lambda_i} \left(\sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \right) \\ & \quad + \frac{\partial}{\partial \lambda_i} \left(\sum_{j=1}^n \mu_j \left(1 - \sum_{k=1}^m b_j(k) \right) \right) = 0 \\ & \Rightarrow 1 - \sum_{j=1}^n a_{ij} = 0 \end{aligned}$$

Substituting \hat{a}_{ij} for a_{ij} , we have:

$$\begin{aligned} 1 - \sum_{j=1}^n \hat{a}_{ij} &= 1 - \sum_{j=1}^n \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j)}{\lambda_i} \\ &= 1 - \frac{1}{\lambda_i} \sum_{j=1}^n \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) = 0 \end{aligned}$$

It implies:

$$\begin{aligned} \hat{\lambda}_i &= \sum_{j=1}^n \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \\ &= \sum_X P(X|O, \Delta_r) \sum_{t=1}^T \sum_{j=1}^n I(x_{t-1} = s_i, x_t = s_j) \\ &= \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i) \end{aligned}$$

Where, $I(s_i = x_{t-1})$ is index function.

$$I(x_{t-1} = s_i) = \begin{cases} 1 & \text{if } x_{t-1} = s_i \\ 0 & \text{otherwise} \end{cases}$$

Substituting $\hat{\lambda}_i$ for λ_i inside

$$\hat{a}_{ij} = \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j)}{\hat{\lambda}_i}$$

We have:

$$\begin{aligned} \hat{a}_{ij} &= \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j)}{\hat{\lambda}_i} \\ &= \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j)}{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i)} \end{aligned}$$

Evaluating the numerator, we have:

$$\begin{aligned} & \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \\ &= \sum_{t=1}^T \sum_X I(x_{t-1} = s_i, x_t = s_j) P(X|O, \Delta_r) \\ &= \sum_{t=1}^T \sum_X I(x_{t-1} = s_i, x_t = s_j) P(x_1, \dots, x_{t-1}, x_t, \dots, x_T | O, \Delta_r) \end{aligned}$$

$$\begin{aligned}
 &= \sum_{t=1}^T P(x_{t-1} = s_i, x_t = s_j | O, \Delta_r) \\
 &\quad (\text{Due to total probability rule specified by formula III.1.1.4}) \\
 &= \sum_{t=1}^T \frac{P(O, x_{t-1} = s_i, x_t = s_j | \Delta_r)}{P(O | \Delta_r)} \\
 &\quad (\text{Due to multiplication rule specified by formula III.1.1.3})
 \end{aligned}$$

Evaluating the denominator, we have:

$$\begin{aligned}
 &\sum_X P(X | O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i) \\
 &= \sum_{t=1}^T \sum_X I(x_{t-1} = s_i) P(X | O, \Delta_r) \\
 &= \sum_{t=1}^T \sum_X I(x_{t-1} = s_i) P(x_1, \dots, x_{t-1}, x_t, \dots, x_T | O, \Delta_r) \\
 &= \sum_{t=1}^T P(x_{t-1} = s_i | O, \Delta_r) \\
 &\quad (\text{Due to total probability rule specified by formula III.1.1.4}) \\
 &= \sum_{t=1}^T \frac{P(O, x_{t-1} = s_i | \Delta_r)}{P(O | \Delta_r)} \\
 &\quad (\text{Due to multiplication rule specified by formula III.1.1.3})
 \end{aligned}$$

It implies

$$\hat{a}_{ij} = \frac{\sum_X P(X | O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j)}{\sum_X P(X | O, \Delta_r) \sum_{t=1}^T I(x_{t-1} = s_i)} = \frac{\sum_{t=1}^T P(O, x_{t-1} = s_i, x_t = s_j | \Delta_r)}{\sum_{t=1}^T P(O, x_{t-1} = s_i | \Delta_r)}$$

Because of the convention $P(x_1 | x_0, \Delta) = P(x_1 | \Delta)$, the estimate \hat{a}_{ij} is fixed as follows:

$$\hat{a}_{ij} = \frac{\sum_{t=2}^T P(O, x_{t-1} = s_i, x_t = s_j | \Delta_r)}{\sum_{t=2}^T P(O, x_{t-1} = s_i | \Delta_r)}$$

The estimate of initial probability $\hat{\pi}_j$ is known as specific estimate \hat{a}_{ij} from pseudo-state x_0 to state $x_1 = s_j$. It means that

$$\hat{\pi}_j = \frac{P(O, x_1 = s_j | \Delta_r)}{\sum_{i=1}^n P(O, x_1 = s_i | \Delta_r)}$$

Recall that the parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k))$ is determined by setting partial derivatives of $l(\Delta, \lambda, \mu)$ with respect to a_{ij} and $b_j(k)$ to be zero. The parameter estimate \hat{a}_{ij} was determined. Now it is required to calculate the parameter estimate $\hat{b}_j(k)$. The partial derivative of Lagrangian function $l(\Delta, \lambda, \mu)$ with respect to $b_j(k)$ is:

$$\begin{aligned}
 \frac{\partial l(\Delta, \lambda, \mu)}{\partial b_j(k)} &= \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial b_j(k)} + \frac{\partial}{\partial b_j(k)} \left(\sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \right) \\
 &\quad + \frac{\partial}{\partial b_j(k)} \left(\sum_{j=1}^n \mu_j \left(1 - \sum_{k=1}^m b_j(k) \right) \right) \\
 &= \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial b_j(k)} - \mu_j \\
 &= \frac{\partial}{\partial b_j(k)} \left(\sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^n \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(a_{ij}) \right. \right. \\
 &\quad \left. \left. + \sum_{j=1}^n \sum_{k=1}^m \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \ln(b_j(k)) \right) \right) - \mu_j \\
 &= \left(\sum_X P(X|O, \Delta_r) \sum_{j=1}^n \sum_{k=1}^m \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \frac{\partial \ln(b_j(k))}{\partial b_j(k)} \right) - \mu_j \\
 &= \left(\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \frac{1}{b_j(k)} \right) - \mu_j \\
 &= \frac{1}{b_j(k)} \left(\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \right) - \mu_j
 \end{aligned}$$

Setting the partial derivative $\frac{\partial l(\Delta, \lambda, \mu)}{\partial b_j(k)}$ to be zero:

$$\frac{\partial l(\Delta, \lambda, \mu)}{\partial b_j(k)} = 0 \Leftrightarrow \frac{1}{b_j(k)} \left(\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \right) - \mu_j = 0$$

The parameter estimate \hat{a}_{ij} is solution of equation $\frac{\partial l(\Delta, \lambda, \mu)}{\partial b_j(k)} = 0$, we have:

$$\hat{b}_j(k) = \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k)}{\mu_j}$$

It is required to estimate the Lagrange multiplier μ_j . The multiplier estimate $\hat{\mu}_j$ is determined by setting the partial derivative of $l(\Delta, \lambda, \mu)$ with respect to μ_j to be zero as follows:

$$\begin{aligned}
 \frac{\partial l(\Delta, \lambda, \mu)}{\partial \mu_j} &= 0 \\
 &\Rightarrow \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial \mu_j} + \frac{\partial}{\partial \mu_j} \left(\sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \right) \\
 &\quad + \frac{\partial}{\partial \mu_j} \left(\sum_{j=1}^n \mu_j \left(1 - \sum_{k=1}^m b_j(k) \right) \right) = 0 \\
 &\Rightarrow 1 - \sum_{k=1}^m b_j(k) = 0
 \end{aligned}$$

Substituting $\hat{b}_j(k)$ for $b_j(k)$ we have:

$$\begin{aligned} 1 - \sum_{k=1}^m \hat{b}_j(k) &= 1 - \sum_{k=1}^m \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k)}{\mu_j} \\ &= 1 - \frac{1}{\mu_j} \sum_{k=1}^m \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) = 0 \end{aligned}$$

It implies:

$$\begin{aligned} \hat{\mu}_j &= \sum_{k=1}^m \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \\ &= \sum_X P(X|O, \Delta_r) \sum_{t=1}^T \sum_{k=1}^m I(x_t = s_j, o_t = \varphi_k) \\ &= \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j) \end{aligned}$$

Where, $I(s_j = x_t)$ is index function.

$$I(x_t = s_j) = \begin{cases} 1 & \text{if } x_t = s_j \\ 0 & \text{otherwise} \end{cases}$$

Substituting $\hat{\mu}_j$ for μ_j inside

$$\hat{b}_j(k) = \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k)}{\hat{\mu}_j}$$

We have:

$$\begin{aligned} \hat{b}_j(k) &= \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k)}{\hat{\mu}_j} \\ &= \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k)}{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j)} \end{aligned}$$

Evaluating the numerator, we have:

$$\begin{aligned} &\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k) \\ &= \sum_{t=1}^T \sum_X I(x_t = s_j, o_t = \varphi_k) P(X|O, \Delta_r) \\ &= \sum_{t=1}^T \sum_X I(x_t = s_j, o_t = \varphi_k) P(x_1, \dots, x_t, \dots, x_T | O, \Delta_r) \\ &= \sum_{\substack{t=1 \\ o_t=\varphi_k}}^T P(x_t = s_j | O, \Delta_r) \\ &\quad \text{(Due to total probability rule specified by formula III.1.1.4)} \\ &= \sum_{\substack{t=1 \\ o_t=\varphi_k}}^T \frac{P(O, x_t = s_j | \Delta_r)}{P(O | \Delta_r)} \\ &\quad \text{(Due to multiplication rule specified by formula III.1.1.3)} \end{aligned}$$

$$= \frac{1}{P(O|\Delta_r)} \sum_{\substack{t=1 \\ o_t=\varphi_k}}^T P(O, x_t = s_j | \Delta_r)$$

Note, the expression $\sum_{\substack{t=1 \\ o_t=\varphi_k}}^T P(O, x_t = s_j | \Delta_r)$ expresses the sum of probabilities $P(O, x_t = s_j | \Delta_r)$ over T time points with condition $o_t = \varphi_k$.

Evaluating the denominator, we have:

$$\begin{aligned} & \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j) = \sum_{t=1}^T \sum_X I(x_t = s_j) P(X|O, \Delta_r) \\ &= \sum_{t=1}^T \sum_X I(x_t = s_j) P(x_1, \dots, x_t, \dots, x_T | O, \Delta_r) \\ &= \sum_{t=1}^T P(x_t = s_j | O, \Delta_r) \end{aligned}$$

(Due to total probability rule specified by formula III.1.1.4)

$$= \sum_{t=1}^T \frac{P(O, x_t = s_j | \Delta_r)}{P(O | \Delta_r)}$$

(Due to multiplication rule specified by formula III.1.1.3)

$$= \frac{1}{P(O | \Delta_r)} \sum_{t=1}^T P(O, x_t = s_j | \Delta_r)$$

It implies

$$\hat{b}_j(k) = \frac{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j, o_t = \varphi_k)}{\sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j)} = \frac{\sum_{\substack{t=1 \\ o_t=\varphi_k}}^T P(O, x_t = s_j | \Delta_r)}{\sum_{t=1}^T P(O, x_t = s_j | \Delta_r)}$$

In general, the parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is totally determined as follows:

$$\begin{aligned} \hat{a}_{ij} &= \frac{\sum_{t=2}^T P(O, x_{t-1} = s_i, x_t = s_j | \Delta_r)}{\sum_{t=2}^T P(O, x_{t-1} = s_i | \Delta_r)} \\ \hat{b}_j(k) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_k}}^T P(O, x_t = s_j | \Delta_r)}{\sum_{t=1}^T P(O, x_t = s_j | \Delta_r)} \\ \hat{\pi}_j &= \frac{P(O, x_1 = s_j | \Delta_r)}{\sum_{i=1}^n P(O, x_1 = s_i | \Delta_r)} \end{aligned}$$

As a convention, we use notation Δ instead of Δ_r for denoting known HMM at current iteration of EM algorithm. We have formula IV.4.3.2.4 for specifying HMM parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ given current parameter $\Delta = (a_{ij}, b_j(k), \pi_j)$ as follows:

$$\begin{aligned} \hat{a}_{ij} &= \frac{\sum_{t=2}^T P(O, x_{t-1} = s_i, x_t = s_j | \Delta)}{\sum_{t=2}^T P(O, x_{t-1} = s_i | \Delta)} \\ \hat{b}_j(k) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_k}}^T P(O, x_t = s_j | \Delta)}{\sum_{t=1}^T P(O, x_t = s_j | \Delta)} \end{aligned}$$

$$\hat{\pi}_j = \frac{P(O, x_1 = s_j | \Delta)}{\sum_{i=1}^n P(O, x_1 = s_i | \Delta)}$$

Formula IV.4.3.2.4. HMM parameter estimate

The parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is the ultimate solution of the learning problem. As seen in formula IV.4.3.2.4, it is necessary to calculate probabilities $P(O, x_{t-1}=s_i, x_t=s_j)$ and $P(O, x_{t-1}=s_i)$ when other probabilities $P(O, x_t=s_j)$, $P(O, x_1=s_i)$, and $P(O, x_1=s_j)$ are represented by the joint probability γ_t specified by formula IV.4.2.1.

$$\begin{aligned} P(O, x_t = s_j | \Delta) &= \gamma_t(j) = \alpha_t(j)\beta_t(j) \\ P(O, x_1 = s_i | \Delta) &= \gamma_1(i) = \alpha_1(i)\beta_1(i) \\ P(O, x_1 = s_j | \Delta) &= \gamma_1(j) = \alpha_1(j)\beta_1(j) \end{aligned}$$

Let $\xi_t(i, j)$ is the joint probability that the stochastic process receives state s_i at time point $t-1$ and state s_j at time point t given observation sequence O (Rabiner, 1989, p. 264).

$$\xi_t(i, j) = P(O, x_{t-1} = s_i, x_t = s_j | \Delta)$$

Given forward variable α_t and backward variable β_t , if $t \geq 2$, we have:

$$\begin{aligned} &\alpha_{t-1}(i)a_{ij}b_j(o_t)\beta_t(j) \\ &= P(o_1, o_2, \dots, o_t, x_{t-1} = s_i | \Delta) * P(x_t = s_j | x_{t-1} = s_i) * b_j(o_t) * \beta_t(j) \\ &= P(o_1, o_2, \dots, o_t | x_{t-1} = s_i, \Delta) * P(x_{t-1} = s_i | \Delta) * P(x_t = s_j | x_{t-1} = s_i) * b_j(o_t) \\ &\quad * \beta_t(j) \\ &\quad (\text{Due to multiplication rule specified by formula III.1.1.3}) \\ &= P(o_1, o_2, \dots, o_t | x_{t-1} = s_i, \Delta) * P(x_t = s_j | x_{t-1} = s_i) * P(x_{t-1} = s_i | \Delta) * b_j(o_t) \\ &\quad * \beta_t(j) \\ &= P(o_1, o_2, \dots, o_t, x_t = s_j | x_{t-1} = s_i, \Delta) * P(x_{t-1} = s_i | \Delta) * b_j(o_t) * \beta_t(j) \\ &\quad (\text{Because the partial observation sequence } \{o_1, o_2, \dots, o_t\} \text{ is independent from current state } x_t \text{ given previous state } x_{t-1}) \\ &= P(o_1, o_2, \dots, o_t, x_{t-1} = s_i, x_t = s_j | \Delta) * b_j(o_t) * \beta_t(j) \\ &= P(o_1, o_2, \dots, o_t, x_{t-1} = s_i | x_t = s_j, \Delta) * P(x_t = s_j | \Delta) * b_j(o_t) * \beta_t(j) \\ &\quad (\text{Due to multiplication rule specified by formula III.1.1.3}) \\ &= P(o_1, o_2, \dots, o_t, x_{t-1} = s_i | x_t = s_j, \Delta) * P(x_t = s_j | \Delta) * P(o_t | x_t = s_j) \\ &\quad * P(o_{t+1}, o_{t+2}, \dots, o_T | x_t = s_j, \Delta) \\ &= P(o_1, o_2, \dots, o_t, x_{t-1} = s_i | x_t = s_j, \Delta) * P(x_t = s_j | \Delta) \\ &\quad * P(o_t, o_{t+1}, o_{t+2}, \dots, o_T | x_t = s_j, \Delta) \\ &\quad (\text{Because observations } o_t, o_{t+1}, o_{t+2}, \dots, o_T \text{ are mutually independent}) \\ &= P(o_1, o_2, \dots, o_t, x_{t-1} = s_i | x_t = s_j, \Delta) * P(o_t, o_{t+1}, o_{t+2}, \dots, o_T | x_t = s_j, \Delta) \\ &\quad * P(x_t = s_j | \Delta) \\ &= P(o_1, o_2, \dots, o_t, o_{t+1}, o_{t+2}, \dots, o_T, x_{t-1} = s_i | x_t = s_j, \Delta) * P(x_t = s_j | \Delta) \\ &\quad (\text{Due to multiplication rule specified by formula III.1.1.3}) \\ &= P(o_1, o_2, \dots, o_T, x_t = s_i, x_{t+1} = s_j | \Delta) \\ &\quad (\text{Due to multiplication rule specified by formula III.1.1.3}) \\ &= P(O, x_{t-1} = s_i, x_t = s_j | \Delta) = \xi_t(i, j) \end{aligned}$$

In general, formula IV.4.3.2.5 determines the joint probability $\xi_t(i, j)$ based on forward variable α_t and backward variable β_t .

$$\xi_t(i, j) = \alpha_{t-1}(i)a_{ij}b_j(o_t)\beta_t(j) \text{ where } t \geq 2$$

Formula IV.4.3.2.5. Joint probability $\xi_t(i, j)$

Where forward variable α_t and backward variable β_t are calculated by previous recurrence formulas IV.4.1.2 and IV.4.1.5.

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^n \alpha_t(i) a_{ij} \right) b_j(o_{t+1})$$

$$\beta_t(i) = \sum_{j=1}^n a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

Shortly, the joint probability $\xi_t(i, j)$ is constructed from forward variable and backward variable, as seen in figure IV.4.3.2.1 (Rabiner, 1989, p. 264).

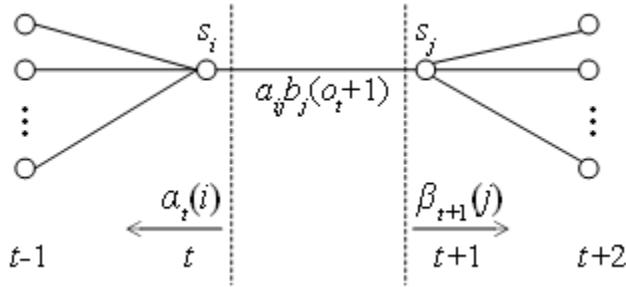


Figure IV.4.3.2.1. Construction of the joint probability $\xi_t(i, j)$

Recall that $\gamma_t(j)$ is the joint probability that the stochastic process is in state s_j at time point t with observation sequence $O = \{o_1, o_2, \dots, o_T\}$, specified by previous formula IV.4.2.1.

$$\gamma_t(j) = P(O, x_t = s_j | \Delta) = \alpha_t(j) \beta_t(j)$$

According to total probability rule specified by formula III.1.1.4, it is easy to infer that γ_t is sum of ξ_t over all states with $t \geq 2$, as seen in following formula IV.4.3.2.6.

$$\forall t \geq 2, \gamma_t(j) = \sum_{i=1}^n \xi_t(i, j) \text{ and } \gamma_{t-1}(i) = \sum_{j=1}^n \xi_t(i, j)$$

Formula IV.4.3.2.6. The γ_t is sum of ξ_t over all states

Deriving from formulas IV.4.3.2.5 and IV.4.3.2.6, we have:

$$P(O, x_{t-1} = s_i, x_t = s_j | \Delta) = \xi_t(i, j)$$

$$P(O, x_{t-1} = s_i | \Delta) = \sum_{j=1}^n \xi_t(i, j), \forall t \geq 2$$

$$P(O, x_t = s_j | \Delta) = \gamma_t(j)$$

$$P(O, x_1 = s_j | \Delta) = \gamma_1(j)$$

By extending formula IV.4.3.2.4, we receive formula IV.4.3.2.7 for specifying HMM parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_i(k), \hat{\pi}_i)$ given current parameter $\Delta = (a_{ij}, b_i(k), \pi_i)$ in detailed.

$$\hat{a}_{ij} = \frac{\sum_{t=2}^T \xi_t(i, j)}{\sum_{t=2}^T \sum_{l=1}^n \xi_t(i, l)}$$

$$\hat{b}_j(k) = \frac{\sum_{\substack{o_t=\varphi_k \\ t=1}}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

$$\hat{\pi}_j = \frac{\gamma_1(j)}{\sum_{i=1}^n \gamma_1(i)}$$

Formula IV.4.3.2.7. HMM parameter estimate in detailed

The formula [IV.4.3.2.7](#) and its proof are found in (Ramage, 2007, pp. 9-12). Followings are interpretations relevant to the joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ with observation sequence O .

- The sum $\sum_{t=2}^T \xi_t(i, j)$ expresses expected number of transitions from state s_i to state s_j (Rabiner, 1989, p. 265).
- The double sum $\sum_{t=2}^T \sum_{l=1}^n \xi_t(i, l)$ expresses expected number of transitions from state s_i (Rabiner, 1989, p. 265).
- The sum $\sum_{\substack{t=1 \\ o_t=\varphi_k}}^T \gamma_t(j)$ expresses expected number of times in state s_j and in observation φ_k (Rabiner, 1989, p. 265).
- The sum $\sum_{t=1}^T \gamma_t(j)$ expresses expected number of times in state s_j (Rabiner, 1989, p. 265).

Followings are interpretations of the parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$:

- The transition estimate \hat{a}_{ij} is expected frequency of transitions from state s_i to state s_j .
- The observation estimate $\hat{b}_j(k)$ is expected frequency of times in state s_j and in observation φ_k .
- The initial estimate $\hat{\pi}_j$ is (normalized) expected frequency of state s_j at the first time point ($t=1$).

It is easy to infer that the parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is based on joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ which, in turn, are based on current parameter $\Delta = (a_{ij}, b_j(k), \pi_j)$. The EM conditional expectation $E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}$ is determined by joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$; so, the main task of E-step in EM algorithm is essentially to calculate the joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ according to formulas [IV.4.3.2.5](#) and [IV.4.2.1](#). The EM conditional expectation $E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}$ gets maximal at estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ and so, the main task of M-step in EM algorithm is essentially to calculate $\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j$ according to formula [IV.4.3.2.7](#). The EM algorithm is interpreted in HMM learning problem, as shown in table [IV.4.3.2.1](#).

Starting with initial value for Δ , each iteration in EM algorithm has two steps:

1. *E-step*: Calculating the joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ according to formulas [IV.4.3.2.5](#) and [IV.4.2.1](#) given current parameter $\Delta = (a_{ij}, b_j(k), \pi_j)$.

$$\xi_t(i, j) = \alpha_{t-1}(i) a_{ij} b_j(o_t) \beta_t(j) \text{ where } t \geq 2$$

$$\gamma_t(j) = P(O, x_t = s_j | \Delta) = \alpha_t(j) \beta_t(j)$$

Where forward variable α_t and backward variable β_t are calculated by previous recurrence formulas [IV.4.1.2](#) and [IV.4.1.5](#).

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^n \alpha_t(i) a_{ij} \right) b_j(o_{t+1})$$

$$\beta_t(i) = \sum_{j=1}^n a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

2. *M-step*: Calculating the estimate $\widehat{\Delta} = (\widehat{a}_{ij}, \widehat{b}_j(k), \widehat{\pi}_j)$ based on the joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ determined at E-step, according to formula IV.4.3.2.7.

$$\begin{aligned}\widehat{a}_{ij} &= \frac{\sum_{t=2}^T \xi_t(i, j)}{\sum_{t=2}^T \sum_{l=1}^n \xi_t(i, l)} \\ \widehat{b}_j(k) &= \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(k)} \\ \widehat{\pi}_j &= \frac{\gamma_1(j)}{\sum_{i=1}^n \gamma_1(i)}\end{aligned}$$

The estimate $\widehat{\Delta}$ becomes the current parameter for next iteration.

EM algorithm stops when it meets the terminating condition, for example, the difference of current parameter Δ and next parameter $\widehat{\Delta}$ is insignificant. It is possible to define a custom terminating condition.

Table IV.4.3.2.1. EM algorithm for HMM learning problem

The algorithm to solve HMM learning problem shown in table IV.4.3.2.1 is known as Baum-Welch algorithm by authors Leonard E. Baum and Lloyd R. Welch (Rabiner, 1989). Please see document “Hidden Markov Models Fundamentals” by author Ramage (Ramage, 2007, pp. 8-13) for more details about HMM learning problem. As aforementioned in previous sub-section IV.4.3.1, the essence of EM algorithm applied into HMM learning problem is to determine the estimate $\widehat{\Delta} = (\widehat{a}_{ij}, \widehat{b}_j(k), \widehat{\pi}_j)$.

As seen in table IV.4.3.2.1, it is not difficult to run E-step and M-step of EM algorithm but how to determine the terminating condition is considerable problem. It is better to establish a computational terminating criterion instead of applying the general statement “EM algorithm stops when it meets the terminating condition, for example, the difference of current parameter Δ and next parameter $\widehat{\Delta}$ is insignificant”. Going back the learning problem that EM algorithm solves, the EM algorithm aims to maximize probability $P(O|\Delta)$ of given observation sequence $O=(o_1, o_2, \dots, o_T)$ so as to find out the estimate $\widehat{\Delta}$. Maximizing the probability $P(O|\Delta)$ is equivalent to maximizing the conditional expectation. So it is easy to infer that EM algorithm stops when probability $P(O|\Delta)$ approaches to maximal value and EM algorithm cannot maximize $P(O|\Delta)$ any more. In other words, the probability $P(O|\Delta)$ is terminating criterion. Calculating criterion $P(O|\Delta)$ is evaluation problem described in sub-section IV.4.1. Criterion $P(O|\Delta)$ is determined according to forward-backward procedure; please see tables IV.4.1.1 and IV.4.1.2 for more details about forward-backward procedure.

1. Initialization step: Initializing $\alpha_1(i) = b_i(o_1)\pi_i$ for all $1 \leq i \leq n$
2. Recurrence step: Calculating all $\alpha_{t+1}(j)$ for all $1 \leq j \leq n$ and $1 \leq t \leq T - 1$ according to formula IV.4.1.2.

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^n \alpha_t(i) a_{ij} \right) b_j(o_{t+1})$$

3. Evaluation step: Calculating the probability $P(O|\Delta) = \sum_{i=1}^n \alpha_T(i)$

At the end of M-step, the next criterion $P(O/\hat{\Delta})$ that is calculated based on the next parameter (also estimate) $\hat{\Delta}$ is compared with the current criterion $P(O/\Delta)$ that is calculated in the previous time. If these two criteria are the same or there is no significantly difference between them then, EM algorithm stops. This implies EM algorithm cannot maximize $P(O/\Delta)$ any more. However, calculating the next criterion $P(O/\hat{\Delta})$ according to forward-backward procedure causes EM algorithm to run slowly. This drawback is overcome by following comment and improvement. The essence of forward-backward procedure is to determine forward variables α_t while EM algorithm must calculate all forward variables and backward variables in its learning process (E-step). Thus, the evaluation of terminating condition is accelerated by executing forward-backward procedure inside the E-step of EM algorithm. In other words, when EM algorithm results out forward variables in E-step, the forward-backward procedure takes advantages of such forward variables so as to determine criterion $P(O/\Delta)$ the at the same time. As a result, the speed of EM algorithm does not decrease. However, there is always a redundant iteration; suppose that the terminating criterion approaches to maximal value at the end of the r^{th} iteration but the EM algorithm only stops at the E-step of the $(r+1)^{th}$ iteration when it really evaluates the terminating criterion. In general, the terminating criterion $P(O/\Delta)$ is calculated based on the current parameter Δ at E-step instead of the estimate $\hat{\Delta}$ at M-step. Table IV.4.3.2.2 shows the proposed implementation of EM algorithm with terminating criterion $P(O/\Delta)$. Pseudo-code like programming language C is used to describe the implementation of EM algorithm. Note, variables are marked as *italic words*, programming language keywords (*while*, *for*, *if*, *[]*, *==*, *!=*, *&&*, *//*, etc.) are marked **blue** and comments are marked **gray**. For example, notation *[]* denotes array index operation; concretely, $\alpha[t][i]$ denotes forward variable $\alpha_t(i)$ at time point t with regard to state s_i .

Input:

HMM with current parameter $\Delta = \{a_{ij}, \pi_j, b_{jk}\}$

Observation sequence $O = \{o_1, o_2, \dots, o_T\}$

Output:

HMM with optimized parameter $\Delta = \{a_{ij}, \pi_j, b_{jk}\}$

Allocating memory for two matrices α and β representing forward variables and backward variables.

previous_criterion = -1

current_criterion = -1

iteration = 0

//Pre-defined number *MAX_ITERATION* is used to prevent from infinite loop.

MAX_ITERATION = 10000

While (*iteration* < *MAX_ITERATION*)

//Calculating forward variables and backward variables

For *t* = 1 **to** *T*

For *i* = 1 **to** *n*

Calculating forward variables $\alpha[t][i]$ and backward variables $\beta[T-t+1][i]$ based on observation sequence O according to formulas IV.4.1.2 and IV.4.1.5.

End for *i*

End for *t*

```

//Calculating terminating criterion current_criterion =  $P(O/\Delta)$ 
current_criterion = 0
For i = 1 to n
    current_criterion = current_criterion +  $\alpha[T][i]$ 
End for i

//Terminating condition
If previous_criterion >= 0 && previous_criterion == current_criterion then
    break //breaking out the loop, the algorithm stops
Else
    previous_criterion = current_criterion
End if

//Updating transition probability matrix
For i = 1 to n
    denominator = 0
    Allocating numerators as a 1-dimension array including n zero elements.
    For t = 2 to T
        For k = 1 to n
             $\zeta = \alpha[t-1][i] * a_{ik} * b_k(o_t) * \beta[t][k]$ 
            numerators[k] = numerators[k] +  $\zeta$ 
            denominator = denominator +  $\zeta$ 
        End for k
    End for t
    If denominator != 0 then
        For j = 1 to n
             $a_{ij} = \text{numerators}[j] / \text{denominator}$ 
        End for j
    End if
End for i

//Updating initial probability matrix
Allocating g as a 1-dimension array including n elements.
sum = 0
For j = 1 to n
    g[j] =  $\alpha[1][j] * \beta[1][j]$ 
    sum = sum + g[j]
End for j

If sum != 0 then
    For j = 1 to n
         $\pi_j = g[j] / \text{sum}$ 
    End for j
End if

//Updating observation probability distribution
For j = 1 to n

```

```

Allocating  $\gamma$  as a 1-dimension array including  $T$  elements.
denominator = 0
For t = 1 to T
     $\gamma[t] = \alpha[t][j] * \beta[t][j]$ 
    denominator = denominator +  $\gamma[t]$ 
End for t

Let m be the columns of observation distribution matrix B.
For k = 1 to m
    numerator = 0
    For t = 1 to T
        If  $o_t == k$  then
            numerator = numerator +  $\gamma[t]$ 
        End if
    End for t

     $b_{jk} = numerator / denominator$ 
End for k
End for j

iteration = iteration + 1
End while

```

Table IV.4.3.2.2. Proposed implementation of EM algorithm for learning HMM with terminating criterion $P(O/\Delta)$

According to table IV.4.3.2.2, the number of iterations is limited by a pre-defined maximum number, which aims to solve a so-called infinite loop optimization. Although it is proved that EM algorithm always converges, maybe there are two different estimates $\hat{\Delta}_1$ and $\hat{\Delta}_2$ at the final convergence. This situation causes EM algorithm to alternate between $\hat{\Delta}_1$ and $\hat{\Delta}_2$ in infinite loop. Therefore, the final estimate $\hat{\Delta}_1$ or $\hat{\Delta}_2$ is totally determined but the EM algorithm does not stop. This is the reason that the number of iterations is limited by a pre-defined maximum number.

Going back given weather HMM Δ whose parameters A , B , and Π are specified in tables IV.4.1, IV.4.2, and IV.4.3, suppose observation sequence is $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$, the EM algorithm and its implementation described in tables IV.4.3.2.1 and IV.4.3.2.2 are applied into calculating the parameter estimate $\hat{\Delta} = (\hat{\alpha}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ which is the ultimate solution of the learning problem, as below.

At the first iteration ($r=1$) we have:

$$\begin{aligned}
\alpha_1(1) &= b_1(o_1 = \varphi_4)\pi_1 = b_{14}\pi_1 = 0.05 * 0.33 = 0.0165 \\
\alpha_1(2) &= b_2(o_1 = \varphi_4)\pi_2 = b_{24}\pi_2 = 0.25 * 0.33 = 0.0825 \\
\alpha_1(3) &= b_3(o_1 = \varphi_4)\pi_3 = b_{34}\pi_3 = 0.5 * 0.33 = 0.165 \\
\alpha_2(1) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_1(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_{11} \\
&= (\alpha_1(1)a_{11} + \alpha_1(2)a_{21} + \alpha_1(3)a_{31})b_{11} \\
&= (0.0165 * 0.5 + 0.0825 * 0.3 + 0.165 * 0.25) * 0.6 = 0.04455
\end{aligned}$$

$$\begin{aligned}
 \alpha_2(2) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_2(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_{21} \\
 &= (\alpha_1(1)a_{12} + \alpha_1(2)a_{22} + \alpha_1(3)a_{32})b_{21} \\
 &= (0.0165 * 0.25 + 0.0825 * 0.4 + 0.165 * 0.25) * 0.25 = 0.019594 \\
 \alpha_2(3) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_3(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_{31} \\
 &= (\alpha_1(1)a_{13} + \alpha_1(2)a_{23} + \alpha_1(3)a_{33})b_{31} \\
 &= (0.0165 * 0.25 + 0.0825 * 0.3 + 0.165 * 0.5) * 0.05 = 0.005569 \\
 \alpha_3(1) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_1(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_{12} \\
 &= (\alpha_2(1)a_{11} + \alpha_2(2)a_{21} + \alpha_2(3)a_{31})b_{12} \\
 &= (0.04455 * 0.5 + 0.019594 * 0.3 + 0.005569 * 0.25) * 0.2 \\
 &= 0.005909 \\
 \alpha_3(2) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_2(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_{22} \\
 &= (\alpha_2(1)a_{12} + \alpha_2(2)a_{22} + \alpha_2(3)a_{32})b_{22} \\
 &= (0.04455 * 0.25 + 0.019594 * 0.4 + 0.005569 * 0.25) * 0.25 \\
 &= 0.005092 \\
 \alpha_3(3) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i3} \right) b_3(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i3} \right) b_{32} \\
 &= (\alpha_2(1)a_{13} + \alpha_2(2)a_{23} + \alpha_2(3)a_{33})b_{32} \\
 &= (0.04455 * 0.25 + 0.019594 * 0.3 + 0.005569 * 0.5) * 0.1 \\
 &= 0.00198 \\
 \beta_3(1) &= \beta_3(2) = \beta_3(3) = 1 \\
 \beta_2(1) &= \sum_{j=1}^n a_{1j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{1j} b_{j2} \beta_3(j) \\
 &= a_{11} b_{12} \beta_3(1) + a_{12} b_{22} \beta_3(2) + a_{13} b_{32} \beta_3(3) \\
 &= 0.5 * 0.2 * 1 + 0.25 * 0.25 * 1 + 0.25 * 0.1 * 1 = 0.1875 \\
 \beta_2(2) &= \sum_{j=1}^n a_{2j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{2j} b_{j2} \beta_3(j) \\
 &= a_{21} b_{12} \beta_3(1) + a_{22} b_{22} \beta_3(2) + a_{23} b_{32} \beta_3(3) \\
 &= 0.3 * 0.2 * 1 + 0.4 * 0.25 * 1 + 0.3 * 0.1 * 1 = 0.19 \\
 \beta_2(3) &= \sum_{j=1}^n a_{3j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{3j} b_{j2} \beta_3(j) \\
 &= a_{31} b_{12} \beta_3(1) + a_{32} b_{22} \beta_3(2) + a_{33} b_{32} \beta_3(3) \\
 &= 0.25 * 0.2 * 1 + 0.25 * 0.25 * 1 + 0.5 * 0.1 * 1 = 0.1625 \\
 \beta_1(1) &= \sum_{j=1}^n a_{1j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{1j} b_{j1} \beta_2(j) \\
 &= a_{11} b_{11} \beta_2(1) + a_{12} b_{21} \beta_2(2) + a_{13} b_{31} \beta_2(3) \\
 &= 0.5 * 0.6 * 0.1875 + 0.25 * 0.25 * 0.19 + 0.25 * 0.05 * 0.1625 \\
 &= 0.070156
 \end{aligned}$$

$$\begin{aligned}
 \beta_1(2) &= \sum_{j=1}^n a_{2j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{2j} b_{j1} \beta_2(j) \\
 &= a_{21} b_{11} \beta_2(1) + a_{22} b_{21} \beta_2(2) + a_{23} b_{31} \beta_2(3) \\
 &= 0.3 * 0.6 * 0.1875 + 0.4 * 0.25 * 0.19 + 0.3 * 0.05 * 0.1625 \\
 &= 0.055188 \\
 \beta_1(3) &= \sum_{j=1}^n a_{3j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{3j} b_{j1} \beta_2(j) \\
 &= a_{31} b_{11} \beta_2(1) + a_{32} b_{21} \beta_2(2) + a_{33} b_{31} \beta_2(3) \\
 &= 0.25 * 0.6 * 0.1875 + 0.25 * 0.25 * 0.19 + 0.5 * 0.05 * 0.1625 \\
 &= 0.044063
 \end{aligned}$$

Within the E-step of the first iteration ($r=1$), the terminating criterion $P(O|\Delta)$ is calculated according to forward-backward procedure (see table IV.4.1.1) as follows:

$$P(O|\Delta) = \alpha_3(1) + \alpha_3(2) + \alpha_3(3) = 0.005909 + 0.005092 + 0.00198 \approx 0.013$$

Within the E-step of the first iteration ($r=1$), the joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ are calculated based on formulas IV.4.3.2.5 and IV.4.2.1 as follows:

$$\begin{aligned}
 \xi_2(1,1) &= \alpha_1(1)a_{11}b_1(o_2 = \varphi_1)\beta_2(1) = \alpha_1(1)a_{11}b_{11}\beta_2(1) \\
 &= 0.0165 * 0.5 * 0.6 * 0.1875 = 0.000928 \\
 \xi_2(1,2) &= \alpha_1(1)a_{12}b_2(o_2 = \varphi_1)\beta_2(2) = \alpha_1(1)a_{12}b_{21}\beta_2(2) \\
 &= 0.0165 * 0.25 * 0.25 * 0.19 = 0.000196 \\
 \xi_2(1,3) &= \alpha_1(1)a_{13}b_3(o_2 = \varphi_1)\beta_2(3) = \alpha_1(1)a_{13}b_{31}\beta_2(3) \\
 &= 0.0165 * 0.25 * 0.05 * 0.1625 = 0.000034 \\
 \xi_2(2,1) &= \alpha_1(2)a_{21}b_1(o_2 = \varphi_1)\beta_2(1) = \alpha_1(2)a_{21}b_{11}\beta_2(1) \\
 &= 0.0825 * 0.3 * 0.6 * 0.1875 = 0.002784 \\
 \xi_2(2,2) &= \alpha_1(2)a_{22}b_2(o_2 = \varphi_1)\beta_2(2) = \alpha_1(2)a_{22}b_{21}\beta_2(2) \\
 &= 0.0825 * 0.4 * 0.25 * 0.19 = 0.001568 \\
 \xi_2(2,3) &= \alpha_1(2)a_{23}b_3(o_2 = \varphi_1)\beta_2(3) = \alpha_1(2)a_{23}b_{31}\beta_2(3) \\
 &= 0.0825 * 0.3 * 0.05 * 0.1625 = 0.000201 \\
 \xi_2(3,1) &= \alpha_1(3)a_{31}b_1(o_2 = \varphi_1)\beta_2(1) = \alpha_1(3)a_{31}b_{11}\beta_2(1) \\
 &= 0.165 * 0.25 * 0.6 * 0.1875 = 0.004641 \\
 \xi_2(3,2) &= \alpha_1(3)a_{32}b_2(o_2 = \varphi_1)\beta_2(2) = \alpha_1(3)a_{32}b_{21}\beta_2(2) \\
 &= 0.165 * 0.25 * 0.25 * 0.19 = 0.001959 \\
 \xi_2(3,3) &= \alpha_1(3)a_{33}b_3(o_2 = \varphi_1)\beta_2(3) = \alpha_1(3)a_{33}b_{31}\beta_2(3) \\
 &= 0.165 * 0.5 * 0.05 * 0.1625 = 0.00067 \\
 \xi_3(1,1) &= \alpha_2(1)a_{11}b_1(o_3 = \varphi_2)\beta_3(1) = \alpha_2(1)a_{11}b_{12}\beta_3(1) \\
 &= 0.04455 * 0.5 * 0.2 * 1 \approx 0.004455 \\
 \xi_3(1,2) &= \alpha_2(1)a_{12}b_2(o_3 = \varphi_2)\beta_3(2) = \alpha_2(1)a_{12}b_{22}\beta_3(2) \\
 &= 0.04455 * 0.25 * 0.25 * 1 = 0.002784 \\
 \xi_3(1,3) &= \alpha_2(1)a_{13}b_3(o_3 = \varphi_2)\beta_3(3) = \alpha_2(1)a_{13}b_{32}\beta_3(3) \\
 &= 0.04455 * 0.25 * 0.1 * 1 = 0.001114 \\
 \xi_3(2,1) &= \alpha_2(2)a_{21}b_1(o_3 = \varphi_2)\beta_3(1) = \alpha_2(2)a_{21}b_{12}\beta_3(1) \\
 &= 0.019594 * 0.3 * 0.2 * 1 = 0.001176 \\
 \xi_3(2,2) &= \alpha_2(2)a_{22}b_2(o_3 = \varphi_2)\beta_3(2) = \alpha_2(2)a_{22}b_{22}\beta_3(2) \\
 &= 0.019594 * 0.4 * 0.25 * 1 = 0.001959 \\
 \xi_3(2,3) &= \alpha_2(2)a_{23}b_3(o_3 = \varphi_2)\beta_3(3) = \alpha_2(2)a_{23}b_{32}\beta_3(3) \\
 &= 0.019594 * 0.3 * 0.1 * 1 = 0.000588 \\
 \xi_3(3,1) &= \alpha_2(3)a_{31}b_1(o_3 = \varphi_2)\beta_3(1) = \alpha_2(3)a_{31}b_{12}\beta_3(1) \\
 &= 0.005569 * 0.25 * 0.2 * 1 = 0.000278
 \end{aligned}$$

$$\begin{aligned}\xi_3(3,2) &= \alpha_2(3)a_{32}b_2(o_3 = \varphi_2)\beta_3(2) = \alpha_2(3)a_{32}b_{22}\beta_3(2) \\ &= 0.005569 * 0.25 * 0.25 * 1 = 0.000348\end{aligned}$$

$$\begin{aligned}\xi_3(3,3) &= \alpha_2(3)a_{33}b_3(o_3 = \varphi_2)\beta_3(3) = \alpha_2(3)a_{33}b_{32}\beta_3(3) \\ &= 0.005569 * 0.5 * 0.1 * 1 = 0.000278\end{aligned}$$

$$\gamma_1(1) = \alpha_1(1)\beta_1(1) = 0.0165 * 0.070156 = 0.001158$$

$$\gamma_1(2) = \alpha_1(2)\beta_1(2) = 0.0825 * 0.055188 = 0.004553$$

$$\gamma_1(3) = \alpha_1(3)\beta_1(3) = 0.165 * 0.044063 = 0.00727$$

$$\gamma_2(1) = \alpha_2(1)\beta_2(1) = 0.04455 * 0.1875 = 0.008353$$

$$\gamma_2(2) = \alpha_2(2)\beta_2(2) = 0.019594 * 0.19 = 0.003723$$

$$\gamma_2(3) = \alpha_2(3)\beta_2(3) = 0.005569 * 0.1625 = 0.000905$$

$$\gamma_3(1) = \alpha_3(1)\beta_3(1) = 0.005909 * 1 = 0.005909$$

$$\gamma_3(2) = \alpha_3(2)\beta_3(2) = 0.005092 * 1 = 0.005092$$

$$\gamma_3(3) = \alpha_3(3)\beta_3(3) = 0.00198 * 1 = 0.00198$$

Within the M-step of the first iteration ($r=1$), the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is calculated based on joint probabilities $\xi_i(i,j)$ and $\gamma_l(j)$ determined at E-step.

$$\begin{aligned}\hat{a}_{11} &= \frac{\sum_{t=2}^3 \xi_t(1,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} \\ &= \frac{\xi_2(1,1) + \xi_3(1,1)}{\xi_2(1,1) + \xi_2(1,2) + \xi_2(1,3) + \xi_3(1,1) + \xi_3(1,2) + \xi_3(1,3)} \\ &= \frac{(0.000928 + 0.004455)}{(0.000928 + 0.000196 + 0.000034 + 0.004455 + 0.002784 + 0.001114)} \approx 0.5660\end{aligned}$$

$$\begin{aligned}\hat{a}_{12} &= \frac{\sum_{t=2}^3 \xi_t(1,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} \\ &= \frac{\xi_2(1,2) + \xi_3(1,2)}{\xi_2(1,1) + \xi_2(1,2) + \xi_2(1,3) + \xi_3(1,1) + \xi_3(1,2) + \xi_3(1,3)} \\ &= \frac{(0.000196 + 0.002784)}{(0.000928 + 0.000196 + 0.000034 + 0.004455 + 0.002784 + 0.001114)} \approx 0.3134\end{aligned}$$

$$\begin{aligned}\hat{a}_{13} &= \frac{\sum_{t=2}^3 \xi_t(1,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} \\ &= \frac{\xi_2(1,3) + \xi_3(1,3)}{\xi_2(1,1) + \xi_2(1,2) + \xi_2(1,3) + \xi_3(1,1) + \xi_3(1,2) + \xi_3(1,3)} \\ &= \frac{(0.000034 + 0.001114)}{(0.000928 + 0.000196 + 0.000034 + 0.004455 + 0.002784 + 0.001114)} \approx 0.1206\end{aligned}$$

$$\begin{aligned}\hat{a}_{21} &= \frac{\sum_{t=2}^3 \xi_t(2,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} \\ &= \frac{\xi_2(2,1) + \xi_3(2,1)}{\xi_2(2,1) + \xi_2(2,2) + \xi_2(2,3) + \xi_3(2,1) + \xi_3(2,2) + \xi_3(2,3)} \\ &= \frac{(0.002784 + 0.001176)}{(0.002784 + 0.001568 + 0.000201 + 0.001176 + 0.001959 + 0.000588)} \approx 0.4785\end{aligned}$$

$$\begin{aligned}
 \hat{a}_{22} &= \frac{\sum_{t=2}^3 \xi_t(2,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} \\
 &= \frac{\xi_2(2,2) + \xi_3(2,2)}{\xi_2(2,1) + \xi_2(2,2) + \xi_2(2,3) + \xi_3(2,1) + \xi_3(2,2) + \xi_3(2,3)} \\
 &= \frac{(0.001568 + 0.001959)}{(0.002784 + 0.001568 + 0.000201 + 0.001176 + 0.001959} \\
 &\quad + 0.000588) \approx 0.4262 \\
 \hat{a}_{23} &= \frac{\sum_{t=2}^3 \xi_t(2,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} \\
 &= \frac{\xi_2(2,3) + \xi_3(2,3)}{\xi_2(2,1) + \xi_2(2,2) + \xi_2(2,3) + \xi_3(2,1) + \xi_3(2,2) + \xi_3(2,3)} \\
 &= \frac{(0.000201 + 0.000588)}{(0.002784 + 0.001568 + 0.000201 + 0.001176 + 0.001959} \\
 &\quad + 0.000588) \approx 0.0953 \\
 \hat{a}_{31} &= \frac{\sum_{t=2}^3 \xi_t(3,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} \\
 &= \frac{\xi_2(3,1) + \xi_3(3,1)}{\xi_2(3,1) + \xi_2(3,2) + \xi_2(3,3) + \xi_3(3,1) + \xi_3(3,2) + \xi_3(3,3)} \\
 &= \frac{(0.004641 + 0.000278)}{(0.004641 + 0.001959 + 0.00067 + 0.000278 + 0.000348} \\
 &\quad + 0.000278) \approx 0.6017 \\
 \hat{a}_{32} &= \frac{\sum_{t=2}^3 \xi_t(3,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} \\
 &= \frac{\xi_2(3,2) + \xi_3(3,2)}{\xi_2(3,1) + \xi_2(3,2) + \xi_2(3,3) + \xi_3(3,1) + \xi_3(3,2) + \xi_3(3,3)} \\
 &= \frac{(0.001959 + 0.000348)}{(0.004641 + 0.001959 + 0.00067 + 0.000278 + 0.000348} \\
 &\quad + 0.000278) \approx 0.2822 \\
 \hat{a}_{33} &= \frac{\sum_{t=2}^3 \xi_t(3,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} \\
 &= \frac{\xi_2(3,3) + \xi_3(3,3)}{\xi_2(3,1) + \xi_2(3,2) + \xi_2(3,3) + \xi_3(3,1) + \xi_3(3,2) + \xi_3(3,3)} \\
 &= \frac{(0.00067 + 0.000278)}{(0.004641 + 0.001959 + 0.00067 + 0.000278 + 0.000348} \\
 &\quad + 0.000278) \approx 0.1161
 \end{aligned}$$

$$\begin{aligned}
 \hat{b}_1(1) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_1}}^3 \gamma_t(1)}{\sum_{t=1}^3 \gamma_t(1)} = \frac{\gamma_2(1)}{\gamma_1(1) + \gamma_2(1) + \gamma_3(1)} \\
 &= 0.008353 / (0.001158 + 0.008353 + 0.005909) \approx 0.5417 \\
 \hat{b}_1(2) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_2}}^3 \gamma_t(1)}{\sum_{t=1}^3 \gamma_t(1)} = \frac{\gamma_3(1)}{\gamma_1(1) + \gamma_2(1) + \gamma_3(1)} \\
 &= 0.005909 / (0.001158 + 0.008353 + 0.005909) \approx 0.3832 \\
 \hat{b}_1(3) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_3}}^3 \gamma_t(1)}{\sum_{t=1}^3 \gamma_t(1)} = \frac{0}{\gamma_1(1) + \gamma_2(1) + \gamma_3(1)} = 0
 \end{aligned}$$

$$\begin{aligned}
 \hat{b}_1(4) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_4}}^3 \gamma_t(1)}{\sum_{t=1}^3 \gamma_t(1)} = \frac{\gamma_1(1)}{\gamma_1(1) + \gamma_2(1) + \gamma_3(1)} \\
 &= 0.001158 / (0.001158 + 0.008353 + 0.005909) \approx 0.0751 \\
 \hat{b}_2(1) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_1}}^3 \gamma_t(2)}{\sum_{t=1}^3 \gamma_t(2)} = \frac{\gamma_2(2)}{\gamma_1(2) + \gamma_2(2) + \gamma_3(2)} \\
 &= 0.003723 / (0.004553 + 0.003723 + 0.005092) \approx 0.2785 \\
 \hat{b}_2(2) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_2}}^3 \gamma_t(2)}{\sum_{t=1}^3 \gamma_t(2)} = \frac{\gamma_3(2)}{\gamma_1(2) + \gamma_2(2) + \gamma_3(2)} \\
 &= 0.005092 / (0.004553 + 0.003723 + 0.005092) \approx 0.3809 \\
 \hat{b}_2(3) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_3}}^3 \gamma_t(2)}{\sum_{t=1}^3 \gamma_t(2)} = \frac{0}{\gamma_1(2) + \gamma_2(2) + \gamma_3(2)} = 0 \\
 \hat{b}_2(4) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_4}}^3 \gamma_t(2)}{\sum_{t=1}^3 \gamma_t(2)} = \frac{\gamma_1(2)}{\gamma_1(2) + \gamma_2(2) + \gamma_3(2)} \\
 &= 0.004553 / (0.004553 + 0.003723 + 0.005092) \approx 0.3406 \\
 \hat{b}_3(1) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_1}}^3 \gamma_t(3)}{\sum_{t=1}^3 \gamma_t(3)} = \frac{\gamma_2(3)}{\gamma_1(3) + \gamma_2(3) + \gamma_3(3)} \\
 &= 0.000905 / (0.00727 + 0.000905 + 0.00198) \approx 0.0891 \\
 \hat{b}_3(2) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_2}}^3 \gamma_t(3)}{\sum_{t=1}^3 \gamma_t(3)} = \frac{\gamma_3(3)}{\gamma_1(3) + \gamma_2(3) + \gamma_3(3)} \\
 &= 0.00198 / (0.00727 + 0.000905 + 0.00198) \approx 0.1950 \\
 \hat{b}_3(3) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_3}}^3 \gamma_t(3)}{\sum_{t=1}^3 \gamma_t(3)} = \frac{0}{\gamma_1(3) + \gamma_2(3) + \gamma_3(3)} = 0 \\
 \hat{b}_3(4) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_4}}^3 \gamma_t(3)}{\sum_{t=1}^3 \gamma_t(3)} = \frac{\gamma_1(3)}{\gamma_1(3) + \gamma_2(3) + \gamma_3(3)} \\
 &= 0.00727 / (0.00727 + 0.000905 + 0.00198) \approx 0.7159
 \end{aligned}$$

$$\begin{aligned}
 \hat{\pi}_1 &= \frac{\gamma_1(1)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.001158 / (0.001158 + 0.004553 + 0.00727) \\
 &\approx 0.0892 \\
 \hat{\pi}_2 &= \frac{\gamma_1(2)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.004553 / (0.001158 + 0.004553 + 0.00727) \\
 &\approx 0.3507 \\
 \hat{\pi}_3 &= \frac{\gamma_1(3)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.00727 / (0.001158 + 0.004553 + 0.00727) \\
 &\approx 0.5601
 \end{aligned}$$

At the second iteration ($r=2$), the current parameter $\Delta = (a_{ij}, b_j(k), \pi_j)$ is received values from the previous estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$, as seen in table IV.4.3.2.3.

$a_{11} = \hat{a}_{11} = 0.5660$	$a_{12} = \hat{a}_{12} = 0.3134$	$a_{13} = \hat{a}_{13} = 0.1206$
$a_{21} = \hat{a}_{21} = 0.4785$	$a_{22} = \hat{a}_{22} = 0.4262$	$a_{23} = \hat{a}_{23} = 0.0953$

$a_{31} = \hat{a}_{31} = 0.6017$	$a_{32} = \hat{a}_{32} = 0.2822$	$a_{33} = \hat{a}_{33} = 0.1161$	
$b_{11} = \hat{b}_1(1) = 0.5417$	$b_{12} = \hat{b}_1(2) = 0.3832$	$b_{13} = \hat{b}_1(3) = 0$	$b_{14} = \hat{b}_1(4) = 0.0751$
$b_{21} = \hat{b}_2(1) = 0.2785$	$b_{22} = \hat{b}_2(2) = 0.3809$	$b_{23} = \hat{b}_2(3) = 0$	$b_{24} = \hat{b}_2(4) = 0.3406$
$b_{31} = \hat{b}_3(1) = 0.0891$	$b_{32} = \hat{b}_3(2) = 0.1950$	$b_{33} = \hat{b}_3(3) = 0$	$b_{34} = \hat{b}_3(4) = 0.7159$
$\pi_1 = \hat{\pi}_1 = 0.0892$		$\pi_2 = \hat{\pi}_2 = 0.3507$	$\pi_3 = \hat{\pi}_3 = 0.5601$
Terminating criterion $P(O/\Delta) = 0.013$			

Table IV.4.3.2.3. HMM parameters resulted from the first iteration of EM algorithm

We have:

$$\begin{aligned}
 \alpha_1(1) &= b_1(o_1 = \varphi_4)\pi_1 = b_{14}\pi_1 = 0.0751 * 0.0892 = 0.006699 \\
 \alpha_1(2) &= b_2(o_1 = \varphi_4)\pi_2 = b_{24}\pi_2 = 0.3406 * 0.3507 = 0.119448 \\
 \alpha_1(3) &= b_3(o_1 = \varphi_4)\pi_3 = b_{34}\pi_3 = 0.7159 * 0.5601 = 0.400976 \\
 \alpha_2(1) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_1(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_{11} \\
 &= (\alpha_1(1)a_{11} + \alpha_1(2)a_{21} + \alpha_1(3)a_{31})b_{11} \\
 &= (0.006699 * 0.5660 + 0.119448 * 0.4785 + 0.400976 * 0.6017) \\
 &\quad * 0.5417 = 0.16371 \\
 \alpha_2(2) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_2(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_{21} \\
 &= (\alpha_1(1)a_{12} + \alpha_1(2)a_{22} + \alpha_1(3)a_{32})b_{21} \\
 &= (0.006699 * 0.3134 + 0.119448 * 0.4262 + 0.400976 * 0.2822) \\
 &\quad * 0.2785 = 0.046277 \\
 \alpha_2(3) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_3(o_2 = \varphi_1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_{31} \\
 &= (\alpha_1(1)a_{13} + \alpha_1(2)a_{23} + \alpha_1(3)a_{33})b_{31} \\
 &= (0.006699 * 0.1206 + 0.119448 * 0.0953 + 0.400976 * 0.1161) \\
 &\quad * 0.0891 = 0.005234 \\
 \alpha_3(1) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_1(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_{12} \\
 &= (\alpha_2(1)a_{11} + \alpha_2(2)a_{21} + \alpha_2(3)a_{31})b_{12} \\
 &= (0.16371 * 0.5660 + 0.046277 * 0.4785 + 0.005234 * 0.6017) \\
 &\quad * 0.3832 = 0.045199 \\
 \alpha_3(2) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_2(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_{22} \\
 &= (\alpha_2(1)a_{12} + \alpha_2(2)a_{22} + \alpha_2(3)a_{32})b_{22} \\
 &= (0.16371 * 0.3134 + 0.046277 * 0.4262 + 0.005234 * 0.2822) \\
 &\quad * 0.3809 = 0.027618
 \end{aligned}$$

$$\begin{aligned}
 \alpha_3(3) &= \left(\sum_{i=1}^3 \alpha_2(i) a_{i3} \right) b_3(o_3 = \varphi_2) = \left(\sum_{i=1}^3 \alpha_2(i) a_{i3} \right) b_{32} \\
 &= (\alpha_2(1)a_{13} + \alpha_2(2)a_{23} + \alpha_2(3)a_{33})b_{32} \\
 &= (0.16371 * 0.1206 + 0.046277 * 0.0953 + 0.005234 * 0.1161) \\
 &\quad * 0.1950 = 0.004828 \\
 \beta_3(1) &= \beta_3(2) = \beta_3(3) = 1 \\
 \beta_2(1) &= \sum_{j=1}^n a_{1j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{1j} b_{j2} \beta_3(j) \\
 &= a_{11} b_{12} \beta_3(1) + a_{12} b_{22} \beta_3(2) + a_{13} b_{32} \beta_3(3) \\
 &= 0.5660 * 0.3832 * 1 + 0.3134 * 0.3809 * 1 + 0.1206 * 0.1950 * 1 \\
 &= 0.359782 \\
 \beta_2(2) &= \sum_{j=1}^n a_{2j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{2j} b_{j2} \beta_3(j) \\
 &= a_{21} b_{12} \beta_3(1) + a_{22} b_{22} \beta_3(2) + a_{23} b_{32} \beta_3(3) \\
 &= 0.4785 * 0.3832 * 1 + 0.4262 * 0.3809 * 1 + 0.0953 * 0.1950 * 1 \\
 &= 0.364284 \\
 \beta_2(3) &= \sum_{j=1}^n a_{3j} b_j(o_3 = \varphi_2) \beta_3(j) = \sum_{j=1}^n a_{3j} b_{j2} \beta_3(j) \\
 &= a_{31} b_{12} \beta_3(1) + a_{32} b_{22} \beta_3(2) + a_{33} b_{32} \beta_3(3) \\
 &= 0.6017 * 0.3832 * 1 + 0.2822 * 0.3809 * 1 + 0.1161 * 0.1950 * 1 \\
 &= 0.360701 \\
 \beta_1(1) &= \sum_{j=1}^n a_{1j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{1j} b_{j1} \beta_2(j) \\
 &= a_{11} b_{11} \beta_2(1) + a_{12} b_{21} \beta_2(2) + a_{13} b_{31} \beta_2(3) \\
 &= 0.5660 * 0.5417 * 0.359782 + 0.3134 * 0.2785 * 0.364284 \\
 &\quad + 0.1206 * 0.0891 * 0.360701 = 0.145981 \\
 \beta_1(2) &= \sum_{j=1}^n a_{2j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{2j} b_{j1} \beta_2(j) \\
 &= a_{21} b_{11} \beta_2(1) + a_{22} b_{21} \beta_2(2) + a_{23} b_{31} \beta_2(3) \\
 &= 0.4785 * 0.5417 * 0.359782 + 0.4262 * 0.2785 * 0.364284 \\
 &\quad + 0.0953 * 0.0891 * 0.360701 = 0.139559 \\
 \beta_1(3) &= \sum_{j=1}^n a_{3j} b_j(o_2 = \varphi_1) \beta_2(j) = \sum_{j=1}^n a_{3j} b_{j1} \beta_2(j) \\
 &= a_{31} b_{11} \beta_2(1) + a_{32} b_{21} \beta_2(2) + a_{33} b_{31} \beta_2(3) \\
 &= 0.6017 * 0.5417 * 0.359782 + 0.2822 * 0.2785 * 0.364284 \\
 &\quad + 0.1161 * 0.0891 * 0.360701 = 0.149629
 \end{aligned}$$

Similarly, within the E-step of the second iteration ($r=2$), the terminating criterion $P(O|\Delta)$ is calculated according to forward-backward procedure (see table IV.4.1.1) as follows:

$P(O|\Delta) = \alpha_3(1) + \alpha_3(2) + \alpha_3(3) = 0.045199 + 0.027618 + 0.004828 \approx 0.0776$
 Within the E-step of the second iteration ($r=2$), the joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ are calculated based on formulas IV.4.3.2.5 and IV.4.2.1 as follows:

$$\begin{aligned}
 \xi_2(1,1) &= \alpha_1(1)a_{11}b_1(o_2 = \varphi_1)\beta_2(1) = \alpha_1(1)a_{11}b_{11}\beta_2(1) \\
 &= 0.006699 * 0.5660 * 0.5417 * 0.359782 = 0.000739 \\
 \xi_2(1,2) &= \alpha_1(1)a_{12}b_2(o_2 = \varphi_1)\beta_2(2) = \alpha_1(1)a_{12}b_{21}\beta_2(2) \\
 &= 0.006699 * 0.3134 * 0.2785 * 0.364284 = 0.000213 \\
 \xi_2(1,3) &= \alpha_1(1)a_{13}b_3(o_2 = \varphi_1)\beta_2(3) = \alpha_1(1)a_{13}b_{31}\beta_2(3) \\
 &= 0.006699 * 0.1206 * 0.0891 * 0.360701 = 0.000026 \\
 \xi_2(2,1) &= \alpha_1(2)a_{21}b_1(o_2 = \varphi_1)\beta_2(1) = \alpha_1(2)a_{21}b_{11}\beta_2(1) \\
 &= 0.119448 * 0.4785 * 0.5417 * 0.359782 = 0.011139 \\
 \xi_2(2,2) &= \alpha_1(2)a_{22}b_2(o_2 = \varphi_1)\beta_2(2) = \alpha_1(2)a_{22}b_{21}\beta_2(2) \\
 &= 0.119448 * 0.4262 * 0.2785 * 0.364284 = 0.005165 \\
 \xi_2(2,3) &= \alpha_1(2)a_{23}b_3(o_2 = \varphi_1)\beta_2(3) = \alpha_1(2)a_{23}b_{31}\beta_2(3) \\
 &= 0.119448 * 0.0953 * 0.0891 * 0.360701 = 0.000366 \\
 \xi_2(3,1) &= \alpha_1(3)a_{31}b_1(o_2 = \varphi_1)\beta_2(1) = \alpha_1(3)a_{31}b_{11}\beta_2(1) \\
 &= 0.400976 * 0.6017 * 0.5417 * 0.359782 = 0.047022 \\
 \xi_2(3,2) &= \alpha_1(3)a_{32}b_2(o_2 = \varphi_1)\beta_2(2) = \alpha_1(3)a_{32}b_{21}\beta_2(2) \\
 &= 0.400976 * 0.2822 * 0.2785 * 0.364284 = 0.011148 \\
 \xi_2(3,3) &= \alpha_1(3)a_{33}b_3(o_2 = \varphi_1)\beta_2(3) = \alpha_1(3)a_{33}b_{31}\beta_2(3) \\
 &= 0.400976 * 0.1161 * 0.0891 * 0.360701 = 0.001496 \\
 \xi_3(1,1) &= \alpha_2(1)a_{11}b_1(o_3 = \varphi_2)\beta_3(1) = \alpha_2(1)a_{11}b_{12}\beta_3(1) \\
 &= 0.16371 * 0.5660 * 0.3832 * 1 = 0.035507 \\
 \xi_3(1,2) &= \alpha_2(1)a_{12}b_2(o_3 = \varphi_2)\beta_3(2) = \alpha_2(1)a_{12}b_{22}\beta_3(2) \\
 &= 0.16371 * 0.3134 * 0.3809 * 1 = 0.019543 \\
 \xi_3(1,3) &= \alpha_2(1)a_{13}b_3(o_3 = \varphi_2)\beta_3(3) = \alpha_2(1)a_{13}b_{32}\beta_3(3) \\
 &= 0.16371 * 0.1206 * 0.1950 * 1 = 0.00385 \\
 \xi_3(2,1) &= \alpha_2(2)a_{21}b_1(o_3 = \varphi_2)\beta_3(1) = \alpha_2(2)a_{21}b_{12}\beta_3(1) \\
 &= 0.046277 * 0.4785 * 0.3832 * 1 = 0.008485 \\
 \xi_3(2,2) &= \alpha_2(2)a_{22}b_2(o_3 = \varphi_2)\beta_3(2) = \alpha_2(2)a_{22}b_{22}\beta_3(2) \\
 &= 0.046277 * 0.4262 * 0.3809 * 1 = 0.007513 \\
 \xi_3(2,3) &= \alpha_2(2)a_{23}b_3(o_3 = \varphi_2)\beta_3(3) = \alpha_2(2)a_{23}b_{32}\beta_3(3) \\
 &= 0.046277 * 0.0953 * 0.1950 * 1 = 0.00086 \\
 \xi_3(3,1) &= \alpha_2(3)a_{31}b_1(o_3 = \varphi_2)\beta_3(1) = \alpha_2(3)a_{31}b_{12}\beta_3(1) \\
 &= 0.005234 * 0.6017 * 0.3832 * 1 = 0.001207 \\
 \xi_3(3,2) &= \alpha_2(3)a_{32}b_2(o_3 = \varphi_2)\beta_3(2) = \alpha_2(3)a_{32}b_{22}\beta_3(2) \\
 &= 0.005234 * 0.2822 * 0.3809 * 1 = 0.000563 \\
 \xi_3(3,3) &= \alpha_2(3)a_{33}b_3(o_3 = \varphi_2)\beta_3(3) = \alpha_2(3)a_{33}b_{32}\beta_3(3) \\
 &= 0.005234 * 0.1161 * 0.1950 * 1 = 0.000118
 \end{aligned}$$

$$\begin{aligned}
 \gamma_1(1) &= \alpha_1(1)\beta_1(1) = 0.006699 * 0.145981 = 0.000978 \\
 \gamma_1(2) &= \alpha_1(2)\beta_1(2) = 0.119448 * 0.139559 = 0.01667 \\
 \gamma_1(3) &= \alpha_1(3)\beta_1(3) = 0.400976 * 0.149629 = 0.059998 \\
 \gamma_2(1) &= \alpha_2(1)\beta_2(1) = 0.16371 * 0.359782 = 0.0589 \\
 \gamma_2(2) &= \alpha_2(2)\beta_2(2) = 0.046277 * 0.364284 = 0.016858 \\
 \gamma_2(3) &= \alpha_2(3)\beta_2(3) = 0.005234 * 0.360701 = 0.001888 \\
 \gamma_3(1) &= \alpha_3(1)\beta_3(1) = 0.045199 * 1 = 0.045199 \\
 \gamma_3(2) &= \alpha_3(2)\beta_3(2) = 0.027618 * 1 = 0.027618 \\
 \gamma_3(3) &= \alpha_3(3)\beta_3(3) = 0.004828 * 1 = 0.004828
 \end{aligned}$$

Similarly, within the M-step of the second iteration ($r=2$), the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is re-calculated based on joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ determined at E-step.

$$\begin{aligned}
 \hat{a}_{11} &= \frac{\sum_{t=2}^3 \xi_t(1,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} \\
 &= \frac{\xi_2(1,1) + \xi_3(1,1)}{\xi_2(1,1) + \xi_2(1,2) + \xi_2(1,3) + \xi_3(1,1) + \xi_3(1,2) + \xi_3(1,3)} \\
 &= \frac{(0.000739 + 0.035507)}{(0.000739 + 0.000213 + 0.000026 + 0.035507 + 0.019543 + 0.00385)} \approx 0.6053 \\
 \hat{a}_{12} &= \frac{\sum_{t=2}^3 \xi_t(1,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} \\
 &= \frac{\xi_2(1,2) + \xi_3(1,2)}{\xi_2(1,1) + \xi_2(1,2) + \xi_2(1,3) + \xi_3(1,1) + \xi_3(1,2) + \xi_3(1,3)} \\
 &= \frac{(0.000213 + 0.019543)}{(0.000739 + 0.000213 + 0.000026 + 0.035507 + 0.019543 + 0.00385)} \approx 0.3299 \\
 \hat{a}_{13} &= \frac{\sum_{t=2}^3 \xi_t(1,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} \\
 &= \frac{\xi_2(1,3) + \xi_3(1,3)}{\xi_2(1,1) + \xi_2(1,2) + \xi_2(1,3) + \xi_3(1,1) + \xi_3(1,2) + \xi_3(1,3)} \\
 &= \frac{(0.000026 + 0.00385)}{(0.000739 + 0.000213 + 0.000026 + 0.035507 + 0.019543 + 0.00385)} \approx 0.0648 \\
 \hat{a}_{21} &= \frac{\sum_{t=2}^3 \xi_t(2,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} \\
 &= \frac{\xi_2(2,1) + \xi_3(2,1)}{\xi_2(2,1) + \xi_2(2,2) + \xi_2(2,3) + \xi_3(2,1) + \xi_3(2,2) + \xi_3(2,3)} \\
 &= \frac{(0.011139 + 0.008485)}{(0.011139 + 0.005165 + 0.000366 + 0.008485 + 0.007513 + 0.00086)} \approx 0.5853 \\
 \hat{a}_{22} &= \frac{\sum_{t=2}^3 \xi_t(2,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} \\
 &= \frac{\xi_2(2,2) + \xi_3(2,2)}{\xi_2(2,1) + \xi_2(2,2) + \xi_2(2,3) + \xi_3(2,1) + \xi_3(2,2) + \xi_3(2,3)} \\
 &= \frac{(0.005165 + 0.007513)}{(0.011139 + 0.005165 + 0.000366 + 0.008485 + 0.007513 + 0.00086)} \approx 0.3781 \\
 \hat{a}_{23} &= \frac{\sum_{t=2}^3 \xi_t(2,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} \\
 &= \frac{\xi_2(2,3) + \xi_3(2,3)}{\xi_2(2,1) + \xi_2(2,2) + \xi_2(2,3) + \xi_3(2,1) + \xi_3(2,2) + \xi_3(2,3)} \\
 &= \frac{(0.000366 + 0.00086)}{(0.011139 + 0.005165 + 0.000366 + 0.008485 + 0.007513 + 0.00086)} \approx 0.0366
 \end{aligned}$$

$$\begin{aligned}
 \hat{a}_{31} &= \frac{\sum_{t=2}^3 \xi_t(3,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} \\
 &= \frac{\xi_2(3,1) + \xi_3(3,1)}{\xi_2(3,1) + \xi_2(3,2) + \xi_2(3,3) + \xi_3(3,1) + \xi_3(3,2) + \xi_3(3,3)} \\
 &= \frac{(0.047022 + 0.001207)}{(0.047022 + 0.01148 + 0.001496 + 0.001207 + 0.000563 \\
 &\quad + 0.000118)} \approx 0.7793 \\
 \hat{a}_{32} &= \frac{\sum_{t=2}^3 \xi_t(3,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} \\
 &= \frac{\xi_2(3,2) + \xi_3(3,2)}{\xi_2(3,1) + \xi_2(3,2) + \xi_2(3,3) + \xi_3(3,1) + \xi_3(3,2) + \xi_3(3,3)} \\
 &= \frac{(0.01148 + 0.000563)}{(0.047022 + 0.01148 + 0.001496 + 0.001207 + 0.000563 \\
 &\quad + 0.000118)} \approx 0.1946 \\
 \hat{a}_{33} &= \frac{\sum_{t=2}^3 \xi_t(3,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} \\
 &= \frac{\xi_2(3,3) + \xi_3(3,3)}{\xi_2(3,1) + \xi_2(3,2) + \xi_2(3,3) + \xi_3(3,1) + \xi_3(3,2) + \xi_3(3,3)} \\
 &= \frac{(0.001496 + 0.000118)}{(0.047022 + 0.01148 + 0.001496 + 0.001207 + 0.000563 \\
 &\quad + 0.000118)} \approx 0.0261
 \end{aligned}$$

$$\begin{aligned}
 \hat{b}_1(1) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_1}}^3 \gamma_t(1)}{\sum_{t=1}^3 \gamma_t(1)} = \frac{\gamma_2(1)}{\gamma_1(1) + \gamma_2(1) + \gamma_3(1)} \\
 &= 0.0589/(0.000978 + 0.0589 + 0.045199) \approx 0.5605 \\
 \hat{b}_1(2) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_2}}^3 \gamma_t(1)}{\sum_{t=1}^3 \gamma_t(1)} = \frac{\gamma_3(1)}{\gamma_1(1) + \gamma_2(1) + \gamma_3(1)} \\
 &= 0.045199/(0.000978 + 0.0589 + 0.045199) \approx 0.4302 \\
 \hat{b}_1(3) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_3}}^3 \gamma_t(1)}{\sum_{t=1}^3 \gamma_t(1)} = \frac{0}{\gamma_1(1) + \gamma_2(1) + \gamma_3(1)} = 0 \\
 \hat{b}_1(4) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_4}}^3 \gamma_t(1)}{\sum_{t=1}^3 \gamma_t(1)} = \frac{\gamma_1(1)}{\gamma_1(1) + \gamma_2(1) + \gamma_3(1)} \\
 &= 0.000978/(0.000978 + 0.0589 + 0.045199) \approx 0.0093 \\
 \hat{b}_2(1) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_1}}^3 \gamma_t(2)}{\sum_{t=1}^3 \gamma_t(2)} = \frac{\gamma_2(2)}{\gamma_1(2) + \gamma_2(2) + \gamma_3(2)} \\
 &= 0.016858/(0.01667 + 0.016858 + 0.027618) \approx 0.2757 \\
 \hat{b}_2(2) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_2}}^3 \gamma_t(2)}{\sum_{t=1}^3 \gamma_t(2)} = \frac{\gamma_3(2)}{\gamma_1(2) + \gamma_2(2) + \gamma_3(2)} \\
 &= 0.027618/(0.01667 + 0.016858 + 0.027618) \approx 0.4517 \\
 \hat{b}_2(3) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_3}}^3 \gamma_t(2)}{\sum_{t=1}^3 \gamma_t(2)} = \frac{0}{\gamma_1(2) + \gamma_2(2) + \gamma_3(2)} = 0
 \end{aligned}$$

$$\begin{aligned}
 \hat{b}_2(4) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_4}}^3 \gamma_t(2)}{\sum_{t=1}^3 \gamma_t(2)} = \frac{\gamma_1(2)}{\gamma_1(2) + \gamma_2(2) + \gamma_3(2)} \\
 &= 0.01667 / (0.01667 + 0.016858 + 0.027618) \approx 0.2726 \\
 \hat{b}_3(1) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_1}}^3 \gamma_t(3)}{\sum_{t=1}^3 \gamma_t(3)} = \frac{\gamma_2(3)}{\gamma_1(3) + \gamma_2(3) + \gamma_3(3)} \\
 &= 0.001888 / (0.059998 + 0.001888 + 0.004828) \approx 0.0283 \\
 \hat{b}_3(2) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_2}}^3 \gamma_t(3)}{\sum_{t=1}^3 \gamma_t(3)} = \frac{\gamma_3(3)}{\gamma_1(3) + \gamma_2(3) + \gamma_3(3)} \\
 &= 0.004828 / (0.059998 + 0.001888 + 0.004828) \approx 0.0724 \\
 \hat{b}_3(3) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_3}}^3 \gamma_t(3)}{\sum_{t=1}^3 \gamma_t(3)} = \frac{0}{\gamma_1(3) + \gamma_2(3) + \gamma_3(3)} = 0 \\
 \hat{b}_3(4) &= \frac{\sum_{\substack{t=1 \\ o_t=\varphi_4}}^3 \gamma_t(3)}{\sum_{t=1}^3 \gamma_t(3)} = \frac{\gamma_1(3)}{\gamma_1(3) + \gamma_2(3) + \gamma_3(3)} \\
 &= 0.059998 / (0.059998 + 0.001888 + 0.004828) \approx 0.8993
 \end{aligned}$$

$$\begin{aligned}
 \hat{\pi}_1 &= \frac{\gamma_1(1)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.000978 / (0.000978 + 0.01667 + 0.059998) \\
 &\approx 0.0126 \\
 \hat{\pi}_2 &= \frac{\gamma_1(2)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.01667 / (0.000978 + 0.01667 + 0.059998) \\
 &\approx 0.2147 \\
 \hat{\pi}_3 &= \frac{\gamma_1(3)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.059998 / (0.000978 + 0.01667 + 0.059998) \\
 &\approx 0.7727
 \end{aligned}$$

Table IV.4.3.2.4 summarizes HMM parameters resulted from the first iteration and the second iteration of EM algorithm.

Iteration	HMM parameters			
1 st	$\hat{a}_{11} = 0.5660$	$\hat{a}_{12} = 0.3134$	$\hat{a}_{13} = 0.1206$	
	$\hat{a}_{21} = 0.4785$	$\hat{a}_{22} = 0.4262$	$\hat{a}_{23} = 0.0953$	
	$\hat{a}_{31} = 0.6017$	$\hat{a}_{32} = 0.2822$	$\hat{a}_{33} = 0.1161$	
	$\hat{b}_1(1) = 0.5417$	$\hat{b}_1(2) = 0.3832$	$\hat{b}_1(3) = 0$	$\hat{b}_1(4) = 0.0751$
	$\hat{b}_2(1) = 0.2785$	$\hat{b}_2(2) = 0.3809$	$\hat{b}_2(3) = 0$	$\hat{b}_2(4) = 0.3406$
	$\hat{b}_3(1) = 0.0891$	$\hat{b}_3(2) = 0.1950$	$\hat{b}_3(3) = 0$	$\hat{b}_3(4) = 0.7159$
	$\hat{\pi}_1 = 0.0892$	$\hat{\pi}_2 = 0.3507$	$\hat{\pi}_3 = 0.5601$	
	Terminating criterion $P(O/\Delta) = 0.013$			
2 nd	$\hat{a}_{11} = 0.6053$	$\hat{a}_{12} = 0.3299$	$\hat{a}_{13} = 0.0648$	
	$\hat{a}_{21} = 0.5853$	$\hat{a}_{22} = 0.3781$	$\hat{a}_{23} = 0.0366$	
	$\hat{a}_{31} = 0.7793$	$\hat{a}_{32} = 0.1946$	$\hat{a}_{33} = 0.0261$	

	$\hat{b}_1(1) = 0.5605$	$\hat{b}_1(2) = 0.4302$	$\hat{b}_1(3) = 0$	$\hat{b}_1(4) = 0.0093$
	$\hat{b}_2(1) = 0.2757$	$\hat{b}_2(2) = 0.4517$	$\hat{b}_2(3) = 0$	$\hat{b}_2(4) = 0.2726$
	$\hat{b}_3(1) = 0.0283$	$\hat{b}_3(2) = 0.0724$	$\hat{b}_3(3) = 0$	$\hat{b}_3(4) = 0.8993$
<hr/>				
	$\hat{\pi}_1 = 0.0126$	$\hat{\pi}_2 = 0.2147$	$\hat{\pi}_3 = 0.7727$	
<hr/>				
	Terminating criterion $P(O \Delta) = 0.0776$			

Table IV.4.3.2.4. HMM parameters resulted from the first iteration and the second iteration of EM algorithm

As seen in table IV.4.3.2.4, the EM algorithm does not converge yet when it produces two different terminating criteria (0.013 and 0.0776) at the first iteration and the second iteration. It is necessary to run more iterations so as to gain the most optimal estimate. Within this example, the EM algorithm converges absolutely after 10 iterations when the criterion $P(O|\Delta)$ approaches to the same value 1 at the 9th and 10th iterations. Table IV.4.3.2.5 shows HMM parameter estimates along with terminating criterion $P(O|\Delta)$ at the 1st, 2nd, 9th, and 10th iterations of EM algorithm.

Iteration	HMM parameters		
1 st	$\hat{a}_{11} = 0.5660$	$\hat{a}_{12} = 0.3134$	$\hat{a}_{13} = 0.1206$
	$\hat{a}_{21} = 0.4785$	$\hat{a}_{22} = 0.4262$	$\hat{a}_{23} = 0.0953$
	$\hat{a}_{31} = 0.6017$	$\hat{a}_{32} = 0.2822$	$\hat{a}_{33} = 0.1161$
	$\hat{b}_1(1) = 0.5417$	$\hat{b}_1(2) = 0.3832$	$\hat{b}_1(3) = 0$
	$\hat{b}_2(1) = 0.2785$	$\hat{b}_2(2) = 0.3809$	$\hat{b}_2(3) = 0$
	$\hat{b}_3(1) = 0.0891$	$\hat{b}_3(2) = 0.1950$	$\hat{b}_3(3) = 0$
	$\hat{b}_3(4) = 0.7159$		
	$\hat{\pi}_1 = 0.0892$	$\hat{\pi}_2 = 0.3507$	$\hat{\pi}_3 = 0.5601$
	Terminating criterion $P(O \Delta) = 0.013$		
2 nd	$\hat{a}_{11} = 0.6053$	$\hat{a}_{12} = 0.3299$	$\hat{a}_{13} = 0.0648$
	$\hat{a}_{21} = 0.5853$	$\hat{a}_{22} = 0.3781$	$\hat{a}_{23} = 0.0366$
	$\hat{a}_{31} = 0.7793$	$\hat{a}_{32} = 0.1946$	$\hat{a}_{33} = 0.0261$
	$\hat{b}_1(1) = 0.5605$	$\hat{b}_1(2) = 0.4302$	$\hat{b}_1(3) = 0$
	$\hat{b}_2(1) = 0.2757$	$\hat{b}_2(2) = 0.4517$	$\hat{b}_2(3) = 0$
	$\hat{b}_3(1) = 0.0283$	$\hat{b}_3(2) = 0.0724$	$\hat{b}_3(3) = 0$
	$\hat{b}_3(4) = 0.8993$		
	$\hat{\pi}_1 = 0.0126$	$\hat{\pi}_2 = 0.2147$	$\hat{\pi}_3 = 0.7727$
	Terminating criterion $P(O \Delta) = 0.0776$		
9 th	$\hat{a}_{11} = 0$	$\hat{a}_{12} = 1$	$\hat{a}_{13} = 0$
	$\hat{a}_{21} = 0$	$\hat{a}_{22} = 1$	$\hat{a}_{23} = 0$
	$\hat{a}_{31} = 1$	$\hat{a}_{32} = 0$	$\hat{a}_{33} = 0$

	$\hat{b}_1(1) = 1$	$\hat{b}_1(2) = 0$	$\hat{b}_1(3) = 0$	$\hat{b}_1(4) = 0$
	$\hat{b}_2(1) = 0$	$\hat{b}_2(2) = 1$	$\hat{b}_2(3) = 0$	$\hat{b}_2(4) = 0$
	$\hat{b}_3(1) = 0$	$\hat{b}_3(2) = 0$	$\hat{b}_3(3) = 0$	$\hat{b}_3(4) = 1$
	$\hat{\pi}_1 = 0$	$\hat{\pi}_2 = 0$	$\hat{\pi}_3 = 1$	
	Terminating criterion $P(O/\Delta) = 1$			
10 th	$\hat{a}_{11} = 0$	$\hat{a}_{12} = 1$	$\hat{a}_{13} = 0$	
	$\hat{a}_{21} = 0$	$\hat{a}_{22} = 1$	$\hat{a}_{23} = 0$	
	$\hat{a}_{31} = 1$	$\hat{a}_{32} = 0$	$\hat{a}_{33} = 0$	
	$\hat{b}_1(1) = 1$	$\hat{b}_1(2) = 0$	$\hat{b}_1(3) = 0$	$\hat{b}_1(4) = 0$
	$\hat{b}_2(1) = 0$	$\hat{b}_2(2) = 1$	$\hat{b}_2(3) = 0$	$\hat{b}_2(4) = 0$
	$\hat{b}_3(1) = 0$	$\hat{b}_3(2) = 0$	$\hat{b}_3(3) = 0$	$\hat{b}_3(4) = 1$
	$\hat{\pi}_1 = 0$	$\hat{\pi}_2 = 0$	$\hat{\pi}_3 = 1$	
	Terminating criterion $P(O/\Delta) = 1$			

Table IV.4.3.2.5. HMM parameters along with terminating criteria after 10 iterations of EM algorithm

As a result, the learned parameters A , B , and Π are shown in table IV.4.3.2.6:

		Weather current day (Time point t)		
		sunny	cloudy	rainy
Weather previous day (Time point $t-1$)	sunny	$a_{11}=0$	$a_{12}=1$	$a_{13}=0$
	cloudy	$a_{21}=0$	$a_{22}=1$	$a_{23}=0$
	rainy	$a_{31}=1$	$a_{32}=0$	$a_{33}=0$
		sunny	cloudy	rainy
		$\pi_1=0$	$\pi_2=0$	$\pi_3=1$
Weather		Humidity		
		dry	dryish	damp
sunny	$b_{11}=1$	$b_{12}=0$	$b_{13}=0$	
cloudy	$b_{21}=0$	$b_{22}=1$	$b_{23}=0$	
rainy	$b_{31}=0$	$b_{32}=0$	$b_{33}=0$	
		soggy		
		$b_{14}=0$		
		$b_{24}=0$		
		$b_{34}=1$		

Table IV.4.3.2.6. HMM parameters of weather example learned from EM algorithm

Such learned parameters are more appropriate to the training observation sequence $O = \{o_1=\varphi_4=soggy, o_2=\varphi_1=dry, o_3=\varphi_2=dryish\}$ than the original ones shown in tables IV.4.1, IV.4.2, and IV.4.3 when the terminating criterion $P(O/\Delta)$ corresponding to its optimal state sequence is 1.

Now three main problems of HMM are described; please see an excellent document “A tutorial on hidden Markov models and selected applications in speech recognition” written by author Rabiner (Rabiner, 1989) for advanced details about

HMM. The next section [IV.5](#) described a HMM whose observations are continuous. Because section [IV.5](#) is extensive sub-section, you can ignore it in order to read section [IV.6](#) that describes the new approach which uses HMM to discover and represent users' learning styles (Nguyen L. , A New Approach for Modeling and Discovering Learning Styles by Using Hidden Markov Model, 2013). Although the purpose of description of HMM is to introduce how to apply HMM into representing learning style model, the next section [IV.5](#) can be read as separated report for continuous observation HMM (Nguyen L. , Continuous Observation Hidden Markov Model, 2016).

IV.5. Continuous observation hidden Markov model

Observations of normal HMM mentioned in previous sub-section [IV.4](#) are quantified by discrete probability distribution that is concretely observation probability matrix B . In the general situation, observation o_t is continuous variable and matrix B is replaced by probability density function (PDF); please see formula [III.1.1.6](#) for more details about PDF. Formula [IV.5.1](#) specifies the PDF of continuous observation o_t given state s_j .

$$b_j(o_t) = p_j(o_t | \theta_j)$$

Formula IV.5.1. Probability density function (PDF) of observation

Where the PDF $p_j(o_t | \theta_j)$ belongs to any probability distribution, for example, normal distribution, exponential distribution, etc. The notation θ_j denotes probabilistic parameters, for instance, if $p_j(o_t | \theta_j)$ is normal distribution PDF, θ_j includes mean m_j and variance σ_j^2 . The HMM now is specified by parameter $\Delta = (a_{ij}, \theta_j, \pi_j)$, which is called *continuous observation HMM* (Rabiner, 1989, p. 267). The PDF $p_j(o_t | \theta_j)$ is known as *single PDF* because it is atom PDF which is not combined with any other PDF. We will research so-called mixture model PDF that is constituted of many partial PDF (s) later. We still apply EM algorithm known as Baum-Welch algorithm into learning continuous observation HMM. In the field of continuous-speech recognition, authors Lee, Rabiner, Pieraccini, and Wilpon (Lee, Rabiner, Pieraccini, & Wilpon, 1990) proposed Bayesian adaptive learning for estimating mean and variance of continuous density HMM. Authors Huo and Lee (Huo & Lee, 1997) proposed a framework of quasi-Bayes (QB) algorithm based on approximate recursive Bayes estimate for learning HMM parameters with Gaussian mixture model; they described that "The QB algorithm is designed to incrementally update the hyper-parameters of the approximate posterior distribution and the continuous density HMM parameters simultaneously" (Huo & Lee, 1997, p. 161). Authors Cheng, Sha, and Saul (Sha & Saul, 2009) (Cheng, Sha, & Saul, 2009) used the approach of large margin training to learn HMM parameters. Such approach is different from Baum-Welch algorithm when it firstly establishes discriminant functions for correct and incorrect label sequences and then, finds parameters satisfying the margin constraint that separates the discriminant functions as much as possible (Sha & Saul, 2009, pp. 106-108). Authors Cheng, Sha, and Saul (Cheng, Sha, & Saul, 2009, p. 4) proposed a fast online algorithm for large margin training, in which "the parameters for discriminant functions are updated according to an online learning rule with given learning rate". Large margin training is very appropriate to speech recognition, which was proposed by authors Sha and Saul (Sha & Saul, 2006) in the article "Large Margin Hidden

Markov Models for Automatic Speech Recognition". Some other authors used different learning approaches such as conditional maximum likelihood and minimizing classification error, mentioned in (Sha & Saul, 2009, pp. 104-105).

Methods to solve evaluation problem and uncovering problem mentioned previous sub-sections IV.4.1, IV.4.2, and IV.4.3 are kept intact by using the observation PDF specified by formula IV.5.1. For example, forward-backward procedure (based on forward variable, shown in table IV.4.1.1) that solves evaluation problem is based on the recurrence formula IV.4.1.2 as follows:

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^n \alpha_t(i) a_{ij} \right) b_j(o_{t+1})$$

In order to apply forward-backward procedure into continuous observation HMM, it is simple to replace the discrete probability $b_j(o_{t+1})$ by the single PDF specified by formula IV.5.1.

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^n \alpha_t(i) a_{ij} \right) p_j(o_{t+1} | \theta_j)$$

However, there is a change in solution of learning problem. Recall that the essence of EM algorithm applied into HMM learning problem is to determine the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{\theta}_j, \hat{\pi}_j)$. Formulas for calculating estimates \hat{a}_{ij} and $\hat{\pi}_j$ are kept intact, as aforementioned in formula IV.4.3.2.7.

$$\begin{aligned}\hat{a}_{ij} &= \frac{\sum_{t=2}^T \xi_t(i, j)}{\sum_{t=2}^T \sum_{l=1}^n \xi_t(i, l)} \\ \hat{\pi}_j &= \frac{\gamma_1(j)}{\sum_{i=1}^n \gamma_1(i)}\end{aligned}$$

Where joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ are modified based on replacing discrete probability $b_j(o_t)$ by the single PDF $p_j(o_t | \theta_j)$ given current parameter $\Delta = (a_{ij}, b_j(k), \pi_j)$.

$$\begin{aligned}\xi_t(i, j) &= \alpha_{t-1}(i) a_{ij} p_j(o_t | \theta_j) \beta_t(j) \text{ where } t \geq 2 \\ \gamma_t(j) &= P(O, x_t = s_j | \Delta) = \alpha_t(j) \beta_t(j)\end{aligned}$$

Where forward variable α_t and backward variable β_t are calculated by recurrence formulas IV.4.1.2 and IV.4.1.5. Recall that $\gamma_t(j)$ is joint probability that the stochastic process is in state s_j at time point t with observation sequence O and $\xi_t(i, j)$ is the joint probability that the stochastic process receives state s_i at time point $t-1$ and state s_j at time point t given observation sequence O .

Your attention please, quantities $\xi_t(i, j)$, $\gamma_t(j)$, $\alpha_t(i)$, and $\beta_t(j)$ are essentially continuous functions because they are based on PDF $p_j(o_t | \theta_j)$. Their values on a concrete observation o_t are zero because the value of PDF $p_j(o_t | \theta_j)$ given such concrete observation o_t is zero. Therefore, in practice, these quantities are calculated according to integral of PDF $p_j(o_t | \theta_j)$ in ε -vicinity of o_t where ε is very small positive number. The number ε can reflect inherent attribute of observation data with regard to measure bias, for example, if atmosphere humidity at time point t is $o_t = 0.5 \mp 0.01$, the measure bias is 0.01 and so we have $\varepsilon=0.01$. In addition, the number ε can be pre-defined fixedly by arbitrary very small number. For example, given $\varepsilon=0.01$ we have:

$$\int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o | \theta_j) do = \int_{0.5-0.01}^{0.5+0.01} p_j(o | \theta_j) do$$

If all o_t are intervals, for example, $0.1 \leq o_t \leq 0.2, 0.3 \leq o_{t+1} \leq 0.4, \dots$ then, the integral of PDF $p_j(o_t | \theta_j)$ is calculated directly over such o_t .

$$\int_{o_t}^{0.2} p_j(o|\theta_j) do = \int_{0.1}^{0.2} p_j(o|\theta_j) do$$

Given the PDF $p_j(o_t|\theta_j)$ conforms normal distribution, it is easy to calculate the probability of o_t in ε -vicinity as the integral of PDF $p_j(o_t|\theta_j)$ in ε -vicinity of o_t as follows:

$$\int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o|\theta_j) do = ? \text{ if } p_j(o|\theta_j) \text{ is normal PDF}$$

The best way is to standardize the normal PDF $p_j(o_t|\theta_j)$ where $\theta_j = (m_j, \sigma_j^2)$ into cumulative standard normal distribution (Montgomery & Runger, 2003, p. 653). Let Φ be cumulative standard normal distribution (Montgomery & Runger, 2003, p. 653); please see formula III.1.1.6 for more details about cumulative distribution function. We have:

$$\begin{aligned} \int_{-\infty}^b p_j(o|\theta_j) do &= \Phi\left(\frac{b - m_j}{\sqrt{\sigma_j^2}}\right) \\ \int_a^b p_j(o|\theta_j) do &= \Phi\left(\frac{b - m_j}{\sqrt{\sigma_j^2}}\right) - \Phi\left(\frac{a - m_j}{\sqrt{\sigma_j^2}}\right) \end{aligned}$$

The quantities $\frac{b-m_j}{\sqrt{\sigma_j^2}}$ and $\frac{a-m_j}{\sqrt{\sigma_j^2}}$ are standardized values of b and a given PDF $p_j(o_t|\theta_j)$,

respectively. The function Φ is always evaluated in popular. For instance, appendix A of the book “Applied Statistics and Probability for Engineers” by authors Montgomery and Runger (Montgomery & Runger, 2003, p. 653) is a good reference for looking up some values of Φ . Please distinguish the function Φ from the set of possible discrete observations $\Phi = \{\varphi_1, \varphi_2, \dots, \varphi_m\}$ aforementioned at the beginning of section IV.4 when they share the same notation.

$$\begin{aligned} \int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o|\theta_j) do \\ &= \Phi\left(\frac{o_t + \varepsilon - m_j}{\sqrt{\sigma_j^2}}\right) - \Phi\left(\frac{o_t - \varepsilon - m_j}{\sqrt{\sigma_j^2}}\right) \text{ if } p_j(o|\theta_j) \text{ is normal PDF} \end{aligned}$$

Formula IV.5.2 specifies quantities $\xi_t(i, j)$, $\gamma_t(j)$ according to integral of PDF $p_j(o_t|\theta_j)$.

$$\begin{aligned} \xi_t(i, j) &= P(O, x_{t-1} = s_i, x_t = s_j | \Delta) = \alpha_{t-1}(i) a_{ij} \left(\int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o|\theta_j) do \right) \beta_t(j) \text{ where } t \\ &\geq 2 \\ \gamma_t(j) &= P(O, x_t = s_j | \Delta) = \alpha_t(j) \beta_t(j) \end{aligned}$$

Where forward variable α_t and backward variable β_t are calculated based on recurrence formulas IV.4.1.2 and IV.4.1.5.

$$\begin{aligned}
 \alpha_{t+1}(j) &= \left(\sum_{i=1}^n \alpha_t(i) a_{ij} \right) \int_{o_{t+1}-\varepsilon}^{o_{t+1}+\varepsilon} p_j(o|\theta_j) do \\
 \beta_t(i) &= \sum_{j=1}^n a_{ij} \left(\int_{o_{t+1}-\varepsilon}^{o_{t+1}+\varepsilon} p_j(o|\theta_j) do \right) \beta_{t+1}(j) \\
 &\quad \int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o|\theta_j) do \\
 &= \Phi \left(\frac{o_t + \varepsilon - m_j}{\sqrt{\sigma_j^2}} \right) - \Phi \left(\frac{o_t - \varepsilon - m_j}{\sqrt{\sigma_j^2}} \right) \text{ if } p_j(o|\theta_j) \text{ is normal PDF}
 \end{aligned}$$

Formula IV.5.2. Joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ based on single PDF

Note that m_j and σ_j^2 are mean and variance of the normal PDF $p_j(o_t|\theta_j)$ and Φ is cumulative standard normal distribution (Montgomery & Rungger, 2003, p. 653).

As a convention, quantities $\xi_t(i, j)$, $\gamma_t(j)$, $\alpha_{t+1}(j)$, and $\beta_t(i)$ are still referred as joint probabilities, forward variable, and backward variable. This convention help us to describe traditional HMM and continuously conventional HMM in coherent way.

Now it is necessary to determine the estimate $\hat{\theta}_j$. Derived from formula IV.4.3.2.2, the expectation $E_{X|O, \Delta_r}\{ln(P(O, X|\Delta))\}$ is modified based on replacing discrete probability $b_j(o_t)$ by the continuous PDF $p_j(o_t|\theta_j)$, as seen in following formula IV.5.3 given current parameter Δ_r .

$$\begin{aligned}
 E_{X|O, \Delta_r}\{ln(P(O, X|\Delta))\} &= \sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^n \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) ln(a_{ij}) \right. \\
 &\quad \left. + \sum_{j=1}^n \sum_{t=1}^T I(x_t = s_j) ln(p_j(o_t|\theta_j)) \right)
 \end{aligned}$$

Formula IV.5.3. EM conditional expectation for continuous observation HMM with single PDF

Where $I(x_{t-1} = s_i, x_t = s_j)$ and $I(x_t = s_j)$ are index functions so that

$$\begin{aligned}
 I(x_{t-1} = s_i, x_t = s_j) &= \begin{cases} 1 & \text{if } x_{t-1} = s_i \text{ and } x_t = s_j \\ 0 & \text{otherwise} \end{cases} \\
 I(x_t = s_j) &= \begin{cases} 1 & \text{if } x_t = s_j \\ 0 & \text{otherwise} \end{cases}
 \end{aligned}$$

Derived from formula IV.4.3.2.3, the HMM Lagrangian function for continuous observation HMM with single PDF is specified by formula IV.5.4.

$$l(\Delta, \lambda) = l(a_{ij}, \theta_j, \lambda_i) = E_{X|O, \Delta_r}\{ln(P(O, X|\Delta))\} + \sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right)$$

Formula IV.5.4. Lagrangian function for continuous observation HMM with single PDF

Where λ is n -component vector $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$. Factors $\lambda_i \geq 0$ are called Lagrange multipliers or Karush-Kuhn-Tucker multipliers (Wikipedia, Karush–Kuhn–Tucker conditions, 2014) or dual variables.

The parameter estimate $\hat{\theta}_j$ which is extreme point of the Lagrangian function $l(\Delta, \lambda)$ is determined by setting partial derivatives of $l(\Delta, \lambda)$ with respect to θ_j to be zero. The partial derivative of $l(\Delta, \lambda)$ with respect to θ_j is:

$$\begin{aligned}
 \frac{\partial l(\Delta, \lambda)}{\partial \theta_j} &= \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial \theta_j} + \frac{\partial}{\partial \theta_j} \left(\sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \right) \\
 &= \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial \theta_j} \\
 &= \frac{\partial}{\partial \theta_j} \left(\sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^n \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(a_{ij}) \right. \right. \\
 &\quad \left. \left. + \sum_{j=1}^n \sum_{t=1}^T I(x_t = s_j) \ln(p_j(o_t|\theta_j)) \right) \right) \\
 &= \sum_X P(X|O, \Delta_r) \sum_{j=1}^n \sum_{t=1}^T I(x_t = s_j) \frac{\partial}{\partial \theta_j} \left(\ln(p_j(o_t|\theta_j)) \right) \\
 &= \sum_X P(X|O, \Delta_r) \sum_{t=1}^T I(x_t = s_j) \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j} \\
 &= \sum_{t=1}^T \sum_X I(x_t = s_j) P(X|O, \Delta_r) \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j} \\
 &= \sum_{t=1}^T \sum_X I(x_t = s_j) P(x_1, \dots, x_t, \dots, x_T|O, \Delta_r) \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j} \\
 &= \sum_{t=1}^T P(x_t = s_j|O, \Delta_r) \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j} \\
 &\quad \text{(Due to total probability rule specified by formula III.1.1.4)} \\
 &= \sum_{t=1}^T \frac{P(O, x_t = s_j|\Delta_r)}{P(O|\Delta_r)} \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j} \\
 &\quad \text{(Due to multiplication rule specified by formula III.1.1.3)} \\
 &= \frac{1}{P(O|\Delta_r)} \sum_{t=1}^T P(O, x_t = s_j|\Delta_r) \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j} \\
 &= \frac{1}{P(O|\Delta_r)} \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j}
 \end{aligned}$$

Setting the partial derivative $\frac{\partial l(\Delta, \lambda)}{\partial \theta_j}$ to be zero, we get the equation whose solution is estimate $\hat{\theta}_j$, specified by formula IV.5.5.

$$\frac{1}{P(O|\Delta_r)} \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j} = 0 \Leftrightarrow \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln(p_j(o_t|\theta_j))}{\partial \theta_j} = 0$$

Formula IV.5.5. Equation of single PDF parameter

It is possible to solve the above equation (formula IV.5.5) by Newton-Raphson method (Burden & Faires, 2011, pp. 67-69) – a numeric analysis method but it is easier and simpler to find out more precise solution if the PDF $p_j(o_t|\theta_j)$ belongs to well-known distributions: normal distribution (Montgomery & Runger, 2003, pp. 109-110), exponential distribution (Montgomery & Runger, 2003, pp. 122-123), etc. According to author Couvreur (Couvreur, 1996, p. 32), the estimate $\hat{\theta}_j$ is determined by the more general formula as follows:

$$\hat{\theta}_j = \operatorname{argmax}_{\theta_j} \sum_{t=1}^T \gamma_t(j) \ln(p_j(o_t|\theta_j))$$

The easy way to find out $\hat{\theta}_j$ is to solve formula II.5 by taking advantages of derivatives.

Suppose $p_j(o_t|\theta_j)$ is normal PDF whose parameter is $\theta_j = (m_j, \sigma_j^2)$ where m_j and σ_j^2 are mean and variance, respectively.

$$p_j(o_t|\theta_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2} \frac{(o_t - m_j)^2}{\sigma_j^2}\right)$$

The equation specified by formula IV.5.5 is re-written with regard to parameter m_j as follows:

$$\begin{aligned} & \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln\left(\frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2} \frac{(o_t - m_j)^2}{\sigma_j^2}\right)\right)}{\partial m_j} = 0 \\ & \Rightarrow \sum_{t=1}^T \gamma_t(j) \frac{\partial \left(-\frac{1}{2} \left(\ln(2\pi) + \ln(\sigma_j^2) + \frac{(o_t - m_j)^2}{\sigma_j^2}\right)\right)}{\partial m_j} = 0 \\ & \Rightarrow -\frac{1}{2} \sum_{t=1}^T \gamma_t(j) \frac{(o_t - m_j)}{\sigma_j^2} = 0 \Rightarrow -\frac{1}{2\sigma_j^2} \sum_{t=1}^T \gamma_t(j)(o_t - m_j) = 0 \\ & \Rightarrow \sum_{t=1}^T \gamma_t(j)(o_t - m_j) = 0 \Rightarrow \sum_{t=1}^T \gamma_t(j)o_t - m_j \sum_{t=1}^T \gamma_t(j) = 0 \\ & \Rightarrow m_j = \frac{\sum_{t=1}^T \gamma_t(j)o_t}{\sum_{t=1}^T \gamma_t(j)} \end{aligned}$$

Therefore, the estimate \hat{m}_j is

$$\hat{m}_j = \frac{\sum_{t=1}^T \gamma_t(j) o_t}{\sum_{t=1}^T \gamma_t(j)}$$

The equation specified by formula IV.5.5 is re-written with regard to parameter σ_j^2 as follows:

$$\begin{aligned} & \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln \left(\frac{1}{\sqrt{2\pi\sigma_j^2}} \exp \left(-\frac{1}{2} \frac{(o_t - m_j)^2}{\sigma_j^2} \right) \right)}{\partial \sigma_j^2} = 0 \\ & \Rightarrow \sum_{t=1}^T \gamma_t(j) \frac{\partial \left(-\frac{1}{2} \left(\ln(2\pi) + \ln(\sigma_j^2) + \frac{(o_t - m_j)^2}{\sigma_j^2} \right) \right)}{\partial \sigma_j^2} = 0 \\ & \Rightarrow -\frac{1}{2} \sum_{t=1}^T \gamma_t(j) \left(\frac{1}{\sigma_j^2} - \frac{(o_t - m_j)^2}{(\sigma_j^2)^2} \right) = 0 \\ & \Rightarrow -\frac{1}{2\sigma_j^2} \sum_{t=1}^T \gamma_t(j) \left(1 - \frac{(o_t - m_j)^2}{\sigma_j^2} \right) = 0 \\ & \Rightarrow \sum_{t=1}^T \gamma_t(j) \left(1 - \frac{(o_t - m_j)^2}{\sigma_j^2} \right) = 0 \\ & \Rightarrow \sum_{t=1}^T \gamma_t(j) - \frac{1}{\sigma_j^2} \sum_{t=1}^T \gamma_t(j) (o_t - m_j)^2 = 0 \\ & \Rightarrow \sigma_j^2 = \frac{\sum_{t=1}^T \gamma_t(j) (o_t - m_j)^2}{\sum_{t=1}^T \gamma_t(j)} \end{aligned}$$

It implies that given the estimate \hat{m}_j , the estimate $\hat{\sigma}_j^2$ is:

$$\hat{\sigma}_j^2 = \frac{\sum_{t=1}^T \gamma_t(j) (o_t - \hat{m}_j)^2}{\sum_{t=1}^T \gamma_t(j)}$$

In general, the normal parameter estimate $\hat{\theta}_j$ is:

$$\hat{\theta}_j = \left(\hat{m}_j = \frac{\sum_{t=1}^T \gamma_t(j) o_t}{\sum_{t=1}^T \gamma_t(j)}, \hat{\sigma}_j^2 = \frac{\sum_{t=1}^T \gamma_t(j) (o_t - \hat{m}_j)^2}{\sum_{t=1}^T \gamma_t(j)} \right)$$

Suppose $p_j(o_t|\theta_j)$ is exponential PDF whose parameter is $\theta_j = \kappa_j$ as follows:

$$p_j(o_t|\theta_j) = \kappa_j e^{-\kappa_j o_t}$$

The equation specified by formula IV.5.5 is re-written with regard to parameter κ_j as follows:

$$\begin{aligned} & \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln(\kappa_j e^{-\kappa_j o_t})}{\partial \kappa_j} = 0 \\ & \Rightarrow \sum_{t=1}^T \gamma_t(j) \frac{\partial (\ln(\kappa_j) - \kappa_j o_t)}{\partial \kappa_j} = 0 \end{aligned}$$

$$\begin{aligned} &\Rightarrow \sum_{t=1}^T \gamma_t(j) \left(\frac{1}{\kappa_j} - o_t \right) = 0 \\ &\Rightarrow \frac{1}{\kappa_j} \sum_{t=1}^T \gamma_t(j) - \sum_{t=1}^T \gamma_t(j) o_t = 0 \\ &\Rightarrow \kappa_j = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j) o_t} \end{aligned}$$

Therefore, the exponential parameter estimate $\hat{\theta}_j$ is

$$\hat{\theta}_j = \hat{\kappa}_j = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j) o_t}$$

Shortly, the continuous observation HMM parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{\theta}_j, \hat{\pi}_i)$ with single PDF given current parameter $\Delta = (a_{ij}, b_i(k), \pi_i)$ is specified by formula IV.5.6.

$$\begin{aligned} \hat{a}_{ij} &= \frac{\sum_{t=2}^T \xi_t(i, j)}{\sum_{t=2}^T \sum_{l=1}^n \xi_t(i, l)} \\ \hat{\pi}_j &= \frac{\gamma_1(j)}{\sum_{i=1}^n \gamma_1(i)} \\ \hat{\theta}_j \text{ is the solution of } &\sum_{t=1}^T \gamma_t(j) \frac{\partial \ln(p_j(o_t | \theta_j))}{\partial \theta_j} = 0 \end{aligned}$$

With normal distribution:

$$\hat{\theta}_j = \left(\hat{m}_j = \frac{\sum_{t=1}^T \gamma_t(j) o_t}{\sum_{t=1}^T \gamma_t(j)}, \hat{\sigma}_j^2 = \frac{\sum_{t=1}^T \gamma_t(j) (o_t - \hat{m}_j)^2}{\sum_{t=1}^T \gamma_t(j)} \right)$$

With exponential distribution:

$$\hat{\theta}_j = \hat{\kappa}_j = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j) o_t}$$

Formula IV.5.6. Continuous observation HMM parameter estimate with single PDF

Where joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ based on single PDF $p_j(o_t | \theta_j)$ is specified by formula IV.5.2.

The EM algorithm applied into learning continuous observation HMM parameter with single PDF is described in table IV.5.1.

Starting with initial value for Δ , each iteration in EM algorithm has two steps:

1. *E-step*: Calculating the joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ according to formula IV.5.2.

$$\begin{aligned} \xi_t(i, j) &= \alpha_{t-1}(i) a_{ij} \left(\int_{o_{t-\varepsilon}}^{o_{t+\varepsilon}} p_j(o | \theta_j) do \right) \beta_t(j) \text{ where } t \geq 2 \\ \gamma_t(j) &= \alpha_t(j) \beta_t(j) \end{aligned}$$

2. *M-step*: Calculating the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{\theta}_j, \hat{\pi}_j)$ based on the joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ determined at E-step, according to formula IV.5.6. The estimate $\hat{\Delta}$ becomes the current parameter for next iteration.

$$\hat{a}_{ij} = \frac{\sum_{t=2}^T \xi_t(i, j)}{\sum_{t=2}^T \sum_{l=1}^n \xi_t(i, l)}$$

$$\hat{\pi}_j = \frac{\gamma_1(j)}{\sum_{i=1}^n \gamma_1(i)}$$

$$\hat{\theta}_j \text{ is the solution of } \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln(p_j(o_t | \theta_j))}{\partial \theta_j} = 0$$

With normal distribution:

$$\hat{\theta}_j = \left(\hat{m}_j = \frac{\sum_{t=1}^T \gamma_t(j) o_t}{\sum_{t=1}^T \gamma_t(j)}, \hat{\sigma}_j^2 = \frac{\sum_{t=1}^T \gamma_t(j) (o_t - \hat{m}_j)^2}{\sum_{t=1}^T \gamma_t(j)} \right)$$

With exponential distribution:

$$\hat{\theta}_j = \hat{\kappa}_j = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j) o_t}$$

EM algorithm stops when it meets the terminating condition, for example, the difference of current parameter Δ and next parameter $\tilde{\Delta}$ is insignificant. It is possible to define a custom terminating condition. The terminating criterion $P(O/\Delta)$ described in table IV.4.3.2.2 is a suggestion.

Table IV.5.1. EM algorithm applied into learning continuous observation HMM parameter with single PDF

Going back the weather example, there are some states of weather: *sunny*, *cloudy*, and *rainy*. Suppose you are in the room and do not know the weather outside but you are notified air humidity measures as observations from someone else. You can forecast weather based on humidity. However, humidity is not still categorized into discrete values such as *dry*, *dryish*, *damp*, and *soggy*. The humidity is now continuous real number, which is used to illustrate continuous observation HMM. It is required to discuss humidity a little bit.

Absolute humidity of atmosphere is measured as amount of water vapor (kilogram) in 1 cubic meter (m^3) volume of air (Gallová & Kučerka).

$$h = \frac{m_w}{V}$$

Where m_w is the amount of water vapor and V is the volume of air. The SI unit (NIST, 2008) of absolute humidity is kg/m^3 .

The amount of water vapor in the air conforms to environment conditions such as temperature and pressure. Given environment conditions, there is a saturation point at which absolute humidity becomes maximal, denoted h_{max} . *Relative humidity* is ratio of the absolute humidity h to its maximal value h_{max} (Gallová & Kučerka).

$$rh = \frac{h}{h_{max}}$$

The relative humidity rh is always less than or equal to 1. Relative humidity rh is near to 0 then, the air is dry. Relative humidity rh is near to 1 then, the air is soggy. It is comfortable for human if relative humidity is between 0.5 and 0.7. Relative humidity is used in our weather example instead of absolute humidity. Suppose continuous observation sequence is

$$O = \{o_1=0.88, o_2=0.13, o_3=0.38\}$$

These observations are relative humidity measures. The bias for all measures is $\varepsilon=0.01$, for example, the first observation $o_1=0.88$ ranges in interval $[0.88-0.01, 0.88+0.01]$. Given weather HMM Δ whose parameters A and Π specified in tables IV.4.1 and IV.4.2 is shown below:

Weather current day

		(Time point t)		
		sunny	cloudy	rainy
Weather previous day (Time point $t-1$)	sunny	$a_{11}=0.50$	$a_{12}=0.25$	$a_{13}=0.25$
	cloudy	$a_{21}=0.30$	$a_{22}=0.40$	$a_{23}=0.30$
	rainy	$a_{31}=0.25$	$a_{32}=0.25$	$a_{33}=0.50$
		sunny	cloudy	rainy
		$\pi_1=0.33$	$\pi_2=0.33$	$\pi_3=0.33$

The observation probability distribution B now includes three normal PDF (s): $p_1(o_t|\theta_1)$, $p_2(o_t|\theta_2)$, and $p_3(o_t|\theta_3)$ corresponding to three states: $s_1=sunny$, $s_2=cloudy$, and $s_3=rainy$. Following is the specification of these normal PDF (s).

$$p_1(o_t|\theta_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2} \frac{(o_t - m_1)^2}{\sigma_1^2}\right)$$

$$p_2(o_t|\theta_2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2} \frac{(o_t - m_2)^2}{\sigma_2^2}\right)$$

$$p_3(o_t|\theta_3) = \frac{1}{\sqrt{2\pi\sigma_3^2}} \exp\left(-\frac{1}{2} \frac{(o_t - m_3)^2}{\sigma_3^2}\right)$$

Where $\theta_1 = (m_1, \sigma_1^2)$, $\theta_2 = (m_2, \sigma_2^2)$, and $\theta_3 = (m_3, \sigma_3^2)$ are means and variances of $p_1(o_t|\theta_1)$, $p_2(o_t|\theta_2)$, and $p_3(o_t|\theta_3)$.

As a convention, observation PDF (s) such as $p_1(o_t|\theta_1)$, $p_2(o_t|\theta_2)$, and $p_3(o_t|\theta_3)$ are represented by theirs means and variances (m_1, σ_1^2) , (m_2, σ_2^2) , and (m_3, σ_3^2) . These means and variances are also called *observation probability parameters* that substitute for discrete matrix B . Table IV.5.2 shows observation probability parameters for our weather example.

$p_1(o_t \theta_1)$	$m_1 = 0.87$	$\sigma_1^2 = 0.9$
$p_2(o_t \theta_2)$	$m_2 = 0.14$	$\sigma_2^2 = 0.9$
$p_3(o_t \theta_3)$	$m_3 = 0.39$	$\sigma_3^2 = 0.9$

Table IV.5.2. Observation probability parameters for weather example

Obviously, we have:

$$p_1(o_t|\theta_1) = \frac{1}{\sqrt{2\pi 0.9}} \exp\left(-\frac{1}{2} \frac{(o_t - 0.87)^2}{0.9}\right)$$

$$p_2(o_t|\theta_2) = \frac{1}{\sqrt{2\pi 0.9}} \exp\left(-\frac{1}{2} \frac{(o_t - 0.14)^2}{0.9}\right)$$

$$p_3(o_t|\theta_3) = \frac{1}{\sqrt{2\pi 0.9}} \exp\left(-\frac{1}{2} \frac{(o_t - 0.39)^2}{0.9}\right)$$

EM algorithm described in table IV.5.1 is applied into calculating the parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{\theta}_j, \hat{\pi}_j)$ given continuous observation sequence $O = \{o_1=0.88, o_2=0.13, o_3=0.38\}$ and continuous normal PDF (s) whose means and variances shown in table IV.5.2. For convenience, all floating-point values are rounded off until ten decimal numbers.

As a convention, let

$$b_j(o_t) = \int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o|\theta_j) do = \Phi\left(\frac{o_t + \varepsilon - m_j}{\sqrt{\sigma_j^2}}\right) - \Phi\left(\frac{o_t - \varepsilon - m_j}{\sqrt{\sigma_j^2}}\right)$$

At the first iteration ($r=1$) we have:

$$\begin{aligned} b_1(o_1) &= b_1(0.88) = \int_{0.88-0.01}^{0.88+0.01} p_1(o|\theta_1) do \\ &= \Phi\left(\frac{0.88 + 0.01 - 0.87}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.88 - 0.01 - 0.87}{\sqrt{0.9}}\right) = 0.00841 \end{aligned}$$

$$\begin{aligned} b_1(o_2) &= b_1(0.13) = \int_{0.13-0.01}^{0.13+0.01} p_1(o|\theta_1) do \\ &= \Phi\left(\frac{0.13 + 0.01 - 0.87}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.13 - 0.01 - 0.87}{\sqrt{0.9}}\right) = 0.006204 \end{aligned}$$

$$\begin{aligned} b_1(o_3) &= b_1(0.38) = \int_{0.38-0.01}^{0.38+0.01} p_1(o|\theta_1) do \\ &= \Phi\left(\frac{0.38 + 0.01 - 0.87}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.38 - 0.01 - 0.87}{\sqrt{0.9}}\right) = 0.00736 \end{aligned}$$

$$\begin{aligned} b_2(o_1) &= b_2(0.88) = \int_{0.13+0.01}^{0.88-0.01} p_2(o|\theta_2) do \\ &= \Phi\left(\frac{0.88 + 0.01 - 0.14}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.88 - 0.01 - 0.14}{\sqrt{0.9}}\right) = 0.006204 \end{aligned}$$

$$\begin{aligned} b_2(o_2) &= b_2(0.13) = \int_{0.38+0.01}^{0.13-0.01} p_2(o|\theta_2) do \\ &= \Phi\left(\frac{0.13 + 0.01 - 0.14}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.13 - 0.01 - 0.14}{\sqrt{0.9}}\right) = 0.00841 \end{aligned}$$

$$\begin{aligned} b_2(o_3) &= b_2(0.38) = \int_{0.88+0.01}^{0.38-0.01} p_2(o|\theta_2) do \\ &= \Phi\left(\frac{0.38 + 0.01 - 0.14}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.38 - 0.01 - 0.14}{\sqrt{0.9}}\right) = 0.008145 \end{aligned}$$

$$\begin{aligned} b_3(o_1) &= b_3(0.88) = \int_{0.13+0.01}^{0.88-0.01} p_3(o|\theta_3) do \\ &= \Phi\left(\frac{0.88 + 0.01 - 0.39}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.88 - 0.01 - 0.39}{\sqrt{0.9}}\right) = 0.00736 \end{aligned}$$

$$\begin{aligned} b_3(o_2) &= b_3(0.13) = \int_{0.88-0.01}^{0.13+0.01} p_3(o|\theta_3) do \\ &= \Phi\left(\frac{0.13 + 0.01 - 0.39}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.13 - 0.01 - 0.39}{\sqrt{0.9}}\right) = 0.0081 \end{aligned}$$

$$b_3(o_3) = b_3(0.38) = \int_{0.38-0.01}^{0.38+0.01} p_3(o|\theta_3) do \\ = \Phi\left(\frac{0.38 + 0.01 - 0.39}{\sqrt{0.9}}\right) - \Phi\left(\frac{0.38 - 0.01 - 0.39}{\sqrt{0.9}}\right) = 0.00841$$

$$\alpha_1(1) = b_1(o_1)\pi_1 = 0.002775$$

$$\alpha_1(2) = b_2(o_1)\pi_2 = 0.002047$$

$$\alpha_1(3) = b_3(o_1)\pi_3 = 0.002429$$

$$\alpha_2(1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_1(o_2) = 0.000016$$

$$\alpha_2(2) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_2(o_2) = 0.000018$$

$$\alpha_2(3) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_3(o_2) = 0.00002$$

$$\alpha_3(1) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_1(o_3) = 0.00000014$$

$$\alpha_3(2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_2(o_3) = 0.00000013$$

$$\alpha_3(3) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i3} \right) b_3(o_3) = 0.00000016$$

$$\beta_3(1) = \beta_3(2) = \beta_3(3) = 1$$

$$\beta_2(1) = \sum_{j=1}^n a_{1j} b_j(o_3) \beta_3(j) = 0.007819$$

$$\beta_2(2) = \sum_{j=1}^n a_{2j} b_j(o_3) \beta_3(j) = 0.007989$$

$$\beta_2(3) = \sum_{j=1}^n a_{3j} b_j(o_3) \beta_3(j) = 0.008081$$

$$\beta_1(1) = \sum_{j=1}^n a_{1j} b_j(o_2) \beta_2(j) = 0.000057$$

$$\beta_1(2) = \sum_{j=1}^n a_{2j} b_j(o_2) \beta_2(j) = 0.000061$$

$$\beta_1(3) = \sum_{j=1}^n a_{3j} b_j(o_2) \beta_2(j) = 0.000062$$

Within the E-step of the first iteration ($r=1$), the terminating criterion $P(O|\Delta)$ is calculated according to forward-backward procedure (see table IV.4.1.1) as follows:

$$P(O|\Delta) = \alpha_3(1) + \alpha_3(2) + \alpha_3(3) = 0.00000043$$

Within the E-step of the first iteration ($r=1$), the joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ are calculated based on formula IV.5.2 as follows:

$$\begin{aligned}
 \xi_2(1,1) &= \alpha_1(1)a_{11}b_1(o_2)\beta_2(1) = 0.00000007 \\
 \xi_2(1,2) &= \alpha_1(1)a_{12}b_2(o_2)\beta_2(2) = 0.00000005 \\
 \xi_2(1,3) &= \alpha_1(1)a_{13}b_3(o_2)\beta_2(3) = 0.00000005 \\
 \xi_2(2,1) &= \alpha_1(2)a_{21}b_1(o_2)\beta_2(1) = 0.00000003 \\
 \xi_2(2,2) &= \alpha_1(2)a_{22}b_2(o_2)\beta_2(2) = 0.00000006 \\
 \xi_2(2,3) &= \alpha_1(2)a_{23}b_3(o_2)\beta_2(3) = 0.00000004 \\
 \xi_2(3,1) &= \alpha_1(3)a_{31}b_1(o_2)\beta_2(1) = 0.00000003 \\
 \xi_2(3,2) &= \alpha_1(3)a_{32}b_2(o_2)\beta_2(2) = 0.00000004 \\
 \xi_2(3,3) &= \alpha_1(3)a_{33}b_3(o_2)\beta_2(3) = 0.00000008 \\
 \xi_3(1,1) &= \alpha_2(1)a_{11}b_1(o_3)\beta_3(1) = 0.00000006 \\
 \xi_3(1,2) &= \alpha_2(1)a_{12}b_2(o_3)\beta_3(2) = 0.00000003 \\
 \xi_3(1,3) &= \alpha_2(1)a_{13}b_3(o_3)\beta_3(3) = 0.00000003 \\
 \xi_3(2,1) &= \alpha_2(2)a_{21}b_1(o_3)\beta_3(1) = 0.00000004 \\
 \xi_3(2,2) &= \alpha_2(2)a_{22}b_2(o_3)\beta_3(2) = 0.00000006 \\
 \xi_3(2,3) &= \alpha_2(2)a_{23}b_3(o_3)\beta_3(3) = 0.00000005 \\
 \xi_3(3,1) &= \alpha_2(3)a_{31}b_1(o_3)\beta_3(1) = 0.00000004 \\
 \xi_3(3,2) &= \alpha_2(3)a_{32}b_2(o_3)\beta_3(2) = 0.00000004 \\
 \xi_3(3,3) &= \alpha_2(3)a_{33}b_3(o_3)\beta_3(3) = 0.00000009
 \end{aligned}$$

$$\begin{aligned}
 \gamma_1(1) &= \alpha_1(1)\beta_1(1) = 0.00000016 \\
 \gamma_1(2) &= \alpha_1(2)\beta_1(2) = 0.00000013 \\
 \gamma_1(3) &= \alpha_1(3)\beta_1(3) = 0.00000015 \\
 \gamma_2(1) &= \alpha_2(1)\beta_2(1) = 0.00000013 \\
 \gamma_2(2) &= \alpha_2(2)\beta_2(2) = 0.00000014 \\
 \gamma_2(3) &= \alpha_2(3)\beta_2(3) = 0.00000017 \\
 \gamma_3(1) &= \alpha_3(1)\beta_3(1) = 0.00000014 \\
 \gamma_3(2) &= \alpha_3(2)\beta_3(2) = 0.00000013 \\
 \gamma_3(3) &= \alpha_3(3)\beta_3(3) = 0.00000016
 \end{aligned}$$

Within the M-step of the first iteration ($r=1$), the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is calculated based on joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ determined at E-step.

$$\begin{aligned}
 \hat{a}_{11} &= \frac{\sum_{t=2}^3 \xi_t(1,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.443786 \\
 \hat{a}_{12} &= \frac{\sum_{t=2}^3 \xi_t(1,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.278330 \\
 \hat{a}_{13} &= \frac{\sum_{t=2}^3 \xi_t(1,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.277883 \\
 \hat{a}_{21} &= \frac{\sum_{t=2}^3 \xi_t(2,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.258587 \\
 \hat{a}_{22} &= \frac{\sum_{t=2}^3 \xi_t(2,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.422909 \\
 \hat{a}_{23} &= \frac{\sum_{t=2}^3 \xi_t(2,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.318504 \\
 \hat{a}_{31} &= \frac{\sum_{t=2}^3 \xi_t(3,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.212952
 \end{aligned}$$

$$\hat{a}_{32} = \frac{\sum_{t=2}^3 \xi_t(3,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.261709$$

$$\hat{a}_{33} = \frac{\sum_{t=2}^3 \xi_t(3,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.525339$$

$$\hat{m}_1 = \frac{\sum_{t=1}^3 \gamma_t(1)o_t}{\sum_{t=1}^3 \gamma_t(1)} = 0.493699$$

$$\hat{\sigma}_1^2 = \frac{\sum_{t=1}^3 \gamma_t(1)(o_t - \hat{m}_1)^2}{\sum_{t=1}^3 \gamma_t(1)} = 0.100098$$

$$\hat{m}_2 = \frac{\sum_{t=1}^3 \gamma_t(2)o_t}{\sum_{t=1}^3 \gamma_t(2)} = 0.447242$$

$$\hat{\sigma}_2^2 = \frac{\sum_{t=1}^3 \gamma_t(2)(o_t - \hat{m}_2)^2}{\sum_{t=1}^3 \gamma_t(2)} = 0.095846$$

$$\hat{m}_3 = \frac{\sum_{t=1}^3 \gamma_t(3)o_t}{\sum_{t=1}^3 \gamma_t(3)} = 0.450017$$

$$\hat{\sigma}_3^2 = \frac{\sum_{t=1}^3 \gamma_t(3)(o_t - \hat{m}_3)^2}{\sum_{t=1}^3 \gamma_t(3)} = 0.094633$$

$$\hat{\pi}_1 = \frac{\gamma_1(1)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.367053$$

$$\hat{\pi}_2 = \frac{\gamma_1(2)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.288002$$

$$\hat{\pi}_3 = \frac{\gamma_1(3)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.344945$$

At the second iteration ($r=2$), the current parameter $\Delta = (a_{ij}, \theta_j, \pi_j)$ is received values from the previous estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{\theta}_j, \hat{\pi}_j)$, as seen in table IV.5.3.

$a_{11} = \hat{a}_{11} = 0.443786$	$a_{12} = \hat{a}_{12} = 0.278330$	$a_{13} = \hat{a}_{13} = 0.277883$
$a_{21} = \hat{a}_{21} = 0.258587$	$a_{22} = \hat{a}_{22} = 0.422909$	$a_{23} = \hat{a}_{23} = 0.318504$
$a_{31} = \hat{a}_{31} = 0.212952$	$a_{32} = \hat{a}_{32} = 0.261709$	$a_{33} = \hat{a}_{33} = 0.525339$
<hr/>		
$m_1 = \hat{m}_1 = 0.493699$	$\sigma_1^2 = \hat{\sigma}_1^2 = 0.100098$	
$m_2 = \hat{m}_2 = 0.447242$	$\sigma_2^2 = \hat{\sigma}_2^2 = 0.095846$	
$m_3 = \hat{m}_3 = 0.450017$	$\sigma_3^2 = \hat{\sigma}_3^2 = 0.094633$	
<hr/>		
$\pi_1 = \hat{\pi}_1 = 0.367053$	$\pi_2 = \hat{\pi}_2 = 0.288002$	$\pi_3 = \hat{\pi}_3 = 0.344945$
<hr/>		
Terminating criterion $P(O/\Delta) = 0.00000043$		

Table IV.5.3. Continuous observation HMM parameters resulted from the first iteration of EM algorithm

We have:

$$\begin{aligned}
 b_1(o_1) &= b_1(0.88) = \int_{\substack{0.88-0.01 \\ 0.13+0.01}}^{0.88+0.01} p_1(o|\theta_1) do \\
 &= \Phi\left(\frac{0.88 + 0.01 - 0.493699}{\sqrt{0.100098}}\right) - \Phi\left(\frac{0.88 - 0.01 - 0.493699}{\sqrt{0.100098}}\right) \\
 &= 0.011968 \\
 b_1(o_2) &= b_1(0.13) = \int_{\substack{0.13-0.01 \\ 0.38+0.01}}^{0.13+0.01} p_1(o|\theta_1) do \\
 &= \Phi\left(\frac{0.13 + 0.01 - 0.493699}{\sqrt{0.100098}}\right) - \Phi\left(\frac{0.13 - 0.01 - 0.493699}{\sqrt{0.100098}}\right) \\
 &= 0.013026 \\
 b_1(o_3) &= b_1(0.38) = \int_{\substack{0.38-0.01 \\ 0.13+0.01}}^{0.38+0.01} p_1(o|\theta_1) do \\
 &= \Phi\left(\frac{0.38 + 0.01 - 0.493699}{\sqrt{0.100098}}\right) - \Phi\left(\frac{0.38 - 0.01 - 0.493699}{\sqrt{0.100098}}\right) \\
 &= 0.023639 \\
 b_2(o_1) &= b_2(0.88) = \int_{\substack{0.88-0.01 \\ 0.13+0.01}}^{0.88+0.01} p_2(o|\theta_2) do \\
 &= \Phi\left(\frac{0.88 + 0.01 - 0.447242}{\sqrt{0.095846}}\right) - \Phi\left(\frac{0.88 - 0.01 - 0.447242}{\sqrt{0.095846}}\right) \\
 &= 0.009703 \\
 b_2(o_2) &= b_2(0.13) = \int_{\substack{0.13-0.01 \\ 0.38+0.01}}^{0.13+0.01} p_2(o|\theta_2) do \\
 &= \Phi\left(\frac{0.13 + 0.01 - 0.447242}{\sqrt{0.095846}}\right) - \Phi\left(\frac{0.13 - 0.01 - 0.447242}{\sqrt{0.095846}}\right) \\
 &= 0.015246 \\
 b_2(o_3) &= b_2(0.38) = \int_{\substack{0.38-0.01 \\ 0.13+0.01}}^{0.38+0.01} p_2(o|\theta_2) do \\
 &= \Phi\left(\frac{0.38 + 0.01 - 0.447242}{\sqrt{0.095846}}\right) - \Phi\left(\frac{0.38 - 0.01 - 0.447242}{\sqrt{0.095846}}\right) \\
 &= 0.025167 \\
 b_3(o_1) &= b_3(0.88) = \int_{\substack{0.88-0.01 \\ 0.13+0.01}}^{0.88+0.01} p_3(o|\theta_3) do \\
 &= \Phi\left(\frac{0.88 + 0.01 - 0.450017}{\sqrt{0.094633}}\right) - \Phi\left(\frac{0.88 - 0.01 - 0.450017}{\sqrt{0.094633}}\right) \\
 &= 0.009767 \\
 b_3(o_2) &= b_3(0.13) = \int_{\substack{0.13-0.01 \\ 0.38+0.01}}^{0.13+0.01} p_3(o|\theta_3) do \\
 &= \Phi\left(\frac{0.13 + 0.01 - 0.450017}{\sqrt{0.094633}}\right) - \Phi\left(\frac{0.13 - 0.01 - 0.450017}{\sqrt{0.094633}}\right) \\
 &= 0.015098
 \end{aligned}$$

$$\begin{aligned}
 b_3(o_3) &= b_3(0.38) = \int_{0.38-0.01}^{0.38+0.01} p_3(o|\theta_3) do \\
 &= \Phi\left(\frac{0.38 + 0.01 - 0.450017}{\sqrt{0.094633}}\right) - \Phi\left(\frac{0.38 - 0.01 - 0.450017}{\sqrt{0.094633}}\right) \\
 &= 0.025269
 \end{aligned}$$

$$\alpha_1(1) = b_1(o_1)\pi_1 = 0.004393$$

$$\alpha_1(2) = b_2(o_1)\pi_2 = 0.002794$$

$$\alpha_1(3) = b_3(o_1)\pi_3 = 0.003369$$

$$\alpha_2(1) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_1(o_2) = 0.000044$$

$$\alpha_2(2) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_2(o_2) = 0.00005$$

$$\alpha_2(3) = \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_3(o_2) = 0.000059$$

$$\alpha_3(1) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_1(o_3) = 0.000001$$

$$\alpha_3(2) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i2} \right) b_2(o_3) = 0.000001$$

$$\alpha_3(3) = \left(\sum_{i=1}^3 \alpha_2(i)a_{i3} \right) b_3(o_3) = 0.000001$$

$$\beta_3(1) = \beta_3(2) = \beta_3(3) = 1$$

$$\beta_2(1) = \sum_{j=1}^n a_{1j} b_j(o_3) \beta_3(j) = 0.024517$$

$$\beta_2(2) = \sum_{j=1}^n a_{2j} b_j(o_3) \beta_3(j) = 0.024805$$

$$\beta_2(3) = \sum_{j=1}^n a_{3j} b_j(o_3) \beta_3(j) = 0.024895$$

$$\beta_1(1) = \sum_{j=1}^n a_{1j} b_j(o_2) \beta_2(j) = 0.000351$$

$$\beta_1(2) = \sum_{j=1}^n a_{2j} b_j(o_2) \beta_2(j) = 0.000362$$

$$\beta_1(3) = \sum_{j=1}^n a_{3j} b_j(o_2) \beta_2(j) = 0.000364$$

Within the E-step of the second iteration ($r=2$), the terminating criterion $P(O|\Delta)$ is calculated according to forward-backward procedure (see table IV.4.1.1) as follows:

$$P(O|\Delta) = \alpha_3(1) + \alpha_3(2) + \alpha_3(3) = 0.000004$$

Within the E-step of the second iteration ($r=2$), the joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ are calculated based on formula IV.5.2 as follows:

$$\begin{aligned}\xi_2(1,1) &= \alpha_1(1)a_{11}b_1(o_2)\beta_2(1) = 0.00000062 \\ \xi_2(1,2) &= \alpha_1(1)a_{12}b_2(o_2)\beta_2(2) = 0.00000046 \\ \xi_2(1,3) &= \alpha_1(1)a_{13}b_3(o_2)\beta_2(3) = 0.00000046 \\ \xi_2(2,1) &= \alpha_1(2)a_{21}b_1(o_2)\beta_2(1) = 0.00000023 \\ \xi_2(2,2) &= \alpha_1(2)a_{22}b_2(o_2)\beta_2(2) = 0.00000045 \\ \xi_2(2,3) &= \alpha_1(2)a_{23}b_3(o_2)\beta_2(3) = 0.00000033 \\ \xi_2(3,1) &= \alpha_1(3)a_{31}b_1(o_2)\beta_2(1) = 0.00000023 \\ \xi_2(3,2) &= \alpha_1(3)a_{32}b_2(o_2)\beta_2(2) = 0.00000033 \\ \xi_2(3,3) &= \alpha_1(3)a_{33}b_3(o_2)\beta_2(3) = 0.00000067 \\ \xi_3(1,1) &= \alpha_2(1)a_{11}b_1(o_3)\beta_3(1) = 0.00000046 \\ \xi_3(1,2) &= \alpha_2(1)a_{12}b_2(o_3)\beta_3(2) = 0.00000031 \\ \xi_3(1,3) &= \alpha_2(1)a_{13}b_3(o_3)\beta_3(3) = 0.00000031 \\ \xi_3(2,1) &= \alpha_2(2)a_{21}b_1(o_3)\beta_3(1) = 0.00000031 \\ \xi_3(2,2) &= \alpha_2(2)a_{22}b_2(o_3)\beta_3(2) = 0.00000053 \\ \xi_3(2,3) &= \alpha_2(2)a_{23}b_3(o_3)\beta_3(3) = 0.00000040 \\ \xi_3(3,1) &= \alpha_2(3)a_{31}b_1(o_3)\beta_3(1) = 0.00000029 \\ \xi_3(3,2) &= \alpha_2(3)a_{32}b_2(o_3)\beta_3(2) = 0.00000039 \\ \xi_3(3,3) &= \alpha_2(3)a_{33}b_3(o_3)\beta_3(3) = 0.00000078\end{aligned}$$

$$\begin{aligned}\gamma_1(1) &= \alpha_1(1)\beta_1(1) = 0.0000015 \\ \gamma_1(2) &= \alpha_1(2)\beta_1(2) = 0.0000010 \\ \gamma_1(3) &= \alpha_1(3)\beta_1(3) = 0.0000012 \\ \gamma_2(1) &= \alpha_2(1)\beta_2(1) = 0.0000011 \\ \gamma_2(2) &= \alpha_2(2)\beta_2(2) = 0.0000012 \\ \gamma_2(3) &= \alpha_2(3)\beta_2(3) = 0.0000015 \\ \gamma_3(1) &= \alpha_3(1)\beta_3(1) = 0.0000011 \\ \gamma_3(2) &= \alpha_3(2)\beta_3(2) = 0.0000012 \\ \gamma_3(3) &= \alpha_3(3)\beta_3(3) = 0.0000015\end{aligned}$$

Within the M-step of the second iteration ($r=2$), the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is calculated based on joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ determined at E-step.

$$\begin{aligned}\hat{a}_{11} &= \frac{\sum_{t=2}^3 \xi_t(1,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.413419 \\ \hat{a}_{12} &= \frac{\sum_{t=2}^3 \xi_t(1,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.293817 \\ \hat{a}_{13} &= \frac{\sum_{t=2}^3 \xi_t(1,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.292764 \\ \hat{a}_{21} &= \frac{\sum_{t=2}^3 \xi_t(2,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.238147 \\ \hat{a}_{22} &= \frac{\sum_{t=2}^3 \xi_t(2,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.434668 \\ \hat{a}_{23} &= \frac{\sum_{t=2}^3 \xi_t(2,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.327184 \\ \hat{a}_{31} &= \frac{\sum_{t=2}^3 \xi_t(3,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.195073\end{aligned}$$

$$\hat{a}_{32} = \frac{\sum_{t=2}^3 \xi_t(3,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.267764$$

$$\hat{a}_{33} = \frac{\sum_{t=2}^3 \xi_t(3,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.537163$$

$$\hat{m}_1 = \frac{\sum_{t=1}^3 \gamma_t(1)o_t}{\sum_{t=1}^3 \gamma_t(1)} = 0.515827$$

$$\hat{\sigma}_1^2 = \frac{\sum_{t=1}^3 \gamma_t(1)(o_t - \hat{m}_1)^2}{\sum_{t=1}^3 \gamma_t(1)} = 0.104459$$

$$\hat{m}_2 = \frac{\sum_{t=1}^3 \gamma_t(2)o_t}{\sum_{t=1}^3 \gamma_t(2)} = 0.436110$$

$$\hat{\sigma}_2^2 = \frac{\sum_{t=1}^3 \gamma_t(2)(o_t - \hat{m}_2)^2}{\sum_{t=1}^3 \gamma_t(2)} = 0.091798$$

$$\hat{m}_3 = \frac{\sum_{t=1}^3 \gamma_t(3)o_t}{\sum_{t=1}^3 \gamma_t(3)} = 0.439658$$

$$\hat{\sigma}_3^2 = \frac{\sum_{t=1}^3 \gamma_t(3)(o_t - \hat{m}_3)^2}{\sum_{t=1}^3 \gamma_t(3)} = 0.091739$$

$$\hat{\pi}_1 = \frac{\gamma_1(1)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.407999$$

$$\hat{\pi}_2 = \frac{\gamma_1(2)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.267524$$

$$\hat{\pi}_3 = \frac{\gamma_1(3)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.324477$$

Table IV.5.4 summarizes HMM parameters resulted from the first iteration and the second iteration of EM algorithm.

Iteration	HMM parameters		
1 st	$\hat{a}_{11} = 0.443786$	$\hat{a}_{12} = 0.278330$	$\hat{a}_{13} = 0.277883$
	$\hat{a}_{21} = 0.258587$	$\hat{a}_{22} = 0.422909$	$\hat{a}_{23} = 0.318504$
	$\hat{a}_{31} = 0.212952$	$\hat{a}_{32} = 0.261709$	$\hat{a}_{33} = 0.525339$
	$\hat{m}_1 = 0.493699$	$\hat{\sigma}_1^2 = 0.100098$	
	$\hat{m}_2 = 0.447242$	$\hat{\sigma}_2^2 = 0.095846$	
	$\hat{m}_3 = 0.450017$	$\hat{\sigma}_3^2 = 0.094633$	
	$\hat{\pi}_1 = 0.367053$	$\hat{\pi}_2 = 0.288002$	$\hat{\pi}_3 = 0.344945$
	Terminating criterion $P(O/\Delta) = 0.00000043$		
	$\hat{a}_{11} = 0.413419$	$\hat{a}_{12} = 0.293817$	$\hat{a}_{13} = 0.292764$
2 nd	$\hat{a}_{21} = 0.238147$	$\hat{a}_{22} = 0.434668$	$\hat{a}_{23} = 0.327184$
	$\hat{a}_{31} = 0.195073$	$\hat{a}_{32} = 0.267764$	$\hat{a}_{33} = 0.537163$
	$\hat{m}_1 = 0.515827$	$\hat{\sigma}_1^2 = 0.104459$	
	$\hat{m}_2 = 0.436110$	$\hat{\sigma}_2^2 = 0.091798$	

$\hat{m}_3 = 0.439658$	$\hat{\sigma}_3^2 = 0.091739$
$\hat{\pi}_1 = 0.407999$	$\hat{\pi}_2 = 0.267524$
Terminating criterion $P(O \Delta) = 0.000004$	

Table IV.5.4. Continuous observation HMM parameters resulted from the first iteration and the second iteration of EM algorithm

As seen in table IV.5.4, the EM algorithm does not converge yet when it produces two different terminating criteria at the first iteration and the second iteration. It is necessary to run more iterations so as to gain the most optimal estimate. Within this example, the EM algorithm converges absolutely after 14 iterations when the criterion $P(O|\Delta)$ approaches to the same value 1 at the 13rd and 14th iterations. Table IV.5.5 shows HMM parameter estimates along with terminating criterion $P(O|\Delta)$ at the 1st, 2nd, 13rd, and 14th iterations of EM algorithm.

Iteration	HMM parameters		
1 st	$\hat{a}_{11} = 0.443786$	$\hat{a}_{12} = 0.278330$	$\hat{a}_{13} = 0.277883$
	$\hat{a}_{21} = 0.258587$	$\hat{a}_{22} = 0.422909$	$\hat{a}_{23} = 0.318504$
	$\hat{a}_{31} = 0.212952$	$\hat{a}_{32} = 0.261709$	$\hat{a}_{33} = 0.525339$
	$\hat{m}_1 = 0.493699$	$\hat{\sigma}_1^2 = 0.100098$	
	$\hat{m}_2 = 0.447242$	$\hat{\sigma}_2^2 = 0.095846$	
	$\hat{m}_3 = 0.450017$	$\hat{\sigma}_3^2 = 0.094633$	
	$\hat{\pi}_1 = 0.367053$	$\hat{\pi}_2 = 0.288002$	$\hat{\pi}_3 = 0.344945$
	Terminating criterion $P(O \Delta) = 0.00000043$		
	$\hat{a}_{11} = 0.413419$	$\hat{a}_{12} = 0.293817$	$\hat{a}_{13} = 0.292764$
2 nd	$\hat{a}_{21} = 0.238147$	$\hat{a}_{22} = 0.434668$	$\hat{a}_{23} = 0.327184$
	$\hat{a}_{31} = 0.195073$	$\hat{a}_{32} = 0.267764$	$\hat{a}_{33} = 0.537163$
	$\hat{m}_1 = 0.515827$	$\hat{\sigma}_1^2 = 0.104459$	
	$\hat{m}_2 = 0.436110$	$\hat{\sigma}_2^2 = 0.091798$	
	$\hat{m}_3 = 0.439658$	$\hat{\sigma}_3^2 = 0.091739$	
	$\hat{\pi}_1 = 0.407999$	$\hat{\pi}_2 = 0.267524$	$\hat{\pi}_3 = 0.324477$
	Terminating criterion $P(O \Delta) = 0.000004$		
	$\hat{a}_{11} = 0$	$\hat{a}_{12} = 1$	$\hat{a}_{13} = 0$
	$\hat{a}_{21} = 0$	$\hat{a}_{22} = 0$	$\hat{a}_{23} = 1$
13 rd	$\hat{a}_{31} = 0$	$\hat{a}_{32} = 0$	$\hat{a}_{33} = 1$
	$\hat{m}_1 = 0.88$	$\hat{\sigma}_1^2 = 4.2 * 10^{-9}$	
	$\hat{m}_2 = 0.13$	$\hat{\sigma}_2^2 = 1.4 * 10^{-14}$	
	$\hat{m}_3 = 0.38$	$\hat{\sigma}_3^2 = 1.0 * 10^{-21}$	
	$\hat{\pi}_1 = 1$	$\hat{\pi}_2 = 0$	$\hat{\pi}_3 = 0$

		Terminating criterion $P(O/\Delta) = 1$		
14 th	$\hat{a}_{11} = 0$	$\hat{a}_{12} = 1$	$\hat{a}_{13} = 0$	
	$\hat{a}_{21} = 0$	$\hat{a}_{22} = 0$	$\hat{a}_{23} = 1$	
	$\hat{a}_{31} = 0$	$\hat{a}_{32} = 0$	$\hat{a}_{33} = 1$	
	$\hat{m}_1 = 0.88$	$\hat{\sigma}_1^2 = 4.2 * 10^{-9}$		
	$\hat{m}_2 = 0.13$	$\hat{\sigma}_2^2 = 1.4 * 10^{-14}$		
	$\hat{m}_3 = 0.38$	$\hat{\sigma}_3^2 = 1.0 * 10^{-21}$		
	$\hat{\pi}_1 = 1$	$\hat{\pi}_2 = 0$	$\hat{\pi}_3 = 0$	
	Terminating criterion $P(O/\Delta) = 1$			

Table IV.5.5. Continuous observation HMM parameters along with terminating criteria after 14 iterations of EM algorithm

As a result, the learned parameters A , B , and Π are shown in table IV.5.6:

		Weather current day (Time point t)		
		<i>sunny</i>	<i>cloudy</i>	<i>rainy</i>
Weather previous day (Time point $t-1$)	<i>sunny</i>	$a_{11}=0$	$a_{12}=1$	$a_{13}=0$
	<i>cloudy</i>	$a_{21}=0$	$a_{22}=0$	$a_{23}=1$
	<i>rainy</i>	$a_{31}=0$	$a_{32}=0$	$a_{33}=1$
		<i>sunny</i>	<i>cloudy</i>	<i>rainy</i>
		$\pi_1=1$	$\pi_2=0$	$\pi_3=0$
Humidity				
Weather	<i>sunny</i>	$p_1(o_t \theta_1)$	$m_1 = 0.88$	$\sigma_1^2 = 4.2 * 10^{-9}$
	<i>cloudy</i>	$p_2(o_t \theta_2)$	$m_2 = 0.13$	$\sigma_2^2 = 1.4 * 10^{-14}$
	<i>rainy</i>	$p_3(o_t \theta_3)$	$m_3 = 0.38$	$\sigma_3^2 = 1.0 * 10^{-21}$

Table IV.5.6. Continuous observation HMM parameters of weather example learned from EM algorithm

Such learned parameters are more appropriate to the continuous observation sequence $O = \{o_1=0.88, o_2=0.13, o_3=0.38\}$ than the original ones shown in tables IV.4.1, IV.4.2, and IV.5.2.

Suppose observation o_t is result of experiment that is done at laboratory with K methods or equipments. Each observation is product of K methods or equipments. Given aforementioned weather example, humidity is measured at time point t by $K=2$ equipments. Equipment 1 and equipment 2 produces two humidity measures $a_1=0.63$ and $a_2=0.88$, respectively at time point t . Hence, the observation o_t receives a random value among $a_1=0.63$ and $a_2=0.88$. If there is 60% that equipment 1 is selected then, we have $o_t=a_1=0.63$ with confidence 60%. The percentage 60% is called normalized weight of equipment 1 and so, it is easy to infer that normalized weight of equipment 2 is 40%.

Shortly, observation o_t is created from mixture of K methods or equipments. If the observation o_t is characterized by a PDF as aforementioned in formula IV.5.1, this

PDF is constituted of K partial PDF (s) and each partial PDF corresponds to one method or equipment. Such PDF is called *mixture model PDF*, specified by formula [IV.5.7](#) (Rabiner, 1989, p. 267).

$$b_j(o_t) = p_j(o_t | \theta_j) = \sum_{k=1}^K c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})$$

Where $c_j^{(k)}$ are non-negative *normalized weights*, $0 \leq c_j^{(k)} \leq 1$ such that

$$\sum_{k=1}^K c_j^{(k)} = 1$$

Formula IV.5.7. Mixture model probability density function (PDF) of observation

Each *partial PDF* $p_j^{(k)}(o_t | \theta_j^{(k)})$ belongs to any probability distribution, for example, normal distribution, exponential distribution, etc. It is not specified that K partial PDF (s) constructing the formula [IV.5.7](#) must have the same type of distribution but they often share the same distribution type in practice. For example, humidity is measured by $K=2$ equipments where the first equipment produces values conforming normal distribution with mean 0 and variance 1 while the second equipment produces values conforming normal distribution with mean 0.5 and variance 2. The case that two equipments measuring the same metric like humidity produce values in accordance with two different distributions (for example, normal distribution and exponential distribution) is very rare or impossible. The notation $\theta_j^{(k)}$ denotes *partial probabilistic parameters*, for instance, if $p_j^{(k)}(o_t | \theta_j^{(k)})$ is normal distribution PDF, $\theta_j^{(k)}$ includes mean $m_j^{(k)}$ and variance $\sigma_j^{(k)}$. The HMM now is specified by parameter $\Delta = (a_{ij}, c_j^{(k)}, \theta_j^{(k)}, \pi_j)$, which is called *mixture continuous observation HMM*. The main subject of learning problem is to calculate partial estimates $\hat{c}_j^{(k)}$ and $\hat{\theta}_j^{(k)}$ when estimates \hat{a}_{ij} and $\hat{\pi}_j$ are kept intact, as aforementioned in formula [IV.4.3.2.7](#).

$$\begin{aligned}\hat{a}_{ij} &= \frac{\sum_{t=2}^T \xi_t(i, j)}{\sum_{t=2}^T \sum_{l=1}^n \xi_t(i, l)} \\ \hat{\pi}_j &= \frac{\gamma_1(j)}{\sum_{i=1}^n \gamma_1(i)}\end{aligned}$$

The joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ according to formulas [IV.5.2](#).

$$\xi_t(i, j) = \alpha_{t-1}(i) a_{ij} \left(\int_{o_t - \varepsilon}^{o_t + \varepsilon} p_j(o | \theta_j) do \right) \beta_t(j) \text{ where } t \geq 2$$

$$\gamma_t(j) = P(O, x_t = s_j | \Delta) = \alpha_t(j) \beta_t(j)$$

Let $\gamma_t(j, k)$ be joint probability that the stochastic process counting for k^{th} partial PDF is in state s_i at time point t with observation sequence O . Let $\xi_t(i, j, k)$ be the joint probability that the stochastic process counting for k^{th} receives state s_i at time point $t-1$ and state s_j at time point t given observation sequence O . Formula [IV.5.8](#) specifies partial $\gamma_t(j, k)$ and $\xi_t(i, j, k)$.

$$\begin{aligned}\xi_t(i, j, k) &= P^{(k)}(O, x_{t-1} = s_i, x_t = s_j | \Delta) \\ &= \alpha_{t-1}(i, k) a_{ij} \left(\int_{o_{t-\varepsilon}}^{o_{t+\varepsilon}} c_j^{(k)} p_j^{(k)}(o | \theta_j^{(k)}) do \right) \beta_t(j, k) \text{ where } t \geq 2\end{aligned}$$

$$\gamma_t(j, k) = P^{(k)}(O, x_t = s_j | \Delta) = \alpha_t(j, k) \beta_t(j, k)$$

Where partial forward variable $\alpha_{t+1}(j, k)$ and partial backward variable $\beta_t(i, k)$ count for k^{th} partial PDF.

$$\begin{aligned}\alpha_{t+1}(j, k) &= \left(\sum_{i=1}^n \alpha_t(i, k) a_{ij} \right) \int_{o_{t+1-\varepsilon}}^{o_{t+1+\varepsilon}} c_j^{(k)} p_j^{(k)}(o | \theta_j^{(k)}) do \\ \beta_t(i, k) &= \sum_{j=1}^n a_{ij} \left(\int_{o_{t+1-\varepsilon}}^{o_{t+1+\varepsilon}} c_j^{(k)} p_j^{(k)}(o | \theta_j^{(k)}) do \right) \beta_{t+1}(j, k) \\ &\quad \int_{o_t-\varepsilon}^{o_t+\varepsilon} c_j^{(k)} p_j^{(k)}(o | \theta_j^{(k)}) do \\ &= c_j^{(k)} \Phi \left(\frac{o_t + \varepsilon - m_j^{(k)}}{\sqrt{\sigma_j^{(k)}}} \right) \\ &\quad - c_j^{(k)} \Phi \left(\frac{o_t - \varepsilon - m_j^{(k)}}{\sqrt{\sigma_j^{(k)}}} \right) \text{ if } p_j^{(k)}(o | \theta_j^{(k)}) \text{ is normal PDF}\end{aligned}$$

Formula IV.5.8. Partial joint probabilities $\xi_t(i, j, k)$ and $\gamma_t(j, k)$ based on mixture model PDF

Note that $m_j^{(k)}$ and $\sigma_j^{(k)}$ are mean and variance of the normal PDF $p_j^{(k)}(o | \theta_j^{(k)})$ and Φ is cumulative standard normal distribution (Montgomery & Runger, 2003, p. 653).

After comparing formula IV.5.2 with formula IV.5.8, it is easy to draw the relationship between quantities $\xi_t(i, j)$, $\gamma_t(j)$ and partial quantities $\xi_t(i, j, k)$, $\gamma_t(j, k)$, as seen in formula IV.5.9.

$$\begin{aligned}\xi_t(i, j, k) &= \frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \xi_t(i, j) \text{ where } t \geq 2 \\ \gamma_t(j, k) &= \frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \gamma_t(j)\end{aligned}$$

Formula IV.5.9. Relationship between quantities $\xi_t(i, j)$, $\gamma_t(j)$ and partial quantities $\xi_t(i, j, k)$, $\gamma_t(j, k)$

The ratio $\frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})}$ reflects how much the partial PDF $p_j^{(k)}(o_t | \theta_j^{(k)})$ contributes to the mixture model PDF $p_j(o_t | \theta_j)$. Following is the proof of formula IV.5.9.

Given $t \geq 2$, we have:

$$\xi_t(i, j, k) = P^{(k)}(O, x_{t-1} = s_i, x_t = s_j | \Delta)$$

$$= \alpha_{t-1}(i) a_{ij} \left(c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)}) \right) \beta_t(j)$$

(The integral $\int_{o_t - \varepsilon}^{o_t + \varepsilon} c_j^{(k)} p_j^{(k)}(o | \theta_j^{(k)}) do$ shown in formula IV.5.8 is turned back the

$$\text{original PDF } c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})$$

$$= \alpha_{t-1}(i) a_{ij} \left(\frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)}) \right) \beta_t(j)$$

$$= \frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \alpha_{t-1}(i) a_{ij} \left(\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)}) \right) \beta_t(j)$$

$$= \frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \alpha_{t-1}(i) a_{ij} p_j(o_t | \theta_j) \beta_t(j)$$

(due to $p_j(o_t | \theta_j) = \sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})$ according to formula IV.5.7)

$$= \frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \xi_t(i, j)$$

We also have:

$$\gamma_t(j, k) = P^{(k)}(O, x_t = s_j | \Delta) = \sum_{i=1}^n \xi_t(i, j, k)$$

(due to $\gamma_t(j, k) = \sum_{i=1}^n \xi_t(i, j, k)$ according to formula IV.4.3.2.6)

$$= \sum_{i=1}^n \frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \xi_t(i, j)$$

$$= \frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \sum_{i=1}^n \xi_t(i, j)$$

$$= \frac{c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})}{\sum_{l=1}^K c_j^{(l)} p_j^{(l)}(o_t | \theta_j^{(l)})} \gamma_t(j)$$

(due to $\gamma_t(j) = \sum_{i=1}^n \xi_t(i, j)$ according to formula IV.4.3.2.6)

Because estimates \hat{a}_{ij} and $\hat{\pi}_j$ are kept intact, it is necessary to calculate partial estimates $\hat{c}_j^{(k)}$ and $\hat{\theta}_j^{(k)}$. Formula IV.5.10 specifies the EM expectation $E_{X|O, \Delta_r} \{ \ln(P(O, X | \Delta)) \}$ for continuous observation HMM given mixture model PDF.

$$\begin{aligned}
 & E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \} \\
 &= \sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^n \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(a_{ij}) \right. \\
 &\quad \left. + \sum_{j=1}^n \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \ln(c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})) \right)
 \end{aligned}$$

Formula IV.5.10. EM conditional expectation for continuous observation HMM given mixture model PDF

Where $I(x_t = s_j, k)$ is index function such that,

$$I(x_t = s_j, k) = \begin{cases} 1 & \text{if } x_t = s_j \text{ counting for } k^{\text{th}} \text{ partial PDF} \\ 0 & \text{otherwise} \end{cases}$$

Derived from formula IV.4.3.2.3, the HMM Lagrangian function for continuous observation HMM given mixture model PDF and constraint $\sum_{k=1}^K c_j^{(k)} = 1$ is specified by formula IV.5.11.

$$\begin{aligned}
 l(\Delta, \lambda, \mu) &= l(a_{ij}, c_j^{(k)}, \theta_j^{(k)}, \lambda_i, \mu_j) \\
 &= E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \} + \sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \\
 &\quad + \sum_{j=1}^n \mu_j \left(1 - \sum_{k=1}^K c_j^{(k)} \right)
 \end{aligned}$$

Formula IV.5.11. Lagrangian function for continuous observation HMM with single PDF

Where λ is n -component vector $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ and μ is n -component vector $\mu = (\mu_1, \mu_2, \dots, \mu_n)$. Factors $\lambda_i \geq 0$ and $\mu_j \geq 0$ are called Lagrange multipliers or Karush-Kuhn-Tucker multipliers (Wikipedia, Karush–Kuhn–Tucker conditions, 2014) or dual variables.

The estimate $\hat{c}_j^{(k)}$ which is extreme point of the Lagrangian function $l(\Delta, \lambda, \mu)$ is determined by setting partial derivatives of $l(\Delta, \lambda, \mu)$ with respect to $c_j^{(k)}$ to be zero.

The partial derivative of $l(\Delta, \lambda)$ with respect to $c_j^{(k)}$ is:

$$\begin{aligned}
 \frac{\partial l(\Delta, \lambda)}{\partial c_j^{(k)}} &= \frac{\partial E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial c_j^{(k)}} + \frac{\partial}{\partial c_j^{(k)}} \left(\sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \right) \\
 &\quad + \frac{\partial}{\partial c_j^{(k)}} \left(\sum_{j=1}^n \mu_j \left(1 - \sum_{k=1}^K c_j^{(k)} \right) \right) \\
 &= \frac{\partial E_{X|O,\Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial c_j^{(k)}} - \mu_j
 \end{aligned}$$

$$\begin{aligned}
 &= \frac{\partial}{\partial c_j^{(k)}} \left(\sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^n \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(a_{ij}) \right. \right. \\
 &\quad \left. \left. + \sum_{j=1}^n \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \ln(c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})) \right) \right) - \mu_j \\
 &= \left(\sum_X P(X|O, \Delta_r) \sum_{j=1}^n \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \frac{\partial}{\partial c_j^{(k)}} \left(\ln(c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})) \right) \right) - \mu_j \\
 &= \left(\sum_X P(X|O, \Delta_r) \sum_{j=1}^n \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \frac{\partial \left(\ln(c_j^{(k)}) + \ln(p_j^{(k)}(o_t | \theta_j^{(k)})) \right)}{\partial c_j^{(k)}} \right) \\
 &\quad - \mu_j \\
 &= \sum_X P(X|O, \Delta_r) \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \frac{1}{c_j^{(k)}} - \mu_j \\
 &= \sum_{k=1}^K \sum_{t=1}^T \sum_X I(x_t = s_j, k) P(X|O, \Delta_r) \frac{1}{c_j^{(k)}} - \mu_j \\
 &= \sum_{k=1}^K \sum_{t=1}^T \sum_X I(x_t = s_j, k) P(x_1, \dots, x_t, \dots, x_T | O, \Delta_r) \frac{1}{c_j^{(k)}} - \mu_j \\
 &= \sum_{k=1}^K \sum_{t=1}^T I(k) P(x_t = s_j | O, \Delta_r) \frac{1}{c_j^{(k)}} - \mu_j
 \end{aligned}$$

(Due to total probability rule specified by formula III.1.1.4)

Where $I(x_t = s_j, k)$ is index function such that,

$$\begin{aligned}
 I(x_t = s_j, k) &= \begin{cases} 1 & \text{if } x_t = s_j \text{ counting for } k^{\text{th}} \text{ partial PDF} \\ 0 & \text{otherwise} \end{cases} \\
 I(k) &= \begin{cases} 1 & \text{if counting for } k^{\text{th}} \text{ partial PDF} \\ 0 & \text{otherwise} \end{cases}
 \end{aligned}$$

It implies

$$\begin{aligned}
 \frac{\partial l(\Delta, \lambda, \mu)}{\partial c_j^{(k)}} &= \frac{1}{c_j^{(k)}} \sum_{k=1}^K \sum_{t=1}^T I(k) P(x_t = s_j | O, \Delta_r) - \mu_j \\
 &= \frac{1}{c_j^{(k)}} \sum_{k=1}^K \sum_{t=1}^T I(k) \frac{P(O, x_t = s_j | \Delta_r)}{P(O | \Delta_r)} - \mu_j
 \end{aligned}$$

(Due to multiplication rule specified by formula III.1.1.3)

$$\begin{aligned}
 &= \frac{1}{c_j^{(k)} P(O | \Delta_r)} \sum_{k=1}^K \sum_{t=1}^T I(k) P(O, x_t = s_j | \Delta_r) - \mu_j \\
 &= \frac{1}{c_j^{(k)}} \sum_{t=1}^T P^{(k)}(O, x_t = s_j | \Delta_r) - \mu_j
 \end{aligned}$$

(Where $P^{(k)}(O, x_t = s_j | \Delta_r)$ is the joint probability of state $x_t = s_j$ and observation sequence O counting for k^{th} PDF)

$$= \frac{1}{c_j^{(k)} P(O | \Delta_r)} \sum_{t=1}^T \gamma_t(j, k) - \mu_j$$

(Due to $\gamma_t(j, k) = P^{(k)}(O, x_t = s_j | \Delta_r)$ according to formula IV.5.8)

Setting the partial derivative $\frac{\partial l(\Delta, \lambda)}{\partial c_j^{(k)}}$ to be zero, we get the equation whose solution is estimate $\hat{c}_j^{(k)}$, as follows:

$$\begin{aligned} & \frac{1}{c_j^{(k)} P(O | \Delta_r)} \sum_{t=1}^T \gamma_t(j, k) - \mu_j = 0 \\ \Rightarrow & \sum_{t=1}^T \gamma_t(j, k) - \mu_j c_j^{(k)} P(O | \Delta_r) = 0 \end{aligned}$$

Summing the expression $\sum_{t=1}^T \gamma_t(j, k) - \mu_j c_j^{(k)} P(O | \Delta_r)$ over K mixture components, we have:

$$\begin{aligned} & \sum_{k=1}^K \sum_{t=1}^T \gamma_t(j, k) - \sum_{k=1}^K \mu_j c_j^{(k)} P(O | \Delta_r) = 0 \\ \Rightarrow & \sum_{k=1}^K \sum_{t=1}^T \gamma_t(j, k) - \mu_j P(O | \Delta_r) \sum_{k=1}^K c_j^{(k)} = 0 \\ \Rightarrow & \sum_{k=1}^K \sum_{t=1}^T \gamma_t(j, k) - \mu_j P(O | \Delta_r) = 0 \end{aligned}$$

(Due to constraint $\sum_{k=1}^K c_j^{(k)} = 1$)

$$\Rightarrow \mu_j P(O | \Delta_r) = \sum_{k=1}^K \sum_{t=1}^T \gamma_t(j, k)$$

Substituting $\mu_j P(O | \Delta_r) = \sum_{k=1}^K \sum_{t=1}^T \gamma_t(j, k)$ into equation $\sum_{t=1}^T \gamma_t(j, k) - \mu_j c_j^{(k)} P(O | \Delta_r) = 0$, we have:

$$\sum_{t=1}^T \gamma_t(j, k) - c_j^{(k)} \sum_{l=1}^K \sum_{t=1}^T \gamma_t(j, l) = 0$$

It implies that the weight estimate $\hat{c}_j^{(k)}$ is specified by formula IV.5.12:

$$\hat{c}_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(j, l)}$$

Formula IV.5.12. Weight estimate of partial PDF

The partial joint probabilities $\gamma_t(j, k)$ is determined by formula IV.5.8.

The parameter estimate $\hat{\theta}_j^{(k)}$ which is extreme point of the Lagrangian function $l(\Delta, \lambda, \mu)$ is determined by setting partial derivatives of $l(\Delta, \lambda, \mu)$ with respect to $\theta_j^{(k)}$ to be zero. The partial derivative of $l(\Delta, \lambda, \mu)$ with respect to $\theta_j^{(k)}$ is:

$$\begin{aligned}
 \frac{\partial l(\Delta, \lambda, \mu)}{\partial \theta_j^{(k)}} &= \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial \theta_j^{(k)}} + \frac{\partial}{\partial \theta_j^{(k)}} \left(\sum_{i=1}^n \lambda_i \left(1 - \sum_{j=1}^n a_{ij} \right) \right) \\
 &\quad + \frac{\partial}{\partial \theta_j^{(k)}} \left(\sum_{j=1}^n \mu_j \left(1 - \sum_{k=1}^K c_j^{(k)} \right) \right) \\
 &= \frac{\partial E_{X|O, \Delta_r} \{ \ln(P(O, X|\Delta)) \}}{\partial \theta_j^{(k)}} \\
 &= \frac{\partial}{\partial \theta_j^{(k)}} \left(\sum_X P(X|O, \Delta_r) \left(\sum_{i=1}^n \sum_{j=1}^n \sum_{t=1}^T I(x_{t-1} = s_i, x_t = s_j) \ln(a_{ij}) \right. \right. \\
 &\quad \left. \left. + \sum_{j=1}^n \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \ln(c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})) \right) \right) \\
 &= \sum_X P(X|O, \Delta_r) \sum_{j=1}^n \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \frac{\partial}{\partial \theta_j^{(k)}} \left(\ln(c_j^{(k)} p_j^{(k)}(o_t | \theta_j^{(k)})) \right) \\
 &= \sum_X P(X|O, \Delta_r) \sum_{j=1}^n \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \frac{\partial \left(\ln(c_j^{(k)}) + \ln(p_j^{(k)}(o_t | \theta_j^{(k)})) \right)}{\partial \theta_j^{(k)}} \\
 &= \sum_X P(X|O, \Delta_r) \sum_{k=1}^K \sum_{t=1}^T I(x_t = s_j, k) \frac{\partial \left(\ln(p_j^{(k)}(o_t | \theta_j^{(k)})) \right)}{\partial \theta_j^{(k)}} \\
 &= \sum_{k=1}^K \sum_{t=1}^T \sum_X I(x_t = s_j, k) P(X|O, \Delta_r) \frac{\partial \left(\ln(p_j^{(k)}(o_t | \theta_j^{(k)})) \right)}{\partial \theta_j^{(k)}} \\
 &= \sum_{k=1}^K \sum_{t=1}^T I(k) P(x_t = s_j | O, \Delta_r) \frac{\partial \left(\ln(p_j^{(k)}(o_t | \theta_j^{(k)})) \right)}{\partial \theta_j^{(k)}}
 \end{aligned}$$

(Due to total probability rule specified by formula III.1.1.4)

Where $I(x_t = s_j, k)$ is index function such that,

$$\begin{aligned}
 I(x_t = s_j, k) &= \begin{cases} 1 & \text{if } x_t = s_j \text{ counting for } k^{\text{th}} \text{ partial PDF} \\ 0 & \text{otherwise} \end{cases} \\
 I(k) &= \begin{cases} 1 & \text{if counting for } k^{\text{th}} \text{ partial PDF} \\ 0 & \text{otherwise} \end{cases}
 \end{aligned}$$

It implies

$$\frac{\partial l(\Delta, \lambda, \mu)}{\partial \theta_j^{(k)}} = \sum_{k=1}^K \sum_{t=1}^T I(k) \frac{P(O, x_t = s_j | \Delta_r)}{P(O | \Delta_r)} \frac{\partial \left(\ln(p_j^{(k)}(o_t | \theta_j^{(k)})) \right)}{\partial \theta_j^{(k)}}$$

(Due to multiplication rule specified by formula III.1.1.3)

$$= \frac{1}{P(O|\Delta_r)} \sum_{k=1}^K \sum_{t=1}^T I(k) P(O, x_t = s_j | \Delta_r) \frac{\partial \left(\ln \left(p_j^{(k)}(o_t | \theta_j^{(k)}) \right) \right)}{\partial \theta_j^{(k)}}$$

$$= \frac{1}{P(O|\Delta_r)} \sum_{t=1}^T P^{(k)}(O, x_t = s_j | \Delta_r) \frac{\partial \left(\ln \left(p_j^{(k)}(o_t | \theta_j^{(k)}) \right) \right)}{\partial \theta_j^{(k)}}$$

(Where $P^{(k)}(O, x_t = s_j | \Delta_r)$ is the joint probability of state $x_t = s_j$ and observation sequence O counting for k^{th} PDF)

$$= \frac{1}{P(O|\Delta_r)} \sum_{t=1}^T \gamma_t(j, k) \frac{\partial \left(\ln \left(p_j^{(k)}(o_t | \theta_j^{(k)}) \right) \right)}{\partial \theta_j^{(k)}}$$

(Due to $\gamma_t(j, k) = P^{(k)}(O, x_t = s_j | \Delta_r)$ according to formula IV.5.8)

Setting the partial derivative $\frac{\partial l(\Delta, \lambda)}{\partial \theta_j^{(k)}}$ to be zero, we get the equation whose solution is estimate $\hat{\theta}_j^{(k)}$, specified by formula IV.5.13.

$$\frac{1}{P(O|\Delta_r)} \sum_{t=1}^T \gamma_t(j, k) \frac{\partial \ln \left(p_j^{(k)}(o_t | \theta_j^{(k)}) \right)}{\partial \theta_j^{(k)}} = 0 \Leftrightarrow \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln \left(p_j^{(k)}(o_t | \theta_j^{(k)}) \right)}{\partial \theta_j^{(k)}} = 0$$

Formula IV.5.13. Equation of partial PDF parameter

The equation specified by formula is similar to the one specified by formula IV.5.5 except that the single PDF $p_j(o_t | \theta_j)$ is replaced by partial PDF $p_j^{(k)}(o_t | \theta_j^{(k)})$. It is possible to solve the above equation (formula IV.5.13) by Newton-Raphson method (Burden & Faires, 2011, pp. 67-69) – a numeric analysis method but it is easier and simpler to find out more precise solution if the PDF $p_j^{(k)}(o_t | \theta_j^{(k)})$ belongs to well-known distributions: normal distribution (Montgomery & Runger, 2003, pp. 109-110), exponential distribution (Montgomery & Runger, 2003, pp. 122-123), etc. Please see formula IV.5.5 for more details of such specific solutions.

Therefore, without replication, if $p_j^{(k)}(o_t | \theta_j^{(k)})$ is normal PDF, its parameter estimate $\hat{\theta}_j^{(k)} = (\hat{m}_j^{(k)}, \sigma_j^{(k)})$ where \hat{m}_j is mean estimate and $\hat{\sigma}_j^2$ is variance estimate is:

$$\hat{\theta}_j^{(k)} = \left(\hat{m}_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k) o_t}{\sum_{t=1}^T \gamma_t(j, k)}, \sigma_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k) (o_t - \hat{m}_j^{(k)})^2}{\sum_{t=1}^T \gamma_t(j, k)} \right)$$

If $p_j^{(k)}(o_t | \theta_j^{(k)})$ is exponential PDF, its parameter estimate $\hat{\theta}_j = \hat{k}_j$ is:

$$\hat{\theta}_j^{(k)} = \hat{k}_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{t=1}^T \gamma_t(j, k) o_t}$$

Shortly, the mixture continuous observation HMM estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{c}_j^{(k)}, \hat{\theta}_j^{(k)}, \hat{\pi}_j)$ with mixture PDF given current parameter $\Delta = (a_{ij}, c_j^{(k)}, \theta_j^{(k)}, \pi_j)$ is specified by formula IV.5.14.

$$\begin{aligned}\hat{a}_{ij} &= \frac{\sum_{t=2}^T \xi_t(i, j)}{\sum_{t=2}^T \sum_{j=1}^n \xi_t(i, j)} \\ \hat{\pi}_j &= \frac{\gamma_1(j)}{\sum_{i=1}^n \gamma_1(i)} \\ \hat{c}_j^{(k)} &= \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(j, l)} \\ \hat{\theta}_j &\text{ is the solution of } \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln(p_j^{(k)}(o_t | \theta_j^{(k)}))}{\partial \theta_j^{(k)}} = 0\end{aligned}$$

With normal distribution:

$$\hat{\theta}_j^{(k)} = \left(\hat{m}_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k) o_t}{\sum_{t=1}^T \gamma_t(j, k)}, \sigma_j^{2(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k) (o_t - \hat{m}_j^{(k)})^2}{\sum_{t=1}^T \gamma_t(j, k)} \right)$$

With exponential distribution:

$$\hat{\theta}_j^{(k)} = \hat{\kappa}_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{t=1}^T \gamma_t(j, k) o_t}$$

Formula IV.5.14. Continuous observation HMM parameter estimate with mixture PDF

The joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ follow formulas IV.5.2.

$$\xi_t(i, j) = \alpha_{t-1}(i) a_{ij} \left(\int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o | \theta_j) do \right) \beta_t(j) \text{ where } t \geq 2$$

$$\gamma_t(j) = P(O, x_t = s_j | \Delta) = \alpha_t(j) \beta_t(j)$$

The partial joint probabilities $\gamma_t(j, k)$ is determined by formula IV.5.8.

$$\gamma_t(j, k) = P^{(k)}(O, x_t = s_j | \Delta) = \alpha_t(j, k) \beta_t(j, k)$$

The EM algorithm applied into learning continuous observation HMM parameter with mixture PDF is described in table IV.5.7.

Starting with initial value for Δ , each iteration in EM algorithm has two steps:

1. *E-step:* Calculating the joint probabilities $\xi_t(i, j)$, $\gamma_t(j)$, and $\gamma_t(j, k)$ according to formulas IV.5.2 and IV.5.8.

$$\begin{aligned}\xi_t(i, j) &= \alpha_{t-1}(i) a_{ij} \left(\int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o | \theta_j) do \right) \beta_t(j) \text{ where } t \geq 2 \\ \gamma_t(j) &= \alpha_t(j) \beta_t(j) \\ \gamma_t(j, k) &= \alpha_t(j, k) \beta_t(j, k)\end{aligned}$$

2. *M-step:* Calculating the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{c}_j^{(k)}, \hat{\theta}_j^{(k)}, \hat{\pi}_j)$ based on the joint probabilities $\xi_t(i, j)$, $\gamma_t(j)$, and $\gamma_t(j, k)$ determined at E-step, according to formula IV.5.14. The estimate $\hat{\Delta}$ becomes the current parameter for next iteration.

$$\hat{a}_{ij} = \frac{\sum_{t=2}^T \xi_t(i, j)}{\sum_{t=2}^T \sum_{j=1}^n \xi_t(i, j)}$$

$$\hat{\pi}_j = \frac{\gamma_1(j)}{\sum_{i=1}^n \gamma_1(i)}$$

$$\hat{c}_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(j, l)}$$

$$\hat{\theta}_j \text{ is the solution of } \sum_{t=1}^T \gamma_t(j) \frac{\partial \ln(p_j^{(k)}(o_t | \theta_j^{(k)}))}{\partial \theta_j^{(k)}} = 0$$

With normal distribution:

$$\hat{\theta}_j^{(k)} = \left(\hat{m}_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k) o_t}{\sum_{t=1}^T \gamma_t(j, k)}, \sigma_j^{(k)} = \sqrt{\frac{\sum_{t=1}^T \gamma_t(j, k) (o_t - \hat{m}_j^{(k)})^2}{\sum_{t=1}^T \gamma_t(j, k)}} \right)$$

With exponential distribution:

$$\hat{\theta}_j^{(k)} = \hat{\kappa}_j^{(k)} = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{t=1}^T \gamma_t(j, k) o_t}$$

EM algorithm stops when it meets the terminating condition, for example, the difference of current parameter Δ and next parameter $\hat{\Delta}$ is insignificant. It is possible to define a custom terminating condition. The terminating criterion $P(O/\Delta)$ described in table IV.4.3.2.2 is a suggestion.

Table IV.5.7. EM algorithm applied into learning continuous observation HMM parameter with mixture PDF

Going back given HMM Δ whose parameters A and Π are specified in tables IV.4.1 and IV.4.2, suppose continuous observation sequence is $O = \{o_1=0.88, o_2=0.13, o_3=0.38\}$, the EM algorithm described in table IV.5.7 is applied into calculating the parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{\theta}_j, \hat{\pi}_j)$.

		Weather current day (Time point t)		
		sunny	cloudy	rainy
Weather previous day (Time point $t-1$)	sunny	$a_{11}=0.50$	$a_{12}=0.25$	$a_{13}=0.25$
	cloudy	$a_{21}=0.30$	$a_{22}=0.40$	$a_{23}=0.30$
	rainy	$a_{31}=0.25$	$a_{32}=0.25$	$a_{33}=0.50$

sunny	cloudy	rainy
$\pi_1=0.33$	$\pi_2=0.33$	$\pi_3=0.33$

Recall that observations are relative humidity measures. The bias for all measures is $\varepsilon=0.01$, for example, the first observation $o_1=0.88$ ranges in interval [0.88–0.01, 0.88+0.01]. Humidity is measured by $K=2$ equipments. The normalized weights of equipments 1 and 2 are 60% and 40%, respectively. Shortly, observation o_t is created from mixture of $K=2$ equipments. The observation probability distribution B is now includes three normal mixture PDF (s): $p_1(o_t | \theta_1)$, $p_2(o_t | \theta_2)$, and $p_3(o_t | \theta_3)$ corresponding to three states: $s_1=\text{sunny}$, $s_2=\text{cloudy}$, and $s_3=\text{rainy}$. Following is specification of these normal PDF (s).

$$p_1(o_t | \theta_1) = \sum_{k=1}^2 c_1^{(k)} p_1^{(k)}(o_t | \theta_1^{(k)})$$

$$p_2(o_t|\theta_2) = \sum_{k=1}^2 c_2^{(k)} p_2^{(k)}(o_t|\theta_2^{(k)})$$

$$p_3(o_t|\theta_3) = \sum_{k=1}^2 c_3^{(k)} p_3^{(k)}(o_t|\theta_3^{(k)})$$

Note that $c_1^{(k)}$, $c_2^{(k)}$, and $c_3^{(k)}$ are normalized weights; $p_1^{(k)}(o|\theta_1^{(k)})$, $p_2^{(k)}(o|\theta_2^{(k)})$, and $p_3^{(k)}(o|\theta_3^{(k)})$ are normal partial PDF (s).

$$p_1^{(k)}(o_t|\theta_1^{(k)}) = \frac{1}{\sqrt{2\pi\sigma_1^{2(k)}}} \exp\left(-\frac{1}{2}\frac{(o_t - m_1^{(k)})^2}{\sigma_1^{2(k)}}\right)$$

$$p_2^{(k)}(o_t|\theta_2^{(k)}) = \frac{1}{\sqrt{2\pi\sigma_2^{2(k)}}} \exp\left(-\frac{1}{2}\frac{(o_t - m_2^{(k)})^2}{\sigma_2^{2(k)}}\right)$$

$$p_3^{(k)}(o_t|\theta_3^{(k)}) = \frac{1}{\sqrt{2\pi\sigma_3^{2(k)}}} \exp\left(-\frac{1}{2}\frac{(o_t - m_3^{(k)})^2}{\sigma_3^{2(k)}}\right)$$

Where $\theta_1^{(k)} = (m_1^{(k)}, \sigma_1^{2(k)})$, $\theta_2^{(k)} = (m_2^{(k)}, \sigma_2^{2(k)})$, and $\theta_3^{(k)} = (m_3^{(k)}, \sigma_3^{2(k)})$ are means and variances of $p_1^{(k)}(o_t|\theta_1^{(k)})$, $p_2^{(k)}(o_t|\theta_2^{(k)})$, and $p_3^{(k)}(o_t|\theta_3^{(k)})$.

As a convention, normal mixture PDF (s) such as $p_1(o_t|\theta_1)$, $p_2(o_t|\theta_2)$, and $p_3(o_t|\theta_3)$ are represented by theirs weights, partial means and partial variances.

$$\theta_1 = (c_1^{(1)}, m_1^{(1)}, \sigma_1^{2(1)}, c_1^{(2)}, m_1^{(2)}, \sigma_1^{2(2)})$$

$$\theta_2 = (c_2^{(1)}, m_2^{(1)}, \sigma_2^{2(1)}, c_2^{(2)}, m_2^{(2)}, \sigma_2^{2(2)})$$

$$\theta_3 = (c_3^{(1)}, m_3^{(1)}, \sigma_3^{2(1)}, c_3^{(2)}, m_3^{(2)}, \sigma_3^{2(2)})$$

These weights, means and variances are also called observation probability parameters that substitute for discrete matrix B . Table IV.5.8 shows observation probability parameters for our weather example.

$p_1(o_t \theta_1)$	$c_1^{(1)} = 0.6$	$m_1^{(1)} = 0.87$	$\sigma_1^{2(1)} = 1$	$c_1^{(2)} = 0.4$	$m_1^{(2)} = 0.15$	$\sigma_1^{2(2)} = 1$
$p_2(o_t \theta_2)$	$c_2^{(1)} = 0.6$	$m_2^{(1)} = 0.39$	$\sigma_2^{2(1)} = 1$	$c_2^{(2)} = 0.4$	$m_2^{(2)} = 0.89$	$\sigma_2^{2(2)} = 1$
$p_3(o_t \theta_3)$	$c_3^{(1)} = 0.6$	$m_3^{(1)} = 0.14$	$\sigma_3^{2(1)} = 1$	$c_3^{(2)} = 0.4$	$m_3^{(2)} = 0.37$	$\sigma_3^{2(2)} = 1$

Table IV.5.8. Observation probability parameters for weather example in case of mixture model

Obviously, we have:

$$p_1(o_t|\theta_1) = \frac{0.6}{\sqrt{2\pi 1}} \exp\left(-\frac{1}{2}\frac{(o_t - 0.87)^2}{1}\right) + \frac{0.4}{\sqrt{2\pi 1}} \exp\left(-\frac{1}{2}\frac{(o_t - 0.15)^2}{1}\right)$$

$$p_2(o_t|\theta_2) = \frac{0.6}{\sqrt{2\pi 1}} \exp\left(-\frac{1}{2}\frac{(o_t - 0.39)^2}{1}\right) + \frac{0.4}{\sqrt{2\pi 1}} \exp\left(-\frac{1}{2}\frac{(o_t - 0.89)^2}{1}\right)$$

$$p_3(o_t|\theta_3) = \frac{0.6}{\sqrt{2\pi 1}} \exp\left(-\frac{1}{2}\frac{(o_t - 0.14)^2}{1}\right) + \frac{0.4}{\sqrt{2\pi 1}} \exp\left(-\frac{1}{2}\frac{(o_t - 0.37)^2}{1}\right)$$

EM algorithm described in table IV.5.7 is applied into calculating the parameter estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{\theta}_j, \hat{\pi}_j)$ given continuous observation sequence $O = \{o_1=0.88, o_2=0.13, o_3=0.38\}$ and mixture normal PDF (s) whose means and variances shown in table IV.5.8. For convenience, all floating-point values are rounded off until ten decimal numbers.

As a convention, let

$$\begin{aligned} b_j(o_t, k) &= \int_{o_t-\varepsilon}^{o_t+\varepsilon} c_j^{(k)} p_j^{(k)}(o|\theta_j^{(k)}) do = c_j^{(k)} \int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j^{(k)}(o|\theta_j^{(k)}) do \\ &= c_j^{(k)} \Phi \left(\frac{o_t + \varepsilon - m_j^{(k)}}{\sqrt{\sigma_j^{(k)}}} \right) - c_j^{(k)} \Phi \left(\frac{o_t - \varepsilon - m_j^{(k)}}{\sqrt{\sigma_j^{(k)}}} \right) \\ b_j(o_t) &= \int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j(o|\theta_j) do = \sum_{k=1}^2 \int_{o_t-\varepsilon}^{o_t+\varepsilon} c_j^{(k)} p_j^{(k)}(o|\theta_j^{(k)}) do \\ &= \sum_{k=1}^2 c_j^{(k)} \int_{o_t-\varepsilon}^{o_t+\varepsilon} p_j^{(k)}(o|\theta_j^{(k)}) do = b_j(o_t, 1) + b_j(o_t, 2) \end{aligned}$$

At the first iteration ($r=1$) we have:

$$b_1(o_1, 1) = b_1(0.88, 1)$$

$$\begin{aligned} &= c_1^{(1)} \Phi \left(\frac{0.88 + 0.01 - m_1^{(1)}}{\sqrt{\sigma_1^{(1)}}} \right) - c_1^{(1)} \Phi \left(\frac{0.88 - 0.01 - m_1^{(1)}}{\sqrt{\sigma_1^{(1)}}} \right) \\ &= 0.004787 \end{aligned}$$

$$b_1(o_1, 2) = b_1(0.88, 2)$$

$$\begin{aligned} &= c_1^{(2)} \Phi \left(\frac{0.88 + 0.01 - m_1^{(2)}}{\sqrt{\sigma_1^{(2)}}} \right) - c_1^{(2)} \Phi \left(\frac{0.88 - 0.01 - m_1^{(2)}}{\sqrt{\sigma_1^{(2)}}} \right) \\ &= 0.002445 \end{aligned}$$

$$b_1(o_1) = b_1(0.88) = b_1(0.88, 1) + b_1(0.88, 2) = 0.007232$$

$$b_1(o_2, 1) = b_1(0.13, 1)$$

$$\begin{aligned} &= c_1^{(1)} \Phi \left(\frac{0.13 + 0.01 - m_1^{(1)}}{\sqrt{\sigma_1^{(1)}}} \right) - c_1^{(1)} \Phi \left(\frac{0.13 - 0.01 - m_1^{(1)}}{\sqrt{\sigma_1^{(1)}}} \right) \\ &= 0.003641 \end{aligned}$$

$$b_1(o_2, 2) = b_1(0.13, 2)$$

$$\begin{aligned} &= c_1^{(2)} \Phi \left(\frac{0.13 + 0.01 - m_1^{(2)}}{\sqrt{\sigma_1^{(2)}}} \right) - c_1^{(2)} \Phi \left(\frac{0.13 - 0.01 - m_1^{(2)}}{\sqrt{\sigma_1^{(2)}}} \right) \\ &= 0.003191 \end{aligned}$$

$$b_1(o_2) = b_1(0.13) = b_1(0.13, 1) + b_1(0.13, 2) = 0.006831$$

$$b_1(o_3, 1) = b_1(0.38, 1)$$

$$= c_1^{(1)} \Phi \left(\frac{0.38 + 0.01 - m_1^{(1)}}{\sqrt{\sigma_1^{(1)}}} \right) - c_1^{(1)} \Phi \left(\frac{0.38 - 0.01 - m_1^{(1)}}{\sqrt{\sigma_1^{(1)}}} \right)$$

$$= 0.004246$$

$$b_1(o_3, 2) = b_1(0.38, 2)$$

$$= c_1^{(2)} \Phi \left(\frac{0.38 + 0.01 - m_1^{(2)}}{\sqrt{\sigma_1^{(2)}}} \right) - c_1^{(2)} \Phi \left(\frac{0.38 - 0.01 - m_1^{(2)}}{\sqrt{\sigma_1^{(2)}}} \right)$$

$$= 0.003108$$

$$b_1(o_3) = b_1(0.38) = b_1(0.38, 1) + b_1(0.38, 2) = 0.007354$$

$$b_2(o_1, 1) = b_2(0.88, 1)$$

$$= c_2^{(1)} \Phi \left(\frac{0.88 + 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right) - c_2^{(1)} \Phi \left(\frac{0.88 - 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right)$$

$$= 0.003538$$

$$b_2(o_1, 2) = b_2(0.88, 2)$$

$$= c_2^{(2)} \Phi \left(\frac{0.88 + 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right) - c_2^{(2)} \Phi \left(\frac{0.88 - 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right)$$

$$= 0.003989$$

$$b_2(o_1) = b_2(0.88) = b_2(0.88, 1) + b_2(0.88, 2) = 0.007527$$

$$b_2(o_2, 1) = b_2(0.13, 1)$$

$$= c_2^{(1)} \Phi \left(\frac{0.13 + 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right) - c_2^{(1)} \Phi \left(\frac{0.13 - 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right)$$

$$= 0.003857$$

$$b_2(o_2, 2) = b_2(0.13, 2)$$

$$= c_2^{(2)} \Phi \left(\frac{0.13 + 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right) - c_2^{(2)} \Phi \left(\frac{0.13 - 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right)$$

$$= 0.002989$$

$$b_2(o_2) = b_2(0.13) = b_2(0.13, 1) + b_2(0.13, 2) = 0.006845$$

$$b_2(o_3, 1) = b_2(0.38, 1)$$

$$= c_2^{(1)} \Phi \left(\frac{0.38 + 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right) - c_2^{(1)} \Phi \left(\frac{0.38 - 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right)$$

$$= 0.003989$$

$$b_2(o_3, 2) = b_2(0.38, 2)$$

$$= c_2^{(2)} \Phi \left(\frac{0.38 + 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right) - c_2^{(2)} \Phi \left(\frac{0.38 - 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right)$$

$$= 0.003503$$

$$b_2(o_3) = b_2(0.38) = b_2(0.38, 1) + b_2(0.38, 2) = 0.007492$$

$$b_3(o_1, 1) = b_3(0.88, 1)$$

$$= c_3^{(1)} \Phi \left(\frac{0.88 + 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right) - c_3^{(1)} \Phi \left(\frac{0.88 - 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right)$$

$$= 0.002427$$

$$b_3(o_1, 2) = b_3(0.88, 2)$$

$$= c_3^{(2)} \Phi \left(\frac{0.88 + 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right) - c_3^{(2)} \Phi \left(\frac{0.88 - 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right)$$

$$= 0.004203$$

$$b_3(o_1) = b_3(0.88) = b_3(0.88, 1) + b_3(0.88, 2) = 0.006631$$

$$b_3(o_2, 1) = b_3(0.13, 1)$$

$$= c_3^{(1)} \Phi \left(\frac{0.13 + 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right) - c_3^{(1)} \Phi \left(\frac{0.13 - 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right)$$

$$= 0.003191$$

$$b_3(o_2, 2) = b_3(0.13, 2)$$

$$= c_3^{(2)} \Phi \left(\frac{0.13 + 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right) - c_3^{(2)} \Phi \left(\frac{0.13 - 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right)$$

$$= 0.004651$$

$$b_3(o_2) = b_3(0.13) = b_3(0.13, 1) + b_3(0.13, 2) = 0.007843$$

$$b_3(o_3, 1) = b_3(0.38, 1)$$

$$= c_3^{(1)} \Phi \left(\frac{0.38 + 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right) - c_3^{(1)} \Phi \left(\frac{0.38 - 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right)$$

$$= 0.003101$$

$$b_3(o_3, 2) = b_3(0.38, 2)$$

$$= c_3^{(2)} \Phi \left(\frac{0.38 + 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right) - c_3^{(2)} \Phi \left(\frac{0.38 - 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right)$$

$$= 0.004787$$

$$b_3(o_3) = b_3(0.38) = b_3(0.38, 1) + b_3(0.38, 2) = 0.007888$$

$$\alpha_1(1,1) = b_1(o_1, 1)\pi_1 = 0.00158$$

$$\begin{aligned}
 \alpha_1(1,2) &= b_1(o_1, 2)\pi_1 = 0.000807 \\
 \alpha_1(1) &= b_1(o_1)\pi_1 = 0.002387 \\
 \alpha_1(2,1) &= b_2(o_1, 1)\pi_2 = 0.001168 \\
 \alpha_1(2,2) &= b_2(o_1, 2)\pi_2 = 0.001316 \\
 \alpha_1(2) &= b_2(o_1)\pi_2 = 0.002484 \\
 \alpha_1(3,1) &= b_3(o_1, 1)\pi_3 = 0.000801 \\
 \alpha_1(3,2) &= b_3(o_1, 2)\pi_3 = 0.001387 \\
 \alpha_1(3) &= b_3(o_1)\pi_3 = 0.002188 \\
 \alpha_2(1,1) &= \left(\sum_{i=1}^3 \alpha_1(i, 1)a_{i1} \right) b_1(o_2, 1) = 0.000005 \\
 \alpha_2(1,2) &= \left(\sum_{i=1}^3 \alpha_1(i, 2)a_{i1} \right) b_1(o_2, 2) = 0.000004 \\
 \alpha_2(1) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_1(o_2) = 0.000017 \\
 \alpha_2(2,1) &= \left(\sum_{i=1}^3 \alpha_1(i, 1)a_{i2} \right) b_2(o_2, 1) = 0.000004 \\
 \alpha_2(2,2) &= \left(\sum_{i=1}^3 \alpha_1(i, 2)a_{i2} \right) b_2(o_2, 2) = 0.000003 \\
 \alpha_2(2) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_2(o_2) = 0.000015 \\
 \alpha_2(3,1) &= \left(\sum_{i=1}^3 \alpha_1(i, 1)a_{i3} \right) b_3(o_2, 1) = 0.000004 \\
 \alpha_2(3,2) &= \left(\sum_{i=1}^3 \alpha_1(i, 2)a_{i3} \right) b_3(o_2, 2) = 0.000006 \\
 \alpha_2(3) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_3(o_2) = 0.000019 \\
 \alpha_3(1,1) &= \left(\sum_{i=1}^3 \alpha_2(i, 1)a_{i1} \right) b_1(o_3, 1) = 0.00000002 \\
 \alpha_3(1,2) &= \left(\sum_{i=1}^3 \alpha_2(i, 2)a_{i1} \right) b_1(o_3, 2) = 0.00000001 \\
 \alpha_3(1) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_1(o_3) = 0.00000013 \\
 \alpha_3(2,1) &= \left(\sum_{i=1}^3 \alpha_2(i, 1)a_{i2} \right) b_2(o_3, 1) = 0.00000002 \\
 \alpha_3(2,2) &= \left(\sum_{i=1}^3 \alpha_2(i, 2)a_{i2} \right) b_2(o_3, 2) = 0.00000001
 \end{aligned}$$

$$\alpha_3(2) = \left(\sum_{i=1}^3 \alpha_2(i) a_{i2} \right) b_2(o_3) = 0.00000011$$

$$\alpha_3(3,1) = \left(\sum_{i=1}^3 \alpha_2(i,1) a_{i3} \right) b_3(o_3,1) = 0.00000001$$

$$\alpha_3(3,2) = \left(\sum_{i=1}^3 \alpha_2(i,2) a_{i3} \right) b_3(o_3,2) = 0.00000002$$

$$\alpha_3(3) = \left(\sum_{i=1}^3 \alpha_2(i) a_{i3} \right) b_3(o_3) = 0.00000014$$

$$\beta_3(1,1) = \beta_3(2,1) = \beta_3(3,1) = 1$$

$$\beta_3(1,2) = \beta_3(2,2) = \beta_3(3,2) = 1$$

$$\beta_3(1) = \beta_3(2) = \beta_3(3) = 1$$

$$\beta_2(1,1) = \sum_{j=1}^n a_{1j} b_j(o_3, 1) \beta_3(j, 1) = 0.003895$$

$$\beta_2(1,2) = \sum_{j=1}^n a_{1j} b_j(o_3, 2) \beta_3(j, 2) = 0.003627$$

$$\beta_2(1) = \sum_{j=1}^n a_{1j} b_j(o_3) \beta_3(j) = 0.007522$$

$$\beta_2(2,1) = \sum_{j=1}^n a_{2j} b_j(o_3, 1) \beta_3(j, 1) = 0.0038$$

$$\beta_2(2,2) = \sum_{j=1}^n a_{2j} b_j(o_3, 2) \beta_3(j, 2) = 0.00377$$

$$\beta_2(2) = \sum_{j=1}^n a_{2j} b_j(o_3) \beta_3(j) = 0.007569$$

$$\beta_2(3,1) = \sum_{j=1}^n a_{3j} b_j(o_3, 1) \beta_3(j, 1) = 0.003609$$

$$\beta_2(3,2) = \sum_{j=1}^n a_{3j} b_j(o_3, 2) \beta_3(j, 2) = 0.004046$$

$$\beta_2(3) = \sum_{j=1}^n a_{3j} b_j(o_3) \beta_3(j) = 0.007655$$

$$\beta_1(1,1) = \sum_{j=1}^n a_{1j} b_j(o_2, 1) \beta_2(j, 1) = 0.000014$$

$$\beta_1(1,2) = \sum_{j=1}^n a_{1j} b_j(o_2, 2) \beta_2(j, 2) = 0.000013$$

$$\beta_1(1) = \sum_{j=1}^n a_{1j} b_j(o_2) \beta_2(j) = 0.000054$$

$$\begin{aligned}\beta_1(2,1) &= \sum_{j=1}^n a_{2j} b_j(o_2, 1) \beta_2(j, 1) = 0.000014 \\ \beta_1(2,2) &= \sum_{j=1}^n a_{2j} b_j(o_2, 2) \beta_2(j, 2) = 0.000014 \\ \beta_1(2) &= \sum_{j=1}^n a_{2j} b_j(o_2) \beta_2(j) = 0.000054 \\ \beta_1(3,1) &= \sum_{j=1}^n a_{3j} b_j(o_2, 1) \beta_2(j, 1) = 0.000013 \\ \beta_1(3,2) &= \sum_{j=1}^n a_{3j} b_j(o_2, 2) \beta_2(j, 2) = 0.000015 \\ \beta_1(3) &= \sum_{j=1}^n a_{3j} b_j(o_2) \beta_2(j) = 0.000056\end{aligned}$$

Within the E-step of the first iteration ($r=1$), the terminating criterion $P(O|\Delta)$ is calculated according to forward-backward procedure (see table IV.4.1.1) as follows:

$$P(O|\Delta) = \alpha_3(1) + \alpha_3(2) + \alpha_3(3) = 0.00000038$$

Within the E-step of the first iteration ($r=1$), the joint probabilities $\xi_t(i,j)$, $\gamma_t(j)$, and $\gamma_t(j,k)$ are calculated based on formulas IV.5.2 and IV.5.8 as follows:

$$\begin{aligned}\xi_2(1,1) &= \alpha_1(1)a_{11}b_1(o_2)\beta_2(1) = 0.00000006 \\ \xi_2(1,2) &= \alpha_1(1)a_{12}b_2(o_2)\beta_2(2) = 0.00000003 \\ \xi_2(1,3) &= \alpha_1(1)a_{13}b_3(o_2)\beta_2(3) = 0.00000004 \\ \xi_2(2,1) &= \alpha_1(2)a_{21}b_1(o_2)\beta_2(1) = 0.00000004 \\ \xi_2(2,2) &= \alpha_1(2)a_{22}b_2(o_2)\beta_2(2) = 0.00000005 \\ \xi_2(2,3) &= \alpha_1(2)a_{23}b_3(o_2)\beta_2(3) = 0.00000004 \\ \xi_2(3,1) &= \alpha_1(3)a_{31}b_1(o_2)\beta_2(1) = 0.00000003 \\ \xi_2(3,2) &= \alpha_1(3)a_{32}b_2(o_2)\beta_2(2) = 0.00000003 \\ \xi_2(3,3) &= \alpha_1(3)a_{33}b_3(o_2)\beta_2(3) = 0.00000007 \\ \xi_3(1,1) &= \alpha_2(1)a_{11}b_1(o_3)\beta_3(1) = 0.00000006 \\ \xi_3(1,2) &= \alpha_2(1)a_{12}b_2(o_3)\beta_3(2) = 0.00000003 \\ \xi_3(1,3) &= \alpha_2(1)a_{13}b_3(o_3)\beta_3(3) = 0.00000003 \\ \xi_3(2,1) &= \alpha_2(2)a_{21}b_1(o_3)\beta_3(1) = 0.00000003 \\ \xi_3(2,2) &= \alpha_2(2)a_{22}b_2(o_3)\beta_3(2) = 0.00000004 \\ \xi_3(2,3) &= \alpha_2(2)a_{23}b_3(o_3)\beta_3(3) = 0.00000003 \\ \xi_3(3,1) &= \alpha_2(3)a_{31}b_1(o_3)\beta_3(1) = 0.00000004 \\ \xi_3(3,2) &= \alpha_2(3)a_{32}b_2(o_3)\beta_3(2) = 0.00000004 \\ \xi_3(3,3) &= \alpha_2(3)a_{33}b_3(o_3)\beta_3(3) = 0.00000008\end{aligned}$$

$$\begin{aligned}\gamma_1(1,1) &= \alpha_1(1,1)\beta_1(1,1) = 0.00000002 \\ \gamma_1(1,2) &= \alpha_1(1,2)\beta_1(1,2) = 0.00000001 \\ \gamma_1(1) &= \alpha_1(1)\beta_1(1) = 0.00000013 \\ \gamma_1(2,1) &= \alpha_1(2,1)\beta_1(2,1) = 0.00000002 \\ \gamma_1(2,2) &= \alpha_1(2,2)\beta_1(2,2) = 0.00000002 \\ \gamma_1(2) &= \alpha_1(2)\beta_1(2) = 0.00000013 \\ \gamma_1(3,1) &= \alpha_1(3,1)\beta_1(3,1) = 0.00000001 \\ \gamma_1(3,2) &= \alpha_1(3,2)\beta_1(3,2) = 0.00000002\end{aligned}$$

$$\begin{aligned}
 \gamma_1(3) &= \alpha_1(3)\beta_1(3) = 0.00000012 \\
 \gamma_2(1,1) &= \alpha_2(1,1)\beta_2(1,1) = 0.00000002 \\
 \gamma_2(1,2) &= \alpha_2(1,2)\beta_2(1,2) = 0.00000001 \\
 \gamma_2(1) &= \alpha_2(1)\beta_2(1) = 0.00000013 \\
 \gamma_2(2,1) &= \alpha_2(2,1)\beta_2(2,1) = 0.00000002 \\
 \gamma_2(2,2) &= \alpha_2(2,2)\beta_2(2,2) = 0.00000001 \\
 \gamma_2(2) &= \alpha_2(2)\beta_2(2) = 0.00000011 \\
 \gamma_2(3,1) &= \alpha_2(3,1)\beta_2(3,1) = 0.00000001 \\
 \gamma_2(3,2) &= \alpha_2(3,2)\beta_2(3,2) = 0.00000002 \\
 \gamma_2(3) &= \alpha_2(3)\beta_2(3) = 0.00000015 \\
 \gamma_3(1,1) &= \alpha_3(1,1)\beta_3(1,1) = 0.00000002 \\
 \gamma_3(1,2) &= \alpha_3(1,2)\beta_3(1,2) = 0.00000001 \\
 \gamma_3(1) &= \alpha_3(1)\beta_3(1) = 0.00000013 \\
 \gamma_3(2,1) &= \alpha_3(2,1)\beta_3(2,1) = 0.00000002 \\
 \gamma_3(2,2) &= \alpha_3(2,2)\beta_3(2,2) = 0.00000001 \\
 \gamma_3(2) &= \alpha_3(2)\beta_3(2) = 0.00000011 \\
 \gamma_3(3,1) &= \alpha_3(3,1)\beta_3(3,1) = 0.00000001 \\
 \gamma_3(3,2) &= \alpha_3(3,2)\beta_3(3,2) = 0.00000002 \\
 \gamma_3(3) &= \alpha_3(3)\beta_3(3) = 0.00000014
 \end{aligned}$$

Within the M-step of the first iteration ($r=1$), the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is calculated based on joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ determined at E-step.

$$\begin{aligned}
 \hat{a}_{11} &= \frac{\sum_{t=2}^3 \xi_t(1,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.483829 \\
 \hat{a}_{12} &= \frac{\sum_{t=2}^3 \xi_t(1,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.245210 \\
 \hat{a}_{13} &= \frac{\sum_{t=2}^3 \xi_t(1,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.270961 \\
 \hat{a}_{21} &= \frac{\sum_{t=2}^3 \xi_t(2,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.287734 \\
 \hat{a}_{22} &= \frac{\sum_{t=2}^3 \xi_t(2,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.388684 \\
 \hat{a}_{23} &= \frac{\sum_{t=2}^3 \xi_t(2,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.323582 \\
 \hat{a}_{31} &= \frac{\sum_{t=2}^3 \xi_t(3,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.235597 \\
 \hat{a}_{32} &= \frac{\sum_{t=2}^3 \xi_t(3,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.238932 \\
 \hat{a}_{33} &= \frac{\sum_{t=2}^3 \xi_t(3,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.525471
 \end{aligned}$$

$$\begin{aligned}
 \hat{c}_1^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(1,1)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(1,l)} = 0.614137 \\
 \hat{m}_1^{(1)} &= \frac{\sum_{t=1}^3 \gamma_t(1,1) o_t}{\sum_{t=1}^3 \gamma_t(1,1)} = 0.484616
 \end{aligned}$$

$$\begin{aligned}
 \hat{\sigma}^2_1^{(1)} &= \frac{\sum_{t=1}^3 \gamma_t(1,1) \left(o_t - \hat{m}_1^{(1)} \right)^2}{\sum_{t=1}^3 \gamma_t(1,1)} = 0.100338 \\
 \hat{c}_1^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(1,2)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(1,l)} = 0.385863 \\
 \hat{m}_1^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(1,2) o_t}{\sum_{t=1}^T \gamma_t(1,2)} = 0.442128 \\
 \sigma^2_1^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(1,2) \left(o_t - \hat{m}_1^{(2)} \right)^2}{\sum_{t=1}^T \gamma_t(1,2)} = 0.093132 \\
 \hat{c}_2^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(2,1)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(2,l)} = 0.582310 \\
 \hat{m}_2^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(2,1) o_t}{\sum_{t=1}^T \gamma_t(2,1)} = 0.524775 \\
 \sigma^2_2^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(2,1) \left(o_t - \hat{m}_2^{(1)} \right)^2}{\sum_{t=1}^T \gamma_t(2,1)} = 0.109779 \\
 \hat{c}_2^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(2,2)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(2,l)} = 0.417690 \\
 \hat{m}_2^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(2,2) o_t}{\sum_{t=1}^T \gamma_t(2,2)} = 0.566734 \\
 \sigma^2_2^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(2,2) \left(o_t - \hat{m}_2^{(2)} \right)^2}{\sum_{t=1}^T \gamma_t(2,2)} = 0.108245 \\
 \hat{c}_3^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(3,1)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(3,l)} = 0.465338 \\
 \hat{m}_3^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(3,1) o_t}{\sum_{t=1}^T \gamma_t(3,1)} = 0.438194 \\
 \sigma^2_3^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(3,1) \left(o_t - \hat{m}_3^{(1)} \right)^2}{\sum_{t=1}^T \gamma_t(3,1)} = 0.099469 \\
 \hat{c}_3^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(3,2)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(3,l)} = 0.534662 \\
 \hat{m}_3^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(3,2) o_t}{\sum_{t=1}^T \gamma_t(3,2)} = 0.448495 \\
 \sigma^2_3^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(3,2) \left(o_t - \hat{m}_3^{(2)} \right)^2}{\sum_{t=1}^T \gamma_t(3,2)} = 0.101523 \\
 \hat{\pi}_1 &= \frac{\gamma_1(1)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.332860 \\
 \hat{\pi}_2 &= \frac{\gamma_1(2)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.349659 \\
 \hat{\pi}_3 &= \frac{\gamma_1(3)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.317481
 \end{aligned}$$

At the second iteration ($r=2$), the current parameter $\Delta = (a_{ij}, \theta_j, \pi_j)$ is received values from the previous estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{\theta}_j, \hat{\pi}_j)$, as seen in table IV.5.9.

$a_{11} = \hat{a}_{11} = 0.483829$	$a_{12} = \hat{a}_{12} = 0.245210$	$a_{13} = \hat{a}_{13} = 0.270961$
$a_{21} = \hat{a}_{21} = 0.287734$	$a_{22} = \hat{a}_{22} = 0.388684$	$a_{23} = \hat{a}_{23} = 0.323582$
$a_{31} = \hat{a}_{31} = 0.235597$	$a_{32} = \hat{a}_{32} = 0.238932$	$a_{33} = \hat{a}_{33} = 0.525471$
$c_1^{(1)} = \hat{c}_1^{(1)} = 0.614137$	$m_1^{(1)} = \hat{m}_1^{(1)} = 0.484616$	$\sigma^2_1^{(1)} = \hat{\sigma}^2_1^{(1)} = 0.100338$
$c_1^{(2)} = \hat{c}_1^{(2)} = 0.385863$	$m_1^{(2)} = \hat{m}_1^{(2)} = 0.442128$	$\sigma^2_1^{(2)} = \hat{\sigma}^2_1^{(2)} = 0.093132$
$c_2^{(1)} = \hat{c}_2^{(1)} = 0.582310$	$m_2^{(1)} = \hat{m}_2^{(1)} = 0.524775$	$\sigma^2_2^{(1)} = \hat{\sigma}^2_2^{(1)} = 0.109779$
$c_2^{(2)} = \hat{c}_2^{(2)} = 0.417690$	$m_2^{(2)} = \hat{m}_2^{(2)} = 0.566734$	$\sigma^2_2^{(2)} = \hat{\sigma}^2_2^{(2)} = 0.108245$
$c_3^{(1)} = \hat{c}_3^{(1)} = 0.465338$	$m_3^{(1)} = \hat{m}_3^{(1)} = 0.438194$	$\sigma^2_3^{(1)} = \hat{\sigma}^2_3^{(1)} = 0.099469$
$c_3^{(2)} = \hat{c}_3^{(2)} = 0.534662$	$m_3^{(2)} = \hat{m}_3^{(2)} = 0.448495$	$\sigma^2_3^{(2)} = \hat{\sigma}^2_3^{(2)} = 0.101523$
$\pi_1 = \hat{\pi}_1 = 0.332860$	$\pi_2 = \hat{\pi}_2 = 0.349659$	$\pi_3 = \hat{\pi}_3 = 0.317481$
Terminating criterion $P(O/\Delta) = 0.00000038$		

Table IV.5.9. Mixture HMM parameters resulted from the first iteration of EM algorithm

We have:

$$b_1(o_1, 1) = b_1(0.88, 1)$$

$$= c_1^{(1)} \Phi \left(\frac{0.88 + 0.01 - m_1^{(1)}}{\sqrt{\sigma^2_1^{(1)}}} \right) - c_1^{(1)} \Phi \left(\frac{0.88 - 0.01 - m_1^{(1)}}{\sqrt{\sigma^2_1^{(1)}}} \right)$$

$$= 0.007099$$

$$b_1(o_1, 2) = b_1(0.88, 2)$$

$$= c_1^{(2)} \Phi \left(\frac{0.88 + 0.01 - m_1^{(2)}}{\sqrt{\sigma^2_1^{(2)}}} \right) - c_1^{(2)} \Phi \left(\frac{0.88 - 0.01 - m_1^{(2)}}{\sqrt{\sigma^2_1^{(2)}}} \right)$$

$$= 0.003605$$

$$b_1(o_1) = b_1(0.88) = b_1(0.88, 1) + b_1(0.88, 2) = 0.010704$$

$$b_1(o_2, 1) = b_1(0.13, 1)$$

$$= c_1^{(1)} \Phi \left(\frac{0.13 + 0.01 - m_1^{(1)}}{\sqrt{\sigma^2_1^{(1)}}} \right) - c_1^{(1)} \Phi \left(\frac{0.13 - 0.01 - m_1^{(1)}}{\sqrt{\sigma^2_1^{(1)}}} \right)$$

$$= 0.008267$$

$$b_1(o_2, 2) = b_1(0.13, 2)$$

$$= c_1^{(2)} \Phi \left(\frac{0.13 + 0.01 - m_1^{(2)}}{\sqrt{\sigma^2_1^{(2)}}} \right) - c_1^{(2)} \Phi \left(\frac{0.13 - 0.01 - m_1^{(2)}}{\sqrt{\sigma^2_1^{(2)}}} \right)$$

$$= 0.00598$$

$$b_1(o_2) = b_1(0.13) = b_1(0.13,1) + b_1(0.13,2) = 0.014246$$

$$b_1(o_3, 1) = b_1(0.38,1)$$

$$= c_1^{(1)} \Phi \left(\frac{0.38 + 0.01 - m_1^{(1)}}{\sqrt{\sigma_1^{(1)}}} \right) - c_1^{(1)} \Phi \left(\frac{0.38 - 0.01 - m_1^{(1)}}{\sqrt{\sigma_1^{(1)}}} \right)$$

$$= 0.014646$$

$$b_1(o_3, 2) = b_1(0.38,2)$$

$$= c_1^{(2)} \Phi \left(\frac{0.38 + 0.01 - m_1^{(2)}}{\sqrt{\sigma_1^{(2)}}} \right) - c_1^{(2)} \Phi \left(\frac{0.38 - 0.01 - m_1^{(2)}}{\sqrt{\sigma_1^{(2)}}} \right)$$

$$= 0.00988$$

$$b_1(o_3) = b_1(0.38) = b_1(0.38,1) + b_1(0.38,2) = 0.024526$$

$$b_2(o_1, 1) = b_2(0.88,1)$$

$$= c_2^{(1)} \Phi \left(\frac{0.88 + 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right) - c_2^{(1)} \Phi \left(\frac{0.88 - 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right)$$

$$= 0.007893$$

$$b_2(o_1, 2) = b_2(0.88,2)$$

$$= c_2^{(2)} \Phi \left(\frac{0.88 + 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right) - c_2^{(2)} \Phi \left(\frac{0.88 - 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right)$$

$$= 0.006437$$

$$b_2(o_1) = b_2(0.88) = b_2(0.88,1) + b_2(0.88,2) = 0.014331$$

$$b_2(o_2, 1) = b_2(0.13,1)$$

$$= c_2^{(1)} \Phi \left(\frac{0.13 + 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right) - c_2^{(1)} \Phi \left(\frac{0.13 - 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right)$$

$$= 0.006896$$

$$b_2(o_2, 2) = b_2(0.13,2)$$

$$= c_2^{(2)} \Phi \left(\frac{0.13 + 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right) - c_2^{(2)} \Phi \left(\frac{0.13 - 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right)$$

$$= 0.004198$$

$$b_2(o_2) = b_2(0.13) = b_2(0.13,1) + b_2(0.13,2) = 0.011094$$

$$b_2(o_3, 1) = b_2(0.38,1)$$

$$= c_2^{(1)} \Phi \left(\frac{0.38 + 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right) - c_2^{(1)} \Phi \left(\frac{0.38 - 0.01 - m_2^{(1)}}{\sqrt{\sigma_2^{(1)}}} \right)$$

$$= 0.012744$$

$$b_2(o_3, 2) = b_2(0.38, 2)$$

$$= c_2^{(2)} \Phi \left(\frac{0.38 + 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right) - c_2^{(2)} \Phi \left(\frac{0.38 - 0.01 - m_2^{(2)}}{\sqrt{\sigma_2^{(2)}}} \right)$$

$$= 0.008622$$

$$b_2(o_3) = b_2(0.38) = b_2(0.38, 1) + b_2(0.38, 2) = 0.021366$$

$$b_3(o_1, 1) = b_3(0.88, 1)$$

$$= c_3^{(1)} \Phi \left(\frac{0.88 + 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right) - c_3^{(1)} \Phi \left(\frac{0.88 - 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right)$$

$$= 0.004414$$

$$b_3(o_1, 2) = b_3(0.88, 2)$$

$$= c_3^{(2)} \Phi \left(\frac{0.88 + 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right) - c_3^{(2)} \Phi \left(\frac{0.88 - 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right)$$

$$= 0.005352$$

$$b_3(o_1) = b_3(0.88) = b_3(0.88, 1) + b_3(0.88, 2) = 0.009766$$

$$b_3(o_2, 1) = b_3(0.13, 1)$$

$$= c_3^{(1)} \Phi \left(\frac{0.13 + 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right) - c_3^{(1)} \Phi \left(\frac{0.13 - 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right)$$

$$= 0.007303$$

$$b_3(o_2, 2) = b_3(0.13, 2)$$

$$= c_3^{(2)} \Phi \left(\frac{0.13 + 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right) - c_3^{(2)} \Phi \left(\frac{0.13 - 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right)$$

$$= 0.008124$$

$$b_3(o_2) = b_3(0.13) = b_3(0.13, 1) + b_3(0.13, 2) = 0.015427$$

$$b_3(o_3, 1) = b_3(0.38, 1)$$

$$= c_3^{(1)} \Phi \left(\frac{0.38 + 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right) - c_3^{(1)} \Phi \left(\frac{0.38 - 0.01 - m_3^{(1)}}{\sqrt{\sigma_3^{(1)}}} \right)$$

$$= 0.011572$$

$$b_3(o_3, 2) = b_3(0.38, 2)$$

$$= c_3^{(2)} \Phi \left(\frac{0.38 + 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right) - c_3^{(2)} \Phi \left(\frac{0.38 - 0.01 - m_3^{(2)}}{\sqrt{\sigma_3^{(2)}}} \right)$$

$$= 0.013081$$

$$b_3(o_3) = b_3(0.38) = b_3(0.38, 1) + b_3(0.38, 2) = 0.024653$$

$$\alpha_1(1,1) = b_1(o_1, 1)\pi_1 = 0.002363$$

$$\begin{aligned}
 \alpha_1(1,2) &= b_1(o_1, 2)\pi_1 = 0.0012 \\
 \alpha_1(1) &= b_1(o_1)\pi_1 = 0.003563 \\
 \alpha_1(2,1) &= b_2(o_1, 1)\pi_2 = 0.00276 \\
 \alpha_1(2,2) &= b_2(o_1, 2)\pi_2 = 0.002251 \\
 \alpha_1(2) &= b_2(o_1)\pi_2 = 0.005011 \\
 \alpha_1(3,1) &= b_3(o_1, 1)\pi_3 = 0.001401 \\
 \alpha_1(3,2) &= b_3(o_1, 2)\pi_3 = 0.001699 \\
 \alpha_1(3) &= b_3(o_1)\pi_3 = 0.003101 \\
 \alpha_2(1,1) &= \left(\sum_{i=1}^3 \alpha_1(i, 1)a_{i1} \right) b_1(o_2, 1) = 0.000019 \\
 \alpha_2(1,2) &= \left(\sum_{i=1}^3 \alpha_1(i, 2)a_{i1} \right) b_1(o_2, 2) = 0.00001 \\
 \alpha_2(1) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i1} \right) b_1(o_2) = 0.000056 \\
 \alpha_2(2,1) &= \left(\sum_{i=1}^3 \alpha_1(i, 1)a_{i2} \right) b_2(o_2, 1) = 0.000014 \\
 \alpha_2(2,2) &= \left(\sum_{i=1}^3 \alpha_1(i, 2)a_{i2} \right) b_2(o_2, 2) = 0.000007 \\
 \alpha_2(2) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i2} \right) b_2(o_2) = 0.00004 \\
 \alpha_2(3,1) &= \left(\sum_{i=1}^3 \alpha_1(i, 1)a_{i3} \right) b_3(o_2, 1) = 0.000017 \\
 \alpha_2(3,2) &= \left(\sum_{i=1}^3 \alpha_1(i, 2)a_{i3} \right) b_3(o_2, 2) = 0.000016 \\
 \alpha_2(3) &= \left(\sum_{i=1}^3 \alpha_1(i)a_{i3} \right) b_3(o_2) = 0.000065 \\
 \alpha_3(1,1) &= \left(\sum_{i=1}^3 \alpha_2(i, 1)a_{i1} \right) b_1(o_3, 1) = 0.00000025 \\
 \alpha_3(1,2) &= \left(\sum_{i=1}^3 \alpha_2(i, 2)a_{i1} \right) b_1(o_3, 2) = 0.0000001 \\
 \alpha_3(1) &= \left(\sum_{i=1}^3 \alpha_2(i)a_{i1} \right) b_1(o_3) = 0.000001 \\
 \alpha_3(2,1) &= \left(\sum_{i=1}^3 \alpha_2(i, 1)a_{i2} \right) b_2(o_3, 1) = 0.00000018 \\
 \alpha_3(2,2) &= \left(\sum_{i=1}^3 \alpha_2(i, 2)a_{i2} \right) b_2(o_3, 2) = 0.00000008
 \end{aligned}$$

$$\alpha_3(2) = \left(\sum_{i=1}^3 \alpha_2(i) a_{i2} \right) b_2(o_3) = 0.00000095$$

$$\alpha_3(3,1) = \left(\sum_{i=1}^3 \alpha_2(i,1) a_{i3} \right) b_3(o_3,1) = 0.00000021$$

$$\alpha_3(3,2) = \left(\sum_{i=1}^3 \alpha_2(i,2) a_{i3} \right) b_3(o_3,2) = 0.00000017$$

$$\alpha_3(3) = \left(\sum_{i=1}^3 \alpha_2(i) a_{i3} \right) b_3(o_3) = 0.000002$$

$$\beta_3(1,1) = \beta_3(2,1) = \beta_3(3,1) = 1$$

$$\beta_3(1,2) = \beta_3(2,2) = \beta_3(3,2) = 1$$

$$\beta_3(1) = \beta_3(2) = \beta_3(3) = 1$$

$$\beta_2(1,1) = \sum_{j=1}^n a_{1j} b_j(o_3, 1) \beta_3(j, 1) = 0.013347$$

$$\beta_2(1,2) = \sum_{j=1}^n a_{1j} b_j(o_3, 2) \beta_3(j, 2) = 0.010439$$

$$\beta_2(1) = \sum_{j=1}^n a_{1j} b_j(o_3) \beta_3(j) = 0.023785$$

$$\beta_2(2,1) = \sum_{j=1}^n a_{2j} b_j(o_3, 1) \beta_3(j, 1) = 0.012912$$

$$\beta_2(2,2) = \sum_{j=1}^n a_{2j} b_j(o_3, 2) \beta_3(j, 2) = 0.010427$$

$$\beta_2(2) = \sum_{j=1}^n a_{2j} b_j(o_3) \beta_3(j) = 0.023339$$

$$\beta_2(3,1) = \sum_{j=1}^n a_{3j} b_j(o_3, 1) \beta_3(j, 1) = 0.012576$$

$$\beta_2(3,2) = \sum_{j=1}^n a_{3j} b_j(o_3, 2) \beta_3(j, 2) = 0.011261$$

$$\beta_2(3) = \sum_{j=1}^n a_{3j} b_j(o_3) \beta_3(j) = 0.023838$$

$$\beta_1(1,1) = \sum_{j=1}^n a_{1j} b_j(o_2, 1) \beta_2(j, 1) = 0.0001$$

$$\beta_1(1,2) = \sum_{j=1}^n a_{1j} b_j(o_2, 2) \beta_2(j, 2) = 0.000066$$

$$\beta_1(1) = \sum_{j=1}^n a_{1j} b_j(o_2) \beta_2(j) = 0.000327$$

$$\begin{aligned}\beta_1(2,1) &= \sum_{j=1}^n a_{2j} b_j(o_2, 1) \beta_2(j, 1) = 0.000096 \\ \beta_1(2,2) &= \sum_{j=1}^n a_{2j} b_j(o_2, 2) \beta_2(j, 2) = 0.000065 \\ \beta_1(2) &= \sum_{j=1}^n a_{2j} b_j(o_2) \beta_2(j) = 0.000317 \\ \beta_1(3,1) &= \sum_{j=1}^n a_{3j} b_j(o_2, 1) \beta_2(j, 1) = 0.000096 \\ \beta_1(3,2) &= \sum_{j=1}^n a_{3j} b_j(o_2, 2) \beta_2(j, 2) = 0.000073 \\ \beta_1(3) &= \sum_{j=1}^n a_{3j} b_j(o_2) \beta_2(j) = 0.000335\end{aligned}$$

Within the E-step of the second iteration ($r=2$), the terminating criterion $P(O|\Delta)$ is calculated according to forward-backward procedure (see table IV.4.1.1) as follows:

$$P(O|\Delta) = \alpha_3(1) + \alpha_3(2) + \alpha_3(3) = 0.000004$$

Within the E-step of the second iteration ($r=2$), the joint probabilities $\xi_t(i,j)$, $\gamma_t(j)$, and $\gamma_t(j,k)$ are calculated based on formulas IV.5.2 and IV.5.8 as follows:

$$\begin{aligned}\xi_2(1,1) &= \alpha_1(1)a_{11}b_1(o_2)\beta_2(1) = 0.00000058 \\ \xi_2(1,2) &= \alpha_1(1)a_{12}b_2(o_2)\beta_2(2) = 0.00000023 \\ \xi_2(1,3) &= \alpha_1(1)a_{13}b_3(o_2)\beta_2(3) = 0.00000036 \\ \xi_2(2,1) &= \alpha_1(2)a_{21}b_1(o_2)\beta_2(1) = 0.00000049 \\ \xi_2(2,2) &= \alpha_1(2)a_{22}b_2(o_2)\beta_2(2) = 0.00000050 \\ \xi_2(2,3) &= \alpha_1(2)a_{23}b_3(o_2)\beta_2(3) = 0.00000060 \\ \xi_2(3,1) &= \alpha_1(3)a_{31}b_1(o_2)\beta_2(1) = 0.00000025 \\ \xi_2(3,2) &= \alpha_1(3)a_{32}b_2(o_2)\beta_2(2) = 0.00000019 \\ \xi_2(3,3) &= \alpha_1(3)a_{33}b_3(o_2)\beta_2(3) = 0.00000060 \\ \xi_3(1,1) &= \alpha_2(1)a_{11}b_1(o_3)\beta_3(1) = 0.00000066 \\ \xi_3(1,2) &= \alpha_2(1)a_{12}b_2(o_3)\beta_3(2) = 0.00000029 \\ \xi_3(1,3) &= \alpha_2(1)a_{13}b_3(o_3)\beta_3(3) = 0.00000037 \\ \xi_3(2,1) &= \alpha_2(2)a_{21}b_1(o_3)\beta_3(1) = 0.00000028 \\ \xi_3(2,2) &= \alpha_2(2)a_{22}b_2(o_3)\beta_3(2) = 0.00000033 \\ \xi_3(2,3) &= \alpha_2(2)a_{23}b_3(o_3)\beta_3(3) = 0.00000032 \\ \xi_3(3,1) &= \alpha_2(3)a_{31}b_1(o_3)\beta_3(1) = 0.00000038 \\ \xi_3(3,2) &= \alpha_2(3)a_{32}b_2(o_3)\beta_3(2) = 0.00000033 \\ \xi_3(3,3) &= \alpha_2(3)a_{33}b_3(o_3)\beta_3(3) = 0.00000084\end{aligned}$$

$$\begin{aligned}\gamma_1(1,1) &= \alpha_1(1,1)\beta_1(1,1) = 0.00000024 \\ \gamma_1(1,2) &= \alpha_1(1,2)\beta_1(1,2) = 0.00000008 \\ \gamma_1(1) &= \alpha_1(1)\beta_1(1) = 0.000001 \\ \gamma_1(2,1) &= \alpha_1(2,1)\beta_1(2,1) = 0.00000027 \\ \gamma_1(2,2) &= \alpha_1(2,2)\beta_1(2,2) = 0.00000015 \\ \gamma_1(2) &= \alpha_1(2)\beta_1(2) = 0.000002 \\ \gamma_1(3,1) &= \alpha_1(3,1)\beta_1(3,1) = 0.00000013 \\ \gamma_1(3,2) &= \alpha_1(3,2)\beta_1(3,2) = 0.00000012\end{aligned}$$

$$\begin{aligned}
 \gamma_1(3) &= \alpha_1(3)\beta_1(3) = 0.000001 \\
 \gamma_2(1,1) &= \alpha_2(1,1)\beta_2(1,1) = 0.00000025 \\
 \gamma_2(1,2) &= \alpha_2(1,2)\beta_2(1,2) = 0.0000001 \\
 \gamma_2(1) &= \alpha_2(1)\beta_2(1) = 0.000001 \\
 \gamma_2(2,1) &= \alpha_2(2,1)\beta_2(2,1) = 0.00000018 \\
 \gamma_2(2,2) &= \alpha_2(2,2)\beta_2(2,2) = 0.00000007 \\
 \gamma_2(2) &= \alpha_2(2)\beta_2(2) = 0.00000092 \\
 \gamma_2(3,1) &= \alpha_2(3,1)\beta_2(3,1) = 0.00000021 \\
 \gamma_2(3,2) &= \alpha_2(3,2)\beta_2(3,2) = 0.00000018 \\
 \gamma_2(3) &= \alpha_2(3)\beta_2(3) = 0.000002 \\
 \gamma_3(1,1) &= \alpha_3(1,1)\beta_3(1,1) = 0.00000025 \\
 \gamma_3(1,2) &= \alpha_3(1,2)\beta_3(1,2) = 0.00000001 \\
 \gamma_3(1) &= \alpha_3(1)\beta_3(1) = 0.000001 \\
 \gamma_3(2,1) &= \alpha_3(2,1)\beta_3(2,1) = 0.00000018 \\
 \gamma_3(2,2) &= \alpha_3(2,2)\beta_3(2,2) = 0.00000008 \\
 \gamma_3(2) &= \alpha_3(2)\beta_3(2) = 0.00000095 \\
 \gamma_3(3,1) &= \alpha_3(3,1)\beta_3(3,1) = 0.00000021 \\
 \gamma_3(3,2) &= \alpha_3(3,2)\beta_3(3,2) = 0.00000017 \\
 \gamma_3(3) &= \alpha_3(3)\beta_3(3) = 0.000002
 \end{aligned}$$

Within the M-step of the second iteration ($r=2$), the estimate $\hat{\Delta} = (\hat{a}_{ij}, \hat{b}_j(k), \hat{\pi}_j)$ is calculated based on joint probabilities $\xi_t(i,j)$ and $\gamma_t(j)$ determined at E-step.

$$\begin{aligned}
 \hat{a}_{11} &= \frac{\sum_{t=2}^3 \xi_t(1,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.499998 \\
 \hat{a}_{12} &= \frac{\sum_{t=2}^3 \xi_t(1,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.208001 \\
 \hat{a}_{13} &= \frac{\sum_{t=2}^3 \xi_t(1,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(1,l)} = 0.292001 \\
 \hat{a}_{21} &= \frac{\sum_{t=2}^3 \xi_t(2,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.305584 \\
 \hat{a}_{22} &= \frac{\sum_{t=2}^3 \xi_t(2,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.331468 \\
 \hat{a}_{23} &= \frac{\sum_{t=2}^3 \xi_t(2,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(2,l)} = 0.362948 \\
 \hat{a}_{31} &= \frac{\sum_{t=2}^3 \xi_t(3,1)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.240779 \\
 \hat{a}_{32} &= \frac{\sum_{t=2}^3 \xi_t(3,2)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.202342 \\
 \hat{a}_{33} &= \frac{\sum_{t=2}^3 \xi_t(3,3)}{\sum_{t=2}^3 \sum_{l=1}^3 \xi_t(3,l)} = 0.556879
 \end{aligned}$$

$$\begin{aligned}
 \hat{c}_1^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(1,1)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(1,l)} = 0.717523 \\
 \hat{m}_1^{(1)} &= \frac{\sum_{t=1}^3 \gamma_t(1,1) o_t}{\sum_{t=1}^3 \gamma_t(1,1)} = 0.438209
 \end{aligned}$$

$$\begin{aligned}
 \hat{\sigma}^2_1^{(1)} &= \frac{\sum_{t=1}^3 \gamma_t(1,1) (o_t - \hat{m}_1^{(1)})^2}{\sum_{t=1}^3 \gamma_t(1,1)} = 0.091906 \\
 \hat{c}_1^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(1,2)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(1,l)} = 0.282477 \\
 \hat{m}_1^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(1,2) o_t}{\sum_{t=1}^T \gamma_t(1,2)} = 0.414942 \\
 \sigma^2_1^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(1,2) (o_t - \hat{m}_1^{(2)})^2}{\sum_{t=1}^T \gamma_t(1,2)} = 0.085791 \\
 \hat{c}_2^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(2,1)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(2,l)} = 0.728952 \\
 \hat{m}_2^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(2,1) o_t}{\sum_{t=1}^T \gamma_t(2,1)} = 0.592928 \\
 \sigma^2_2^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(2,1) (o_t - \hat{m}_2^{(1)})^2}{\sum_{t=1}^T \gamma_t(2,1)} = 0.104641 \\
 \hat{c}_2^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(2,2)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(2,l)} = 0.271048 \\
 \hat{m}_2^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(2,2) o_t}{\sum_{t=1}^T \gamma_t(2,2)} = 0.623464 \\
 \sigma^2_2^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(2,2) (o_t - \hat{m}_2^{(2)})^2}{\sum_{t=1}^T \gamma_t(2,2)} = 0.100478 \\
 \hat{c}_3^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(3,1)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(3,l)} = 0.675739 \\
 \hat{m}_3^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(3,1) o_t}{\sum_{t=1}^T \gamma_t(3,1)} = 0.370764 \\
 \sigma^2_3^{(1)} &= \frac{\sum_{t=1}^T \gamma_t(3,1) (o_t - \hat{m}_3^{(1)})^2}{\sum_{t=1}^T \gamma_t(3,1)} = 0.071249 \\
 \hat{c}_3^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(3,2)}{\sum_{l=1}^K \sum_{t=1}^T \gamma_t(3,l)} = 0.324261 \\
 \hat{m}_3^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(3,2) o_t}{\sum_{t=1}^T \gamma_t(3,2)} = 0.399837 \\
 \sigma^2_3^{(2)} &= \frac{\sum_{t=1}^T \gamma_t(3,2) (o_t - \hat{m}_3^{(2)})^2}{\sum_{t=1}^T \gamma_t(3,2)} = 0.082774 \\
 \hat{\pi}_1 &= \frac{\gamma_1(1)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.307239 \\
 \hat{\pi}_2 &= \frac{\gamma_1(2)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.418962 \\
 \hat{\pi}_3 &= \frac{\gamma_1(3)}{\gamma_1(1) + \gamma_1(2) + \gamma_1(3)} = 0.273799
 \end{aligned}$$

Table IV.5.10 summarizes mixture HMM parameters resulted from the first iteration and the second iteration of EM algorithm.

Iteration	HMM parameters		
1 st	$\hat{a}_{11} = 0.483829$	$\hat{a}_{12} = 0.245210$	$\hat{a}_{13} = 0.270961$
	$\hat{a}_{21} = 0.287734$	$\hat{a}_{22} = 0.388684$	$\hat{a}_{23} = 0.323582$
	$\hat{a}_{31} = 0.235597$	$\hat{a}_{32} = 0.238932$	$\hat{a}_{33} = 0.525471$
	$\hat{c}_1^{(1)} = 0.614137$	$\hat{m}_1^{(1)} = 0.484616$	$\hat{\sigma}^2_1^{(1)} = 0.100338$
	$\hat{c}_1^{(2)} = 0.385863$	$\hat{m}_1^{(2)} = 0.442128$	$\hat{\sigma}^2_1^{(2)} = 0.093132$
	$\hat{c}_2^{(1)} = 0.582310$	$\hat{m}_2^{(1)} = 0.524775$	$\hat{\sigma}^2_2^{(1)} = 0.109779$
	$\hat{c}_2^{(2)} = 0.417690$	$\hat{m}_2^{(2)} = 0.566734$	$\hat{\sigma}^2_2^{(2)} = 0.108245$
	$\hat{c}_3^{(1)} = 0.465338$	$\hat{m}_3^{(1)} = 0.438194$	$\hat{\sigma}^2_3^{(1)} = 0.099469$
	$\hat{c}_3^{(2)} = 0.534662$	$\hat{m}_3^{(2)} = 0.448495$	$\hat{\sigma}^2_3^{(2)} = 0.101523$
	$\hat{\pi}_1 = 0.332860$	$\hat{\pi}_2 = 0.349659$	$\hat{\pi}_3 = 0.317481$
Terminating criterion $P(O \Delta) = 0.00000038$			
2 nd	$\hat{a}_{11} = 0.499998$	$\hat{a}_{12} = 0.208001$	$\hat{a}_{13} = 0.292001$
	$\hat{a}_{21} = 0.305584$	$\hat{a}_{22} = 0.331468$	$\hat{a}_{23} = 0.362948$
	$\hat{a}_{31} = 0.240779$	$\hat{a}_{32} = 0.202342$	$\hat{a}_{33} = 0.556879$
	$\hat{c}_1^{(1)} = 0.717523$	$\hat{m}_1^{(1)} = 0.438209$	$\hat{\sigma}^2_1^{(1)} = 0.091906$
	$\hat{c}_1^{(2)} = 0.282477$	$\hat{m}_1^{(2)} = 0.414942$	$\hat{\sigma}^2_1^{(2)} = 0.085791$
	$\hat{c}_2^{(1)} = 0.728952$	$\hat{m}_2^{(1)} = 0.592928$	$\hat{\sigma}^2_2^{(1)} = 0.104641$
	$\hat{c}_2^{(2)} = 0.271048$	$\hat{m}_2^{(2)} = 0.623464$	$\hat{\sigma}^2_2^{(2)} = 0.100478$
	$\hat{c}_3^{(1)} = 0.675739$	$\hat{m}_3^{(1)} = 0.370764$	$\hat{\sigma}^2_3^{(1)} = 0.071249$
	$\hat{c}_3^{(2)} = 0.324261$	$\hat{m}_3^{(2)} = 0.399837$	$\hat{\sigma}^2_3^{(2)} = 0.082774$
	$\hat{\pi}_1 = 0.307239$	$\hat{\pi}_2 = 0.418962$	$\hat{\pi}_3 = 0.273799$
Terminating criterion $P(O \Delta) = 0.000004$			

Table IV.5.10. Mixture HMM parameters resulted from the first iteration and the second iteration of EM algorithm

As seen in table IV.5.10, the EM algorithm does not converge yet when it produces two different terminating criteria at the first iteration and the second iteration. It is necessary to run more iterations so as to gain the most optimal estimate. Within this example, the EM algorithm converges absolutely after 11 iterations when the criterion $P(O|\Delta)$ approaches to value 1 at the 10th and 11st iterations. Table IV.5.11 shows mixture HMM parameter estimates along with terminating criterion $P(O|\Delta)$ at the 1st, 2nd, 10th, and 11st iterations of EM algorithm.

Iteration	HMM parameters
-----------	----------------

	$\hat{a}_{11} = 0.483829$	$\hat{a}_{12} = 0.245210$	$\hat{a}_{13} = 0.270961$
	$\hat{a}_{21} = 0.287734$	$\hat{a}_{22} = 0.388684$	$\hat{a}_{23} = 0.323582$
	$\hat{a}_{31} = 0.235597$	$\hat{a}_{32} = 0.238932$	$\hat{a}_{33} = 0.525471$
1 st	$\hat{c}_1^{(1)} = 0.614137$	$\hat{m}_1^{(1)} = 0.484616$	$\hat{\sigma}^2_1^{(1)} = 0.100338$
	$\hat{c}_1^{(2)} = 0.385863$	$\hat{m}_1^{(2)} = 0.442128$	$\hat{\sigma}^2_1^{(2)} = 0.093132$
	$\hat{c}_2^{(1)} = 0.582310$	$\hat{m}_2^{(1)} = 0.524775$	$\hat{\sigma}^2_2^{(1)} = 0.109779$
	$\hat{c}_2^{(2)} = 0.417690$	$\hat{m}_2^{(2)} = 0.566734$	$\hat{\sigma}^2_2^{(2)} = 0.108245$
	$\hat{c}_3^{(1)} = 0.465338$	$\hat{m}_3^{(1)} = 0.438194$	$\hat{\sigma}^2_3^{(1)} = 0.099469$
	$\hat{c}_3^{(2)} = 0.534662$	$\hat{m}_3^{(2)} = 0.448495$	$\hat{\sigma}^2_3^{(2)} = 0.101523$
	$\hat{\pi}_1 = 0.332860$	$\hat{\pi}_2 = 0.349659$	$\hat{\pi}_3 = 0.317481$
	Terminating criterion $P(O/\Delta) = 0.00000038$		
2 nd	$\hat{a}_{11} = 0.499998$	$\hat{a}_{12} = 0.208001$	$\hat{a}_{13} = 0.292001$
	$\hat{a}_{21} = 0.305584$	$\hat{a}_{22} = 0.331468$	$\hat{a}_{23} = 0.362948$
	$\hat{a}_{31} = 0.240779$	$\hat{a}_{32} = 0.202342$	$\hat{a}_{33} = 0.556879$
	$\hat{c}_1^{(1)} = 0.717523$	$\hat{m}_1^{(1)} = 0.438209$	$\hat{\sigma}^2_1^{(1)} = 0.091906$
	$\hat{c}_1^{(2)} = 0.282477$	$\hat{m}_1^{(2)} = 0.414942$	$\hat{\sigma}^2_1^{(2)} = 0.085791$
	$\hat{c}_2^{(1)} = 0.728952$	$\hat{m}_2^{(1)} = 0.592928$	$\hat{\sigma}^2_2^{(1)} = 0.104641$
	$\hat{c}_2^{(2)} = 0.271048$	$\hat{m}_2^{(2)} = 0.623464$	$\hat{\sigma}^2_2^{(2)} = 0.100478$
	$\hat{c}_3^{(1)} = 0.675739$	$\hat{m}_3^{(1)} = 0.370764$	$\hat{\sigma}^2_3^{(1)} = 0.071249$
	$\hat{c}_3^{(2)} = 0.324261$	$\hat{m}_3^{(2)} = 0.399837$	$\hat{\sigma}^2_3^{(2)} = 0.082774$
	$\hat{\pi}_1 = 0.307239$	$\hat{\pi}_2 = 0.418962$	$\hat{\pi}_3 = 0.273799$
	Terminating criterion $P(O/\Delta) = 0.000004$		
10 th	$\hat{a}_{11} = 0.998665$	$\hat{a}_{12} = 0$	$\hat{a}_{13} = 0.001335$
	$\hat{a}_{21} = 0$	$\hat{a}_{22} = 0$	$\hat{a}_{23} = 1$
	$\hat{a}_{31} = 1$	$\hat{a}_{32} = 0$	$\hat{a}_{33} = 0$
	$\hat{c}_1^{(1)} = 1$	$\hat{m}_1^{(1)} = 0.38$	$\hat{\sigma}^2_1^{(1)} = 4.1 * 10^{-16}$
	$\hat{c}_1^{(2)} = 0$	$\hat{m}_1^{(2)} = 0.255083$	$\hat{\sigma}^2_1^{(2)} = 0.015675$
	$\hat{c}_2^{(1)} = 1$	$\hat{m}_2^{(1)} = 0.88$	$\hat{\sigma}^2_2^{(1)} = 3.3 * 10^{-33}$
	$\hat{c}_2^{(2)} = 0$	$\hat{m}_2^{(2)} = 0.88$	$\hat{\sigma}^2_2^{(2)} = 9.0 * 10^{-10}$
	$\hat{c}_3^{(1)} = 1$	$\hat{m}_3^{(1)} = 0.130001$	$\hat{\sigma}^2_3^{(1)} = 3.2 * 10^{-7}$
	$\hat{c}_3^{(2)} = 0$	$\hat{m}_3^{(2)} = 0.463332$	$\hat{\sigma}^2_3^{(2)} = 0.097222$
	$\hat{\pi}_1 = 0$	$\hat{\pi}_2 = 1$	$\hat{\pi}_3 = 0$

	Terminating criterion $P(O/\Delta) = 1$		
11 st	$\hat{a}_{11} = 0.998665$	$\hat{a}_{12} = 0$	$\hat{a}_{13} = 0.001335$
	$\hat{a}_{21} = 0$	$\hat{a}_{22} = 0$	$\hat{a}_{23} = 1$
	$\hat{a}_{31} = 1$	$\hat{a}_{32} = 0$	$\hat{a}_{33} = 0$
	$\hat{c}_1^{(1)} = 1$	$\hat{m}_1^{(1)} = 0.38$	$\hat{\sigma}_1^{2(1)} = 4.1 * 10^{-16}$
	$\hat{c}_1^{(2)} = 0$	$\hat{m}_1^{(2)} = 0.255083$	$\hat{\sigma}_1^{2(2)} = 0.015675$
	$\hat{c}_2^{(1)} = 1$	$\hat{m}_2^{(1)} = 0.88$	$\hat{\sigma}_2^{2(1)} = 3.3 * 10^{-33}$
	$\hat{c}_2^{(2)} = 0$	$\hat{m}_2^{(2)} = 0.88$	$\hat{\sigma}_2^{2(2)} = 9.0 * 10^{-10}$
	$\hat{c}_3^{(1)} = 1$	$\hat{m}_3^{(1)} = 0.130001$	$\hat{\sigma}_3^{2(1)} = 3.2 * 10^{-7}$
	$\hat{c}_3^{(2)} = 0$	$\hat{m}_3^{(2)} = 0.463332$	$\hat{\sigma}_3^{2(2)} = 0.097222$
	$\hat{\pi}_1 = 0$	$\hat{\pi}_2 = 1$	$\hat{\pi}_3 = 0$
	Terminating criterion $P(O/\Delta) = 1$		

Table IV.5.11. Mixture HMM parameters along with terminating criteria after 14 iterations of EM algorithm

As a result, the learned parameters A , B , and Π are shown in table IV.5.12:

		Weather current day (Time point t)		
		<i>sunny</i>	<i>cloudy</i>	<i>rainy</i>
Weather previous day (Time point $t-1$)	<i>sunny</i>	$a_{11}=0.998665$	$a_{12}=0$	$a_{13}=0.001335$
	<i>cloudy</i>	$a_{21}=0$	$a_{22}=0$	$a_{23}=1$
	<i>rainy</i>	$a_{31}=1$	$a_{32}=0$	$a_{33}=0$
		<i>sunny</i>	<i>cloudy</i>	<i>rainy</i>
		$\pi_1=0$	$\pi_2=1$	$\pi_3=0$

		Humidity			
Weather	<i>sunny</i>	$p_1(o_t \theta_1)$	$\hat{c}_1^{(1)} = 1$	$\hat{m}_1^{(1)} = 0.38$	$\hat{\sigma}_1^{2(1)} = 4.1 * 10^{-16}$
			$\hat{c}_1^{(2)} = 0$	$\hat{m}_1^{(2)} = 0.255083$	$\hat{\sigma}_1^{2(2)} = 0.015675$
	<i>cloudy</i>	$p_2(o_t \theta_2)$	$\hat{c}_2^{(1)} = 1$	$\hat{m}_2^{(1)} = 0.88$	$\hat{\sigma}_2^{2(1)} = 3.3 * 10^{-33}$
			$\hat{c}_2^{(2)} = 0$	$\hat{m}_2^{(2)} = 0.88$	$\hat{\sigma}_2^{2(2)} = 9.0 * 10^{-10}$
	<i>rainy</i>	$p_3(o_t \theta_3)$	$\hat{c}_3^{(1)} = 1$	$\hat{m}_3^{(1)} = 0.130001$	$\hat{\sigma}_3^{2(1)} = 3.2 * 10^{-7}$
			$\hat{c}_3^{(2)} = 0$	$\hat{m}_3^{(2)} = 0.463332$	$\hat{\sigma}_3^{2(2)} = 0.097222$

Table IV.5.12. Mixture HMM parameters of weather example learned from EM algorithm

Such learned parameters are more appropriate to the continuous observation sequence $O = \{o_1=0.88, o_2=0.13, o_3=0.38\}$ than the original ones shown in tables IV.4.1, IV.4.2, and IV.5.8.

My main contribution in area of continuous observation HMM is to propose the practical technique to calculate basic quantities such as forward variable α_t , backward variable β_t , and joint probabilities ζ_t , γ_t based on integral of observation PDF within a very small velocity of observation value (Nguyen L., Continuous Observation Hidden Markov Model, 2016). These quantities are necessary to evaluation problem, uncovering problem, and learning problem of HMM. However, the most hazardous issue of continuous observation HMM is to determine parameter estimate $\hat{\theta}$ of the observation PDF $p(o_t|\theta)$. In previous sections, it is easy to solve the equation specified by formulas IV.5.5 and IV.5.13 to find out $\hat{\theta}$ if the PDF $p(o_t|\theta)$ belongs to normal/exponential distributions when the natural logarithm function eliminates “exponent form” of these distributions, which in turn leads to take derivatives easier. Otherwise, if such PDF belongs to complicated distributions, a new hazardous problem is issued when equation IV.5.5 is relevant to complex derivatives as follows:

$$\sum_{t=1}^T \gamma_t \frac{\partial \ln(p(o_t|\theta))}{\partial \theta} = 0$$

The suggestion of using Newton-Raphson method (Burden & Faires, 2011, pp. 67-69) to solve such equation is not optimal solution. Your attention please, we should concern the “exponent form” of normal/exponential distributions. The recognition of “exponent form” is very important. In fact, probabilistic distributions such as normal and exponent belong to exponential family of distribution. We should take advantages of exponential family instead of finding the most general method to solve formula IV.5.5. Therefore, this section focuses on exponential family of observation PDF. Later on, we will know that exponential family ranges over most of common distributions. In other words, solving successfully formula IV.5.5 with regard to exponential family is close to finding the general method for determining parameter estimate $\hat{\theta}$.

According to (Wikipedia, Exponential family, 2016), exponential family refers to a set of probabilistic distributions whose PDF (s) have the same exponential form as follows:

$$f(x|\theta) = h(x) \exp(\eta(\theta) \cdot T(x) - A(\eta(\theta)))$$

Where,

- The $\theta = (\theta_1, \theta_2, \dots, \theta_n)$ is vector of n parameters θ_i (s). Note that a scalar known as a number is considered as 1-element vector.
- The $h(x)$ is a function of x , which is called base measure.
- The $\eta(\theta)$ is function of θ , which is a vector. If $\eta(\theta) = \theta$, the PDF is called canonical. The function $\eta(\theta)$ is known as natural parameter η whose dimension may be larger than θ .
- The $T(x)$ is function of x , which is sufficient statistic.
- The notation $\eta \cdot T(x)$ denotes scalar product of $\eta(\theta)$ and $T(x)$. Note that the scalar product of two scalars is multiplication of such two scalars.
- The $A(\theta)$ is log-partition function which is used to normalize PDF. It will be mentioned later on.

All functions $\eta(\theta)$, $T(x)$, and $A(\eta(\theta))$ are known. Formula IV.5.15 specifies the exponential family PDF with note that function $\eta(\theta)$ is considered as natural parameter η .

$$f(x|\eta) = h(x) \exp(\eta \cdot T(x) - A(\eta))$$

Where η is natural parameter which is function of θ and so $A(\eta)$ is always specified by function of θ .

Formula IV.5.15. Exponential family PDF

According to (Wikipedia, Exponential family, 2016), most of common distributions belong to exponential family, for instance: normal, log-normal, exponential, gamma, chi-squared, beta, Dirichlet, Bernoulli, categorical, Poisson, geometric, inverse Gaussian, von Mises, von Mises-Fisher, Wishart, Inverse Wishart, binomial with fixed number of trials, multinomial with fixed number of trials, negative binomial with fixed number of failures.

The log-partition function $A(\eta)$ is determined based on a property of PDF that integral of any PDF from negative infinity to positive infinity is 1.

$$\begin{aligned} \int_x f(x|\theta) dx &= \int_x h(x) \exp(\eta \cdot T(x) - A(\eta)) dx \\ &= \exp(-A(\eta)) \int_x h(x) \exp(\eta \cdot T(x)) dx = 1 \end{aligned}$$

It implies $A(\eta)$ is determined by formula IV.5.16 (Jebara, 2015, p. 7).

$$A(\eta) = \ln \left(\int_x h(x) \exp(\eta \cdot T(x)) dx \right)$$

Formula IV.5.16. Log-partition function $A(\eta)$

According to author Jebara (Jebara, 2015, p. 7), the first derivative of $A(\eta)$ with regard to η is:

$$\begin{aligned} \frac{dA(\eta)}{d\eta} &= \frac{1}{\int_x h(x) \exp(\eta \cdot T(x)) dx} \frac{d(\int_x h(x) \exp(\eta \cdot T(x)) dx)}{d\eta} \\ &= \frac{\int_x h(x) \exp(\eta \cdot T(x)) T(x) dx}{\int_x h(x) \exp(\eta \cdot T(x)) dx} \\ &= \frac{\exp(-A(\eta)) \int_x h(x) \exp(\eta \cdot T(x)) T(x) dx}{\exp(-A(\eta)) \int_x h(x) \exp(\eta \cdot T(x)) dx} \\ &= \frac{\int_x h(x) \exp(\eta \cdot T(x) - A(\eta)) T(x) dx}{\int_x h(x) \exp(\eta \cdot T(x) - A(\eta)) dx} = \frac{\int_x f(x) T(x) dx}{\int_x f(x) dx} \\ &= E(T(x)|\eta) \end{aligned}$$

In general, the first derivative of $A(\eta)$ is the expectation of $T(x)$ given exponential family PDF according to formula IV.5.17 as follows:

$$A'(\eta) = E(T(x)|\eta) = E(T(x)|\theta)$$

Formula IV.5.17. First derivative of log-partition function $A(\eta)$ with regard to η

Note that the exponential family PDF is specified by formula IV.5.15 and it is possible to determine the derivative $A'(\eta)$ if $A(\eta)$ is known; for example, function $A(\eta)$ of normal PDF with mean μ and variance σ^2 is $\frac{\mu^2}{2\sigma^2} + \ln\sigma$. It is not always to calculate $A'(\eta)$ based on the expectation $E(T(x)|\eta)$.

Let $LnL(\eta)$ be the log-likelihood of exponential family PDF which is natural logarithm of exponential family PDF as seen in formula IV.5.18.

$$\begin{aligned} LnL(\eta) &= \ln(f(x|\eta)) = \ln(h(x)\exp(\eta \cdot T(x) - A(\eta))) \\ &= \ln(h(x)) + \eta \cdot T(x) - A(\eta) \end{aligned}$$

Formula IV.5.18. Log-likelihood function of exponential family PDF

Note that $LnL(\eta)$ is function of parameters η and it is also a vector. The first derivative of $LnL(\eta)$ is specified by formula IV.5.19.

$$DLnL(\eta) = \frac{dLnL(\eta)}{d\eta} = T(x) - A'(\eta) = T(x) - E(T(x)|\eta)$$

Formula IV.5.19. Derivative of log-likelihood function of exponential family PDF

The formula IV.5.5 becomes formula IV.5.20 as follows:

$$\sum_{t=1}^T \gamma_t DLnL(\eta|o_t) = 0$$

Formula IV.5.20. Equation of exponential family PDF parameters

Where o_t is continuous observation, T is the number of observations and γ_t is the joint probability specified by formula IV.4.2.1. As seen in the beginning of this section IV.5, formulas IV.5.5, IV.5.13 and IV.5.20 are derived from EM algorithm and so they are essentially maximum likelihood equations with additional information of joint probabilities γ_t (s). Please refer to article “Maximum Likelihood from Incomplete Data via the EM Algorithm” by authors Dempster, Laird, and Rubin (Dempster, Laird, & Rubin, 1977, p. 5) for more details about maximum likelihood method and EM algorithm.

For example, we have normal PDF with mean μ and variance σ^2 . Its specification includes:

$$\begin{aligned} h(x) &= \frac{1}{\sqrt{2\pi}} \\ T(x) &= (x, x^2) \\ \theta &= (\mu, \sigma^2) \\ \eta &= \left(\eta_1 = \frac{\mu}{\sigma^2}, \eta_2 = -\frac{1}{2\sigma^2} \right) \\ A(\eta) &= -\frac{\eta_1^2}{4\eta_2} + \frac{1}{2} \ln \left| \frac{1}{2\eta_2} \right| = \frac{\mu^2}{2\sigma^2} + \ln \sigma \end{aligned}$$

According to formula IV.5.15, the normal PDF is:

$$\begin{aligned} f(x|\eta) &= h(x)\exp(\eta \cdot T(x) - A(\eta)) \\ &= \frac{1}{\sqrt{2\pi}} \exp \left(\left(\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2} \right) \cdot (x, x^2) - \left(\frac{\mu^2}{2\sigma^2} + \ln \sigma \right) \right) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \end{aligned}$$

We have:

$$E(x^2) = \mu^2 + E((x-\mu)^2) = \mu^2 + \sigma^2$$

The derivative of $A(\eta)$ of normal PDF is a vector as follows:

$$A'(\eta) = E(T(x)|\eta) = (E(x), E(x^2)) = (\mu, \mu^2 + \sigma^2)$$

We have the same result if taking directly derivative on $A(\eta)$:

$$\frac{dA(\eta)}{d\eta_1} = \frac{d\left(-\frac{\eta_1^2}{4\eta_2} + \frac{1}{2}\ln\left|\frac{1}{2\eta_2}\right|\right)}{d\eta_1} = -\frac{\eta_1}{2\eta_2} = \mu$$

$$\frac{dA(\eta)}{d\eta_2} = \frac{d\left(-\frac{\eta_1^2}{4\eta_2} + \frac{1}{2}\ln\left|\frac{1}{2\eta_2}\right|\right)}{d\eta_2} = \frac{\eta_1^2}{4\eta_2^2} - \frac{1}{2\eta_2} = \mu^2 + \sigma^2$$

The derivative of log-likelihood of normal PDF is:

$$DLnL(\eta) = T(x) - E(T(x)|\eta) = (x, x^2) - (\mu, \mu^2 + \sigma^2) = (x - \mu, x^2 - \mu^2 - \sigma^2)$$

Now we solve formula IV.5.20 given normal observation PDF $p(o_t|\theta)$ with mean μ and variance σ^2 . The natural parameter of such PDF is:

$$\eta = \left(\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2} \right)$$

We have:

$$\sum_{t=1}^T \gamma_t DLnL(\eta|o_t) = (0, 0) \Leftrightarrow \begin{cases} \sum_{t=1}^T \gamma_t (o_t - \mu) = 0 \\ \sum_{t=1}^T \gamma_t (o_t^2 - \hat{\mu}^2 - \hat{\sigma}^2) = 0 \end{cases}$$

$$\Rightarrow \begin{cases} \hat{\mu} = \frac{\sum_{t=1}^T \gamma_t o_t}{\sum_{t=1}^T \gamma_t} \\ \hat{\sigma}^2 = \frac{\sum_{t=1}^T \gamma_t (o_t^2 - \hat{\mu}^2)}{\sum_{t=1}^T \gamma_t} \end{cases}$$

The estimate $\hat{\mu}$ is the same to one in formula IV.5.6 but the estimate $\hat{\sigma}^2$ is different. The previous estimate $\hat{\sigma}^2$ specified by formula IV.5.6 is:

$$\hat{\sigma}^2 - \text{previous} = \frac{\sum_{t=1}^T \gamma_t (o_t - \hat{\mu})^2}{\sum_{t=1}^T \gamma_t}$$

However, our method is still correct because the expectation of $\hat{\sigma}^2$ is kept intact. In fact, we have:

$$\begin{aligned} E(\hat{\sigma}^2 - \text{previous}) &= E\left(\frac{\sum_{t=1}^T \gamma_t (o_t - \hat{\mu})^2}{\sum_{t=1}^T \gamma_t}\right) = \frac{\sum_{t=1}^T \gamma_t E(o_t - \hat{\mu})^2}{\sum_{t=1}^T \gamma_t} \\ &= \frac{\sum_{t=1}^T \gamma_t (E(o_t^2) - 2\hat{\mu}E(o_t) + \hat{\mu}^2)}{\sum_{t=1}^T \gamma_t} = \frac{\sum_{t=1}^T \gamma_t (E(o_t^2) - 2\hat{\mu}\hat{\mu} + \hat{\mu}^2)}{\sum_{t=1}^T \gamma_t} \\ &= \frac{\sum_{t=1}^T \gamma_t (E(o_t^2) - \hat{\mu}^2)}{\sum_{t=1}^T \gamma_t} = \frac{\sum_{t=1}^T \gamma_t E(o_t^2 - \hat{\mu}^2)}{\sum_{t=1}^T \gamma_t} \\ &= E\left(\frac{\sum_{t=1}^T \gamma_t (o_t^2 - \hat{\mu}^2)}{\sum_{t=1}^T \gamma_t}\right) = E(\hat{\sigma}^2) \end{aligned}$$

In general, it is easy for us to solve estimation equation IV.5.20 for finding parameter estimates or to know that the equation IV.5.20 is impossible if the continuous PDF belongs to exponential family because the derivative of log-likelihood specified by formula IV.5.19 always exists and it is totally possible to calculate the expectation of function $T(x)$.

Now there is a question ‘‘how to estimate parameters of a PDF which does not conform to exponential family, for instance, Student’s t-distribution, F-distribution, Cauchy distribution (Wikipedia, Exponential family, 2016). In that case, we will solve formula IV.5.5 with subject to each concrete distribution. The research gives an

example of estimating t-distribution parameters. Similarly, our main point is to calculate the derivative $D\ln L(\theta)$ of log-likelihood function of t-distribution. Formula IV.5.21 specifies the PDF of t-distribution (Wikipedia, Student's t-distribution, 2016).

$$f(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

Where ν is the number of degrees of freedom and Γ is gamma function.

Formula IV.5.21. Student's t-distribution

Following are gamma function $\Gamma(x)$ and digamma function $\psi(x)$ which is derivative of natural logarithm of gamma function. Trigamma function $\psi_1(x)$ is derivative of digamma function $\psi(x)$. Formula IV.5.22 specifies $\Gamma(x)$, $\psi(x)$, and $\psi_1(x)$ (Nguyen L., Beta Likelihood Estimation and Its Application to Specify Prior Probabilities in Bayesian Network, 2016).

$$\begin{aligned}\Gamma(x) &= \int_0^{+\infty} t^{x-1} \exp(-t) dt \\ \psi(x) &= d \left(\ln(\Gamma(x)) \right) = \frac{\Gamma'(x)}{\Gamma(x)} \\ \psi_1(x) &= \psi'(x)\end{aligned}$$

Formula IV.5.22. Functions gamma, digamma, trigamma

The t-distribution replaces normal distribution for testing on means of normal population in case of small size samples. It is important distribution to hypothesis testing. When ν is parameter of t-distribution, we need to estimate it according to formula IV.5.5. The log-likelihood of t-distribution PDF is:

$$\begin{aligned}L\ln L(\nu) &= \ln \left(\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}} \right) \\ &= \ln \left(\Gamma\left(\frac{\nu+1}{2}\right) \right) - \frac{1}{2} \ln \pi - \frac{1}{2} \ln \nu - \ln \left(\Gamma\left(\frac{\nu}{2}\right) \right) - \frac{\nu+1}{2} \ln \left(1 + \frac{x^2}{\nu} \right)\end{aligned}$$

The derivative $D\ln L(\nu)$ of log-likelihood function is:

$$D\ln L(\nu) = \frac{x^2 - 1}{2(x^2 + \nu)} + \frac{1}{2} \ln \nu - \frac{1}{2} \ln(x^2 + \nu) + \psi\left(\frac{\nu+1}{2}\right) - \psi\left(\frac{\nu}{2}\right)$$

Given the t-distribution PDF of observation, the formula IV.5.5 to find out the parameter estimate $\hat{\nu}$ becomes:

$$\begin{aligned}\sum_{t=1}^T \gamma_t D\ln L(\nu|o_t) &= 0 \\ \Rightarrow \sum_{t=1}^T \gamma_t \left(\frac{o_t^2 - 1}{2(o_t^2 + \nu)} + \frac{1}{2} \ln \nu - \frac{1}{2} \ln(o_t^2 + \nu) - \psi\left(\frac{\nu+1}{2}\right) + \psi\left(\frac{\nu}{2}\right) \right) &= 0 \\ \Rightarrow \left(\ln \nu + 2\psi\left(\frac{\nu+1}{2}\right) - 2\psi\left(\frac{\nu}{2}\right) \right) \sum_{t=1}^T \gamma_t + \sum_{t=1}^T \gamma_t \left(\frac{o_t^2 - 1}{o_t^2 + \nu} - \ln(o_t^2 + \nu) \right) &= 0\end{aligned}$$

Shortly, the estimate $\hat{\nu}$ is solution of equation specified by formula IV.5.23.

$$\begin{aligned}
 g(v) &= 0 \text{ where } g(v) \\
 &= \left(\ln v + 2\psi\left(\frac{v+1}{2}\right) - 2\psi\left(\frac{v}{2}\right) \right) \sum_{t=1}^T \gamma_t \\
 &\quad + \sum_{t=1}^T \gamma_t \left(\frac{o_t^2 - 1}{o_t^2 + v} - \ln(o_t^2 + v) \right)
 \end{aligned}$$

Formula IV.5.23. Equation of t-distribution PDF parameter

Where o_t is continuous observation, T is the number of observations and γ_t is the joint probability specified by formula IV.4.2.1.

The formula IV.5.23 is much more complicated than formula IV.5.20. As aforementioned suggestion, the Newton-Raphson method (Burden & Faires, 2011, pp. 67-69) is used to find solution of given equation $g(v)=0$ along with the tangent of $g(v)$. It starts with an arbitrary value of v_0 as a solution candidate. Suppose the current value is v_k , the next value v_{k+1} is calculated based on following formula:

$$v_{k+1} = v_k - \frac{g(v_k)}{g'(v_k)}$$

The value v_k is solution of $g(v)=0$ if $g(v_k)=0$ which means that $v_{k+1}=v_k$. The most important aspect of Newton-Raphson method is that derivative of $g(v)$ must be determinate within domain of $g(v)$. Fortunately, the derivative $g'(v)$ is always determinate for all $v > 0$.

$$g'(v) = \left(\frac{1}{v} + \psi_1\left(\frac{v+1}{2}\right) - \psi_1\left(\frac{v}{2}\right) \right) \sum_{t=1}^T \gamma_t - \sum_{t=1}^T \gamma_t \frac{2o_t^2 - 1 + v}{(o_t^2 + v)^2}$$

Suppose there is only one observation $o_1=1$ and the joint probability γ_1 is 1, we have:

$$\begin{aligned}
 g(v) &= \ln\left(\frac{v}{1+v}\right) + 2\psi\left(\frac{v+1}{2}\right) - 2\psi\left(\frac{v}{2}\right) \\
 g'(v) &= \frac{1}{v(1+v)} + \psi_1\left(\frac{v+1}{2}\right) - \psi_1\left(\frac{v}{2}\right)
 \end{aligned}$$

Figure IV.5.1 shows function $g(v)$ with $o_1=1$ and $\gamma_1=1$.

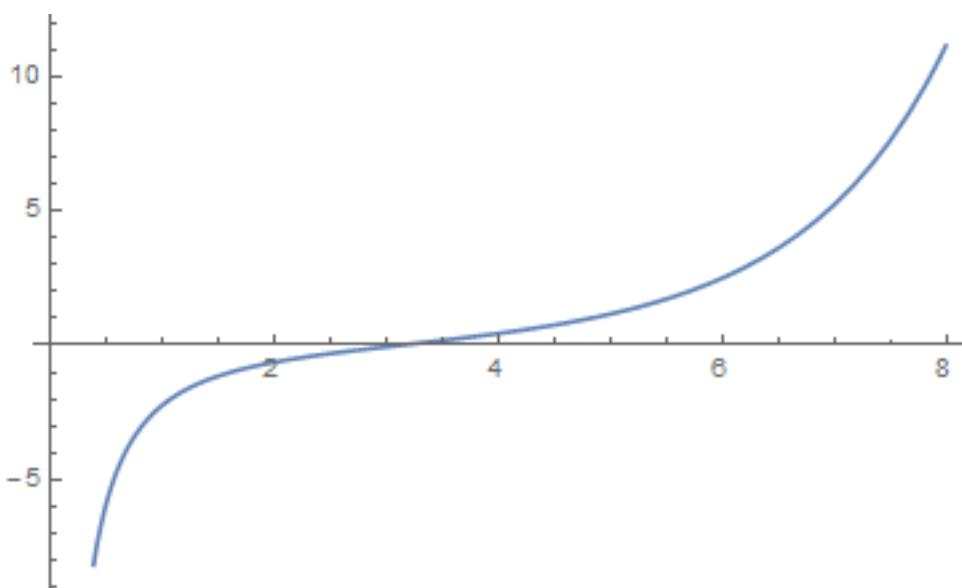


Figure IV.5.1. Graph of t-distribution parameter function

By applying the Newton-Raphson method, we receive the solution $\hat{v} \approx 3.12$ of $g(v)=0$ given starting value 3, after 6 times to run.

In conclusion, it is optimal if observation PDF belongs to exponential family where the derivative $D\ln L(\eta)$ of log-likelihood specified by formula IV.5.19 is simple, which makes solving estimation equation easier. Otherwise, Newton-Raphson method is a work-around because it can find out solution if the target function is increasing (decreasing) within the interval containing solution. We cannot solve effectively an arbitrary equation. In future, I try my best to prove whether the solution of general estimation equation specified by formula IV.5.5 exists. When asserting the existence of solution, we can apply artificial intelligence approaches such as neural network and genetic algorithm into solving formula IV.5.5 instead of using only Newton-Raphson method.

This section IV.5 ends up full introduction of HMM. The successive section IV.6 describes the new approach that uses HMM to discover and represent users' learning styles (Nguyen L., A New Approach for Modeling and Discovering Learning Styles by Using Hidden Markov Model, 2013).

IV.6. 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 parameters S, Θ, A, B, Π . 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) aforementioned in table IV.4.2.2. Suppose we choose Honey-Mumford model and Felder-Silverman model as principal models which are presented by HMM. The similar work that uses dynamic Bayesian network (DBN) to model students' learning styles is (Carmona, Castillo, & Millán, 2008). This work discovers learning styles according to students' ratings on learning objects whereas I mine state sequences of students from their observation sequences in order to discover their sequences of learning styles.

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:

- $\Delta^{(1)} = \langle S^{(1)}, \Theta^{(1)}, A^{(1)}, B^{(1)}, \Pi^{(1)} \rangle$.
- $\Delta^{(2)} = \langle S^{(2)}, \Theta^{(2)}, A^{(2)}, B^{(2)}, \Pi^{(2)} \rangle$.
- $\Delta^{(3)} = \langle S^{(3)}, \Theta^{(3)}, A^{(3)}, B^{(3)}, \Pi^{(3)} \rangle$.

Note that superscripts ⁽¹⁾, ⁽²⁾, ⁽³⁾ indicate dimensions *Verbal/Visual*, *Activist/Reflector*, *Theorist/Pragmatist*, respectively.

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.

$$S_1 = \{s_1^{(1)} = \text{verbal}, s_2^{(1)} = \text{visual}\},$$

$$S_2 = \{s_1^{(2)} = \text{activist}, s_2^{(2)} = \text{reflector}\},$$

$$S_3 = \{s_1^{(3)} = \text{theorist}, s_2^{(3)} = \text{pragmatist}\}.$$

2. Defining **initial state distributions**: Uniform probability distribution is used for each Π_i .

$$\Pi^{(1)} = \{\pi_1^{(1)} = 0.5, \pi_2^{(1)} = 0.5\}; \text{ it means that } P(\text{verbal}) = P(\text{visual}) = 0.5$$

$$\Pi^{(2)} = \{\pi_1^{(2)} = 0.5, \pi_2^{(2)} = 0.5\}; P(\text{activist}) = P(\text{reflector}) = 0.5$$

$$\Pi^{(3)} = \{\pi_1^{(3)} = 0.5, \pi_2^{(3)} = 0.5\}; P(\text{theorist}) = P(\text{pragmatist}) = 0.5$$

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, $P(\text{verbal} | \text{verbal}) = 0.7$ is obviously higher than $P(\text{visual} | \text{verbal}) = 0.3$. Transition probability matrices are shown in table IV.6.1.

$A^{(1)}$	<i>verbal</i>	<i>visual</i>
<i>verbal</i>	$a_{11}^{(1)} = 0.7$	$a_{12}^{(1)} = 0.3$
<i>visual</i>	$a_{21}^{(1)} = 0.3$	$a_{22}^{(1)} = 0.7$
<hr/>		
$A^{(2)}$	<i>activist</i>	<i>reflector</i>
<i>activist</i>	$a_{11}^{(2)} = 0.7$	$a_{12}^{(2)} = 0.3$
<i>reflector</i>	$a_{21}^{(2)} = 0.3$	$a_{22}^{(2)} = 0.7$
<hr/>		
$A^{(3)}$	<i>theorist</i>	<i>pragmatist</i>
<i>theorist</i>	$a_{11}^{(3)} = 0.7$	$a_{12}^{(3)} = 0.3$
<i>pragmatist</i>	$a_{21}^{(3)} = 0.3$	$a_{22}^{(3)} = 0.7$

Table IV.6.1. Transition probability matrices: $A^{(1)}, A^{(2)}, A^{(3)}$

4. Defining **observations**. There is a relationship between learning object learned by user and her/his learning styles. Three attributes are assigned to each learning object such as lecture, lesson, example, and test. These attributes were mentioned in (Carmona, Castillo, & Millán, 2008, p. 3).

- *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, and 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), observations are raised according to the attributes of such learning object. We must account for 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:

- $\Theta^{(1)} = \{\varphi_1^{(1)} = \text{text}, \varphi_2^{(1)} = \text{picture}, \varphi_3^{(1)} = \text{video}\}$
- $\Theta^{(2)} = \{\varphi_1^{(2)} = \text{theory}, \varphi_2^{(2)} = \text{example}, \varphi_3^{(2)} = \text{exercise}, \varphi_4^{(2)} = \text{puzzle}, \varphi_5^{(2)} = \text{low (interaction)}, \varphi_6^{(2)} = \text{medium (interaction)}, \varphi_7^{(2)} = \text{high (interaction)}\}$
- $\Theta^{(3)} = \{\varphi_1^{(3)} = \text{theory}, \varphi_2^{(3)} = \text{example}, \varphi_3^{(3)} = \text{exercise}, \varphi_4^{(3)} = \text{puzzle}, \varphi_5^{(3)} = \text{low (interaction)}, \varphi_6^{(3)} = \text{medium (interaction)}, \varphi_7^{(3)} = \text{high (interaction)}\}$

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:

- $P(\text{text} | \text{verbal}) = 0.6, P(\text{picture} | \text{verbal}) = 0.3, P(\text{video} | \text{verbal}) = 0.1.$
- $P(\text{text} | \text{visual}) = 0.2, P(\text{picture} | \text{visual}) = 0.4, P(\text{video} | \text{visual}) = 0.4.$

There are some differences in specifying observation probabilities of dimensions *Activist/Reflector* and *Theorist/Pragmatist*. As discussed in section IV.3, active learners are provided activity-oriented approach: showing content of activity (puzzle, game, etc.) 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. Note that activity here indicates puzzle, game, team building (Wikipedia, Team building, 2015), etc. 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:

- $P(\text{puzzle} | \text{activist}) = 0.4, P(\text{example} | \text{activist}) = 0.3, P(\text{theory} | \text{activist}) = 0.2, P(\text{exercise} | \text{activist}) = 0.1.$
 $P(\text{low} | \text{activist}) = 0, P(\text{medium} | \text{activist}) = 0, P(\text{high} | \text{activist}) = 1.$
- $P(\text{example} | \text{reflector}) = 0.4, P(\text{theory} | \text{reflector}) = 0.3, P(\text{exercise} | \text{reflector}) = 0.2, P(\text{puzzle} | \text{reflector}) = 0.1.$

$$P(\text{low} | \text{reflector}) = 1, P(\text{medium} | \text{reflector}) = 0, P(\text{high} | \text{reflector}) = 0.$$

Because the sum of conditional probabilities of observations on each state equals 1, we should normalize above probabilities.

- $P(\text{puzzle} \mid \text{activist}) = 0.4*4/7 = 0.23$, $P(\text{example} \mid \text{activist}) = 0.3*4/7 = 0.17$, $P(\text{theory} \mid \text{activist}) = 0.2*4/7 = 0.11$, $P(\text{exercise} \mid \text{activist}) = 0.1*4/7 = 0.06$.
 $P(\text{low} \mid \text{activist}) = 0*3/7 = 0$, $P(\text{medium} \mid \text{activist}) = 0*3/7 = 0$, $P(\text{high} \mid \text{activist}) = 1*3/7 = 0.43$.
- $P(\text{example} \mid \text{reflector}) = 0.4*4/7 = 0.23$, $P(\text{theory} \mid \text{reflector}) = 0.3*4/7 = 0.17$, $P(\text{exercise} \mid \text{reflector}) = 0.2*4/7 = 0.11$, $P(\text{puzzle} \mid \text{reflector}) = 0.1*4/7 = 0.06$.
 $P(\text{low} \mid \text{reflector}) = 1*3/7 = 0.43$, $P(\text{medium} \mid \text{reflector}) = 0*3/7 = 0$, $P(\text{high} \mid \text{reflector}) = 0*3/7 = 0$.

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 IV.6.2.

$B^{(1)}$	<i>text</i>	<i>picture</i>	<i>video</i>
<i>verbal</i>	$b_{11}^{(1)}=0.6$	$b_{12}^{(1)}=0.3$	$b_{13}^{(1)}=0.1$
<i>visual</i>	$b_{21}^{(1)}=0.2$	$b_{22}^{(1)}=0.4$	$b_{23}^{(1)}=0.4$

$B^{(2)}$	<i>theory</i>	<i>example</i>	<i>exercise</i>	<i>puzzle</i>	<i>low</i>	<i>medium</i>	<i>high</i>
<i>activist</i>	$b_{11}^{(2)}=0.11$	$b_{12}^{(2)}=0.17$	$b_{13}^{(2)}=0.06$	$b_{14}^{(2)}=0.23$	$b_{15}^{(2)}=0$	$b_{16}^{(2)}=0$	$b_{17}^{(2)}=0.43$
<i>reflector</i>	$b_{21}^{(2)}=0.17$	$b_{22}^{(2)}=0.23$	$b_{23}^{(2)}=0.11$	$b_{24}^{(2)}=0.06$	$b_{25}^{(2)}=0.43$	$b_{26}^{(2)}=0$	$b_{27}^{(2)}=0$

$B^{(3)}$	<i>theory</i>	<i>example</i>	<i>exercise</i>	<i>puzzle</i>	<i>low</i>	<i>medium</i>	<i>high</i>
<i>pragmatist</i>	$b_{11}^{(3)}=0.11$	$b_{12}^{(3)}=0.17$	$b_{13}^{(3)}=0.23$	$b_{14}^{(3)}=0.06$	$b_{15}^{(3)}=0.04$	$b_{16}^{(3)}=0.09$	$b_{17}^{(3)}=0.3$
<i>theorist</i>	$b_{21}^{(3)}=0.23$	$b_{22}^{(3)}=0.17$	$b_{23}^{(3)}=0.11$	$b_{24}^{(3)}=0.06$	$b_{25}^{(3)}=0.3$	$b_{26}^{(3)}=0.09$	$b_{27}^{(3)}=0.04$

Table IV.6.2. 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 IV.6.1, IV.6.2, and IV.6.3.

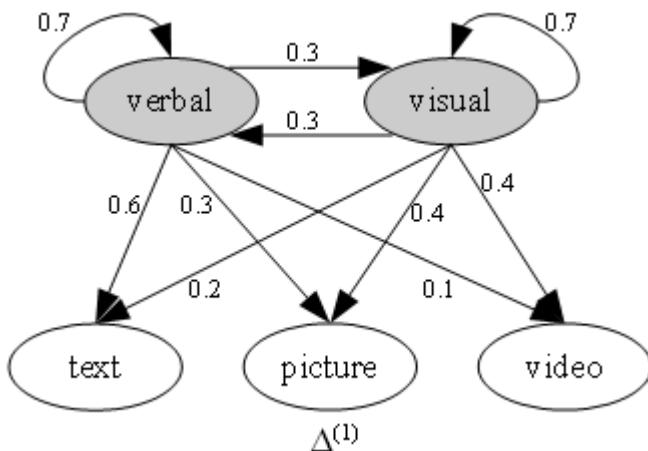


Figure IV.6.1. HMM of learning styles $\Delta^{(1)}$ (hidden states are shaded)

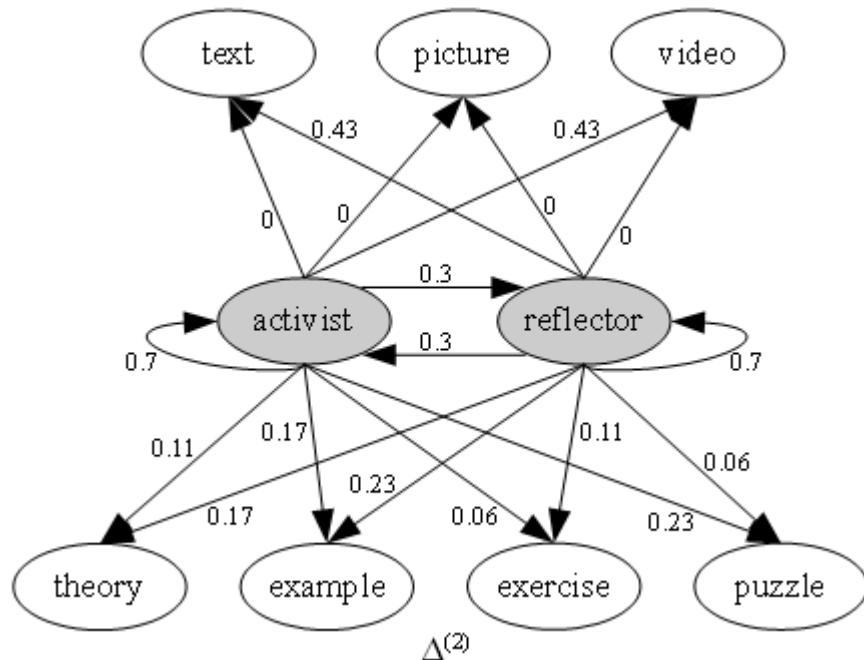


Figure IV.6.2. HMM of learning styles $\Delta^{(2)}$ (hidden states are shaded)

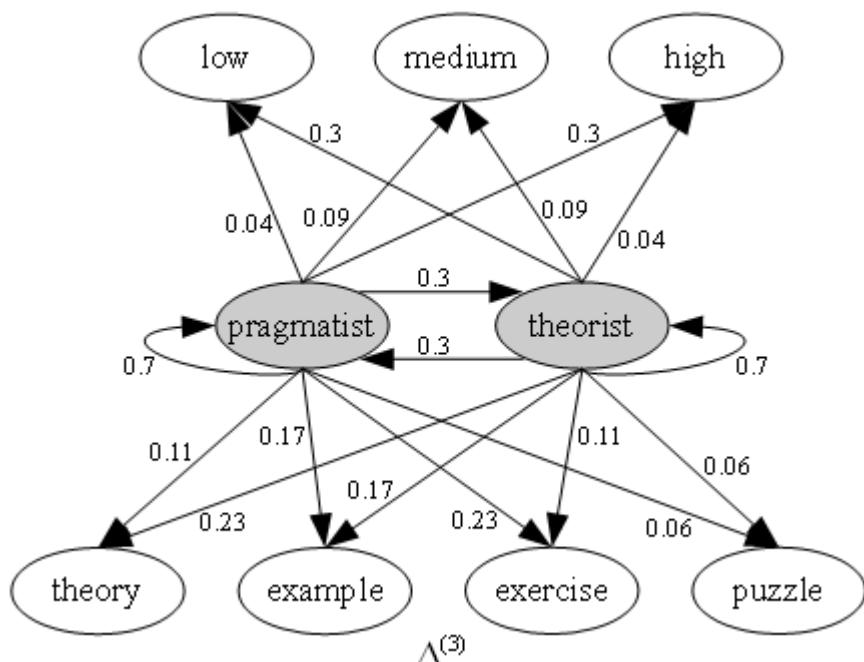


Figure IV.6.3. HMM of learning styles $\Delta^{(3)}$ (hidden states are shaded)

An example for inferring student's learning styles

Suppose learning objects that a student selects in sessions 1, 2 and 3 are LO_1 , LO_2 and LO_3 respectively, as seen in table IV.6.3.

	Format	Type	Interactive
LO_1	picture	theory	not assigned
LO_2	text	example	not assigned

LO_3	text	not assigned	low
--------	------	--------------	-----

Table IV.6.3. Learning objects selected

It is easy to recognize the sequence of user observations from the attributes *format*, *type*, *interactive*, as shown in table [IV.6.4](#).

HMM – Dimension	Observation sequence
$\Delta^{(1)}$: Dimension Verbal/Visual	picture → text → text
$\Delta^{(2)}$: Dimension Activist/Reflector	theory → example → low
$\Delta^{(3)}$: Dimension Pragmatist/Theorist	theory → example → low

Table IV.6.4. Sequence of student observations

These sequences are denoted as follows:

$$O^{(1)} = \{o_1^{(1)} = \varphi_2^{(1)} = \text{picture}, o_2^{(1)} = \varphi_1^{(1)} = \text{text}, o_3^{(1)} = \varphi_1^{(1)} = \text{text}\}$$

$$O^{(2)} = \{o_1^{(2)} = \varphi_1^{(2)} = \text{theory}, o_2^{(2)} = \varphi_2^{(2)} = \text{example}, o_3^{(2)} = \varphi_5^{(2)} = \text{low}\}$$

$$O^{(3)} = \{o_1^{(3)} = \varphi_1^{(3)} = \text{theory}, o_2^{(3)} = \varphi_2^{(3)} = \text{example}, o_3^{(3)} = \varphi_5^{(3)} = \text{low}\}$$

It is necessary to find corresponding state sequences $X^{(1)} = \{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}\}$, $X^{(2)} = \{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}\}$, and $X^{(3)} = \{x_1^{(3)}, x_2^{(3)}, x_3^{(3)}\}$ that are most suitable to have produced such observation sequences. This is uncovering problem aforementioned in subsection [IV.4.2](#).

Firstly, the Viterbi algorithm (see table [IV.4.2.2](#)) is applied into uncovering the state sequence $X^{(1)} = \{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}\}$ corresponding to HMM $\Delta^{(1)}$: Dimension Verbal/Visual. According to initialization step of Viterbi algorithm, we have:

$$\delta_1^{(1)}(1) = b_1^{(1)}(o_1^{(1)} = \varphi_2^{(1)})\pi_1^{(1)} = b_{12}^{(1)}\pi_1^{(1)} = 0.3 * 0.5 = 0.15$$

$$\delta_1^{(1)}(2) = b_2^{(1)}(o_1^{(1)} = \varphi_2^{(1)})\pi_2^{(1)} = b_{22}^{(1)}\pi_2^{(1)} = 0.4 * 0.5 = 0.2$$

$$q_1^{(1)}(1) = q_1^{(1)}(2) = 0$$

According to recurrence step of Viterbi algorithm, we have:

$$\delta_1^{(1)}(1)a_{11}^{(1)} = 0.15 * 0.7 = 0.105$$

$$\delta_1^{(1)}(2)a_{21}^{(1)} = 0.2 * 0.3 = 0.06$$

$$\begin{aligned} \delta_2^{(1)}(1) &= \left(\max_i \{\delta_1^{(1)}(i)a_{i1}^{(1)}\} \right) b_1(o_2^{(1)} = \varphi_1^{(1)}) = \left(\max_i \{\delta_1^{(1)}(i)a_{i1}^{(1)}\} \right) b_{11}^{(1)} \\ &= \left(\max_i \{\delta_1^{(1)}(1)a_{11}^{(1)}, \delta_1^{(1)}(2)a_{21}^{(1)}\} \right) b_{11}^{(1)} = 0.105 * 0.6 = 0.063 \end{aligned}$$

$$\begin{aligned} q_2^{(1)}(1) &= \operatorname{argmax}_i \{\delta_1^{(1)}(i)a_{i1}^{(1)}\} = \operatorname{argmax}_i \{\delta_1^{(1)}(1)a_{11}^{(1)}, \delta_1^{(1)}(2)a_{21}^{(1)}\} = s_1^{(1)} \\ &= \text{verbal} \end{aligned}$$

$$\delta_1^{(1)}(1)a_{12}^{(1)} = 0.15 * 0.3 = 0.045$$

$$\delta_1^{(1)}(2)a_{22}^{(1)} = 0.2 * 0.7 = 0.14$$

$$\begin{aligned} \delta_2^{(1)}(2) &= \left(\max_i \{\delta_1^{(1)}(i)a_{i2}^{(1)}\} \right) b_2(o_2^{(1)} = \varphi_1^{(1)}) = \left(\max_i \{\delta_1^{(1)}(i)a_{i2}^{(1)}\} \right) b_{21}^{(1)} \\ &= \left(\max_i \{\delta_1^{(1)}(1)a_{12}^{(1)}, \delta_1^{(1)}(2)a_{22}^{(1)}\} \right) b_{21}^{(1)} = 0.14 * 0.2 = 0.028 \end{aligned}$$

$$\begin{aligned} q_2^{(1)}(2) &= \operatorname{argmax}_i \{\delta_1^{(1)}(i)a_{i2}^{(1)}\} = \operatorname{argmax}_i \{\delta_1^{(1)}(1)a_{12}^{(1)}, \delta_1^{(1)}(2)a_{22}^{(1)}\} = s_2^{(1)} \\ &= \text{visual} \end{aligned}$$

$$\begin{aligned}
 \delta_2^{(1)}(1)a_{11}^{(1)} &= 0.063 * 0.7 = 0.0441 \\
 \delta_2^{(1)}(2)a_{21}^{(1)} &= 0.028 * 0.3 = 0.0084 \\
 \delta_3^{(1)}(1) &= \left(\max_i \{\delta_2^{(1)}(i)a_{i1}^{(1)}\} \right) b_1(o_3^{(1)} = \varphi_1^{(1)}) = \left(\max_i \{\delta_2^{(1)}(i)a_{i1}^{(1)}\} \right) b_{11}^{(1)} \\
 &= \left(\max_i \{\delta_2^{(1)}(1)a_{11}^{(1)}, \delta_2^{(1)}(2)a_{21}^{(1)}\} \right) b_{11}^{(1)} = 0.0441 * 0.6 = 0.02646 \\
 q_3^{(1)}(1) &= \operatorname{argmax}_i \{\delta_2^{(1)}(i)a_{i1}^{(1)}\} = \operatorname{argmax}_i \{\delta_2^{(1)}(1)a_{11}^{(1)}, \delta_2^{(1)}(2)a_{21}^{(1)}\} = s_1^{(1)} = \text{verbal}
 \end{aligned}$$

$$\begin{aligned}
 \delta_2^{(1)}(1)a_{12}^{(1)} &= 0.063 * 0.3 = 0.0189 \\
 \delta_2^{(1)}(2)a_{22}^{(1)} &= 0.028 * 0.7 = 0.0196 \\
 \delta_3^{(1)}(2) &= \left(\max_i \{\delta_2^{(1)}(i)a_{i2}^{(1)}\} \right) b_2(o_3^{(1)} = \varphi_1^{(1)}) = \left(\max_i \{\delta_2^{(1)}(i)a_{i2}^{(1)}\} \right) b_{21}^{(1)} \\
 &= \left(\max_i \{\delta_2^{(1)}(1)a_{12}^{(1)}, \delta_2^{(1)}(2)a_{22}^{(1)}\} \right) b_{21}^{(1)} = 0.0196 * 0.2 = 0.00392 \\
 q_3^{(1)}(2) &= \operatorname{argmax}_i \{\delta_2^{(1)}(i)a_{i2}^{(1)}\} = \operatorname{argmax}_i \{\delta_2^{(1)}(1)a_{12}^{(1)}, \delta_2^{(1)}(2)a_{22}^{(1)}\} = s_2^{(1)} \\
 &= \text{visual}
 \end{aligned}$$

According to state sequence backtracking of Viterbi algorithm, we have:

$$\begin{aligned}
 x_3^{(1)} &= \operatorname{argmax}_j \{\delta_3^{(1)}(j)\} = \operatorname{argmax}_j \{\delta_3^{(1)}(1), \delta_3^{(1)}(2)\} = s_1^{(1)} = \text{verbal} \\
 x_2^{(1)} &= q_3^{(1)}(x_3^{(1)} = s_1^{(1)}) = q_3^{(1)}(1) = s_1^{(1)} = \text{verbal} \\
 x_1^{(1)} &= q_2^{(1)}(x_2^{(1)} = s_1^{(1)}) = q_2^{(1)}(1) = s_1^{(1)} = \text{verbal}
 \end{aligned}$$

As a result, the optimal state sequence is $X^{(1)} = \{x_1^{(1)} = \text{verbal}, x_2^{(1)} = \text{verbal}, x_3^{(1)} = \text{verbal}\}$. Because the occurrences of style *verbal* in the state sequence $X^{(2)}$ are the most (3 times), it is easy to infer that student's learning style with regard to dimension Verbal/Visual is *verbal*.

Secondly, the Viterbi algorithm (see table IV.4.2.2) is applied into uncovering the state sequence $X^{(2)} = \{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}\}$ corresponding to HMM $\Delta^{(2)}$: Dimension Activist/Reflector. According to initialization step of Viterbi algorithm, we have:

$$\begin{aligned}
 \delta_1^{(2)}(1) &= b_1^{(2)}(o_1^{(2)} = \varphi_1^{(2)})\pi_1^{(2)} = b_{11}^{(2)}\pi_1^{(2)} = 0.11 * 0.5 = 0.055 \\
 \delta_1^{(2)}(2) &= b_2^{(2)}(o_1^{(2)} = \varphi_1^{(2)})\pi_2^{(2)} = b_{21}^{(2)}\pi_2^{(2)} = 0.17 * 0.5 = 0.085 \\
 q_1^{(2)}(1) &= q_1^{(2)}(2) = 0
 \end{aligned}$$

According to recurrence step of Viterbi algorithm, we have:

$$\begin{aligned}
 \delta_1^{(2)}(1)a_{11}^{(2)} &= 0.055 * 0.7 = 0.0385 \\
 \delta_1^{(2)}(2)a_{21}^{(2)} &= 0.085 * 0.3 = 0.0255 \\
 \delta_2^{(2)}(1) &= \left(\max_i \{\delta_1^{(2)}(i)a_{i1}^{(2)}\} \right) b_1(o_2^{(2)} = \varphi_2^{(2)}) = \left(\max_i \{\delta_1^{(2)}(i)a_{i1}^{(2)}\} \right) b_{12}^{(2)} \\
 &= \left(\max_i \{\delta_1^{(2)}(1)a_{11}^{(2)}, \delta_1^{(2)}(2)a_{21}^{(2)}\} \right) b_{12}^{(2)} = 0.0385 * 0.17 = 0.006545 \\
 q_2^{(2)}(1) &= \operatorname{argmax}_i \{\delta_1^{(2)}(i)a_{i1}^{(2)}\} = \operatorname{argmax}_i \{\delta_1^{(2)}(1)a_{11}^{(2)}, \delta_1^{(2)}(2)a_{21}^{(2)}\} = s_1^{(2)} \\
 &= \text{activist}
 \end{aligned}$$

$$\begin{aligned}
 \delta_1^{(2)}(1)a_{12}^{(2)} &= 0.055 * 0.3 = 0.0165 \\
 \delta_1^{(2)}(2)a_{22}^{(2)} &= 0.085 * 0.7 = 0.0595
 \end{aligned}$$

$$\begin{aligned}\delta_2^{(2)}(2) &= \left(\max_i \{\delta_1^{(2)}(i) a_{i2}^{(2)}\} \right) b_2 \left(o_2^{(2)} = \varphi_2^{(2)} \right) = \left(\max_i \{\delta_1^{(2)}(i) a_{i2}^{(2)}\} \right) b_{22}^{(2)} \\ &= \left(\max_i \{\delta_1^{(2)}(1) a_{12}^{(2)}, \delta_1^{(2)}(2) a_{22}^{(2)}\} \right) b_{22}^{(2)} = 0.0595 * 0.23 = 0.013685 \\ q_2^{(2)}(2) &= \operatorname{argmax}_i \{\delta_1^{(2)}(i) a_{i2}^{(2)}\} = \operatorname{argmax}_i \{\delta_1^{(2)}(1) a_{12}^{(2)}, \delta_1^{(2)}(2) a_{22}^{(2)}\} = s_2^{(2)} \\ &= \text{reflector}\end{aligned}$$

$$\begin{aligned}\delta_2^{(2)}(1) a_{11}^{(2)} &= 0.006545 * 0.7 = 0.004582 \\ \delta_2^{(2)}(2) a_{21}^{(2)} &= 0.013685 * 0.3 = 0.004106 \\ \delta_3^{(2)}(1) &= \left(\max_i \{\delta_2^{(2)}(i) a_{i1}^{(2)}\} \right) b_1 \left(o_3^{(2)} = \varphi_5^{(2)} \right) = \left(\max_i \{\delta_2^{(2)}(i) a_{i1}^{(2)}\} \right) b_{15}^{(2)} \\ &= \left(\max_i \{\delta_2^{(2)}(1) a_{11}^{(2)}, \delta_2^{(2)}(2) a_{21}^{(2)}\} \right) b_{15}^{(2)} = 0.004582 * 0 = 0 \\ q_3^{(2)}(1) &= \operatorname{argmax}_i \{\delta_2^{(2)}(i) a_{i1}^{(2)}\} = \operatorname{argmax}_i \{\delta_2^{(2)}(1) a_{11}^{(2)}, \delta_2^{(2)}(2) a_{21}^{(2)}\} = s_1^{(2)} \\ &= \text{activist}\end{aligned}$$

$$\begin{aligned}\delta_2^{(2)}(1) a_{12}^{(2)} &= 0.006545 * 0.3 = 0.001964 \\ \delta_2^{(2)}(2) a_{22}^{(2)} &= 0.013685 * 0.7 = 0.009580 \\ \delta_3^{(2)}(2) &= \left(\max_i \{\delta_2^{(2)}(i) a_{i2}^{(2)}\} \right) b_2 \left(o_3^{(2)} = \varphi_5^{(2)} \right) = \left(\max_i \{\delta_2^{(2)}(i) a_{i2}^{(2)}\} \right) b_{25}^{(2)} \\ &= \left(\max_i \{\delta_2^{(2)}(1) a_{12}^{(2)}, \delta_2^{(2)}(2) a_{22}^{(2)}\} \right) b_{25}^{(2)} = 0.009580 * 0.43 \\ &= 0.004119 \\ q_3^{(2)}(2) &= \operatorname{argmax}_i \{\delta_2^{(2)}(i) a_{i2}^{(2)}\} = \operatorname{argmax}_i \{\delta_2^{(2)}(1) a_{12}^{(2)}, \delta_2^{(2)}(2) a_{22}^{(2)}\} = s_2^{(2)} \\ &= \text{reflector}\end{aligned}$$

According to state sequence backtracking of Viterbi algorithm, we have:

$$\begin{aligned}x_3^{(2)} &= \operatorname{argmax}_j \{\delta_3^{(2)}(j)\} = \operatorname{argmax}_j \{\delta_3^{(2)}(1), \delta_3^{(2)}(2)\} = s_2^{(2)} = \text{reflector} \\ x_2^{(2)} &= q_3^{(2)}(x_3^{(2)} = s_2^{(2)}) = q_3^{(2)}(2) = s_2^{(2)} = \text{reflector} \\ x_1^{(2)} &= q_2^{(2)}(x_2^{(2)} = s_2^{(2)}) = q_2^{(2)}(2) = s_2^{(2)} = \text{reflector}\end{aligned}$$

As a result, the optimal state sequence is $X^{(2)} = \{x_1^{(2)} = \text{reflector}, x_2^{(2)} = \text{reflector}, x_3^{(2)} = \text{reflector}\}$. Because the occurrences of style *reflector* in the state sequence $X^{(2)}$ are the most (3 times), it is easy to infer that student's learning style with regard to dimension Activist/Reflector is *reflector*.

Thirdly, the Viterbi algorithm (see table IV.4.2.2) is applied into uncovering the state sequence $X^{(3)} = \{x_1^{(3)}, x_2^{(3)}, x_3^{(3)}\}$ corresponding to HMM $\Delta^{(3)}$: Dimension Pragmatist/Theorist. According to initialization step of Viterbi algorithm, we have:

$$\begin{aligned}\delta_1^{(3)}(1) &= b_1^{(3)} \left(o_1^{(3)} = \varphi_1^{(3)} \right) \pi_1^{(3)} = b_{11}^{(3)} \pi_1^{(3)} = 0.11 * 0.5 = 0.055 \\ \delta_1^{(3)}(2) &= b_2^{(3)} \left(o_1^{(3)} = \varphi_1^{(3)} \right) \pi_2^{(3)} = b_{21}^{(3)} \pi_2^{(3)} = 0.23 * 0.5 = 0.115 \\ q_1^{(3)}(1) &= q_1^{(3)}(2) = 0\end{aligned}$$

According to recurrence step of Viterbi algorithm, we have:

$$\begin{aligned}\delta_1^{(3)}(1) a_{11}^{(3)} &= 0.055 * 0.7 = 0.0385 \\ \delta_1^{(3)}(2) a_{21}^{(3)} &= 0.115 * 0.3 = 0.0345\end{aligned}$$

$$\begin{aligned}\delta_2^{(3)}(1) &= \left(\max_i\{\delta_1^{(3)}(i)a_{i1}^{(3)}\}\right)b_1(o_2^{(3)} = \varphi_2^{(3)}) = \left(\max_i\{\delta_1^{(3)}(i)a_{i1}^{(3)}\}\right)b_{12}^{(3)} \\ &= \left(\max_i\{\delta_1^{(3)}(1)a_{11}^{(3)}, \delta_1^{(3)}(2)a_{21}^{(3)}\}\right)b_{12}^{(3)} = 0.0385 * 0.17 = 0.006545 \\ q_2^{(3)}(1) &= \operatorname{argmax}_i\{\delta_1^{(3)}(i)a_{i1}^{(3)}\} = \operatorname{argmax}_i\{\delta_1^{(3)}(1)a_{11}^{(3)}, \delta_1^{(3)}(2)a_{21}^{(3)}\} = s_1^{(3)} \\ &= \text{pragmatist}\end{aligned}$$

$$\begin{aligned}\delta_1^{(3)}(1)a_{12}^{(3)} &= 0.055 * 0.3 = 0.0165 \\ \delta_1^{(3)}(2)a_{22}^{(3)} &= 0.115 * 0.7 = 0.0805 \\ \delta_2^{(3)}(2) &= \left(\max_i\{\delta_1^{(3)}(i)a_{i2}^{(3)}\}\right)b_2(o_2^{(3)} = \varphi_2^{(3)}) = \left(\max_i\{\delta_1^{(3)}(i)a_{i2}^{(3)}\}\right)b_{22}^{(3)} \\ &= \left(\max_i\{\delta_1^{(3)}(1)a_{12}^{(3)}, \delta_1^{(3)}(2)a_{22}^{(3)}\}\right)b_{22}^{(3)} = 0.0805 * 0.17 = 0.013685 \\ q_2^{(3)}(2) &= \operatorname{argmax}_i\{\delta_1^{(3)}(i)a_{i2}^{(3)}\} = \operatorname{argmax}_i\{\delta_1^{(3)}(1)a_{12}^{(3)}, \delta_1^{(3)}(2)a_{22}^{(3)}\} = s_2^{(3)} \\ &= \text{theorist}\end{aligned}$$

$$\begin{aligned}\delta_2^{(3)}(1)a_{11}^{(3)} &= 0.006545 * 0.7 = 0.004582 \\ \delta_2^{(3)}(2)a_{21}^{(3)} &= 0.013685 * 0.3 = 0.004106 \\ \delta_3^{(3)}(1) &= \left(\max_i\{\delta_2^{(3)}(i)a_{i1}^{(3)}\}\right)b_1(o_3^{(3)} = \varphi_5^{(3)}) = \left(\max_i\{\delta_2^{(3)}(i)a_{i1}^{(3)}\}\right)b_{15}^{(3)} \\ &= \left(\max_i\{\delta_2^{(3)}(1)a_{11}^{(3)}, \delta_2^{(3)}(2)a_{21}^{(3)}\}\right)b_{15}^{(3)} = 0.004582 * 0.04 \\ &= 0.000183 \\ q_3^{(3)}(1) &= \operatorname{argmax}_i\{\delta_2^{(3)}(i)a_{i1}^{(3)}\} = \operatorname{argmax}_i\{\delta_2^{(3)}(1)a_{11}^{(3)}, \delta_2^{(3)}(2)a_{21}^{(3)}\} = s_1^{(3)} \\ &= \text{pragmatist}\end{aligned}$$

$$\begin{aligned}\delta_2^{(3)}(1)a_{12}^{(3)} &= 0.006545 * 0.3 = 0.001964 \\ \delta_2^{(3)}(2)a_{22}^{(3)} &= 0.013685 * 0.7 = 0.009580 \\ \delta_3^{(3)}(2) &= \left(\max_i\{\delta_2^{(3)}(i)a_{i2}^{(3)}\}\right)b_2(o_3^{(3)} = \varphi_5^{(3)}) = \left(\max_i\{\delta_2^{(3)}(i)a_{i2}^{(3)}\}\right)b_{25}^{(3)} \\ &= \left(\max_i\{\delta_2^{(3)}(1)a_{12}^{(3)}, \delta_2^{(3)}(2)a_{22}^{(3)}\}\right)b_{25}^{(3)} = 0.009580 * 0.3 \\ &= 0.002874 \\ q_3^{(3)}(2) &= \operatorname{argmax}_i\{\delta_2^{(3)}(i)a_{i2}^{(3)}\} = \operatorname{argmax}_i\{\delta_2^{(3)}(1)a_{12}^{(3)}, \delta_2^{(3)}(2)a_{22}^{(3)}\} = s_2^{(3)} \\ &= \text{theorist}\end{aligned}$$

According to state sequence backtracking of Viterbi algorithm, we have:

$$\begin{aligned}x_3^{(3)} &= \operatorname{argmax}_j\{\delta_3^{(3)}(j)\} = \operatorname{argmax}_j\{\delta_3^{(3)}(1), \delta_3^{(3)}(2)\} = s_2^{(3)} = \text{theorist} \\ x_2^{(3)} &= q_3^{(3)}(x_3^{(3)} = s_2^{(3)}) = q_3^{(3)}(2) = s_2^{(3)} = \text{theorist} \\ x_1^{(3)} &= q_2^{(3)}(x_2^{(3)} = s_2^{(3)}) = q_2^{(3)}(2) = s_2^{(3)} = \text{theorist}\end{aligned}$$

As a result, the optimal state sequence is $X^{(3)} = \{x_1^{(3)} = \text{theorist}, x_2^{(3)} = \text{theorist}, x_3^{(3)} = \text{theorist}\}$. Because the occurrences of style *theorist* in the state sequence $X^{(3)}$ are the most (3 times), it is easy to infer that student's learning style with regard to dimension Pragmatist/Theorist is *theorist*.

Table IV.6.5 shows the results including state sequences and student's learning styles.

HMM - Dimension	Observation sequence	State sequence	Student style
$\Delta^{(1)}$	picture → text → text	verbal → verbal → verbal	verbal
$\Delta^{(2)}$	theory → example → low	reflector → reflector → reflector	reflector
$\Delta^{(3)}$	theory → example → low	theorist → theorist → theorist	theorist

Table IV.6.5. 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, for example, providing her/him theoretical textbooks.

Table [IV.6.5](#) shows the ultimate result of proposed approach mentioned in this section [IV.6](#). The next section [IV.7](#) is the evaluation of the proposed approach.

IV.7. 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 chapter [IV](#) 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 chapter [III](#)) 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 called learning history sub-model which is described in next chapter [V](#). Are you ready to survey the most important sub-model which will be discussed in next chapter [V](#)?

Chapter V. Learning history sub-model

The 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 which are mentioned in successive chapters [III](#), [IV](#), and [V](#), respectively. Please pay attention to the coherence of such three main chapters together with respective sub-models. Now this chapter [V](#) focuses on the learning style sub-model.

Recall that 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 chapters [III](#) and [IV](#), 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 chapters [III](#) and [IV](#) 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. The [AHA! engine](#) (De Bra, Smits, & Stash, 2006) provides an excellent mechanism for storing and manipulating learner profiles in form of XML files and relation database; please see sub-section [I.2.3.3](#) for more details about AHA! engine. The user modeling system Zebra inherits all excellent aspects of AHA! engine in order to support this function.
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 sub-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 chapters [III](#) and [IV](#) about knowledge sub-model and learning style sub-model. So I will describe in this section [V](#) the approaches that learning history sub-model applies to perform recommendation tasks, discover another learners’ characteristic (namely, user interest) 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.2.1.2) 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 (section V.1), discovering user interests (section V.2) and constructing user groups (section V.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 section V.1 which mentions how to recommend learning concepts based on mining learning history.

V.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 section V.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 like **WOW** (see figure II.2.4.18), student accesses learning materials / does exercises relating domain concepts in sessions. Her/his learning sequences, which are lists of concepts accessed after the whole of 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 student in her/his next visits. This results in enhancing the quality of adaptive learning system. The process of mining sequential patterns from sequence database S is sequential pattern mining as aforementioned. 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 part and right-hand part in form $c_i \rightarrow c_j$. The left-hand part c_i is considered as source concept, the right-hand part c_j 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 her/his series of chosen items are recorded as sequences to

construct the sequence database which is mined to find frequently repeated patterns she/he can choose in future. In learning context, items can be domain concepts / learning objects which students access or learn. First, we should glance over basic concepts and applications of sequential pattern mining, especially in context of 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. There is no order in itemset, for example, two notations (i_1, i_2, i_3) and (i_2, i_3, i_1) denote the same itemset. 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. Your attention please, order is ignored in itemset but it is mandatory in sequence. 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 section [V.1](#) from state sequences and observation sequences relevant to hidden Markov model aforementioned previous section [IV.4](#).

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 $1 \leq k_1 < k_2 < \dots < k_n \leq m$ 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 $s_1 = \langle x_1, x_4 \rangle, s_2 = \langle x_1, x_2, x_3 \rangle, s_3 = \langle x_5, x_6 \rangle$ but s_1, s_2 are often shown in detailed forms: $s_1 = \langle i_1(i_1, i_2, i_3, i_4) \rangle, s_2 = \langle i_1(i_1, i_3)(i_1, i_2, i_3) \rangle, s_3 = \langle i_3(i_1, i_2) \rangle$

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 α if $\alpha \subseteq s$. The support of sequence α is ratio of records containing α to the size of database S , denoted $support(\alpha) = |\{\langle sid, s \rangle / \alpha \subseteq s\}| / |S|$. In practice, support of sequence α is the number of records containing α , $support(\alpha) = |\{\langle sid, s \rangle / \alpha \subseteq s\}|$ given fixed database S . Similar to association rules, given the number min_sup as support threshold, if the support of α is greater than or equals min_sup , namely $support(\alpha) \geq min_sup$ then α 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 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. The term “**class & OOP**” is abbreviation of “**class and object-oriented programming**”. Note that Java is a popular object-oriented programming language <https://www.oracle.com/java>. At our e-learning website like **WOW** (see figure II.2.4.18), 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 students’ sessions. Table V.1.1 shows sequence database S created from these learning sequences which are constructed from learning sessions, in turn. There are 5 sequences: $\langle od \rangle$, $\langle (sc)o(fld) \rangle$, $\langle (ov)fp \rangle$, $\langle o(fd)p \rangle$, $\langle p \rangle$ and 8 itemsets: o , d , (sc) , (fld) , (ov) , f , p , (fd) in sequence database S .

Student	Session	Concept accessed
1	Aug 5 10:20:01	o
1	Aug 5 10:26:12	d
2	Aug 6 08:20:01	s, c
2	Aug 6 14:15:01	o
2	Aug 6 15:00:00	f, l, d
3	Aug 7 12:30:00	o, v
3	Aug 7 13:35:00	f
3	Aug 7 15:50:00	p
4	Aug 8 07:14:20	o
4	Aug 8 07:40:25	f, d
4	Aug 8 10:17:20	p
5	Aug 8 10:26:15	p



ID	Learning sequences	Length
1	$\langle od \rangle$	2
2	$\langle (sc)o(fld) \rangle$	6
3	$\langle (ov)fp \rangle$	4
4	$\langle o(fd)p \rangle$	4
5	$\langle p \rangle$	1

Table V.1.1. Learning sessions → learning sequences

Students accessed learning material in their past sessions, how system recommends appropriate domain concepts to students 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 V.1.1. The approach to break patterns into concepts is proposed in sub-section V.1.2. Sub-section V.1.3 is the evaluation. Note, sequences mentioned in the research are considered as learning sequences by default.

V.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 sub-section [V.1.1.1](#).
2. *Pattern-growth approach* based on pattern-growth mining of frequent patterns in transaction database. Pattern-growth approach is mentioned in sub-section [V.1.1.2](#).

V.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 given frequent sequence is a frequent sequence too and vice versa, every super-sequence of a given infrequent sequence is an infrequent sequence too*” 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 4 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 number of sequences which containing x_i given fixed database S . Note that x_i is contained in a sequence if any itemset in such sequence contains x_i . 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 ($\geq min_sup$). If sequence s is recognized as large sequence (frequent sequence) then all its itemsets are litemsets.

Suppose all itemsets 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 litemsets in S with support threshold 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 $min_sup=2$, there are just litemsets $o, f, d, (fd), p$ from learning sequence database S shown in table [V.1.1.1](#). Recall that letters o, f, d , and p denote Java course concepts: class & OOP, interface, data type, and package, respectively. All litemsets are mapped to integers for convenience and each of them looks like as single item appropriate to re-definition of sequence length. Table [V.1.1.1.1](#) shows litemsets mapped to integers together their supports.

<i>litemset</i>	<i>mapped to</i>	<i>support</i>
<i>o</i>	1	4
<i>f</i>	2	3
<i>d</i>	3	3
<i>(fd)</i>	4	2
<i>p</i>	5	3

Table V.1.1.1.1. Large itemsets are mapped to integers

For explanation, the support of itemset *o* is 4 because there are 4 sequences in database *S* (see table V.1.1) such as $\langle od \rangle$, $\langle (sc)o(fd) \rangle$, $\langle (ov)fp \rangle$, and $\langle o(fd)p \rangle$ containing itemset *o*. Thus, *o* is litemset.

2. **Transformation phase** (Agrawal & Srikant, 1995, p. 4): In this phase, each sequence contains only litemsets occurring in such sequence. If a sequence has no litemset, it will be removed from sequence database *S* but it still contributes to the count of sequences. Table V.1.1.1.2 shows transformed (learning) sequences.

ID	Original sequence	Transformed sequence	Mapped
1	$\langle od \rangle$	$\langle od \rangle$	$\langle 13 \rangle$
2	$\langle (sc)o(fd) \rangle$	$\langle o(fd(fd)) \rangle$	$\langle 1(234) \rangle$
3	$\langle (ov)fp \rangle$	$\langle opf \rangle$	$\langle 125 \rangle$
4	$\langle o(fd)p \rangle$	$\langle o(fd(fd))p \rangle$	$\langle 1(234)5 \rangle$
5	$\langle p \rangle$	$\langle p \rangle$	$\langle 5 \rangle$

Table V.1.1.1.2. Transformed sequences

For example, given original sequence $\langle (sc)o(fd) \rangle$, infrequent itemset *(sc)* is removed, itemset *o* is kept and itemset *(fd)* contains three potential litemsets such as *f*, *d*, and *(fd)*. It is the reason that the itemset *(fd)* is transformed into *(fd(fd))*. As a result, the original sequence $\langle (dp)o(fd) \rangle$ is transformed into $\langle o(fd(fd)) \rangle$ which, in turn, is mapped to $\langle 1(234) \rangle$. Note, as seen in table V.1.1.1.1, itemsets *o*, *f*, *d*, and *(fd)* are mapped to integers 1, 2, 3, and 4, respectively.

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 large *k*-sequences where *k* is increased until no large sequence can be found. Let L_k is a set of large *k*-sequences. Each *k*th iteration includes three steps as follows:

- a. At the beginning of *k*th scan, all sequences in L_{k-1} are joined to generate candidate set C_k which is the set of new potential large *k*-sequences. These potential large *k*-sequences are also called candidate *k*-sequences. Two sequences in L_{k-1} that form a candidate *k*-sequences have *k*-2 same litemsets.

- b. After that, the supports of candidate *k*-sequences in C_k are computed and assessed whether they satisfy *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} at the next $(k+1)^{th}$ iteration.

The technique for generating candidate set C_k (in step 3.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 sequence are large sequences too, it is necessary to prune off sequence $c \in C_k$ if any sub-sequence of c does not occur in L_{k-1} .

For instance, table V.1.1.1.3 shows results from the 2th iteration of sequence phrase including generating candidate 2-sequences and large 2-sequences together their supports. For convention, the notation $\langle 12 \rangle : 3$ indicates that the sequence $\langle 12 \rangle$ has support $support(\langle 12 \rangle) = 3$.

L_1	C_2	L_2
$\langle 1 \rangle : 4$	$\langle 12 \rangle : 3$	$\langle 12 \rangle : 3$
$\langle 2 \rangle : 3$	$\langle 13 \rangle : 3$	$\langle 13 \rangle : 3$
$\langle 3 \rangle : 3$	$\langle 14 \rangle : 2$	$\langle 14 \rangle : 2$
$\langle 4 \rangle : 2$	$\langle 15 \rangle : 2$	$\langle 15 \rangle : 2$
$\langle 5 \rangle : 3$	$\langle 21 \rangle : 0$	$\langle 25 \rangle : 2$
	$\langle 23 \rangle : 0$	
	$\langle 24 \rangle : 0$	
	$\langle 25 \rangle : 2$	
	$\langle 31 \rangle : 0$	
	$\langle 32 \rangle : 0$	
	$\langle 34 \rangle : 0$	
	$\langle 35 \rangle : 1$	
	$\langle 41 \rangle : 0$	
	$\langle 42 \rangle : 0$	
	$\langle 43 \rangle : 0$	
	$\langle 45 \rangle : 1$	
	$\langle 51 \rangle : 0$	
	$\langle 52 \rangle : 0$	
	$\langle 53 \rangle : 0$	
	$\langle 54 \rangle : 0$	

Table V.1.1.1.3. Candidate 2-sequences and large 2-sequences together their supports

For explanation, candidate 2-sequence $\langle 12 \rangle$ is formed by joining two large 1-sequences $\langle 1 \rangle$ and $\langle 2 \rangle$. The support of the sequence $\langle 12 \rangle$ is 3 because there are 3 sequences in database S (table V.1.1) such as $\langle 1(234) \rangle$, $\langle 125 \rangle$, and $\langle 1(234)5 \rangle$ containing the sequence $\langle 12 \rangle$.

Consequently, table V.1.1.1.4 shows results from the 3th iteration of sequence phrase including generating candidate 3-sequences and large 3-sequences together their supports.

L_2	C_3	C_3 (pruned)	L_3

$\langle 12 \rangle : 3$	$\langle 123 \rangle$	$\langle 123 \rangle : 0$	$\langle 125 \rangle : 2$
$\langle 13 \rangle : 3$	$\langle 124 \rangle$	$\langle 124 \rangle : 0$	
$\langle 14 \rangle : 2$	$\langle 125 \rangle$	$\langle 125 \rangle : 2$	
$\langle 15 \rangle : 2$	$\langle 132 \rangle$		
$\langle 25 \rangle : 2$	$\langle 134 \rangle$ $\langle 135 \rangle$ $\langle 142 \rangle$ $\langle 143 \rangle$ $\langle 145 \rangle$ $\langle 152 \rangle$ $\langle 153 \rangle$ $\langle 154 \rangle$		

Table V.1.1.1.4. Candidate 3-sequences and large 3-sequences together their supports

For explanation, candidate 3-sequence $\langle 132 \rangle$ is formed by joining two large 2-sequences $\langle 13 \rangle$ and $\langle 12 \rangle$ when such two large 2-sequences share the same itemset (1). It is also necessary to explain pruning task which accelerates the sequence phrase by reducing candidate set. For example, the candidate 3-sequence $\langle 132 \rangle$ is pruned off before its support is calculated because its subsequence $\langle 32 \rangle$ is not contained in L_2 . The iterative process stops because there is only one large 3-sequence $\langle 125 \rangle$.

4. **Maximal phase** (Agrawal & Srikant, 1995, p. 4): Finally, let L be the set of all larger sequences that were discovered previously, 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  $s$  do “remove sub-sequences of  $s$  from  $L$ ”
```

In this example, by collecting large sequences from tables [V.1.1.1.3](#) and [V.1.1.1.4](#), the set L has 6 large sequences as follows: $L = \{\langle 12 \rangle, \langle 13 \rangle, \langle 14 \rangle, \langle 15 \rangle, \langle 25 \rangle, \langle 125 \rangle\}$. Table [V.1.1.1.5](#) shows these 6 large sequences together their inverse mappings and supports.

large sequence	inversely mapped	support
$\langle 12 \rangle$	$\langle of \rangle$	3
$\langle 13 \rangle$	$\langle od \rangle$	3
$\langle 14 \rangle$	$\langle o(fd) \rangle$	2
$\langle 15 \rangle$	$\langle op \rangle$	2
$\langle 25 \rangle$	$\langle fp \rangle$	2
$\langle 125 \rangle$	$\langle ofp \rangle$	2

Table V.1.1.1.5. Large sequences resulted from sequence phrase

Please see table [V.1.1.1.1](#) for mapping itemsets to integers; recall that letters o, f, d , and p denote Java course concepts: class & OOP, interface, data type, and package, respectively.

Because large sequences such as $\langle 12 \rangle$, $\langle 15 \rangle$, and $\langle 25 \rangle$ are sub-sequences of $\langle 125 \rangle$, they are removed from L . As a result, sequential patterns are $\langle 13 \rangle$, $\langle 14 \rangle$,

and $\langle 125 \rangle$ which are mapped inversely as $\langle od \rangle$, $\langle o(fd) \rangle$, and $\langle ofp \rangle$. Table V.1.1.1.6 shows these sequential patterns.

<i>sequential pattern</i>	<i>inversely mapped</i>	<i>support</i>
$\langle 13 \rangle$	$\langle od \rangle$	3
$\langle 14 \rangle$	$\langle o(fd) \rangle$	2
$\langle 125 \rangle$	$\langle ofp \rangle$	2

Table V.1.1.1.6. Sequential patterns resulted from AprioriAll algorithm

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 tests them to find the actually frequent sequences which will be re-used for next pass (Han, Pei, & Yan, 2005, p. 188), 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 twelve 2-sequences, $C_2 = \{\langle aa \rangle, \langle ab \rangle, \langle ac \rangle, \langle ba \rangle, \langle bb \rangle, \langle bc \rangle, \langle ca \rangle, \langle cb \rangle, \langle cc \rangle, \langle (ab) \rangle, \langle (ac) \rangle, \langle (bc) \rangle\}$. If L_1 have n 1-sequence, it generates $n^2 + \frac{n(n-1)}{2}$ elements in candidate set C_2 (Han, Pei, & Yan, 2005, p. 189).

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 V.1.1.1.7 shows how to crack sequence database (table V.1.1) into vertical format data sets.

<i>sid</i>	<i>eid</i>	<i>itemsets (item)</i>
1	1	<i>o</i>
1	2	<i>d</i>
2	1	<i>sc</i>

2	2	<i>o</i>
2	3	<i>fld</i>
3	1	<i>ov</i>
3	2	<i>f</i>
3	3	<i>p</i>
4	1	<i>o</i>
4	2	<i>fd</i>
4	3	<i>p</i>
5	1	<i>p</i>

Table V.1.1.1.7. 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 V.1.1.1.8 shows how to join two frequent itemsets (Han, Pei, & Yan, 2005, p. 191).

<i>o</i>		<i>p</i>		<i>op</i>		
<i>sid</i>	<i>eid</i>	<i>sid</i>	<i>eid</i>	<i>sid</i>	<i>eid(o)</i>	<i>eid(f)</i>
1	1	3	3			
2	2	4	3			
3	1	5	1			
4	1					

(1) (2)

Table V.1.1.1.8. (1). frequent itemsets (*o*) and (*p*) are associated with pairs (*sid*, *eid*). (2). 2-sequence *{op}* are found by joining itemsets (*o*), (*p*) in (1)

SPADE reduces frequency of global database access because candidate sequence generation is based on local vertical format data set in which itemsets and subsequence 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 sub-section V.1.1.2.

V.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 reducing the number of elements is idea of **FreeSpan** (**Frequent pattern-projected Sequential pattern**) algorithm (Han, Pei, Mortazavi-Asl, Chen, Dayal, & Hsu, 2000).

Given a sequence $s = \langle s_1, s_2, \dots, s_k \rangle$, 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 items $f_list = \{x_1, x_2, \dots, x_n\}$. 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 V.1.1, 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:

$$f_list = \{o:4, f:3, d:3, p:2, s:1, c:1, l:1, v:1\}$$

Let min_sup be 2, applying FreeSpan for o,f -projected databases, we have results shown in table V.1.1.2.1, as follows:

	Sequences	Frequent sequences (mined by frequent item matrix, GPS, AprioriAll, SPADE, etc.)
o -projected database	$\langle o \rangle, \langle o \rangle, \langle o \rangle, \langle o \rangle$	$\langle o \rangle:4$
f -projected database	$\langle o \rangle, \langle of \rangle, \langle of \rangle, \langle of \rangle$	$\langle f \rangle:3, \langle of \rangle:3$

Table V.1.1.2.1. 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 V.1.1.2 ends up the brief survey of sequential pattern mining. Next sub-section V.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.

V.1.2. A proposal of breaking sequential pattern in learning context

Given sequence database (shown in table V.1.1) managed by learning history sub-model, suppose we discovered the sequential pattern $\langle ofp \rangle$ which means:

$$\text{"class \& OOP"} \rightarrow \text{"interface"} \rightarrow \text{"package"}$$

and some other patterns as results from aforementioned mining algorithms such as [AprioriAll](#), [GSP](#), and [SPADE](#). Note that term “class & OOP” is abbreviation of “class and object-oriented programming”. Please see AprioriAll algorithm in sub-section V.1.1.1 to know how to extract the sequential pattern $\langle ofp \rangle$ in detailed. We accept that these patterns like learning “routes” that student preferred or learned often in past but in the next time if a student chooses the concept “class & OOP”, the adaptive learning system should recommend which next concepts in above patterns? So the patterns will

be broken into association rules with their confidences. For example, breaking above pattern $\langle ofp \rangle$ follows two steps:

1. Breaking entire $\langle ofp \rangle$ into itemsets such as o, f, p 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 three large 2-sequences: $\langle of \rangle, \langle op \rangle, \langle fp \rangle$ broken from the pattern $\langle ofp \rangle$. Thus, we have three rules derived from these large 2-sequences in form: “left-hand itemset \rightarrow right-hand itemset”, for example, rule “ $o \rightarrow f$ ” derived from 2-sequence $\langle of \rangle$.

In general, given sequential pattern $\langle xy(zt)u \rangle$ having $n=4$ itemsets such as $x, y, (zt),$ and u is broken into $\binom{n}{2} = \binom{4}{2} = \frac{4!}{2!2!} = 6$ large 2-sequences: $\langle xy \rangle, \langle x(zt) \rangle, \langle xu \rangle, \langle y(zt) \rangle, \langle yu \rangle,$ and $\langle (zt)u \rangle.$ Thus there are 6 rules derived from 6 large 2-sequences: $x \rightarrow y, x \rightarrow (zt), x \rightarrow u, y \rightarrow (zt), y \rightarrow u, (zt) \rightarrow u.$

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 to the support of left-hand itemset. For example, confidences of large 2-sequences: $x \rightarrow y, x \rightarrow (zt), x \rightarrow u, y \rightarrow (zt), y \rightarrow u, (zt) \rightarrow u$ are:

$$\begin{aligned} \text{confidence}(x \rightarrow y) &= \frac{\text{support}(\langle xy \rangle)}{\text{support}(x)} \\ \text{confidence}(x \rightarrow (zt)) &= \frac{\text{support}(\langle x(zt) \rangle)}{\text{support}(x)} \\ \text{confidence}(x \rightarrow u) &= \frac{\text{support}(\langle xu \rangle)}{\text{support}(x)} \\ \text{confidence}(y \rightarrow (zt)) &= \frac{\text{support}(\langle y(zt) \rangle)}{\text{support}(y)} \\ \text{confidence}(y \rightarrow u) &= \frac{\text{support}(\langle yu \rangle)}{\text{support}(y)} \\ \text{confidence}((zt) \rightarrow u) &= \frac{\text{support}(\langle (zt)u \rangle)}{\text{support}((zt))} \end{aligned}$$

Strong rules are defined as rules whose confidences are greater than or equal to a pre-defined threshold $\text{min_conf}.$ Strong rules are also called **sequential rules** and their confidences are known as sequential confidences. Only sequential rules are used for learning concept recommendation. Given $\text{min_conf}=0.5,$ table V.1.2.1 shows sequential rules extracted from the sequential pattern $\langle ofp \rangle.$

sequential rule	sequential confidence
$o \rightarrow f$	$3 / 4 = 0.75$
$o \rightarrow p$	$2 / 4 = 0.50$
$f \rightarrow p$	$2 / 3 = 0.66$

Table V.1.2.1. Sequential rules

You can ask how to determine confidences of these sequential rules, as seen in table V.1.2.1 because it is necessary to calculate supports of three large 2-sequences: $\langle of \rangle, \langle op \rangle, \langle fp \rangle$ and three itemsets: $o, f, p.$ Please see tables V.1.1.1.1 and V.1.1.1.5 listing these supports that were calculated in large

itemset phrase and sequence phrase of AprioriAll algorithm. For example, $\text{confidence}(o \rightarrow f) = \text{support}(\langle of \rangle) / \text{support}(o) = 3 / 4 = 0.75$.

Now, if student choose the concept (itemset) x , system will find whole sequential rules broken from all sequential patterns and the *left-hand litemset* of each sequential rule must contain x . Then, these sequential rules are sorted by their confidences in descending order, like table V.1.2.1. Final outcome is an ordered list of *right-hand litemsets* (concepts) of sequential rules, which are recommended to students. The top concept (litemset) in such list is referred as the most necessary concept. For example, according to sequential rules (shown in table V.1.2.1) resulted from the proposed method, if a student choose concept “*class & OOP*” denoted o , we recommend firstly her/his another concept “*interface*” denoted f which is right-hand litemset of the sequential rule $o \rightarrow f$ when such rule is the one whose left-hand litemset is o and gains highest confidence (0.75). Table V.1.2.2 includes sequential rules whose left-hand litemset is o and recommended concepts. This is the outcome of the proposed method that breaks sequential pattern into adaptive learning concepts which are recommended to students.

sequential rule	confidence	recommended concept
$o \rightarrow f$	0.75	“ <i>interface</i> ”
$o \rightarrow p$	0.50	“ <i>package</i> ”

Table V.1.2.2. 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 V.1.3 is the evaluation of proposed method of breaking sequential patterns.

V.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 [V.2](#), which is also supported by ME and learning history sub-model.

V.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.2.1.1](#) when TLM consists of knowledge sub-model (see chapter [III](#)), learning style sub-model (see chapter [IV](#)) and learning history sub-model (in this chapter [V](#)). However, user interest is a feature of extended TLM and managed by learning history sub-model, as shown in figure [II.2.1.2](#). Similar to learning concept recommendation mentioned in previous section [V.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 user modeling system Zebra. As aforementioned at [the beginning of this chapter V](#), 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 L., 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's accesses 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 [V.2.1](#).

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) in order to choose which classes are most suitable to these interesting documents. *Such classes are user interests.*

Table V.2.1. 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 V.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 V.2.2, according to step 2 (Nguyen L., Discovering User Interests by Document Classification, 2010). The main approach to discover user interest is described in sub-section V.2.3, according to steps 3 and 4. Sub-section V.2.4 is the evaluation.

V.2.1. Vector model for representing documents

The main purpose of this sub-section V.2.1 is to introduce vector model of document according to step 1 of proposed approach to discover user interests as described in table V.2.1. 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 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 V.2.1.1 as follows:

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{ik}}$$

Formula V.2.1.1. 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 in which t_j occurs. The inverse document frequencies exist if all documents are indexed. Formula V.2.1.2 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 V.2.1.2. 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 V.2.1.3 as follows:

$$w_{ij} = tf_{ij} * idf_j$$

Formula V.2.1.3. 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 V.2.1.4 expresses the document vector which is the result of step 1 of proposed approach for discovering user interests (see table V.2.1).

$$D_i = (w_{i1}, w_{i2}, w_{i3}, \dots, w_{ip})$$

Formula V.2.1.4. Vector model of document D_i

Where the weights $w_{ij} = tf_{ij} * idf_j$ specified by formula V.2.1.3.

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 V.2.1.5 specifies such brief representation of document vector.

$$D_i = (tf_{i1}, tf_{i2}, tf_{i3}, \dots, tf_{ip})$$

Formula V.2.1.5. 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 V.2.2.

V.2.2. Methods of document classification

This sub-section V.2.2 describes three supervised learning methods such as support vector machine (sub-section V.2.2.1), decision tree (sub-section V.2.2.2), neural network (sub-section V.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 V.2.1. The sub-section V.2.2.1 is also published as a tutorial in (Nguyen L. , Tutorial on Support Vector Machine, 2016).

V.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 by one class. There is the

condition for this separating hyperplane: “it must maximize the margin between two sub-sets”. Figure V.2.2.1.1 (Wikibooks, 2008) shows separating hyperplanes H_1 , H_2 , and H_3 in which only H_2 gets maximum margin according to this condition.

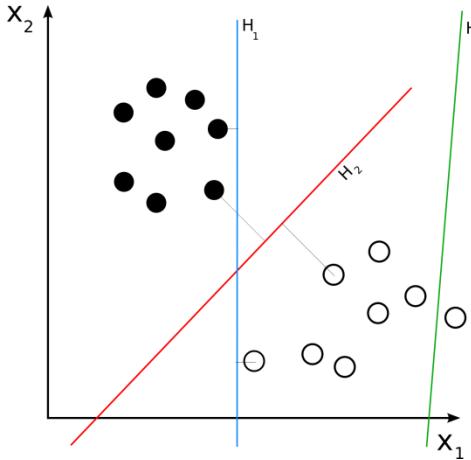


Figure V.2.2.1.1. Separating hyperplanes

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 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 V.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 V.2.2.1.1.

$$W \circ X_i - b = 0$$

Formula V.2.2.1.1. Hyperplane equation

Where the sign “ \circ ” 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 $W \circ X_i$ is:

$$W \circ X_i = X_i \circ W = w_1 x_{i1} + w_2 x_{i2} + \dots + w_p x_{ip} = \sum_{j=1}^p w_j x_{ij}$$

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 $W \circ X_i$ 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 V.2.2.1.1 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 .

$$|W| = \|W\| = \sqrt{W \circ W} = \sqrt{w_1^2 + w_2^2 + \dots + w_p^2} = \sqrt{\sum_{i=1}^p w_i^2}$$

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 V.2.2.1.2. 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 V.2.2.1.2 as follows.

$$\begin{aligned} W \circ X_i - b &= 1 \\ W \circ X_i - b &= -1 \end{aligned}$$

Formula V.2.2.1.2. Equations of two parallel hyperplanes

Figure V.2.2.1.2 (Wikibooks, 2008) illustrates maximum-margin hyperplane, weight vector W and two parallel hyperplanes. As seen in the figure V.2.2.1.2, 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. You can imagine such margin as a road and SVM method aims to maximize the width of such road. Data points lying on (or are very near to) two parallel hyperplanes are called support vectors because they construct mainly the maximum-margin hyperplane in the middle. This is the reason that the classification method is called support vector machine (SVM).

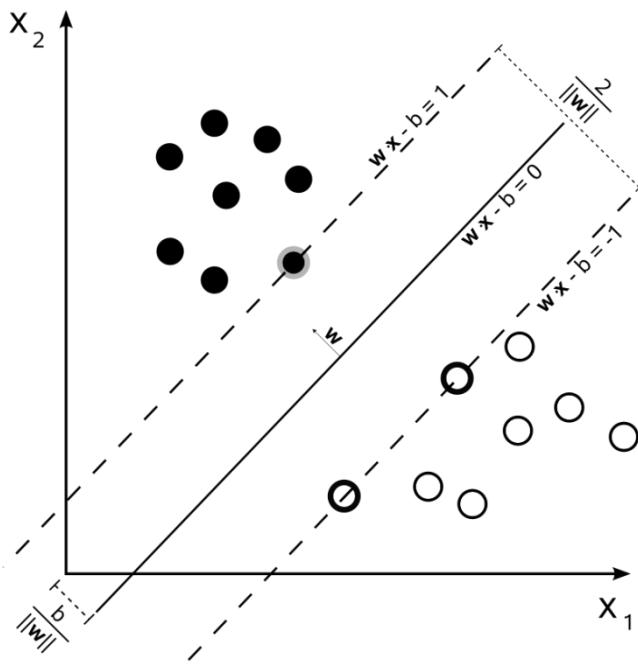


Figure V.2.2.1.2. Maximum-margin hyperplane, parallel hyperplanes and weight vector W

To prevent vectors from falling into the margin, all vectors belonging to two classes $y_i=1$ and $y_i=-1$ have two following constraints, respectively:

$$\begin{cases} W \circ X_i - b \geq 1 & (\text{for } X_i \text{ belonging to class } y_i = +1) \\ W \circ X_i - b \leq -1 & (\text{for } X_i \text{ belonging to class } y_i = -1) \end{cases}$$

As seen in figure V.2.2.1.2, vectors (data points) belonging to classes $y_i=+1$ and $y_i=-1$ are depicted as black circles and white circles, respectively. Such two constraints are unified into the so-called classification constraint specified by formula V.2.2.1.3 as follows:

$$y_i(W \circ X_i - b) \geq 1 \Leftrightarrow 1 - y_i(W \circ X_i - b) \leq 0$$

Formula V.2.2.1.3. 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:

$$\underset{W,b}{\text{minimize}} \frac{1}{2} |W|^2 \text{ subject to } y_i(W \circ X_i - b) \geq 1, \forall i = \overline{1,n}$$

Where $|W|$ is the length of weight vector W and $y_i(W \circ X_i - b) \geq 1$ is the classification constraint specified by formula V.2.2.1.3. 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 $W \circ W$ as follows:

$$|W|^2 = \|W\|^2 = W \circ W = w_1^2 + w_2^2 + \dots + w_p^2$$

The constrained optimization problem is re-written, shown in formula V.2.2.1.4 as below:

$$\begin{cases} \underset{W,b}{\text{minimize}} f(W) = \underset{W,b}{\text{minimize}} \frac{1}{2} |W|^2 \\ \text{subject to } g_i(W, b) = 1 - y_i(W \circ X_i - b) \leq 0, \forall i = \overline{1,n} \end{cases}$$

Formula V.2.2.1.4. Constrained optimization problem for constructing maximum-margin hyperplane

Where $f(W) = \frac{1}{2}|W|^2$ is called target function with regard to variable W . Function $g_i(W, b) = 1 - y_i(W \circ X_i - b)$ is called constraint function with regard to two variables W, b and it is derived from the classification constraint specified by formula V.2.2.1.3. There are n constraints $g_i(W, b) \leq 0$ because training set $\{X_1, X_2, \dots, X_n\}$

has n data points X_i (s). Constraints $g_i(W, b) \leq 0$ inside formula V.2.2.1.3 implicate the perfect separation in which there is no data point falling into the margin (between two parallel hyperplanes, see figure V.2.2.1.2). On the other hand, the imperfect separation allows some data points to fall into the margin, which means that each constraint function $g_i(W, b)$ is subtracted by an error $\xi_i \geq 0$. The constraints become (Honavar, p. 5):

$$g_i(W, b) = 1 - y_i(W \circ X_i - b) - \xi_i \leq 0, \forall i = \overline{1, n}$$

We have a n -component error vector $\xi = (\xi_1, \xi_2, \dots, \xi_n)$ for n constraints. The penalty $C \geq 0$ is added to the target function in order to penalize data points falling into the margin. The penalty C is a pre-defined constant. Thus, the target function $f(W)$ becomes:

$$f(W) = \frac{1}{2} |W|^2 + C \sum_{i=1}^n \xi_i$$

If the positive penalty is infinity, $C = +\infty$ then, target function $f(W) = \frac{1}{2} |W|^2 + C \sum_{i=1}^n \xi_i$ may get maximal when all errors ξ_i must be 0, which leads to the perfect separation specified by aforementioned formula V.2.2.1.4.

Formula V.2.2.1.5 specifies the general form of constrained optimization originated from formula V.2.2.1.4.

$$\begin{cases} \underset{W, b, \xi}{\text{minimize}} \frac{1}{2} |W|^2 + C \sum_{i=1}^n \xi_i \\ \text{subject to } \begin{aligned} & 1 - y_i(W \circ X_i - b) - \xi_i \leq 0, \forall i = \overline{1, n} \\ & -\xi_i \leq 0, \forall i = \overline{1, n} \end{aligned} \end{cases}$$

Formula V.2.2.1.5. General SVM constrained optimization problem

Where $C \geq 0$ is the penalty.

The *Lagrangian function* (Boyd & Vandenberghe, 2009, p. 215) is constructed from constrained optimization problem specified by formula V.2.2.1.5. Let $L(W, b, \xi, \lambda, \mu)$ be Lagrangian function where $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ and $\mu = (\mu_1, \mu_2, \dots, \mu_n)$ are n -component vectors, $\lambda_i \geq 0$ and $\mu_i \geq 0, \forall i = \overline{1, n}$. We have:

$$\begin{aligned} L(W, b, \xi, \lambda, \mu) &= f(W) + \sum_{i=1}^n \lambda_i g_i(W, b) - \mu_i \xi_i \\ &= \frac{1}{2} |W|^2 + C \sum_{i=1}^n \xi_i + \sum_{i=1}^n \lambda_i (1 - y_i(W \circ X_i - b) - \xi_i) - \sum_{i=1}^n \mu_i \xi_i \\ &= \frac{1}{2} |W|^2 + C \sum_{i=1}^n \xi_i + \sum_{i=1}^n \lambda_i - \sum_{i=1}^n \lambda_i y_i (W \circ X_i) + \sum_{i=1}^n b \lambda_i y_i - \sum_{i=1}^n \lambda_i \xi_i - \sum_{i=1}^n \mu_i \xi_i \\ &= \frac{1}{2} |W|^2 - W \circ \left(\sum_{i=1}^n \lambda_i y_i X_i \right) + \sum_{i=1}^n \lambda_i + b \sum_{i=1}^n \lambda_i y_i + \sum_{i=1}^n (C - \lambda_i - \mu_i) \xi_i \end{aligned}$$

In general, formula V.2.2.1.6 represents Lagrangian function as follows:

$$\begin{aligned}
 L(W, b, \xi, \lambda, \mu) &= \frac{1}{2} |W|^2 - W \circ \left(\sum_{i=1}^n \lambda_i y_i X_i \right) + \sum_{i=1}^n \lambda_i + b \sum_{i=1}^n \lambda_i y_i \\
 &\quad + \sum_{i=1}^n (C + \lambda_i - \mu_i) \xi_i
 \end{aligned}$$

Where $\xi_i \geq 0, \lambda_i \geq 0, \mu_i \geq 0, \forall i = \overline{1, n}$

Formula V.2.2.1.6. Lagrangian function

Note that $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ and $\mu = (\mu_1, \mu_2, \dots, \mu_n)$ are called Lagrange multipliers or Karush-Kuhn-Tucker multipliers (Wikipedia, Karush–Kuhn–Tucker conditions, 2014) or dual variables. The sign “ \circ ” denotes scalar product and every training data point X_i was assigned by a class y_i before.

Suppose (W^*, b^*) is solution of constrained optimization problem specified by formula V.2.2.1.5 then, the pair (W^*, b^*) is minimum point of target function $f(W)$ or target function $f(W)$ gets minimum at (W^*, b^*) with all constraints $g_i(W, b) = 1 - y_i(W \circ X_i - b) + \xi_i \leq 0, \forall i = \overline{1, n}$. 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:

$$\begin{aligned}
 (W^*, b^*) &= \underset{W, b}{\operatorname{argmin}} L(W, b, \xi, \lambda, \mu) \\
 \lambda^* &= \underset{\lambda \geq 0}{\operatorname{argmax}} \left(\min_{W, b} L(W, b, \lambda, \mu) \right)
 \end{aligned}$$

Formula V.2.2.1.7. Lagrangian duality problem

Where Lagrangian function $L(W, b, \xi, \lambda, \mu)$ is specified by formula V.2.2.1.6.

Now it is necessary to solve the Lagrangian duality problem represented by formula V.2.2.1.7 to find out W^* . Thus, the Lagrangian function $L(W, b, \xi, \lambda, \mu)$ is minimized with respect to the primal variables W, b and maximized with respect to the dual variables $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ and $\mu = (\mu_1, \mu_2, \dots, \mu_n)$, in turn. If gradient of $L(W, b, \xi, \lambda, \mu)$ is equal to zero then, $L(W, b, \xi, \lambda, \mu)$ 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 $L(W, b, \xi, \lambda, \mu)$ with respect to W, b , and ξ to zero, we have:

$$\begin{cases} \frac{\partial L(W, b, \xi, \lambda, \mu)}{\partial W} = 0 \\ \frac{\partial L(W, b, \xi, \lambda, \mu)}{\partial b} = 0 \\ \frac{\partial L(W, b, \xi, \lambda, \mu)}{\partial \xi_i} = 0, \forall i = \overline{1, n} \end{cases} \Leftrightarrow \begin{cases} W - \sum_{i=1}^n \lambda_i y_i X_i = 0 \\ \sum_{i=1}^n \lambda_i y_i = 0 \\ C - \lambda_i - \mu_i = 0, \forall i = \overline{1, n} \end{cases}$$

$$\Rightarrow \begin{cases} W = \sum_{i=1}^n \lambda_i y_i X_i \\ \sum_{i=1}^n \lambda_i y_i = 0 \\ \lambda_i = C - \mu_i, \forall i = \overline{1, n} \end{cases}$$

In general, W^* is determined by formula V.2.2.1.8 as follows:

$$\begin{cases} W^* = \sum_{i=1}^n \lambda_i y_i X_i \\ \sum_{i=1}^n \lambda_i y_i = 0 \\ \lambda_i = C - \mu_i, \lambda_i \geq 0, \mu_i \geq 0, \forall i = \overline{1, n} \end{cases}$$

Formula V.2.2.1.8. 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 V.2.2.1.8 into Lagrangian function $L(W, b, \xi, \lambda, \mu)$ specified by formula V.2.2.1.6, we have:

$$\begin{aligned} l(\lambda) &= \min_{W,b} L(W, b, \xi, \lambda, \mu) \\ &= \min_{W,b} \left(\frac{1}{2} |W|^2 - W \circ \left(\sum_{i=1}^n \lambda_i y_i X_i \right) + \sum_{i=1}^n \lambda_i + b \sum_{i=1}^n \lambda_i y_i + \sum_{i=1}^n (C + \lambda_i - \mu_i) \xi_i \right) \\ &= \frac{1}{2} \left(\sum_{i=1}^n \lambda_i y_i X_i \right)^2 - \left(\sum_{i=1}^n \lambda_i y_i X_i \right) \circ \left(\sum_{i=1}^n \lambda_i y_i X_i \right) + \sum_{i=1}^n \lambda_i \end{aligned}$$

(according to formula V.2.2.1.8, $L(W, b, \xi, \lambda, \mu)$ gets minimum at $W = \sum_{i=1}^n \lambda_i y_i X_i$ and

$$\begin{aligned} &\sum_{i=1}^n \lambda_i y_i = 0 \text{ and } \lambda_i = C - \mu_i \\ &= \frac{1}{2} \left(\sum_{i=1}^n \lambda_i y_i X_i \right) \circ \left(\sum_{i=1}^n \lambda_i y_i X_i \right) - \left(\sum_{i=1}^n \lambda_i y_i X_i \right) \circ \left(\sum_{i=1}^n \lambda_i y_i X_i \right) + \sum_{i=1}^n \lambda_i \\ &= -\frac{1}{2} \left(\sum_{i=1}^n \lambda_i y_i X_i \right) \circ \left(\sum_{i=1}^n \lambda_i y_i X_i \right) + \sum_{i=1}^n \lambda_i \\ &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (X_i \circ X_j) + \sum_{i=1}^n \lambda_i \end{aligned}$$

Where $l(\lambda)$ is called *dual function* represented by formula V.2.2.1.9.

$$l(\lambda) = \min_{W,b} L(W, b, \xi, \lambda, \mu) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (X_i \circ X_j) + \sum_{i=1}^n \lambda_i$$

Formula V.2.2.1.9. Dual function for determining Lagrange multipliers

According to Lagrangian duality problem represented by formula V.2.2.1.7, $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ is calculated as the maximum point $\lambda^* = (\lambda_1^*, \lambda_2^*, \dots, \lambda_n^*)$ of dual function $l(\lambda)$. In conclusion, maximizing $l(\lambda)$ is the main task of SVM method because the optimal weight vector W^* is calculated based on the optimal point λ^* of dual function $l(\lambda)$ according to formula V.2.2.1.8.

$$W^* = \sum_{i=1}^n \lambda_i y_i X_i = \sum_{i=1}^n \lambda_i^* y_i X_i$$

Maximizing $l(\lambda)$ is quadratic programming (QP) problem, specified by formula V.2.2.1.10.

$$\begin{cases} \text{maximize}_{\lambda} -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (X_i \circ X_j) + \sum_{i=1}^n \lambda_i \\ \text{subject to } \sum_{i=1}^n \lambda_i y_i = 0 \\ 0 \leq \lambda_i \leq C, \forall i = \overline{1, n} \end{cases}$$

Formula V.2.2.1.10. Quadratic programming of support vector machine

The constraints $0 \leq \lambda_i \leq C, \forall i = \overline{1, n}$ are implied from the equations $\lambda_i = C - \mu_i, \forall i = \overline{1, n}$ when $\mu_i \geq 0, \forall i = \overline{1, n}$. The QP problem specified by formula V.2.2.1.10 is also known as Wolfe problem (Honavar, p. 42).

There are some methods to solve this QP problem but this report introduces a so-called Sequential Minimal Optimization (SMO) developed by author Platt (Platt, 1998). The SMO algorithm is very effective method to find out the optimal (maximum) point λ^* of dual function $l(\lambda)$.

$$l(\lambda) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (X_i \circ X_j) + \sum_{i=1}^n \lambda_i$$

Moreover SMO algorithm also finds out the optimal bias b^* , which means that SVM classifier (W^*, b^*) is totally determined by SMO algorithm.

The ideology of SMO algorithm is to divide the whole QP problem into many smallest optimization problems. Each smallest problem relates to only two Lagrange multipliers. For solving each smallest optimization problem, SMO algorithm includes two nested loops as shown in table V.2.2.1.1 (Platt, 1998, pp. 8-9):

SMO algorithm solves each smallest optimization problem via two nested loops:

- The outer loop finds out the first Lagrange multiplier λ_i whose associated data point X_i violates KKT condition (Wikipedia, Karush–Kuhn–Tucker conditions, 2014). Violating KKT condition is known as the first choice heuristic.
- The inner loop finds out the second Lagrange multiplier λ_j according to the second choice heuristic. The second choice heuristic that maximizes optimization step will be described later.

- Two Lagrange multipliers λ_i and λ_j are optimized jointly according to QP problem specified by formula V.2.2.1.10.

SMO algorithm continues to solve another smallest optimization problem. SMO algorithm stops when there is convergence in which no data point violating KKT condition is found; consequently, all Lagrange multipliers $\lambda_1, \lambda_2, \dots, \lambda_n$ are optimized.

Table V.2.2.1.1. Ideology of SMO algorithm

Before describing SMO algorithm in detailed, the KKT condition with subject to SVM is mentioned firstly because violating KKT condition is known as the first choice heuristic of SMO algorithm. KKT condition indicates both partial derivatives of Lagrangian function and complementary slackness are zero (Wikipedia, Karush–Kuhn–Tucker conditions, 2014). Referring formulas V.2.2.1.8 and V.2.2.1.4, the KKT function of SVM is summarized as formula V.2.2.1.11:

$$\begin{cases} W = \sum_{i=1}^n \lambda_i y_i X_i \\ \sum_{i=1}^n \lambda_i y_i = 0 \\ \lambda_i = C - \mu_i, \lambda_i \geq 0, \mu_i \geq 0, \forall i \\ \lambda_i(1 - y_i(W \circ X_i - b) - \xi_i) = 0, \forall i \\ -\mu_i \xi_i = 0, \forall i \end{cases}$$

Formula V.2.2.1.11. KKT condition of SVM

When we understand deeply convex optimization, the KKT condition is the same to the QP problem specified by formula V.2.2.1.10 if target function and constraint sets are convex. Thus, the solution (W^*, λ^*) is saddle point of Lagrangian function.

KKT condition is analyzed into three following cases (Honavar, p. 7):

1. If $\lambda_i=0$ then, $\mu_i = C - \lambda_i = C$. It implies $\xi_i=0$ from equation $\mu_i \xi_i = 0$. Then, from equation $\lambda_i(1 - y_i(W \circ X_i - b) - \xi_i) = 0$ we have:

$$1 - y_i(W \circ X_i - b) \leq 0$$

2. If $0 < \lambda_i < C$ then, we have $1 - y_i(W \circ X_i - b) - \xi_i = 0$. Due to $\mu_i = C - \lambda_i > 0$, it implies $\xi_i=0$ from equation $\mu_i \xi_i = 0$. It is easy to infer that:

$$1 - y_i(W \circ X_i - b) = 0$$

3. If $\lambda_i=C$ then, we have $\mu_i = C - \lambda_i = 0$ and $1 - y_i(W \circ X_i - b) - \xi_i = 0$. Due to $\mu_i = 0$, it implies $\xi_i \geq 0$ from equation $\mu_i \xi_i = 0$. Given $\xi_i \geq 0$ the equation $1 - y_i(W \circ X_i - b) - \xi_i = 0$ leads to:

$$1 - y_i(W \circ X_i - b) \geq 0$$

Let $E_i = y_i - (W \circ X_i - b)$ be prediction error, we have:

$$y_i E_i = (y_i)^2 - y_i(W \circ X_i - b) = 1 - y_i(W \circ X_i - b)$$

The KKT condition implies:

$$\begin{aligned} \lambda_i = 0 &\Rightarrow y_i E_i \leq 0 \\ 0 < \lambda_i < C &\Rightarrow y_i E_i = 0 \\ \lambda_i = C &\Rightarrow y_i E_i \geq 0 \end{aligned}$$

Where E_i is prediction error:

$$E_i = y_i - (W \circ X_i - b)$$

Formula V.2.2.1.12. KKT corollaries of SVM

Formula V.2.2.1.12 expresses directed corollaries from KKT condition. It is commented on formula V.2.2.1.12 that if $E_i=0$, the KKT condition is always satisfied. Data points X_i satisfying equation $y_iE_i=0$ lie on the margin (lie on the two parallel hyperplanes). These points are called *support vectors*. According to KKT corollary, support vectors are always associated with non-zero Lagrange multipliers such that $0 < \lambda_i < C$. Note, such Lagrange multipliers $0 < \lambda_i < C$ are also called non-boundary multipliers because they are not bounds such as 0 and C . So, support vectors are also known as *non-boundary* data points. It is easy to infer from formula V.2.2.1.8

$$W^* = \sum_{i=1}^n \lambda_i y_i X_i$$

that support vectors along with their non-zero Lagrange multipliers form mainly the optimal weight vector W^* representing the maximum-margin hyperplane – the SVM classifier. This is the reason that this classification approach is called support vector machine (SVM). Figure V.2.2.1.3 (Moore, 2001, p. 5) illustrates an example of support vectors.

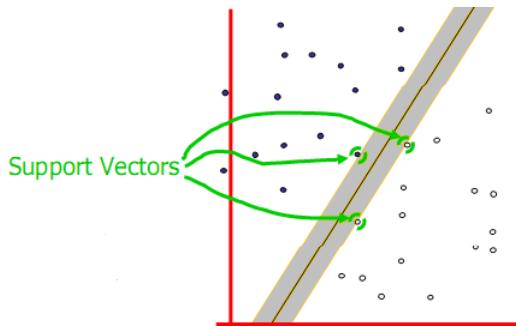


Figure V.2.2.1.3. Support vectors

Violating KKT condition is the first choice heuristic of SMO algorithm. By negating three corollaries specified by formula V.2.2.1.12, KKT condition is violated in three following cases:

$$\begin{aligned} \lambda_i &= 0 && \text{and } y_i E_i > 0 \\ 0 &< \lambda_i < C && \text{and } y_i E_i \neq 0 \\ \lambda_i &= C && \text{and } y_i E_i < 0 \end{aligned}$$

By logic induction, these cases are reduced into two cases specified by formula V.2.2.1.13.

$$\begin{aligned} \lambda_i &< C && \text{and } y_i E_i > 0 \\ \lambda_i &> 0 && \text{and } y_i E_i < 0 \end{aligned}$$

Where E_i is prediction error:

$$E_i = y_i - (W \circ X_i - b)$$

Formula V.2.2.1.13. Two cases of KKT condition violation

Formula V.2.2.1.13 is used to check whether given data point X_i violates KKT condition.

The main task of SMO algorithm (see table V.2.2.1.1) is to optimize jointly two Lagrange multipliers in order to solve each smallest optimization problem, which maximizes the dual function $l(\lambda)$.

$$l(\lambda) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (X_i \circ X_j) + \sum_{i=1}^n \lambda_i$$

Where,

$$\sum_{i=1}^n \lambda_i y_i = 0$$

$$0 \leq \lambda_i \leq C, \forall i = \overline{1, n}$$

Without loss of generality, two Lagrange multipliers λ_i and λ_j that will be optimized are λ_1 and λ_2 while all other multipliers $\lambda_3, \lambda_4, \dots, \lambda_n$ are fixed. Old values of λ_1 and λ_2 are denoted λ_1^{old} and λ_2^{old} . Your attention please, old values are known as current values. Thus, λ_1 and λ_2 are optimized based on the set: $\lambda_1^{\text{old}}, \lambda_2^{\text{old}}, \lambda_2, \lambda_3, \dots, \lambda_n$. The old values λ_1^{old} and λ_2^{old} are initialized by zero (Honavar, p. 9). From the condition $\sum_{i=1}^n \lambda_i y_i = 0$, we have:

$$\lambda_1^{\text{old}} y_1 + \lambda_2^{\text{old}} y_2 + \lambda_3 y_3 + \lambda_4 y_4 + \dots + \lambda_n y_n = 0$$

and

$$\lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 + \lambda_4 y_4 + \dots + \lambda_n y_n = 0$$

It implies following equation of line with regard to two variables λ_1 and λ_2 :

$$\lambda_1 y_1 + \lambda_2 y_2 = \lambda_1^{\text{old}} y_1 + \lambda_2^{\text{old}} y_2$$

Formula V.2.2.1.14. Linear constraint of two Lagrange multipliers

Formula V.2.2.1.14 specifies the linear constraint of two Lagrange multipliers λ_1 and λ_2 . This constraint is drawn as diagonal lines in figure V.2.2.1.4 (Honavar, p. 9).

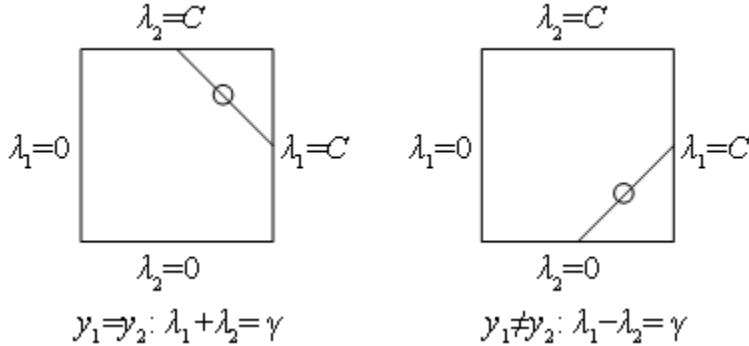


Figure V.2.2.1.4. Linear constraint of two Lagrange multipliers

In figure V.2.2.1.4, the box is bounded by the interval $[0, C]$ of Lagrange multipliers, $0 \leq \lambda_i \leq C$. SMO algorithm moves λ_1 and λ_2 along diagonal lines so as to maximize the dual function $l(\lambda)$. Multiplying two sides of equation $\lambda_1 y_1 + \lambda_2 y_2 = \lambda_1^{\text{old}} y_1 + \lambda_2^{\text{old}} y_2$ by y_1 , we have:

$$\begin{aligned} \lambda_1 y_1 y_1 + \lambda_2 y_1 y_2 &= \lambda_1^{\text{old}} y_1 y_1 + \lambda_2^{\text{old}} y_1 y_2 \\ \Rightarrow \lambda_1 + s\lambda_2 &= \lambda_1^{\text{old}} + s\lambda_2^{\text{old}} \end{aligned}$$

Where $s=y_1 y_2$. Let,

$$\gamma = \lambda_1 + s\lambda_2 = \lambda_1^{\text{old}} + s\lambda_2^{\text{old}}$$

We have formula V.2.2.1.15 as a variant of the linear constraint of two Lagrange multipliers λ_1 and λ_2 (Honavar, p. 9):

$$\lambda_1 = \gamma - s\lambda_2$$

Where,

$$s = y_1 y_2$$

$$\gamma = \lambda_1 + s\lambda_2 = \lambda_1^{\text{old}} + s\lambda_2^{\text{old}}$$

Formula V.2.2.1.15. A variant of linear constraint of two Lagrange multipliers

By fixing multipliers $\lambda_3, \lambda_4, \dots, \lambda_n$, all arithmetic combinations of $\lambda_1^{\text{old}}, \lambda_2^{\text{old}}, \lambda_3, \lambda_4, \dots, \lambda_n$ are constants denoted by term “*const*”. The dual function $l(\lambda)$ is re-written (Honavar, pp. 9-11):

$$\begin{aligned} l(\lambda) &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (X_i \circ X_j) + \sum_{i=1}^n \lambda_i \\ &= -\frac{1}{2} \left((\lambda_1)^2 y_1 y_1 (X_1 \circ X_1) + (\lambda_2)^2 y_2 y_2 (X_2 \circ X_2) + 2\lambda_1 \lambda_2 y_1 y_2 (X_1 \circ X_2) \right. \\ &\quad \left. + 2 \left(\sum_{i=3}^n \lambda_i \lambda_1 y_i y_1 (X_i \circ X_1) \right) + 2 \left(\sum_{i=3}^n \lambda_i \lambda_2 y_i y_2 (X_i \circ X_2) \right) + \text{const} \right) \\ &\quad + \lambda_1 + \lambda_2 + \text{const} \\ &= -\frac{1}{2} \left((X_1 \circ X_1)(\lambda_1)^2 + (X_2 \circ X_2)(\lambda_2)^2 + 2s(X_1 \circ X_2)\lambda_1 \lambda_2 \right. \\ &\quad \left. + 2 \left(\sum_{i=3}^n \lambda_i \lambda_1 y_i y_1 (X_i \circ X_1) \right) + 2 \left(\sum_{i=3}^n \lambda_i \lambda_2 y_i y_2 (X_i \circ X_2) \right) \right) + \lambda_1 + \lambda_2 \\ &\quad + \text{const} \end{aligned}$$

Let,

$$K_{11} = X_1 \circ X_1$$

$$K_{12} = X_1 \circ X_2$$

$$K_{22} = X_2 \circ X_2$$

Let W^{old} be the optimal weight vector $W = \sum_{i=1}^n \lambda_i y_i X_i$ based on old values of two aforementioned Lagrange multipliers. Following linear constraint of two Lagrange multipliers specified by formula V.2.2.1.14, we have:

$$W^{\text{old}} = \lambda_1^{\text{old}} y_1 X_1 + \lambda_2^{\text{old}} y_2 X_2 + \sum_{i=3}^n \lambda_i y_i X_i = \sum_{i=1}^n \lambda_i y_i X_i = W$$

Let,

$$\begin{aligned} v_j &= \sum_{i=3}^n \lambda_i y_i (X_i \circ X_j) = \sum_{i=3}^n (\lambda_i y_i X_i) \circ X_j = \left(\sum_{i=3}^n \lambda_i y_i X_i \right) \circ X_j \\ &= (W^{\text{old}} - \lambda_1^{\text{old}} y_1 X_1 - \lambda_2^{\text{old}} y_2 X_2) \circ X_j \\ &= W^{\text{old}} \circ X_j - \lambda_1^{\text{old}} y_1 X_1 \circ X_j - \lambda_2^{\text{old}} y_2 X_2 \circ X_j \end{aligned}$$

We have (Honavar, p. 10):

$$\begin{aligned} l(\lambda) &= -\frac{1}{2} (K_{11}(\lambda_1)^2 + K_{22}(\lambda_2)^2 + 2sK_{12}\lambda_1\lambda_2 + 2y_1 v_1 \lambda_1 + 2y_2 v_2 \lambda_2) + \lambda_1 + \lambda_2 \\ &\quad + \text{const} \\ &= -\frac{1}{2} (K_{11}(\gamma - s\lambda_2)^2 + K_{22}(\lambda_2)^2 + 2sK_{12}(\gamma - s\lambda_2)\lambda_2 + 2y_1 v_1 (\gamma - s\lambda_2) \\ &\quad + 2y_2 v_2 \lambda_2) + (\gamma - s\lambda_2) + \lambda_2 + \text{const} \end{aligned}$$

$$\begin{aligned}
 &= -\frac{1}{2}(K_{11}\gamma^2 - 2sK_{11}\gamma\lambda_2 + K_{11}(\lambda_2)^2 + K_{22}(\lambda_2)^2 + 2sK_{12}\gamma\lambda_2 - 2K_{12}(\lambda_2)^2 \\
 &\quad + 2y_1v_1\gamma - 2sy_1v_1\lambda_2 + 2y_2v_2\lambda_2) + (1-s)\lambda_2 + \gamma + const \\
 &= -\frac{1}{2}(K_{11} + K_{22} - 2K_{12})(\lambda_2)^2 + sK_{11}\gamma\lambda_2 - sK_{12}\gamma\lambda_2 + sy_1v_1\lambda_2 - y_2v_2\lambda_2 \\
 &\quad + (1-s)\lambda_2 - \frac{1}{2}K_{11}\gamma^2 - y_1v_1\gamma + \gamma + const \\
 &= -\frac{1}{2}(K_{11} + K_{22} - 2K_{12})(\lambda_2)^2 + sK_{11}\gamma\lambda_2 - sK_{12}\gamma\lambda_2 + sy_1v_1\lambda_2 - y_2v_2\lambda_2 \\
 &\quad + (1-s)\lambda_2 + const \\
 &\quad \left(\text{Because } -\frac{1}{2}K_{11}\gamma^2 - y_1v_1\gamma + \gamma \text{ is also constant} \right) \\
 &= -\frac{1}{2}(K_{11} + K_{22} - 2K_{12})(\lambda_2)^2 + (1-s + sK_{11}\gamma - sK_{12}\gamma + sy_1v_1 - y_2v_2)\lambda_2 \\
 &\quad + const \\
 &= -\frac{1}{2}(K_{11} + K_{22} - 2K_{12})(\lambda_2)^2 + (1-s + sK_{11}\gamma - sK_{12}\gamma + y_2v_1 - y_2v_2)\lambda_2 \\
 &\quad + const
 \end{aligned}$$

Let $\eta = K_{11} - 2K_{12} + K_{22}$ and assessing the coefficient of λ_2 , we have (Honavar, p. 11):

$$\begin{aligned}
 &1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2v_1 - y_2v_2 \\
 &= 1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2(v_1 - v_2) \\
 &= 1 - s + sK_{11}\gamma - sK_{12}\gamma \\
 &\quad + y_2(W^{old} \circ X_1 - \lambda_1^{old}y_1X_1 \circ X_1 - \lambda_2^{old}y_2X_2 \circ X_1 - W^{old} \circ X_2 \\
 &\quad + \lambda_1^{old}y_1X_1 \circ X_2 + \lambda_2^{old}y_2X_2 \circ X_2) \\
 &= 1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2(W^{old} \circ X_1 - W^{old} \circ X_2) - \lambda_1^{old}y_1y_2X_1 \circ X_1 \\
 &\quad - \lambda_2^{old}y_2y_2X_2 \circ X_1 + \lambda_1^{old}y_1y_2X_1 \circ X_2 + \lambda_2^{old}y_2y_2X_2 \circ X_2 \\
 &= 1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2(W^{old} \circ X_1 - W^{old} \circ X_2) - \lambda_1^{old}y_1y_2X_1 \circ X_1 \\
 &\quad - \lambda_2^{old}X_2 \circ X_1 + \lambda_1^{old}y_1y_2X_1 \circ X_2 + \lambda_2^{old}X_2 \circ X_2 \\
 &= 1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2(W^{old} \circ X_1 - W^{old} \circ X_2) - sK_{11}\lambda_1^{old} - K_{12}\lambda_2^{old} \\
 &\quad + sK_{12}\lambda_1^{old} + K_{22}\lambda_2^{old} \\
 &= 1 - s + sK_{11}(\lambda_1^{old} + s\lambda_2^{old}) - sK_{12}(\lambda_1^{old} + s\lambda_2^{old}) + y_2(W^{old} \circ X_1 - W^{old} \circ X_2) \\
 &\quad - sK_{11}\lambda_1^{old} - K_{12}\lambda_2^{old} + sK_{12}\lambda_1^{old} + K_{22}\lambda_2^{old} \\
 &= 1 - s + sK_{11}\lambda_1^{old} + K_{11}\lambda_2^{old} - sK_{12}\lambda_1^{old} - K_{12}\lambda_2^{old} + y_2(W^{old} \circ X_1 - W^{old} \circ X_2) \\
 &\quad - sK_{11}\lambda_1^{old} - K_{12}\lambda_2^{old} + sK_{12}\lambda_1^{old} + K_{22}\lambda_2^{old} \\
 &= 1 - s + (sK_{11} - sK_{12} - sK_{11} + sK_{12})\lambda_1^{old} + (K_{11} - K_{12} - K_{12} + K_{22})\lambda_2^{old} \\
 &\quad + y_2(W^{old} \circ X_1 - W^{old} \circ X_2) \\
 &= 1 - s + (K_{11} - 2K_{12} + K_{22})\lambda_2^{old} + y_2(W^{old} \circ X_1 - W^{old} \circ X_2) \\
 &= 1 - s + \eta\lambda_2^{old} + y_2(W^{old} \circ X_1 - W^{old} \circ X_2) \\
 &\quad (\text{due to } \eta = K_{11} - 2K_{12} + K_{22}) \\
 &= 1 - s + \eta\lambda_2^{old} + y_2((y_2 - (W^{old} \circ X_2 - b^{old})) - (y_1 - (W^{old} \circ X_1 - b^{old}))) \\
 &\quad - y_2y_2 + y_1y_2 \\
 &\quad (\text{Where } b^{old} \text{ is the old value of the bias } b) \\
 &= 1 - s + \eta\lambda_2^{old} + y_2((y_2 - (W^{old} \circ X_2 - b^{old})) - (y_1 - (W^{old} \circ X_1 - b^{old}))) \\
 &\quad - 1 + s
 \end{aligned}$$

$$= \eta \lambda_2^{\text{old}} + y_2 \left(\left(y_2 - (W^{\text{old}} \circ X_2 - b^{\text{old}}) \right) - \left(y_1 - (W^{\text{old}} \circ X_1 - b^{\text{old}}) \right) \right)$$

$$= \eta \lambda_2^{\text{old}} + y_2 (E_2^{\text{old}} - E_1^{\text{old}})$$

According to formula V.2.2.1.13, E_2^{old} and E_1^{old} are old prediction errors on X_2 and X_1 , respectively:

$$E_j^{\text{old}} = y_j - (W^{\text{old}} \circ X_j - b^{\text{old}})$$

Recall that we had:

$$l(\lambda) = -\frac{1}{2} (K_{11} + K_{22} - 2K_{12})(\lambda_2)^2 + (1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2 v_1 - y_2 v_2)\lambda_2 + \text{const}$$

Thus, formula V.2.2.1.16 specifies dual function with subject to the second Lagrange multiplier λ_2 that is optimized in conjunction with the first one λ_1 by SMO algorithm.

$$l(\lambda_2) = -\frac{1}{2} \eta (\lambda_2)^2 + (\eta \lambda_2^{\text{old}} + y_2 (E_2^{\text{old}} - E_1^{\text{old}})) \lambda_2 + \text{const}$$

Where,

$$\begin{aligned} \eta &= K_{11} - 2K_{12} + K_{22} = X_1 \circ X_1 - 2X_1 \circ X_2 + X_2 \circ X_2 \\ E_j^{\text{old}} &= y_j - (W^{\text{old}} \circ X_j - b^{\text{old}}) \\ W^{\text{old}} &= \lambda_1^{\text{old}} y_1 X_1 + \lambda_2^{\text{old}} y_2 X_2 + \sum_{i=3}^n \lambda_i y_i X_i \\ \lambda_1 &= \gamma - s\lambda_2 \\ \gamma &= \lambda_1 + s\lambda_2 = \lambda_1^{\text{old}} + s\lambda_2^{\text{old}} \\ s &= y_1 y_2 \end{aligned}$$

Formula V.2.2.1.16. Dual function with regard to λ_2 that is optimized in conjunction with λ_1

The first and second derivatives of dual function $l(\lambda_2)$ with regard to λ_2 are:

$$\begin{aligned} \frac{dl(\lambda_2)}{d\lambda_2} &= -\eta \lambda_2 + \eta \lambda_2^{\text{old}} + y_2 (E_2^{\text{old}} - E_1^{\text{old}}) \\ \frac{d^2 l(\lambda_2)}{d(\lambda_2)^2} &= -\eta \end{aligned}$$

The quantity η is always non-negative due to:

$$\eta = X_1 \circ X_1 - 2X_1 \circ X_2 + X_2 \circ X_2 = (X_1 - X_2) \circ (X_1 - X_2) = |X_1 - X_2|^2 \geq 0$$

Recall that the goal of QP problem is to maximize the dual function $l(\lambda_2)$ so as to find out the optimal multiplier (maximum point) λ_2^* . The second derivative of $l(\lambda_2)$ is always non-negative and so, $l(\lambda_2)$ is concave function and there always exists the maximum point λ_2^* . The function $l(\lambda_2)$ gets maximal if its first derivative is equal to zero:

$$\begin{aligned} \frac{dl(\lambda_2)}{d\lambda_2} &= 0 \\ \Rightarrow -\eta \lambda_2 + \eta \lambda_2^{\text{old}} + y_2 (E_2^{\text{old}} - E_1^{\text{old}}) &= 0 \\ \Rightarrow \lambda_2 &= \lambda_2^{\text{old}} + \frac{y_2 (E_2^{\text{old}} - E_1^{\text{old}})}{\eta} \end{aligned}$$

Therefore, the new values of λ_1 and λ_2 that are solutions of the smallest optimization problem of SMO algorithm are:

$$\lambda_2^* = \lambda_2^{\text{new}} = \lambda_2^{\text{old}} + \frac{y_2 (E_2^{\text{old}} - E_1^{\text{old}})}{\eta}$$

$$\begin{aligned}\lambda_1^* = \lambda_1^{\text{new}} &= \lambda_1^{\text{old}} + s\lambda_2^{\text{old}} - s\lambda_2^{\text{new}} = \lambda_1^{\text{old}} + s\lambda_2^{\text{old}} - s\lambda_2^{\text{old}} - s\frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{\eta} \\ &= \lambda_1^{\text{old}} - s\frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{\eta}\end{aligned}$$

Obviously, λ_1^{new} is totally determined in accordance with λ_2^{new} , thus we should focus on λ_2^{new} . Because multipliers λ_i are bounded, $0 \leq \lambda_i \leq C$, it is required to find out the range of λ_2^{new} . Let L and U be lower bound and upper bound of λ_2^{new} , respectively. We have (Honavar, pp. 11-13):

1. If $s = 1$, then $\lambda_1 + \lambda_2 = \gamma$. There are two sub-cases (see figure V.2.2.1.5) as follows (Honavar, p. 11):
 - If $\gamma \geq C$ then $L = \gamma - C$ and $U = C$.
 - If $\gamma < C$ then $L = 0$ and $U = \gamma$.
2. If $s = -1$, then $\lambda_1 - \lambda_2 = \gamma$. There are two sub-cases (see figure V.2.2.1.6) as follows (Honavar, pp. 11-12):
 - If $\gamma \geq 0$ then $L = 0$ and $U = C - \gamma$.
 - If $\gamma < 0$ then $L = -\gamma$ and $U = C$.

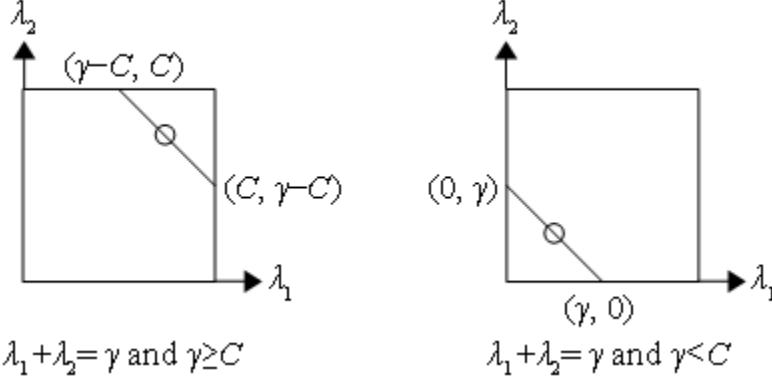


Figure V.2.2.1.5. Lower bound and upper bound of two new multipliers in case $s = 1$

(Honavar, p. 12)

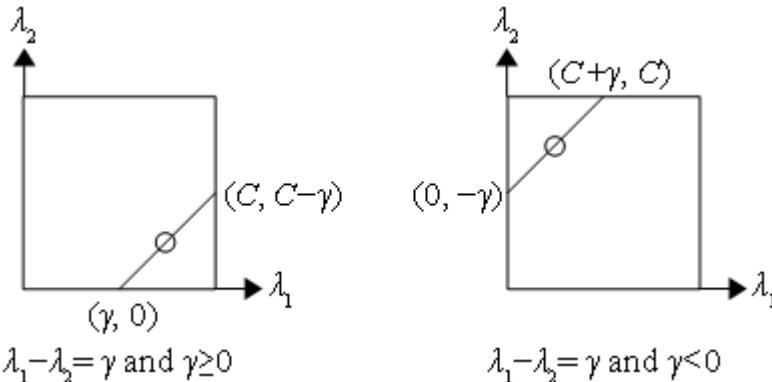


Figure V.2.2.1.6. Lower bound and upper bound of two new multipliers in case $s = -1$

(Honavar, p. 13)

The value λ_2^{new} is clipped as follows (Honavar, p. 12):

$$\lambda_2^{\text{new,clipped}} = \begin{cases} L & \text{if } \lambda_2^{\text{new}} < L \\ \lambda_2^{\text{new}} & \text{if } L \leq \lambda_2^{\text{new}} \leq U \\ U & \text{if } U < \lambda_2^{\text{new}} \end{cases}$$

In the case $\eta=0$ that λ_2^{new} is undetermined, $\lambda_2^{\text{new,clipped}}$ is assigned by which bound (L or U) maximizes the dual function $l(\lambda_2)$.

$$\lambda_2^{\text{new,clipped}} = \underset{\lambda_2}{\operatorname{argmax}} \{l(\lambda_2 = L), l(\lambda_2 = U)\} \text{ if } \eta = 0$$

In general, table V.2.2.1.2 summarizes how SMO algorithm optimizes jointly two Lagrange multipliers.

If $\eta > 0$:

$$\lambda_2^{\text{new}} = \lambda_2^{\text{old}} + \frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{\eta}$$

$$\lambda_2^* = \lambda_2^{\text{new,clipped}} = \begin{cases} L & \text{if } \lambda_2^{\text{new}} < L \\ \lambda_2^{\text{new}} & \text{if } L \leq \lambda_2^{\text{new}} \leq U \\ U & \text{if } U < \lambda_2^{\text{new}} \end{cases}$$

If $\eta = 0$:

$$\lambda_2^* = \lambda_2^{\text{new,clipped}} = \underset{\lambda_2}{\operatorname{argmax}} \{l(\lambda_2 = L), l(\lambda_2 = U)\}$$

Where prediction errors E_j^{old} and dual function $l(\lambda_2)$ are specified by formula V.2.2.1.16. Lower bound L and upper bound U are described as follows:

- If $s=1$ and $\gamma > C$ then $L = \gamma - C$ and $U = C$.
- If $s=1$ and $\gamma < C$ then $L = 0$ and $U = \gamma$.
- If $s = -1$ and $\gamma > 0$ then $L = 0$ and $U = C - \gamma$.
- If $s = -1$ and $\gamma < 0$ then $L = -\gamma$ and $U = C$.

Where $\gamma = \lambda_1 + s\lambda_2 = \lambda_1^{\text{old}} + s\lambda_2^{\text{old}}$ according to formula V.2.2.1.15.

Let $\Delta\lambda_1$ and $\Delta\lambda_2$ represent the changes in multipliers λ_1 and λ_2 , respectively.

$$\Delta\lambda_2 = \lambda_2^* - \lambda_2^{\text{old}}$$

$$\Delta\lambda_1 = -s\Delta\lambda_2$$

Where $s = y_1 y_2$.

The new value of the first multiplier λ_1 is re-written in accordance with the change $\Delta\lambda_1$.

$$\lambda_1^* = \lambda_1^{\text{new}} = \lambda_1^{\text{old}} + \Delta\lambda_1$$

Table V.2.2.1.2. SMO algorithm optimizes jointly two Lagrange multipliers

Basic tasks of SMO algorithm to optimize jointly two Lagrange multipliers are now described in detailed. The ultimate goal of SVM method is to determine the classifier (W^*, b^*) . Thus, SMO algorithm updates optimal weight W^* and optimal bias b^* based on the new values λ_1^{new} and λ_2^{new} at each optimization step.

Let $W^* = W^{\text{new}}$ be the new (optimal) weight vector, according formula V.2.2.1.11 we have:

$$W^{\text{new}} = \sum_{i=1}^n \lambda_i y_i X_i = \lambda_1^{\text{new}} y_1 X_1 + \lambda_2^{\text{new}} y_2 X_2 + \sum_{i=3}^n \lambda_i y_i X_i$$

Let $W^* = W^{\text{old}}$ be the old weight vector:

$$W^{\text{old}} = \sum_{i=1}^n \lambda_i y_i X_i = \lambda_1^{\text{old}} y_1 X_1 + \lambda_2^{\text{old}} y_2 X_2 + \sum_{i=3}^n \lambda_i y_i X_i$$

It implies:

$$W^* = W^{\text{new}} = \lambda_1^{\text{new}} y_1 X_1 - \lambda_1^{\text{old}} y_1 X_1 + \lambda_2^{\text{new}} y_2 X_2 - \lambda_2^{\text{old}} y_2 X_2 + W^{\text{old}}$$

Let E_2^{new} be the new prediction error on X_2 :

$$E_2^{\text{new}} = y_2 - (W^{\text{new}} \circ X_2 - b^{\text{new}})$$

The new (optimal) bias $b^* = b^{\text{new}}$ is determining by setting $E_2^{\text{new}} = 0$ with reason that the optimal classifier (W^*, b^*) has zero error.

$$E_2^{\text{new}} = 0 \Leftrightarrow y_2 - (W^{\text{new}} \circ X_2 - b^{\text{new}}) = 0 \Leftrightarrow b^{\text{new}} = W^{\text{new}} \circ X_2 - y_2$$

In general, formula V.2.2.1.17 specifies the optimal classifier (W^*, b^*) resulted from each optimization step of SMO algorithm.

$$\begin{aligned} W^* &= W^{\text{new}} = (\lambda_1^{\text{new}} - \lambda_1^{\text{old}}) y_1 X_1 + (\lambda_2^{\text{new}} - \lambda_2^{\text{old}}) y_2 X_2 + W^{\text{old}} \\ b^* &= b^{\text{new}} = W^{\text{new}} \circ X_2 - y_2 \end{aligned}$$

Where W^{old} is the old value of weight vector, of course we have:

$$W^{\text{old}} = \lambda_1^{\text{old}} y_1 X_1 + \lambda_2^{\text{old}} y_2 X_2 + \sum_{i=3}^n \lambda_i y_i X_i$$

Formula V.2.2.1.17. Optimal classifier (W^*, b^*) resulted from each optimization step of SMO algorithm

By extending the ideology shown in table V.2.2.1.1, SMO algorithm is described particularly in table V.2.2.1.3 (Platt, 1998, pp. 8-9) (Honavar, p. 14).

All multipliers λ_i (s), weight vector W , and bias b are initialized by zero. SMO algorithm divides the whole QP problem into many smallest optimization problems. Each smallest optimization problem focuses on optimizing two joint multipliers. SMO algorithm solves each smallest optimization problem via two nested loops:

1. The outer loop alternates one sweep through all data points and as many sweeps as possible through non-boundary data points (support vectors) so as to find out the data point X_i that violates KKT condition according to formula V.2.2.1.13. *The Lagrange multiplier λ_i associated with such X_i is selected as the first multiplier aforementioned as λ_1 .* Violating KKT condition is known as the first choice heuristic of SMO algorithm.
2. The inner loop browses all data points at the first sweep and non-boundary ones at later sweeps so as to find out the data point X_j that maximizes the deviation $|E_j^{\text{old}} - E_i^{\text{old}}|$ where E_j^{old} and E_i^{old} are prediction errors on X_i and X_j , respectively, as seen in formula V.2.2.1.16. *The Lagrange multiplier λ_j associated with such X_j is selected as the second multiplier aforementioned as λ_2 .* Maximizing the deviation $|E_2^{\text{old}} - E_1^{\text{old}}|$ is known as the second choice heuristic of SMO algorithm.
 - a. *Two Lagrange multipliers λ_1 and λ_2 are optimized jointly,* which results optimal multipliers λ_1^{new} and λ_2^{new} , as seen in table V.2.2.1.2.
 - b. *SMO algorithm updates optimal weight W^* and optimal bias b^* based on the new values λ_1^{new} and λ_2^{new} according to formula V.2.2.1.17.*

SMO algorithm continues to solve another smallest optimization problem. SMO algorithm stops when there is convergence in which no data point violating KKT

condition is found. Consequently, all Lagrange multipliers $\lambda_1, \lambda_2, \dots, \lambda_n$ are optimized and the optimal SVM classifier (W^*, b^*) is totally determined.

Table V.2.2.1.3. SMO algorithm

When both optimal weight vector W^* and optimal bias b^* are determined by SMO algorithm or other methods, the maximum-margin hyperplane known as SVM classifier is totally determined. According to formula V.2.2.1.1, the equation of maximum-margin hyperplane is expressed in formula V.2.2.1.18 as follows:

$$W^* \circ X - b^* = 0$$

Formula V.2.2.1.18. Equation of maximum-margin hyperplane (SVM classifier)

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 section V.2, data points are documents mentioned in sub-section V.2.1. Let R be the classification rule, formula V.2.2.1.19 specifies the classification rule as the sign function of point X .

$$R \stackrel{\text{def}}{=} \text{sign}(W^* \circ X - b^*) = \begin{cases} +1 & \text{if } W^* \circ X - b^* \geq 0 \\ -1 & \text{if } W^* \circ X - b^* < 0 \end{cases}$$

Formula V.2.2.1.19. Classification rule derived from maximum-margin hyperplane (SVM classifier)

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.

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 V.2.1.4 for more details about document vector. Term frequency (tf) and inverse document frequency (idf) are mentioned in formulas V.2.1.1 and V.2.1.2.

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 V.2.1.5 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 section V.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 is generalized by formula V.2.2.1.8 and implemented by SMO algorithm seen in table V.2.2.1.3 or by other methods. Table V.2.2.1.4 shows k classes along with k classifiers and k classification rules.

Class	Classifier	Classification rule
c_1	(W_1^*, b_1^*)	$R_1 = \text{sign}(W_1^* \circ X - b_1^*)$
c_2	(W_2^*, b_2^*)	$R_2 = \text{sign}(W_2^* \circ X - b_2^*)$
\vdots	\vdots	\vdots
c_k	(W_k^*, b_k^*)	$R_k = \text{sign}(W_k^* \circ X - b_k^*)$

Table V.2.2.1.4. 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:

1. For each classification rule $R_i = W_i^* \circ X + b_i^*$, 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 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 V.2.2.1.5 shows how to classify document with multi-classes.

Classification rule evaluation	Value
$R_1(D) = \text{sign}(W_1^* \circ D - b_1^*)$	$= +1$ then $D \in c_1$ $= -1$ then $D \notin c_1$
$R_2(D) = \text{sign}(W_2^* \circ D - b_2^*)$	$= +1$ then $D \in c_2$ $= -1$ then $D \notin c_2$
\vdots	\vdots
$R_k(D) = \text{sign}(W_k^* \circ D - b_k^*)$	$= +1$ then $D \in c_k$ $= -1$ then $D \notin c_k$

Table V.2.2.1.5. 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 $C = \{\text{computer science, math}\}$, a set of terms $T = \{\text{computer, derivative}\}$ and the corpus $\mathcal{D} = \{\text{doc1.txt, doc2.txt, doc3.txt, doc4.txt}\}$. The training corpus (training data) is shown in following table V.2.2.1.6 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 V.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

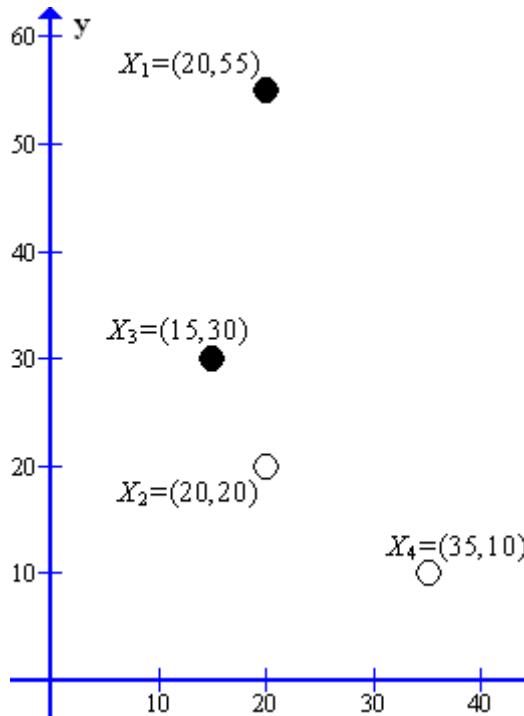
Table V.2.2.1.6. Term frequencies of documents (SVM)

Let X_i be data points representing documents *doc1.txt*, *doc2.txt*, *doc3.txt*, *doc4.txt*, *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 V.2.2.1.6 is interpreted as SVM input training corpus shown in table V.2.2.1.7.

	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 V.2.2.1.7. Training corpus (SVM)

Data points X_1 , X_2 , X_3 , and X_4 are depicted in figure V.2.2.1.7 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 V.2.2.1.7) in this research are drawn by the software *Graph* <http://www.padowan.dk> developed by author Ivan Johansen (Johansen, 2012). I express my deep gratitude to the author Ivan Johansen for providing the great software Graph.


Figure V.2.2.1.7. Data points in training data (SVM)

By applying SMO algorithm described in table V.2.2.1.3 into training corpus shown in table V.2.2.1.7, it is easy to calculate optimal multiplier λ^* , optimal weight vector

W^* and optimal bias b^* . Firstly, all multipliers λ_i (s), weight vector W , and bias b are initialized by zero. This example focuses on perfect separation and so, $C = +\infty$.

$$\begin{aligned}\lambda_1 &= \lambda_2 = \lambda_3 = \lambda_4 = 0 \\ W &= (0,0) \\ b &= 0 \\ C &= +\infty\end{aligned}$$

At the first sweep:

The outer loop of SMO algorithm searches for a data point X_i that violates KKT condition according to formula V.2.2.1.13 through all data points so as to select two multipliers that will be optimized jointly. We have:

$$E_1 = y_1 - (W \circ X_1 - b) = 1 - ((0,0) \circ (20,55) - 0) = 1$$

Due to $\lambda_1=0 < C=+\infty$ and $y_1E_1=1*1=1 > 0$, point X_1 violates KKT condition according to formula V.2.2.1.13. Then, λ_1 is selected as the first multiplier. The inner loop finds out the data point X_j that maximizes the deviation $|E_j^{\text{old}} - E_1^{\text{old}}|$. We have:

$$E_2 = y_2 - (W \circ X_2 - b) = -1 - ((0,0) \circ (20,20) - 0) = -1$$

$$|E_2 - E_1| = |-1 - 1| = 2$$

$$E_3 = y_3 - (W \circ X_3 - b) = 1 - ((0,0) \circ (15,30) - 0) = 1$$

$$|E_3 - E_1| = |1 - 1| = 0$$

$$E_4 = y_4 - (W \circ X_4 - b) = -1 - ((0,0) \circ (35,10) - 0) = -1$$

$$|E_4 - E_1| = |-1 - 1| = 2$$

Because the deviation $|E_2 - E_1|$ is maximal, the multiplier λ_2 associated with X_2 is selected as the second multiplier. Now λ_1 and λ_2 are optimized jointly according to table V.2.2.1.2.

$$\begin{aligned}\eta &= X_1 \circ X_1 - 2X_1 \circ X_2 + X_2 \circ X_2 = |X_1 - X_2|^2 = |(20,55) - (20,20)|^2 = |(0,35)|^2 \\ &= 35^2 = 1225\end{aligned}$$

$$s = y_1 y_2 = 1 * (-1) = -1$$

$$\gamma = \lambda_1^{\text{old}} + s\lambda_2^{\text{old}} = 0 + (-1) * 0 = 0$$

$$\lambda_2^{\text{new}} = \lambda_2^{\text{old}} + \frac{y_2(E_2 - E_1)}{\eta} = 0 + \frac{-1 * (-1 - 1)}{1225} = \frac{2}{1225}$$

$$\Delta\lambda_2 = \lambda_2^{\text{new}} - \lambda_2^{\text{old}} = \frac{2}{1225} - 0 = \frac{2}{1225}$$

$$\lambda_1^{\text{new}} = \lambda_1^{\text{old}} - y_1 y_2 \Delta\lambda_2 = 0 - 1 * (-1) * \frac{2}{1225} = \frac{2}{1225}$$

Optimal classifier (W^*, b^*) is updated according to formula V.2.2.1.17.

$$\begin{aligned}W^* &= W^{\text{new}} = (\lambda_1^{\text{new}} - \lambda_1^{\text{old}})y_1 X_1 + (\lambda_2^{\text{new}} - \lambda_2^{\text{old}})y_2 X_2 + W^{\text{old}} \\ &= \left(\frac{2}{1225} - 0\right) * 1 * (20,55) + \left(\frac{2}{1225} - 0\right) * (-1) * (20,20) + (0,0) \\ &= \left(0, \frac{2}{35}\right)\end{aligned}$$

$$b^* = b^{\text{new}} = W^{\text{new}} \circ X_2 - y_2 = \left(0, \frac{2}{35}\right) \circ (20,20) - (-1) = \frac{15}{7}$$

Now we have:

$$\begin{aligned}\lambda_1 &= \lambda_1^{\text{new}} = \frac{2}{1225}, \lambda_2 = \lambda_2^{\text{new}} = \frac{2}{1225}, \lambda_3 = \lambda_4 = 0 \\ W &= W^* = \left(0, \frac{2}{35}\right) \\ b &= b^* = \frac{15}{7}\end{aligned}$$

The outer loop of SMO algorithm continues to search for another data point X_i that violates KKT condition according to formula V.2.2.1.13 through all data points so as to select two other multipliers that will be optimized jointly. We have:

$$E_2 = y_2 - (W \circ X_2 - b) = -1 - \left(\left(0, \frac{2}{35} \right) \circ (20, 20) - \frac{15}{7} \right) = 0$$

$$E_3 = y_3 - (W \circ X_3 - b) = 1 - \left(\left(0, \frac{2}{35} \right) \circ (15, 30) - \frac{15}{7} \right) = \frac{10}{7}$$

Due to $\lambda_3=0 < C$ and $y_3 E_3 = 1 * (10/7) > 0$, point X_3 violates KKT condition according to formula V.2.2.1.13. Then, λ_3 is selected as the first multiplier. The inner loop finds out the data point X_j that maximizes the deviation $|E_j^{\text{old}} - E_3^{\text{old}}|$. We have:

$$E_1 = y_1 - (W \circ X_1 - b) = 1 - \left(\left(0, \frac{2}{35} \right) \circ (20, 55) - \frac{15}{7} \right) = 0$$

$$|E_1 - E_3| = \left| 0 - \frac{10}{7} \right| = \frac{10}{7}$$

$$E_2 = y_2 - (W \circ X_2 - b) = -1 - \left(\left(0, \frac{2}{35} \right) \circ (20, 20) - \frac{15}{7} \right) = 0$$

$$|E_2 - E_3| = \left| 0 - \frac{10}{7} \right| = \frac{10}{7}$$

$$E_4 = y_4 - (W \circ X_4 - b) = -1 - \left(\left(0, \frac{2}{35} \right) \circ (35, 10) - \frac{15}{7} \right) = \frac{6}{7}$$

$$|E_4 - E_3| = \left| \frac{6}{7} - \frac{10}{7} \right| = \frac{4}{7}$$

Because both deviations $|E_1 - E_3|$ and $|E_2 - E_3|$ are maximal, the multiplier λ_2 associated with X_2 is selected randomly among $\{\lambda_1, \lambda_2\}$ as the second multiplier. Now λ_3 and λ_2 are optimized jointly according to table V.2.2.1.2.

$$\begin{aligned} \eta &= X_3 \circ X_2 - 2X_3 \circ X_2 + X_3 \circ X_2 = |X_3 - X_2|^2 = |(15, 30) - (20, 20)|^2 \\ &= |(-5, 10)|^2 = (-5)^2 + 10^2 = 125 \end{aligned}$$

$$s = y_3 y_2 = 1 * (-1) = -1$$

$$\gamma = \lambda_3^{\text{old}} + s \lambda_2^{\text{old}} = 0 + (-1) * \frac{2}{1225} = -\frac{2}{1225}$$

$$L = -\gamma = \frac{2}{1225}$$

$$U = C = +\infty$$

(L and U are lower bound and upper bound of λ_2^{new})

$$\lambda_2^{\text{new}} = \lambda_2^{\text{old}} + \frac{y_2(E_2 - E_3)}{\eta} = \frac{2}{1225} + \frac{-1 * \left(0 - \frac{10}{7} \right)}{125} = \frac{16}{1225}$$

$$\Delta \lambda_2 = \lambda_2^{\text{new}} - \lambda_2^{\text{old}} = \frac{16}{1225} - \frac{2}{1225} = \frac{2}{175}$$

$$\lambda_3^{\text{new}} = \lambda_3^{\text{old}} - y_3 y_2 \Delta \lambda_2 = 0 - 1 * (-1) * \frac{2}{175} = \frac{2}{175}$$

Optimal classifier (W^*, b^*) is updated according to formula V.2.2.1.17.

$$\begin{aligned}
 W^* = W^{\text{new}} &= (\lambda_3^{\text{new}} - \lambda_3^{\text{old}})y_3X_3 + (\lambda_2^{\text{new}} - \lambda_2^{\text{old}})y_2X_2 + W^{\text{old}} \\
 &= \left(\frac{2}{175} - 0\right) * 1 * (15,30) + \left(\frac{16}{1225} - \frac{2}{1225}\right) * (-1) * (20,20) \\
 &\quad + \left(0, \frac{2}{35}\right) = \left(-\frac{2}{35}, \frac{6}{35}\right) \\
 b^* = b^{\text{new}} &= W^{\text{new}} \circ X_2 - y_2 = \left(-\frac{2}{35}, \frac{6}{35}\right) \circ (20,20) - (-1) = \frac{23}{7}
 \end{aligned}$$

Now we have:

$$\begin{aligned}
 \lambda_1 &= \frac{2}{1225}, \lambda_2 = \lambda_2^{\text{new}} = \frac{16}{1225}, \lambda_3 = \lambda_3^{\text{new}} = \frac{2}{175}, \lambda_4 = 0 \\
 W &= W^* = \left(-\frac{2}{35}, \frac{6}{35}\right) \\
 b &= b^* = \frac{23}{7}
 \end{aligned}$$

The outer loop of SMO algorithm continues to search for another data point X_i that violates KKT condition according to formula V.2.2.1.13 through all data points so as to select two other multipliers that will be optimized jointly. We have:

$$E_4 = y_4 - (W \circ X_4 - b) = -1 - \left(\left(-\frac{2}{35}, \frac{6}{35}\right) \circ (35,10) - \frac{23}{7}\right) = \frac{18}{7}$$

Due to $\lambda_4=0 < C$ and $y_4E_4=(-1)*(18/7) < 0$, point X_4 does not violate KKT condition according to formula V.2.2.1.13. Then, the first sweep of outer loop stops with the results as follows:

$$\begin{aligned}
 \lambda_1 &= \frac{2}{1225}, \lambda_2 = \lambda_2^{\text{new}} = \frac{16}{1225}, \lambda_3 = \lambda_3^{\text{new}} = \frac{2}{175}, \lambda_4 = 0 \\
 W &= W^* = \left(-\frac{2}{35}, \frac{6}{35}\right) \\
 b &= b^* = \frac{23}{7}
 \end{aligned}$$

Note that λ_1 is approximated to 0 because it is very small.

At the second sweep:

The outer loop of SMO algorithm searches for a data point X_i that violates KKT condition according to formula V.2.2.1.13 through non-boundary data points so as to select two multipliers that will be optimized jointly. Recall that non-boundary data points (support vectors) are ones whose associated multipliers are not bounds 0 and C ($0 < \lambda_i < C$). At the second sweep, there are three non-boundary data points X_1 , X_2 and X_3 . We have:

$$E_1 = y_1 - (W \circ X_1 - b) = 1 - \left(\left(-\frac{2}{35}, \frac{6}{35}\right) \circ (20,55) - \frac{23}{7}\right) = -4$$

Due to $\lambda_1 = \frac{2}{1225} > 0$ and $y_1E_1 = 1*(-4) < 0$, point X_1 violates KKT condition according to formula V.2.2.1.13. Then, λ_1 is selected as the first multiplier. The inner loop finds out the data point X_j among non-boundary data points ($0 < \lambda_i < C$) that maximizes the deviation $|E_j^{\text{old}} - E_1^{\text{old}}|$. We have:

$$E_2 = y_2 - (W \circ X_2 - b) = -1 - \left(\left(-\frac{2}{35}, \frac{6}{35}\right) \circ (20,20) - \frac{23}{7}\right) = 0$$

$$|E_2 - E_1| = |0 - (-4)| = 4$$

$$E_3 = y_3 - (W \circ X_3 - b) = 1 - \left(\left(-\frac{2}{35}, \frac{6}{35}\right) \circ (15,30) - \frac{23}{7}\right) = 0$$

$$|E_3 - E_1| = |0 - (-4)| = 4$$

Because the deviation $|E_2 - E_1|$ is maximal, the multiplier λ_2 associated with X_2 is selected as the second multiplier. Now λ_1 and λ_2 are optimized jointly according to table V.2.2.1.2.

$$\begin{aligned}\eta &= X_1 \circ X_1 - 2X_1 \circ X_2 + X_2 \circ X_2 = |X_1 - X_2|^2 = |(20,55) - (20,20)|^2 = |(0,35)|^2 \\ &= 35^2 = 1225\end{aligned}$$

$$s = y_1 y_2 = 1 * (-1) = -1$$

$$\gamma = \lambda_1^{\text{old}} + s\lambda_2^{\text{old}} = \frac{2}{1225} + (-1) * \frac{16}{1225} = -\frac{14}{1225}$$

$$L = -\gamma = \frac{14}{1225} = \frac{2}{175}$$

$$U = C = +\infty$$

(L and U are lower bound and upper bound of λ_2^{new})

$$\begin{aligned}\lambda_2^{\text{new}} &= \lambda_2^{\text{old}} + \frac{y_2(E_2 - E_1)}{\eta} = \frac{16}{1225} + \frac{-1 * (0 - (-4))}{1225} = \frac{16}{1225} - \frac{4}{1225} = \frac{12}{1225} \\ &< L = \frac{2}{175}\end{aligned}$$

$$\Rightarrow \lambda_2^{\text{new}} = L = \frac{2}{175}$$

$$\Delta\lambda_2 = \lambda_2^{\text{new}} - \lambda_2^{\text{old}} = \frac{2}{175} - \frac{16}{1225} = -\frac{2}{1225}$$

$$\lambda_1^{\text{new}} = \lambda_1^{\text{old}} - y_1 y_2 \Delta\lambda_2 = \frac{2}{1225} - 1 * (-1) * \left(-\frac{2}{1225}\right) = 0$$

Optimal classifier (W^*, b^*) is updated according to formula V.2.2.1.17.

$$\begin{aligned}W^* &= W^{\text{new}} = (\lambda_1^{\text{new}} - \lambda_1^{\text{old}})y_1 X_1 + (\lambda_2^{\text{new}} - \lambda_2^{\text{old}})y_2 X_2 + W^{\text{old}} \\ &= \left(0 - \frac{2}{1225}\right) * 1 * (20,55) + \left(\frac{2}{175} - \frac{16}{1225}\right) * (-1) * (20,20) \\ &\quad + \left(-\frac{2}{35}, \frac{6}{35}\right) = \left(-\frac{2}{35}, \frac{4}{35}\right)\end{aligned}$$

$$b^* = b^{\text{new}} = W^{\text{new}} \circ X_2 - y_2 = \left(-\frac{2}{35}, \frac{4}{35}\right) \circ (20,20) - (-1) = \frac{15}{7}$$

The second sweep stops with results as follows:

$$\begin{aligned}\lambda_1 &= \lambda_1^{\text{new}} = 0, \lambda_2 = \lambda_2^{\text{new}} = \frac{2}{175}, \lambda_3 = \frac{2}{175}, \lambda_4 = 0 \\ W &= W^* = \left(-\frac{2}{35}, \frac{4}{35}\right) \\ b &= b^* = \frac{15}{7}\end{aligned}$$

At the third sweep:

The outer loop of SMO algorithm searches for a data point X_i that violates KKT condition according to formula V.2.2.1.13 through non-boundary data points so as to select two multipliers that will be optimized jointly. Recall that non-boundary data points (support vectors) are ones whose associated multipliers are not bounds 0 and C ($0 < \lambda_i < C$). At the third sweep, there are only two non-boundary data points X_2 and X_3 . We have:

$$E_2 = y_2 - (W \circ X_2 - b) = -1 - \left(\left(-\frac{2}{35}, \frac{4}{35}\right) \circ (20,20) - \frac{15}{7}\right) = 0$$

$$E_3 = y_3 - (W \circ X_3 - b) = 1 - \left(\left(-\frac{2}{35}, \frac{4}{35} \right) \circ (15, 30) - \frac{15}{7} \right) = \frac{4}{7}$$

Due to $\lambda_3 = \frac{2}{175} < C = +\infty$ and $y_3 E_3 = 1 * (4/7) > 0$, point X_3 violates KKT condition according to formula V.2.2.1.13. Then, λ_3 is selected as the first multiplier. Because there are only two non-boundary data points X_2 and X_3 , the second multiplier is λ_2 . Now λ_3 and λ_2 are optimized jointly according to table V.2.2.1.2.

$$\begin{aligned} \eta &= X_3 \circ X_2 - 2X_3 \circ X_2 + X_3 \circ X_2 = |X_3 - X_2|^2 = |(15, 30) - (20, 20)|^2 \\ &= |(-5, 10)|^2 = (-5)^2 + (10)^2 = 125 \end{aligned}$$

$$s = y_3 y_2 = 1 * (-1) = -1$$

$$\gamma = \lambda_3^{\text{old}} + s\lambda_2^{\text{old}} = \frac{2}{175} + (-1) * \frac{2}{175} = 0$$

$$L = 0$$

$$U = C - \gamma = +\infty$$

(L and U are lower bound and upper bound of λ_2^{new})

$$\lambda_2^{\text{new}} = \lambda_2^{\text{old}} + \frac{y_2(E_2 - E_3)}{\eta} = \frac{2}{175} + \frac{-1 * \left(0 - \frac{4}{7}\right)}{125} = \frac{2}{125}$$

$$\Delta\lambda_2 = \lambda_2^{\text{new}} - \lambda_2^{\text{old}} = \frac{2}{125} - \frac{2}{175} = \frac{4}{875}$$

$$\lambda_3^{\text{new}} = \lambda_3^{\text{old}} - y_3 y_2 \Delta\lambda_2 = \frac{2}{175} - 1 * (-1) * \frac{4}{875} = \frac{2}{125}$$

Optimal classifier (W^* , b^*) is updated according to formula V.2.2.1.17.

$$\begin{aligned} W^* &= W^{\text{new}} = (\lambda_3^{\text{new}} - \lambda_3^{\text{old}}) y_3 X_3 + (\lambda_2^{\text{new}} - \lambda_2^{\text{old}}) y_2 X_2 + W^{\text{old}} \\ &= \left(\frac{2}{125} - \frac{2}{175} \right) * 1 * (15, 30) + \left(\frac{2}{125} - \frac{2}{175} \right) * 1 * (20, 20) \\ &\quad + \left(-\frac{2}{35}, \frac{4}{35} \right) = \left(\frac{18}{175}, \frac{12}{35} \right) \end{aligned}$$

$$b^* = b^{\text{new}} = W^{\text{new}} \circ X_2 - y_2 = \left(\frac{18}{175}, \frac{12}{35} \right) \circ (20, 20) - (-1) = \frac{347}{35}$$

The third sweep stops with results as follows:

$$\lambda_1 = 0, \lambda_2 = \lambda_2^{\text{new}} = \frac{2}{125}, \lambda_3 = \lambda_3^{\text{new}} = \frac{2}{125}, \lambda_4 = 0$$

$$W = W^* = \left(\frac{18}{175}, \frac{12}{35} \right)$$

$$b = b^* = \frac{347}{35}$$

After the third sweep, two non-boundary multipliers were optimized jointly. You can sweep more times to get more optimal results because data point X_3 still violates KKT condition as follows:

$$E_3 = y_3 - (W \circ X_3 - b) = 1 - \left(\left(\frac{18}{175}, \frac{12}{35} \right) \circ (15, 30) - \frac{347}{35} \right) = -\frac{32}{35}$$

Due to $\lambda_3 = \frac{2}{125} > 0$ and $y_3 E_3 = 1 * (-32/35) < 0$, point X_3 violates KKT condition according to formula V.2.2.1.13. But it takes a lot of sweeps so that SMO algorithm reaches absolute convergence ($E_3=0$ and hence, no KKT violation) because the penalty C is set to be $+\infty$, which implicates the perfect separation. This is the reason that we can stop the SMO algorithm at the third sweep in this example. In general, you can totally stop the SMO algorithm after optimizing two last multipliers which implies that all multipliers were optimized.

As a result, W^* and b^* were determined:

$$W^* = \left(\frac{18}{175}, \frac{12}{35} \right)$$

$$b^* = \frac{347}{35}$$

The maximum-margin hyperplane (SVM classifier) is totally determined as below:

$$W^* \circ X - b^* = 0 \Rightarrow \left(\frac{18}{175}, \frac{12}{35} \right) \circ (x, y) - \frac{347}{35} = 0$$

$$\Rightarrow y = -0.3x + 28.92$$

The SVM classifier $y = -0.3x + 28.9$ is depicted in figure V.2.2.1.8.

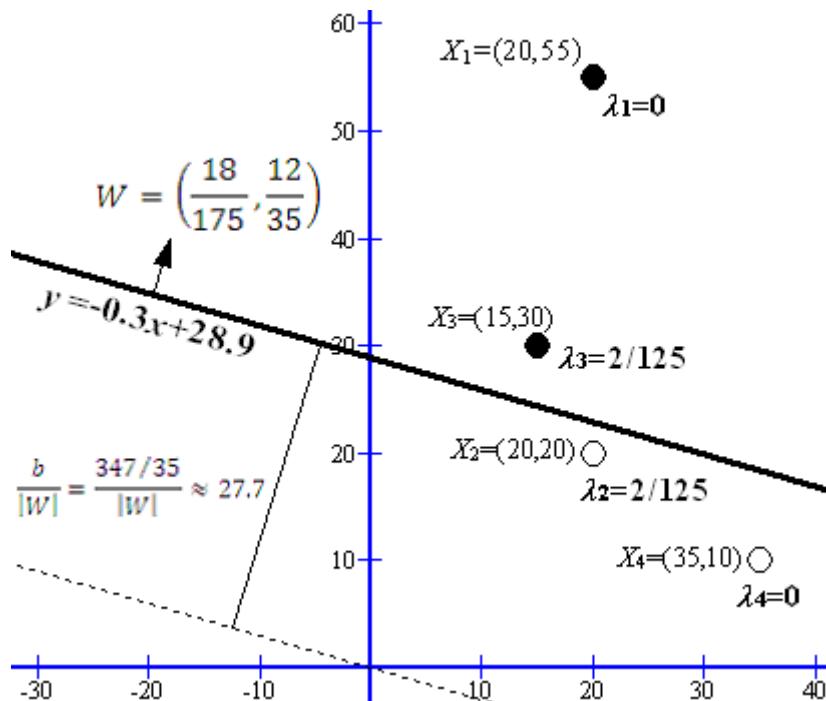


Figure V.2.2.1.8. An example of maximum-margin hyperplane

Where the maximum-margin hyperplane is draw as bold line. Data points X_2 and X_3 are support vectors because their associated multipliers λ_2 and λ_3 are non-zero ($0 < \lambda_2 = 2/125 < C = +\infty$, $0 < \lambda_3 = 2/125 < C = +\infty$). Your attention please, that weight vector W is depicted as an arrow indicates mainly its direction. The scale of weight vector $W = \left(\frac{18}{175}, \frac{12}{35} \right)$ in this figure is very small.

Derived from the above classifier $y = -0.3x + 28.9$, the classification rule is:

$$R \stackrel{\text{def}}{=} \text{sign}(0.3x + y - 28.9) = \begin{cases} +1 & \text{if } 0.3x + y - 28.9 \geq 0 \\ -1 & \text{if } 0.3x + y - 28.9 < 0 \end{cases}$$

Now we apply classification rule $R \stackrel{\text{def}}{=} \text{sign}(0.3x + y - 28.9)$ into document classification. Suppose the numbers of times that terms “computer” and “derivative” occur in document D are 40 and 10, respectively. We need to determine which class document $D=(40, 10)$ belongs to. We have:

$$R(D) = \text{sign}(0.3 * 40 + 10 - 28.9) = \text{sign}(-6.9) = -1$$

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 sub-section [V.2.2.2](#) mentions another document classification method based on decision tree.

V.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 [V.2.2.2.1](#) 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 [V.2.1](#) for more details about document vectors.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	<i>class</i>
<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 V.2.2.2.1. 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:

$$tf_{11} = \frac{tf_{11}}{tf_{11} + tf_{12} + tf_{13} + tf_{14}} = \frac{5}{5 + 3 + 1 + 1} = 0.5$$

$$tf_{12} = \frac{tf_{12}}{tf_{11} + tf_{12} + tf_{13} + tf_{14}} = \frac{3}{5 + 3 + 1 + 1} \approx 0.3$$

$$tf_{13} = \frac{tf_{13}}{tf_{11} + tf_{12} + tf_{13} + tf_{14}} = \frac{1}{5 + 3 + 1 + 1} = 0.1$$

$$tf_{14} = \frac{tf_{14}}{tf_{11} + tf_{12} + tf_{13} + tf_{14}} = \frac{1}{5 + 3 + 1 + 1} = 0.1$$

Table [V.2.2.2.2](#) shows normalized term frequencies in corpus \mathcal{D} .

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	<i>class</i>
<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 V.2.2.2.2. 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:

$0 \leq tf < 0.2$	\rightarrow	low
$0.2 \leq tf < 0.5$	\rightarrow	medium
$0.5 \leq tf$	\rightarrow	high

Table [V.2.2.2.3](#) shows nominal term frequencies according to these converted specifications.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	<i>class</i>
<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 V.2.2.2.3. 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 [V.2.2.2.3](#)) and non-leaf nodes are attributes of data (like “computer”, “programming language”, “algorithm”, and “derivative”). In similar to SVM method aforementioned in previous sub-section [V.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 [V.2.2.2.1](#), [V.2.2.2.2](#), and [V.2.2.2.3](#) 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 $Entropy(\mathcal{D})$ denote entropy of corpus \mathcal{D} , formula [V.2.2.2.1](#) (Han & Kamber, 2006, p. 297) is used to calculate $Entropy(\mathcal{D})$.

$$Entropy(\mathcal{D}) = - \sum_{i=1}^n P_i \log_2(P_i)$$

Formula V.2.2.2.1. Entropy of training corpus

Where P_i is the frequency of occurrence of class C_i , for example, we have $C_1=\text{"computer science"}$ and $C_2=\text{"math"}$ given training corpus shown in table V.2.2.2.3. 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 V.2.2.2.3, 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 V.2.2.2.2 (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 V.2.2.2.3, 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 V.2.2.2.2. 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 V.2.2.2.3 as follows (Han & Kamber, 2006, p. 298):

$$\text{Gain}(A, \mathcal{D}) = \text{Entropy}(\mathcal{D}) - \text{Entropy}(A, \mathcal{D})$$

Formula V.2.2.2.3. Information gain of given attribute inside training corpus

Given training corpus shown in table V.2.2.2.3, information gains of attributes $A:=\text{"computer"}$, $A:=\text{"programming language"}$, $A:=\text{"algorithm"}$, and $A:=\text{"derivative"}$ are calculated according to formula V.2.2.2.3, in turn. We have:

$A:=\text{"computer"}$

$\text{Entropy}(\mathcal{D}) = 1$

\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
$\text{Entropy}(\mathcal{D}_1) = - \frac{2}{2} \log_2 \left(\frac{2}{2} \right) = 0$					

\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 language</i>	<i>algorithm</i>	<i>derivative</i>	class
<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	computer	programming language	algorithm	derivative	class
doc1.txt	high	medium	low	low	computer science
doc4.txt	medium	high	low	medium	computer science
$Entropy(\mathcal{D}_1) = -\frac{2}{2} \log_2 \left(\frac{2}{2} \right) = 0$					
\mathcal{D}_2	computer	programming language	algorithm	derivative	class
doc2.txt	low	low	medium	high	math
doc3.txt	medium	low	medium	high	math
doc5.txt	low	low	medium	medium	math
doc6.txt	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”

$$Entropy(\mathcal{D}) = -1$$

\mathcal{D}_1	computer	programming language	algorithm	derivative	class
doc1.txt	high	medium	low	low	computer science
doc6.txt	medium	low	medium	low	computer science
$Entropy(\mathcal{D}_1) = -\frac{2}{2} \log_2 \left(\frac{2}{2} \right) = 0$					
\mathcal{D}_2	computer	programming	algorithm	derivative	class

		<i>language</i>			
<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
$Entropy(\mathcal{D}_3) = -\frac{2}{2} \log_2 \left(\frac{2}{2}\right) = 0$					

$$Entropy(A, \mathcal{D}) = Entropy("derivative", \mathcal{D}) = \sum_{j=1}^3 \frac{|\mathcal{D}_j|}{|\mathcal{D}|} Entropy(\mathcal{D}_j)$$

$$= \frac{2}{6} * 0 + \frac{2}{6} * 1 + \frac{2}{6} * 0 = \frac{1}{3}$$

$$Gain(A, \mathcal{D}) = Gain("derivative", \mathcal{D}) = Entropy(\mathcal{D}) - Entropy("derivative", \mathcal{D}) \\ = 1 - \frac{1}{3} \approx 0.67$$

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 V.2.2.2.1 depicts the decision constructed from nominal term frequencies shown in table V.2.2.2.3 at the first splitting.

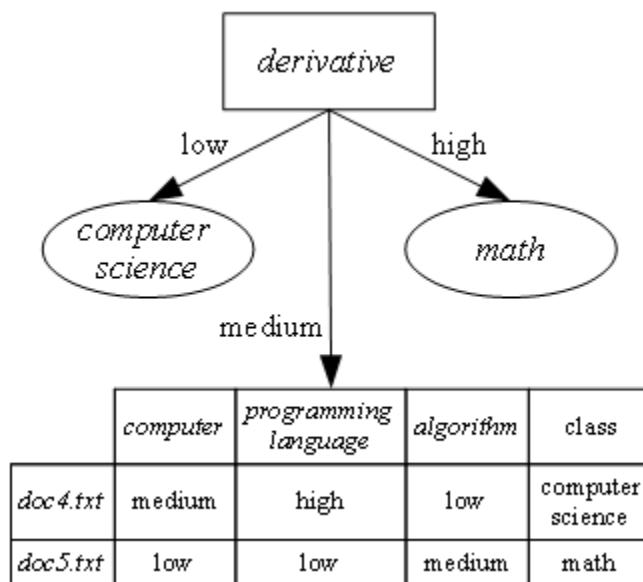


Figure V.2.2.2.1. Decision tree constructed from nominal term frequencies at the first splitting

As seen in figure V.2.2.2.1, 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 V.2.2.2.4.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	class
<i>doc4.txt</i>	medium	high	low	computer science
<i>doc5.txt</i>	low	low	medium	math

Table V.2.2.2.4. 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 [V.2.2.2.4](#), 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 [V.2.2.2.5](#):

A	Partitions					$\text{Entropy}(A, \mathcal{D}')$	$\text{Gain}(A, \mathcal{D}')$		
“ <i>computer</i> ”	\mathcal{D}'_1	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	class	$\text{Entropy}(A, \mathcal{D}') = \frac{1}{2} * 0 + \frac{1}{2} * 0 = 0$	$\text{Gain}(A, \mathcal{D}') = 1 - 0 = 1$		
	<i>doc4.txt</i>	medium	high	low	computer science				
	$\text{Entropy}(\mathcal{D}'_1) = -\frac{1}{2} \log_2\left(\frac{1}{2}\right) = 0$								
	\mathcal{D}'_2	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	class				
	<i>doc5.txt</i>	low	low	medium	math				
	$\text{Entropy}(\mathcal{D}'_2) = -\frac{1}{2} \log_2\left(\frac{1}{2}\right) = 0$								
“ <i>programming language</i> ”	\mathcal{D}'_1	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	class	$\text{Entropy}(A, \mathcal{D}') = \frac{1}{2} * 0 + \frac{1}{2} * 0 = 0$	$\text{Gain}(A, \mathcal{D}') = 1 - 0 = 1$		
	<i>doc4.txt</i>	medium	high	low	computer science				
	$\text{Entropy}(\mathcal{D}'_1) = -\frac{1}{2} \log_2\left(\frac{1}{2}\right) = 0$								
	\mathcal{D}'_2	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	class				
	<i>doc5.txt</i>	low	low	medium	math				
	$\text{Entropy}(\mathcal{D}'_2) = -\frac{1}{2} \log_2\left(\frac{1}{2}\right) = 0$								
“ <i>algorithm</i> ”	\mathcal{D}'_1	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	class	$\text{Entropy}(A, \mathcal{D}') = \frac{1}{2} * 0 + \frac{1}{2} * 0 = 0$	$\text{Gain}(A, \mathcal{D}') = 1 - 0 = 1$		
	<i>doc4.txt</i>	medium	high	low	computer science				
	$\text{Entropy}(\mathcal{D}'_1) = -\frac{1}{2} \log_2\left(\frac{1}{2}\right) = 0$								
	\mathcal{D}'_2	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	class				
	<i>doc5.txt</i>	low	low	medium	math				
	$\text{Entropy}(\mathcal{D}'_2) = -\frac{1}{2} \log_2\left(\frac{1}{2}\right) = 0$								

Table V.2.2.2.5. Information gains at the second splitting

Because attributes “*computer*”, “*programming language*”, and “*algorithm*” have the same information gain $\text{Gain}(A, \mathcal{D}')=1$, the decision tree is refined by the second splitting at attributes “*computer*”, “*programming language*”, and “*algorithm*”. Following figure [V.2.2.2.2](#) 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 [V.2.2.2.3](#). Note that leaf nodes representing

document classes are drawn as ellipses and non-leaf nodes representing attributes (terms) are drawn as rectangles.

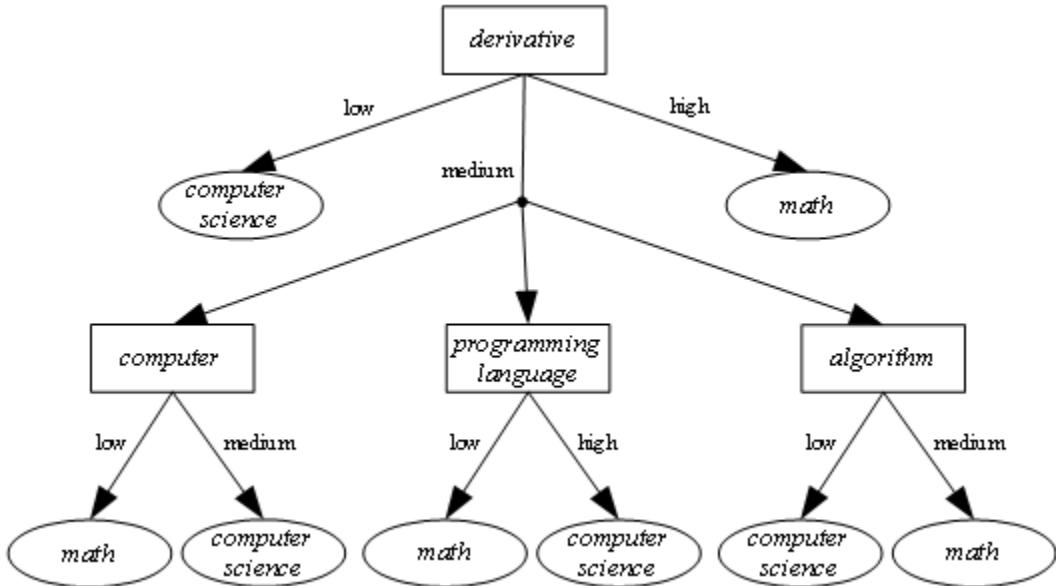


Figure V.2.2.2.2. Final decision tree constructed from nominal term frequencies

The decision tree as shown in figure V.2.2.2.2 is the classifier of decision tree method. It is easy to extract classification rules from this decision tree. Table V.2.2.6 expresses these classification rules.

Rule	Description
R_1	If frequency of term “derivative” is <i>low</i> then document belongs to class “computer science”.
R_2	If frequency of term “derivative” is <i>medium</i> and frequency of term “computer” is <i>medium</i> then document belongs to class “computer science”.
R_3	If frequency of term “derivative” is <i>medium</i> and frequency of term “computer” is <i>low</i> then document belongs to class “math”.
R_4	If frequency of term “derivative” is <i>medium</i> and frequency of term “programming language” is <i>high</i> then document belongs to class “computer science”.
R_5	If frequency of term “derivative” is <i>medium</i> and frequency of term “programming language” is <i>low</i> then document belongs to class “math”.
R_6	If frequency of term “derivative” is <i>medium</i> and frequency of term “algorithm” is <i>low</i> then document belongs to class “computer science”.
R_7	If frequency of term “derivative” is <i>medium</i> and frequency of term “algorithm” is <i>medium</i> then document belongs to class “math”.
R_8	If frequency of term “derivative” is <i>high</i> then document belongs to class “math”.

Table V.2.2.6. 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 according to following specifications

$0 \leq tf < 0.2$	→ low
$0.2 \leq tf < 0.5$	→ medium
$0.5 \leq tf$	→ high

We have:

$$D = (\text{medium}, \text{medium}, \text{low}, \text{medium})$$

According to the rule R_2 in above table V.2.2.2.6, 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 sub-section V.2.2.3 mentions another document classification method based on neural network.

V.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). 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 V.2.2.3.1 (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.

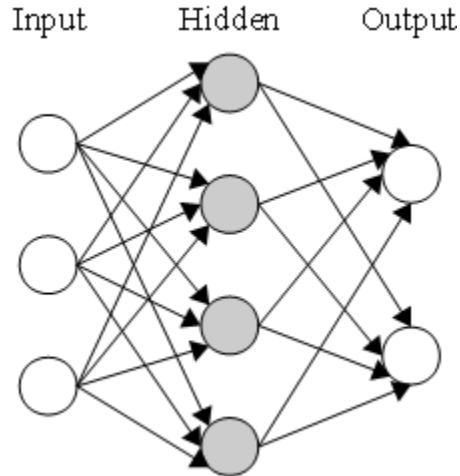


Figure V.2.2.3.1. Simplest topology of neural network with three layers such as input layer, hidden layer, and output layer

Each connection between unit i and unit j is defined by the weight w_{ij} determining the effect of unit i on unit j . Connection in neural network and arc in Bayesian network (see sub-section III.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 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 V.2.2.3.2 (Han & Kamber, 2006, p. 331) describes the process of computing the output.

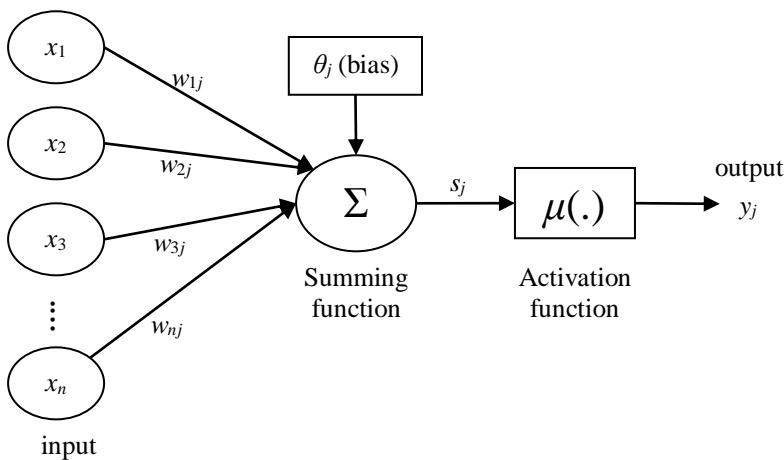


Figure V.2.2.3.2. Process of computing output of a unit

For example as seen in figure V.2.2.3.2, 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 V.2.2.3.1 (Han & Kamber, 2006, p. 331) for computing the output value of a unit.

$$I_j = \sum_{i=1}^n w_{ij} O_i + \theta_j$$

$$O_j = \mu(I_j)$$

Formula V.2.2.3.1. 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 θ_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.

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. 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 data 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 V.2.2.2.1 and V.2.2.2.2 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 V.2.2.3.1 describes back-propagation algorithm for learning neural network by pseudo-code like programming language.

1. Initializing the weights: The weights w_{ij} of connections between units are initialized as random real numbers which should be in space $[0, 1]$. Each bias θ_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 θ_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 [V.2.2.3.1](#) 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 [V.2.2.3.2](#) as below:

$$Err_j = O_j(1 - O_j)(V_j - O_j)$$

Formula V.2.2.3.2. 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 [V.2.2.3.3](#) as below:

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$

Formula V.2.2.3.3. 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 Δw_{ij} is the change in weight w_{ij} , the weight w_{ij} is updated according to formula [V.2.2.3.4](#) as below:

$$\begin{aligned}\Delta w_{ij} &= l * Err_j O_i \\ w_{ij} &= w_{ij} + \Delta w_{ij}\end{aligned}$$

Formula V.2.2.3.4. 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 θ_j over the whole neural network. The bias θ_j of hidden or output unit j is updated according to formula [V.2.2.3.5](#) as below:

$$\Delta\theta_j = l * Err_j$$

$$\theta_j = \theta_j + \Delta\theta_j$$

Formula V.2.2.3.5. Updating bias

Where l is learning rate ranging from 0 to 1.

End for each bias θ_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 Δw_{ij} in some iteration are smaller than given threshold.
- Or, iterating through all possible training data rows.

Table V.2.2.3.1. 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 sub-section [V.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 [V.2.2.3.3](#) is the one that has no cycle in its model.

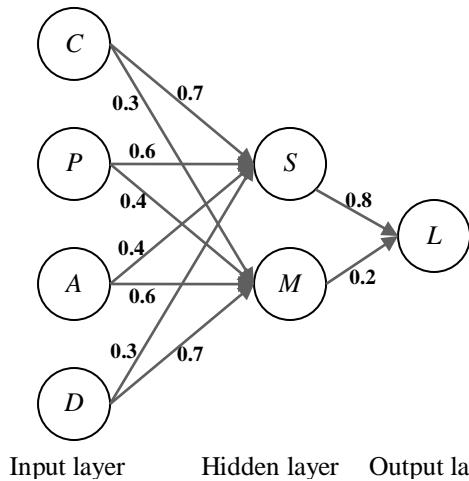


Figure V.2.2.3.3. The neural network for document classification

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 sub-section V.2.2.2, given corpus $\mathcal{D} = \{doc1.txt, doc2.txt, doc3.txt, doc4.txt, doc5.txt, doc6.txt\}$. The training corpus (training data) is shown in following table V.2.2.3.2 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 V.2.2.3.2 is the replication of table V.2.2.2.1 because it is convenient for readers to keep tract main content.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
<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 V.2.2.3.2. 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:

$$tf_{11} = \frac{tf_{11}}{tf_{11} + tf_{12} + tf_{13} + tf_{14}} = \frac{5}{5 + 3 + 1 + 1} = 0.5$$

$$tf_{12} = \frac{tf_{12}}{tf_{11} + tf_{12} + tf_{13} + tf_{14}} = \frac{3}{5 + 3 + 1 + 1} \approx 0.3$$

$$tf_{13} = \frac{tf_{13}}{tf_{11} + tf_{12} + tf_{13} + tf_{14}} = \frac{1}{5 + 3 + 1 + 1} = 0.1$$

$$tf_{14} = \frac{tf_{14}}{tf_{11} + tf_{12} + tf_{13} + tf_{14}} = \frac{1}{5 + 3 + 1 + 1} = 0.1$$

Table V.2.2.3.3 shows normalized term frequencies in corpus \mathcal{D} . Table V.2.2.3.3 is the replication of table V.2.2.2.2 because it is convenient for readers to keep tract main content.

	<i>computer</i>	<i>programming language</i>	<i>algorithm</i>	<i>derivative</i>	class
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 V.2.2.3.3. Training corpus – Normalized term frequencies (neural network)

Data rows in table [V.2.2.3.3](#) representing normalized document vectors are fed to our original neural network in figure [V.2.2.3.3](#) for supervised learning. The back-propagation algorithm is used to train network, as described in table [V.2.2.3.1](#).

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 θ_S, θ_M , and θ_L be biases 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 [V.2.2.3.3](#), 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 [V.2.2.3.3](#), 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 [V.2.2.3.1](#) (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 [V.2.2.3.2](#) 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 [V.2.2.3.3](#) for updating error of hidden units, we have:

$$Err_S = O_S(1 - O_S)Err_L W_{SL} = 0.65 * (1 - 0.65) * 0.08 * 0.8 \approx 0.01$$

$$Err_M = O_M(1 - O_M)Err_L W_{ML} = 0.6 * (1 - 0.6) * 0.08 * 0.2 \approx 0$$

According to formula [V.2.2.3.4](#) for updating connection weights given learning rate $l=1$, we have:

$$w_{CS} = w_{CS} + \Delta w_{CS} = w_{CS} + 1 * Err_S O_C = 0.7 + 1 * 0.01 * 0.5 \approx 0.71$$

$$w_{CM} = w_{CM} + \Delta w_{CM} = w_{CM} + 1 * Err_M O_C = 0.3 + 1 * 0 * 0.5 \approx 0.3$$

$$w_{PS} = w_{PS} + \Delta w_{PS} = w_{PS} + 1 * Err_S O_P = 0.6 + 1 * 0.01 * 0.3 \approx 0.6$$

$$\begin{aligned}
 w_{PM} &= w_{PM} + \Delta w_{PM} = w_{PM} + 1 * Err_M O_P = 0.4 + 1 * 0 * 0.3 \approx 0.4 \\
 w_{AS} &= w_{AS} + \Delta w_{AS} = w_{AS} + 1 * Err_S O_A = 0.4 + 1 * 0.01 * 0.1 \approx 0.4 \\
 w_{AM} &= w_{AM} + \Delta w_{AM} = w_{AM} + 1 * Err_M O_A = 0.6 + 1 * 0 * 0.1 \approx 0.6 \\
 w_{DS} &= w_{DS} + \Delta w_{DS} = w_{DS} + 1 * Err_S O_D = 0.3 + 1 * 0.01 * 0.1 \approx 0.3 \\
 w_{DM} &= w_{DM} + \Delta w_{DM} = w_{DM} + 1 * Err_M O_D = 0.7 + 1 * 0 * 0.1 \approx 0.7 \\
 w_{SL} &= w_{SL} + \Delta w_{SL} = w_{SL} + 1 * Err_L O_S = 0.8 + 1 * 0.08 * 0.65 \approx 0.85 \\
 w_{ML} &= w_{ML} + \Delta w_{ML} = w_{ML} + 1 * Err_L O_M = 0.2 + 1 * 0.08 * 0.6 \approx 0.25
 \end{aligned}$$

According to formula V.2.2.3.5 for updating biases θ_S , θ_M , and θ_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 V.2.2.3.4 shows results from this training process based on back-propagation algorithm.

	Inputs	Outputs	Weights	Biases
D_1	$I_C=0.5$	$O_S=0.65$	$w_{CS}=0.70$	$\theta_S=0.01$
	$I_P=0.3$	$O_M=0.60$	$w_{CM}=0.30$	$\theta_M=0.00$
	$I_A=0.1$	$O_L=0.65$	$w_{PS}=0.60$	$\theta_L=0.08$
	$I_D=0.1$		$w_{PM}=0.40$	
			$w_{AS}=0.40$	
			$w_{AM}=0.60$	
			$w_{DS}=0.30$	
			$w_{DM}=0.70$	
			$w_{SL}=0.85$	
			$w_{ML}=0.25$	
D_2	$I_C=0.05$	$O_S=0.60$	$w_{CS}=0.70$	$\theta_S=-0.02$
	$I_P=0.05$	$O_M=0.65$	$w_{CM}=0.30$	$\theta_M=-0.01$
	$I_A=0.40$	$O_L=0.71$	$w_{PS}=0.60$	$\theta_L=-0.07$
	$I_D=0.50$		$w_{PM}=0.40$	
			$w_{AS}=0.39$	
			$w_{AM}=0.59$	
			$w_{DS}=0.29$	
			$w_{DM}=0.69$	
			$w_{SL}=0.76$	
			$w_{ML}=0.40$	
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	$I_C=0.20$	$O_S=0.62$	$w_{CS}=0.70$	$\theta_S=-0.03$
	$I_P=0.05$	$O_M=0.60$	$w_{CM}=0.30$	$\theta_M=-0.02$

	$I_A=0.20$ $I_D=0.55$	$O_L=0.62$	$w_{PS}=0.61$ $w_{PM}=0.41$ $w_{AS}=0.38$ $w_{AM}=0.59$ $w_{DS}=0.27$ $w_{DM}=0.68$ $w_{SL}=0.73$ $w_{ML}=0.55$	$\theta_L=-0.13$
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 V.2.2.3.4. Results from training process based on back-propagation algorithm

According to the training results shown in table V.2.2.3.4, the weights and biases of origin neural network are changed. It means that neural network is already trained. Thus, figure V.2.2.3.4 expresses the neural network learned by back-propagation algorithm.

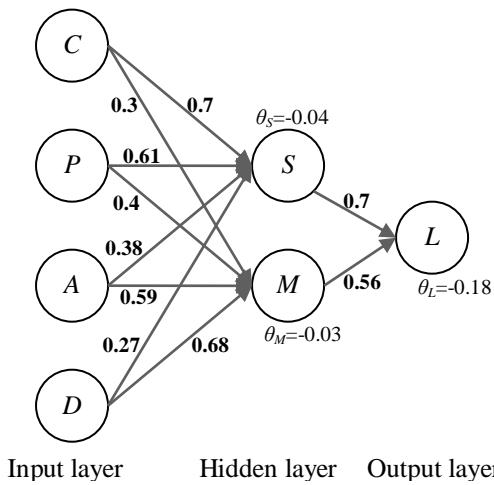


Figure V.2.2.3.4. Trained neural network

The trained neural network depicted in figure V.2.2.3.4 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 is 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 V.2.2.3.4 has connection weights $w_{CS}=0.7$, $w_{CM}=0.3$, $w_{PS}=0.61$, $w_{PM}=0.4$, $w_{AS}=0.38$, $w_{AM}=0.59$, $w_{DS}=0.27$, $w_{DM}=0.68$, $w_{SL}=0.7$, $w_{ML}=0.56$ 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 V.2.2.3.1 (Han & Kamber, 2006, p. 331) for computing the output value of a unit, we have:

$$I_S = w_{CS}O_C + w_{PS}O_P + w_{AS}O_A + w_{DS}O_D + \theta_s \\ = 0.7 * 0.4 + 0.61 * 0.3 + 0.38 * 0.1 + 0.27 * 0.2 - 0.04 \approx 0.52$$

$$O_S = \mu(I_S) = \frac{1}{1 + \exp(-I_S)} = \frac{1}{1 + \exp(-0.52)} \approx 0.63$$

$$I_M = w_{CM}O_C + w_{PM}O_P + w_{AM}O_A + w_{DM}O_D + \theta_M \\ = 0.3 * 0.4 + 0.4 * 0.3 + 0.59 * 0.1 + 0.68 * 0.2 - 0.03 \approx 0.41$$

$$O_M = \mu(I_M) = \frac{1}{1 + \exp(-I_M)} = \frac{1}{1 + \exp(-0.41)} \approx 0.6$$

$$I_L = w_{SL}O_S + w_{ML}O_M + \theta_L = 0.7 * 0.63 + 0.56 * 0.6 - 0.18 \approx 0.6$$

$$O_L = \frac{1}{1 + \exp(-I_L)} = \frac{1}{1 + \exp(-0.6)} \approx 0.65$$

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 sub-section V.2.2.3 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 V.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 manages learning history sub-model. Please see section II.2 for more details about ME and the user modeling system Zebra.

V.2.3. Discovering user interests based on document classification

As aforementioned at the beginning of the chapter V, 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 (V.2.1) and supervised learning classification methods such as support vector machine, decision tree, and neural network (V.2.2). So this sub-section V.2.3 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 V.2.1. Recall that previous sub-sections V.2.1 and V.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 looks for, her/his searching history is shown in following table V.2.3.1:

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 V.2.3.1. User's searching history

The searching history is stored in learning history sub-model; please see the [beginning of this chapter V](#) to know what is stored in learning history sub-model.

This searching history is considered as training dataset (database) for mining maximum frequent itemsets. The keywords (terms) are now considered items. An itemset is constituted of some items. The support of itemset x is defined as the fraction of total transaction which containing x . In practice, support of itemset x is the number of transactions containing x given fixed training dataset. 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 is contained in y ($x \subset y$). The itemset that has k items is called *k-itemset*. Table V.2.3.2 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 V.2.3.2. 1-itemsets

For explanation, as seen in table V.2.3.1, there are 4 transactions such as (Aug 28 10:20:01), (Aug 28 13:00:00), (Aug 29 8:15:01) and (Aug 30 8:15:06) that contain 1-itemset “*computer*” and so, the support of term “*computer*” is 4.

By 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. The training dataset shown in table V.2.3.1 is very simple, in which the largest itemset $\{computer, programming\ language, algorithm, derivative\}$ occurs one time in transaction (Aug 28 10:20:01). Therefore, given $min_sup = 1$, it is possible to infer that such itemset is the maximum frequent itemset that user searches, as shown in below table V.2.3.3:

N _o	maximum frequent itemset
1	<i>computer, programming language, algorithm, derivative</i>

Table V.2.3.3. 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 $C = \{\text{computer science}, \text{math}\}$, a set of terms $T = \{\text{computer}, \text{programming language}, \text{algorithm}, \text{derivative}\}$ and the set of classification rules in table V.2.2.2.6. 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 V.2.3.4.

No.	vector
1	(computer=4, programming language=3, algorithm=1, derivative=2)

Table V.2.3.4. Interesting document vector

The interesting document vector is normalized as in table V.2.3.5. It is easy to recognize that supports inside interesting document vector are normalized.

No.	vector
1	(computer=0.4, programming language=0.3, algorithm=0.1, derivative=0.2)

Table V.2.3.5. 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:

$$\begin{array}{ll} 0 \leq \text{support} < 0.2 & \rightarrow \text{low} \\ 0.2 \leq \text{support} < 0.5 & \rightarrow \text{medium} \\ 0.5 \leq \text{support} & \rightarrow \text{high} \end{array}$$

Normalized document vector is converted into nominal interesting document vector as in table V.2.3.6.

No.	vector
1	(computer=medium, programming language=medium, algorithm=low, derivative=medium)

Table V.2.3.6. 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 V.2.2.2.6. 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 V.2.2.2.6, 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 V.2.2.3.4. After that the output value of output unit specifies the class of the document. For instance, as seen in table V.2.3.5, the normalized interesting document vector $U = (0.4, 0.3, 0.1, 0.2)$ is fed into the trained neural network shown in figure V.2.2.3.4. 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 sub-section [V.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 [V.2.4](#) is the evaluation.

V.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:

- The series of user’s accesses in her/his 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 like trained neural networks 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 section [V.2](#) – how to discover user interests based on document classification. The next section [V.3](#) mentions the last extended function of learning history sub-model – how to construct learner groups.

V.3. Constructing user groups or user communities

As aforementioned at [the beginning of this chapter V](#), 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 sections [V.1](#) and [V.2](#). This section [V.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 her/his model is assigned by 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 V.3.1 and V.3.2 discuss about user model clustering techniques, namely k -mean algorithm for vector model, overlay model, and Bayesian network. In sub-section V.3.2, I propose the formulas so as to compute the dissimilarity of two overlay models or two Bayesian network (Nguyen L. , User Model Clustering, 2014). The k -medoid algorithm and similarity measures such as cosine similarity measure and correlation coefficient are discussed in sub-section V.3.3. Sub-section V.3.4 is the conclusion.

In general, these sections focus on how to construct user groups which is an extended function supported by **mining engine** (ME) and **learning history sub-model**, along with other extended functions such as learning concept recommendation and discovering user interests aforementioned in previous sections V.1 and V.2.

V.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 expressing how much user masters 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 called k *user model clusters*. A user model cluster is a set of similar user models, so user models in the same cluster are dissimilar to ones in other clusters. The dissimilarity of two user models is defined as Euclidean distance between them, as shown in formula V.3.1.1.

$$\begin{aligned} \text{dissim}(U_1, U_2) &= \text{distance}(U_1, U_2) \\ &= \sqrt{(u_{11} - u_{21})^2 + (u_{12} - u_{22})^2 + \dots + (u_{1n} - u_{2n})^2} \end{aligned}$$

Formula V.3.1.1. 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 user model, the dissimilarities between it and k cluster means are computed. Such user model belongs to the cluster to which it is nearest. In other words, if user model U_i belongs to cluster C_j , the dissimilarity between U_i and mean M_j of cluster C_j , denoted $dissim(U_i, M_j)$, is minimal for all clusters.
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. There are two typical terminating conditions (stopping conditions) for k -mean algorithm:

- The k means are not changed. In other words, k clusters are not changed. This condition indicates a perfect clustering task.
- Alternatively, error criterion is less than a pre-defined threshold.

If the stopping condition is that the error criterion is less than a pre-defined threshold, the error criterion is defined by formula V.3.1.2 as follows:

$$Err = \sum_{i=1}^k \sum_{U \in C_i} dissim(U, M_i)$$

Formula V.3.1.2. 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.

The mean M_i of cluster C_i is the center of such cluster, which is the vector whose t^{th} component is the average of t^{th} components of all user model vectors in cluster C_i . Formula V.3.1.3 formulates 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 V.3.1.3. 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 $U_j \in C_i$.

It is necessary to give an example for illustrating the k -mean algorithm with dissimilarity measure. Suppose user models are 2-dimension vectors, figure V.3.1.1 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 -mean algorithm in order to group these vectors into two clusters ($k=2$).

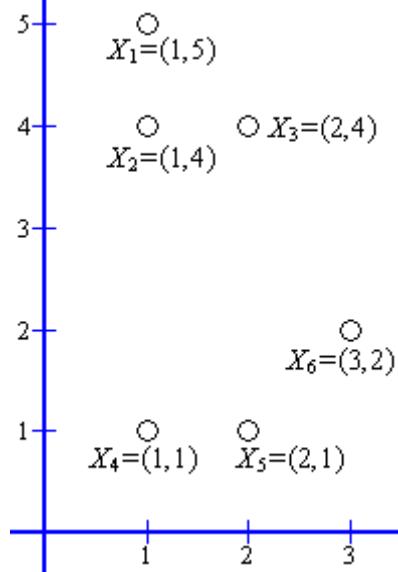


Figure V.3.1.1. A sample of six user vectors

Two vectors X_1 and X_2 are selected arbitrarily as two means M_1 and M_2 , respectively. We have:

$$\begin{aligned} M_1 &= X_1 = (1, 5) \\ M_2 &= X_2 = (1, 4) \end{aligned}$$

Means M_1 and M_2 represent cluster 1 and cluster 2, respectively. According to step 2 of k -mean 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 dissimilarities. For example, according to formula V.3.1.1, the dissimilarity between X_3 and M_1 is:

$$dissim(X_3, M_1) = \sqrt{(2-1)^2 + (4-5)^2} \approx 1.41$$

Table V.3.1.1 shows dissimilarities between user vectors (X_3 , X_4 , X_5 , X_6) and means (M_1 , M_2).

Vector	Cluster	Dissimilarity	Min dissimilarity
$X_3=(2,4)$	$M_1=(1,5)$	$dissim(X_3, M_1) \approx 1.41$	$dissim(X_3, M_2) = 1$
	$M_2=(1,4)$	$dissim(X_3, M_2) = 1$	
$X_4=(1,1)$	$M_1=(1,5)$	$dissim(X_4, M_1) = 4$	$dissim(X_4, M_2) = 3$
	$M_2=(1,4)$	$dissim(X_4, M_2) = 3$	
$X_5=(2,1)$	$M_1=(1,5)$	$dissim(X_5, M_1) \approx 4.12$	$dissim(X_5, M_2) \approx 3.16$
	$M_2=(1,4)$	$dissim(X_5, M_2) \approx 3.16$	
$X_6=(3,2)$	$M_1=(1,5)$	$dissim(X_6, M_1) \approx 3.61$	$dissim(X_6, M_2) \approx 2.83$
	$M_2=(1,4)$	$dissim(X_6, M_2) \approx 2.83$	

Table V.3.1.1. Dissimilarities between user vectors and means

Because $dissim(X_3, M_2)$, $dissim(X_4, M_2)$, $dissim(X_5, M_2)$, and $dissim(X_6, M_2)$ get minimal, 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 -mean algorithm. Of course, X_1 and X_2 belong to cluster 1 and cluster 2, respectively because they are themselves means M_1 and M_2 . Figure V.3.1.2 shows cluster 1 containing X_1 and cluster 2 containing X_2 , X_3 , X_4 , X_5 , X_6 .

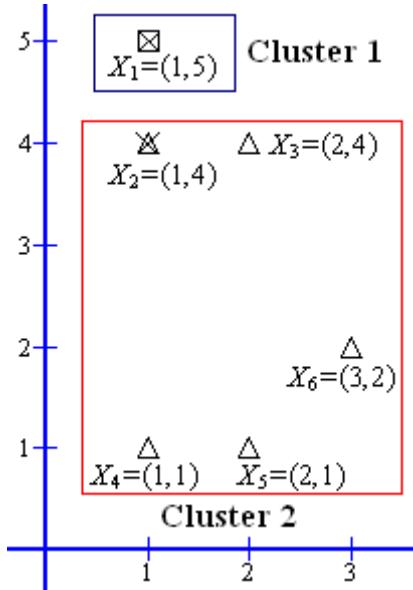


Figure V.3.1.2. Two clusters resulted from k -means algorithm at the first running time

Where rectangles represent vectors that belong to cluster 1 and triangles represent vectors that belong to cluster 2. Means M_1 and M_2 are marked by signs “ \times ”.

According to step 3 of k -mean algorithm, means M_1 and M_2 are re-calculated. The mean M_1 is not changed because cluster 1 contains only $M_1=X_1$. Following formula V.3.1.3, the mean M_2 which is center of cluster 2 is re-calculated as follows:

$$M_2 = \frac{1}{4}(X_3 + X_4 + X_5 + X_6) = \left(\frac{1}{4}(2 + 1 + 2 + 3), \frac{1}{4}(4 + 1 + 1 + 2) \right) = (2, 2)$$

At the second running time of step 2 of k -mean algorithm, it is required to determine which cluster each user vector belongs to, based on dissimilarity measure. Table V.3.1.2 shows dissimilarities between user vectors ($X_1, X_2, X_3, X_4, X_5, X_6$) and means (M_1, M_2).

Vector	Cluster	Dissimilarity	Min dissimilarity
$X_1=(1,5)$	$M_1=(1,5)$	$dissim(X_1, M_1)=0$	$dissim(X_1, M_1)=0$
	$M_2=(2,2)$	$dissim(X_1, M_2) \approx 3.16$	
$X_2=(1,4)$	$M_1=(1,5)$	$dissim(X_2, M_1)=1$	$dissim(X_2, M_1)=1$
	$M_2=(2,2)$	$dissim(X_2, M_2) \approx 2.24$	
$X_3=(2,4)$	$M_1=(1,5)$	$dissim(X_3, M_1) \approx 1.41$	$dissim(X_3, M_1) \approx 1.41$
	$M_2=(2,2)$	$dissim(X_3, M_2)=2$	
$X_4=(1,1)$	$M_1=(1,5)$	$dissim(X_4, M_1)=4$	$dissim(X_4, M_2) \approx 1.41$
	$M_2=(2,2)$	$dissim(X_4, M_2) \approx 1.41$	
$X_5=(2,1)$	$M_1=(1,5)$	$dissim(X_5, M_1) \approx 4.12$	$dissim(X_5, M_2)=1$
	$M_2=(2,2)$	$dissim(X_5, M_2)=1$	
$X_6=(3,2)$	$M_1=(1,5)$	$dissim(X_6, M_1) \approx 3.61$	$dissim(X_6, M_2)=1$
	$M_2=(2,2)$	$dissim(X_6, M_2)=1$	

Table V.3.1.2. Dissimilarities between user vectors and means at the second running time of k -mean algorithm

Because $dissim(X_1, M_1)$, $dissim(X_2, M_1)$, $dissim(X_3, M_1)$, $dissim(X_4, M_2)$, $dissim(X_5, M_2)$, and $dissim(X_6, M_2)$ get maximal, it is easy to infer that X_1 , X_2 , X_3 belong to cluster 1 and X_4 , X_5 , X_6 belong to cluster 2, respectively. 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 -mean algorithm, means M_1 and M_2 are re-calculated. Following formula V.3.1.3, means M_1 and M_2 which are centers of clusters 1 and 2, respectively are re-calculated as follows:

$$M_1 = \frac{1}{3}(X_1 + X_2 + X_3) = \left(\frac{1}{3}(1 + 1 + 2), \frac{1}{3}(5 + 4 + 4) \right) \approx (1.33, 4.33)$$

$$M_2 = \frac{1}{3}(X_4 + X_5 + X_6) = \left(\frac{1}{3}(1 + 2 + 3), \frac{1}{3}(1 + 1 + 2) \right) \approx (2, 1.33)$$

Figure V.3.1.3 shows cluster 1 containing X_1 , X_2 , X_3 and cluster 2 containing X_4 , X_5 , X_6 .

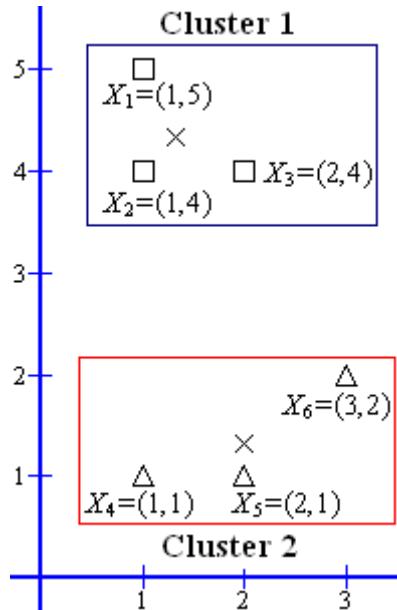


Figure V.3.1.3. Two clusters resulted from k -mean algorithm at the second running time

Where rectangles represent vectors that belong to cluster 1 and triangles represent vectors that belong to cluster 2. Means M_1 and M_2 are marked by signs “ \times ”.

At the third running time of step 3 of k -mean algorithm, it is required to determine which cluster each user vector belongs to, based on dissimilarity measure. Table V.3.1.3 shows dissimilarities between user vectors (X_1 , X_2 , X_3 , X_4 , X_5 , X_6) and means (M_1 , M_2).

Vector	Cluster	Dissimilarity	Min dissimilarity
$X_1=(1,5)$	$M_1=(1.33,4.33)$	$dissim(X_1, M_1) \approx 0.75$	$dissim(X_1, M_1) \approx 0.75$
	$M_2=(2,1.33)$	$dissim(X_1, M_2) \approx 3.8$	
$X_2=(1,4)$	$M_1=(1.33,4.33)$	$dissim(X_2, M_1) \approx 0.47$	$dissim(X_2, M_1) \approx 0.47$
	$M_2=(2,1.33)$	$dissim(X_2, M_2) \approx 2.85$	
$X_3=(2,4)$	$M_1=(1.33,4.33)$	$dissim(X_3, M_1) \approx 0.75$	$dissim(X_3, M_1) \approx 0.75$
	$M_2=(2,1.33)$	$dissim(X_3, M_2) = 2.67$	

$X_4=(1,1)$	$M_1=(1.33,4.33)$	$dissim(X_4,M_1) \approx 3.35$	$dissim(X_4,M_2) \approx 1.05$
	$M_2=(2,1.33)$	$dissim(X_4,M_2) \approx 1.05$	
$X_5=(2,1)$	$M_1=(1.33,4.33)$	$dissim(X_5,M_1) \approx 3.4$	$dissim(X_5,M_2) = 0.33$
	$M_2=(2,1.33)$	$dissim(X_5,M_2) = 0.33$	
$X_6=(3,2)$	$M_1=(1.33,4.33)$	$dissim(X_6,M_1) \approx 2.87$	$dissim(X_6,M_2) \approx 1.2$
	$M_2=(2,1.33)$	$dissim(X_6,M_2) \approx 1.2$	

Table V.3.1.3. Dissimilarities between user vectors and means at the third running time of k -mean algorithm

Because $dissim(X_1,M_1)$, $dissim(X_2,M_1)$, $dissim(X_3,M_1)$, $dissim(X_4,M_2)$, $dissim(X_5,M_2)$, and $dissim(X_6,M_2)$ get maximal, it is easy to infer that X_1 , X_2 , X_3 belong to cluster 1 and X_4 , X_5 , X_6 belong to cluster 2, respectively. It is easy to recognize that there is no change in clusters. Of course, means M_1 and M_2 are not changed. According to step 3, the k -mean algorithm is stopped. Therefore, the figure V.3.1.3 shows the final result of k -mean algorithm given our sample. We have:

- Cluster 1: $X_1=(1, 5)$, $X_2=(1, 4)$, $X_3=(2, 4)$ with mean $M_1=(1.33, 4.33)$.
- Cluster 2: $X_4=(1, 1)$, $X_5=(2, 1)$, $X_6=(3, 2)$ with mean $M_2=(2, 1.33)$.

In general, 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-section V.3.2 mentions solutions of clustering overlay models.

V.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 V.3.2.1 which is a replication of figure I.1.2.2.1 depicts an example of overlay model; please see sub-section I.1.2.2 for more details about overlay model.

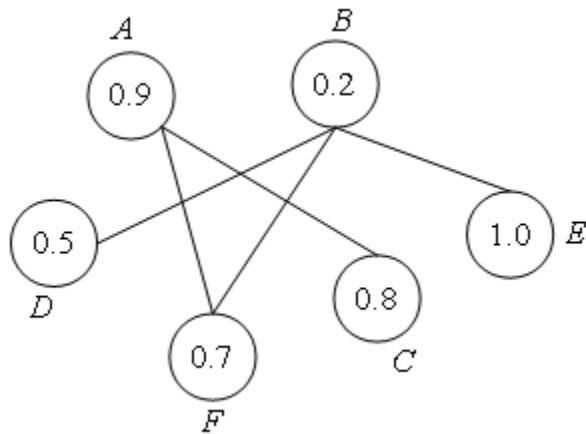


Figure V.3.2.1. Overlay model

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 much 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.1.2 and III.1.3 for more details about aggregation relationship. Figure V.3.2.2 depicts a graph model in form of tree.

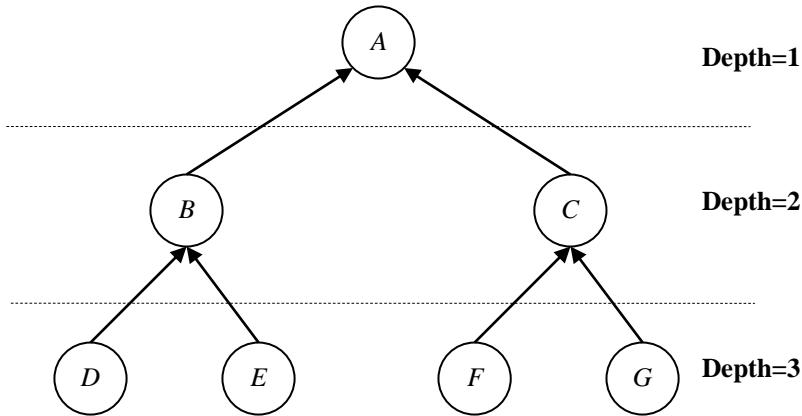


Figure V.3.2.2. Graph model in form of tree and relationships

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 V.3.2.2, 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 V.3.2.1 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 V.3.2.1. 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 [V.3.2.1](#) 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 [V.3.2.1](#) for specifying the dissimilarity of two graphs is only applied into the case that graph satisfies the condition that graph must be the multi-depth tree like the one depicted in figure [V.3.2.2](#).

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 [V.3.2.2](#) but their node values are different. The values of their nodes are shown in following table [V.3.2.1](#):

	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 V.3.2.1. Values of graph nodes

According to formula [V.3.2.1](#), the dissimilarities (or distances) between G_3 and G_1 , G_2 , respectively are computed as below:

$$\begin{aligned} \text{dissim}(G_1, G_3) &= \frac{|2 - 2|}{1} + \frac{|1 - 1|}{2} + \frac{|1 - 1|}{2} + \frac{|0 - 1|}{3} + \frac{|3 - 4|}{3} + \frac{|2 - 5|}{3} \\ &\quad + \frac{|1 - 4|}{3} = 0 + 0 + 0 + \frac{1}{3} + \frac{1}{3} + 1 + 1 \approx 2.66 \\ \text{dissim}(G_2, G_3) &= \frac{|1 - 2|}{1} + \frac{|1 - 1|}{2} + \frac{|0 - 1|}{2} + \frac{|1 - 1|}{3} + \frac{|4 - 4|}{3} + \frac{|5 - 5|}{3} \\ &\quad + \frac{|4 - 4|}{3} = 1 + 0 + 0.5 + 0 + 0 + 0 + 0 = 1.5 \end{aligned}$$

So G_2 is more similar to G_3 than G_1 is because $\text{dissim}(G_2, G_3)$ is smaller than $\text{dissim}(G_1, G_3)$.

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](#) for more details about BN.

Dissimilarity measure relevant to such two cases are mentioned in successive sub-sections [V.3.2.1](#) and [V.3.2.2](#).

V.3.2.1. In case that arcs in graph are weighted

In case that each arc is assigned by a weight representing the strength of relationship between parent node and child node, the figure [V.3.2.1.1](#) shows this situation:

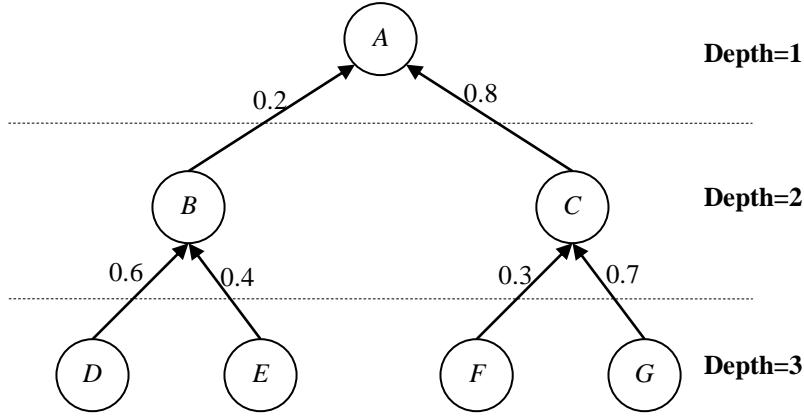


Figure V.3.2.1.1. Graph model and weighted arcs

The structure of graph model in figure V.3.2.1.1 is the same to the one in figure V.3.2.2 except that arcs are weighted. Recall that each node is a variable.

The dissimilarity (or distance) of two weighted graph models G_1 and G_2 is re-defined by formula V.3.2.1.1 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 V.3.2.1.1. 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 sub-section V.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 V.3.2.1.1 becomes formula V.3.2.1.2:

$$dissim(G_1, G_2) = distance(G_1, G_2) = \sum_{j=1}^n \frac{weight(v_{1j})}{depth(v_{1j})}$$

Formula V.3.2.1.2. Dissimilarity of two weighted graph models without node values

In general, formulas [V.3.2.1.1](#) and [V.3.2.1.2](#) 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 [V.3.2.2](#) and [V.3.2.1.1](#).
- 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 sub-section [III.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 figures [V.3.2.2](#) and [V.3.2.1.1](#). The weights of their arcs are expressed in figure [V.3.2.1.1](#). The values of their nodes were shown in table [V.3.2.1](#), 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 [V.3.2.1.1](#) into these node values, dissimilarity measures between G_3 and G_1 , G_2 , respectively are computed as below:

$$dissim(G_1, G_3)$$

$$\begin{aligned} &= \frac{|2 - 2|}{1} * 1 + \frac{|1 - 1|}{2} * 0.2 + \frac{|1 - 1|}{2} * 0.8 + \frac{|0 - 1|}{3} * 0.6 \\ &\quad + \frac{|3 - 4|}{3} * 0.4 + \frac{|2 - 5|}{3} * 0.3 + \frac{|1 - 4|}{3} * 0.7 \\ &= 0 + 0 + 0 + 0.2 + \frac{0.4}{3} + 0.3 + 0.7 \approx 1.33 \end{aligned}$$

$$dissim(G_2, G_3)$$

$$\begin{aligned} &= \frac{|1 - 2|}{1} * 1 + \frac{|1 - 1|}{2} * 0.2 + \frac{|0 - 1|}{2} * 0.8 + \frac{|1 - 1|}{3} * 0.6 \\ &\quad + \frac{|4 - 4|}{3} * 0.4 + \frac{|5 - 5|}{3} * 0.3 + \frac{|4 - 4|}{3} * 0.7 \\ &= 1 + 0 + 0.4 + 0 + 0 + 0 + 0 = 1.4 \end{aligned}$$

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 [V.3.2.1.1](#) will be modified when graph model is evolved as Bayesian network. The successive sub-section [V.3.2.2](#) describes how to specify dissimilarity measure in case that graph model is Bayesian network.

V.3.2.2. In case that graph model is Bayesian network

Bayesian network (BN) is the directed acrylic 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](#) for more details about BN.

Suppose the structure of BN is in form of tree, figure V.3.2.2.1 shows our considered BN.

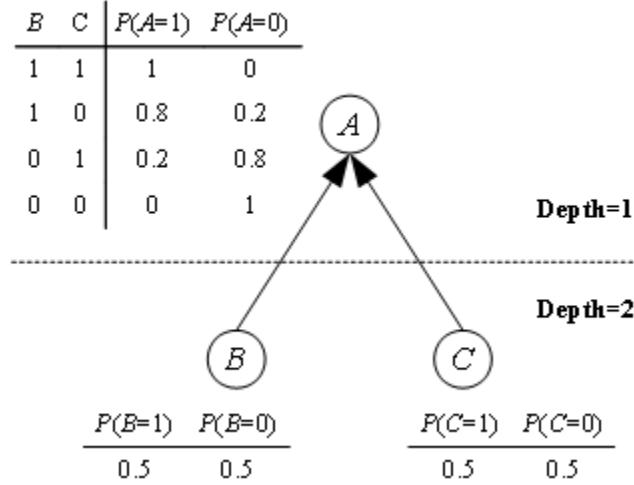


Figure V.3.2.2.1. Bayesian network and its CPT (s)

It is easy to recognize that BN in form of tree as shown in figure V.3.2.2.1 is a [sigma graph](#); please see sub-section III.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.

$$U_1 = G_1 = \langle V_1, E_1 \rangle \text{ and } U_2 = G_2 = \langle V_2, E_2 \rangle$$

Where V_i and E_i are 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 V.3.2.2.1 as follows:

$$dissim(G_1, G_2) = distance(G_1, G_2) = \sum_{j=1}^n \frac{|P(v_{1j}) - P(v_{2j})|}{depth(v_{1j})}$$

Formula V.3.2.2.1. 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 for more details about BN and posterior probability.

In general, formula V.3.2.2.1 for specifying the dissimilarity of two BN (s) is only applied into the case that BN satisfies the condition that the structure of BN must be [sigma graph](#) mentioned in sub-section III.1.3.

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 V.3.2.2.1. The values of their nodes are shown in table V.3.2.2.1 as follows:

	<i>A</i>	<i>B</i>	<i>C</i>
G_1	1	1	1
G_2	1	1	0
G_3	0	0	1

Table V.3.2.2.1. 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 $depth(v_{11}) = depth(v_{21}) = depth(v_{31}) = depth(A) =$

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 V.3.2.2.1, the joint probability distribution is:

$$P(A, B, C) = P(B) * P(C) * P(A|B, C)$$

Please see formula III.1.1.8 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 V.3.2.2.1. We have:

$$\begin{aligned} & P(A = 1) \\ &= P(A = 1, B = 1, C = 1) + P(A = 1, B = 1, C = 0) + P(A = 1, B = 0, C = 1) \\ &\quad + P(A = 1, B = 0, C = 0) \\ &= P(B = 1) * P(C = 1) * P(A = 1|B = 1, C = 1) \\ &\quad + P(B = 1) * P(C = 0) * P(A = 1|B = 1, C = 0) \\ &\quad + P(B = 0) * P(C = 1) * P(A = 1|B = 0, C = 1) \\ &\quad + P(B = 0) * P(C = 0) * P(A = 1|B = 0, C = 0) \\ &= 0.5 * 0.5 * 1 + 0.5 * 0.5 * 0.8 + 0.5 * 0.5 * 0.2 + 0.5 * 0.5 * 0 = 0.5 \end{aligned}$$

Suppose network G_1 has two evidences $B=1$ and $C=1$, we have:

$$P(v_{12}) = P(B = 1) = 0.5$$

$$P(v_{13}) = P(C = 1) = 0.5$$

$$\begin{aligned} P(v_{11}) &= P(A = 1|B = 1, C = 1) = \frac{P(A = 1, B = 1, C = 1)}{P(A = 1)} \\ &\quad (\text{by applying formula III.1.1.10 for specifying posterior probability}) \\ &= \frac{P(B = 1) * P(C = 1) * P(A = 1|B = 1, C = 1)}{P(A = 1)} = \frac{0.5 * 0.5 * 1}{0.5} = 0.5 \end{aligned}$$

Suppose network G_2 has two evidences $B=1$ and $C=0$, we have:

$$P(v_{22}) = P(B = 1) = 0.5$$

$$P(v_{23}) = P(C = 0) = 1 - P(C = 1) = 1 - 0.5 = 0.5$$

$$P(v_{21}) = P(A = 1|B = 1, C = 0) = \frac{P(A = 1, B = 1, C = 0)}{P(A = 1)}$$

$$\begin{aligned} &\quad (\text{by applying formula III.1.1.10 for specifying posterior probability}) \\ &= \frac{P(B = 1) * P(C = 0) * P(A = 1|B = 1, C = 0)}{P(A = 1)} = \frac{0.5 * 0.5 * 0.8}{0.5} = 0.4 \end{aligned}$$

Suppose network G_3 has two evidences $B=0$ and $C=1$, we have:

$$P(v_{32}) = P(B = 0) = 1 - P(B = 1) = 1 - 0.5 = 0.5$$

$$P(v_{33}) = P(C = 1) = 0.5$$

$$P(v_{31}) = P(A = 1|B = 0, C = 1) = \frac{P(A = 1, B = 0, C = 1)}{P(A = 1)}$$

$$\begin{aligned} &\quad (\text{by applying formula III.1.1.10 for specifying posterior probability}) \\ &= \frac{P(B = 0) * P(C = 1) * P(A = 1|B = 0, C = 1)}{P(A = 1)} = \frac{0.5 * 0.5 * 0.2}{0.5} = 0.1 \end{aligned}$$

Applying formula V.3.2.2.1 into node values shown in table V.3.2.2.1, dissimilarity measures between G_3 and G_1 , G_2 , respectively are computed as below:

$$\begin{aligned} \text{dissim}(G_1, G_3) &= \frac{|P(v_{11}) - P(v_{31})|}{\text{depth}(v_{11})} + \frac{|P(v_{12}) - P(v_{32})|}{\text{depth}(v_{12})} + \frac{|P(v_{13}) - P(v_{33})|}{\text{depth}(v_{12})} \\ &= \frac{|0.5 - 0.1|}{1} + \frac{|0.5 - 0.5|}{2} + \frac{|0.5 - 0.5|}{2} = 0.4 \end{aligned}$$

$$\begin{aligned} dissim(G_2, G_3) &= \frac{|P(v_{21}) - P(v_{31})|}{depth(v_{11})} + \frac{|P(v_{22}) - P(v_{32})|}{depth(v_{12})} + \frac{|P(v_{23}) - P(v_{33})|}{depth(v_{12})} \\ &= \frac{|0.4 - 0.1|}{1} + \frac{|0.5 - 0.5|}{2} + \frac{|0.5 - 0.5|}{2} = 0.3 \end{aligned}$$

So G_2 is more similar to G_3 than G_1 is because $dissim(G_2, G_3)$ is smaller than $dissim(G_1, G_3)$.

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 V.3.3 mentions similarity measure and how to modify k -mean algorithm so as to take advantages of similarity measure.

V.3.3. Similarity measures for clustering algorithms

Although dissimilarity measure considered as the physical distance between two objects is used to cluster user models, we can use another measure 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 V.3.3.1 as follows:

$$sim(U_i, U_j) = cosine(U_i, U_j) = \frac{U_i \circ U_j}{|U_i| |U_j|} = \frac{\sum_{k=1}^n u_{ik} u_{jk}}{\sqrt{\sum_{k=1}^n u_{ik}^2} \sqrt{\sum_{k=1}^n u_{jk}^2}}$$

Formula V.3.3.1. Cosine similarity measure

Where the sign “ \circ ” denotes scalar product (dot product) of two vectors and the notation $|.|$ denotes the length of a vector. Previous sub-section V.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 V.3.3.1 shows four user models.

	feature ₁	feature ₂	feature ₃
user ₁	1	2	1
user ₂	2	1	2
user ₃	4	1	5
user ₄	1	2	0

Table V.3.3.1. Four user models

It is easy to interpret table V.3.3.1 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:

$$sim(user_4, user_1) = \frac{1 * 1 + 2 * 2 + 0 * 1}{\sqrt{1^2 + 2^2 + 0^2} \sqrt{1^2 + 2^2 + 1^2}} \approx 0.91$$

$$\begin{aligned} sim(user_4, user_2) &= \frac{1 * 2 + 2 * 1 + 0 * 2}{\sqrt{1^2 + 2^2 + 0^2} \sqrt{2^2 + 1^2 + 2^2}} \approx 0.60 \\ sim(user_4, user_3) &= \frac{1 * 4 + 2 * 1 + 0 * 5}{\sqrt{1^2 + 2^2 + 0^2} \sqrt{4^2 + 1^2 + 5^2}} \approx 0.41 \end{aligned}$$

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 V.3.3.2 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 V.3.3.2. 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:

$$\begin{aligned} \bar{U}_4 &= \frac{1 + 2 + 0}{3} = 1 \\ \bar{U}_1 &= \frac{1 + 2 + 1}{3} \approx 1.33 \\ \bar{U}_2 &= \frac{2 + 1 + 2}{3} \approx 1.66 \\ \bar{U}_3 &= \frac{4 + 1 + 5}{3} \approx 3.33 \end{aligned}$$

$$\begin{aligned} sim(user_4, user_1) &= \frac{(1 - 1)(1 - 1.33) + (2 - 1)(2 - 1.33) + (0 - 1)(1 - 1.33)}{\sqrt{(1 - 1)^2 + (2 - 1)^2 + (0 - 1)^2} \sqrt{(1 - 1.33)^2 + (2 - 1.33)^2 + (1 - 1.33)^2}} \\ &\approx 0.86 \end{aligned}$$

$$\begin{aligned} sim(user_4, user_2) &= \frac{(1 - 1)(2 - 1.66) + (2 - 1)(1 - 1.66) + (0 - 1)(2 - 1.66)}{\sqrt{(1 - 1)^2 + (2 - 1)^2 + (0 - 1)^2} \sqrt{(2 - 1.66)^2 + (1 - 1.66)^2 + (2 - 1.66)^2}} \\ &\approx -0.86 \end{aligned}$$

$$\begin{aligned}
 & sim(user_4, user_2) \\
 &= \frac{(1 - 1)(4 - 3.33) + (2 - 1)(1 - 3.33) + (0 - 1)(5 - 3.33)}{\sqrt{(1 - 1)^2 + (2 - 1)^2 + (0 - 1)^2} \sqrt{(4 - 3.33)^2 + (1 - 3.33)^2 + (5 - 3.33)^2}} \\
 &\approx -0.96
 \end{aligned}$$

According to correlation coefficient, 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 $sim(user_4, user_3)$.

If cosine measure and correlation coefficient are used as the similarity between two user vectors, the serious problem will occur in k -mean algorithm when clustering vectors. 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 L. , 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 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 terminating condition is satisfied, the algorithm is terminated; otherwise returning step 2. There are three optional terminating condition: “ k medoids are not changed”, “global average similarity mentioned later is equal to or larger than a pre-defined threshold”, “the error criterion is less than a pre-defined threshold”.

Figure V.3.3.1 is the flow chart of k -medoid algorithm modified with similarity measure:

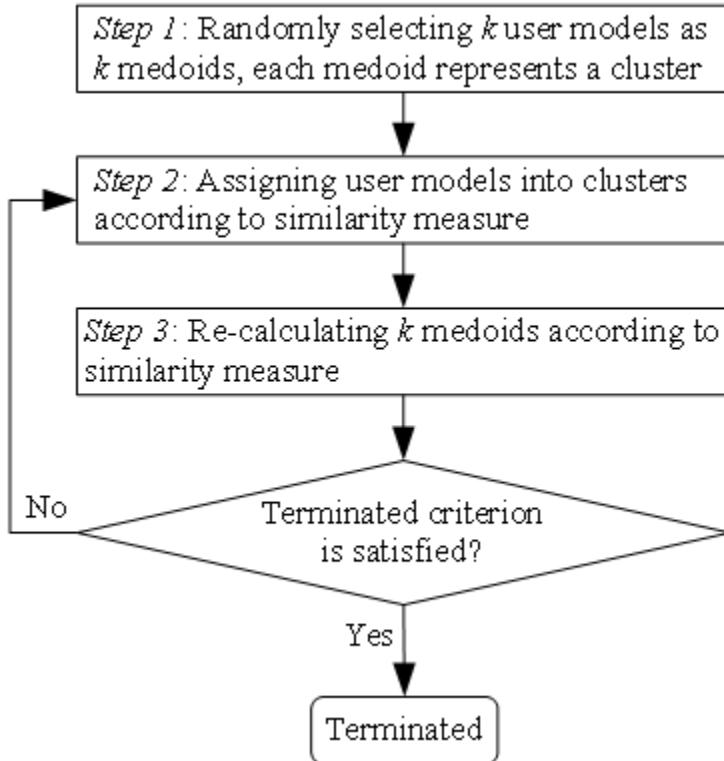


Figure V.3.3.1. Flow chart of k -medoid algorithm modified with similarity measure

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 V.3.3.3. Average similarity

The average similarity a_i of medoid M_i specified by formula V.3.3.3 is used in step 3 of k -medoid algorithm.

The *global average similarity* over k clusters is the average of all similarities between user models and medoids M_i (s). Let A be global average similarity, formula V.3.3.4 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 V.3.3.4. 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 V.3.3.1 and V.3.3.2 for more details about cosine measure and correlation coefficient.

As seen in formula V.3.3.4, 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 three alternative terminating conditions (stopping conditions) for k -medoid algorithm:

- The k medoids are not changed. In other words, k clusters are not changed. This condition indicates a perfect clustering task.
- The global average similarity is equal to or larger than a pre-defined threshold.
- 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 V.3.3.4.

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 V.3.3.2 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 ($k=2$).

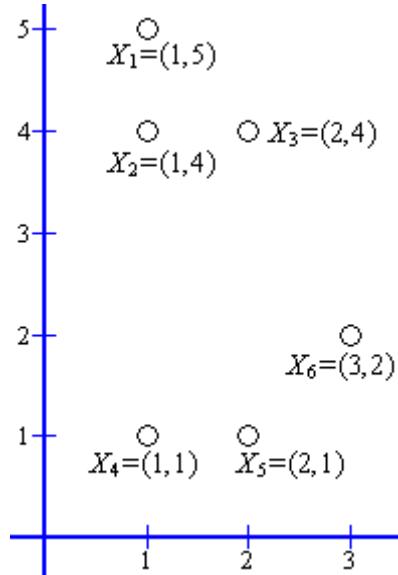


Figure V.3.3.2. A sample of six user vectors

Two vectors X_1 and X_2 are selected arbitrarily as two medoids M_1 and M_2 , respectively. We have:

$$M_1 = X_1 = (1,5)$$

$$M_2 = X_2 = (1,4)$$

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 V.3.3.1, 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 V.3.3.2 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 V.3.3.2. 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 cluster 2, respectively because they are themselves medoids M_1 and M_2 . Figure V.3.3.3 shows cluster 1 containing X_1 and cluster 2 containing X_2, X_3, X_4, X_5, X_6 .

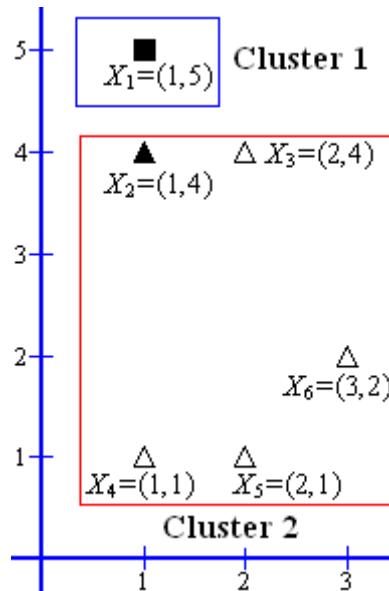


Figure V.3.3.3. Two clusters resulted from k -medoid algorithm at the first running time

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

formula V.3.3.3 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. If X_2 is the medoid, we have:

$$\begin{aligned} a_2 &= \frac{1}{4}(sim(X_2, X_3) + sim(X_2, X_4) + sim(X_2, X_5) + sim(X_2, X_6)) \\ &= \frac{1}{4}(cosine(X_2, X_3) + cosine(X_2, X_4) + cosine(X_2, X_5) \\ &\quad + cosine(X_2, X_6)) \\ &= \frac{1}{4}\left(\frac{1 * 2 + 4 * 4}{\sqrt{1^2 + 4^2}\sqrt{2^2 + 4^2}} + \frac{1 * 1 + 4 * 1}{\sqrt{1^2 + 4^2}\sqrt{1^2 + 1^2}} + \frac{1 * 2 + 4 * 1}{\sqrt{1^2 + 4^2}\sqrt{2^2 + 1^2}}\right. \\ &\quad \left.+ \frac{1 * 3 + 4 * 2}{\sqrt{1^2 + 4^2}\sqrt{3^2 + 2^2}}\right) \approx 0.806 \end{aligned}$$

Similarly, we have:

$$\begin{aligned} a_3 &= \frac{1}{4}(sim(X_3, X_2) + sim(X_3, X_4) + sim(X_3, X_5) + sim(X_3, X_6)) \\ &= \frac{1}{4}\left(\frac{2 * 1 + 4 * 4}{\sqrt{2^2 + 4^2}\sqrt{1^2 + 4^2}} + \frac{2 * 1 + 4 * 1}{\sqrt{2^2 + 4^2}\sqrt{1^2 + 1^2}} + \frac{2 * 2 + 4 * 1}{\sqrt{2^2 + 4^2}\sqrt{2^2 + 1^2}}\right. \\ &\quad \left.+ \frac{2 * 3 + 4 * 2}{\sqrt{2^2 + 4^2}\sqrt{3^2 + 2^2}}\right) \approx 0.898 \end{aligned}$$

$$\begin{aligned} a_4 &= \frac{1}{4}(sim(X_4, X_2) + sim(X_4, X_3) + sim(X_4, X_5) + sim(X_4, X_6)) \\ &= \frac{1}{4}\left(\frac{1 * 1 + 1 * 4}{\sqrt{1^2 + 1^2}\sqrt{1^2 + 4^2}} + \frac{1 * 2 + 1 * 4}{\sqrt{1^2 + 1^2}\sqrt{2^2 + 4^2}} + \frac{1 * 2 + 1 * 1}{\sqrt{1^2 + 1^2}\sqrt{2^2 + 1^2}}\right. \\ &\quad \left.+ \frac{1 * 3 + 1 * 2}{\sqrt{1^2 + 1^2}\sqrt{3^2 + 2^2}}\right) \approx 0.934 \end{aligned}$$

$$\begin{aligned} a_5 &= \frac{1}{4}(sim(X_5, X_2) + sim(X_5, X_3) + sim(X_5, X_4) + sim(X_5, X_6)) \\ &= \frac{1}{4}\left(\frac{2 * 1 + 1 * 4}{\sqrt{2^2 + 1^2}\sqrt{1^2 + 4^2}} + \frac{2 * 2 + 1 * 4}{\sqrt{2^2 + 1^2}\sqrt{2^2 + 4^2}} + \frac{2 * 1 + 1 * 1}{\sqrt{2^2 + 1^2}\sqrt{1^2 + 1^2}}\right. \\ &\quad \left.+ \frac{2 * 3 + 1 * 2}{\sqrt{2^2 + 1^2}\sqrt{3^2 + 2^2}}\right) \approx 0.848 \end{aligned}$$

$$\begin{aligned} a_6 &= \frac{1}{4}(sim(X_6, X_2) + sim(X_6, X_3) + sim(X_6, X_4) + sim(X_6, X_5)) \\ &= \frac{1}{4}\left(\frac{3 * 1 + 2 * 4}{\sqrt{3^2 + 2^2}\sqrt{1^2 + 4^2}} + \frac{3 * 2 + 2 * 4}{\sqrt{3^2 + 2^2}\sqrt{2^2 + 4^2}} + \frac{3 * 1 + 2 * 1}{\sqrt{3^2 + 2^2}\sqrt{1^2 + 1^2}}\right. \\ &\quad \left.+ \frac{3 * 2 + 2 * 1}{\sqrt{3^2 + 2^2}\sqrt{2^2 + 1^2}}\right) \approx 0.895 \end{aligned}$$

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 V.3.3.3 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
$X_2=(1,4)$	$M_1=(1,5)$	$cosine(X_2, M_1) \approx 1$	$cosine(X_2, M_1) \approx 1$

	$M_2=(1,1)$	$\text{cosine}(X_2, M_2) \approx 0.86$	
$X_3=(2,4)$	$M_1=(1,5)$	$\text{cosine}(X_3, M_1) \approx 0.96$	$\text{cosine}(X_3, M_1) \approx 0.96$
	$M_2=(1,1)$	$\text{cosine}(X_3, M_2) \approx 0.95$	
$X_5=(2,1)$	$M_1=(1,5)$	$\text{cosine}(X_5, M_1) \approx 0.61$	$\text{cosine}(X_5, M_2) \approx 0.95$
	$M_2=(1,1)$	$\text{cosine}(X_5, M_2) \approx 0.95$	
$X_6=(3,2)$	$M_1=(1,5)$	$\text{cosine}(X_6, M_1) \approx 0.71$	$\text{cosine}(X_6, M_2) \approx 0.98$
	$M_2=(1,1)$	$\text{cosine}(X_6, M_2) \approx 0.98$	

Table V.3.3.3. 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 re-calculated. 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:

$$a_1 = \frac{1}{2}(\text{sim}(X_1, X_2) + \text{sim}(X_1, X_3)) = \frac{1}{2}\left(\frac{1 * 1 + 5 * 4}{\sqrt{1^2 + 5^2}\sqrt{1^2 + 4^2}} + \frac{1 * 2 + 5 * 4}{\sqrt{1^2 + 5^2}\sqrt{2^2 + 4^2}}\right) \approx 0.982$$

$$a_2 = \frac{1}{2}(\text{sim}(X_2, X_1) + \text{sim}(X_2, X_3)) = \frac{1}{2}\left(\frac{1 * 1 + 4 * 5}{\sqrt{1^2 + 4^2}\sqrt{1^2 + 5^2}} + \frac{1 * 2 + 4 * 4}{\sqrt{1^2 + 4^2}\sqrt{2^2 + 4^2}}\right) \approx 0.988$$

$$a_3 = \frac{1}{2}(\text{sim}(X_3, X_1) + \text{sim}(X_3, X_2)) = \frac{1}{2}\left(\frac{2 * 1 + 4 * 5}{\sqrt{2^2 + 4^2}\sqrt{1^2 + 5^2}} + \frac{2 * 1 + 4 * 4}{\sqrt{2^2 + 4^2}\sqrt{1^2 + 4^2}}\right) \approx 0.970$$

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:

$$a_4 = \frac{1}{2}(\text{sim}(X_4, X_5) + \text{sim}(X_4, X_6)) = \frac{1}{2}\left(\frac{1 * 2 + 1 * 1}{\sqrt{1^2 + 1^2}\sqrt{2^2 + 1^2}} + \frac{1 * 3 + 1 * 2}{\sqrt{1^2 + 1^2}\sqrt{3^2 + 2^2}}\right) \approx 0.965$$

$$a_5 = \frac{1}{2}(\text{sim}(X_5, X_4) + \text{sim}(X_5, X_6)) = \frac{1}{2}\left(\frac{2 * 1 + 1 * 1}{\sqrt{2^2 + 1^2}\sqrt{1^2 + 1^2}} + \frac{2 * 1 + 1 * 1}{\sqrt{2^2 + 1^2}\sqrt{1^2 + 1^2}}\right) \approx 0.970$$

$$a_6 = \frac{1}{2}(\text{sim}(X_6, X_4) + \text{sim}(X_6, X_5)) = \frac{1}{2}\left(\frac{3 * 1 + 2 * 1}{\sqrt{3^2 + 2^2}\sqrt{1^2 + 1^2}} + \frac{3 * 2 + 2 * 1}{\sqrt{3^2 + 2^2}\sqrt{2^2 + 1^2}}\right) \approx 0.986$$

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 V.3.3.4 shows cluster 1 containing X_1 , X_2 , X_3 and cluster 2 containing X_4 , X_5 , X_6 .

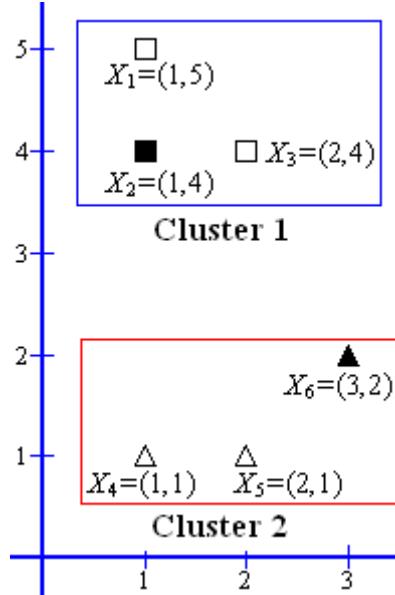


Figure V.3.3.4. Two clusters resulted from k -medoid algorithm at the second running time

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 V.3.3.4 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 V.3.3.4. 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 V.3.3.4 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 V.3.3 which ends up main contents of constructing user groups based on clustering user

models, in turn. The successive sub-section [V.3.4](#) is the evaluation of the third extended function of learning history sub-model that is constructing user groups – the main subject of section [V.3](#).

V.3.4. Evaluation

It is easy to infer that the essence of constructing user groups is to cluster user models and so this section [V.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 chapter [V](#). Three extended functions of learning history sub-model such as learning concept recommendation, discovering user interests and constructing user groups were mentioned successively in sections [V.1](#), [V.2](#), and [V.3](#). The last section [V.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 TLM is constructed and how the user modeling system [Zebra](#) is built up and manipulates TLM. Now the next chapter [VI](#) is the evaluation on TLM.

Chapter VI. Evaluation of Triangular Learner Model

Chapter I is the state-of-art of user model and user modeling system. The 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 which are mentioned in successive chapters III, IV, and V, respectively. Please pay attention to the coherence of such three main chapters together with respective sub-models. Now this chapter VI focuses on the evaluation of TLM.

In general, this research is fundamental research. Thus, approaches, architectures, models and mathematical formulas are proposed in the coherence of fundamental methodologies. 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 models and architectures is authenticated via the computer software which implements Zebra and TLM. Moreover, the adaptive learning system that interacts with Zebra is also implemented as an e-learning web application [WOW](#). The user modeling system Zebra and the e-learning application WOW can be downloaded via internet link <http://zebra.locnguyen.net>. Please see sub-section II.2.4 for more details about the implementation of Zebra and WOW. Moreover, examples, mathematical formulas, and mathematical models inside the research are implemented separately and integrated into another computer software.
- The effectiveness of adaptive learning is proven via approaches described in this chapter VI.

This chapter VI focuses on evaluating learner model TLM and user modeling system Zebra with regard to their effectiveness. Section VI.1 is the evaluation on knowledge sub-model. Section VI.2 is the evaluation on the effectiveness of adaptive learning model, especially, the whole TLM and modeling system Zebra.

VI.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 core of Zebra. Note that the core of Zebra is composition of two engines: belief network engine (BNE) and mining engine (ME). Bayesian network infers and assesses user knowledge while ME aims to provide extended functions of Zebra such as learning concept recommendation (section V.1), discovering user interest (section V.2), and constructing user groups (section V.3). 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-section III.1.1 and section IV.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 VI.1 includes three successive sub-sections:

- Sub-section [VI.1.1](#) aims to evaluate Bayesian model in methodological point of view. For example, this sub-section answers the question “what the aspects of proposed Bayesian model are and how effective the proposed Bayesian model is”.
- Sub-section [VI.1.2](#) mentions how to exploit Bayesian model in practical assessment. For example, this sub-section answers the question “how to apply Bayesian model into determining how much a user masters over a knowledge item”. The research uses Computerized Adaptive Testing (CAT) as an advanced technique to assess user’s knowledge when knowledge items are tests or examinations, which is focused on sub-section [VI.1.3](#). In general, sub-section [VI.1.2](#) gives an overview of assessment of Bayesian network while sub-section [VI.1.3](#) introduces CAT – a concrete method of assessment.

VI.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 in expert-centric model or in 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.1.2](#) 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 section [III.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.4.2.1](#)) 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^{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.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 section [III.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 [VI.1.2](#) will mention how to exploit Bayesian model in practical assessment.

VI.1.2. Assessment of Bayesian network

Besides providing information about users and deducing new assumptions about them, the ultimate purpose 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. 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 such knowledge items. Observed variables called evidences are questions, exercises which are used to test user's knowledge. It is possible to imagine that hidden variables represent 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 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 (CAT) based on

item response theory, in which evidences are test items like exams, questions, etc. CAT will be mentioned in next sub-section VI.1.3.

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 VI.1.2.1 (Mayo, 2001, p. 43).

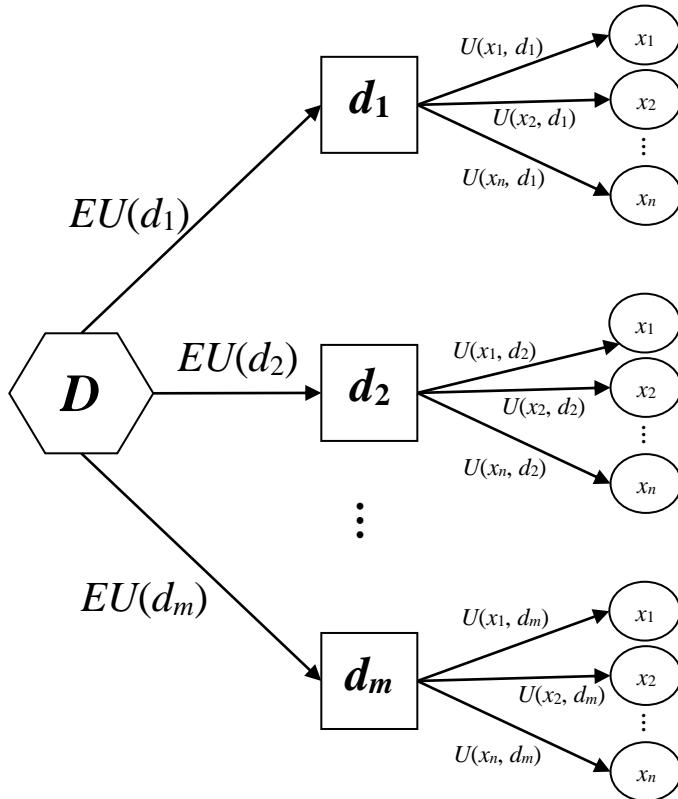


Figure VI.1.2.1. Decision-theoretic tree

The decision-theoretic tree depicted in figure VI.1.2.1 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 $U(X=x_i, D=d_i)$. Of course $P(X=x_j | D=d_i)$ and $U(X=x_i, D=d_i)$ 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 VI.1.2.1 as follows (Mayo, 2001, p. 44):

$$EU(D) = \sum_X P(X|D)U(X,D)$$

Formula VI.1.2.1. 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 alternate 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 . As advancement, [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 technique belongs to diagnostic approach. CAT is described in successive sub-section [VI.1.3](#).

VI.1.3. Towards the Computerized Adaptive Testing

Computer-based testing has more advantages than traditional paper-based testing when there is a 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). CAT is considered as a branch of computer-based testing but it improves the accuracy of test core when CAT systems try to choose items (tests, exams, questions, etc.) 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 the 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 (Nguyen L., The Bayesian approach and suggested stopping criterion in Computerized Adaptive Testing, 2013). 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 [VI.1.3](#) includes three partial sub-sections:

- Sub-section [VI.1.3.1](#) gives an overview of CAT.
- Sub-section [VI.1.3.2](#) describes the proposed MLE method for estimating examinee's ability.

- Sub-section VI.1.3.3 describes advanced CAT algorithm in detailed. Ability error is also mentioned in this sub-section.

VI.1.3.1. 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 θ . Given an examinee and item i , IRT is modeled as a function of a true ability θ of given examinee with three parameters of item i such as a_i , b_i , and c_i . This function 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 VI.1.3.1.1 as follows:

$$IRF(\theta) = P_i(\theta) = c_i + \frac{1 - c_i}{1 + exp(-a_i(\theta_j - b_i))}$$

Formula VI.1.3.1.1. 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. Suppose that a_i is greater than 0.

IRF, a variant of logistic function, is plotted as the curve in following figure VI.1.3.1.1 with $a_i=6$, $b_i=0.4$, $c_i=0.2$.

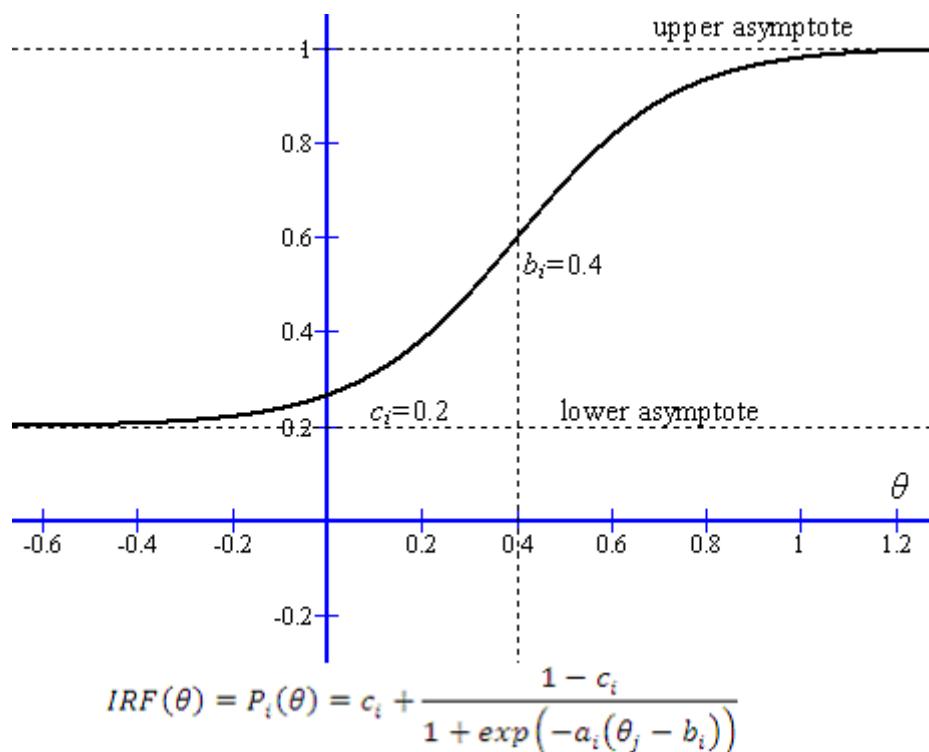


Figure VI.1.3.1.1. Item Response Function curve

The horizontal axis θ 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 VI.1.3.1.1, 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 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 called *difficult parameter* (Rudner, 1998) indicates how difficult the item is. It specifies the location of inflection point of the curve along the θ 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 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 computerized adaptive testing for choosing the best item which is given to examinee and estimating examinee' true ability θ . 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 stopping criterion is met. This algorithm aims to make a series of 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 VI.1.3.1.1:

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 θ of examinee at current time point.

4. Steps 1 through 3 are repeated until stopping criterion is met.

Table VI.1.3.1.1. 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 θ 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 θ ; such *information function* is denoted $I_i(\theta)$. The best next item is the one that gets most informative or provides highest value of $I_i(\theta)$. Formula VI.1.3.1.2 specifies information function for item i (Rudner, 1998).

$$I_i(\theta) = \frac{(P'_i(\theta))^2}{P_i(\theta)(1 - P_i(\theta))}$$

Formula VI.1.3.1.2. 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 VI.1.3.1.1 and $P'_i(\theta)$ is the first-order derivative of $P_i(\theta)$. According to formula VI.1.3.1.1, we have:

$$P_i(\theta) = IRF(\theta) = c_i + \frac{1 - c_i}{1 + \exp(-a_i(\theta - b_i))}$$

$$P'_i(\theta) = \frac{dP_i(\theta)}{d\theta} = \frac{a_i(1 - c_i)\exp(-a_i(\theta - b_i))}{(1 + \exp(-a_i(\theta - b_i)))^2}$$

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 (Linden & Pashley, 2002, pp. 3-7). However, I propose a new method to calculate ability estimate based on maximization likelihood estimation (MLE). The successive sub-section VI.1.3.2 describes a modified MLE method for CAT (Nguyen L. , New version of CAT algorithm by maximum likelihood estimation, 2016).

VI.1.3.2. Maximum likelihood estimation (MLE) for CAT

Let $\hat{\theta}$ be the ability estimate of examinee, the goal of this sub-section VI.1.3.2 is to calculate $\hat{\theta}$. Recall that the ability estimate is very important to step 3 of CAT algorithm; please see table VI.1.3.1.1 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 θ . Note that the probability $P(\theta)$ is IRF function specified by formula VI.1.3.1.1. For convenience, let guessing parameter be zero ($c=0$). It means that the probability that examinee guesses correct response equals 0. Formula VI.1.3.2.1 specifies $P(\theta)$ with $c=0$.

$$P(\theta) = IRF(\theta) = \frac{1}{1 + \exp(-a(\theta - b))}$$

Formula VI.1.3.2.1. IRF with zero guessing parameter

Where a and b are discriminatory parameter and difficult parameter, respectively. Without loss of generality, formula VI.1.3.2.1 implicates that guessing parameter is fixed.

Let θ_0 be the examinee's initial ability. When examinee's initial ability is not determined yet, θ_0 can be initialized by zero, as specified by formula VI.1.3.2.2.

$$\theta_0 = 0$$

Formula VI.1.3.2.2. Examinee's initial ability

Note that it is possible to initialize θ_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:

$$\binom{q_i}{r_i} (P(\theta_0))^{q_i-r_i} (1 - P(\theta_0))^{r_i}$$

Where probability $P(\theta_0)$ is specified by formula VI.1.3.2.1 and θ_0 is examinee's initial ability specified by formula VI.1.3.2.2.

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 VI.1.3.2.3 as follows:

$$\begin{aligned} L(a, b) &= \prod_{i=1}^N \binom{q_i}{r_i} (P(\theta_0))^{q_i-r_i} (1 - P(\theta_0))^{r_i} = \prod_{i=1}^N \binom{q_i}{r_i} (P(\theta_0))^{q_i} \left(\frac{1 - P(\theta_0)}{P(\theta_0)} \right)^{r_i} \\ &= \prod_{i=1}^N \binom{q_i}{r_i} \left(\frac{1}{1 + \exp(ab - a\theta_0)} \right)^{q_i} (\exp(ab - a\theta_0))^{r_i} \end{aligned}$$

Formula VI.1.3.2.3. Likelihood function of examinee's ability

Note, a and b become variables of the likelihood function $L(a, b)$. The notation $\binom{q_i}{r_i}$ denotes combination taken r_i of q_i elements and so we have $\binom{q_i}{r_i} = \frac{q_i!}{r_i!(q_i-r_i)!}$.

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 , respective; 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.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 $L(a, b)$ called log-likelihood function is denoted $LnL(a, b)$. We have:

$$\begin{aligned} LnL(a, b) &= \sum_{i=1}^N \ln \left(\frac{q_i}{r_i} \right) - \sum_{i=1}^N q_i \ln(1 + \exp(ab - a\theta_0)) + \sum_{i=1}^N r_i(ab - a\theta_0) \\ &= \sum_{i=1}^N \ln \left(\frac{q_i}{r_i} \right) + (ab - a\theta_0) \sum_{i=1}^N r_i - \ln(1 + \exp(ab - a\theta_0)) \sum_{i=1}^N q_i \end{aligned}$$

Briefly, formula VI.1.3.2.4 specifies the log-likelihood $LnL(a, b)$.

$$LnL(a, b) = \sum_{i=1}^N \ln \left(\frac{q_i}{r_i} \right) + (ab - a\theta_0) \sum_{i=1}^N r_i - \ln(1 + \exp(ab - a\theta_0)) \sum_{i=1}^N q_i$$

Formula VI.1.3.2.4. Log-likelihood function of examinee's ability

Where $\ln(\cdot)$ denotes natural logarithm function, θ_0 is examinee's initial ability and r_i is examinee's response. The notation $\binom{q_i}{r_i}$ denotes combination taken r_i of q_i elements and so we have $\binom{q_i}{r_i} = \frac{q_i!}{r_i!(q_i-r_i)!}$.

Maximizing the likelihood function is equivalent to maximizing $LnL(a, b)$.

$$(\hat{a}, \hat{b}) = \underset{a,b}{\operatorname{argmax}} L(a, b) = \underset{a,b}{\operatorname{argmax}} LnL(a, b)$$

The maximization can be done by setting first-order partial derivatives of $LnL(a, b)$ 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 $LnL(a, b)$ with respect to parameters a and b are:

$$\begin{aligned} \frac{\partial LnL(a, b)}{\partial a} &= (b - \theta_0) \left(\sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right) \\ \frac{\partial LnL(a, b)}{\partial b} &= a \left(\sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right) \end{aligned}$$

Setting these first-order partial derivatives to 0, we have following equations for solving estimates \hat{a} and \hat{b} .

$$\begin{cases} \frac{\partial LnL(a, b)}{\partial a} = 0 \\ \frac{\partial LnL(a, b)}{\partial b} = 0 \end{cases}$$

$$\Rightarrow \begin{cases} (b - \theta_0) \left(\sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right) = 0 \\ a \left(\sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i \right) = 0 \end{cases}$$

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 ignore the case $b = \theta_0$. Without loss of generality, we have formula VI.1.3.2.5 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 VI.1.3.2.5. 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 VI.1.3.2.5, we have:

$$\begin{aligned} & \sum_{i=1}^N r_i - \frac{\exp(ab - a\theta_0)}{1 + \exp(ab - a\theta_0)} \sum_{i=1}^N q_i = 0 \\ & \Leftrightarrow \sum_{i=1}^N r_i + \exp(ab - a\theta_0) \sum_{i=1}^N r_i - \exp(ab - a\theta_0) \sum_{i=1}^N q_i = 0 \\ & \Leftrightarrow \exp(ab - a\theta_0) \left(\sum_{i=1}^N (q_i - r_i) \right) = \sum_{i=1}^N r_i \\ & \Rightarrow \exp(ab - a\theta_0) = \frac{\sum_{i=1}^N r_i}{\sum_{i=1}^N (q_i - r_i)} \\ & \Rightarrow ab - a\theta_0 = \ln \left(\sum_{i=1}^N r_i \right) - \ln \left(\sum_{i=1}^N (q_i - r_i) \right) \\ & \Rightarrow b = \frac{\ln(\sum_{i=1}^N r_i) - \ln(\sum_{i=1}^N (q_i - r_i))}{a} + \theta_0 \end{aligned}$$

In general, two possible parameter estimates \hat{a} and \hat{b} are specified by formula VI.1.3.2.6 as follows:

$$\begin{cases} \hat{a} \text{ is arbitrary} \\ \hat{b} = \frac{\ln(\sum_{i=1}^N r_i) - \ln(\sum_{i=1}^N (q_i - r_i))}{\hat{a}} + \theta_0 \\ \hat{a} > 0 \end{cases}$$

Formula VI.1.3.2.6. 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 θ_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 VI.1.3.2.1 is essentially cumulative distribution function (CDF). Please see sub-section III.1.1 for more details about cumulative distribution function and probability density function. Let $p(\theta)$ be the probability density function of ability θ , we have:

$$p(\theta) = P'(\theta) = \frac{(a)\exp(-a(\theta - b))}{(1 + \exp(-a(\theta - b)))^2}$$

Formula VI.1.3.2.7. Probability density function of examinee's ability

Formula VI.1.3.2.7 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 VI.1.3.2.6 into formula VI.1.3.2.7, we get the optimal density function $\hat{p}(\theta)$ of examinee's ability, specified by formula VI.1.3.2.8.

$$\hat{p}(\theta) = \frac{(\hat{a})\exp(-\hat{a}(\theta - \hat{b}))}{(1 + \exp(-\hat{a}(\theta - \hat{b})))^2}$$

Formula VI.1.3.2.8. Optimal probability density function of examinee's ability

The ability estimate $\hat{\theta}$ is the expectation of θ given optimal density function $\hat{p}(\theta)$. We have:

$$\begin{aligned} \hat{\theta} &= E(\theta|\hat{p}(\theta)) = \int_{-\infty}^{+\infty} \theta \hat{p}(\theta) d\theta = \int_{-\infty}^{+\infty} \theta \frac{(\hat{a})\exp(-\hat{a}(\theta - \hat{b}))}{(1 + \exp(-\hat{a}(\theta - \hat{b})))^2} d\theta \\ &= \int_{-\infty}^{+\infty} \theta d\left(\frac{1}{1 + \exp(-\hat{a}(\theta - \hat{b}))}\right) \\ &= \frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \Big|_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} \frac{1}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta \end{aligned}$$

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:

$$A = \lim_{\theta \rightarrow +\infty} \left(\frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) - \lim_{\theta \rightarrow -\infty} \left(\frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right)$$

$$\begin{aligned}
 &= \lim_{\theta \rightarrow +\infty} \left(\frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) - \lim_{\theta \rightarrow -\infty} \left(\frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) \\
 &\quad (\text{using L'Hôpital's rule by taking derivatives of both numerator and denominator}) \\
 &\quad (\text{Wikipedia, Indeterminate form, 2014}) \\
 &= \lim_{\theta \rightarrow +\infty} \left(\frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) - \lim_{\theta \rightarrow -\infty} \left(\frac{1}{(-\hat{a})\exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) \\
 &= \lim_{\theta \rightarrow +\infty} \left(\frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right)
 \end{aligned}$$

$$\begin{aligned}
 B &= \int_{-\infty}^{+\infty} \frac{1}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta = \int_{-\infty}^{+\infty} d\left(\theta + \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) \\
 &= \left(\theta + \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \Big|_{-\infty}^{+\infty} \\
 &= \lim_{\theta \rightarrow +\infty} \theta + \lim_{\theta \rightarrow +\infty} \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) - \lim_{\theta \rightarrow -\infty} \theta \\
 &\quad - \lim_{\theta \rightarrow -\infty} \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \\
 &= \lim_{\theta \rightarrow +\infty} \theta - \lim_{\theta \rightarrow -\infty} \theta - \lim_{\theta \rightarrow -\infty} \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))
 \end{aligned}$$

We have:

$$\begin{aligned}
 \hat{\theta} &= E(\theta | \hat{p}(\theta)) = A - B \\
 &= \lim_{\theta \rightarrow +\infty} \left(\frac{\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) - \lim_{\theta \rightarrow +\infty} \theta + \lim_{\theta \rightarrow -\infty} \theta \\
 &\quad + \lim_{\theta \rightarrow -\infty} \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \\
 &= \lim_{\theta \rightarrow +\infty} \left(-\frac{(\theta)\exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \right) + \lim_{\theta \rightarrow -\infty} \left(\frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) - (-\theta) \right) \\
 &= \lim_{\theta \rightarrow +\infty} \left(-\frac{\theta}{\frac{1}{\exp(\hat{a}\hat{b} - \hat{a}\theta)} + 1} \right) + \lim_{\theta \rightarrow -\infty} \ln \left(\frac{\exp\left(\frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right)}{\exp(-\theta)} \right) \\
 &= \lim_{\theta \rightarrow +\infty} \left(-\frac{d(\theta)}{d\left(\frac{1}{\exp(\hat{a}\hat{b} - \hat{a}\theta)} + 1\right)} \right) \\
 &\quad + \lim_{\theta \rightarrow -\infty} \ln \left(\frac{\exp\left(\frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right)}{\exp(-\theta)} \right)
 \end{aligned}$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator
 (Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}
 &= \lim_{\theta \rightarrow +\infty} \left(\frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\hat{a}} \right) + \ln \left(\lim_{\theta \rightarrow -\infty} \frac{(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))^{\frac{1}{\hat{a}}}}{\exp(-\theta)} \right) \\
 &= \ln \left(\lim_{\theta \rightarrow -\infty} \frac{(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))^{\frac{1}{\hat{a}}}}{\exp(-\theta)} \right) \\
 &= \ln \left(\lim_{\theta \rightarrow -\infty} \left(\frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(-\hat{a}\theta)} \right)^{\frac{1}{\hat{a}}} \right) = \ln \left(\left(\lim_{\theta \rightarrow -\infty} \frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(-\hat{a}\theta)} \right)^{\frac{1}{\hat{a}}} \right) \\
 &= \ln \left(\lim_{\theta \rightarrow -\infty} \left(\frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(-\hat{a}\theta)} \right)^{\frac{1}{\hat{a}}} \right) \\
 &= \ln \left(\left(\lim_{\theta \rightarrow -\infty} \frac{d(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))}{d(\exp(-\hat{a}\theta))} \right)^{\frac{1}{\hat{a}}} \right)
 \end{aligned}$$

(using L'Hôpital's rule by taking derivatives of both numerator and denominator
 (Wikipedia, Indeterminate form, 2014))

$$\begin{aligned}
 &= \ln \left(\left(\lim_{\theta \rightarrow -\infty} \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(-\hat{a}\theta)} \right)^{\frac{1}{\hat{a}}} \right) \\
 &= \ln \left(\left(\lim_{\theta \rightarrow -\infty} \exp(\hat{a}\hat{b}) \right)^{\frac{1}{\hat{a}}} \right) = \ln \left((\exp(\hat{a}\hat{b}))^{\frac{1}{\hat{a}}} \right) = \ln(\exp(\hat{b})) = \hat{b}
 \end{aligned}$$

In general, we have a very important result in which the ability estimate $\hat{\theta}$ equals the difficult parameter estimate \hat{b} . Formula VI.1.3.2.9 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 VI.1.3.2.9. 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 θ_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 (calculating ability estimate of examinee) of the advanced CAT algorithm which will be shown in

the next table VI.1.3.3.1. The formula VI.1.3.2.6 implicates that each examinee is modeled virtually as a test or exam; please pay attention to this interesting thing because it is the core of the advanced CAT algorithm for multi-user test.

Figure VI.1.3.2.1 is an example of IRF function $P(\theta)$ specified by formula VI.1.3.2.1, density function of examinee's ability specified by formula VI.1.3.2.6 together with ability $\hat{\theta} = \hat{b} = 0.5$ given discriminatory parameter $a=5$.

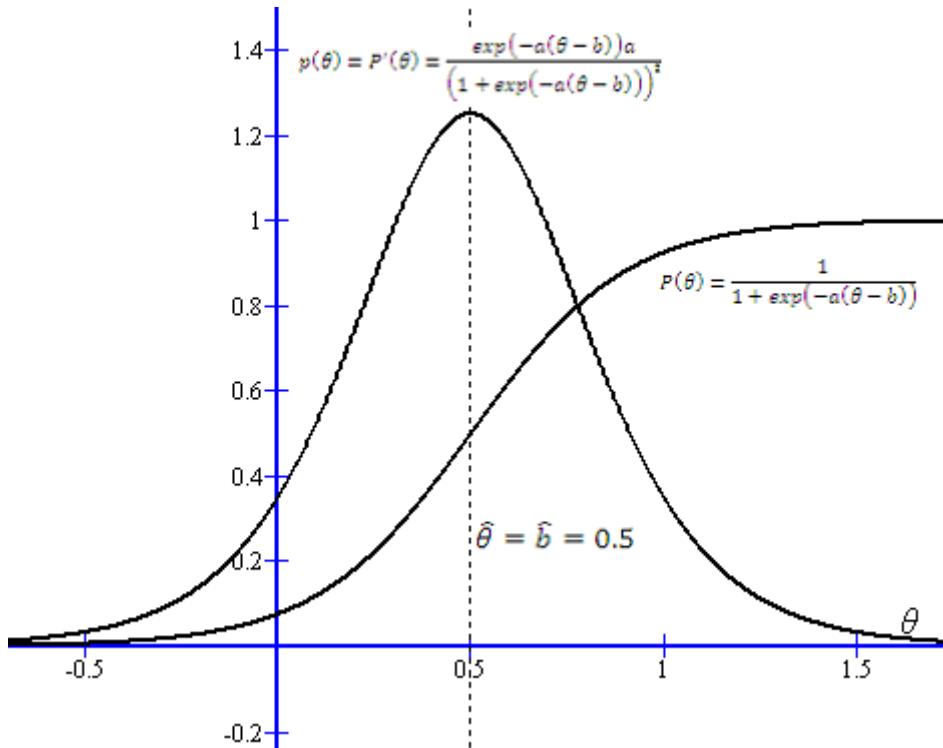


Figure VI.1.3.2.1. IRF function and density function of examinee's ability together with ability estimate (0.5) given discriminatory parameter $a=5$

Based on the proposed MLE method for estimating examinee's ability (formula VI.1.3.2.9), the new version of CAT algorithm is proposed in successive sub-section VI.1.3.3.

VI.1.3.3. New version of CAT algorithm based on MLE

As aforementioned in formula VI.1.3.2.9, 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 VI.1.3.2.8. There is a demand of how to specify ability variance of examinee. The new version of CAT algorithm, 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 VI.1.3.3.1. Examinee's ability variance

Where $\hat{p}(\theta)$ is optimal density function of examinee's ability, specified by formula VI.1.3.2.8.

$$\hat{p}(\theta) = \frac{(\hat{a})\exp(-\hat{a}(\theta - \hat{b}))}{\left(1 + \exp(-\hat{a}(\theta - \hat{b}))\right)^2}$$

It is very easy to infer from formula VI.1.3.3.1 that the square root of $\text{Var}(\hat{\theta})$ is sample standard error according to study of statistics (Montgomery & Runger, 2003, p. 225). The variance $\text{Var}(\hat{\theta})$ is totally determined by calculation of $E(\theta^2|\hat{p}(\theta))$. We have:

$$E(\theta^2|\hat{p}(\theta)) = \int_{-\infty}^{+\infty} \theta^2 \hat{p}(\theta) d\theta = \int_{-\infty}^{+\infty} \theta^2 \frac{(\hat{a})\exp(-\hat{a}(\theta - \hat{b}))}{\left(1 + \exp(-\hat{a}(\theta - \hat{b}))\right)^2} d\theta$$

We have following indefinite integral:

$$\begin{aligned} & \int \theta^2 \frac{(\hat{a})\exp(-\hat{a}(\theta - \hat{b}))}{\left(1 + \exp(-\hat{a}(\theta - \hat{b}))\right)^2} d\theta \\ &= \int \theta^2 d\left(\frac{1}{1 + \exp(-\hat{a}(\theta - \hat{b}))}\right) \\ &= \frac{\theta^2}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \int_{-\infty}^{+\infty} \frac{2\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta \end{aligned}$$

We have:

$$\begin{aligned} & \int \frac{2\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta = \int 2\theta d\left(\theta + \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) \\ &= \left(2\theta^2 + \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) - 2 \int \left(\theta + \frac{1}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) d\theta \\ &= \left(2\theta^2 + \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))\right) - \theta^2 - \frac{2}{\hat{a}} \int \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) d\theta \\ &= \theta^2 + \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) - \frac{2}{\hat{a}} \int \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) d\theta \end{aligned}$$

Given $\hat{a} > 0$ and let $x = \hat{a}\hat{b} - \hat{a}\theta$, we have:

$$d\theta = -\frac{dx}{\hat{a}}$$

And

$$\begin{aligned} & \frac{2}{\hat{a}} \int \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) d\theta = -\frac{2}{\hat{a}^2} \int \ln(1 + \exp(x)) dx \\ &= -\frac{2}{\hat{a}^2} \int \left(\sum_{k=1}^{+\infty} \frac{(-1)^{k+1}}{k} (\exp(x))^k \right) dx = \frac{2}{\hat{a}^2} \int \left(\sum_{k=1}^{+\infty} \frac{(-\exp(x))^k}{k} \right) dx \\ &= \frac{2}{\hat{a}^2} \sum_{k=1}^{+\infty} \int \frac{(-\exp(x))^k}{k} dx = \frac{2}{\hat{a}^2} \sum_{k=1}^{+\infty} \frac{(-\exp(x))^k}{k^2} \\ &= \frac{2}{\hat{a}^2} \text{Li}_2(-\exp(x)) = \frac{2}{\hat{a}^2} \text{Li}_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \end{aligned}$$

Where $Li_2(x)$ is dilogarithm function (Wikipedia, Polylogarithm, 2014). Formula VI.1.3.3.2 expresses dilogarithm function.

$$Li_2(x) = \sum_{k=1}^{+\infty} \frac{x^k}{k^2} = - \int_0^x \frac{\ln(1-t)}{t} dt = \frac{\pi^2}{6} - \int_1^x \frac{\ln(1-t)}{t} dt = - \int_0^1 \frac{\ln(1-xt)}{t} dt$$

Formula VI.1.3.3.2. Dilogarithm function

Where $Li_2(0) = 0$.

You can also find out formula VI.1.3.3.2 in (Wolfram|Alpha, n.d.) and (Weisstein, Dilogarithm). We have

$$\begin{aligned} & \int \frac{2\theta}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} d\theta \\ &= \theta^2 + \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) - \frac{2}{\hat{a}^2} Li_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \end{aligned}$$

It is easy to infer that

$$\begin{aligned} & \int \theta^2 \frac{(\hat{a}) \exp(-\hat{a}(\theta - \hat{b}))}{(1 + \exp(-\hat{a}(\theta - \hat{b})))^2} d\theta \\ &= \frac{\theta^2}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \theta^2 - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \\ & \quad + \frac{2}{\hat{a}^2} Li_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \\ &= -\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) + \frac{2}{\hat{a}^2} Li_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \end{aligned}$$

It implies that

$$\begin{aligned} E(\theta^2 | \hat{p}(\theta)) &= \int_{-\infty}^{+\infty} \theta^2 \hat{p}(\theta) d\theta \\ &= \lim_{\theta \rightarrow +\infty} \left(-\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\ & \quad \left. + \frac{2}{\hat{a}^2} Li_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\ & \quad - \lim_{\theta \rightarrow -\infty} \left(-\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\ & \quad \left. + \frac{2}{\hat{a}^2} Li_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \end{aligned}$$

Formula VI.1.3.3.3. Expectation of square of examinee's ability

Formula VI.1.3.3.3 expresses how to calculate the expectation $E(\theta^2 | \hat{p}(\theta))$ used to determine ability variance $Var(\hat{\theta})$. It is easy to get a formula similar to the formula

[VI.1.3.3.3](#) for calculating $E(\theta^2 | \hat{p}(\theta))$ by using the mathematics engine (Wolfram|Alpha, n.d.). According to formula [VI.1.3.3.3](#), 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 $\frac{2}{\hat{a}^2} Li_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta))$ as θ 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))

$$\begin{aligned} &= \frac{2}{\hat{a}} \lim_{\theta \rightarrow +\infty} \frac{\frac{(\hat{a}) \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}}{\frac{1}{\theta^2}} = 2 \lim_{\theta \rightarrow +\infty} \frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \\ &= 2 \frac{\lim_{\theta \rightarrow +\infty} (\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta))}{\lim_{\theta \rightarrow +\infty} (1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))} = 2 \lim_{\theta \rightarrow +\infty} (\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)) \\ &= 2 \lim_{\theta \rightarrow +\infty} \frac{\theta^2}{\exp(-\hat{a}\hat{b} + \hat{a}\theta)} = 2 \lim_{\theta \rightarrow +\infty} \frac{d(\theta^2)}{d(\exp(-\hat{a}\hat{b} + \hat{a}\theta))} \end{aligned}$$

(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:

$$\begin{aligned} \lim_{\theta \rightarrow -\infty} &\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} \\ &= \lim_{\theta \rightarrow -\infty} \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\frac{1}{\theta^2} + \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\theta^2}} = \frac{\lim_{\theta \rightarrow -\infty} \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\lim_{\theta \rightarrow -\infty} \left(\frac{1}{\theta^2} + \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\theta^2} \right)} \\ &= \frac{\lim_{\theta \rightarrow -\infty} \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\lim_{\theta \rightarrow -\infty} \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\theta^2}} = \lim_{\theta \rightarrow -\infty} \frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\frac{\exp(\hat{a}\hat{b} - \hat{a}\theta)}{\theta^2}} = \lim_{\theta \rightarrow -\infty} \theta^2 \end{aligned}$$

We have:

$$\begin{aligned} \lim_{\theta \rightarrow -\infty} &\left(\frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\ &= \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) - \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} (\hat{a}\hat{b} - \hat{a}\theta) + \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} (\hat{a}\hat{b} - \hat{a}\theta) \\ &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} \left(\ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) - (\hat{a}\hat{b} - \hat{a}\theta) \right) + \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} (\hat{a}\hat{b} - \hat{a}\theta) \\ &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} \ln\left(\frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(\hat{a}\hat{b} - \hat{a}\theta)}\right) + \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \\ &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left(\lim_{\theta \rightarrow -\infty} \ln\left(\frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(\hat{a}\hat{b} - \hat{a}\theta)}\right) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\ &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left(\ln\left(\lim_{\theta \rightarrow -\infty} \left(\frac{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)}{\exp(\hat{a}\hat{b} - \hat{a}\theta)}\right)\right) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\ &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left(\ln\left(\lim_{\theta \rightarrow -\infty} \left(\frac{d(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta))}{d(\exp(\hat{a}\hat{b} - \hat{a}\theta))}\right)\right) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \end{aligned}$$

(by using L'Hôpital's rule by taking derivatives of both numerator and denominator
(Wikipedia, Indeterminate form, 2014))

$$\begin{aligned} &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left(\ln\left(\lim_{\theta \rightarrow -\infty} \left(\frac{-(\hat{a})\exp(\hat{a}\hat{b} - \hat{a}\theta)}{-(\hat{a})\exp(\hat{a}\hat{b} - \hat{a}\theta)}\right)\right) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\ &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \left(\ln(1) + \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) \right) \end{aligned}$$

$$\begin{aligned}
 &= \lim_{\theta \rightarrow -\infty} \frac{2\theta}{\hat{a}} \lim_{\theta \rightarrow -\infty} (\hat{a}\hat{b} - \hat{a}\theta) = \lim_{\theta \rightarrow -\infty} \left(\frac{2\theta}{\hat{a}} (\hat{a}\hat{b} - \hat{a}\theta) \right) \\
 &= -2 \lim_{\theta \rightarrow -\infty} \theta^2 + 2\hat{b} \lim_{\theta \rightarrow -\infty} \theta
 \end{aligned}$$

According to (Wikipedia, Polygamma function, 2014), given $x < 0$ formula VI.1.3.3.4 is an *inversion property* of dilogarithm, as follows:

$$Li_2(x) = -Li_2\left(\frac{1}{x}\right) - \frac{(ln(-x))^2}{2} - \frac{\pi^2}{6}$$

Formula VI.1.3.3.4. 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:

$$\begin{aligned}
 &\lim_{\theta \rightarrow -\infty} \frac{2}{\hat{a}^2} Li_2(-exp(\hat{a}\hat{b} - \hat{a}\theta)) = \frac{2}{\hat{a}^2} \lim_{\theta \rightarrow -\infty} Li_2(-exp(\hat{a}\hat{b} - \hat{a}\theta)) \\
 &= -\frac{2}{\hat{a}^2} \left(\lim_{\theta \rightarrow -\infty} Li_2(-exp(\hat{a}\theta - \hat{a}\hat{b})) + \lim_{\theta \rightarrow -\infty} \frac{(\hat{a}\hat{b} - \hat{a}\theta)^2}{2} + \frac{\pi^2}{6} \right) \\
 &= -\frac{2}{\hat{a}^2} \left(Li_2\left(\lim_{\theta \rightarrow -\infty} (-exp(\hat{a}\theta - \hat{a}\hat{b}))\right) + \lim_{\theta \rightarrow -\infty} \frac{\hat{a}^2(\hat{b} - \theta)^2}{2} + \frac{\pi^2}{6} \right) \\
 &= -\frac{2}{\hat{a}^2} \left(Li_2(0) + \lim_{\theta \rightarrow -\infty} \frac{\hat{a}^2(\hat{b} - \theta)^2}{2} + \frac{\pi^2}{6} \right) = -\frac{2}{\hat{a}^2} \left(\lim_{\theta \rightarrow -\infty} \frac{\hat{a}^2(\hat{b} - \theta)^2}{2} + \frac{\pi^2}{6} \right) \\
 &\quad \text{(due to } Li_2(0) = 0\text{)} \\
 &= -\hat{b}^2 + \lim_{\theta \rightarrow -\infty} 2\hat{b}\theta - \lim_{\theta \rightarrow -\infty} \theta^2 - \frac{\pi^2}{3\hat{a}^2}
 \end{aligned}$$

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 VI.1.3.3.3 for calculating the expectation $E(\theta^2 | \hat{p}(\theta))$, we have

$$E(\theta^2 | \hat{p}(\theta)) = \int_{-\infty}^{+\infty} \theta^2 \hat{p}(\theta) d\theta$$

$$\begin{aligned}
 &= \lim_{\theta \rightarrow +\infty} \left(-\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\
 &\quad \left. + \frac{2}{\hat{a}^2} \text{Li}_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\
 &- \lim_{\theta \rightarrow -\infty} \left(-\frac{\theta^2 \exp(\hat{a}\hat{b} - \hat{a}\theta)}{1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)} - \frac{2\theta}{\hat{a}} \ln(1 + \exp(\hat{a}\hat{b} - \hat{a}\theta)) \right. \\
 &\quad \left. + \frac{2}{\hat{a}^2} \text{Li}_2(-\exp(\hat{a}\hat{b} - \hat{a}\theta)) \right) \\
 &= 0 - \left(-\hat{b}^2 - \frac{\pi^2}{3\hat{a}^2} \right) = \frac{\pi^2}{3\hat{a}^2} + \hat{b}^2
 \end{aligned}$$

According to formula VI.1.3.3.1 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 VI.1.3.3.5 indicates how to compute the variance of examinee's ability estimate. Please pay attention to the important result of formula VI.1.3.3.5 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 VI.1.3.3.5. 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 VI.1.3.3.5, we have:

$$\sqrt{\text{Var}(\hat{\theta})} = \sqrt{\frac{\pi^2}{3\hat{a}^2}} = 0.36$$

Figure VI.1.3.3.1 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 VI.1.3.2.8.

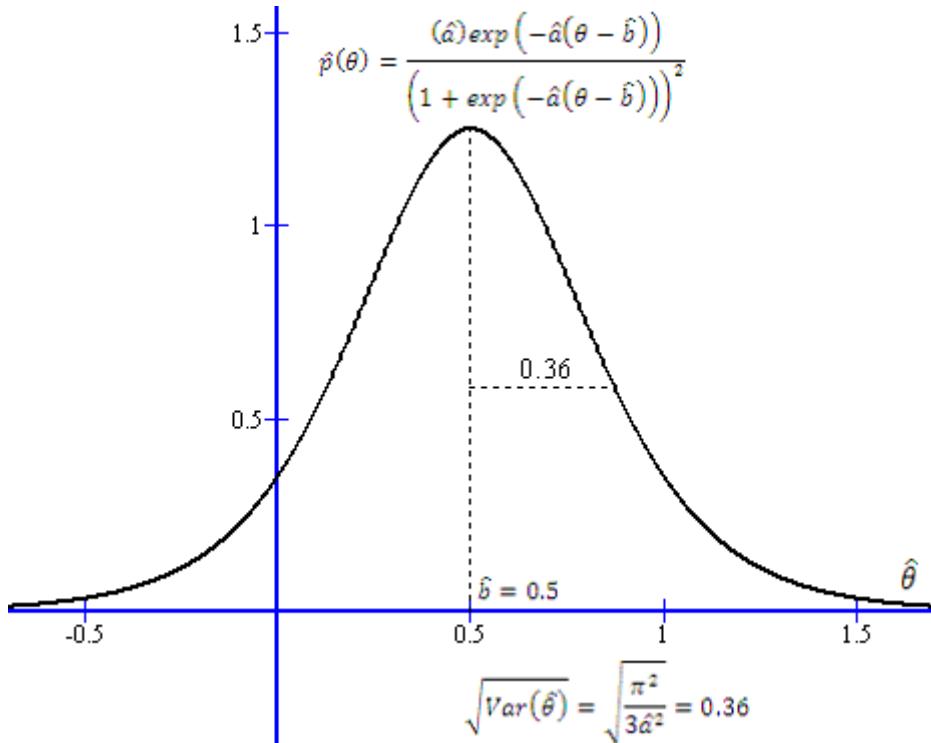


Figure VI.1.3.3.1. An example of examinee's ability variance

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 $\bar{\theta}$ be statistical sample mean (Montgomery & Runger, 2003, p. 190) of examinees' ability estimates, formula VI.1.3.3.6 specifies the mean $\bar{\theta}$.

$$\bar{\theta} = \frac{1}{k} \sum_{i=1}^k \hat{\theta}_i$$

Formula VI.1.3.3.6. 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 VI.1.3.3.7 as follows:

$$Var(\hat{\theta}) = \frac{1}{k-1} \sum_{i=1}^k (\hat{\theta}_i - \bar{\theta})^2$$

Formula VI.1.3.3.7. Examinee's ability variance as statistical sample variance

Where $\bar{\theta}$ is the ability mean specified by formula VI.1.3.3.6.

Formula VI.1.3.3.7 shares the same purpose with the formula VI.1.3.3.5 but their approaches are different. The variance $Var(\hat{\theta})$ specified by formula VI.1.3.3.5 is essentially the *local variance* focusing on individual examinee. The variance $Var(\hat{\theta})$ specified by formula VI.1.3.3.7 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 sub-section VI.1.3.2) is computed based on these ability variances.

As aforementioned in previous sub-section VI.1.3.2, at the step 1 of CAT algorithm shown table VI.1.3.1.1, the best item is the one that maximizes the information function $I_i(\theta)$ specified by formula VI.1.3.1.2. 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 VI.1.3.3.5 we have:

$$Var(\hat{\theta}) = \frac{\pi^2}{3(a^*)^2} \Rightarrow a^* = \frac{\pi}{\sqrt{3Var(\hat{\theta})}}$$

Applying formula VI.1.3.3.7 into determining $Var(\hat{\theta})$, we have:

$$a^* = \frac{\pi}{\sqrt{3Var(\hat{\theta})}} = \frac{\pi}{\sqrt{\frac{3}{k-1} \sum_{i=1}^k (\hat{\theta}_i - \bar{\theta})^2}} = \sqrt{\frac{(k-1)\pi^2}{3 \sum_{i=1}^k (\hat{\theta}_i - \bar{\theta})^2}}$$

Briefly, formula VI.1.3.3.8 specifies discriminatory estimate.

$$a^* = \sqrt{\frac{(k-1)\pi^2}{3 \left(\sum_{i=1}^k (\hat{\theta}_i - \bar{\theta})^2 \right)}}$$

Formula VI.1.3.3.8. 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 $\bar{\theta}$ is the ability mean specified by formula VI.1.3.3.6 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 VI.1.3.3.1. Such new CAT algorithm is called *advanced CAT algorithm*.

- 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 in 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 choose 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 $\frac{\bar{\theta}a^*}{a_i}$ 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 VI.1.3.2.9. 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 δ . The δ 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 δ , 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 VI.1.3.3.8. 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 items are the most suitable to examinees' current ability estimates. Not like traditional CAT mentioned in table VI.1.3.1.1, it is not necessary to calculate information function specified by formula VI.1.3.1.2 for selecting the best item. For this reason, the computation cost is decreased significantly.

2. Such best items are given to examinees and examinees make responses to these items. 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 subsection VI.1.3.2 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 VI.1.3.2.9. Moreover, discriminatory estimate a^* is re-computed based on new estimates $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$ according to formula VI.1.3.3.8. 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 VI.1.3.3.1. 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 section [V.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 10 items then the examinee's final estimate is specified at 10th 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 θ . Given threshold ζ , if the ability error is less than ζ then the CAT algorithm terminates; hence this is the new stopping criterion for CAT algorithm. Formula [VI.1.3.3.9](#) specifies ability error denoted Err .

$$Err = |\hat{\theta} - \theta|$$

Formula VI.1.3.3.9. 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 ζ .

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, as seen in table [VI.1.3.3.2](#). 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
	<i>Ability</i> (θ_i)
Examinee 1	$\theta_1=0$
Examinee 2	$\theta_2=0$
Examinee 3	$\theta_3=0$
Examinee 4	$\theta_4=0$

Table VI.1.3.3.2. 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 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 $\left| b_i - \frac{\bar{\theta}a^*}{a_i} \right|$ are the smallest ones according to step 1 of advanced CAT algorithm. So, the informative set C is:

$$C = \{\text{item 1, item 4}\}$$

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 VI.1.3.2.9 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 VI.1.3.3.8, we have:

$$a^* = \sqrt{\frac{(4-1)\pi^2}{3\left(\left(\hat{\theta}_1 - \bar{\theta}\right)^2 + \left(\hat{\theta}_2 - \bar{\theta}\right)^2 + \left(\hat{\theta}_3 - \bar{\theta}\right)^2 + \left(\hat{\theta}_4 - \bar{\theta}\right)^2\right)}} = \sqrt{\frac{\pi^2}{(0.41 - 0.225)^2 + (0.25 - 0.225)^2 + (0.12 - 0.225)^2 + (0.12 - 0.225)^2}} \approx 13.24$$

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 θ_4 are re-assigned as follows:

$$\begin{aligned}\theta_1 &= \hat{\theta}_1 \approx 0.1047 \\ \theta_2 &= \hat{\theta}_2 \approx 0.0640 \\ \theta_3 &= \hat{\theta}_3 \approx 0.0306 \\ \theta_4 &= \hat{\theta}_4 \approx 0.0306\end{aligned}$$

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:

$$\begin{cases} \left| b_1 - \frac{\bar{\theta}a^*}{a_1} \right| = \left| 2 - \frac{0.0575 * 13.24}{2} \right| \approx 1.62 \\ \left| b_2 - \frac{\bar{\theta}a^*}{a_2} \right| = \left| 3 - \frac{0.0575 * 13.24}{4} \right| \approx 2.81 \\ \left| b_3 - \frac{\bar{\theta}a^*}{a_3} \right| = \left| 5 - \frac{0.0575 * 13.24}{3} \right| \approx 4.75 \\ \left| b_5 - \frac{\bar{\theta}a^*}{a_5} \right| = \left| 4 - \frac{0.0575 * 13.24}{5} \right| = 3.85 \end{cases}$$

Suppose we choose two items (item 1 and item 2) whose deviations $\left| b_i - \frac{\bar{\theta}a^*}{a_i} \right|$ 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:

$$|a_1 - a^*| = |2 - 13.24| = 11.24$$

$$|a_2 - a^*| = |4 - 13.24| = 9.240$$

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 VI.1.3.2.9 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 VI.1.3.3.8, 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 θ_1 , θ_2 , θ_3 , and θ_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 VI.1.3.3.3 shows results of our multi-user test.

		θ_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 VI.1.3.3.3. 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 VI.1.3.3.3, the final discriminatory estimate $a^*=616$ is very high while the initial value of a^* is 3.4. It implies that the variance of θ_1 , θ_2 , θ_3 , and θ_4 gets small; please see formula VI.1.3.3.8 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 VI.1.3.2.9 for more details about ability estimate. Then, the discriminatory parameter is estimated based on the variance of examinee's ability. Please see formulas VI.1.3.3.5 and VI.1.3.3.8 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 VI.1.3.2.9, VI.1.3.3.5 and VI.1.3.3.8. 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 ζ , if the ability error is less than ζ 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. The idea of ability error is not actually new but I hope that it may be useful for researchers.

Formulas VI.1.3.2.1 and VI.1.3.2.7 represent a so-called logistic distribution. Note that the logistic distribution had been discovered before I researched it (I used to think that I discovered it from item response theory) but I calculated the mean and variance of logistic distribution independently without referring its documents such as (Wikipedia, Logistic distribution, 2016) and (Weisstein, Logistic Distribution, n.d.).

I had a problem of calculating the variance among examinees' abilities but finally, I found out the formula to calculate this variance with support of (Wolfram|Alpha, n.d.) engine. Formula VI.1.3.3.5 is the most important finding in the research, in which the dilogarithm function is key solution. So, ability variance is the main aspect of the paper that makes it worth worldwide. However, both ability estimate and ability variance are based on the condition that guessing parameter c_i of IRF is zero. Consequently, IRF is considered as cumulative distribution function according to formula VI.1.3.2.1. When guessing parameter c_i is non-zero, IRF does not satisfy an axiom of probability distribution because the IRF approaches $c_i \neq 0$ if the true ability θ approaches negative infinity. If we translate the IRF to horizontal axis $IRF(\theta) = 0$ then, the IRF will approach $1-c_i$ below 1 when the true ability θ approaches positive infinity. This is also invalid, which raises a hazardous problem. Therefore, for further research, I will try my best to re-calculate ability estimate and ability variance in the most general case $c_i \neq 0$.

As aforementioned at the beginning of this section VI.1, knowledge sub-model in **Triangular Learner Model** (TLM) is evaluated by two viewpoints:

- Methodological viewpoint described in sub-section VI.1.1 mentions theoretical aspects of Bayesian knowledge model.
- Practical viewpoint described in sub-sections VI.1.2 and VI.1.3 mention how to exploit Bayesian model in practical assessment.

The sub-section VI.1.3 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 she/he is modeled as a so-called TLM model. Finally, the subject of evaluating knowledge model is closed by this sub-section VI.1.3.3 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 VI.2 in which the whole of adaptive learning system and user modeling system is evaluated.

VI.2. Evaluation of adaptive learning model

Recall that the chapter VI focuses on evaluating [Triangular Learner Model](#) (TLM) and user modeling system [Zebra](#) with regard to their effectiveness. Previous section VI.1 is the evaluation on knowledge sub-model. This section VI.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 provides necessary information about user to adaptive system. Following figure VI.2.1 describes the interaction between user modeling system and adaptive system.

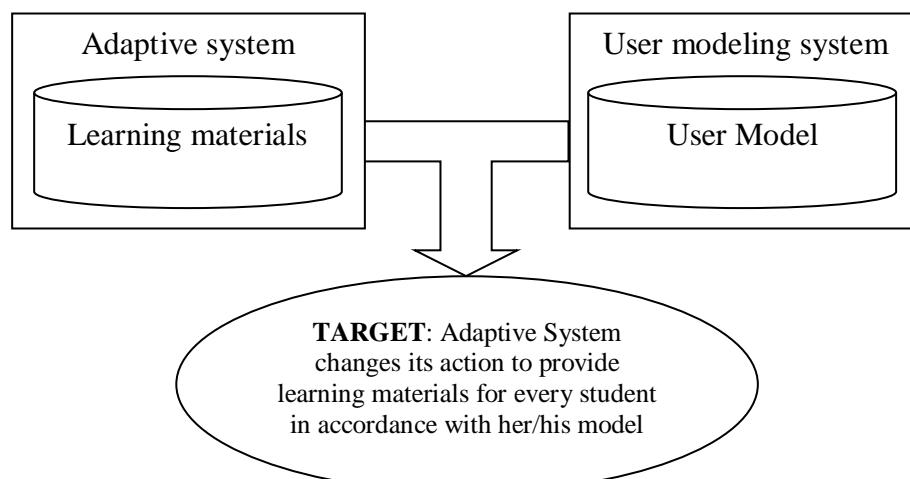


Figure VI.2.1. Interaction between user modeling system and adaptive system

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 different from studying at a course with support of network. This section VI.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 L. , Evaluating Adaptive Learning Model, 2014). We can consider the corporation between adaptive system and user modeling system as an integrated model called adaptive learning model. Thus, this section has two goals:

- Firstly, research proposes criteria to evaluate adaptive learning model. Successive sub-section VI.2.1 describes evaluation criteria in detailed.
- Secondly, research gives a scenario as an example that applies criteria above into performing evaluation task in concrete situations. Sub-section VI.2.2 introduces an evaluation scenario.

In other words, this section VI.2 gives criteria 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.

VI.2.1. Evaluation criteria

This research proposes three criteria of evaluation:

- Criterion α called *system criterion* tells us how adaptive learning system works with/without user modeling system. For example, when modeling server applies Bayesian network into building up learner model, criterion α 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 β called *academic criterion* tells us how well modeling server helps users to study. This criterion surveys users' study result. The higher criterion β is, the better study result is.
- Criterion γ 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 γ is calculated based on these feedbacks in order to estimate level of students' satisfaction from adaptive system. The higher criterion γ is, the better quality of adaptation is.

Now we discuss methods to determine these criteria. Note that in this research, the default user model is Bayesian network model if there is no additional explanation; please see section III.1 for more details about Bayesian model. Successive sub-sections VI.2.1.1, VI.2.1.2, and VI.2.1.3 describe how to calculate criterion α , β , and γ , respectively.

VI.2.1.1. Calculating system criterion α

There are two ways to calculate system criterion α 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:

$K_i = \{k_1^i, k_2^i, \dots, k_n^i\}$ has sample variance s_i^2 .

$K_j = \{k_1^j, k_2^j, \dots, k_n^j\}$ has sample variance s_j^2 .

As an extension, K_i and K_j can be replaced by G_A and G_B that are study results of groups A and B , respectively. So the construction of vectors K_i and K_j is dependent on application context.

Let \bar{K}_i and \bar{K}_j be sample means of K_i and K_j , respectively, we have formula [VI.2.1.1.1](#) 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 VI.2.1.1.1. Sample means

Sample variances s_i^2 and s_j^2 are specified by following formula [VI.2.1.1.2](#).

$$s_i^2 = \frac{1}{n-1} \sum_{c=1}^n (k_c^i - \bar{K}_i)^2 \text{ and } s_j^2 = \frac{1}{n-1} \sum_{c=1}^n (k_c^j - \bar{K}_j)^2$$

Formula VI.2.1.1.2. Sample variances

Where \bar{K}_i and \bar{K}_j are sample means.

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 [VI.2.1.1.3](#) for determining sample standard deviations.

$$s_i = \sqrt{s_i^2} = \sqrt{\frac{1}{n-1} \sum_{c=1}^n (k_c^i - \bar{K}_i)^2}$$

$$s_j = \sqrt{s_j^2} = \sqrt{\frac{1}{n-1} \sum_{c=1}^n (k_c^j - \bar{K}_j)^2}$$

Formula VI.2.1.1.3. Sample standard deviations

Where \bar{K}_i and \bar{K}_j are sample means.

Please read (Montgomery & Runger, 2003, pp. 190-191) for more details about sample mean, and sample variance.

Criterion α 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:

$$H_0: s_i^2 = s_j^2$$

$$H_1: s_i^2 < s_j^2$$

Suppose the significant level is 0.05, F-distribution (Montgomery & Runger, 2003, p. 356) is used to test two variances. Formula VI.2.1.1.4 expresses F-distribution test for determining criterion α .

$$F = \frac{s_i^2}{s_j^2}$$

Formula VI.2.1.1.4. System criterion α determined based on F-distribution test

If $F < f_{0.95,n-1,n-1}$ then the null hypothesis is rejected, we can conclude that group A with support of user modeling system improves adaptive learning system much more than group B. Note that $f_{0.95,n-1,n-1}$ is the *lower-tail 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). Note,

$$f_{0.95,n-1,n-1} = \frac{1}{f_{0.05,n-1,n-1}}$$

Where $f_{0.05,n-1,n-1}$ is the *upper-tail 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 α gets Boolean value *true* or 1, indicating the preeminence of user modeling system.

The essence of α 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 represent 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 VI.2.1.1.5 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 VI.2.1.1.5. 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 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 α . In general, the process to calculate α 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). 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.

In group A: $ek_i^A = Y^A = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + \dots + a_nX_n$

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 errors of user i^{th} in group A and B, respectively. These errors 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 α is simple inverse of square sum of all prediction errors. Formula VI.2.1.1.6 specifies criterion α 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 VI.2.1.1.6. System criterion α determined based errors of linear regression model

The larger the measure α is, the better the level of precision of inference method is.

Next sub-section VI.2.1.2 mentions the second measure called academic criterion β .

VI.2.1.2. Calculating academic criterion β

Let K_A , K_B be the knowledge vectors of group A and B, respectively where k_i^A and k_i^B are the 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 VI.2.1.1.1 and VI.2.1.1.2, respectively. The measure β for each group is computed as accumulative probability of assumption user in such group has mastered over course. Formula VI.2.1.2.1 indicates how to calculate criterion β .

$$\beta_A = \Phi \left(\frac{\bar{A} - \mu}{\sqrt{s_A^2/n}} \right)$$

$$\beta_B = \Phi \left(\frac{\bar{B} - \mu}{\sqrt{s_B^2/n}} \right)$$

Where μ is the average point in the grade system, for example, μ is 5 in ten-point grade system and is 2.5 in five-point grade system. If the grade system is normalized in range [0, 1] then, $\mu=0.5$. Function Φ is cumulative function for standard normal distribution (Montgomery & Runger, 2003, p. 111).

Formula VI.2.1.2.1. Academic criterion β based cumulative function

It is more accurate in calculating criterion β if function Φ is cumulative function for t -distribution with $n-1$ degree of freedom (Montgomery & Runger, 2003, p. 301). Here we still use standard normal distribution as example for convenience. It is easy to look up values of standard normal cumulative function Φ in the appendix A of the book

“Applied Statistics and Probability for Engineers” by the authors Montgomery and Runger (Montgomery & Runger, 2003, p. 653).

The higher criterion β is, the better study result is because the academy criterion α is measured as the probability tending to event that student’s grade is higher than or equal to the average point.

Successive sub-section [VI.2.1.3](#) mentions the last measure called adaptation criterion γ .

VI.2.1.3. Calculating adaptation criterion γ

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 γ is defined as the ratio of the number of satisfied users to the whole number of users. Formula [VI.2.1.3.1](#) specifies simple criterion γ according to the number of satisfied users.

$$\gamma = \frac{\text{The number of satisfied users}}{\text{The whole number of users}}$$

Formula VI.2.1.3.1. Simple adaptation criterion γ based the number of satisfied users

In enhance method, the rating values range in an interval, for example [0...5], 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 μ_A and μ_B be the mean vectors of group A and B , respectively. The measure γ is computed as the module of mean vector. Which group has higher measure γ 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 γ :

- 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. So it is easy to infer that the purpose of “shrinking” rating matrix is to remove out unessential questions to which very few users respond. Table [VI.2.1.3.1](#) 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 VI.2.1.3.1. Rating matrix as collection of users’ feedbacks

Table [VI.2.1.3.2](#) shows shrunk matrix created by projecting origin rating matrix shown in table [VI.2.1.3.1](#) 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 VI.2.1.3.2. 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 μ_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 μ_i and the criterion γ is the module of such mean vector. Formula VI.2.1.3.2 expresses criterion γ as module of mean 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 VI.2.1.3.2. Adaptation criterion γ as module of mean vector

In general, the higher criterion γ is, the better quality of adaptation is. Now all evaluation criteria such as α , β , and γ were determined, it is necessary to organize an evaluation scenario so as to apply these measures. The next sub-section VI.2.2 mentions a proposed evaluation scenario for adaptive learning model.

VI.2.2. An evaluation scenario

Evaluation scenario is the example for demonstrating how to calculate and apply aforementioned criteria 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, criteria α , β and γ 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 criteria.

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. It is recommended that **WOW** is an adaptive e-learning system with the support of the user modeling system **Zebra**, which is introduced in sub-section [II.2.4](#).
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 website. Note that website / learning materials are adapted to each student based on her/his 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 criteria α , β , γ based on students' feedbacks 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 in adaptive learning environment like **WOW**. Study results and students' feedbacks are collected from both two classes. Each class has the same number of students, namely 10. Let G_A and G_B be the study results of classes *A* and *B*, respectively.

$$G_A = \{4, 5, 3, 6, 2, 8, 3, 5, 8, 6\}$$

$$G_B = \{6, 8, 9, 10, 6, 8, 8, 6, 9, 10\}$$

Suppose there are 2 students in class *A* and 6 students in class *B* who satisfy teaching curriculum. Let \bar{A} and s_A^2 be sample mean and sample variance of class *A*. Let \bar{B} and

s_B^2 be sample mean and sample variance of class B . By applying formulas VI.2.1.1.1 and VI.2.1.1.2, it is easy to calculate these means and variances.

$$\bar{A} = \frac{1}{10}(4 + 5 + 3 + 6 + 2 + 8 + 3 + 5 + 8 + 6) = 5$$

$$\bar{B} = \frac{1}{10}(6 + 8 + 9 + 10 + 6 + 8 + 8 + 6 + 9 + 10) = 8$$

$$s_A^2 = \frac{1}{9}((4 - 5)^2 + (5 - 5)^2 + (3 - 5)^2 + (6 - 5)^2 + (2 - 5)^2 + (8 - 5)^2 + (3 - 5)^2 + (5 - 5)^2 + (8 - 5)^2 + (6 - 5)^2) \approx 4.2$$

$$s_B^2 = \frac{1}{9}((6 - 8)^2 + (8 - 8)^2 + (9 - 8)^2 + (10 - 8)^2 + (6 - 8)^2 + (8 - 8)^2 + (8 - 8)^2 + (6 - 8)^2 + (9 - 8)^2 + (10 - 8)^2) \approx 2.7$$

The system criterion is calculated based on F -test. According to formula VI.2.1.1.4, the F value is:

$$F = \frac{s_B^2}{s_A^2} = \frac{2.7}{4.2} \approx 0.58$$

Let $f_{0.95,9,9}$ be the lower-tail percentage point of F-distribution given numerator degrees of freedom 9 and denominator degrees of freedom 9 at significant level 0.05. Looking up appendix A of the book “Applied Statistics and Probability for Engineers” by the authors Montgomery & Runger (Montgomery & Runger, 2003, p. 657), we have:

$$f_{0.95,9,9} = \frac{1}{f_{0.05,9,9}} = \frac{1}{1.59} \approx 0.63$$

Due to $F = 0.58 < f_{0.95,9,9} = 0.63$, it is included that teaching method in class B with support of adaptive environment achieves more advances with regard to system criterion. Please see sub-section VI.2.1.1 for more details about system criterion and hypothesis test. Thus, let α_A and α_B be system criteria of classes A and B , respectively, we have:

$$\begin{aligned}\alpha_A &= 0 \\ \alpha_B &= 1\end{aligned}$$

The grade system is ten-point grade system and so, the average point μ is 5. Let β_A and β_B be academy criteria of class A and class B , respectively; according to formula VI.2.1.2.1, we have:

$$\beta_A = \Phi\left(\frac{\bar{A} - \mu}{\sqrt{s_A^2/10}}\right) = \Phi\left(\frac{\bar{A} - 5.0}{\sqrt{4.2/10}}\right) = \Phi\left(\frac{5.0 - 0.5}{\sqrt{4.2/10}}\right) = \Phi(0) = 0.5$$

$$\beta_B = \Phi\left(\frac{\bar{B} - \mu}{\sqrt{s_B^2/10}}\right) = \Phi\left(\frac{\bar{B} - 5.0}{\sqrt{2.7/10}}\right) = \Phi\left(\frac{8 - 5.0}{\sqrt{2.7/10}}\right) = \Phi(3) = 0.998650 \approx 1$$

Recall that Φ is cumulative function for standard normal distribution (Montgomery & Runger, 2003, p. 111). It is easy to look up values of function Φ in the appendix A of the book “Applied Statistics and Probability for Engineers” by the authors Montgomery and Runger (Montgomery & Runger, 2003, p. 653).

Let γ_A and γ_B be satisfaction criteria of class A and class B , respectively, we calculate γ_A and γ_B according to formula VI.2.1.3.1 given there are 2 students in class A and 6 students in class B who satisfy teaching curriculum, as follows:

$$\gamma_A = \frac{2}{10} = 0.2$$

$$\gamma_B = \frac{6}{10} = 0.6$$

As a result, system criterion, academy criterion and satisfaction criterion of classes A and B are:

	Class A	Class B
System criterion	$\alpha_A = 0.0$	$\alpha_B = 1.0$
Academy criterion	$\beta_A = 0.5$	$\beta_B = 1.0$
Satisfaction criterion	$\gamma_A = 0.2$	$\gamma_B = 0.6$

Suppose the weights of system criterion, academy criterion and satisfaction criterion are 0.3, 0.5, and 0.2, respectively. Let $eval_A$ and $eval_B$ be the total evaluations on class A and class B , respectively.

$$eval_A = 0.3*\alpha_A + 0.5*\beta_A + 0.2*\gamma_A = 0.3*0.0 + 0.5*0.5 + 0.2*0.2 = 0.29$$

$$eval_B = 0.3*\alpha_B + 0.5*\beta_B + 0.2*\gamma_B = 0.3*1.0 + 0.5*1.0 + 0.2*0.6 = 0.92$$

When $eval_B$ is greater than $eval_A$, 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 environment.

This section [VI.2](#) is finished with some comments on evaluation criteria. As aforementioned, there are three criteria such as system criterion α , academy criterion β and adaptation criterion γ . That two of three criteria, concretely α and β , 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 criteria, 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 [VI.2](#) ends up the main contents of this research and gives you the comprehensive and detailed description about the research and so the next chapter [VII](#) is the conclusion and future trend.

Chapter VII. Future Trend of Triangular Learner Model

Recall that chapter I is the state-of-art of user model and user modeling system. Chapters II, III, IV, and V 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 VI is the evaluation of TLM and Zebra. Now main contents of this research were introduced wholly to you. This chapter VII only aims to conclude the whole research with the viewpoint of general architecture when main details and formulas are described in the most important chapters III, IV, and V. Therefore, the section VII.1 is the general conclusion of the research. Additionally, section VII.2 mentions the main future trend of the research; thus the proposed user modeling system [Zebra](#) will support ubiquitous system.

VII.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 the standard of living is, the more important the adaptive information technology (IT) systems are. These adaptive systems have ability to change their behaviors so as to be in accordance users' characteristics. Note that this research 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 such 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 VII.1.1 is replication of figure II.2.1.1 in previous sub-section II.2.1, which depicts TLM.

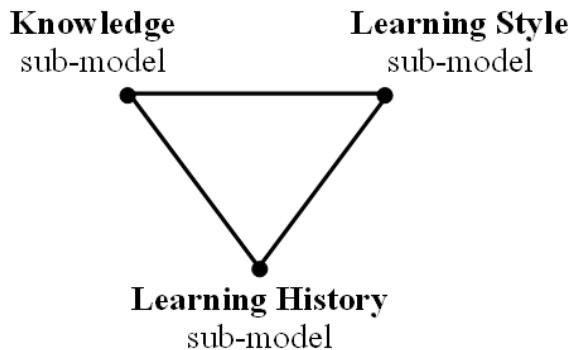


Figure VII.1.1. Triangular Learner Model

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 VII.1.2 is replication of figure II.2.1.2 in previous sub-section II.2.1, which depicts extended TLM.

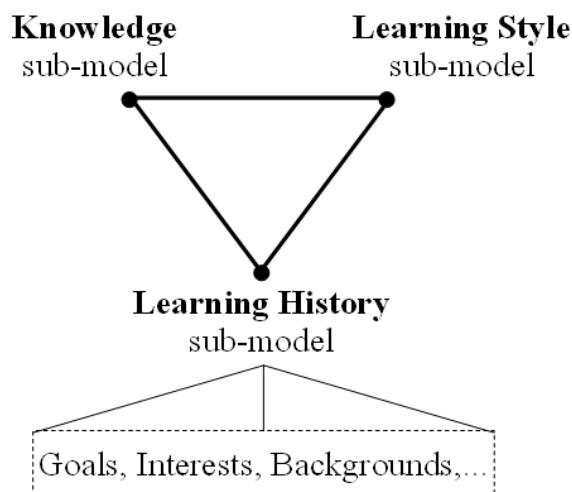


Figure VII.1.2. extended Triangular Learner Model

I also introduce the architecture of the user modeling system 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 chapters III, IV, and V.

- Belief network engine (BNE) is responsible for inferring new user information 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 chapters III and IV.

Zebra provides *communication interfaces* (CI) that allow users and adaptive systems to see or modify restrictedly TLM. Adaptive applications also interact with Zebra by these interfaces. Following figure VII.1.3 is replication of figure II.2.2.3 in previous sub-section II.2.2, which depicts the general architecture of Zebra.

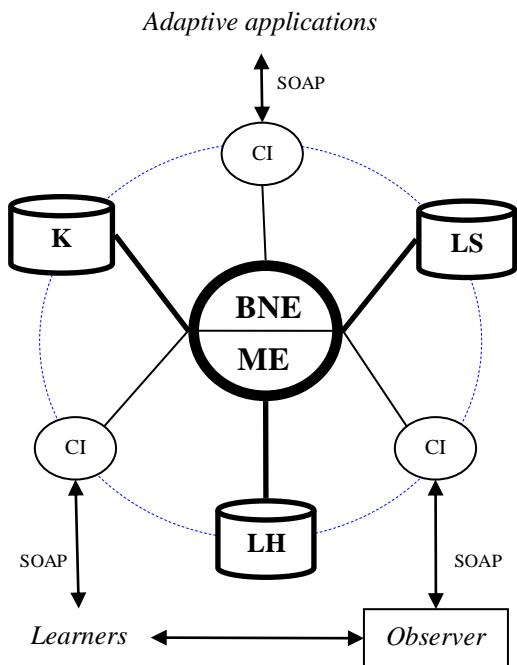


Figure VII.1.3. The architecture of Zebra

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 **Zebra** which constructs and manages TLM. Moreover, the adaptive learning system that interacts with Zebra is also implemented as e-learning web-based application **WOW**. 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 VII.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 both a

methodology for constructing user model and a whole learner model TLM (Nguyen L. , 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 sub-section 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.

Zebra will be developed towards ubiquitous user modeling. This future trend is mentioned in successive section VII.2.

VII.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 VII.2.1 (Heckmann D. , 2005, p. 6).

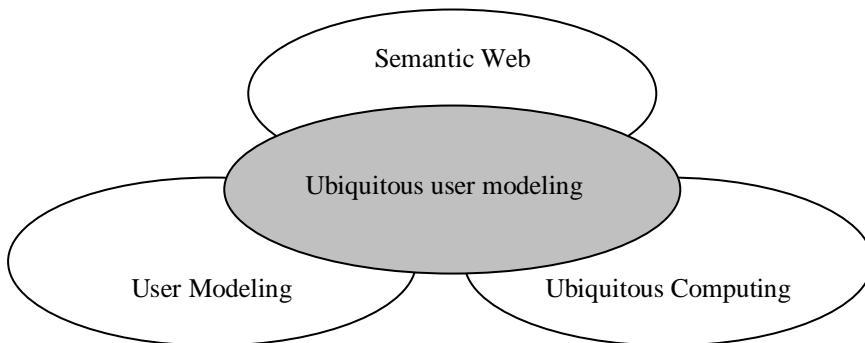


Figure VII.2.1. Ubiquitous user modeling

All contents relevant to ubiquitous user modeling are extracted from the PhD research “Ubiquitous User Modeling” of the author Dominikus Heckmann (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. I express my deep gratitude to the author Dominikus Heckmann for providing her/his great research. This section VII.2 includes following sub-sections:

- Sub-section [VII.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 [VII.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 [VII.2.3](#) sketches such ubiquitous world.
- Ubiquitous user modeling system is implemented as a ubiquitous user model service. Sub-section [VII.2.4](#) describes architecture and main actions of ubiquitous user model service. Author Dominikus Heckmann (Heckmann D., 2005, pp. 155-169) proposed and implemented this ubiquitous user model service.
- Sub-section [VII.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.

VII.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. Partial sub-sections [VII.2.1.1](#), [VII.2.1.2](#), and [VII.2.1.3](#) browse context-aware user modeling, ubiquitous computing, and semantic web, respectively.

VII.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 [VII.2.1.1.1](#) (Heckmann D., 2005, p. 15). Because user modeling is surveyed carefully in previous section [I.1](#), this sub-section [VII.2.1.1](#) only mentions context-aware user modeling when ubiquitous system always focuses on environment and context.

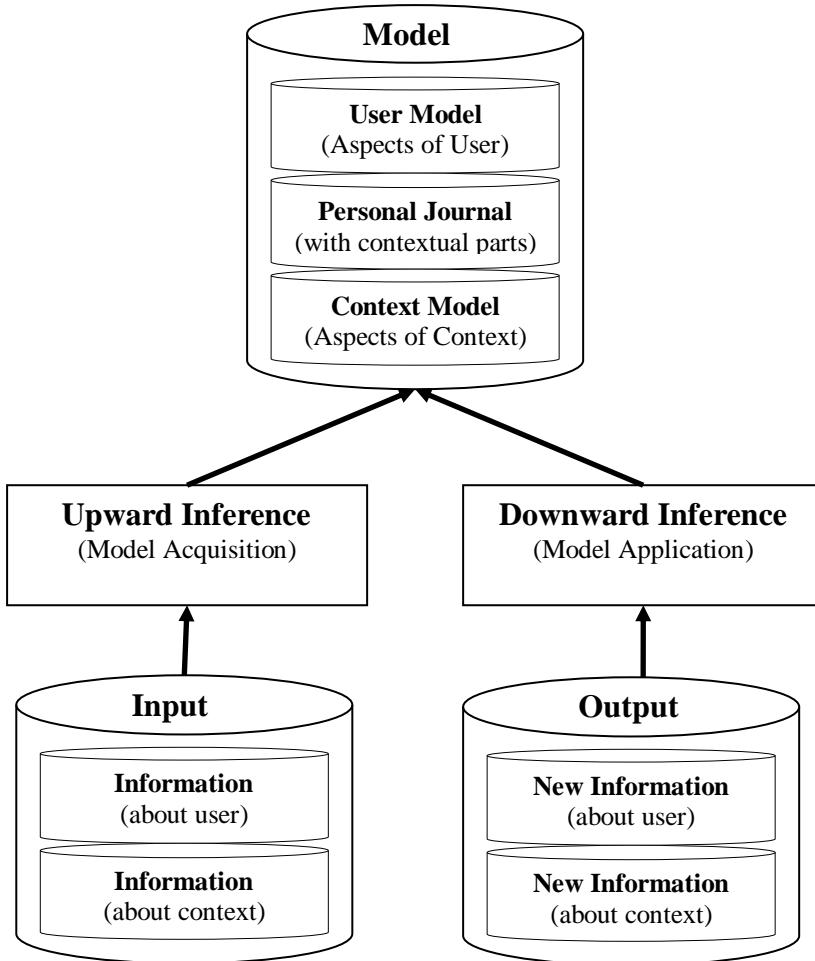


Figure VII.2.1.1.1. User modeling with integrated context-awareness

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 VII.2.1, ubiquitous user modeling is the intersection of user modeling, ubiquitous computing, and semantic web. The successive subsection VII.2.1.2 mentions ubiquitous computing.

VII.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 (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 VII.2.1.2.1 (Heckmann D. , 2005, p. 20).

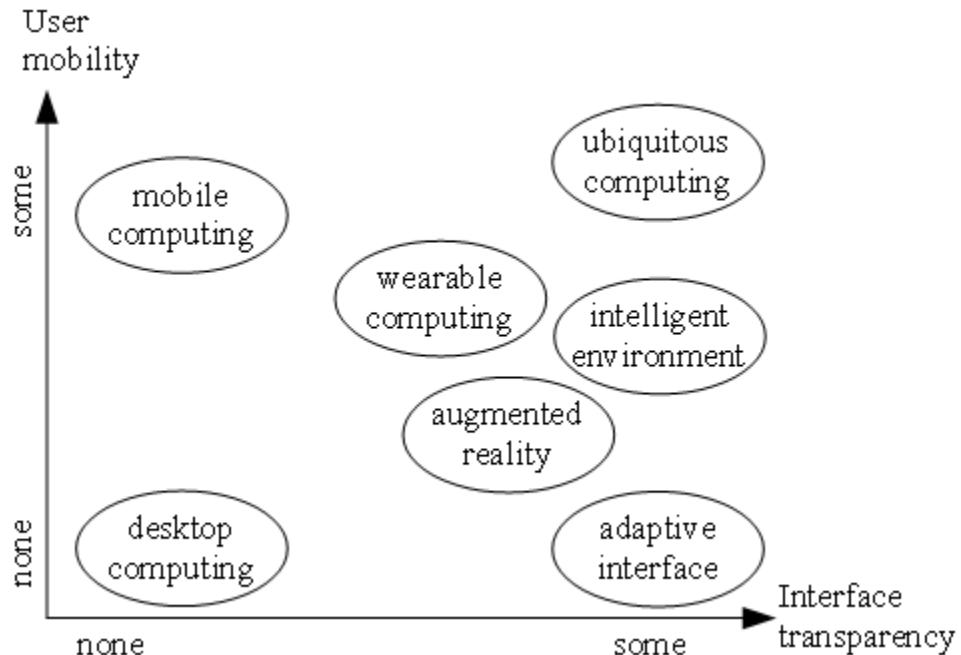


Figure VII.2.1.2.1. Spatial arrangement of ubiquitous computing similar fields according to two dimensions: user mobility and interface transparency

As shown in figure VII.2.1.2.1, 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 and Perry (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: 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 VII.2.1.2.2 depicts situation model comprising of user model, context model, and resource model.

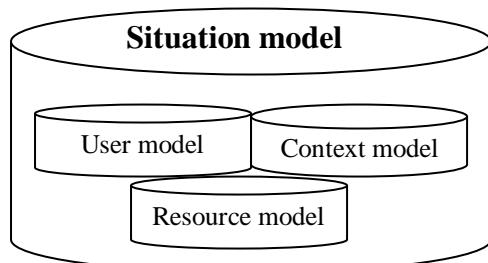


Figure VII.2.1.2.2. Situation model

However it can 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 VII.2.1.2.3 is the extended situation model (Heckmann D. , 2005, p. 63).

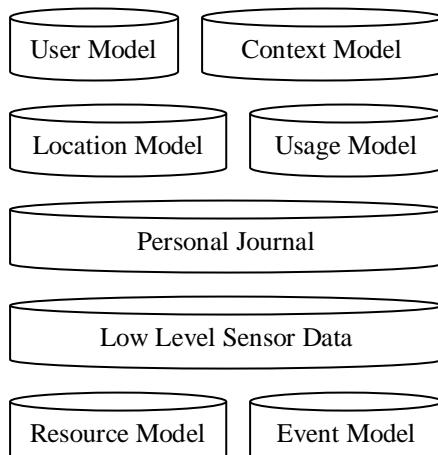


Figure VII.2.1.2.3. Extended situation model

The process called *situation modeling* process is responsible for constructing situation model. The following figure VII.2.1.2.4 (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*.

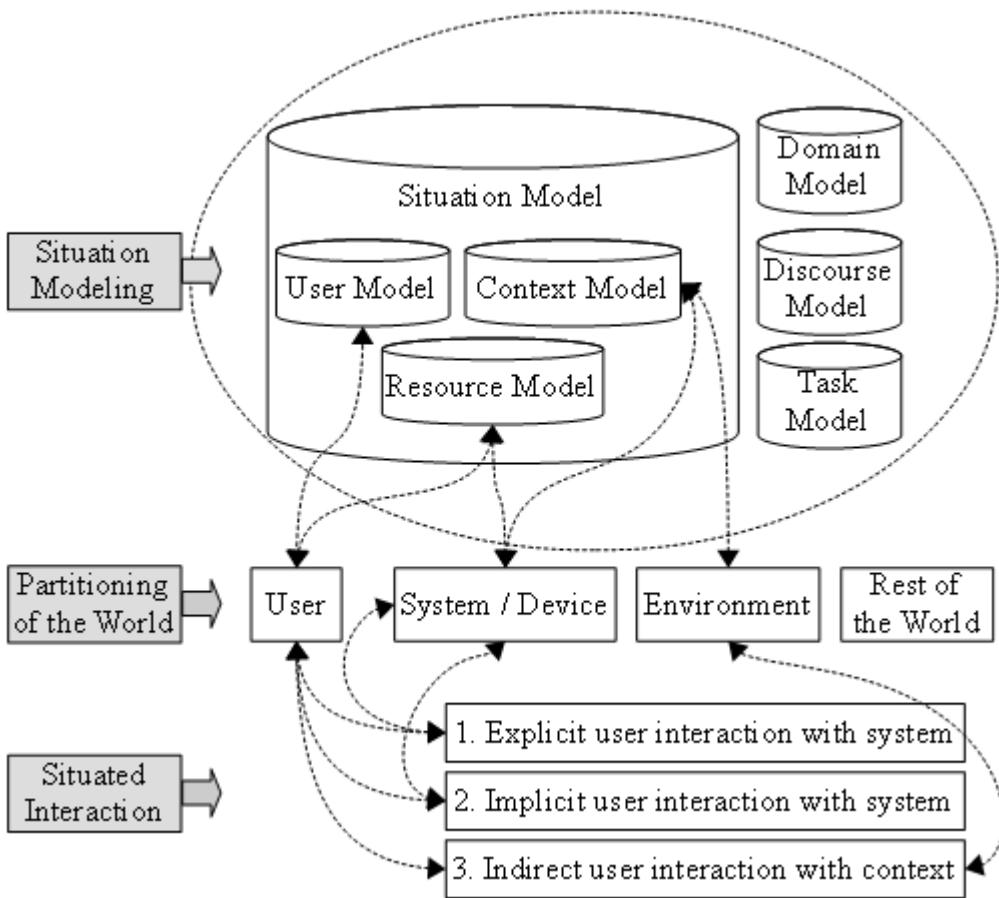


Figure VII.2.1.2.4. Situated interaction and situation modeling for ubiquitous computing

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 VII.2.1, ubiquitous user modeling is the intersection of user modeling, ubiquitous computing, and semantic web. The successive subsection VII.2.1.3 mentions semantic web.

VII.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). According to

W3Consortium, RDF is integrated into 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 (Daly, Forgue, & Hirakawa, 2004). In brief, RDF is the means 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 means 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 [VII.2.2](#) mentions knowledge representation of situation model.

VII.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 gives 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 construction 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 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 VII.2.2.1 expresses RDF triple that represents a statement.

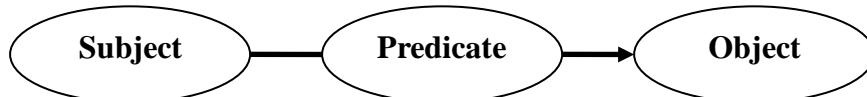


Figure VII.2.2.1. RDF triple

Each situational statement is represented as the onion model with five layers (Heckmann D. , 2005, pp. 58-59), shown in figure VII.2.2.2. 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.

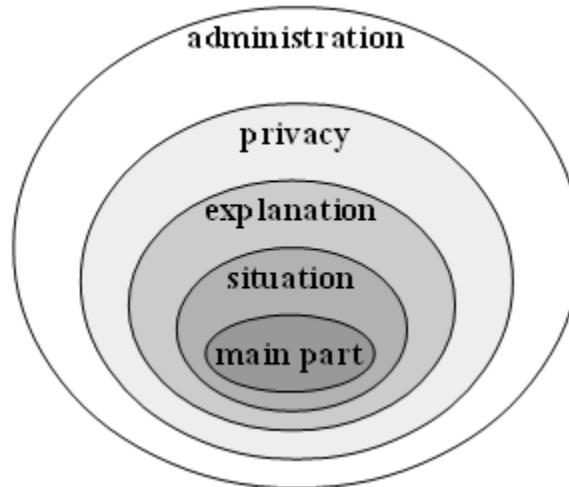


Figure VII.2.2.2. Onion model of statement

- *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 VII.2.2.1 (Heckmann D. , 2005, p. 59) shows the main part box.

Main part
Subject
Auxiliary
Predicate
Range
Object

Table VII.2.2.1. 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 becomes 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 [VII.2.2.2](#) (Heckmann D. , 2005, p. 61) shows the situation box.

Situation
Start
End
Durability
Location
Position

Table VII.2.2.2. Situation box

- *Explanation box* consists of attributes which are required by the user's right for explanation and by the conflict resolution mechanism for 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 (Heckmann D. , 2005, p. 62). 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 evidences that support statement. The attribute *confidence* provides the place to store the creator's expected level of confidence in statement. Table [VII.2.2.3](#) (Heckmann D. , 2005, p. 62) shows the explanation box.

Explanation
Source
Creator
Method
Evidence
Confidence

Table VII.2.2.3. 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 puts 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 VII.2.2.4 shows the privacy box.

Privacy
Key
Owner
Access
Purpose
Retention

Table VII.2.2.4. 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 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 VII.2.2.5 shows the administration box.

Administration
Id
Unique
Replaces
Group
Notes

Table VII.2.2.5. 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>
  </mainpart>
</statement>

```

```

        <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 VII.2.2.3 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 their values can be defined as RDF objects.

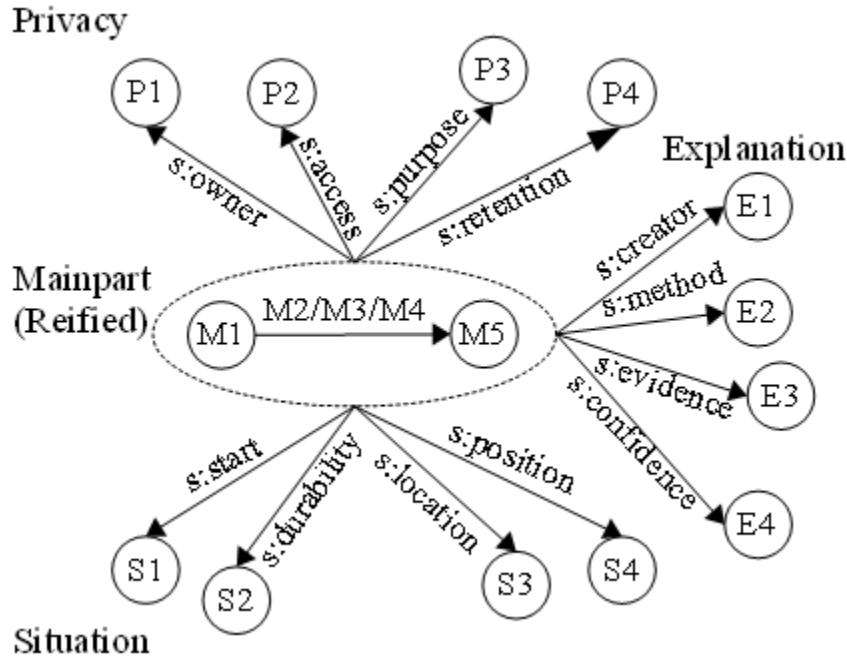


Figure VII.2.2.3. The schema of statement model in RDF graph notation

Based on situation model, ubiquitous user modeling system simulates the real world as virtual world that is called ubiquitous world. Successive sub-section [VII.2.3](#) sketches such ubiquitous world.

VII.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 called Ubiquitous World. Moreover statements in situation model refer to various ontologies. So we should glance over user model ontologies (sub-section [VII.2.3.1](#)) and ubiquitous world (sub-section [VII.2.3.2](#)).

VII.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 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 [VII.2.3.1.1](#) shows the situational statement represented by RDF triple.

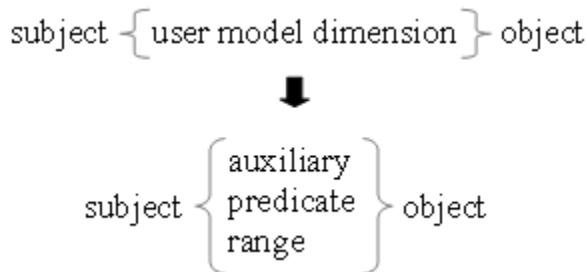


Figure VII.2.3.1.1. Situational statement represented by RDF triple

So user model dimensions such as auxiliary, predicate, range or any other information about user are entities in GUMO. The GUMO gives 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 VII.2.3.1.2 as follows:

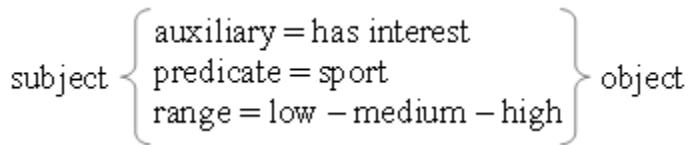


Figure VII.2.3.1.2. An example of statement represented by RDF triple

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 VII.2.3.1.1 (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 VII.2.3.1.1. Basic user dimensions in GUMO

Table VII.2.3.1.2 (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 VII.2.3.1.2. 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, and inference ontology.

When user model ontologies are defined according to GUMO, it is easy to describe ubiquitous world in successive sub-section [VII.2.3.2](#).

VII.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, and airport. 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 [VII.2.3.2.1](#) (Heckmann D. , 2005, p. 100).

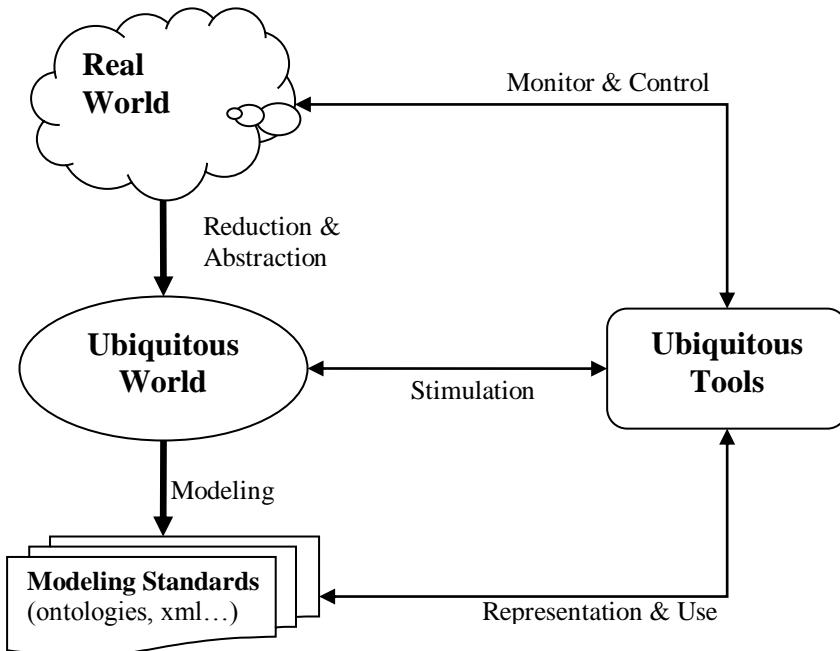


Figure VII.2.3.2.1. The correlation between UbisWorld and real world

There are three main processes 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 (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 VII.2.3.2.2 (Heckmann D. , 2005, p. 102) gives a combination of UbisWorld and ontologies.

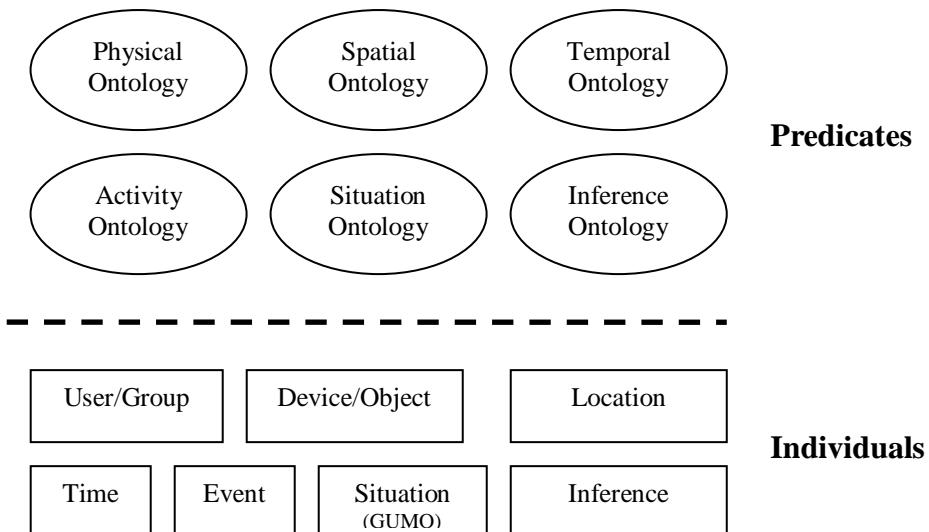


Figure VII.2.3.2.2. UbisWorld and ontologies

Ubiquitous user modeling system is implemented as ubiquitous user model service. Successive sub-section VII.2.4 describes architecture and main actions of ubiquitous user model service.

VII.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 VII.2.4.1 and VII.2.4.2, respectively. Author

Dominikus Heckmann (Heckmann D. , 2005, pp. 155-169) proposed and implemented this architecture.

VII.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 VII.2.4.1.1 (Heckmann D. , 2005, p. 156) depicts this architecture together such five boxes.

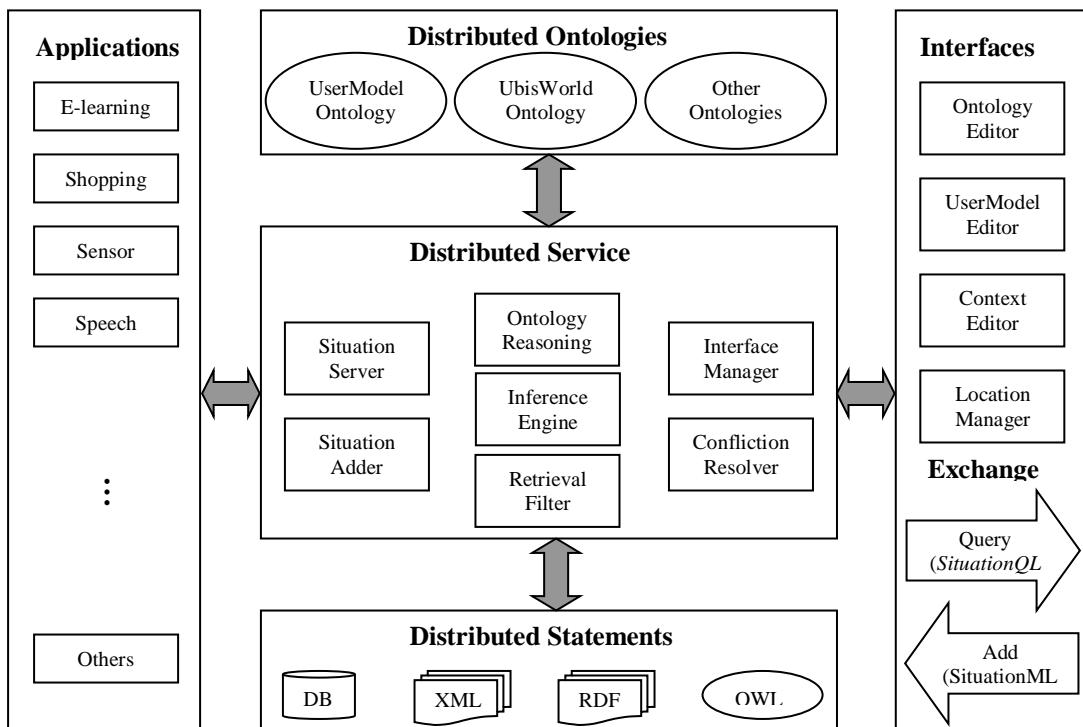


Figure VII.2.4.1.1. Architecture of ubiquitous user model service

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. 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. Please read the book “Principles of Distributed Database Systems” (Özsu & Valduriez, 2011, pp. 1-38) for more details about distributed database management system.

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 VII.2.4.1.1). 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 VII.2.4.1.1, the work flow of ubiquitous user model service is mentioned in successive sub-section VII.2.4.2.

VII.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 VII.2.4.2.1:

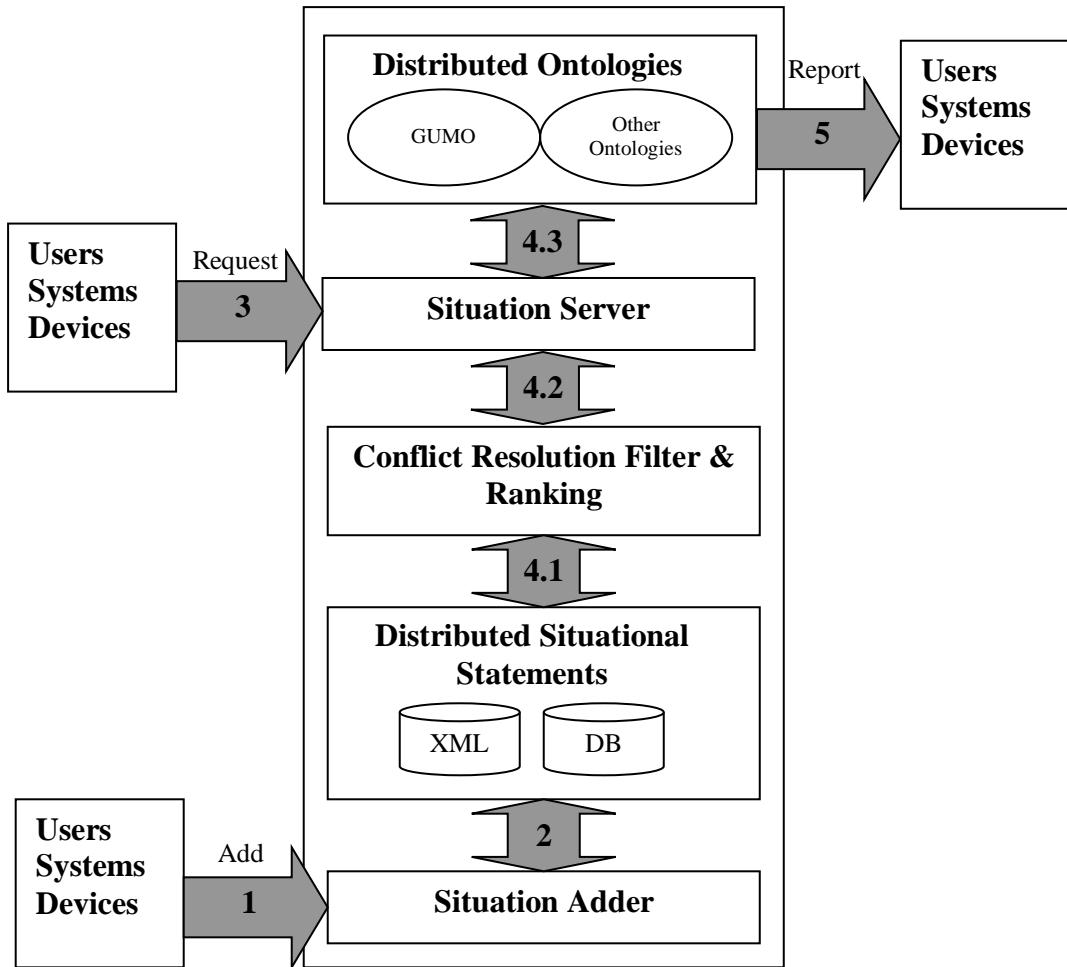


Figure VII.2.4.2.1. The work flow of ubiquitous user model service

The work flow includes steps below (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 [VII.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.

VII.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 VII.2.5.1 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, IV.4, and V.1.1 for more details about Bayesian network, hidden Markov model, and sequential pattern, respectively.
- *Secondly*, most 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.

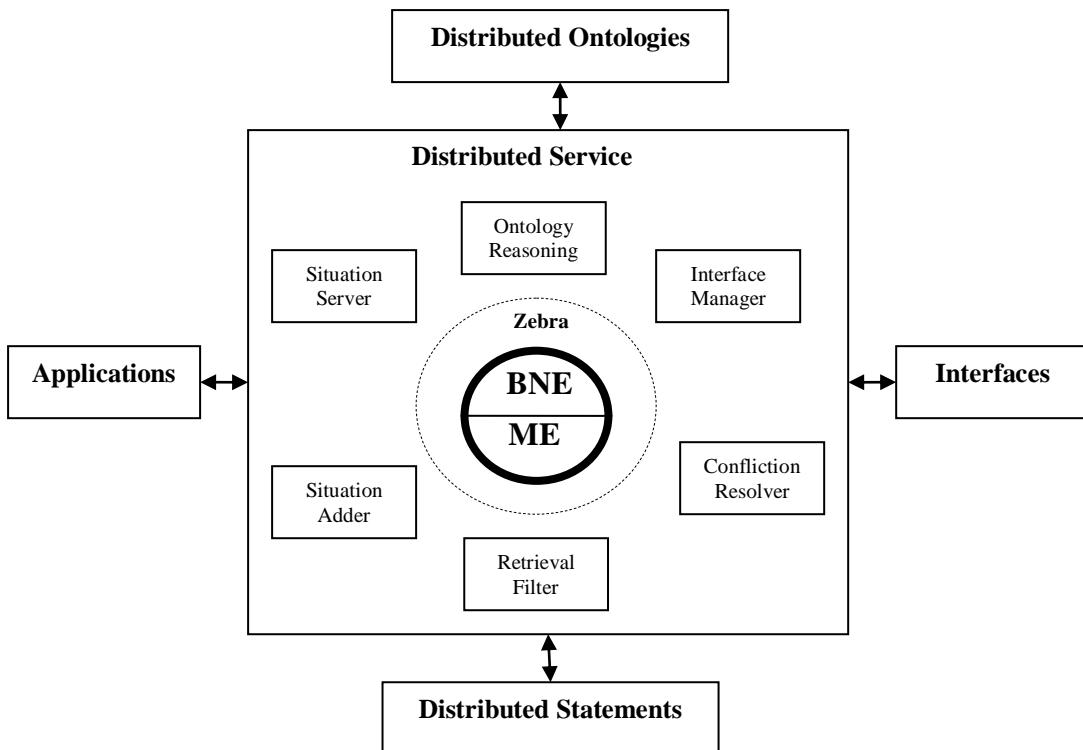


Figure VII.2.5.1. Incorporating Zebra into ubiquitous service

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 VII.2.5.2 interprets such five steps of communication protocol between Zebra and ubiquitous service.

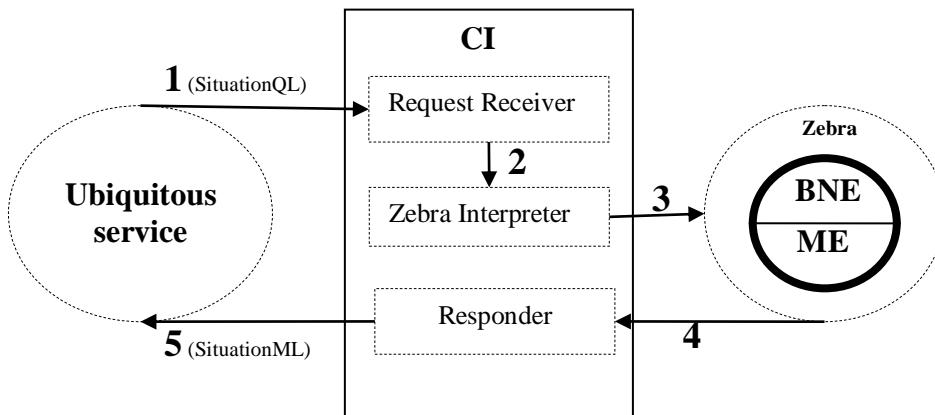


Figure VII.2.5.2. Request-Response communication protocol between Zebra and ubiquitous service

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 all aspects of the Triangular Learner Model (TLM) and its user modeling system Zebra, main subjects of this research, have already been introduced to you. The book focuses on mathematical viewpoint as its title “Mathematical Approaches to User Modeling” suggests. I think that it is better if there are some extra chapters reserved for purely mathematical subjects. The application of Bayesian network into constructing knowledge sub-model in chapter III is a tip of iceberg “Bayesian network”. The forthcoming chapter VIII, which is the full description of Bayesian network according to mathematical viewpoint, will be included in second edition of the book.

Thank you for your attention to the research.

References

- Abdelnur, A., & Hepper, S. (2003). *JavaTM Portlet Specification version 1.0*. Java Community Process.
- 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, January). 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. (1996). Methods and Techniques of Adaptive Hypermedia. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 6(2-3), 87-129.
- 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., & 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.
- Burden, R. L., & Faires, D. J. (2011). *Numerical Analysis* (9th Edition ed.). (M. Julet, Ed.) Brooks/Cole Cengage Learning.
- Carmona, C., Castillo, G., & Millán, E. (2008). Designing a Dynamic Bayesian Network for Modeling Students' Learning Styles. In P. Diaz, Kinshuk, I. Aedo, & E. Mora (Ed.), *2008 Eighth IEEE International Conference on Advanced Learning Technologies* (pp. 346-350). Santander, Spain: IEEE. doi:10.1109/ICALT.2008.116

- Carmona, C., Millán, E., Pérez-de-la-Cruz, J.-L., Trella, M., & Conejo, R. (2005). Introducing Prerequisite Relations in a Multi-layered Bayesian Student Model. In L. Ardissono, P. Brna, & A. Mitrovic (Ed.), *Lecture Notes in Computer Science*. 3538, pp. 347-356. Edinburgh: Springer Berlin Heidelberg. doi:10.1007/11527886_46
- Cheng, C.-C., Sha, F., & Saul, L. K. (2009). A Fast Online Algorithm for Large Margin Training of Continuous Density Hidden Markov Models. *Proceedings of the Tenth Annual Conference of the International Speech Communication Association (INTERSPEECH 2009)*, (pp. 668-671). Brighton. Retrieved May 28, 2016, from http://cseweb.ucsd.edu/~saul/papers/is09_hmm.pdf
- Cheng, J., Bell, D. A., & Liu, W. (1997, Januray 1). Learning Belief Networks from Data: An information theory based approach. (F. Golshani, K. Makki, C. Nicholas, & N. Pissinou, Eds.) *Proceedings of the sixth international conference on Information and knowledge management (CIKM '97)*, 325-331.
- Conati, C., Gertner, A., & Vanlehn, K. (2002, November 1). Using Bayesian Networks to Manage Uncertainty in Student Modeling. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 12(4), 371-417.
- Couvreur, C. (1996). *Hidden Markov Models and Their Mixtures*. Université Catholique de Louvain, Department of Mathematics. Louvain-la-Neuve: ResearchGate. Retrieved May 7, 2016, from https://www.researchgate.net/publication/2677995_Hidden_Markov_Models_and_Their_Mixtures
- Czepiel, S. A. (2002). *Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation*. Czepiel's website <http://czep.net>.
- Daly, J., Forgue, M.-C., & Hirakawa, Y. (2004, February 10). *World Wide Web Consortium Issues RDF and OWL Recommendations - Semantic Web emerges as commercial-grade infrastructure for sharing data on the Web*. Retrieved 2009, from World Wide Web Consortium website: <http://www.w3.org/2004/01/sws-pressrelease>
- 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: returning 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.
- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. (M. Stone, Ed.) *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1), 1-38.
- Díez, F. J., & Druzdzel, M. J. (2007). *Canonical Probabilistic Models*. National University for Distance Education, Department of Inteligencia Artificial.

- Madrid: Research Centre on Intelligent Decision-Support Systems. Retrieved May 9, 2016, from <http://www.cisiad.uned.es/techreports/canonical.pdf>
- 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.
- Gallová, J., & Kučerka, N. (n.d.). *Measurement of air humidity*. Technical Report, Comenius University in Bratislava, Faculty of Pharmacy.
- 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.
- Heckerman, D. (1995). *A Tutorial on Learning With Bayesian Networks*. Microsoft Corporation, Microsoft Research. Redmond: Microsoft Research. Retrieved from <ftp://ftp.research.microsoft.com/pub/dtg/david/tutorial.ps>

- 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.
- Honavar, V. G. (n.d.). *Sequential Minimal Optimization for SVM*. Iowa State University, Department of Computer Science. Ames, Iowa, USA: Vasant Honavar homepage.
- Honey, P., & Mumford, A. (2000). *Learning Styles Helper's Guide*. Maidenhead, Berkshire, England: Peter Honey Publications Limited.
- Huo, Q., & Lee, C.-H. (1997, March). On-Line Adaptive Learning of the Continuous Density Hidden Markov Model Based on Approximate Recursive Bayes Estimate. *IEEE Transactions on Speech and Audio Processing*, 5(2), 161-172. doi:10.1109/89.554778
- Jebara, T. (2015). *The Exponential Family of Distributions*. Columbia University, Computer Science Department. New York: Columbia Machine Learning Lab. Retrieved April 27, 2016, from <http://www.cs.columbia.edu/~jebara/4771/tutorials/lecture12.pdf>
- 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, June 1). 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. (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. (2001, March 27). 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., & Pohl, W. (1995). The User Modeling Shell System BGP-MS. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 4(2), 59-106.
- Kofman, V. (2009, November 02). *An Introduction to Java Enterprise Portals and Portlet Development*. Retrieved July 03, 2015, from The Flagship of The Developer.com Network: <http://www.developer.com/java/web/article.php/3846491/An-Introduction-to-Java-Enterprise-Portals-and-Portlet-Development.htm>
- 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.
- Kschischang, F. R., Frey, B. J., & Loeliger, H.-A. (2001, February). Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47(2), 498-519. doi:10.1109/18.910572
- 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, Maryland, USA: Pearson Education.
- Lee, C. H., Rabiner, L. R., Pieraccini, R., & Wilpon, J. G. (1990, April). Acoustic modeling for large vocabulary speech recognition. *Computer Speech & Language*, 4(2), 127-165. doi:10.1016/0885-2308(90)90002-N
- 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, June). Student assessment using Bayesian nets. (B. R. Gaines, D. Heckerman, A. Mamdani, & M. P. Wellman, Eds.) *International*

- Journal of Human-Computer Studies - Special issue: real-world applications of uncertain reasoning*, 42(6), 575-591.
- Mayo, M. J. (2001). *Bayesian Student Modelling and Decision-Theoretic Selection of Tutorial Actions in Intelligent Tutoring Systems*. PhD Thesis, University of Canterbury, New Zealand.
- 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.
- Millán, E., & Pérez-de-la-Cruz, J. L. (2002, June). A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation. (A. Kobsa, Ed.) *User Modeling and User-Adapted Interaction*, 12(2-3), 281-330. doi:10.1023/A:1015027822614
- Millán, E., Loboda, T., & Pérez-de-la-Cruz, J. L. (2010, July 29). Bayesian networks for student model engineering. (R. S. Heller, J. D. Underwood, & C.-C. Tsai, Eds.) *Computers & Education*, 55(4), 1663-1683. doi:10.1016/j.compedu.2010.07.010
- 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.
- Mitchell, T. M. (1997). *Machine Learning*. Burr Ridge, Illinois, USA: McGraw-Hill Science/Engineering/Math.
- Mitrovic, A. (1998, March 1). Learning SQL with a computerized tutor. (D. Joyce, & J. Impagliazzo, Eds.) *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education (SIGCSE '98)*, 30(1), 307-311.
- 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
- 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. (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. doi:10.1109/CSE.2009.112
- 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. Retrieved from <http://www.icl-conference.org/dl/proceedings/2009/archive.htm>
- 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. Retrieved from https://www.researchgate.net/publication/221186474_State_of_the_Art_of_Adaptive_Learning
- Nguyen, L. (2009). ZEBRA: A new User Modeling System for Triangular Model of Learners' Characteristics. In G. D. Magoulas, P. Charlton, D. Laurillard, K. Papanikolaou, & M. Grigoriadou (Ed.), *AIED 2009: 14th conference on Artificial Intelligence in Education, Proceedings of the Workshop on "Enabling creative learning design: how HCI, User Modeling and Human Factors Help"* (pp. 42-51). Brighton, United Kingdom: IOS Press Amsterdam, The Netherlands, The Netherlands. Retrieved from <https://goo.gl/cVzC6h>
- Nguyen, L. (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. doi:10.1007/978-3-642-13422-7_9
- 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. Retrieved from <http://socialscienceresearch.org/index.php/GJHSS/article/view/609>
- Nguyen, L. (2013). *Overview of Bayesian Network*. University of Technology, Ho Chi Minh city, Vietnam. Warri, Delta State, Nigeria: Science Journal Publication. doi:10.7237/sjms/105
- Nguyen, L. (2013, June 23-24). 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), 36-38. Retrieved from https://www.researchgate.net/publication/282685470_The_Bayesian_approach_and_suggested_stopping_criterion_in_Computerized_Adaptive_Testing
- Nguyen, L. (2014, May). 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). Retrieved from <http://www.srl-journal.org/paperInfo.aspx?ID=6933>
- Nguyen, L. (2014, April). *A User Modeling System for Adaptive Learning*. University of Science, Ho Chi Minh city, Vietnam. Abuja, Nigeria: Standard Research Journals. Retrieved from <http://standresjournals.org/journals/SSRE/Abstract/2014/april/Loc.html>
- Nguyen, L. (2014). A User Modeling System for Adaptive Learning - Abstract. *Interactive Collaborative Learning (ICL), 2014 International Conference on* (pp. 864-866). Dubai, UAE: IEEE Publisher. doi:10.1109/ICL.2014.7017887
- Nguyen, L. (2014, December 5). *A User Modeling System for Adaptive Learning - Demonstration*. (L. Nguyen, Director, & L. Nguyen, Performer) The 17th

- International Conference on Interactive Computer aided Learning (ICL2014), The 2014 World Engineering Education Forum, Engineering Education For A Global Community, Dubai, UAE.
- Nguyen, L. (2014). Evaluating Adaptive Learning Model. *Interactive Collaborative Learning (ICL), 2014 International Conference on* (pp. 818-822). Dubai, UAE: IEEE Publisher. doi:10.1109/ICL.2014.7017878
- Nguyen, L. (2014, May 28). User Model Clustering. (F. Shi, Ed.) *Journal of Data Analysis and Information Processing (JDAIP)*, 2(2), 41-48. doi:10.4236/jdaip.2014.22006
- Nguyen, L. (2016, April 27). Beta Likelihood Estimation and Its Application to Specify Prior Probabilities in Bayesian Network. (H. M. Srivastava, P. Bracken, R. Jedynak, anonymous, & S. Zimeras, Eds.) *British Journal of Mathematics*, 16(3), 1-21. doi:10.9734/BJMCS/2016/25731
- Nguyen, L. (2016, April 27). Beta Likelihood Estimation and Its Application to Specify Prior Probabilities in Bayesian Network. (T.-X. He, P. Bracken, J. C. Valverde, & J. Li, Eds.) *British Journal of Mathematics*, 16(3). doi:10.9734/BJMCS/2016/25731
- Nguyen, L. (2016, June 11). Continuous Observation Hidden Markov Model. (M. G. De Garcia, Ed.) *Kasmera Journal*, 44(6), 65-149. Retrieved from <http://kasmerajournal.com/show.php?v=44&i=6>
- Nguyen, L. (2016, June 17). Longest-path Algorithm to Solve Uncovering Problem of Hidden Markov Model. (L. Nguyen, & M. A. MELLAL, Eds.) *Special Issue "Some Novel Algorithms for Global Optimization and Relevant Subjects"*, *Applied and Computational Mathematics (ACM)*, 6(4-1), 39-47. doi:10.11648/j.acm.s.2017060401.13
- Nguyen, L. (2016, April 8). New version of CAT algorithm by maximum likelihood estimation. (B. N. Buszewski, Ed.) *Sylwan Journal*, 160(4), 218-244.
- Nguyen, L. (2016, February). Specifying Prior Probabilities in Bayesian Network by Maximum Likelihood Estimation method. (B. N. Buszewski, Ed.) *Sylwan Journal*, 160(2), 281-298.
- Nguyen, L. (2016, March 28). Theorem of SIGMA-gate Inference in Bayesian Network. (V. S. Franz, Ed.) *Wulfenia Journal*, 23(3), 280-289.
- Nguyen, L. (2016, June 17). Tutorial on Hidden Markov Model. (L. Nguyen, & M. A. MELLAL, Eds.) *Special Issue "Some Novel Algorithms for Global Optimization and Relevant Subjects"*, *Applied and Computational Mathematics (ACM)*, 6(4-1), 16-38. doi:10.11648/j.acm.s.2017060401.12
- Nguyen, L. (2016, June 17). Tutorial on Support Vector Machine. (L. Nguyen, & M. A. MELLAL, Eds.) *Special Issue "Some Novel Algorithms for Global Optimization and Relevant Subjects"*, *Applied and Computational Mathematics (ACM)*, 6(4-1), 1-15. doi:10.11648/j.acm.s.2017060401.11
- Nguyen, L., & Do, P. (2008). Learner Model in Adaptive Learning. *The 2018 World Congress on Science, Engineering and Technology (WCSET2008)*. 35, pp. 396-400. Paris, France: World Congress on Science, Engineering and Technology (WASET). Retrieved from https://www.researchgate.net/publication/282679719_Learner_Model_in_Adaptive_Learning
- Nguyen, L., & Do, P. (2009). Combination of Bayesian Network and Overlay Model in User Modeling. (M. E. Auer, Ed.) *International Journal of Emerging Technologies in Learning (iJET)*, 4(4), 41-45. doi:10.3991/ijet.v4i4.684

- 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 Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'09)* (pp. 324-329). Monte Carlo Resort, Las Vegas, Nevada, USA: CSREA Press USA. Retrieved from http://www.researchgate.net/publication/220834553_Evolution_of_Parameters_in_Bayesian_Overlay_Model
- 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. doi:10.1109/DEST.2009.5276694
- Nguyen, V. H. (1999). *Linear Algebra*. Hanoi, Vietnam: Hanoi National University Publishing House.
- NIST. (2008, March). International System of Units (SI). (330 - 2008 Edition). (B. N. Taylor, & A. Thompson, Eds.) Gaithersburg, Maryland, USA: National Institute of Standards and Technology.
- Oracle. (2009). *The Java™ Tutorials*. Retrieved 2009, from Oracle Help Center website for Java Standard Edition: <http://docs.oracle.com/javase/tutorial>
- Oracle. (n.d.). *Java language*. (Oracle Corporation) Retrieved December 25, 2014, from Java website: <https://www.oracle.com/java>
- 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.
- Pearl, J. (1986, September). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3), 241-288. doi:10.1016/0004-3702(86)90072-X
- Platt, J. C. (1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. Microsoft Research.
- 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.
- Ramage, D. (2007). *Hidden Markov Models Fundamentals*. Lecture Notes, Stanford University, USA.
- 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.
- Sean, B. (2009). *The Expectation Maximization Algorithm - A short tutorial*. University of Notre Dame, Indiana, Department of Electrical Engineering. Sean Borman's Homepage.
- Sha, F., & Saul, L. K. (2006). Large Margin Hidden Markov Models for Automatic Speech Recognition. (B. Schölkopf, J. C. Platt, & T. Hoffman, Eds.) *Advances in Neural Information Processing Systems (NIPS 2006)*, 19, 1249-1256. Retrieved May 29, 2016, from http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2006_143.pdf
- Sha, F., & Saul, L. K. (2009). Large margin training of continuous-density hidden Markov models. In J. Keshet, S. Bengio, J. Keshet, & S. Bengio (Eds.), *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods* (pp. 101-114). Chichester, UK: Wiley & Sons. doi:10.1002/9780470742044.ch7
- 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. (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, . . . D. Winer, 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. (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. (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. (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. (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>
- W3Schools. (n.d.). *AJAX Tutorial*. Retrieved July 03, 2015, from W3Schools website: <http://www.w3schools.com/ajax/>
- 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.). *Logistic Distribution*. (Wolfram Research) Retrieved September 4, 2016, from MathWorld--A Wolfram Web Resource: <http://mathworld.wolfram.com/LogisticDistribution.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
- Wikipedia. (2006). *Bayesian inference*. (Wikimedia Foundation) Retrieved 2007, from Wikipedia website: http://en.wikipedia.org/wiki/Bayesian_inference
- Wikipedia. (2009, January 4). *Artificial neural network*. (Wikimedia Foundation) Retrieved 2009, from Wikipedia website: http://en.wikipedia.org/wiki/Artificial_neural_network
- Wikipedia. (2014, October 16). *Beta function*. (Wikimedia Foundation) Retrieved November 9, 2014, from Wikipedia website: 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
- 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
- Wikipedia. (2014, October 12). *Digamma function*. (Wikimedia Foundation) Retrieved November 9, 2014, from Wikipedia website: 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
- Wikipedia. (2014, October 1). *Indeterminate form*. (Wikimedia Foundation) Retrieved November 11, 2014, from Wikipedia website: 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
- 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
- Wikipedia. (2014, September 11). *Learning management system*. (Wikimedia Foundation) Retrieved September 18, 2014, from Wikipedia website: 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

- 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
- 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
- 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: https://en.wikipedia.org/wiki/Polygamma_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, October 5). *Web page*. (Wikimedia Foundation) Retrieved October 7, 2014, from Wikipedia website: 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>
- Wikipedia. (2015, November 22). *Factor graph*. (Wikimedia Foundation) Retrieved February 8, 2017, from Wikipedia website: https://en.wikipedia.org/wiki/Factor_graph
- Wikipedia. (2015, July 18). *Team building*. (Wikimedia Foundation) Retrieved July 22, 2015, from Wikipedia website: https://en.wikipedia.org/wiki/Team_building
- Wikipedia. (2015, June 7). *Trigamma function*. (Wikimedia Foundation) Retrieved April 9, 2016, from Wikipedia website: https://en.wikipedia.org/wiki/Trigamma_function
- Wikipedia. (2016, June 10). *Abel-Ruffini theorem*. (Wikimedia Foundation) Retrieved June 26, 2016, from Wikipedia website: https://en.wikipedia.org/wiki/Abel%E2%80%93Ruffini_theorem
- Wikipedia. (2016, March September). *Exponential family*. (Wikimedia Foundation) Retrieved 2015, from Wikipedia website: https://en.wikipedia.org/wiki/Exponential_family
- Wikipedia. (2016, June 2). *Logic gate*. (Wikimedia Foundation) Retrieved June 4, 2016, from Wikipedia website: https://en.wikipedia.org/wiki/Logic_gate

References

- Wikipedia. (2016, August 25). *Logistic distribution*. (Wikimedia Foundation) Retrieved September 4, 2016, from Wikipedia website: https://en.wikipedia.org/wiki/Logistic_distribution
- Wikipedia. (2016, April 12). *Student's t-distribution*. (Wikimedia Foundation) Retrieved April 24, 2016, from Wikipedia website: https://en.wikipedia.org/wiki/Student%27s_t-distribution
- Wikipedia. (2018, March 30). *Beta distribution*. (Wikimedia Foundation) Retrieved 2008, from Wikipedia website: https://en.wikipedia.org/wiki/Beta_distribution
- 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.
- Wolfram. (n.d.). Mathematica. *Wolfram Mathematica*, 10(4). Wolfram Research. Retrieved March 2016, from <http://www.wolfram.com/mathematica>
- Wolfram|Alpha. (n.d.). *Wolfram|Alpha Mathematics Engine*. (Wolfram Alpha LLC-A Wolfram Research Company) Retrieved December 16, 2014, from Wolfram|Alpha: <http://www.wolframalpha.com>
- Zaki, M. J. (2001, January 1). 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.

Appendices

A. List of Tables

Table I.1.1.2.1. Some common learning styles	4
Table I.1.1.2.2. Eight forms of aptitudes	5
Table I.3.1.2.1. CPT of a YACF cluster node.....	28
Table I.3.1.2.2. CPT of a child node Y given cluster node H.....	29
Table I.3.3.1. Conditional probability table of $relevantSS_{c,p}$ given $relevantIS_{c,p}$	39
Table II.2.4.1. Relationships of two engines, TLM, and 9 daemons	59
Table II.2.4.2. Responsibilities of 9 daemons.....	60
Table III.1.2.1. Prior probabilities (CPT) of nodes C , O and I	99
Table III.1.2.2. CPT of node J	100
Table III.1.2.3. CPT of evidence node E	101
Table III.1.4.1.1. Sufficient diagnostic proposition	112
Table III.1.4.2.1. Binary arrangements	120
Table III.1.4.2.2. Code snippet generating all binary arrangements.....	121
Table III.1.4.2.3. Bi-inferences for AND-gate, OR-gate, NAND-gate, NOR-gate, XOR-gate, XNOR-gate, and U-gate	133
Table III.1.4.3.1. Diagnostic theorem	139
Table III.2.1. ECA and CA rules for Bayesian inference	154
Table III.3.1.1. Evidence sample corresponding to 5 trials (sample of size 5)	172
Table III.3.1.2. Posterior density functions calculated based on count numbers s_{ij} and t_{ij}	172
Table III.3.1.3. Updated CPT (s) of X_1 and X_2	172
Table III.3.2.1. Evidence sample with missing data	173
Table III.3.2.2. New split evidences for missing data.....	174
Table III.3.2.3. Complete evidence sample in E-step of EM algorithm	174
Table III.3.2.4. Counters s_{11} , t_{11} , s_{21} , t_{21} , s_{22} , and t_{22} from estimated values (of missing values).....	175
Table III.3.2.5. Posterior density functions and posterior probabilities are updated in M-step of EM algorithm	175
Table III.3.3.1. All variables and their density functions, prior probabilities	180
Table III.3.3.2. All parameters of prior density functions.....	181
Table III.3.3.3. Incomplete sample with evidence $D = (E=1)$	182
Table III.3.3.4. New split evidences for missing values of O , I , and J	182
Table III.3.3.5. Complete sample with evidence $D = (E=1)$	184
Table III.3.3.6. Counters s_{ij} and t_{ij} (s) from estimated values.....	185
Table III.3.3.7. Posterior density functions and posterior probabilities are evolved based on counters s_{ij} and t_{ij}	185
Table III.4.2.1. Six steps of new algorithm for modeling and inferring user's knowledge by using DBN.....	191
Table III.4.2.2. The weights relating $x_1[t]$, $x_2[t]$, and $x_3[t]$ are normalized.....	196
Table III.4.2.3. CPT (s) of $X[t-1] = \{x_1[t-1], x_2[t-1], x_3[t-1]\}$	198
Table III.4.2.4. CPT (s) of $X[t] = \{x_1[t], x_2[t], x_3[t], d_1[t]\}$	199
Table III.4.2.5. The results of probabilistic inference – posterior probabilities	204
Table III.5.3.1. Iterative algorithm solving simplest equations specified by formula III.5.3.1 for estimating parameters a and b	223

Table III.5.4.1. The evidences corresponding to 5 trials.....	224
Table III.5.4.2. The values of L_{ij} corresponding to beta density function β_1 , β_2 , and β_3	225
Table III.5.4.3. The normal biases of (a_1, b_1) with respect to β_1	226
Table III.5.4.4. The normal biases of (a_2, b_2) with respect to β_2	226
Table III.5.4.5. The normal biases of (a_3, b_3) with respect to β_3	226
Table III.5.5.1. The evidences corresponding to 5 trials.....	230
Table III.5.5.2. The normal biases of (a_1, b_1) with respect to β_1	231
Table III.5.5.3. The normal biases of (a_2, b_2) with respect to β_2	231
Table III.5.5.4. The normal biases of (a_3, b_3) with respect to β_3	232
Table IV.4.1. Transition probability matrix A	245
Table IV.4.2. Uniform initial state distribution \prod	245
Table IV.4.3. Observation probability matrix B	245
Table IV.4.1.1. Forward-backward procedure based on forward variable to calculate the probability $P(O/\Delta)$	253
Table IV.4.1.2. Forward-backward procedure based on backward variable to calculate the probability $P(O/\Delta)$	256
Table IV.4.2.1. Individually optimal procedure to solve uncovering problem.....	259
Table IV.4.2.2. Viterbi algorithm to solve uncovering problem.....	263
Table IV.4.2.3. Longest-path algorithm	271
Table IV.4.2.4. Advanced longest-path algorithm.....	273
Table IV.4.3.1.1. General EM algorithm.....	278
Table IV.4.3.2.1. EM algorithm for HMM learning problem.....	293
Table IV.4.3.2.2. Proposed implementation of EM algorithm for learning HMM with terminating criterion $P(O/\Delta)$	296
Table IV.4.3.2.3. HMM parameters resulted from the first iteration of EM algorithm	302
Table IV.4.3.2.4. HMM parameters resulted from the first iteration and the second iteration of EM algorithm	308
Table IV.4.3.2.5. HMM parameters along with terminating criteria after 10 iterations of EM algorithm	309
Table IV.4.3.2.6. HMM parameters of weather example learned from EM algorithm	309
Table IV.5.1. EM algorithm applied into learning continuous observation HMM parameter with single PDF	318
Table IV.5.2. Observation probability parameters for weather example	319
Table IV.5.3. Continuous observation HMM parameters resulted from the first iteration of EM algorithm	323
Table IV.5.4. Continuous observation HMM parameters resulted from the first iteration and the second iteration of EM algorithm	328
Table IV.5.5. Continuous observation HMM parameters along with terminating criteria after 14 iterations of EM algorithm	329
Table IV.5.6. Continuous observation HMM parameters of weather example learned from EM algorithm.....	329
Table IV.5.7. EM algorithm applied into learning continuous observation HMM parameter with mixture PDF.....	339
Table IV.5.8. Observation probability parameters for weather example in case of mixture model	340
Table IV.5.9. Mixture HMM parameters resulted from the first iteration of EM algorithm.....	349

Table IV.5.10. Mixture HMM parameters resulted from the first iteration and the second iteration of EM algorithm	357
Table IV.5.11. Mixture HMM parameters along with terminating criteria after 14 iterations of EM algorithm.....	359
Table IV.5.12. Mixture HMM parameters of weather example learned from EM algorithm.....	359
Table IV.6.1. Transition probability matrices: $A^{(1)}, A^{(2)}, A^{(3)}$	367
Table IV.6.2. Observation probability matrices: $B^{(1)}, B^{(2)}, B^{(3)}$	369
Table IV.6.3. Learning objects selected	371
Table IV.6.4. Sequence of student observations	371
Table IV.6.5. Sequence of state transitions	375
Table V.1.1. Learning sessions → learning sequences	379
Table V.1.1.1.1. Large itemsets are mapped to integers.....	381
Table V.1.1.1.2. Transformed sequences.....	381
Table V.1.1.1.3. Candidate 2-sequences and large 2-sequences together their supports	382
Table V.1.1.1.4. Candidate 3-sequences and large 3-sequences together their supports	383
Table V.1.1.1.5. Large sequences resulted from sequence phrase	383
Table V.1.1.1.6. Sequential patterns resulted from AprioriAll algorithm	384
Table V.1.1.1.7. Cracking sequence database into vertical format data sets.....	385
Table V.1.1.1.8. (1). frequent itemsets (o) and (p) are associated with pairs (sid, eid). (2). 2-sequence $\langle op \rangle$ are found by joining itemsets (o), (p) in (1)	385
Table V.1.1.2.1. Frequent sequences mined from projected databases	386
Table V.1.2.1. Sequential rules	387
Table V.1.2.2. Recommended learning concepts constructed from sequential rules	388
Table V.2.1. Proposed approach to discover user interests	389
Table V.2.2.1.1. Ideology of SMO algorithm	399
Table V.2.2.1.2. SMO algorithm optimizes jointly two Lagrange multipliers	406
Table V.2.2.1.3. SMO algorithm	408
Table V.2.2.1.4. k couple (W_i^*, b_i^*) corresponds with k class $\{c_1, c_2, \dots, c_k\}$	409
Table V.2.2.1.5. Classifying document D with multi-classes	409
Table V.2.2.1.6. Term frequencies of documents (SVM)	410
Table V.2.2.1.7. Training corpus (SVM).....	410
Table V.2.2.2.1. Training corpus – Term frequencies of documents (decision tree)	417
Table V.2.2.2.2. Training corpus – Normalized term frequencies (decision tree)....	417
Table V.2.2.2.3. Training corpus – Nominal term frequencies.....	418
Table V.2.2.2.4. Reduced training corpus.....	423
Table V.2.2.2.5. Information gains at the second splitting	423
Table V.2.2.2.6. Classification rules derived from decision tree induction.....	424
Table V.2.2.3.1. Back-propagation algorithm for learning neural network	430
Table V.2.2.3.2. Training corpus – Term frequencies of documents (neural network)	431
Table V.2.2.3.3. Training corpus – Normalized term frequencies (neural network)	432
Table V.2.2.3.4. Results from training process based on back-propagation algorithm	434
Table V.2.3.1. User's searching history.....	436
Table V.2.3.2. 1-itemsets	436
Table V.2.3.3. Maximum frequent itemset that user searches	436
Table V.2.3.4. Interesting document vector.....	437

Table V.2.3.5. Interesting document vector is normalized	437
Table V.2.3.6. Nominal interesting document vector.....	437
Table V.3.1.1. Dissimilarities between user vectors and means	441
Table V.3.1.2. Dissimilarities between user vectors and means at the second running time of k -mean algorithm.....	442
Table V.3.1.3. Dissimilarities between user vectors and means at the third running time of k -mean algorithm.....	444
Table V.3.2.1. Values of graph nodes	446
Table V.3.2.2.1. Values of nodes in Bayesian network.....	449
Table V.3.3.1. Four user models	451
Table V.3.3.2. Cosine similarities between user vectors and medoids.....	456
Table V.3.3.3. Cosine similarities between user vectors and medoids at the second running time of k -medoid algorithm	458
Table V.3.3.4. Cosine similarities between user vectors and medoids at the third running time of k -medoid algorithm	459
Table VI.1.3.1.1. Computerized adaptive testing (CAT) algorithm	469
Table VI.1.3.3.1. Advanced CAT algorithm	486
Table VI.1.3.3.2. A multi-user test with 5 items and 4 examinees	487
Table VI.1.3.3.3. Results of multi-user test with 5 items and 4 examinees	490
Table VI.2.1.3.1. Rating matrix as collection of users' feedbacks	497
Table VI.2.1.3.2. Rating matrix is shrunk by projecting it onto its eigenvectors	498
Table VII.2.2.1. Main part box	513
Table VII.2.2.2. Situation box	513
Table VII.2.2.3. Explanation box.....	513
Table VII.2.2.4. Privacy box.....	514
Table VII.2.2.5. Administrator box.....	514
Table VII.2.3.1.1. Basic user dimensions in GUMO	517
Table VII.2.3.1.2. User model interest categories in GUMO	518

B. List of Figures

Figure I.1.1. User modeling	2
Figure I.1.2. Learner profile and learner model in adaptation.....	2
Figure I.1.2.1.1. An example of stereotypes of Java learner	6
Figure I.1.2.2.1. An example of overlay model having six concepts.....	7
Figure I.1.2.4.1. Illustration of perturbation model.....	8
Figure I.2.2.1.1. General Architecture of ITS.....	13
Figure I.2.2.3.1. User modeling component in ANATOM-TUTOR	14
Figure I.2.3.1. Adaptive representation	15
Figure I.2.3.2. Adaptive navigation	16
Figure I.2.3.3. A typical example of adaptive navigation	17
Figure I.2.3.1.1. General architecture of AEHS	18
Figure I.2.3.3.1. AHAM – The architecture of AHA!	21
Figure I.2.3.3.2. Implemented framework of AHA!	21
Figure I.2.3.3.3. AHA! engine	22
Figure I.3.1.1. An example of dependency graph.....	24
Figure I.3.1.2. The levels of <i>KI</i> (s).....	25
Figure I.3.1.3. Root tree.....	25
Figure I.3.1.4. Clustering approach.....	26
Figure I.3.1.1.1. YACF method	27
Figure I.3.1.2.1. Cluster node	27
Figure I.3.2.1. Student modeling in Andes	30
Figure I.3.2.1.1. Sample problem to find normal force.....	31
Figure I.3.2.1.2. Solution graph in Andes	32
Figure I.3.2.2.1. Relationship between rule and context-rule nodes.....	33
Figure I.3.2.2.2. Relationship between nodes in task-specific part	34
Figure I.3.2.2.3. Mutually exclusive strategy	35
Figure I.3.2.2.4. Strategy node.....	35
Figure I.3.2.2.5. Prior/Posterior probabilities in the task-specific part	36
Figure I.3.3.1. Architecture of SQL-Tutor	37
Figure I.3.3.2. Bayesian network in SQL-Tutor	39
Figure II.1.2.1. The architecture of GUMS	44
Figure II.1.3.1. The architecture of BGP-MS	46
Figure II.1.3.2. The architecture of CUMULATE.....	47
Figure II.1.3.3. The architecture of Personis	48
Figure II.1.3.4. The architecture of LDAP-UMS	49
Figure II.2.1.1. Triangular Learner Model	51
Figure II.2.1.2. extended Triangular Learner Model	52
Figure II.2.2.1. Modeling task is similar to profile mining task.....	53
Figure II.2.2.2. The modeling process in Zebra	54
Figure II.2.2.3. The architecture of Zebra	55
Figure II.2.3.1. The new architecture of AEHS and the interaction between AEHS and Zebra	57
Figure II.2.3.2. Steps in adaptation process with support of Zebra	58
Figure II.2.4.1. Zebra is running	61
Figure II.2.4.2. Zebra control panel	62
Figure II.2.4.3. Monitoring a learner	63
Figure II.2.4.4. TLM of given learner	64
Figure II.2.4.5. Learning style sub-model constructed by hidden Markov model	65

Figure II.2.4.6. Learning history sub-model stores coarse information in XML files	66
Figure II.2.4.7. Learning concept recommendation.....	67
Figure II.2.4.8. Learning path recommendation	68
Figure II.2.4.9. Discovering user interests based on document classification	69
Figure II.2.4.10. Constructing user groups.....	70
Figure II.2.4.11. Evaluation of learner's knowledge	71
Figure II.2.4.12. Evaluation of learner's knowledge in detailed	72
Figure II.2.4.13. Updating personal information.....	73
Figure II.2.4.14. Learning report	74
Figure II.2.4.15. Mailing list tool.....	75
Figure II.2.4.16. Feedback report	75
Figure II.2.4.17. WOW login web page.....	77
Figure II.2.4.18. Typical adaptive learning web page supported by WOW.....	78
Figure II.2.4.19. Leaner does online test via WOW	79
Figure II.2.4.20. Result of online test	80
Figure II.2.4.21. Knowledge evaluation is increased after learner finishes test successfully	81
Figure II.2.4.22. Posterior probability of given concept	82
Figure II.2.4.23. Search engine in WOW	83
Figure II.2.4.24. WOW feedback tool.....	84
Figure II.2.4.25. Collaborative area for collaborative learning	85
Figure II.2.4.26. Thin client connecting Zebra server	86
Figure III.1.1.1. Bayesian network (a classic example about wet grass)	93
Figure III.1.2.1. Structure of Bayesian overlay model for Java course	99
Figure III.1.2.2. Bayesian network (Bayesian overlay model) of Java course with full of CPT (s)	101
Figure III.1.3.1. Sigma graph	103
Figure III.1.3.2. Sigma condition	104
Figure III.1.3.3. SIGMA-gate model	105
Figure III.1.3.4. An example of sigma graph	109
Figure III.1.3.5. Bayesian network transformed from sigma graph	110
Figure III.1.4.1.1. Diagnosis and prediction with hypothesis X and evidence D	112
Figure III.1.4.2.1. Simple graph or simple network	116
Figure III.1.4.2.2. Extended X-gate network with accountable variables A_i (s)	117
Figure III.1.4.2.3. Bi-weight simple graph.....	132
Figure III.1.4.3.1. Augmented X-D network.....	135
Figure III.1.4.3.2. Direct SIGMA-gate diagnostic network (direct SIGMA-D network).....	142
Figure III.1.4.3.3. Diagnostic relationship with multiple evidences (M-E-D network)	146
Figure III.1.4.3.4. M-HE-D network	148
Figure III.2.1. Bayesian overlay model of Java course with full of CPT (s)	153
Figure III.3.1.1. Beta density functions with various parameters a and b	159
Figure III.3.1.2. The simple augmented BN with only one hypothesis node X	160
Figure III.3.1.3. The sample $\mathcal{D}=(X^{(1)}, X^{(2)}, \dots, X^{(m)})$ of size m	161
Figure III.3.1.4. BN (a) and complex augmented BN (b)	164
Figure III.3.1.5. Expanded binomial BN sample of size m	167
Figure III.3.1.6. Updated version of BN (a) and augmented BN (b).....	173
Figure III.3.2.1. Updated version of BN (a) and augmented BN (b) in case of missing data	178

Figure III.3.3.1. BN structure as weighted graph (a) and augmented BN (b) of Java course.....	179
Figure III.3.3.2. Augmented BN with initial parameters in full.....	182
Figure III.3.3.3. Evolutional version of BN (a) and augmented BN (b) for Java course.....	186
Figure III.4.1.1. DBN for $t = 0, 1, 2$	189
Figure III.4.2.1. The BN sample with full of weights	192
Figure III.4.2.2. Initial BN G_0 derived from BN in figure III.4.2.1	192
Figure III.4.2.3. Transition weights	194
Figure III.4.2.4. DBN or expended BN with full of weights at time point t	195
Figure III.4.2.5. DBN or expended BN (at time point t) whose weights are normalized	197
Figure III.4.2.6. DBN at time point t with full of CPT (s).....	200
Figure III.4.2.7. The cycle of proposed algorithm to construct DBN	205
Figure III.5.2.1. Log-likelihood function with regard to variables a and b	218
Figure III.5.4.1. Bayesian network without CPT (s)	224
Figure III.5.4.2. Bayesian network with full of prior CPT (s)	227
Figure III.5.5.1. Bayesian network without CPT (s)	229
Figure III.5.5.2. Bayesian network with full of prior CPT (s)	232
Figure IV.2.2.1. Two dimensions in Riding model.....	238
Figure IV.2.4.1. Four-stage learning process of Kolb's model	239
Figure IV.2.4.2. Learning styles in Honey and Mumford model	240
Figure IV.4.1. HMM of weather forecast (hidden states are shaded)	246
Figure IV.4.1.1. Construction of recurrence formula for forward variable	252
Figure IV.4.1.2. Construction of recurrence formula for backward variable	255
Figure IV.4.2.1. Weighted arcs from null node X_0 to n nodes $X_{11}, X_{12}, \dots, X_{1n}$	268
Figure IV.4.2.2. Weighted arcs from n node $X_{(t-1)i}$ to n nodes X_{ij} at time point t	268
Figure IV.4.2.3. State transition graph	270
Figure IV.4.2.4. State transition graph of weather example	270
Figure IV.4.2.5. 2-weight interval.....	273
Figure IV.4.2.6. Longest path drawn as bold line inside state transition graph	275
Figure IV.4.3.1.1. Relationship between the log-likelihood function and its lower-bound	280
Figure IV.4.3.2.1. Construction of the joint probability $\xi_t(i, j)$	291
Figure IV.5.1. Graph of t-distribution parameter function	365
Figure IV.6.1. HMM of learning styles $\Delta^{(1)}$ (hidden states are shaded).....	369
Figure IV.6.2. HMM of learning styles $\Delta^{(2)}$ (hidden states are shaded).....	370
Figure IV.6.3. HMM of learning styles $\Delta^{(3)}$ (hidden states are shaded).....	370
Figure V.2.2.1.1. Separating hyperplanes.....	392
Figure V.2.2.1.2. Maximum-margin hyperplane, parallel hyperplanes and weight vector W	394
Figure V.2.2.1.3. Support vectors	400
Figure V.2.2.1.4. Linear constraint of two Lagrange multipliers	401
Figure V.2.2.1.5. Lower bound and upper bound of two new multipliers in case $s = 1$	405
Figure V.2.2.1.6. Lower bound and upper bound of two new multipliers in case $s = -1$	405
Figure V.2.2.1.7. Data points in training data (SVM)	410
Figure V.2.2.1.8. An example of maximum-margin hyperplane	416

Figure V.2.2.2.1. Decision tree constructed from nominal term frequencies at the first splitting	422
Figure V.2.2.2.2. Final decision tree constructed from nominal term frequencies ...	424
Figure V.2.2.3.1. Simplest topology of neural network with three layers such as input layer, hidden layer, and output layer	426
Figure V.2.2.3.2. Process of computing output of a unit.....	426
Figure V.2.2.3.3. The neural network for document classification.....	430
Figure V.2.2.3.4. Trained neural network.....	434
Figure V.3.1.1. A sample of six user vectors.....	441
Figure V.3.1.2. Two clusters resulted from k -means algorithm at the first running time.....	442
Figure V.3.1.3. Two clusters resulted from k -mean algorithm at the second running time.....	443
Figure V.3.2.1. Overlay model.....	444
Figure V.3.2.2. Graph model in form of tree and relationships	445
Figure V.3.2.1.1. Graph model and weighted arcs.....	447
Figure V.3.2.2.1. Bayesian network and its CPT (s).....	449
Figure V.3.3.1. Flow chart of k -medoid algorithm modified with similarity measure	454
Figure V.3.3.2. A sample of six user vectors.....	455
Figure V.3.3.3. Two clusters resulted from k -medoid algorithm at the first running time.....	456
Figure V.3.3.4. Two clusters resulted from k -medoid algorithm at the second running time	459
Figure VI.1.2.1. Decision-theoretic tree.....	465
Figure VI.1.3.1.1. Item Response Function curve	467
Figure VI.1.3.2.1. IRF function and density function of examinee's ability together with ability estimate (0.5) given discriminatory parameter $a=5$	476
Figure VI.1.3.3.1. An example of examinee's ability variance	483
Figure VI.2.1. Interaction between user modeling system and adaptive system.....	492
Figure VII.1.1. Triangular Learner Model	503
Figure VII.1.2. extended Triangular Learner Model	503
Figure VII.1.3. The architecture of Zebra	504
Figure VII.2.1. Ubiquitous user modeling	505
Figure VII.2.1.1.1. User modeling with integrated context-awareness	507
Figure VII.2.1.2.1. Spatial arrangement of ubiquitous computing similar fields according to two dimensions: user mobility and interface transparency	508
Figure VII.2.1.2.2. Situation model	509
Figure VII.2.1.2.3. Extended situation model	509
Figure VII.2.1.2.4. Situated interaction and situation modeling for ubiquitous computing	510
Figure VII.2.2.1. RDF triple	512
Figure VII.2.2.2. Onion model of statement	512
Figure VII.2.2.3. The schema of statement model in RDF graph notation	516
Figure VII.2.3.1.1. Situational statement represented by RDF triple	516
Figure VII.2.3.1.2. An example of statement represented by RDF triple	517
Figure VII.2.3.2.1. The correlation between UbisWorld and real world	518
Figure VII.2.3.2.2. UbisWorld and ontologies	519
Figure VII.2.4.1.1. Architecture of ubiquitous user model service	520
Figure VII.2.4.2.1. The work flow of ubiquitous user model service.....	522

Figure VII.2.5.1. Incorporating Zebra into ubiquitous service.....	523
Figure VII.2.5.2. Request-Response communication protocol between Zebra and ubiquitous service.....	524

C. List of Formulas

Formula I.3.1.2.1. Condition probability of cluster node	28
Formula I.3.1.2.2. Conditional probability of child node Y_i	28
Formula I.3.1.2.3. Conditional probability of child node Y_i given parent nodes X_1, \dots, X_N	29
Formula I.3.3.1. Posterior probability of student's mastery	39
Formula III.1.1.1. Bayes' rule.....	90
Formula III.1.1.2. Additional rule	91
Formula III.1.1.3. Multiplication rule	91
Formula III.1.1.4. Total probability rule	91
Formula III.1.1.5. Total probability rule in continuous case	91
Formula III.1.1.6. Definition of cumulative distribution function (CDF) and probability density function (PDF)	93
Formula III.1.1.7. Two conditions of joint probability distribution.....	94
Formula III.1.1.8. Reduced global joint probability distribution of random vector ..	94
Formula III.1.1.9. Posterior probability of variable X_i given evidence \mathcal{D}	94
Formula III.1.1.10. Advanced posterior probability of variable X_i given evidence \mathcal{D}	94
Formula III.1.1.11. Global joint probability distribution of wet grass Bayesian network	95
Formula III.1.1.12. Posterior probability of rain given wet grass evidence	95
Formula III.1.1.13. Posterior probability of sprinkler given wet grass evidence	95
Formula III.1.2.1. Conditional probability of $J=1$ given $C=1, O=1$, and $I=1$	100
Formula III.1.2.2. Conditional probability of E given O	100
Formula III.1.3.1. Sigma condition	104
Formula III.1.3.2. Probability of sigma sum.....	105
Formula III.1.3.3. Theorem of SIGMA-gate inference	107
Formula III.1.3.4. Theorem of SIGMA-gate inference with weighted graph	108
Formula III.1.3.5. Mastery distribution	108
Formula III.1.4.1.1. CPT of binary evidence of diagnostic relationship	112
Formula III.1.4.1.2. Transformation coefficient given uniform distribution of X ...	113
Formula III.1.4.1.3. CPT of integer evidence ranging in $\{0, 1, 2, \dots, \eta\}$ of diagnostic relationship.....	113
Formula III.1.4.1.4. CPT of general discrete evidence of diagnostic relationship ..	114
Formula III.1.4.1.5. CPT of continuous evidence of diagnostic relationship	114
Formula III.1.4.1.6. CPT of evidence of diagnostic relationship.....	116
Formula III.1.4.2.1. NOT-gate inference.....	117
Formula III.1.4.2.2. Probabilities of inhibitions I_i (s) and accountable variables A_i (s)	118
Formula III.1.4.2.3. Conditional probability of accountable variables	118
Formula III.1.4.2.4. X-gate probability	119
Formula III.1.4.2.5. Connection among arrangement sum and counter.....	121
Formula III.1.4.2.6. Sets K and L	122
Formula III.1.4.2.7. AND-gate condition	122
Formula III.1.4.2.8. AND-gate inference	122
Formula III.1.4.2.9. OR-gate condition	123
Formula III.1.4.2.10. OR-gate inference	123
Formula III.1.4.2.11. NAND-gate inference and NOR-gate inference	124
Formula III.1.4.2.12. Two XOR-gate conditions.....	124

Formula III.1.4.2.13. XOR-gate inference	126
Formula III.1.4.2.14. Two XNOR-gate conditions	127
Formula III.1.4.2.15. XNOR-gate inference.....	128
Formula III.1.4.2.16. U-gate conditions	128
Formula III.1.4.2.17. U-gate inference.....	130
Formula III.1.4.2.18. SIGMA-gate condition	131
Formula III.1.4.2.19. SIGMA-gate inference	132
Formula III.1.4.2.20. Conditional probability of accountable variables with regard to bi-weight graph	132
Formula III.1.4.2.21. Total clockwise and counterclockwise weights.....	133
Formula III.1.4.2.22. SIGMA-gate bi-inference	134
Formula III.1.4.3.1. Joint probability of X-D network.....	136
Formula III.1.4.3.2. Conditional probability $P(D X_i)$ and posterior probability $P(X_i D)$	136
Formula III.1.4.3.3. Joint probability $P(X_i, D)$ and marginal probability $P(D)$ given uniform distribution of all sources	138
Formula III.1.4.3.4. Conditional probability, posterior probability, and transformation coefficient of X-D network	138
Formula III.1.4.3.5. Conditional probability and posterior probability of AND-D network	140
Formula III.1.4.3.6. Conditional probability, posterior probability, and transformation coefficient of SIGMA-D network.....	141
Formula III.1.4.3.7. CPT of direct SIGMA-D network.....	142
Formula III.1.4.3.8. Joint probability of direct SIGMA-D network	142
Formula III.1.4.3.9. Joint probability $P(X_i, D)$ and marginal probability $P(D)$ of direct SIGMA-D network	143
Formula III.1.4.3.10. Singular GL-gate inference	145
Formula III.1.4.3.11. Diagnostic condition equation of binary M-E-D network....	146
Formula III.2.1. Attribute $J.\text{do\$bayes\$infer}$ represents posterior probability	154
Formula III.3.1.1. Beta density function	158
Formula III.3.1.2. Gamma function	159
Formula III.3.1.3. Important property of gamma function with regard to factorial function.....	159
Formula III.3.1.4. Integral of product expression $x^a(1-x)^b$	160
Formula III.3.1.5. Conditional probability (relative frequency) of X as value of parameter variable F	160
Formula III.3.1.6. Probability of hypothesis X as expectation of beta variable F ...	161
Formula III.3.1.7. Expectation of expression $F^t(1-F)^{1-t}$	162
Formula III.3.1.8. Probability $P(\mathcal{D})$ of evidences \mathcal{D}	162
Formula III.3.1.9. Posterior beta density function	163
Formula III.3.1.10. Posterior probability of X	163
Formula III.3.1.11. Conditional probability (relative frequency) of X_i given a parent instance PA_{ij} , as value of parameter variable in multi-node BN.....	165
Formula III.3.1.12. Beta density function $\beta(F_{ij})$ corresponding to an instance of a parent of node X_i	165
Formula III.3.1.13. Joint beta density function of variable X_i having c_i parent instances.....	165
Formula III.3.1.14. Global joint beta density function of n independent variable X_i (s).....	165

Formula III.3.1.15. Probability of variable X_i with respect to its parent instance as expectation of beta variable	166
Formula III.3.1.16. Probability of evidences corresponding to variable X_i	167
Formula III.3.1.17. Probability of evidence sample \mathcal{D} given vectors F_i	168
Formula III.3.1.18. Whole probability of evidence sample \mathcal{D}	169
Formula III.3.1.19. Expectation of binomial trials.....	170
Formula III.3.1.20. Posterior beta density function in multi-node BN	171
Formula III.3.1.21. Posterior probability of variable X_i given its parent instance PA_{ij}	171
Formula III.3.3.1. Definition of equivalent sample size N	180
Formula III.3.3.2. Theorem of equivalent sample size N	180
Formula III.4.1.1. Distributed global joint probability distribution of DBN.....	188
Formula III.4.2.1. Transition probability distribution	190
Formula III.4.2.2. Markov property according to transition probability distribution	190
Formula III.4.2.3. Formula of slip factor and guess factor.....	193
Formula III.4.2.4. Conditional probability a	193
Formula III.4.2.5. The bias b of user knowledge	193
Formula III.4.2.6. The weight w as product of conditional probability a and bias b	193
Formula III.4.2.7. The weight w implicates that the conditional transition probability satisfies both Markov property and stationary property.....	194
Formula III.4.2.8. The set Y of all variables (nodes) of expended BN $G'[t]$ at time point t	194
Formula III.4.2.9. Normalizing weights w_2, w_3	195
Formula III.4.2.10. Normalizing weights in general case	196
Formula III.4.2.11. Posterior probability of each $x_i[t-1]$ given evidences $\mathcal{D}[t-1]$...197	197
Formula III.4.2.12. Conditional probability of each variable $x_i[t]$ at current time point t	198
Formula III.4.2.13. Posterior probability of each $x_i[t]$ given evidences $\mathcal{D}[t]$	200
Formula III.4.2.14. An example of DBN global joint probability	201
Formula III.5.1. Beta density function $\beta(F; a, b)$	207
Formula III.5.2. Probability of variable X as expectation of beta variable F	207
Formula III.5.3. Posterior probability of variable X as conditional expectation of beta variable F	208
Formula III.5.1.1. Likelihood function.....	208
Formula III.5.1.2. Optimal parameter vector	209
Formula III.5.1.3. Log-likelihood function and optimal parameter vector	209
Formula III.5.1.4. Co-variance matrix of parameter vector.....	210
Formula III.5.2.1. Beta density function (beta distribution) $\beta(F; a, b)$	210
Formula III.5.2.2. Definition of digamma function	210
Formula III.5.2.3. Integral form of digamma function.....	211
Formula III.5.2.4. Recurrence formula of digamma function.....	211
Formula III.5.2.5. Digamma function in case of positive integer variable	213
Formula III.5.2.6. Trigamma function.....	214
Formula III.5.2.7. Recurrence formula of trigamma function	215
Formula III.5.2.8. Trigamma function in case of positive integer variable.....	216
Formula III.5.2.9. Beta function $B(x, y)$	216
Formula III.5.2.10. First-order partial derivative of beta function $B(x, y)$	217
Formula III.5.2.11. Beta density function with regard to beta function	217

Formula III.5.2.12. Log-likelihood function of beta density function (beta distribution).....	217
Formula III.5.2.13. First-order partial derivative of log-likelihood function of beta density function with regard to parameter a	218
Formula III.5.2.14. First-order partial derivative of log-likelihood function of beta density function with regard to parameter b	218
Formula III.5.2.15. The set of differential equations for estimating parameters a and b	219
Formula III.5.2.16. Co-variance matrix of parameters of beta density function	220
Formula III.5.2.17. Standard errors of parameter estimates of beta distribution	220
Formula III.5.3.1. Two simplest equations for estimating positive integer parameters a and b	222
Formula III.5.5.1. New version of equations for estimating positive integer parameters a and b	229
Formula IV.4.1.1. Forward variable	251
Formula IV.4.1.2. Recurrence property of forward variable	252
Formula IV.4.1.3. Probability $P(O/\Delta)$ based on forward variable	253
Formula IV.4.1.4. Backward variable.....	254
Formula IV.4.1.5. Recurrence property of backward variable.....	255
Formula IV.4.1.6. Probability $P(O/\Delta)$ based on backward variable	256
Formula IV.4.2.1. Joint probability of being in state s_i at time point t with observation sequence O	258
Formula IV.4.2.2. Optimal state at time point t	259
Formula IV.4.2.3. Joint optimal criterion at time point t	261
Formula IV.4.2.4. Recurrence property of joint optimal criterion	262
Formula IV.4.2.5. Backtracking state	263
Formula IV.4.2.6. Optimal criterion of longest-path algorithm	267
Formula IV.4.2.7. Weights at the first time point.....	267
Formula IV.4.2.8. Weights at the time point t	268
Formula IV.4.3.1.1. EM optimization criterion based on conditional expectation ..	278
Formula IV.4.3.1.2. Lower-bound of log-likelihood function	279
Formula IV.4.3.2.1. General EM conditional expectation for HMM	282
Formula IV.4.3.2.2. EM conditional expectation for HMM	283
Formula IV.4.3.2.3. Lagrangian function for HMM	283
Formula IV.4.3.2.4. HMM parameter estimate	290
Formula IV.4.3.2.5. Joint probability $\xi_t(i, j)$	291
Formula IV.4.3.2.6. The γ_t is sum of ξ_t over all states	291
Formula IV.4.3.2.7. HMM parameter estimate in detailed	292
Formula IV.5.1. Probability density function (PDF) of observation	310
Formula IV.5.2. Joint probabilities $\xi_t(i, j)$ and $\gamma_t(j)$ based on single PDF	313
Formula IV.5.3. EM conditional expectation for continuous observation HMM with single PDF	313
Formula IV.5.4. Lagrangian function for continuous observation HMM with single PDF.....	314
Formula IV.5.5. Equation of single PDF parameter	315
Formula IV.5.6. Continuous observation HMM parameter estimate with single PDF	317
Formula IV.5.7. Mixture model probability density function (PDF) of observation	330

Formula IV.5.8. Partial joint probabilities $\xi_t(i, j, k)$ and $\gamma_t(j, k)$ based on mixture model PDF	331
Formula IV.5.9. Relationship between quantities $\xi_t(i, j)$, $\gamma_t(j)$ and partial quantities $\xi_t(i, j, k)$, $\gamma_t(j, k)$	331
Formula IV.5.10. EM conditional expectation for continuous observation HMM given mixture model PDF.....	333
Formula IV.5.11. Lagrangian function for continuous observation HMM with single PDF.....	333
Formula IV.5.12. Weight estimate of partial PDF	335
Formula IV.5.13. Equation of partial PDF parameter	337
Formula IV.5.14. Continuous observation HMM parameter estimate with mixture PDF.....	338
Formula IV.5.15. Exponential family PDF	361
Formula IV.5.16. Log-partition function $A(\eta)$	361
Formula IV.5.17. First derivative of log-partition function $A(\eta)$ with regard to η ...	361
Formula IV.5.18. Log-likelihood function of exponential family PDF.....	362
Formula IV.5.19. Derivative of log-likelihood function of exponential family PDF	362
Formula IV.5.20. Equation of exponential family PDF parameters	362
Formula IV.5.21. Student's t-distribution	364
Formula IV.5.22. Functions gamma, digamma, trigamma	364
Formula IV.5.23. Equation of t-distribution PDF parameter	365
Formula V.2.1.1. Term frequency	390
Formula V.2.1.2. Inverse document frequency	390
Formula V.2.1.3. The weight w_{ij} of term t_j in document D_i	391
Formula V.2.1.4. Vector model of document D_i	391
Formula V.2.1.5. Document vector as a set of term frequencies.....	391
Formula V.2.2.1.1. Hyperplane equation	392
Formula V.2.2.1.2. Equations of two parallel hyperplanes	393
Formula V.2.2.1.3. Classification constraint	394
Formula V.2.2.1.4. Constrained optimization problem for constructing maximum-margin hyperplane	394
Formula V.2.2.1.5. General SVM constrained optimization problem	395
Formula V.2.2.1.6. Lagrangian function	396
Formula V.2.2.1.7. Lagrangian duality problem	396
Formula V.2.2.1.8. Formula for determining optimal weight vector W^*	397
Formula V.2.2.1.9. Dual function for determining Lagrange multipliers	398
Formula V.2.2.1.10. Quadratic programming of support vector machine	398
Formula V.2.2.1.11. KKT condition of SVM	399
Formula V.2.2.1.12. KKT corollaries of SVM.....	399
Formula V.2.2.1.13. Two cases of KKT condition violation	400
Formula V.2.2.1.14. Linear constraint of two Lagrange multipliers	401
Formula V.2.2.1.15. A variant of linear constraint of two Lagrange multipliers	402
Formula V.2.2.1.16. Dual function with regard to λ_2 that is optimized in conjunction with λ_1	404
Formula V.2.2.1.17. Optimal classifier (W^*, b^*) resulted from each optimization step of SMO algorithm	407
Formula V.2.2.1.18. Equation of maximum-margin hyperplane (SVM classifier) ..	408
Formula V.2.2.1.19. Classification rule derived from maximum-margin hyperplane (SVM classifier)	408

Formula V.2.2.2.1. Entropy of training corpus	419
Formula V.2.2.2.2. Entropy of given attribute inside training corpus	419
Formula V.2.2.2.3. Information gain of given attribute inside training corpus	419
Formula V.2.2.3.1. Formula for computing the output of a unit	427
Formula V.2.2.3.2. Error of output unit	429
Formula V.2.2.3.3. Error of hidden unit.....	429
Formula V.2.2.3.4. Updating connection weight.....	429
Formula V.2.2.3.5. Updating bias.....	430
Formula V.3.1.1. Dissimilarity of two user model vectors according to Euclidean distance	440
Formula V.3.1.2. Error criterion for k -mean algorithms	440
Formula V.3.1.3. The mean of a cluster.....	440
Formula V.3.2.1. Dissimilarity of two graph models	445
Formula V.3.2.1.1. Dissimilarity of two weighted graph models	447
Formula V.3.2.1.2. Dissimilarity of two weighted graph models without node values	447
Formula V.3.2.2.1. Dissimilarity of two Bayesian networks	449
Formula V.3.3.1. Cosine similarity measure	451
Formula V.3.3.2. Correlation coefficient	452
Formula V.3.3.3. Average similarity	454
Formula V.3.3.4. Global average similarity	454
Formula VI.1.2.1. Expected utility of an action	466
Formula VI.1.3.1.1. Item Response Function	467
Formula VI.1.3.1.2. Information function for item i	469
Formula VI.1.3.2.1. IRF with zero guessing parameter.....	470
Formula VI.1.3.2.2. Examinee's initial ability.....	470
Formula VI.1.3.2.3. Likelihood function of examinee's ability.....	470
Formula VI.1.3.2.4. Log-likelihood function of examinee's ability	471
Formula VI.1.3.2.5. Equation for finding out parameter estimates	472
Formula VI.1.3.2.6. Discriminatory and difficult estimates	473
Formula VI.1.3.2.7. Probability density function of examinee's ability	473
Formula VI.1.3.2.8. Optimal probability density function of examinee's ability	473
Formula VI.1.3.2.9. Examinee's ability estimate	475
Formula VI.1.3.3.1. Examinee's ability variance	476
Formula VI.1.3.3.2. Dilogarithm function	478
Formula VI.1.3.3.3. Expectation of square of examinee's ability.....	478
Formula VI.1.3.3.4. Inversion property of dilogarithm	481
Formula VI.1.3.3.5. Examinee's explicit ability variance	482
Formula VI.1.3.3.6. Examinee's statistical ability mean	483
Formula VI.1.3.3.7. Examinee's ability variance as statistical sample variance	483
Formula VI.1.3.3.8. Discriminatory estimate.....	484
Formula VI.1.3.3.9. Ability error used as stopping criterion of CAT algorithm	486
Formula VI.2.1.1.1. Sample means	494
Formula VI.2.1.1.2. Sample variances.....	494
Formula VI.2.1.1.3. Sample standard deviations	494
Formula VI.2.1.1.4. System criterion α determined based on F-distribution test	495
Formula VI.2.1.1.5. Linear regression function of user's total knowledge	495
Formula VI.2.1.1.6. System criterion α determined based errors of linear regression model	496
Formula VI.2.1.2.1. Academic criterion β based cumulative function.....	496

Formula VI.2.1.3.1. Simple adaptation criterion γ based the number of satisfied users	497
Formula VI.2.1.3.2. Adaptation criterion γ as module of mean vector	498

D. Index

- 2-weight interval, 272
 a , 159
ability. *See* examinee's ability
ability error, 486
ability estimate, 473
ability variance, 476
absolute humidity, 318
Abstract Conceptualization, 239
AC. *See* adaptive component
academic criterion, 493
access attribute, 150
Accommodating, 239
accountability, 117
accountable variables, 117
activation function, 426
Active, 240
Active Experimentation, 239
Activist, 240
adaptation criterion, 493
adaptation model, 18
adaptation process, 57
adaptation rules, 150
adaptive, 9
adaptive component, 20
Adaptive Education System, 18
adaptive educational hypermedia system, 15
Adaptive Educational Intelligent Web-Based System, 22
Adaptive Educational Web-Based System, 22
adaptive hypermedia applications, 48
Adaptive Hypermedia for All, 20
Adaptive Hypermedia System, 11
adaptive learning, 1
adaptive learning model, 493
adaptive learning system, 1
adaptive learning web page, 78
adaptive link hiding, 16
adaptive link sorting, 16
adaptive navigation, 15
adaptive presentation, 15
adaptive procedure, 22
adaptive process, 22
adaptive systems, 9
additional rule, 91
administered item, 469
administration box, 514
administration process, 469
advanced CAT algorithm, 476
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, 47
aggregation, 97
aggregation inhibition, 104
aggregation relationship, 97
aggregative node, 103
AHA!. *See* Adaptive Hypermedia for All
AHA! engine, 21
AHAM, 20
AHS. *See* Adaptive Hypermedia System
 a_i , 468
 a_{ij} , 165
Ajax, 85
ALS. *See* adaptive learning system
alternate approach, 464
alternative hypothesis, 494
AM. *See* adaptation model
Analytic, 237
Analytical, 237
ANATOM-TUTOR, 14
AND-D network, 140
Andes, 29
AND-gate inference, 122
ANN. *See* artificial neural network
Application box, 521
Application-directed, 241
AprioriAll, 380
aptitudes, 4
aptitude-treatment interactions system, 10
arc, 92
arrangement sum, 120
artificial intelligence, 5
artificial neural network, 425
assessment of Bayesian network, 464
Assimilating, 239
ATI. *See* aptitude-treatment interactions system
attribute-value pairs, 150
Auditory, 237
augmented Bayesian network (augmented BN), 160
automatic stereotype management, 46
auxiliary, 516
average similarity, 453
 b , 159
background, 4
back-propagation algorithm, 428
backtracking quantity, 263
backward reasoning, 151
backward variable, 254
basic user dimensions, 517
Baum Welch algorithm, 189
Bayes' rule, 90
Bayes' theorem, 90
Bayesian inference, 90
Bayesian model, 98
Bayesian network, 92
Bayesian overlay model, 98
belief network engine, 55
Belief, Goal and Plan Maintenance System, 45
best_grade, 28
beta density function, 158
beta distribution. *See* beta density function
Beta likelihood estimation, 210
 $\beta(x; a, b)$. *See* beta density function
BGP-MS. *See* Belief, Goal and Plan Maintenance System
 b_i , 468
bias, 392
bi-inferences, 133
 b_{ij} , 165
binary arrangements, 119
binary random variable, 98
binomial augmented Bayesian network (binomial augmented BN), 162
binomial sample, 162
binomial trials, 167
bi-weight simple graph, 132
BN. *See* Bayesian network
BNE. *See* belief network engine
breaking sequential pattern, 386
CA. *See* condition-action
CAI. *See* Computer Assisted Instructional
candidate generation-and-test approach, 380
canned text adaptation, 16
CAT. *See* Computerized Adaptive Testing
cause inhibition, 104
cause-effect, 97
CBM. *See* constraint-based modeling
CDF. *See* cumulative distribution function
characteristics, 9
chronologic order, 193
 c_i , 468
CI. *See* communication interface
c-id, 150
c-info. *See* concept information

Appendix D. Index

- class attribute, 418
classification constraint, 394
classification rule, 408
classifier, 389
clockwise adder, 133
clockwise weight, 132
cluster node, 26
clustering approach, 26
cognitive structure, 236
collaborative area, 84
collaborative learning, 439
communication interface, 55
communication sub-component, 48
community adaptation, 439
Computer Assisted Instructional, 11
Computerized Adaptive Testing, 468
concept, 6
concept information, 150
concept selection process, 57
concept selection rules, 56
concept tier, 22
Concrete Experience, 239
condition-action, 151
conditional approach, 26
conditional dependence, 97
conditional independence, 95
conditional mutual information, 40
conditional probability, 90
conditional probability distribution, 92
conditional probability table, 92
conditional text, 16
confidence, 387
Conflict Resolution, 520
connection (neural network connection), 426
connection weight, 429
constitutionally based learning styles and preferences, 236
constrained optimization problem, 394
constraint-based approach, 189
constraint-based modeling, 36
content selection process, 57
content selection rules, 56
context model, 508
context-awareness, 506
context-rule nodes, 33
continuous observation HMM, 310
Converging, 239
corpus, 90
correct knowledge, 8
correct response, 470
correlation coefficient, 452
cosine similarity measure, 451
counter, 168
counterclockwise adder, 133
counterclockwise weight, 132
co-variance matrix, 209
CPD. *See* conditional probability distribution
CPT. *See* conditional probability table
Cr. *See* relevance condition
Cramer-Rao lower bound, 210
Cs. *See* satisfaction condition
CUMULATE, 47
cumulative distribution function, 92
cumulative function. *See* cumulative distribution function
daemon, 59
DAG. *See* directed acyclic graph
data mining, 5
data point, 392
data sample, 90
data-centric method, 23
DBN. *See* dynamic Bayesian network
decision base, 14
decision tree, 418
decision-theoretic approach, 465
default stereotype, 5
degrees of freedom, 495
demographic, 5
density function. *See* probability density function
dependency, 97
dependency graph, 24
 $depth(v_{ij})$, 445
Dexter Hypertext Reference Model, 20
DGJPD. *See* Distributed Global Joint Probability Distribution
diagnosis, 13
diagnostic, 97
diagnostic approach, 464
diagnostic condition, 112
diagnostic relationship, 97
diagnostic theorem, 139
didactics, 13
didactics module. *See* pedagogical module
differential model, 8
difficult estimate, 473
difficult parameter, 468
digamma function, 210
dilogarithm function, 478
direct guidance, 16
direct SIGMA-D network, 141
directed acyclic graph, 92
directory component, 48
discriminatory estimate, 484
discriminatory item, 484
discriminatory parameter, 468
dissim. *See* dissimilarity
dissimilarity measure, 439
distance, 439
distance learning, 1
Distributed Global Joint Probability Distribution, 188
Distributed ontologies box, 521
Distributed Services box, 520
Distributed Statements box, 521
distribution. *See* probability distribution
Diverging, 240
do\$bayes\$infer, 153
DOCS, 19
document classification, 391
document vector, 391
domain, 7
domain expert, 12
domain independence information, 4
domain knowledge, 7
domain model, 6
domain specific information, 3
domain-general part, 33
Doppelgänger, 46
dot product, 392
downward inference direction, 507
drafting phase, 41
d-separation, 40
dual function, 397
dummy attribute, 152
Dunn and Dunn model, 236
dynamic Bayesian network, 188
ECA. *See* event-condition-action
edge, 445
efficiency-centric method, 23
eigenvectors, 497
e-learning, 1
element, 378
EM. *See* Expectation Maximization
EM algorithm, 173
EM conditional expectation, 279
entropy, 418
equivalent sample size, 180
error criterion, 440
error of hidden unit, 429
error of output unit, 429
E-step. *See* Expectation step
estimated knowledge, 495
Euler's number, 159

- Euler-Mascheroni constant, 212
 evaluation act, 498
 evaluation criteria, 493
 evaluation of Bayesian network, 462
 evaluation problem, 246
 evaluation scenario, 498
 event, 91
 event storage, 47
 event-condition-action, 151
 event-driven approach, 47
 evidence, 90
 evidence matrix, 167
 evidence nodes, 98
 evidence sample, 90
 evidence variables, 97
 evidence vector, 161
 evolution of Bayesian network. *See* evolution of Bayesian overlay model
 evolution of Bayesian overlay model, 157
 evolution of CPT, 157
 evolution of parameters, 158
 examinee's ability, 467
 exception independence, 104
 exclusive aggregation, 103
 exclusive union, 103
 Expectation Maximization, 173
 Expectation step, 173
 expected knowledge, 7
 expected utility, 465
 expended Bayesian network (expended BN), 194
 experience, 4
 expert-centric method, 23
 explanation box, 513
 explanation problem, 246
 explanations, 95
 explicit ability variance, 482
 extended functions, 377
 extended X-gate network, 117
 external inference agents, 47
Extravert, 238
 f_{list} , 386
 fact nodes, 34
 fault model, 3
 FD. *See* Field-dependence
 F-distribution, 495
 feedback act, 498
 feedback report, 75
 feedback tool, 83
 feedbacks, 75
 feed-forward neural network, 427
Feeler, 238
 Felder-Silverman model, 240
 F_i , 168
 FI. *See* Field-independency
Field-dependence, 237
Field-independency, 237
 F_{ij} , 165
 first-order logic, 19
 fix stereotype, 5
 flexibly learning preferences, 236
 FOL. *See* first-order logic
 format attribute, 367
 forward reasoning, 151
 forward variable, 251
 forward-backward procedure, 253
 frame, 511
 FrameNet, 511
 FreeSpan. *See* Frequent pattern-projected Sequential pattern
 frequent item matrix, 386
 frequent itemset, 380
 Frequent pattern-projected Sequential pattern, 385
 frequent sequence, 378
 functional interface, 46
 gamma function, 158
 general EM algorithm, 278
 General User Model Ontology, 511
 General User Modeling System, 43
 Generalized Sequential Pattern, 384
 generating candidate set, 381
 generic scrutiny tools, 48
 GJPD. *See* Global Joint Probability Distribution
 GL-gate inference, 145
Global, 237
 global average similarity, 454
 Global Joint Probability Distribution, 94
 global parameter independence, 165
 global teaching knowledge, 15
 goal nodes, 34
 goals, 4
 gradient, 396
 graph model, 7
 group adaptation, 439
 GRUNDY, 43
 GSP. *See* Generalized Sequential Pattern
 guess. *See* lucky guess
 guessing parameter, 468
 GUMO. *See* General User Model Ontology
 GUMS. *See* General User Modeling System
 Hessian matrix, 219
 hidden layer, 425
 hidden Markov model, 244
 hidden nodes, 98
 hidden state, 244
 hidden unit, 425
 hidden variables, 97
 HMM. *See* hidden Markov model
 HMM Lagrangian function, 283
 HMM parameter estimate, 289
 Honey and Mumford model, 240
 horizontal data format, 380
 HTML, 62
 human-machine interface, 2
 hypermedia, 15
 hyperplane, 392
 hyperplane equation, 392
 hypothesis, 90
 hypothesis test, 494
 ICC. *See* Item Characteristic Curve
 idf. *See* inverse document frequency
 $I_i(\theta)$. *See* information function
Imager, 238
 imperfect separation, 395
Impulsive, 237
 incomplete sample, 182
 incorrect knowledge, 8
 indicator mastered, 70
 individual adaptation, 439
 individual user model, 45
 individually optimal criterion, 258
 individually optimal procedure, 259
 inference (stereotype), 6
 inference agents, 47
 Inference Engine, 520
 inferred user model, 47
 information function, 469
 information gain, 419
 information matrix, 209
 informative item, 485
 inhibition independence, 104
 initial ability, 470
 initial Bayesian network (initial BN), 188
 initial state distribution, 244
 inner loop, 407
 input layer, 425
 input unit, 425
 instructional meta-strategy, 243
 instructional strategy, 243
 intelligent tutoring system, 11
 interactive attribute, 367

Appendix D. Index

- interesting document, 389
interesting document vector, 437
interests. *See* user interests
Interface Manager, 520
interface transparency, 508
Interfaces & Exchange box, 521
Internal Kinesthetic, 237
intersection operator, 91
Introvert, 238
Intuitive, 238
inverse document frequency, 390
inversion property, 481
IRF. *See* Item Response Function
IRT. *See* Item Response Theory
item, 378
Item Characteristic Curve, 467
item pool, 468
Item Response Function, 467
Item Response Theory, 467
itemset, 378
ITS. *See* intelligent tutoring system
Java, 5
Java course, 62
Java tutorial, 62
joint optimal criterion, 261
joint probability, 91
Judger, 238
K. *See* knowledge sub-model
Karush-Kuhn-Tucker multiplier, 396
KBS hyperbook system, 23
k-class classification, 408
KI. *See* knowledge item
KKT condition violation, 400
k-mean algorithm, 440
k-medoid algorithm, 453
KN-IPCMS. *See* KoNstanz Inter-Process Communication Management System, 46
Lagrange multiplier, 396
Lagrangian duality, 396
Lagrangian function, 395
large itemset. *See* frequent itemset
large itemset phase, 380
large margin training, 310
large sequence. *See* frequent sequence
LDAP-UMS, 48
leaf nodes, 418
leaky-OR, 34
learner model, 1
learner modeling, 1
learners, 2
learning concept recommendation, 67
learning context, 9
learning history, 52
learning history sub-model, 51
learning management systems, 1
learning materials, 1
learning object, 7
learning path recommendation, 66
learning problem, 246
learning process, 2
learning rate, 429
learning report, 73
learning resources, 18
learning sequence, 379
learning sequence database, 377
learning style sub-model, 51
learning styles, 4
learning-by-doing, 12
left-hand itemset, 387
LH. *See* learning history sub-model
likelihood function, 90
linear regression function, 495
link annotation, 16
link generation, 16
itemset. *See* frequent itemset
local parameter independence, 165
local teaching knowledge, 15
log tier, 22
logical predicates, 19
logistic function, 427
log-likelihood function, 209
longest-path algorithm, 265
lower-bound, 279
LS. *See* learning history sub-model
l-sequence, 378
lucky guess, 187
machine learning, 5
macro-adaptive system, 10
mailing list tool, 74
main part box, 512
mal-knowledge, 8
map adaptation, 17
margin, 393
marginal probability, 90
Markov condition. *See* Markov property
Markov model, 244
Markov property, 190
mastered, 7
mastered_c, 38
mastery, 7
maximal frequent sequence, 378
maximal phase, 383
Maximization step, 173
maximum frequent itemset, 389
Maximum Likelihood Estimation, 208
maximum margin, 392
maximum-margin hyperplane, 392
ME. *See* mining engine
Meaning-oriented, 241
M-E-D network, 146
media space, 18
medoid, 453
M-HE-D network, 148
micro-adaptive system, 10
midpoint, 272
min_conf, 387
min_sup, 378
minimal infrequent sequence, 379
mining engine, 54
missing data, 173
missing values, 173
mixture continuous observation HMM, 330
mixture model, 330
MLE. *See* Maximum Likelihood Estimation
MM. *See* Markov model
M-step. *See* Maximization step
multi-evidence diagnostic relationship, 145
multi-hypothesis diagnostic relationship, 134
multi-hypothesis multi-evidence diagnostic relationship, 148
multi-node Bayesian network (multi-node BN), 164
multiplication rule, 91
mutual information, 40
mutually exclusive, 91
mutually independent, 91
Myers-Briggs Type Indicator, 238
NAND-gate inference, 124
neural network. *See* artificial neural network
NN. *See* neural network
node, 92

- noisy OR-gate, 108
 noisy-AND, 34
 nominal interesting document, 437
 nominal term frequency, 418
 non-boundary, 400
 non-class attribute, 418
 non-evidence variable, 189
 non-leaf nodes, 418
 NOR-gate inference, 124
 normal bias, 223
 not mastered, 7
 NOT-D network, 136
 null hypothesis, 494
 OBS, 19
 observable stochastic process, 244
 observation, 244
 observation probability matrix, 245
 observation probability parameters, 319
 observation sequence, 244
 observer, 55
 occurrences, 174
 OLAE, 29
 onion model, 512
 online learning, 1
 ontology, 7
 Ontology Reasoning, 520
 Ontology Web Language, 511
 optimal bias, 396
 optimal criterion, 258
 optimal parameter. *See* parameter estimate
 optimal parameter vector, 209
 optimal weight vector, 396
 optimized jointly, 399
 optional response, 469
 ordinary weight, 192
 OR-gate inference, 122
 outcome, 161
 outer loop, 407
 output layer, 425
 output unit, 425
 overlay model, 6
 overlay model clustering, 444
 OWL. *See* Ontology Web Language
 PA_i , 168
 PA_{ij} . *See* parent instance
 parallel hyperplanes, 393
 parameter estimate, 209
 parameter estimator. *See* parameter estimate
 parameter evolution, 158
 parameter learning, 94
 parameter vector estimate, 209
 parameter vector estimator, 209
 parent instance, 164
 parent-child, 97
 partial diagnostic condition, 147
 partial node, 103
 Pask model, 240
 Pattern-growth approach, 385
 PDF. *See* probability density function
 pedagogical module, 12
 penalty, 395
Perceiver, 238
 percentage point, 495
 perception group, 241
 perfect separation, 395
 $performance_{cp}$, 39
 personal traits, 4
 Personis, 47
 perturbation model, 8
 physics rules, 30
 piecewise-linear function, 427
 plan model, 8
 plan recognition, 8
 plans of action, 13
 Poisson distribution, 315
 portal, 85
 portlet, 85
 posterior beta density function, 163
 posterior density function, 163
 posterior probability, 90
Pragmatist, 240
 predicate, 516
 prediction error, 399
 prerequisite, 97
 prerequisite relationship, 23
 presentation specification, 150
 presenter, 57
 prior density function, 161
 prior probability, 90
 privacy box, 513
 probabilistic inference, 191
 probability density function, 92
 probability distribution, 92
 processing unit, 425
 profile mining, 53
 profile tier, 22
 projected database, 386
 projected itemset, 385
 PROlog based Tool for User Modeling, 44
 proposition nodes, 34
 PROTUM. *See* PROlog based Tool for User Modeling
 q_i . *See* optional responses
 QP. *See* quadratic programming
 quadratic programming, 398
 qualitative model, 206
 quantitative model, 206
 random event, 91
 random probability, 90
 random variable, 92
 range, 516
 rating matrix, 497
 RDF. *See* Resource Description Framework
 RDF triple, 516
 real knowledge, 495
 reasoning, 89
 recurrence relation, 211
 recurrent neural network, 427
Reflective, 237
Reflective Observation, 239
Reflector, 240
 regression coefficients, 495
 reinforcement learning, 428
 relationship powers, 133
 relationships, 7
 relative humidity, 318
 relevance condition, 36
 $relevantIS_{cp}$, 39
 $relevantSS_{cp}$, 39
 representation sub-component, 49
 representation system, 45
Reproduction-oriented, 241
 Resource Description Framework, 510
 resource model, 56
 retraction, 6
 Retrieval Filter, 520
 r_i . *See* correct responses
 Riding model, 237
 right-hand itemset, 387
 rule-application nodes, 34
 runtime layer, 17
 s , 162
 sample, 90
 sample mean, 494
 sample variance, 494
 satisfaction condition, 36
 satisfaction criterion, 493
 SB-ONE, 45
 scalar product. *See* dot product
 scheduler sub-component, 49
 scored-based approach, 189

Appendix D. Index

- search engine, 82
searching history, 436
SEM. *See* Structural Expectation Maximization
semantic web, 510
Sensing, 240
Sensor, 238
sensor level, 46
separating hyperplane, 391
sequence, 378
sequence of anchors, 150
sequence phase, 381
Sequential, 240
Sequential Minimal Optimization, 398
sequential pattern, 377
Sequential PAttern Discovery using Equivalent classes, 384
sequential pattern mining, 377
sequential rule, 387
Serialist, 241
server level, 46
sigma condition, 104
sigma graph, 97
sigma sum, 99
SIGMA-D network, 141
SIGMA-gate bi-inference, 134
SIGMA-gate inference, 99
sigmoid function, 427
significant level, 495
 s_{ij} , 168
similarity measure, 451
situated interaction, 510
situation, 508
Situation Adder, 520
situation box, 513
situation model, 508
situation modeling, 509
situation semantics, 508
Situation Server, 520
situational statement, 511
SituationML, 521
SituationQL, 521
slip. *See* temporary slip
smallest optimization problem, 398
SMO. *See* Sequential Minimal Optimization
solution graph, 29
sorting fragments, 16
source node, 103
SPADE. *See* Sequential PAttern Discovery using Equivalent classes
specifying prior probabilities, 206
SQL-Tutor, 37
squashing function, 427
stable personality type, 236
standard deviation, 220
standard error, 220
standard normal distribution, 496
state, 244
state sequence, 244
state stochastic process, 244
state transition graph, 269
statement, 512
statement model, 512
stationary, 191
statistical ability mean, 483
statistics, 5
stereotype, 5
stochastic approach, 26
stochastic process, 244
stopping criterion (CAT), 469
storage layer, 17
strategic contexts, 13
strategy nodes, 34
stretch text, 16
Structural Expectation Maximization, 189
structure learning, 94
student model, 2
students, 2
study act, 498
sub-sequence, 378
sufficient diagnostic proposition, 112
sufficient evidence, 112
summing function, 426
super-sequence, 378
supervised learning, 428
support, 378
support threshold, 378
support vector machine, 391
support vectors, 393
SVM. *See* support vector machine
system criterion, 493
 t , 162
Tactile Kinesthetic, 237
TAGUS, 44
target function, 394
target level, 14
target node, 103
task-specific part, 33
temporal dependency, 196
temporal random vector, 188
temporal relationship, 188
temporal variable, 190
temporal weight. *See* transition weight
temporary slip, 187
term, 390
term frequency, 390
tf. *See* term frequency
the first choice heuristic, 398
the first Lagrange multiplier, 398
the second choice heuristic, 398
the second Lagrange multiplier, 398
theorem of equivalent sample size, 180
theorem of SIGMA-gate inference, 107
Theorist, 240
thickening phase, 41
thin client, 86
Thinker, 238
thinning phase, 41
threshold function, 427
 t_{ij} , 168
TLM. *See* Triangular Learner Model
topology, 425
total clockwise weight, 133
total counterclockwise weight, 133
total probability rule, 91
trained neural network, 435
training data, 90
transformation coefficient, 112
transformation phase, 381
transition Bayesian network (transition BN), 188
transition dependency, 188
transition probability, 188
transition probability distribution, 188
transition probability matrix, 244
transition weight, 191
trials, 90
Triangular Learner Model, 51
trigamma function, 214
trigger, 6
trust Bayesian network (trust BN), 165
tutoring module. *See* pedagogical module
two-class classification, 408
two-class classifier, 408
type attribute, 367
ubiquitous computing, 507
ubiquitous user model service, 519
ubiquitous user modeling, 505
Ubiquitous World, 518
UbiTools, 519
UbisWorld. *See* Ubiquitous World
U-gate inference, 129

- um (a user modeling shell), 45
- um toolkit, 45
- UMS. *See* user modeling system
- UMT. *See* User Modeling Tool
- uncovering problem, 246
- understanding group, 242
- Undirected*, 241
- uniform virtual world, 518
- union operator, 91
- unit. *See* processing unit
- unsupervised learning, 428
- upward inference direction, 507
- user communities, 438
- user groups, 438
- user interests, 389
- user interface, 12
- user mobility, 507
- user model, 1
- user model cluster, 439
- user model clustering, 439
- user model interest categories, 517
- user modeler, 12
- user modeling, 1
- user modeling component, 48
- user modeling server, 45
- user modeling shell, 43
- user modeling system, 1
- User Modeling Tool, 44
- user profile, 2
- UserML, 521
- UserQL, 521
- users, 2
- utility function, 465
- variable, 92
- vector space, 391
- Verbal*, 237
- Verbalizer*, 238
- Vermunt model, 241
- vertex, 444
- vertical data format, 380
- views, 48
- visited attribute, 150
- Visual*, 237
- Viterbi algorithm, 263
- web-based learning, 1
- weight vector, 392
- $\text{weight}(v_{ij})$, 447
- weighted graph, 99
- weighted sum, 426
- wet grass, 93
- Wholist*, 237
- within-component layer, 20
- Witkin model, 237
- Wolfe problem, 398
- WOW, 76
- X-D network, 134
- X-gate bi-inferences, 132
- X-gate diagnostic network, 134
- X-gate diagnostic relationship, 134
- X-gate inhibition, 117
- X-gate network, 116
- XHTML, 62
- XNOR-gate inference, 127
- XOR-gate inference, 126
- YACF. *See* Yet Another Clustering Formalism
- Yet Another Clustering Formalism, 26
- Zebra, 50
- Zebra control panel, 61
- Zebra server, 59
- $\beta(x; a, b)$. *See* beta density function
- Γ . *See* gamma function
- ζ , 395

Research History

Author:	Loc Nguyen
Editor:	Prof. Dr. Dong, Bich-Thuy T.
Affiliations:	University of Science – Vietnam National University, Ho Chi Minh city, Vietnam. Sunflower Soft Company, Ho Chi Minh city, Vietnam. Scientific Research Publishing Inc. London Mathematical Society, London, UK.
Research version:	4.8 build 2018.04
Book edition:	1 st

Primary tasks	
2006-2007	- <i>Researching task:</i> Researching domains: data mining, Bayesian network, user modeling and adaptive learning.
2007-2008	- <i>Researching task:</i> Making the surveys of user modeling and adaptive learning. - <i>Publishing task:</i> Publishing the conference paper “Learner Model in Adaptive Learning” in University of Information Technology, Ho Chi Minh, Vietnam. This paper is study report which does not contains my scientific findings. I keep it in my publications so as to note my first effort in research career and express my deep gratitude to authors who provided me theirs invaluable materials and works for studying in my early research career. For authors who I did not make references to their works, I apologize deeply them for this error. For fixing this error, I have made the full references in the book that contains this report. - <i>Programming task 1:</i> Finding out the generic adaptive system AHA! developed by the author Paul De Bra. Begin to modify AHA!
Version 1.0 build 2009.04.25	- <i>Designing task 1:</i> designing the architecture of user model called Triangular Learner Model (TLM) that constituted of three sub-models: Knowledge (K), Learning Style (LS) and Learning History (LH). - <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 allow users and adaptive systems to see or modify restrictedly TLM. - <i>Publishing task:</i> Publishing the article “ Combination of Bayesian Network and Overlay Model in User Modeling ” in International Journal of Emerging Technologies in Learning (iJET). - <i>Writing tasks:</i> collecting all relative papers and considering them as resource to compose the dissertation. - <i>Writing bug fixing:</i> checking spelling and grammar.
Version 1.1 build 2009.04.26	- <i>Writing tasks:</i> structuring report, aggregating particular parts. - <i>Writing bug fixing:</i> checking spelling & grammar, repainting figures, normalizing formulas, aligning tables.
Version 1.1 build 2009.05.07	- <i>Writing tasks:</i> composing section III.3.2 “discovering user interests”.
Version 1.2 build 2009.06.11	- <i>Designing task:</i> the architectures of both Triangular Learner Model (TLM) and Zebra are completed. - <i>Publishing task 1:</i> Publishing the conference paper “ Learning Concept Recommendation based on Sequential Pattern Mining ” in Proceedings of the 2009 Third International Digital Ecosystems and Technologies Conference (IEEE-DEST 2009).

	<ul style="list-style-type: none"> - <i>Publishing task</i> 2: Publishing the conference paper “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”. - <i>Writing tasks</i>: did finish section discussing about user interest. - <i>Programming tasks</i>: did build up Zebra server, Zebra client. So the architectures of both Triangular Learner Model (TLM) and Zebra are implemented. Note that Zebra is written by Java language. The software Zebra was finished basically. - <i>Writing bug fixing</i>: not yet.
Version 1.3 build 2009.07.18	<ul style="list-style-type: none"> - <i>Writing tasks</i>: composing Bayesian approach and overview of ubiquitous user modeling (in chapter: Conclusion & Future Trend) but not finish yet. - <i>Publishing task</i> 1: Publishing the conference paper “State of the Art of Adaptive Learning” in 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). This paper is study report which does not contains my scientific findings. I keep it in my publications so as to note my first effort in research career and express my deep gratitude to authors who provided me theirs invaluable materials and works for studying in my early research career. For authors who I did not make references to their works, I apologize deeply them for this error. For fixing this error, I have made the full references in the book that contains this report. - <i>Publishing task</i> 2: Publishing the conference paper “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 Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP’09).
Version 1.3 build 2009.07.22	<ul style="list-style-type: none"> - <i>Writing tasks</i>: composing Bayesian approach and overview of ubiquitous user modeling (in chapter: Conclusion & Future Trend) but not finish yet.
Version 1.3 build 2009.07.25	<ul style="list-style-type: none"> - <i>Writing tasks</i>: composing Bayesian approach and overview of ubiquitous user modeling (in chapter: Conclusion & Future Trend) but not finish yet.
Version 1.3 build 2009.07.27	<ul style="list-style-type: none"> - <i>Writing tasks</i>: concluding Bayesian approach (in chapter: Conclusion & Future Trend) but not finish yet.
Version 1.3 build 2009.07.29	<ul style="list-style-type: none"> - <i>Writing tasks</i>: finish the evaluation of Bayesian network but not finish the assessment of such network yet.
Version 1.3 build 2009.08.01	<ul style="list-style-type: none"> - <i>Writing tasks</i>: composing CAT in the assessment of Bayesian network.
Version 1.4 build 2009.08.02	<ul style="list-style-type: none"> - <i>Writing tasks</i>: doing finish the draft of dissertation but still missing many tables and figures in the last chapter.
Version 1.4 build 2009.08.05	<ul style="list-style-type: none"> - <i>Writing tasks</i>: All tables and figures in the last chapter are included. Doing finish sections Acknowledgement, Abstract, Reference & Appendix. The dissertation was finished basically.
Version 1.4 build 2009.08.11	<ul style="list-style-type: none"> - <i>Writing tasks</i>: composing state of the art of Bayesian network user model (section I.3).
Version 1.5 build 2009.08.14	<ul style="list-style-type: none"> - <i>Writing tasks</i>: do finishing the state of the art of Bayesian network user model (section I.3).
Version 1.5 build 2009.08.16	<ul style="list-style-type: none"> - <i>Writing tasks</i>: adding a little content, repairing some thing.
Version 1.5 build 2009.08.25	<ul style="list-style-type: none"> - <i>Writing tasks</i>: doing finish important report slides.
Version 1.5 build 2009.08.27	<ul style="list-style-type: none"> - <i>Writing tasks</i>: drafting the likelihood estimation technique used to specify prior probabilities (new sub-section III.1.5).
Version 1.5 build 2009.08.30	<ul style="list-style-type: none"> - <i>Writing tasks</i>: doing finish the likelihood estimation technique (sub-section III.1.5).
Version 1.6 build 2009.08.31	<ul style="list-style-type: none"> - <i>Writing tasks</i>: re-arranging all chapters, items and decorating the form of dissertation. The dissertation was finished totally, which was submitted for

	<p>the Degree of Doctor of Philosophy in Computer Science at University of Science, Vietnam National University, Ho Chi Minh city, Vietnam.</p> <ul style="list-style-type: none"> - <i>Publishing task:</i> Publishing the conference paper “A Proposal Discovering User Interests by Support Vector Machine and Decision Tree on Document Classification” in 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). - <i>Programming tasks:</i> re-arranging source code.
Version 1.6 build 2009.09.04	<ul style="list-style-type: none"> - <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.
Version 1.7 build 2009.09.09	<ul style="list-style-type: none"> - <i>Writing tasks:</i> checking spelling and grammar. - <i>Programming tasks:</i> re-arranging and optimizing source code.
Version 1.8 build 2009.09.14	<ul style="list-style-type: none"> - <i>Writing tasks:</i> checking spelling and grammar. - <i>Programming tasks:</i> re-arranging and optimizing source code. The software Zebra was finished totally.
Version 1.9 build 2009.09.16	<ul style="list-style-type: none"> - <i>Writing tasks:</i> checking grammar and decorating representation. - <i>Programming tasks:</i> programming some utility services.
Version 2.0 build 2009.09.18	<ul style="list-style-type: none"> - <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.
Version 2.0 build 2009.09.22	<ul style="list-style-type: none"> - <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.
Version 2.0 build 2009.09.24	<ul style="list-style-type: none"> - <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 and publishing tasks:</i> researching the computer-based testing technique and integrating such technique into Zebra. Publishing the conference paper “Incorporating Bayesian Inference into Adaptation Rules in AHA architecture” in Proceedings of 12th International Conference Interests Interactive Computer aided Learning (ICL2009).
Version 2.0 build 2009.09.27	<ul style="list-style-type: none"> - <i>Learning material composing task:</i> designing the concepts and 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 sponsor’s advice (so difficult but more effective). - <i>Researching task 2:</i> researching the computer-based testing technique and integrating such technique into Zebra.
Version 2.1 build 2009.09.29	<ul style="list-style-type: none"> - <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.
Version 2.4 build 2009.10.07	<ul style="list-style-type: none"> - <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>Publishing task:</i> Publishing the book chapter “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).
Version 2.6 build 2009.10.09	<ul style="list-style-type: none"> - <i>Designing task:</i> improving the physical architecture of the adaptive learning system WOW. WOW interacts with Zebra via communication interfaces (CI). WOW is modified version of AHA! developed by the author Paul De Bra. - <i>Programming task:</i> doing finish integrating the computer-based testing technique into Zebra.
Version 2.9 build 2009.10.11	<ul style="list-style-type: none"> - <i>Designing task:</i> The physical architecture of the adaptive learning system WOW is improved.

	<ul style="list-style-type: none"> - <i>Programming task</i>: doing finish improving the physical architecture of the adaptive learning system WOW.
Version 3.0 build 2009.10.12	<ul style="list-style-type: none"> - <i>Programming task</i>: fixing and improving Zebra. - <i>Designing task</i>: Improving physical architecture of Zebra.
Version 3.0 build 2009.10.14	<ul style="list-style-type: none"> - <i>Programming task 1</i>: fixing new fatal errors in Zebra server control panel when the physical architecture was changed. - <i>Programming task 2</i>: enhancing EM algorithm when evaluating (assessing) user knowledge. - <i>Programming task 3</i>: fixing Dynamic Bayesian network.
Version 3.0 build 2009.10.16	<ul style="list-style-type: none"> - <i>Learning material composing task</i>: doing finish basically designing the concepts and media for the course “Java tutorial”. - <i>Programming task</i>: enhancing Dynamic Bayesian network. - <i>Testing task</i>: testing Bayesian network, EM algorithm for determining prior probabilities, dynamic Bayesian network when evaluating user knowledge.
Version 3.0 build 2009.10.17	<ul style="list-style-type: none"> - <i>Programming task</i>: fixing some bugs in Zebra server control panel. - <i>Testing task</i>: testing Bayesian network, EM algorithm for determining prior probabilities, dynamic Bayesian network when evaluating user knowledge.
Version 3.1 build 2009.10.21	<ul style="list-style-type: none"> - <i>Programming task 1</i>: improving pattern mining algorithm - <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. - <i>Bug fixing</i>: fixing some bugs in Zebra control panel.
Version 3.1 build 2009.10.22	<ul style="list-style-type: none"> - <i>Writing tasks</i>: checking spelling and grammar, adding a little content.
Version 3.2 build 2009.10.26	<ul style="list-style-type: none"> - <i>Programming task</i>: add the new module “Searching Engine” into WOW so as to allow students to search their considerable learning materials.
Version 3.2 build 2009.10.27	<ul style="list-style-type: none"> - <i>Bug fixing</i>: fixing some bugs in module “Searching Engine”.
Version 3.2 build 2009.10.29	<ul style="list-style-type: none"> - <i>Bug fixing</i>: fixing some bugs in module “Searching Engine”.
Version 3.2 build 2009.10.31	<ul style="list-style-type: none"> - <i>Programming task</i>: enhancing the ajax GUI of module “Searching Engine”.
Version 3.2 build 2009.11.05	<ul style="list-style-type: none"> - <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 use collaborative function (so as to chat, discuss, etc.). - <i>Programming tasks</i>: optimizing source code.
Version 3.2 build 2009.11.11	<ul style="list-style-type: none"> - <i>Writing tasks</i>: doing finish the presentation slide “Zebra: A Modeling System for Triangular Learner Model” that summarizes dissertation.
Version 3.2 build 2009.11.14	<ul style="list-style-type: none"> - <i>Writing tasks</i>: composing section III.3.3 “constructing user group or user community” (user model clustering). - <i>Programming tasks</i>: fixing bugs and optimizing source code. - <i>Testing task</i>: testing Zebra and WOW.
Version 3.3 build 2009.11.18	<ul style="list-style-type: none"> - <i>Writing tasks</i>: doing finish section III.3.3 “constructing user group or user community”. - <i>Programming tasks</i>: fixing bugs and optimizing source code. - <i>Testing task</i>: testing Zebra and WOW.
Final version 3.3 build 2009.11.21	<ul style="list-style-type: none"> - Writing tasks: finish checking spelling and grammar in the PhD dissertation. - Programming tasks: finish fixing bugs and optimizing source code in program. - Testing task: finish testing Zebra and WOW. - Publishing task: The research is presented at Athabasca University, Athabasca, Alberta, Canada.
<h2>Extension 1</h2> <p>(Enhancing the software Zebra)</p>	
Version 3.4 build 2009.11.26	<ul style="list-style-type: none"> - <i>Learning material composing task</i>: doing repair tests in Java tutorial course - <i>Programming tasks</i>: adding function that allows students to see their

	knowledge evaluation in form of hierarchical domain.
Version 3.4 build 2009.12.21	<ul style="list-style-type: none"> - <i>Programming tasks:</i> Enhancing Zebra server control panel.
Version 3.4 build 2009.12.25	<ul style="list-style-type: none"> - <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.
Version 3.4 build 2009.12.26	<ul style="list-style-type: none"> - <i>Bug fixing and testing tasks:</i> Fixing bugs and testing feedback module.
Version 3.5 build 2009.12.31	<ul style="list-style-type: none"> - <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.
Version 3.5 build 2010.01.01	<ul style="list-style-type: none"> - <i>Programming tasks:</i> improving Zebra server control panel totally.
Version 3.5 build 2010.01.03	<ul style="list-style-type: none"> - <i>Programming tasks 1:</i> discovering user interest based on document classification. This function is described in section III.3.2 in this dissertation. Therefore, the series of user's accesses in her/his 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.
Version 3.5 build 2010.01.04	<ul style="list-style-type: none"> - <i>Bug fixing tasks:</i> fixing some errors occurring in classification algorithm (Decision Tree) and bugs in word segmentation. Zebra uses inside word segmentation module for English language. Because Zebra doesn't use another open source tool, it has some limitation.
Version 3.6 build 2010.01.15	<ul style="list-style-type: none"> - <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 TriUMQuery or Query Delegator. - <i>Bug fixing tasks:</i> fixing some errors in Zebra control panel.
Version 3.6 build 2010.01.22	<ul style="list-style-type: none"> - <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).
Version 3.6 build 2010.01.26	<ul style="list-style-type: none"> - <i>Programming tasks:</i> enhancing Zebra control panel.
Version 3.7 build 2010.01.28	<ul style="list-style-type: none"> - <i>Programming task 1:</i> supporting report on user's learning process. - <i>Programming task 2:</i> improving Zebra client.
Version 3.7 build 2010.01.30	<ul style="list-style-type: none"> - <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

(Enhancing the software Zebra)

Version 3.8 build 2010.02.01	<ul style="list-style-type: none"> - <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.
Version 3.8 build 2010.02.02	<ul style="list-style-type: none"> - <i>Review task:</i> reviewing source code and changing some method names, class names, etc. and adding some utility methods.
Version 3.8 build 2010.02.03	<ul style="list-style-type: none"> - <i>Programming task:</i> adding finding user function into Zebra control panel.
Version 3.8 build 2010.02.05	<ul style="list-style-type: none"> - <i>Programming task:</i> adding more utility methods and changing some method names, class names.
Version 3.9 build 2010.02.11	<ul style="list-style-type: none"> - <i>Programming task 1:</i> reviewing and improving source code. - <i>Programming task 2:</i> Programming sending e-mail function so as to support

	<p>mailing list in future.</p> <ul style="list-style-type: none"> - <i>Bug fixing task:</i> the engines (ME and BNE) of Zebra are composed physically by many threads 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, there are 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.
Version 3.9 build 2010.02.16	<ul style="list-style-type: none"> - <i>Bug fixing task:</i> Zebra server now runs steadily and stably. The out of memory error is fixed totally.
Version 3.9 build 2010.02.18	<ul style="list-style-type: none"> - <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".
Version 3.10 build 2010.02.20	<ul style="list-style-type: none"> - <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) called TriUMQuery or Query Delegator. - <i>Programming task 2:</i> enhancing source code.
Version 3.10 build 2010.09	<ul style="list-style-type: none"> - <i>Publishing tasks:</i> The research is presented at Doctoral Consortium of the 3rd International Conference on Theories and Applications of Computer Science (ICTACS2010), Can Tho, Vietnam, September 26, 2010 and Doctoral Consortium of The 4th Conference on Information Technology and Telecommunications (ICTFIT2012), University of Science, Ho Chi Minh city, Vietnam.
Version 3.10 build 2012	<ul style="list-style-type: none"> - <i>Publishing tasks:</i> The research is presented at the Doctoral Consortium of the 4th Conference on Information Technology and Telecommunications (ICTFIT2012), University of Science, Ho Chi Minh city, Vietnam.
Version 3.11 build 2013.06.12	<ul style="list-style-type: none"> - <i>Writing task and researching task:</i> Re-structuring the dissertation, writing additional chapter "Evaluation of Triangular Learner Model". - <i>Publishing tasks:</i> Publishing 3 articles "A New Approach for Modeling and Discovering Learning Styles by Using Hidden Markov Model" in (G. Perry et al., Eds.) Global Journal of Human Social Science: G - Linguistics & Education, "The Bayesian approach and suggested stopping criterion in Computerized Adaptive Testing" in International Journal of Research in Engineering and Technology (IJRET), and "Overview of Bayesian Network" in Science Journal Publication, Warri, Delta State, Nigeria.
Version 3.11 build 2013.08.03	<ul style="list-style-type: none"> - <i>Writing tasks:</i> Checking dissertation thoroughly for claims, content and grammar. It is perfect.
Version 3.11 build 2013.10	<ul style="list-style-type: none"> - <i>Publishing task:</i> The research is evaluated secondly at Athabasca University, Athabasca, Alberta, Canada.
Extension 3	
(The PhD dissertation is developed as advanced scientific research focused on mathematics)	
Version 4 build 2014.11.24	<ul style="list-style-type: none"> - <i>Publishing task 1:</i> The article "Specifying Prior Probabilities in Bayesian Network by Maximum Likelihood Estimation method" is accepted to be published in Sylwan journal. - <i>Publishing task 2:</i> Publishing the article "User Model Clustering" in Journal of Data Analysis and Information Processing (JDAIP).

	<ul style="list-style-type: none"> - <i>Publishing task 3:</i> Publishing the article “A New Algorithm for Modeling and Inferring User’s Knowledge by Using Dynamic Bayesian Network” in Statistics Research Letters (SRL). - <i>Publishing task 4:</i> Publishing the entire PhD dissertation “A User Modeling System for Adaptive Learning” in Standard Research Journals. The dissertation is awarded as excellent research. - <i>Writing task 1:</i> Re-organize the whole research. - <i>Writing task 2:</i> Completing references over the whole research. - <i>Writing task 3:</i> Writing additional contents in sub-section III.1.5.2 “Beta likelihood estimation”. - <i>Writing task 4:</i> Writing additional contents in sub-section III.3.2.2 “Methods of document classification”, especially, support vector machine. - <i>Writing task 5:</i> Writing additional contents in sub-section IV.1.2.2 “Towards the Computerized Adaptive Testing”. - <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”. - <i>Writing task 7:</i> All tables, figures, and formulas are indexed. - <i>Writing task 8:</i> Fixing writing errors, creating figures, adding more examples, and enhancing representation over the whole research. - <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”. However, this algorithm is incorrect and it was removed. - <i>Creative work 2:</i> Inventing new formula for beta likelihood estimation in sub-section III.1.5.2 “Beta likelihood estimation”. - <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”.
Version 4.1 build 2014.12.18	<ul style="list-style-type: none"> - <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. - <i>Writing task 2:</i> Composing examples for advanced CAT algorithm. - <i>Publishing task 1:</i> Publishing the conference paper “Evaluating Adaptive Learning Model” in Interactive Collaborative Learning (ICL), 2014 International Conference on. - <i>Publishing task 2:</i> Demonstrating the user modeling system Zebra at The 2014 World Engineering Education Forum (WEEF2014). The research is certificated by World Engineering Education Forum (WEEF2014). - <i>Creative work 2:</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”. - <i>Programming task 1:</i> Implementing document classification based on neural network for discovering user interests. Besides computer softwares such as user modeling system Zebra and adaptive learning system WOW, the new module called Mumples is associated to this book so as to implement algorithms, mathematical models, examples inside the book. - <i>Programming task 2:</i> Implementing the advanced CAT algorithm inside Mumples.
Version 4.2 build 2014.12.27	<ul style="list-style-type: none"> - <i>Writing task 1:</i> Composing the new sub-section II.2.4 “Implementation of Zebra” when Zebra is implemented as computer software. - <i>Writing task 2:</i> Composing the new sub-section II.2.4 “New version of simple equations whose solutions are parameter estimators”. - <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

	<p>parameter estimators”.</p> <ul style="list-style-type: none"> - <i>Programming task</i> 1: Implementing the iterative algorithm solving new simple equations whose solutions are parameter estimators, sub-section III.1.5.5 “New version of the equations whose solutions are parameter estimators”.
Version 4.3 build 2015.01.02	<ul style="list-style-type: none"> - <i>Writing task</i> 1: Indexing all terms, keywords and completing appendix D “Index”. - <i>Writing task</i> 2: Creating more figures. - <i>Writing task</i> 3: Composing an example for k-medoid algorithm described in sub-section III.3.3.3 “Similarity measures for clustering algorithms”.

Extension 4

(The advanced research is composed as the book “**Mathematical Approaches to User Modeling**” with the first edition)

Version 4.4 build 2015.03.01 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Publishing task</i> 1: The article “Theorem of SIGMA-gate Inference in Bayesian Network” is accepted to be published in Wulfenia journal. - <i>Writing task</i> 1: Explaining hidden Markov model in detailed in section IV. 4 “Hidden Markov model” with full of proofs and examples. - <i>Writing task</i> 2: Fixing the error in calculating partial derivatives of dual function in sub-section V.2.2.1 “Document classification based on support vector machine”. - <i>Writing task</i> 3: Re-structuring the book “Mathematical Approaches to User Modeling” that includes seven chapters: <ul style="list-style-type: none"> · Chapter I is a survey of user model, user modeling, and adaptive learning. · Chapter II introduces the general architecture of the proposed user modeling system Zebra and TLM. · Chapter III, IV, V describes three sub-models of TLM such as knowledge sub-model, learning style sub-model, and learning history sub-model in full of mathematical formulas and fundamental methods. These are the most important chapters. · Chapter VI gives some approaches to evaluate TLM and Zebra. · Chapter VII summarizes the research and discusses future trend of Zebra. - <i>Creative work</i> 1: Inventing the algorithm to solve HMM uncovering problem based on finding out the longest path of graph in sub-section IV.4.2 “HMM uncovering problem”. - <i>Reviewing task</i> 1: Reviewing the book “Mathematical Approaches to User Modeling” thoroughly.
Version 4.5 build 2015.05.01 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Writing task</i> 1: Re-indexing all tables, figures, and formulas. - <i>Writing task</i> 2: Completing examples of hidden Markov model in section IV.4 “Hidden Markov model”. - <i>Writing task</i> 3: Explaining example of breaking sequential pattern, concretely, Apriori algorithm in sub-section V.1.2 “A proposal of breaking sequential pattern in learning context”. - <i>Writing task</i> 4: Composing the subject “quadratic programming to solve multipliers in support vector machine” in sub-section V.2.2.1 “Document classification based on support vector machine”. - <i>Writing task</i> 5: Enhancing the method of breaking sequential pattern in learning context in sub-section V.1.2 “A proposal of breaking sequential pattern in learning context”. - <i>Writing task</i> 6: Enhancing the example of adaptive learning evaluation in sub-section VI.2.2 “An evaluation scenario”. - <i>Writing task</i> 7: Composing figures, checking spelling and grammar,

	<p>enhancing appendix “D. Index”.</p> <ul style="list-style-type: none"> - <i>Researching task 1:</i> Researching the subject “quadratic programming to solve multipliers in support vector machine”. The algorithm “Sequential Minimal Optimization (SMO)” is studied in prior.
Version 4.6 build 2015.06.01 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Creative work 1:</i> Proposing an advanced implementation of EM algorithm for learning hidden Markov model with the support of terminating criterion $P(O/\Delta)$. The criterion $P(O/\Delta)$ is evaluated by taking advantages of forward variables and backward variables at E-step so as to keep the EM algorithm in stable speed. The creative work is described in section IV.5 “Continuously observational hidden Markov model”. - <i>Writing task 1:</i> Complete advanced examples relevant to continuously observational hidden Markov model in section IV.5 “Continuously observational hidden Markov model”. - <i>Writing task 2:</i> Re-creating some figures. - <i>Reviewing task 1:</i> Checking spelling and grammar and reviewing entire the book. - <i>Programming task 1:</i> Implementing entire hidden Markov model, which solves three problems such as evaluation problem, uncovering problem, and learning problem. Advanced EM algorithm with the support of terminating criterion $P(O/\Delta)$ is implemented. The longest-path algorithm described in sub-section IV.4.2 “HMM uncovering problem” is also implemented.
Version 4.7 build 2015.07.01 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Writing task 1:</i> Enhancing sub-section II.2.4 “Implementation of Zebra”. - <i>Writing task 2:</i> Adding more references into the sub-section II.2.4 “Implementation of Zebra”. - <i>Writing task 3:</i> Enhancing section III.2 “Incorporate Bayesian inference into adaptation rules” by explaining how to incorporate Bayesian inference into adaptation rules in detail. - <i>Writing task 4:</i> Enhancing sub-section III.4.2 “Using dynamic Bayesian network to model user’s knowledge”. Explaining more the proposed algorithm to construct optimal dynamic Bayesian network. - <i>Writing task 5:</i> Composing an example for k-mean algorithm in sub-section V.3.1 “User model clustering”. - <i>Reviewing task 1:</i> Checking spelling and grammar and reviewing entire the book.
Version 4.7 build 2016.05.08 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Reviewing task 1:</i> Checking spelling and grammar and reviewing entire the book. - <i>Publishing task 1:</i> Publishing the article “Specifying Prior Probabilities in Bayesian Network by Maximum Likelihood Estimation method” in ISI Sylwan Journal on February 2016. - <i>Publishing task 2:</i> Publishing the article “Theorem of SIGMA-gate Inference in Bayesian Network” in ISI Wulfenia Journal on March 28, 2016. - <i>Publishing task 3:</i> Publishing the article “New version of CAT algorithm by maximum likelihood estimation” in ISI Sylwan Journal on April 8, 2016. - <i>Publishing task 3:</i> Publishing the article “Beta Likelihood Estimation and Its Application to Specify Prior Probabilities in Bayesian Network” in British Journal of Mathematics on April 27, 2016. - <i>Publishing task 3:</i> Publishing this post-doctorate dissertation of Mathematics and Statistics in Journals Consortium on July 27, 2015. - <i>Writing task 1:</i> Adding reference to the work “A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation” of authors Eva Millán and José Luis Pérez-de-la-Cruz. This work is base of theorem of SIGMA-gate inference in Bayesian network in section III.1.3.

Extension 5

(Enhancing the first edition of the book “**Mathematical Approaches to User Modeling**”)

Version 4.8 build 2016.05.26 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Researching task 1:</i> Researching methods to convert prerequisite relationship of overlay model into conditional probability tables (CPT) of Bayesian network in order to complete the section III.1 “Combination of Bayesian network and overlay model in building up knowledge sub-model”. - <i>Writing task 1:</i> Enhancing the proof of SIGMA-gate theorem so that it is more accurate. - <i>Writing task 2:</i> Enhancing sub-section III.1.2 “Applying Bayesian network to overlay model” by adding the formula to specify conditional probabilities of diagnosis relationship. - <i>Writing task 3:</i> Fixing the calculation of Bayesian network posterior probabilities in section III.2 for adaptation rules. The hypothesis node now has only one evidence. - <i>Writing task 4:</i> Fixing the example of learning parameters of Java course Bayesian network in sub-section III.3.3. The hypothesis node now has only one evidence.
Version 4.8 build 2016.06.15 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Researching task 1:</i> Researching methods to convert prerequisite relationship of overlay model into conditional probability tables (CPT) of Bayesian network in order to complete the section III.1 “Combination of Bayesian network and overlay model in building up knowledge sub-model”. - <i>Publishing task 1:</i> Publishing the article “Continuous Observation Hidden Markov Model” in ISI Kasmera Journal, volume 44, issue 6. The article is also described in the section IV.5 “Continuous Observation Hidden Markov Model”. - <i>Publishing task 2:</i> Publishing three articles “Tutorial on Support Vector Machine”, “Tutorial on Hidden Markov Model”, and “Longest-path Algorithm to Solve Uncovering Problem of Hidden Markov Model” in Special Issue “Some Novel Algorithms for Global Optimization and Relevant Subjects” of Applied and Computational Mathematics (ACM) on June 17, 2016. - <i>Writing task 1:</i> Enhancing section IV.5 “Continuous observation hidden Markov model”. - <i>Writing task 2:</i> Composing the subsection III.1.4 “Relationship conversion in Bayesian network” in order to complete the section III.1 “Combination of Bayesian network and overlay model in building up knowledge sub-model”.
Version 4.8 build 2016.07.13 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Reviewing task 1:</i> Reviewing the subsection III.1.4 “Relationship conversion in Bayesian network”. It is also a scientific manuscript.
Version 4.8 build 2016.10.21 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Reviewing task 1:</i> Re-formatting entirely the book with font “Times New Roman 12pt”.
Version 4.8 build 2016.11.29 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Publishing task 1:</i> Publishing the book chapter “Beta Likelihood Estimation in Learning Bayesian Network Parameter” in the book “Advances in Computer Networks and Information Technology (Vol. II)” by author United Scholars Publications on October 31, 2016. - <i>Publishing task 2:</i> Publishing the book chapter “Converting Graphic Relationships into Conditional Probabilities in Bayesian Network” in the book “Bayesian Inference” by author Javier Prieto Tejedor on November 2, 2017. - <i>Reviewing task 1:</i> Reviewing entirely the book.
Version 4.8 build 2018.04.09 Book edition: 1 st	<ul style="list-style-type: none"> - <i>Reviewing task 1:</i> Fixing errors in beta likelihood estimation in sections III.5.3, III.5.4, and III.5.5.

Author Biography



Loc Nguyen is an independent scholar at Loc Nguyen's Academic Network from 2017. He holds Master degree in Computer Science from University of Science, Vietnam in 2005. He holds PhD degree in Computer Science and Education at Ho Chi Minh University of Science in 2009. His PhD dissertation was certificated by World Engineering Education Forum (WEEF) and awarded by Standard Scientific Research and Essays as excellent PhD dissertation in 2014. He holds Postdoctoral degree in Computer Science, certified by Institute for Systems and Technologies of Information, Control and Communication (INSTICC) in 2015. Now he is interested in poetry, computer science, statistics, mathematics, education, and medicine. He serves as reviewer and editor in a wide range of international journals and conferences from 2014. He is volunteer of Statistics Without Borders from 2015. He was granted as Mathematician by London Mathematical Society for Postdoctoral research in Mathematics from 2016. He is awarded as Professor by Scientific Advances and Science Publishing Group from 2016. He was awarded Doctorate of Medicine by Ho Chi Minh City Society for Reproductive Medicine (HOSREM) from 2016. He has published 55 papers in journals, books and conference proceedings. He is author of 2 scientific books and 1 postdoctoral dissertation. He is author and creator of 6 scientific and technology products. Moreover, he is Vietnamese-language poet who has composed 1 verse story and 7 collections of 303 poems from 1993. He also has 3 music albums in which many poems are chanted by famous artists. Especially, he is very attractive, enthusiastic, and creative. His favorite statement is "Creative man is The Creator". Thus, why don't you contact him for sharing inspiration and knowledge? His online homepage is

<http://www.locnguyen.net>