Standard Scientific Research and Essays Vol1 (8): xxx-xxx, September 2013
http://www.standresjournals.org/journals/SSRE

Research Article

A User Modeling System for Adaptive Learning

Nguyen Phuoc Loc

Department of Information System, Faculty of Information Technology, University of Science Vietnam National University, Ho Chi Minh city

Email: ng_phloc@yahoo.com, Phone: +84-90-8222007

Accepted 18 July 2013

Abstract

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

Keywords: User Modeling System, E-learning, Triangular Learner Model

INTRODUCTION

According to this ideology, TLM will cover the whole of user's information required by learning adaptation process and give the best support to adaptive learning. The reasons for such assertion are:

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

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

User Model and Adaptive Learning

I.1. User Model

Formerly, Learning Management Systems (LMS) support well for interaction between learner and lesson, learner and teacher. However, every student has individual features such as knowledge, goal, experience, interest, background, etc. So, there is emergent demand for tailoring learning material (lesson, exercise, test…) to each person. This is learning personalized process and system, which supports such process, was called Adaptive Learning System.

Therefore, learning adaptive system is able to change its action to provide both learning content and pedagogic environment/method for student. Adaptive systems base on the "description of learner's properties" called user model or learner model. The process, which gathers information to build up learner model and update it, was named: learner modeling. Adaptive system tunes learning material & teaching method to learner model. However, learner model is focused in this thesis.

Note that user model, student model and learner model are synonymic terms because users are regarded as learners or students in learning context.

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.

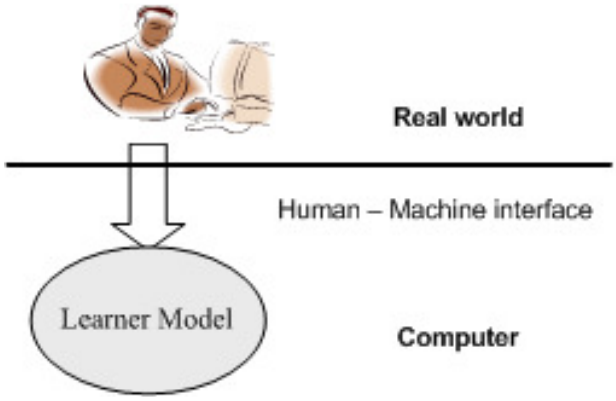


Figure I.1.1. User modeling

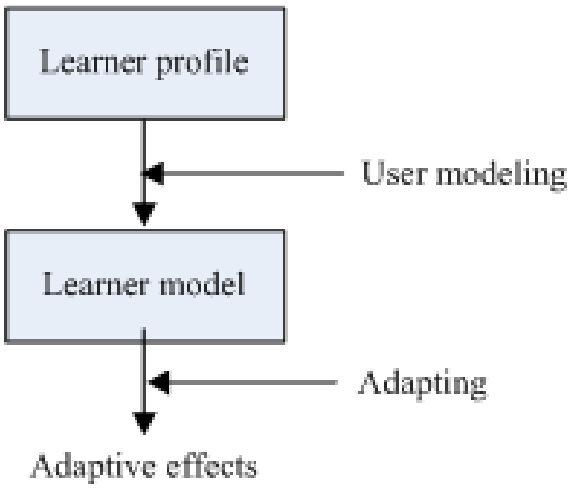


Figure I.1.2. Learner profile & model in adaptation [Brusilovsky, Maybury 2002]

Before discussing about user model, we should glance over what adaptive learning system is although that subject is explained in detailed in section I.2. As we know that our environment is very complex and everyone has individual characteristics. Learners are not the same "size" (physically and mentally). Moreover, user's preferences are various. So, adaptation becomes imperative, especially in education. The most popular adaptive learning system supporting personalized learning environment is AEHS (Adaptive Education Hypermedia System). Adaptation is ability to change system's behaviors to tune with learner model. 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, electronic book…) tailored to learner's preference. After that, description of learner's essential features and classification of learner models are described in section I.1.1 and I.1.2.

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…), environment (context of work) and other useful features.
[Brusilovsky 1994] stated that content of user model can be divided into two categories: domain specific information [Fröschl 2005] and domain independent information [Fröschl 2005].

I.1.1.1. Domain specific information

This information 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 concepts or topics or subjects in domain. Each element of vector which is a real number or integer number (range within an interval) shows the degree which learner gains knowledge about those concepts, topics or subjects. 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 section I.1.2).
- *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, evaluation, etc.

I.1.1.2. Domain Independent Information

Besides information about knowledge, domain independence information may include goals, interests, background and experience, individual traits, aptitudes and demographic information.

- *Interests*. Interest is particularly essential in commercial recommendation system. It is also important in adaptive educational system.
- *Goals*. In most cases, goal expresses learner's purpose, in other words; it is an answer for the question what learners want to achieve in learning course. There are two kinds of goal: long-term and short-term. Long-term goal is relatively permanent in course. Moreover, learner can propose themselves long-term plans for lifelong study. By short-term goal, learner intends to solve certain problem such as passing an examination, doing exercise, etc. Short-term goal was 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 AEHS should deliver high level exercises to him/her.
- *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 were defined as the way learner prefers to study. Table below shows some common learning styles.

Table I.1.1. Some common learning styles

GROUP	DESCRIPTION
ACTIVIST	LIKE TO HAVE A GO AND SEE WHAT HAPPENS
REFLECTOR	LIKE TO GATHER INFORMATION AND MULL THINGS OVER
PRAGMATISTS	LIKE TRIED AND TESTED TECHNIQUES WHICH ARE RELEVANT TO MY PROBLEMS
THEORIST	PREFER TO PERCEIVE LEARNING MATERIAL AS TEXT

There are eight forms of aptitudes [Fröschl 2005] [Lane 2000]: linguistic, logical/mathematical, spatial, kinesthetic, musical, interpersonal, intrapersonal, naturalist.

Table I.1.2. Eight forms of aptitudes

APTITUDE	DESCRIPTION
LINGUISTIC	COMPETENCE TO USE LANGUAGE
LOGICAL/MATHEMATICAL	COMPETENCE TO USE REASON, NUMBER AND LOGIC
SPATIAL	COMPETENCE TO PERCEIVE THE VISUAL
KINESTHETIC	COMPETENCE TO HANDLE OBJECTS SKILLFULLY
MUSICAL	COMPETENCE TO CREATE AND COMPOSE MUSIC
INTERPERSONAL	COMPETENCE TO COMMUNICATE WITH OTHER PERSON
INTRAPERSONAL	COMPETENCE TO SELF-REFLECT
NATURALIST	COMPETENCE TO REALIZE FLORA AND FAUNA

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… In general, demographic information is used to identify person.

I.1.2. Classification of User Models

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. By small amount of information in stereotype, it is able to infer much more new assumptions about user. 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*.

In fix stereotype, learner is assigned to predefined stereotype at abstract level. For example, in Java tutorial course, students are divided into five group, corresponding to five level, each level is more difficult than previous level: novice, begin, known, advanced and expert. After obtaining individual information such as former knowledge, experience; system will assign each student to one of five levels and never change.

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

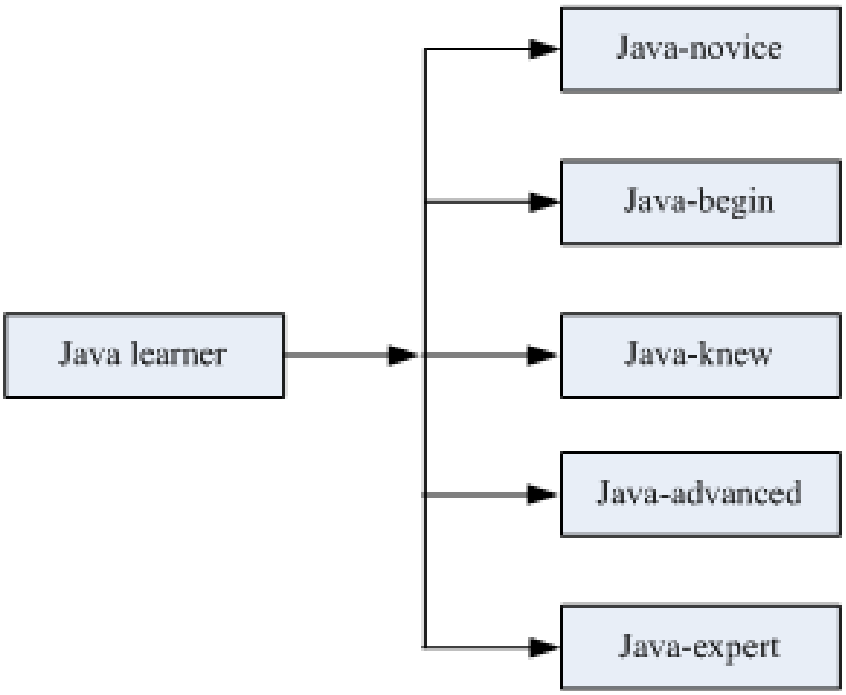


Figure I.1.3. Stereotypes

There are three important components in a stereotype: trigger, inference and retraction:

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

I.1.2.2. Overlay model

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

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

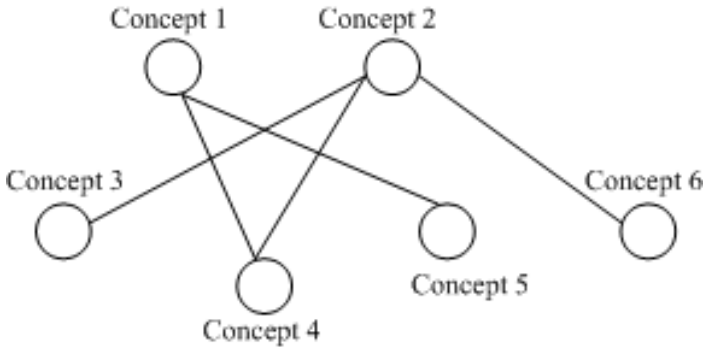


Figure I.1.4. Overlay Model

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 trend to build up domain model by ontology.

I.1.2.3. Differential model

Overlay model was based on expert's domain knowledge but there is need for learner/teacher to suppose the knowledge which is necessary to learner. That knowledge was called *expected knowledge*. In other word, expected knowledge is domain knowledge that learner should be mastered at the certain time.
Therefore, differential model is basically an overlay on expected knowledge, which in turn is an overlay on expert's domain knowledge. With the overview of top-down methodology, differential model is a variant of overlay model. But in detailed, the differential model is instance of the class "fault model" (see section I.1.1.2) 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 were considered as *mal-knowledge* or incorrect beliefs.
Perturbation model 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.

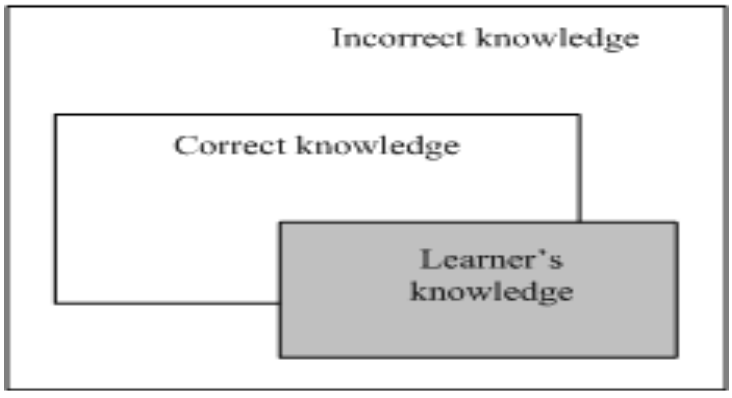


Figure I.1.5. Perturbation model adds incorrect knowledge to subset of expertise's knowledge

I.1.2.5. Plan model

Plan is a sequence of learners' actions to achieve desires or concrete goals. Plan recognition was based on tracking input user's performance [Kobsa 1993]. 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 & 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 were supposed in section I.1.2. However, we believe 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.
- *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 were done implicitly or explicitly.
- *Reasoning* new information about user out from available data in user model.

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

I.2. Adaptive Learning

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

Here, I intend to systematize methods and theories in developing adaptive learning systems along with their features. This section brings out the entire overview of adaptive learning system.

The term *adaptive* is defined as "able to change when necessary in order to deal with different situations". 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. Therefore, adaptive leaning systems tailor learning material to user information. The survey of existing adaptive systems is represented in this section

Section I.2.1 is to classify existing adaptive systems in their development history. Two modern and popular systems: ITS and AEHS are described in section I.2.2 and I.2.3; each system is surveyed entirely and enclosed with specific example. Section I.2.4 is the evaluation of existing 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]:
- Macro-adaptive system
- Micro-adaptive system
- Aptitude-treatment interactions system (ATI)
- Intelligent tutoring system (ITS)
- Adaptive Hypermedia System (or Adaptive Educational Hypermedia System)

Macro-adaptive system

The early researches on adaptive learning intend to adapt the instructional performances to students on the macro level. Such system was called macro-adaptive system [Fröschl 2005]. Students are classified into groups by grades from tests. Students in the same group have similar adaptive instruction. To identify each student with his/her 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, cognitive styles… ATI [Mödritscher 2004] also permits user to control partially or totally the learning process. User can control learning instruction or content presentation in course. Researches prove that successful level of user's control depends on his/her aptitudes.

Micro-adaptive system

This system, so-called micro-adaptive [Fröschl 2005] [Mödritscher 2004] performs adaptivities on micro level since it discovers and analyzes individuals need to provide user the appropriate instructions. When student is ongoing learning process, system observes and diagnoses [Mödritscher 2004] continuously his/her activities. 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: domain expert, user modeler, tutoring module and user interface which have respective functions (see I.2.2).

Adaptive Hypermedia System (AHS)

AHS has also been researched for a long time until now, which is the next generation of ITS. AHS combine 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. Hypermedia is an extension of hypertext which makes use of multiple forms of media, such as text, video, audio, graphics, etc…

According to [Brusilovsky 1994], 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. Users 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 his/her 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 I.2.3). For example, if user is a novice, system gives more annotation about the lecture which he/she 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 (see I.2.3). We will survey AEHS instead of AHS in I.3.

I.2.2. Intelligent Tutoring System

Formerly, intelligent tutoring system (ITS) and artificial intelligence (AI) are areas which have been researched separately. AI developed fast in 1960's; Alan Turing thought that computer can "think" as human. Education becomes the fertile ground for applying AI methods since computer plays the role of human teacher. More and more people attends distance courses in the universities and they want to become self-taught mans who prefer a lifelong study; so computer is the best choice. Early ITS is the Computer Assisted Instructional (CAI) system that was generative. 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. The available information in user model, especially knowledge becomes more and more important.

I.2.2.1. Architecture

As discussed, ITS [Mayo 2001] is the modern system since it inherits all strong points of both micro-adaptive system and ATI. ITS has components expressing the content taught, adaptive procedures and the techniques for collecting, storing user characteristics and inferring new assumptions from them. General architecture of ITS is constituted of four main parts:

- *Domain expert* 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* constructs and manages user information represented by user model. This includes long-term information such as goals, demography information, mastered knowledge and short-term information such as whether or not students do exercises, visit a web site… User modeler also interacts with pedagogical module to catch and log user's tasks. User model can be stored in database or XML files. User model has critical role; if it is bad in that it don't express solidly user's characteristics, the pedagogical module cannot make decisions in the proper way.
- *Pedagogical module* also called tutoring module or didactics module is the centric component in ITS, which adapts instructional procedures to students. This module makes decisions about the teaching process relating to the next problem selection, next topic selection, adaptive presentation of error messages, and selective highlighting or hiding of text. Pedagogical module co-operates with the user modeler and domain expert to draw information about domain knowledge and user when making decisions.
- *User interface* is the component taking full responsibility for user-machine interaction. The user interface is rather necessary when it gives user friendly environment and provides the motivation for student to learn. If other parts don't work properly or raises error, user interface can notice or guide user to overcome troubles when using ITS. The interface can improve learning by reducing the cognitive load. While three other parts focus on learning and adaptive procedures, the user interface aims to users.

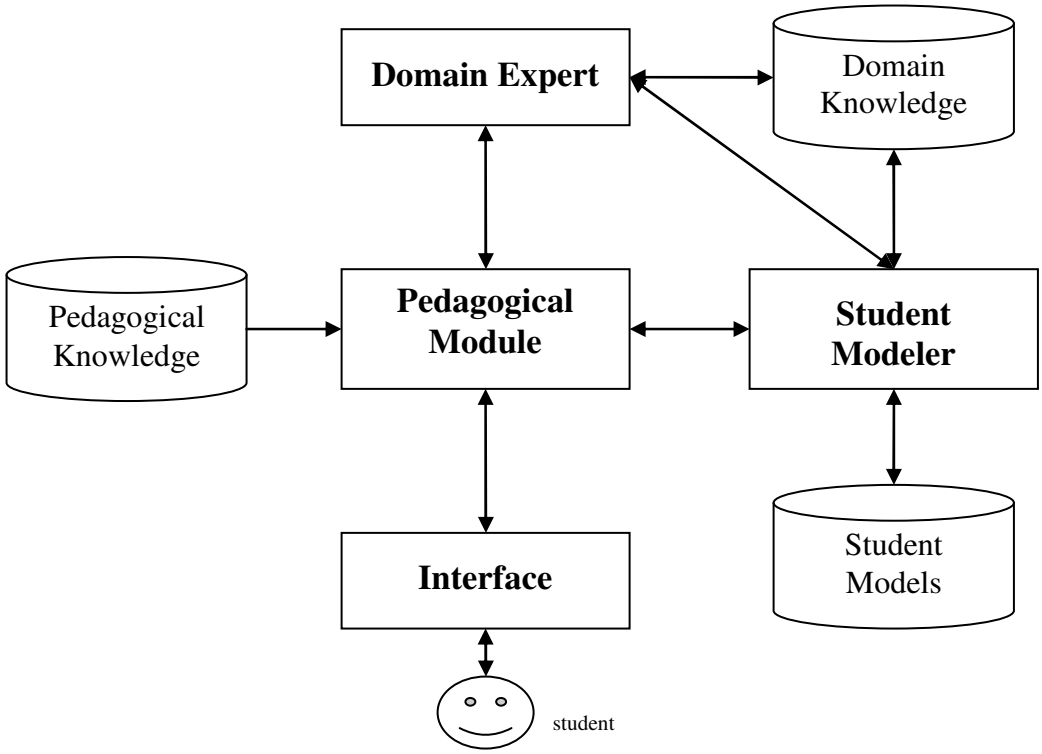


Figure I.2.1. General Architecture of ITS [Mayo 2001]

I.2.2.2. Characteristics

According to [Wenger 1987], 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". He state that the core of ITS – pedagogical module provides two main adaptive tasks (makes decisions): diagnosis and didactics.

- **Diagnosis**. The ITS "diagnoses" students' states as three levels:
  - *Behavioral level*: ignoring the learner's knowledge and focusing only on the observable behavior
  - *Epistemic level*: dealing with the learner's knowledge state and attempting to infer that state based on observed behavior
  - *Individual level*: covering such areas as the learner's personal traits, motivational style, self-concept in relation to the domain, and conceptions the learner has of the ITS. At this level, students become active learners.
- **Didactics** is the "delivery" aspect of teaching, which is also referred as making decisions process. Wenger [Wenger 1987] claims that didactics is implemented around four principles:
  - *Plans of action*: are used to lead the student and provide the context for diagnostic operations.
  - *Strategic contexts*: in which the plans of action are implemented.
  - *Target level* of the student model: selecting the level at which the teaching takes place. Depending on user state level, the pedagogical module will make appropriate instructional decisions.

I.2.2.3. An example: ANATOM-TUTOR

As the name suggests, ANATOM-TUTOR [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. Concepts associated to their relations are located in the concept hierarchy. The reasoning is executed by built-in mechanism.

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

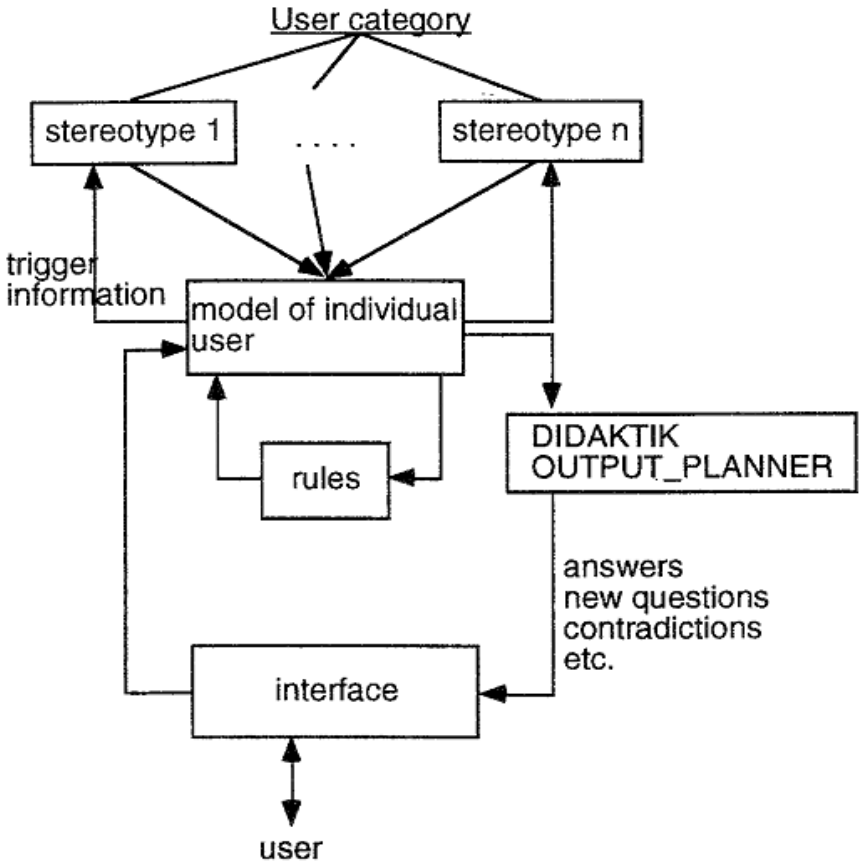


Figure I.2.2. User modeling component in ANATOM-TUTOR

The didactic module "teaches" user by providing the adaptive knowledge (in form of lesson, explanation…). There are two kinds of teaching knowledge:

- Global teaching knowledge refers to the general structure of a lesson.
- Local teaching knowledge refers here to what to do when a student gets into difficulties.

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 inherits basic components of ITS in respect of implementation but takes advantage of plentiful supplies of learning material in hypermedia space. As discussed, adaptation [Brusilovsky 1994] 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,

electronic book) tailored to learner's preference. According to [Brusilovsky 1996], there are two forms of adaptation: adaptive presentation and adaptive navigation:

- *Adaptive presentation* refers to the information which is show, in other word, what is shown to the user
- *Adaptive navigation* refers to manipulation of the links, thereby; the user can navigate through in hypermedia. In other word, it is where user can go
-

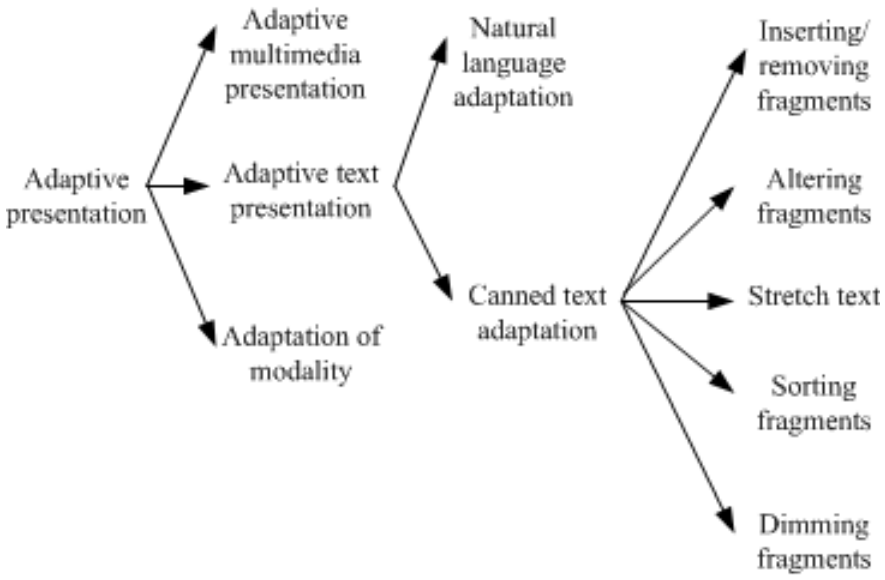


Figure I.2.3. Adaptive representation

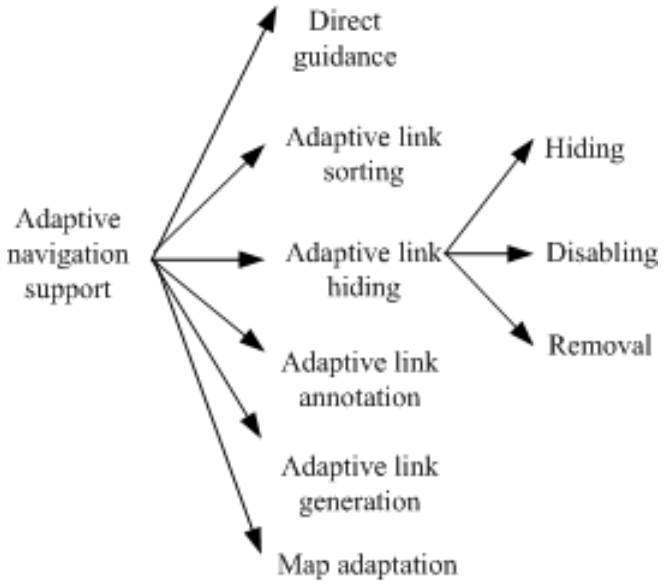


Figure I.2.4. Adaptive navigation

Canned text adaptation 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 supports some following cases of navigations:
- Direct guidance: guide the 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 the links of hypermedia due to their relevance for the user
- Adaptive link hiding: limit 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 the 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 the colored ball which is annotated the link pointing to a document in hypermedia space. The red ball indicates that document is not recommended to user. The yellow ball has the same meaning to red ball but it is less strict than red ball. The green ball hints that document should be recommended to user. The grey ball indicates that user has already known this document.
- Link generation: generating appropriate links so that system prevents user from getting involved in large hyperspace.
- Map adaptation: graphical overviews of adaptive links.

I.2.3.1. Architecture

In general, the architecture of AEHS [Karampiperis, Sampson 2005] 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 such as
- Initialize and update learner model.
- Choose concepts in domain model, educational resource in Media Space by selection rules.
- Store learning resources, domain ontology, learner model, etc.
-

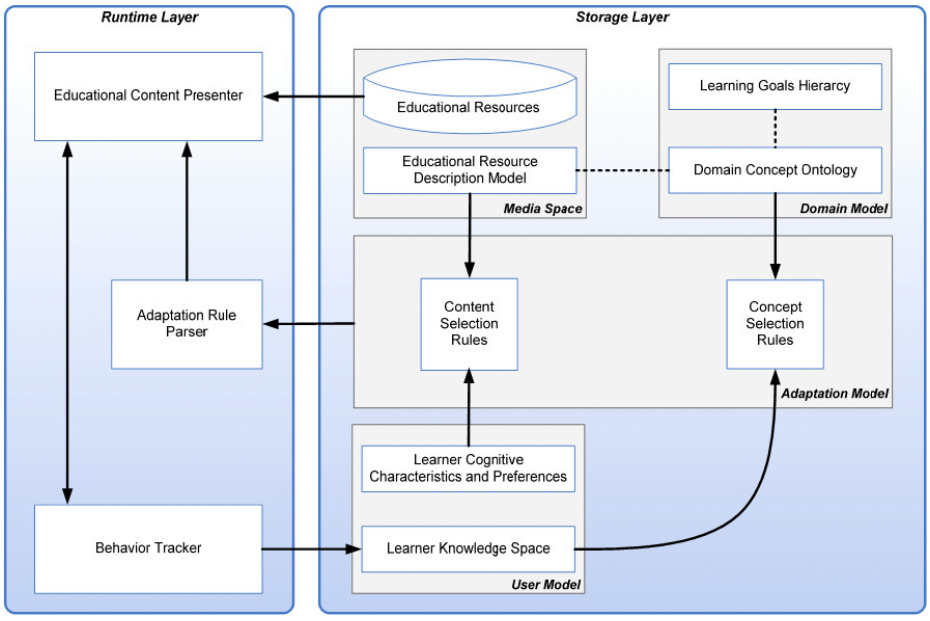


Figure I.2.5. General architecture of AEHS

As seen in general architecture, storage layer has four models:
- *Media model*: contains learning resource and associated descriptive information (metadata).

- *Domain model*: constitutes the structure of domain knowledge. Domain model was often shown in the form of graph. Nowadays, researchers intend to build domain model according to ontology.
- *Adaptation model*: is the centric component which gives effect to adaptation. It contains content selection rules and concept selection rules. We apply content selection rules into choosing suitable educational resource from medial model. On the other hand, concept selection rules are used to choose appropriate concept from domain model. These rules must tune with user model so that the selection gets correct.
- *User Model*: information and data about user.

I.2.3.2. Characteristics

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

- Predicate *part_of*(doc_id_1, doc_id_2) means that doc_id_2 is a part of doc_id_1
- Predicate *preq*(doc_id_1, doc_id_2) means that doc_id_2 is prerequisite of doc_id_1
- Predicate *is_a*(doc_id_1, doc_id_2) expresses the taxonomy on documents.
- Predicate *is_dependent*(doc_id_1, doc_id_2) expresses the dependencies between documents.

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

UM is responsible for managing information about user such as knowledge, goals, learning styles, etc. In short, UM has below functions:

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

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

Each element in UM represents a user, which denoted logically by identifier *user_id*. Characteristics assigned to user are expressed by predicates: *has_property(user id, characteristic x), has_property(user id, characteristic x, value)*. The very important characteristic "knowledge" is expressed by predicates:

- *know(doc_id, user_id)*: tell us whether user knows document.
- *know(doc_id, user_id, value)*: tell us amount of knowledge user has on document. The variable *value* refers the user's knowledge level.

OBS. 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:

- *obs(doc_id, user_id)*
- *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, the most important component, contains rules supporting adaptive procedures. In other words, the 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 he/she has already visited (learned) all prerequisites of this document. This rule is expressed as FOL predicate:

$$\forall user\_id, \forall doc\_id_1\ (\forall doc\_id_2, preq(doc\_id_1, doc\_id_2) \Rightarrow obs(doc\_id_2, userid, visited))$$

$$\Rightarrow learning\_state(doc\_id_1, user\_id, recommended\_for\_reading)$$

I.2.3.3. An example: AHA!

(**A**daptive **H**ypermedia for **A**ll)
AHA! [De Bra, Calvi 1998], a Java-based open source soft is based on Dexter Hypertext Reference Model [Halasz, Schwartz 1990] and aims to general-purpose adaptive hypermedia system. The AHA! architecture so-called AHAM complies with the standards of general architecture of AEHS (see section I.2.3.1). AHAM [De Bra, Houben, Wu 1999] 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”.
- 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 (see next section).
- Within-component layer describes the internal data objects, e.g. the resources (x)html linked to concepts. AHA! accesses these objects through “anchoring”.
-

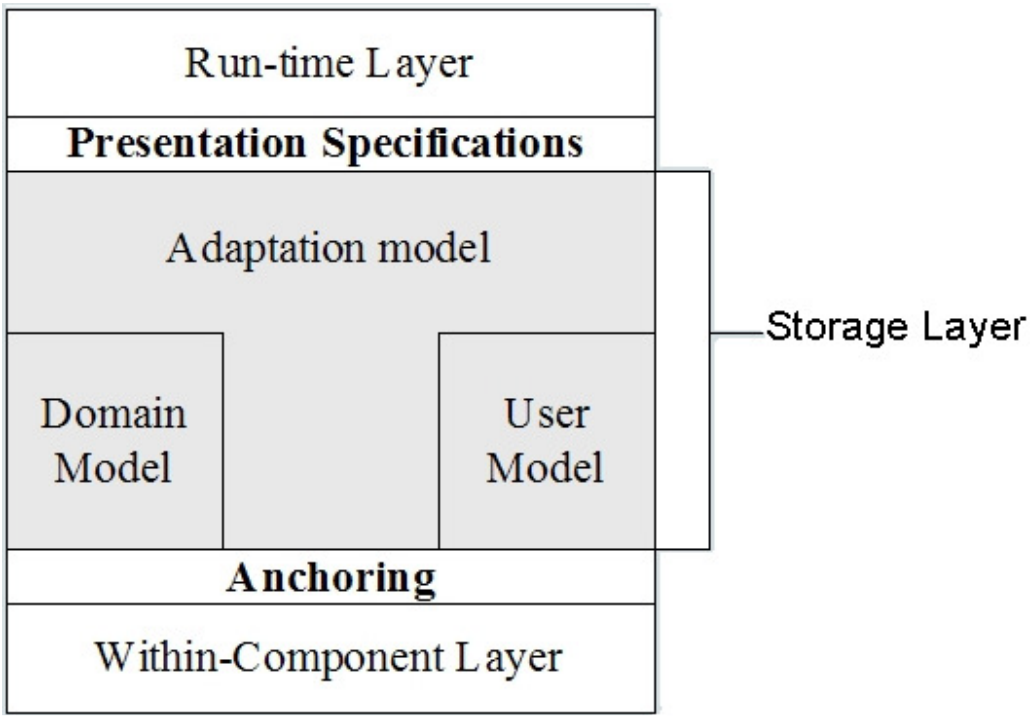


Figure I.2.6. AHAM – The architecture of AHA!

The implemented framework of AHA! [De Bra, Smits, Stash 2006] is constituted of Java web server, connection servlets 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 and send back the HTTP response containing adaptive instructions.

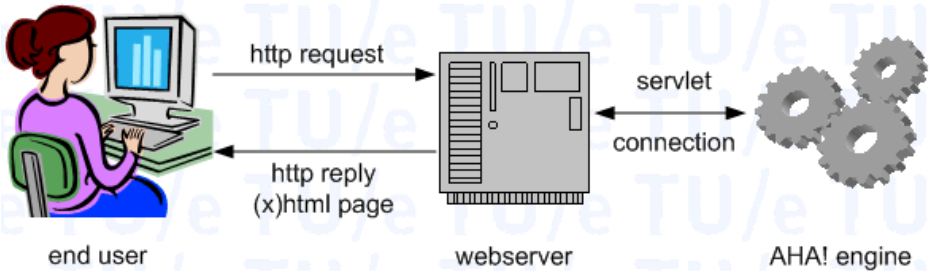


Figure I.2.7. Implemented framework of AHA!

AHA! Engine

The AHA! engine [De Bra, Smits, Stash 2006] manipulates three main models as below:

- Domain model (DM) contains domain concepts associated web learning resources. The relationships between concepts are also specified.
- User model (UM) describes user information. UM is built up by applying overlay modeling approach. Hence, UM is the mastered subset of DM.
- Adaptation model (AM) contains adaptive rules; each rule is associated to a domain concept and used to update UM when executing.

The combination of DM and AM represents a model of the *conceptual* structure of the application. So AHA! engine has responsibility for executing rules, updating user model, filtering retrieved resources.

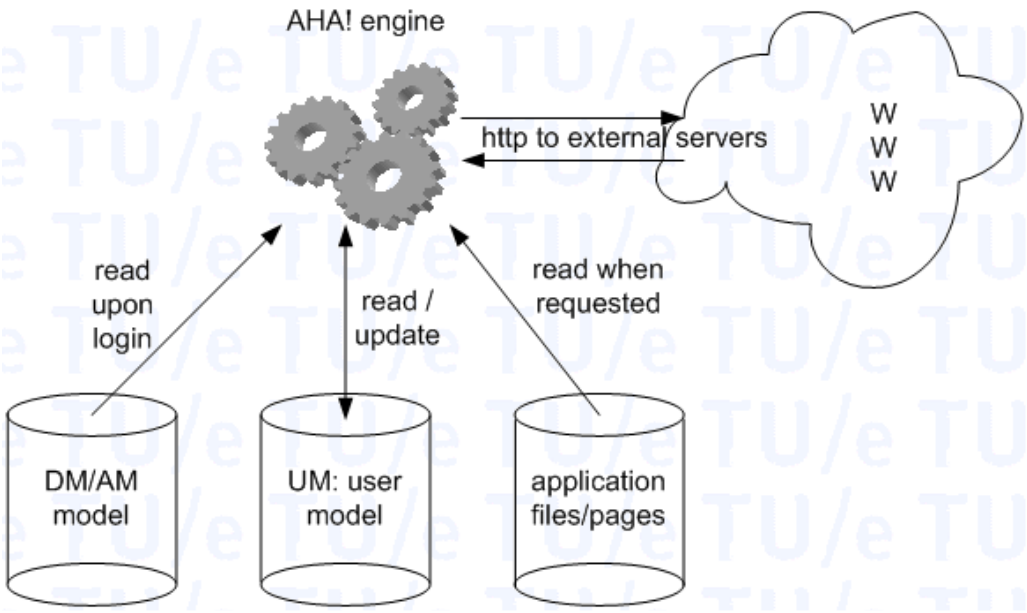


Figure I.2.8. AHA! engine

Moreover, AHA! engine [De Bra, Smits, Stash 2006] also manages application files, resource files in (x)html. DM, UM and AM are store in MySQL database or xml files. The engine manipulates them through three abstract tiers:

- 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

Adaptive procedure is processed by the engine described above. The interaction between user and AHA! happens through HTTP protocol; whenever users send the HHTP 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…) associated this concept.
2. Retrieving UM.
3. Executing adaptive rules (in AM) attached to the request concept. These rules will update attributes of UM (users' characteristics). The executing process spreads over AM since the update can trigger other rules associated updated attributes. So this process continues until no rule is triggered.
4. The resources associated to request concept are retrieved and filtered by adaptive rules in AM so that they are suitable to users (tuned to UM).

I.2.4. Evaluation of existing ITS and AEHS

We already described architecture and basic features of adaptive learning systems in which ITS and AEHS are researched and developed continuously nowadays. When AEHS and ITS are compared together, their architectures are very similar. For example, adaptation component in AEHS "plays the role" of pedagogical module in ITS. However AEHS has some prominent advantages when it makes use of web technology. Hypermedia space in AEHS is more plentiful and convenient with non-linear navigation than knowledge space in ITS. Moreover, the interface in AEHS is more friendly than ITS due to using web page and HTTP protocol 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) focusing on web technology and AI techniques but the ideology of such variants does not go beyond AEHS. So, I don't include them in this chapter.

I.3. Bayesian network user model

The user modeling system proposed in this thesis uses not only Bayesian network but also other techniques such as hidden Markov model, 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 for glancing over the state of the art of Bayesian network model. Bayesian inference 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.

- *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 [Mayo, Mitrovic 2000] system applies this method.
- *Data-centric method*: The structure and conditional probabilities are learned directly from real-world data by learning machine algorithms.

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.

I.3.1. KBS Hyperbook System

According to [Henze, 2000], 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. User knowledge is represented as a knowledge vector in which the i^{th} component of this vector is the conditional probability expresing how user masters the KI_i.

$KV(u) = \{ Pr(KI_1|E), Pr(KI_2|E),..., Pr(KI_n|E) \}$

Where $KI_1, KI_2,..., KI_n$ denotes knowledge items and E is evidence that system observes about user in learning process.

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

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

In each level, there are always *KI* (s) so-called main *KI* (s) that have no parent. Main *KI* (s) in the same level are assumed to be mutually independent.

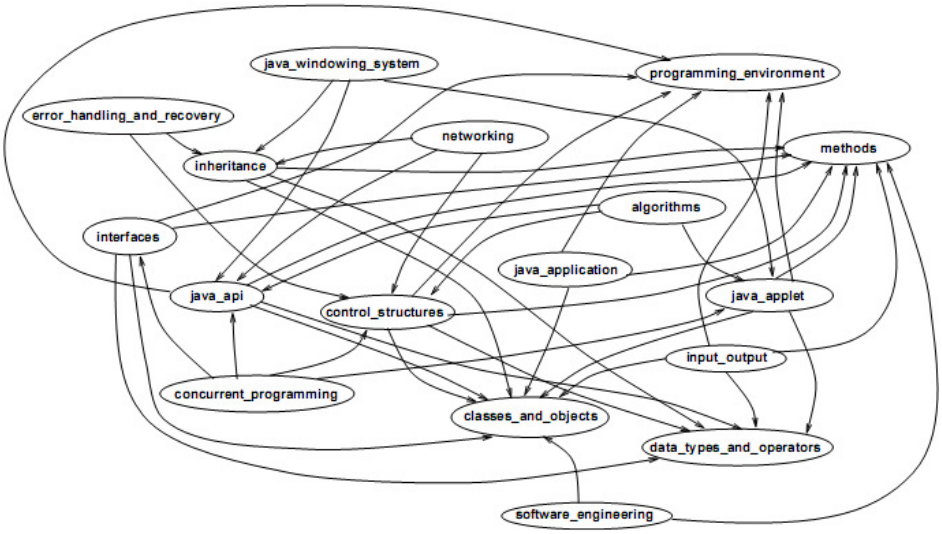


Figure I.3.1. An example of dependency graph

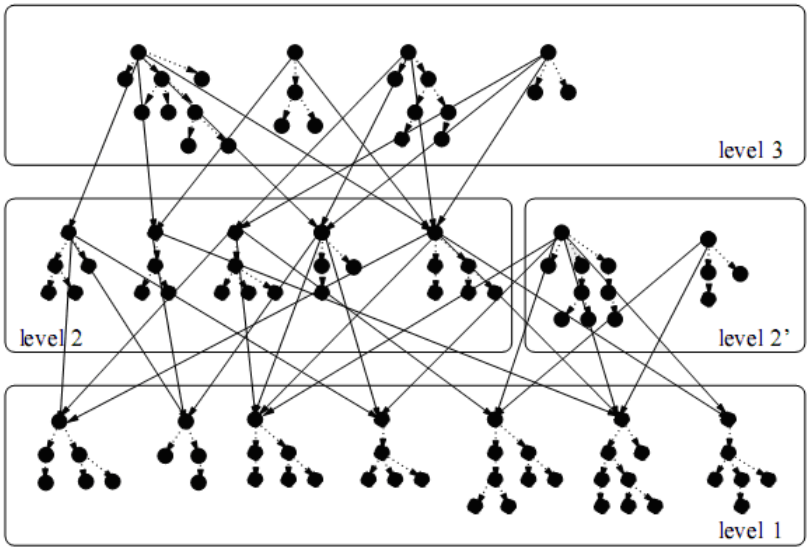


Figure I.3.2. The levels of *KI* (s)

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

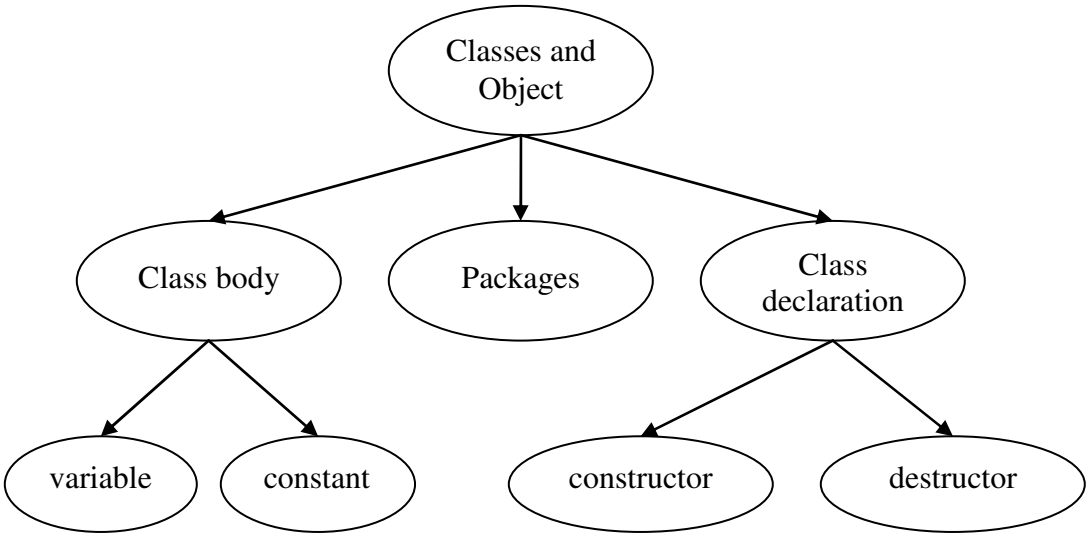


Figure I.3.3. Root tree

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*}:

- *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 don't 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.

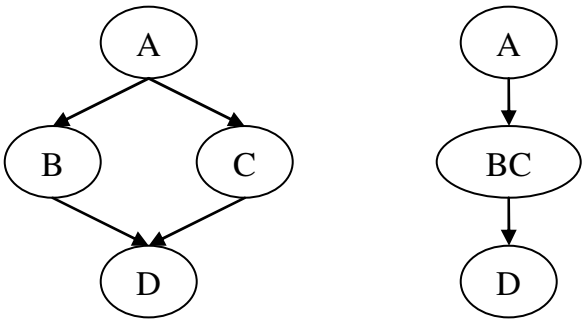


Figure I.3.4. Clustering approach

- *Conditional approach*: The whole network having directed cycles are 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.

I.3.1.1. Yet Another Clustering Formalism (YACF)

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

The additional cluster node [Henze. 2000] is the node that owns income nodes (so-called parent nodes) and outcome nodes (so-called child nodes). The cluster node receives information (maybe evidence) from parent nodes and distributes it to child nodes. This is the information propagation from parents to children. The cluster node is realized as the random cluster variable whose range is the sum of the ranges of all child nodes. One part of the range of cluster variable holds for a particular child node. Thus each child has to listen only to the part of cluster's variable which holds information about it. For example, there are three variables KI_1, KI_2, KI_3 and both KI_1 and KI_2 are parents of KI_3. If KI_1 has value *E* (expert) and KI_2 has value *A* (beginner) then the value of KI_3 is the best grade among values of KI_1 and KI_2; so KI_3 has value *E* (expert). It means that the information is passed from KI_1 to KI_3.

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

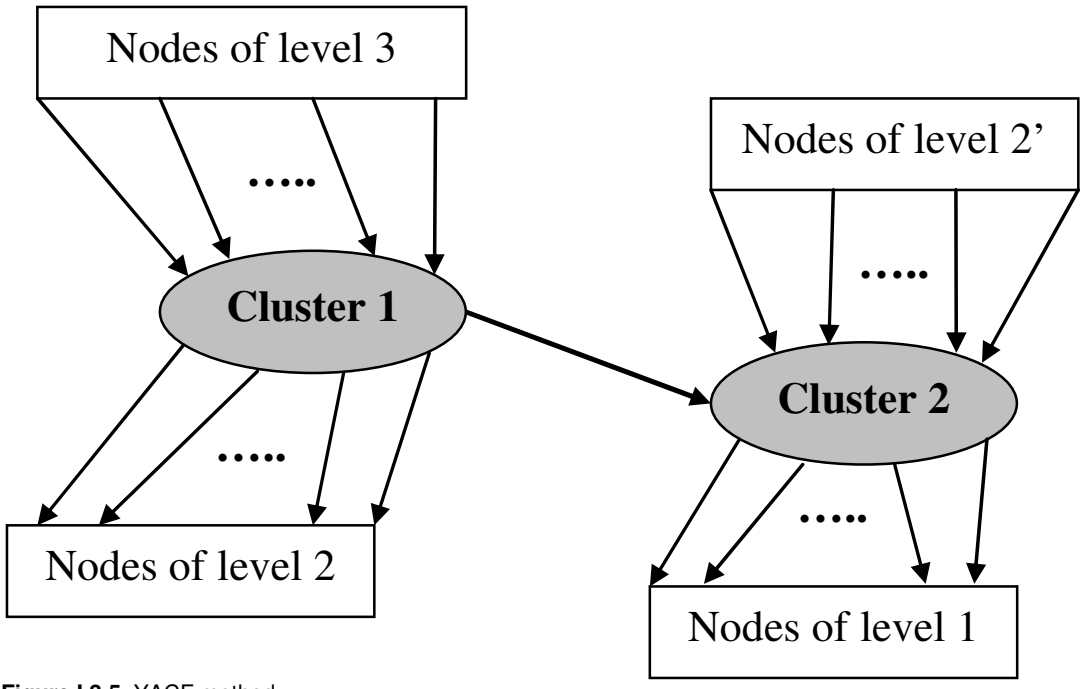


Figure I.3.5. YACF method

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

Suppose $X_1, X_2,\ldots, X_N$ are parent nodes and $Y_1, Y_2,\ldots, Y_M$ are child nodes and *H* is cluster node. Each Y_i, $i = \overline{1,M}$ is depedent from at least one X_k, $k = \overline{1,N}$. Let $1Y_i,\ldots, LY_i$ 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,…, *L*} is the {*E, F, A, N*}.

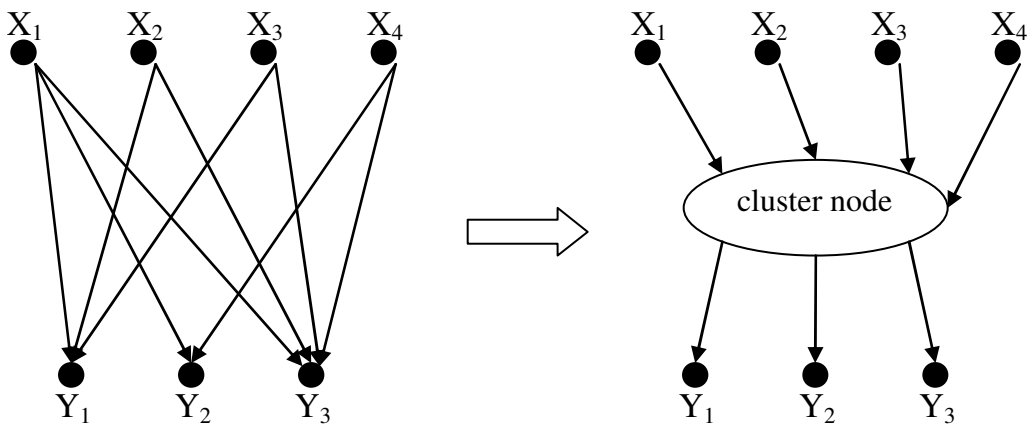


Figure I.3.6. Cluster node

The CPT of cluster node *H* is defined as the matrix $\prod_{l=1}^{N} | R(X_l) | \times \sum_{i=1}^{M} | R(Y_i) |$ where *R*(*X*) denotes the range of parent variable *X*. We have:

$$\Pr(H = kY_i \mid X_1,...,X_N) =$$

$$\begin{cases} \frac{1}{M}, \text{if k = best\_grade}(X_l \text{ (s) on which } \mathrm{Y_i} \text{ is depedent}) \\ 0, \text{else} \end{cases}$$

Where *best_grade* return the maximum value of parent variables X_l on which Y_i is dependent

Table I.3.1. CPT of a YACF-Cluster node

	range of H holding evidence for node Y_1					***range of H holding evidence for node Y_M***			
X_1,……..X_k,……..,X_n	**Pr(H=E_Y_1|…)**		**…Pr(H=N_Y_1)**		**…**	**Pr(H=E_Y_M|…)**		**…Pr(H=N_Y_M)**	
X_1=E,…, X_k=E,…, X_N=E	*1/M*	*0*	*0*	*0*	**…**	*1/M*	*0*	*0*	*0*
.									
.									
X_1=F,…, X_k=N,…, X_N=E	*0*	*1/M*	*0*	*0*	**…**	*1/M*	*0*	*0*	*0*
.									
.									
X_1=N,…, X_k=E,…, X_N=N	*1/M*	*0*	*0*	*0*	**…**	*0*	*0*	*0*	*1/M*
.									
.									

The conditional probability of child node Y_i, $i = \overline{1, M}$ given cluster node H is defined in the following:

$$\Pr(Y_i \mid H = h) = \begin{cases} (\frac{1}{| R(Y_i) |},...,\frac{1}{| R(Y_i) |}), \text{if } h \notin \{1Y_i,...,LY_i\} \\ \Pr(Y_i \mid X = h(Y_i)), \text{else} \end{cases}$$

The conditional probability of child node Y_i given parent nodes $X_1,…, X_N$ is defined as below:

$$\Pr(Y_i \mid X_1,...,X_N) = (M-1)\frac{1}{L \times N} + \frac{1}{N}\Pr(Y_i \mid H = h(Y_i))$$

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

Cluster node (H)	***Pr(Y=E|H=…)***	***Pr(Y=F|H=…)***	***Pr(Y=A|H=…)***	***Pr(Y=N|H=…)***
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**
H_i=A	**0.2**	**0.6**	**0.2**	**0**
H_i=F	**0**	**0.2**	**0.6**	**0.2**
H_i=N	**0**	**0**	**0.2**	**0.8**
H_{i+1}=E	0.25	0.25	0.25	0.25
.	.	.	.	.
.	.	.	.	.
H_n=N	0.25	0.25	0.25	0.25

I.3.2. Andes

The special thing in Andes [Conati, Gertner, VanLehn 2002], the intelligent tutoring system that hepls students to solve physic problems or exercises, is that the Bayesian network is not built up directly from training data or by experts like other systems. For each problem or exercise, the rule-based problem solver generates the data structure so-called *solution graph* which is then converted into Bayesian network. The solution graph is initilized right before student solves a problem.

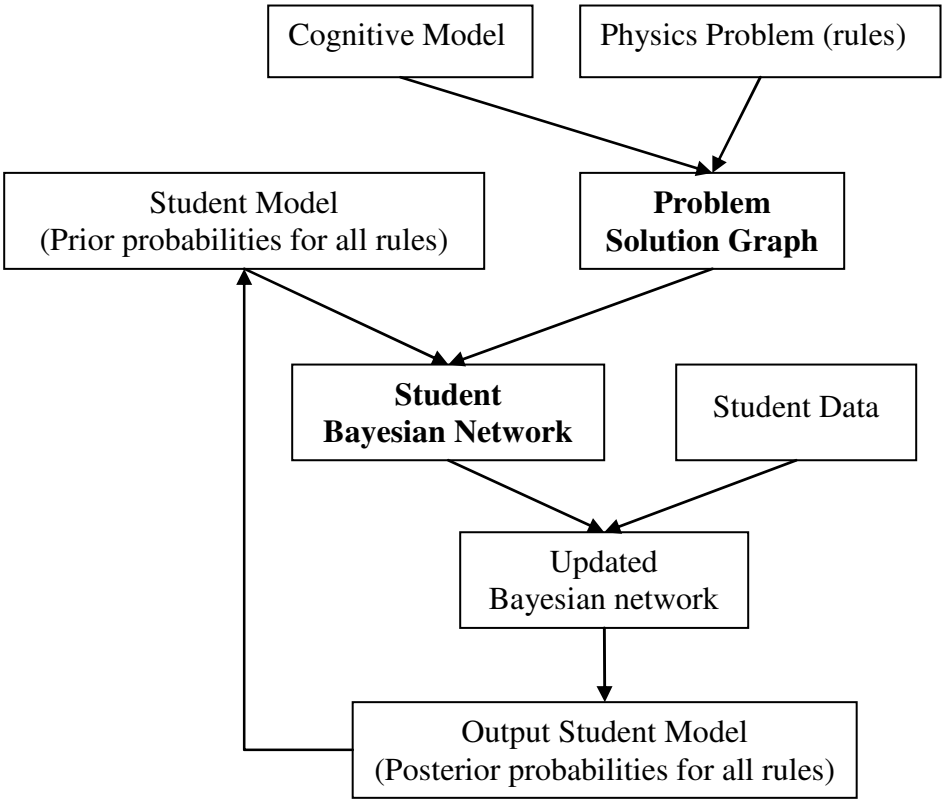


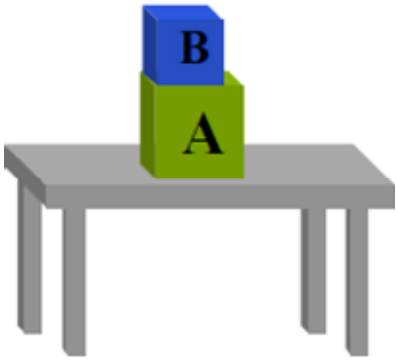
Figure I.3.7. Student modeling in Andes

I.3.2.1. Solution graph

Andes also constructs physics knowledge base including physics rules which are used to encode solution graph. The following are some sample physics rules.

- *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 shown in figure below:



A 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?

Figure I.3.8. Sample problem

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* $F = m\vec{a}$". The solution graph is shown in following figure.

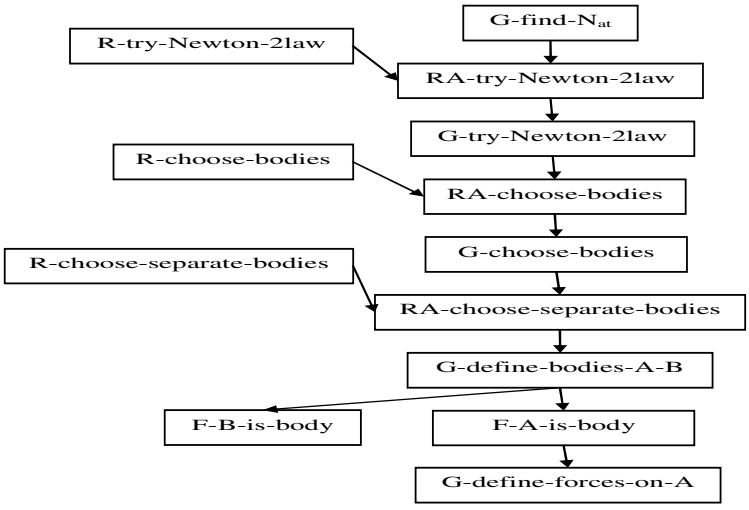


Figure I.3.9. Solution graph

Each node in this graph denotes a particular type of information (goal, rule, rule application, strategy). Namely, 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 [Conati, Gertner, VanLehn 2002] includes two parts: one part so-called *domain-general part* that encodes the domain-general knowledge and another part so-called *task-specific part* that encodes the task-specific knowledge. While domain-general knowledge base includes general concepts and procedures which define the proficiency in domain, task-specific knowledge base represents knowledge related to students' performance on problems and exercises.

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

Domain-general part in Bayesian network

This part models student knowledge, whose nodes are classified into two types: *rule* and *context-rule*. 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. It means that the parent (rule node) represents the general knowledge and the child (context-node) tells us how the student masters such general knowledge in specific context.

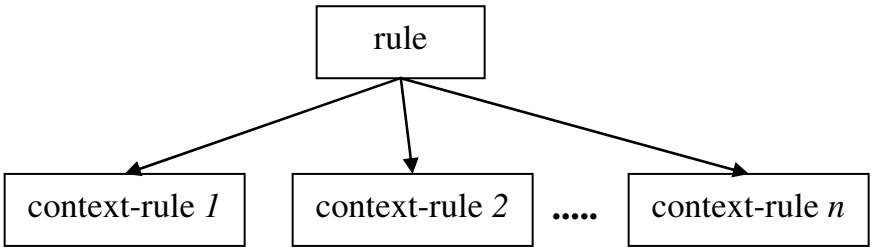


Figure I.3.10. Relationship between rule and context-rule nodes

The conditional probability *Pr*(*context-rule=mastered* | *rule=mastered*) equals *1* because if the student master the general knowledge then she/he can apply it to solve any problems or exercise. The conditional probability *Pr*(*context-rule=mastered* | *rule=not-mastered*) expresses the probability that student solve successfully a problem or exercise even if she/he doesn't master the general knowledge. How to specify the CPT 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 *rule-context* nodes and four other nodes: *fact, goal, rule-application,*

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

Rule-application nodes are responsible for aggregating context-rule nodes, proposition nodes and strategy node so as to derive a new proposition node. One of the parents of a rule-application node must be a context-rule node. It implicates whether students can apply the rule into solving their problem or exercise. The relationship between rule-application node and its parents is *noisy-AND relationship* in which the conditional probability of rule-application node given its parents equals *1-α* only if all such parents gets *true*. The predefined real number *α* is called a "*noise*". If at least one of its parents gets *false*, this conditional probability equals *0*. It means that the student must master all context rules before she/he applies such rules into solving problem or exercise. In case that she/he even knows whole rules, it is possible to assert totally that she/he can apply perfectly rules.

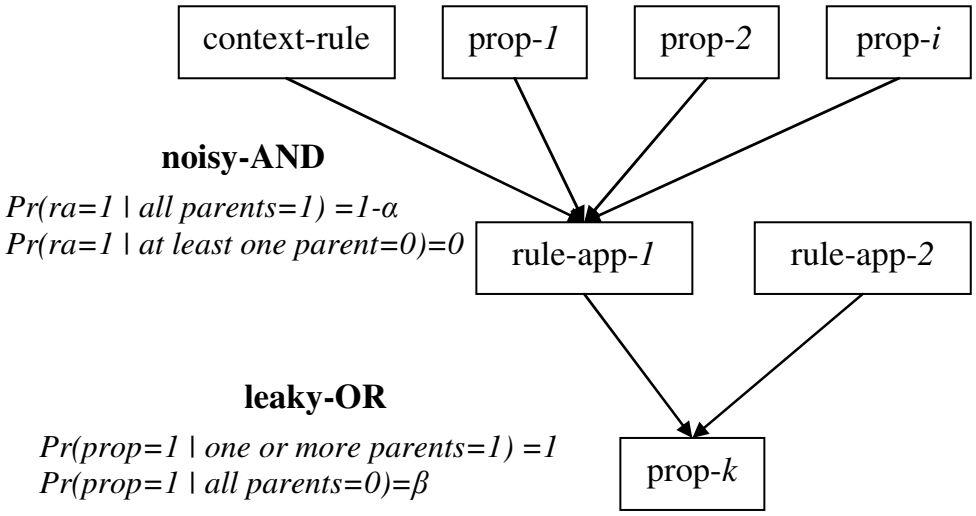


Figure I.3.11. Relationship between nodes in task-specific part

Strategy nodes are used in situation that there are different solutions to a problem. For example, there are two application rules aiming to solve the same goal. When the posterior probability of one application rule lessens the posterior probability of another, it raises the issue so-called mutually exclusive strategy.

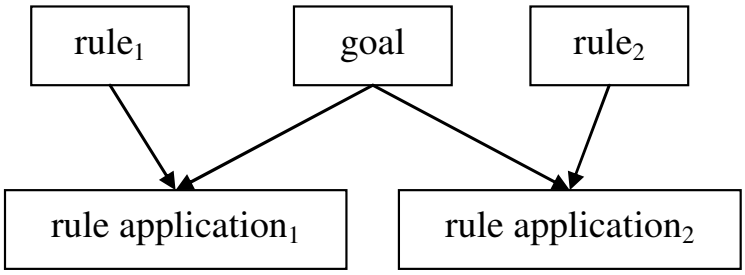


Figure I.3.12. Mutually exclusive strategy

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

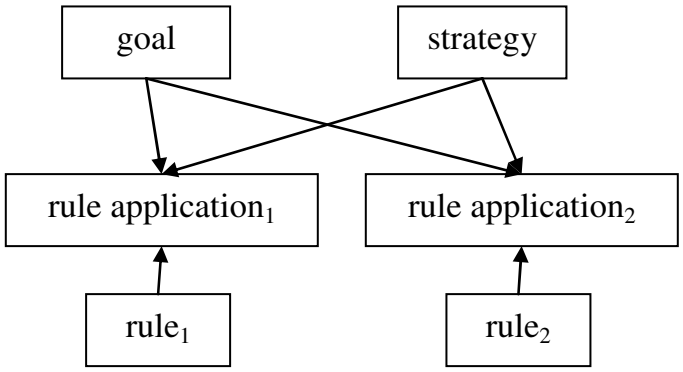


Figure I.3.13. Strategy node

Inference mechanism in Bayesian network

Suppose the student who solves the problem of finding normal force in our example chose block A as body. At that time, the fact node *F-A-is-body* gets value *1* (*true*). When the evidence raised by this fact node is entered, the posterior probabilities of all nodes that derives to evidence become higher and otherwise. The following figure tells us the prior/posterior probabilities of all nodes in the task-specific part (also solution graph) in Bayesian network.

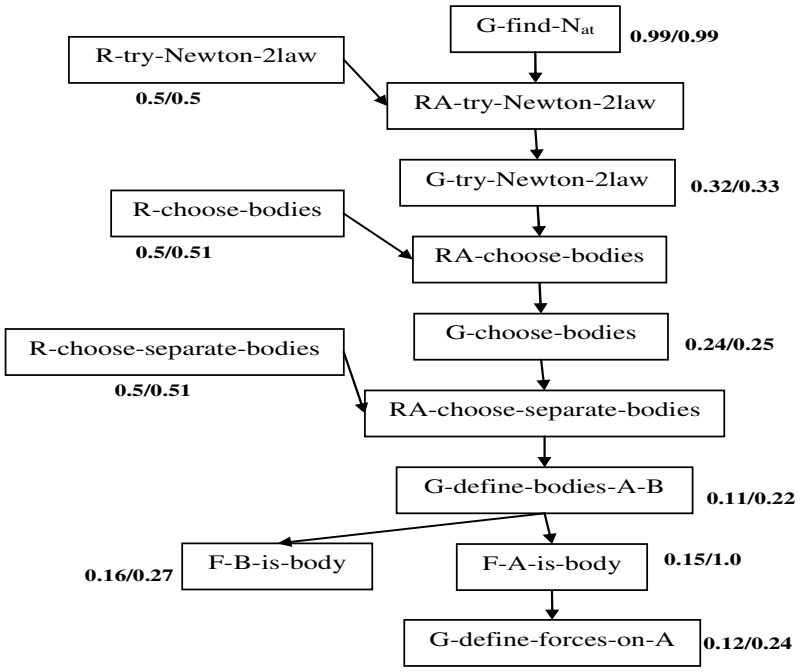


Figure I.3.14. Prior/Posterior probabilities in the task-specific part

I.3.3. SQL-Tutor and Constraint-Based Modeling

Constraint-Based Modeling (CBM)

There are two types of user knowledge: generative and evaluative. Generative knowledge means that user has actual ability about some learning skills. But in the real situation, students may discriminate between the correct and incorrect solution to a problem before they master such problem. This is the evaluative knowledge. Constraint-Based Modeling (CBM) aims to model evaluative knowledge. A constraint is a pair <*Cr*, *Cs*> denoting *relevance condition* and *satisfaction condition*, respectively. 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*" don't. *Cr* defines the class of student's solutions.

Cs is more specific than *Cr* and it defines the correctness of student's solutions. An example for *Cs* is n_1+n_2=*add*(n_1, n_2) where the function *add* is responsible for adding two numbers. So the expression "*1+1=2*" matches this *Cs* but the expression "*3+2=6*" is wrong.

If student's solution is matched with both *Cr* and *Cs*, the constraint <*Cr, Cs*> is *satisfied* for this solution. If only *Cr* matches the solution, we call that the constraint is *relevant* to solution. If both *Cr* and *Cs* don't match this solution, the constraint is *violated*

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

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

Architecture of SQL-Tutor

The SQL-Tutor [Mayo, Mitrovic 2000] 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. The architecture of SQL-Tutor has three functional models: CBM student modeler, pedagogical module and the interface.

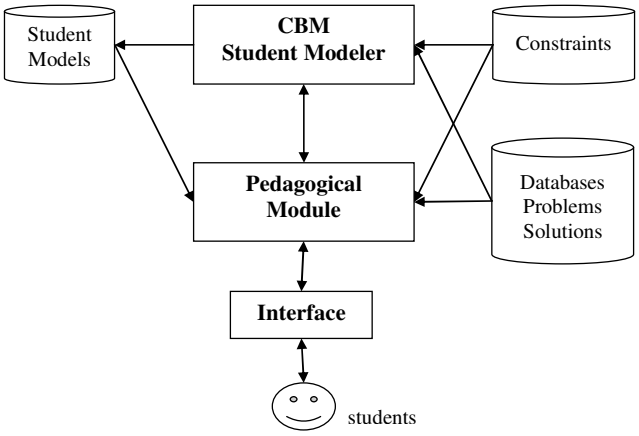


Figure I.3.15. Architecture of SQL-Tutor

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

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

Bayesian network in SQL-Tutor

Now please focus on how to representing student model by probabilistic approach instead of increasing or decreasing student's knowledge and how to apply Bayesian network into SQL-Tutor student model. The student model is constituted of a set of binary variables ($matered_1, matered_2, \ldots, matered_n$) where $matered_c$ ($c=\overline{1,n}$) expresses whether the constraint *c* is mastered ($matered_c=1$) by user or not ($matered_c=0$). $Pr(matered_c=1)$ is the certain probability that student masters constraint *c*. The initial value of $Pr(matered_c=1)$ is the ratio of the frequency that constraint *c* is satisfied to the frequency that constraint *c* is relevant in the past

$$\Pr_0(master_c = 1) = \frac{\text{The frequency that c is satisfied}}{\text{The frequency that c is relevant}}$$

After student solves her/his problem and receives the feedback from pedagogical module, the probability $Pr(matered_c=1)$ is updated according following heuristic rules:

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

Instead of using such rules, SQL-Tutor proposes another method which applies Bayes' rule to update the probability that constraint is mastered. Let *L* denote student's mastery of constraint and let *M* denote the outcome of the last student's solution at this constraint. Both *L* and *M* are binary variables in which *L* takes values *1* (mastered) and *0* (not mastered). *M* takes value *1* (satisfied) and *0* (violated). Suppose the prior probability of student's mastery is $Pr(L)$, the essence of updating such probability is to compute the posterior probability $Pr(L|M)$ when *M* is observed. $Pr(L|M)$ denotes the probability that student masters (doesn't master) the constraint given that this constraint is satisfied (violated).

$$\Pr(M = m \mid L = l) = \frac{\Pr(L = l \mid M = m)\Pr(M = m)}{\Pr(L = l \mid M = 1)\Pr(M = 1) + \Pr(L = l \mid M = 0)\Pr(M = 0)}$$

Where $l, m \in \{0,1\}$ denote values of *L, M*, respectively and $Pr(M|L)$ is the probability that constraint is satisfied (violated) given that student masters (doesn't master) this constraint. $Pr(M|L)$ 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 $matered_c$ 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 $relevant\ SS_{c,p} \in \{0,1\}$ expresses whether constraint *c* is relevant to the student's solution to problem *p*. That variable $relevant\ SS_{c,p}$ 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 $matered_c$. The Bayesian network representing these variables and relationships among them is shown in following figure.

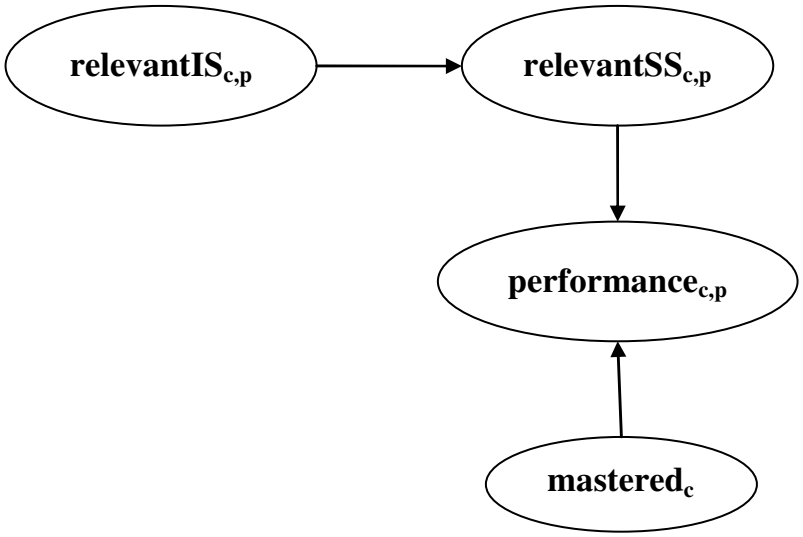


Figure I.3.16. Bayesian network in SQL-Tutor

As discussed, the prior probability of $matered_c$, $Pr_0(matered_c=1)$, is defined by Bayes' rule or heuristic rule. The prior probability of $relevantIS_{c,p}$ is specified by expert. The conditional probability table (CPT) of $relevantSS_{c,p}$ given $relevantIS_{c,p}$ is defined as:

RelevantS$_{c,p}$ \ RelevantIS$_{c,p}$	**yes**	**no**
yes	α_c	β_c
no	$1-\alpha_c$	$1-\beta_c$

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 relevant to ideal solution.

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

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

I.3.4. Data-centric approach

[Cheng, Bell, Liu 1997a] 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 named mutual information. The higher the mutual information of two nodes, the more the correlation between them is, in other words, the more likely there is an arc connecting them. The mutual information of two nodes X, Y is defined as:

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

And the conditional mutual information is defined as:

$$I(X,Y \mid C) = \sum_{x,y,c} \Pr(x, y, c) \log \frac{\Pr(x, y \mid c)}{\Pr(x \mid c) \Pr(y \mid c)}$$

Where *C* is a set of nodes

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 algorithm [Cheng, Bell, Liu 1997a] for learning the structure of Bayesian network includes three phases: *drafting*, *thickening* and *thinning*. However, before discussing about this algorithm, we should 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 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*, 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 network. It means that maybe there is lack of some arcs in networks.

The second phase, *thickening phase*, 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* includes two steps:

- Firstly, for each pair of adjacent nodes (*X,Y*), we remove temporarily the arc connecting them.
- Secondly, we try 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.

Zebra: A User Modeling System for Triangular Learner Model

The core of adaptive system aforementioned in 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). There are two main tendencies towards implementing UMS: domain-dependent UMS and domain-independent UMS. The latter is called generic UMS known widely but our 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. We 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 in section II.1. Section II.2 described the Zebra – our modeling system in detailed. Section II.3 is the conclusion.

II.1. Existing user modeling systems

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

II.1.1. 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 [Rich 1979] developed by Rich in his 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… are represented as user model. GRUNDY use stereotype method [Rich 1979] to build up user model, based on users' answers to questions during their first usage of system. For example, if user has a male first name, GRUNDY infers a high sex tolerance and a low one for romance.

II.1.2. User modeling shells

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

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

GUMS

GUMS [Finin 1986] is the abbreviation of "General User Modeling System" developed by Tim Finin in his PhD thesis. 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. 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 1).

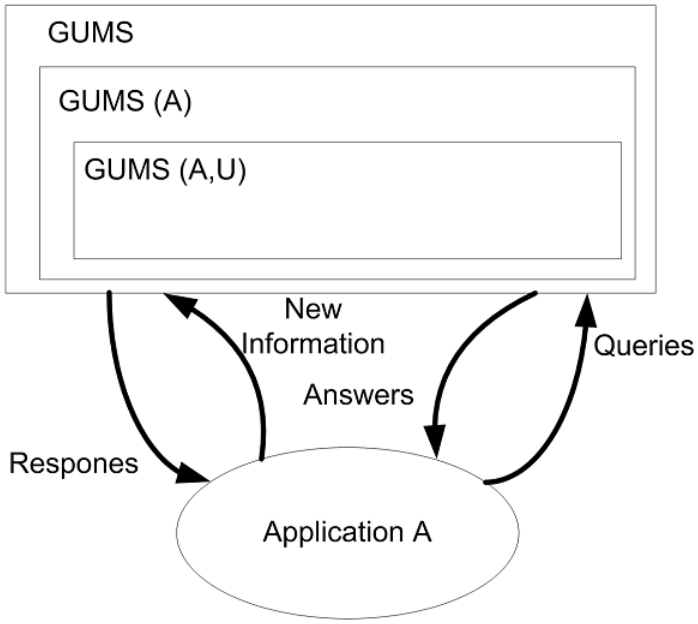


Figure II.1. The architecture of GUMS

UMT

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

PROTUM

PROTUM (PROlog based Tool for User Modeling) [Vergara 1994] is written by the programming language PROLOG. It 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. Moreover PROTUM determines the activation rates of triggers in order to activate or deactivate stereotypes which have already assigned to user. These rates are used to resolve conflicts between inconsistent assumptions of two activated stereotypes.

TAGUS

TAGUS developed by Paiva and Self [Paiva, Self 1994] 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, goal… TAGUS also has inference mechanism and truth maintenance discovering contradictions of assumptions and diagnosing unexpected user's behaviors.

Um

um developed by [Kay 1995] aims to provide the library of user modeling functionalities in which assumptions about users' knowledge, goal, background… 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 five types of components: *observation*, *stereotype activation*, *rule invocation*, *user input* and *told to the user*. So um combines stereotype method and rule-based method in implementation.

II.1.3. User modeling servers

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

BGP-MS

BGP-MS developed by [Kobsa, Pohl 1995] is the user modeling server taking interest in user's knowledge, belief and goal. It receives user's observations provided by adaptive application and processes internal operations of classification and calculation based on these observations. BGP-MS uses stereotype method, natural language dialogs and questionnaires to build up user model. BGP-MS has four essential components:

- *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 SB-ONE [Kobsa 1991] knowledge-representation tool.
- *Automatic stereotype management* is responsible for the activation and deactivation of assigned stereotypes of an individual user model

These main components communicate together through the *functional interface*. Developer interacts with BGB-MS by the *graphic interface*. The architecture of BGB-MS is represented in following figure.

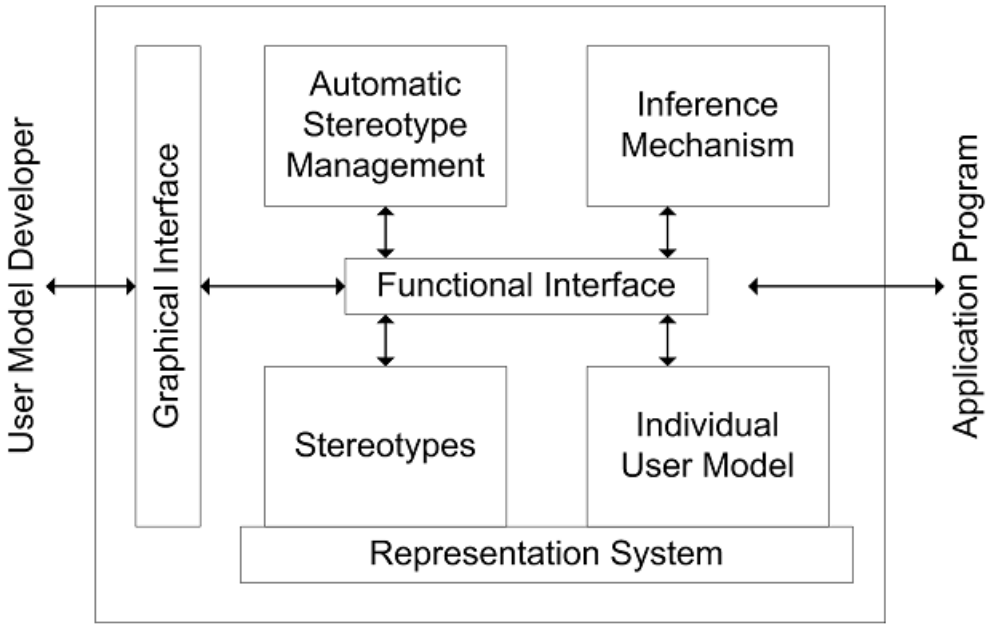


Figure II.2. The architecture of BGP-MS

See in figure II.2, the adaptive system communicates with main components through functional interface and BGP-MS defines the new communication protocol called KN-IPCMS (KoNstanz Inter-Process Communication Management System) [Kobsa, Pohl 1995].

Doppelgänger

Doppelgänger developed by [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.

- *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
- *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.

CUMMULATE

CUMMULATE [Brusilovsky 2004] is developed as 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. Such actions are considered events. CUMMULATE adds a timestamp to each event and stores it permanently in event storage. The event storage allows several *inference agents* that process events in different ways and convert these events into the inferenced user model, for example, some agents monitor user's knowledge and others predict user's interests. The architecture of CUMMULATE is open to a variety of *external inference agents* that receive requests from event storage and manipulate user model. Figure 3 show its architecture.

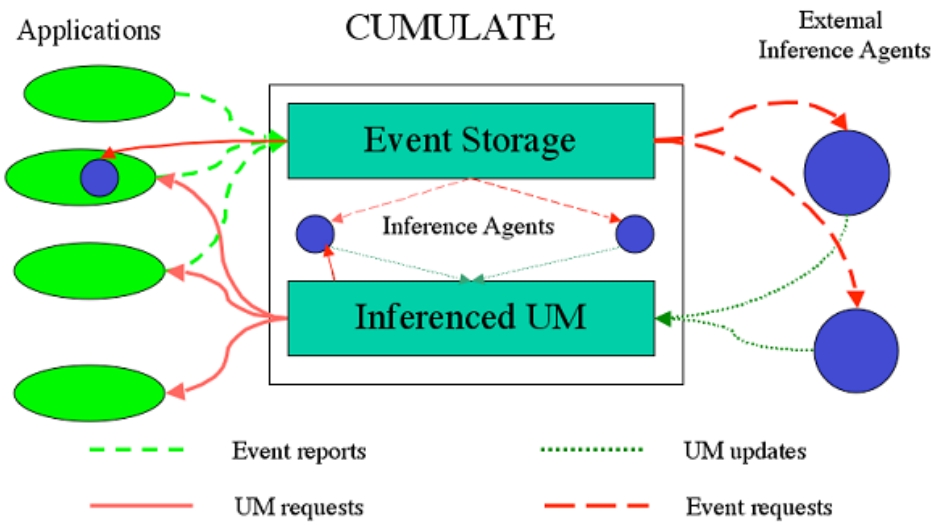


Figure II.3. The architecture of CUMMULATE

Personis

Personis [Kay 2002] is the modeling server whose considerable feature is to allow user to control and scrutinize his/her model. Such user model is called scrutable user model. Personis is based on um toolkit but more complicated than um toolkit. The architecture of Personis is divided into four parts:

- The *server itself* is responsible for managing user model.
- A set of *generic scrutiny tools* allow users to see and control their own user models. Users interact with these tools through a generic scrutiny interface. This interface is application-independent.
- A set of *Adaptive Hypermedia Application* (AHA) are denoted AHA_1, AHA_2,…, AHA_n. Each AHA includes two parts: first, the core enables user to do some learning tasks and second, the scrutiny interface associated to the core enables the core to interact with scrutiny tools. This interface is similar to generic scrutiny interface except that it belongs to the context of AHA.
- A set of *views* in which each view of user model available to each AHA is responsible for defining the components used by such AHA. Applications can use either the same or different views.
-

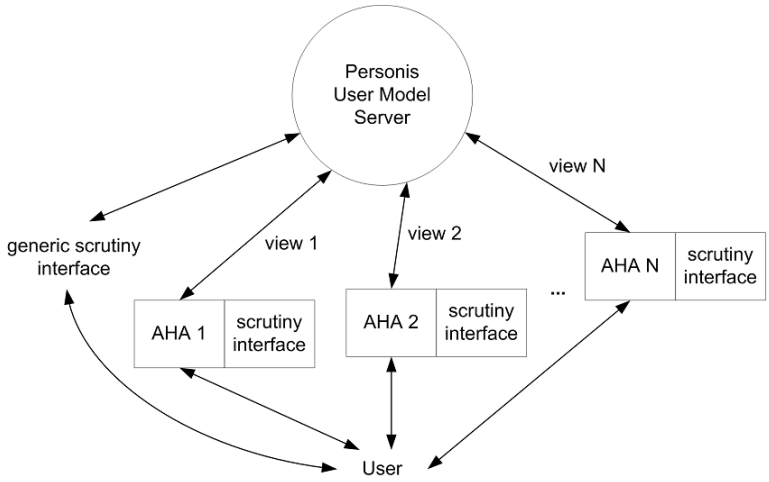


Figure II.4. The architecture of Personis

LDAP-UMS

LDAP-UMS [Kobsa, Fink 2006] is the user modeling severs based on Lightweight Directory Access Protocol (LDAP). 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. 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*.

- *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, inferring new assumptions about user…The *Directory Component* and *User Modeling Components* interact together via CORBA 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

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. Following figure shows the architecture of LDAP-UMS.

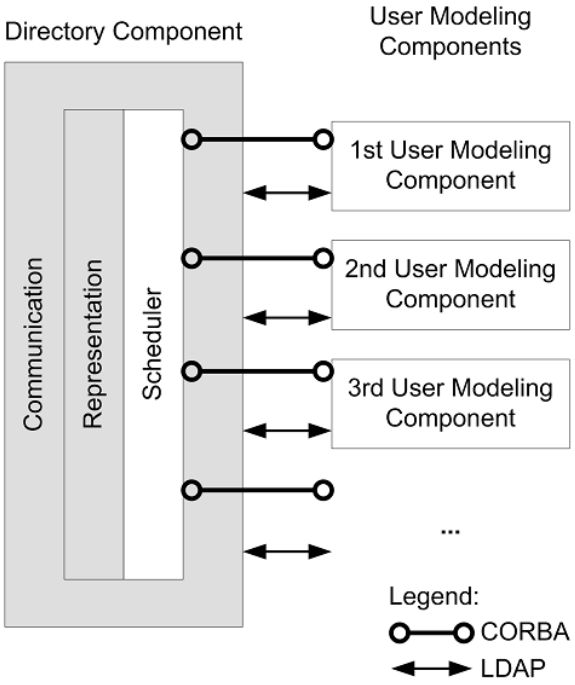


Figure II.5. The architecture of LDAP-UMS

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

Exist user modeling systems 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) 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) prove 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, demographic information, etc. Each kind of information is in accordance with respective modeling method. For example, knowledge model is often created by overlay method, which called overlay model. So it is impossible to model all learners' characteristics because of such reason "there is no modeling method fit all characteristics". Moreover, almost UMS (s) require effective inference techniques in their modeling tasks but this is impossible if we cannot recognize which individual characteristics are important.

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

- Knowledge, learning styles and learning history are prerequisite for modeling learner.
- While learning history and knowledge change themselves frequently, learning styles are relatively stable. The combination of them ensures the integrity of information about learner.
- User'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 our UMS will run fast and be powerful like African zebra.

II.2.1. Triangular Learner Model

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

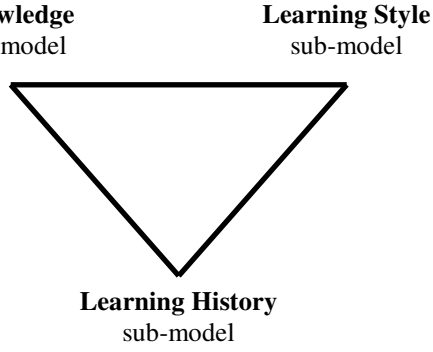


Figure II.6. Triangular Learner Model

I propose the method that combines overlay method and Bayesian network [Charniak 1991] to build up knowledge. 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 is the directed acyclic graph (DAG) in which nodes are linked together by arcs; each arc expresses the dependence relationships between nodes. The

strengths of dependences are quantified by Conditional Probability Table (CPT). The combination between overlay model and BN 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 prerequisite 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.

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. There are many models of learning styles in theory of psychology such as Dunn and Dunn, Witkin, Riding, Myers-Briggs, Kolb, Honey-Mumford, Felder-Silverman, etc. I choose Honey-Mumford [Honey, Mumford 1992] and Felder-Silverman model [Felder, Silverman 1988] as principal models which are presented by Hidden Markov Model (HMM) [Dugad, Desai 1996]. According to Honey-Mumford 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, assimilate different facts into coherent theory. Pragmatists have practical mind, prefer to try and test techniques relevant to problems.

For modeling learning style using 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.

The last sub-model stores and manipulates learner's **learning history** in form of XML data files. *All learners' actions: learning materials access, duration of computer use, doing exercise, taking an examination, doing test, communicating with teachers or classmates… are logged in this 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, background, goals, etc. The mining engine in the core of Zebra often uses this sub-model for many mining tasks. For this reason, this sub-model is drawn as the apex at the bottom of triangle in architecture of TLM. This implicates that learning history sub-model is the most important sub-model in TLM when it is considered as the basic of two other sub-models. Figure II.7 shows the extended TLM in which learning history sub-model is the root for attaching more learners' characteristics such as interests, background, goals… to TLM.

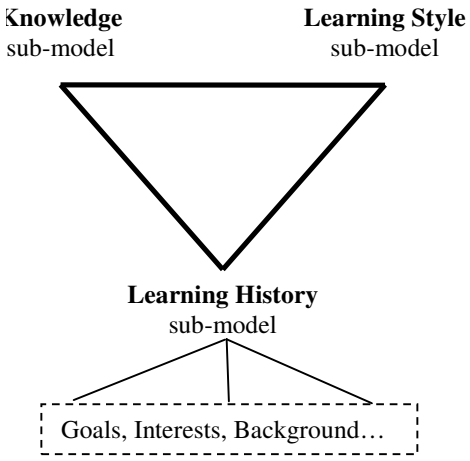


Figure II.7. extended Triangular Learner Model

II.2.2. The architecture of Zebra

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

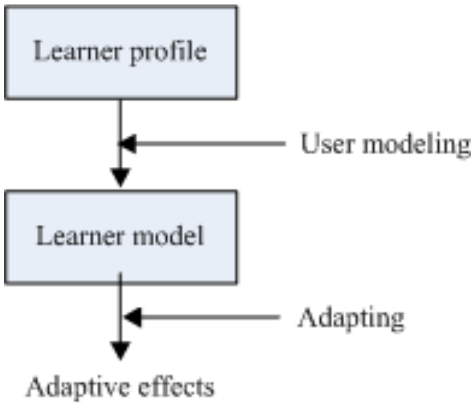


Figure II.8. Modeling tasks are similar to profile mining tasks.(*Note that user is considered as learner in e-learning context*)

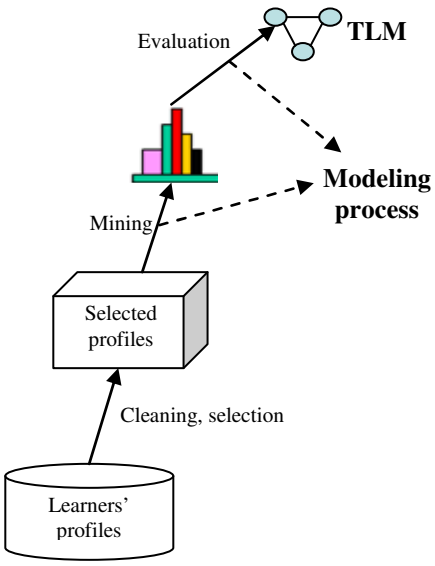


Figure II.9. The mining process in Zebra

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

evidences. Zebra must have another inside engine – the *belief network engine*.

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

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

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

Zebra provides *communication interfaces* (**CI**) that allows users and adaptive systems to see or modify restrictedly their TLM. Adaptive applications also interact with Zebra by these interfaces. *Communication interfaces* are implemented as web services used widely on internet. According to World Wide Web Consortium, a web service is defined as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (especially WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. When complying with web service standard, it is possible to publish CI (s) on internet for third-parties to communicate with Zebra more effectively.

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

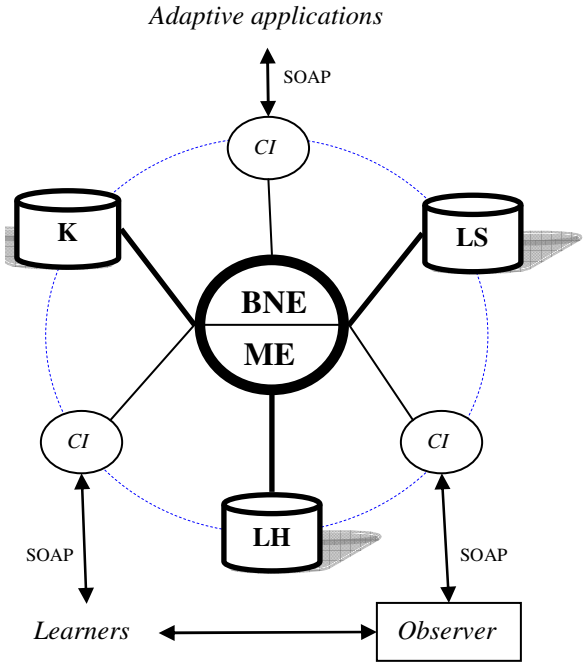


Figure II.10. The architecture of Zebra(*ME and BNE denote mining engine and belief network engine respectively. LH, K and LS denote Learning History sub-model, Knowledge sub-model and Learning Styles respectively. CI denotes communication interface*)

II.2.3. The interaction between Zebra and adaptive applications

Zebra aims to support adaptive learning applications; so in this section we should glance over what adaptive applications are and discuss about how Zebra interacts with such applications. The most popular adaptive learning system supporting personalized learning environment is **AES** (**A**daptive **E**ducation **S**ystem). Here, AES is regarded as an example for illustrating prominent traits of adaptive learning system. The origin architecture of AES has four components:

- *Resource component*: contains learning resources (lecture, test, example, exercise…) and associated descriptive information (metadata).
- *Domain component*: constitutes the structure of domain knowledge so-called domain model. Domain model was often shown in the form of graph.
- *Adaptation component*: is the centric component which gives effect to adaptation. It contains *content selection rules* and *concept selection rules*. We apply content selection rules into choosing suitable educational resources from resource domain (component). On the other hand, concept selection rules are used to choose appropriate concept from domain. These rules must obey user model so that the selection gets correct.
- *User Model*: information and data about user.

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

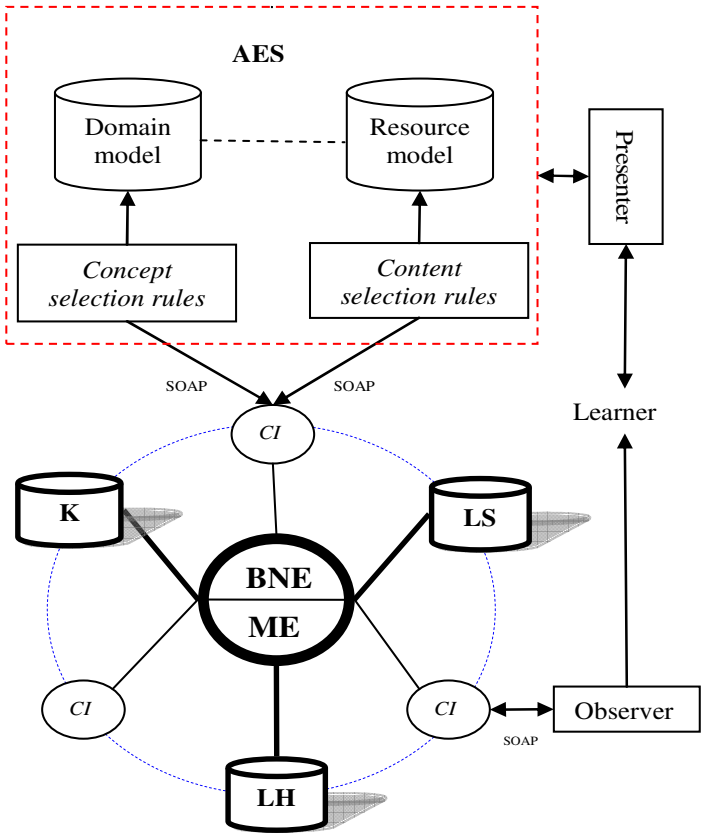


Figure II.11. The new architecture of AES and the interaction between AES and Zebra

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

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

The adaptation process includes following steps:

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

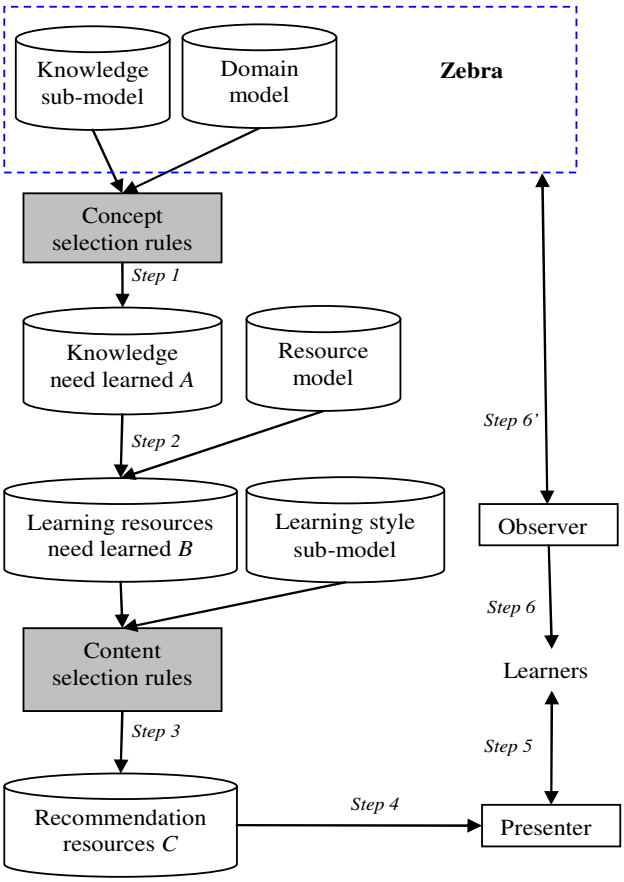


Figure II.12. Steps in adaptation process

CONCLUSION

In this chapter, the "Triangular Learner Model (TLM)" composed by three underlying characteristics: knowledge, learning style and learning history aims to cover the whole of learner's information required by learning adaptation process. Hence TLM has three sub-models: knowledge sub-model, learning style sub-model and learning history sub-model which are considered as three apexes of a triangle. Next chapter is the comprehensive description of these sub-models. UMS which builds up and manipulates TLM so-called Zebra includes the core engine and a set of communication interfaces. The core engine is the composition of two sub-engines: mining engine and belief network engine. Mining engine applies data mining algorithms into discovering and structuring TLM. Belief network engine is responsible for inferring new personal traits from TLM by using deduction mechanism available in belief network. Communication interfaces allow users to see or modify restrictedly their TLM and adaptive applications also interact with Zebra through them. I also propose the new architecture of Adaptive Education System (AES) that interacts with Zebra.

Three sub-models in Triangular Learner Model

The previous chapter gives us a general architecture of Triangular Learner Model (TLM) and its user modeling system Zebra. TLM is composed of three sub-models such as knowledge sub-model, learning style sub-model and learning history sub-model. This chapter is the most detailed one that describes thoroughly these sub-models together with evaluations on each of them.

III.1. Knowledge sub-model

The core of adaptive system is user model containing personal information such as knowledge, learning styles, goals which are requisite for learning personalized process. There are many modeling approaches, for example: stereotype, overlay, plan recognition but they don't bring out the solid method for reasoning from user model. This thesis 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). The strong point of this model is the inference mechanism when it is constructed in form of Bayesian network; so it has ability to predict and assess user knowledge through observing their actions like: accessing lectures, doing exercise, doing test, etc. Assessing user knowledge is not easy in traditional education that teacher and student teach and learn face-to-face. In the complex teaching curriculum, almost teachers can't assess user knowledge unless they use powerful math tools which are implemented in this sub-model. This section has five main parts:

1. Combination of Bayesian network and overlay model in building up knowledge sub-model
2. Incorporating Bayesian inference into adaptation rules
3. Evolution of Bayesian overlay model
4. Improving knowledge sub-model by using Dynamic Bayesian network
5. Specifying prior probabilities

The first section III.1.1 is the most important; it describes in detailed how to construct knowledge sub-model by applying Bayesian network. Section III.1.2 discusses about how to take advantage of knowledge sub-model so as to perform adaptive learning tasks. Sections III.1.3 and III.1.4 discusses two approaches to improve Bayesian network: parameter learning by Expectation Maximization (EM) algorithm and using Dynamic Bayesian network to model temporal knowledge. Section III.1.5 is an extra part which mentions how to specify prior probabilities more accurately so as to enhance the inference process in Bayesian network. Knowledge sub-model is managed by belief network engine (BNE).

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

User modeling is the new trend of enhancing the adaptability of e-learning system. User models are classified into: stereotype model, overlay model, differential model, perturbation model, and plan model.

- Stereotype [Rich 1979] is a set of user's frequent characteristics. In general, stereotype represents a category or group of learners.
- In overlay modeling, 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 is basically an overlay on expected knowledge, which in turn is an overlay on expert's domain knowledge.
- Perturbation model represents learners as the subset of expert's knowledge plus their mal-knowledge.

Modeling user must follow three below steps:

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

Reasoning is complex but essential and interesting, especially, there is need to deal with uncertain or imprecise information in user modeling. For example, answering the question: "The student failed the exam, so most probably he doesn't master the knowledge" is involved in processing uncertain information. The approaches which solve this problem primarily base on theory of artificial intelligence (AI) or statistics. Both AI and statistics have particular advantages and drawbacks but statistical method is appropriate to evaluate learner's performance by collecting evidences. Bayesian network which is the marriage between Bayesian inference and graph theory has a solid mathematical fundamental. Additionally, overlay model can represent very clearly user's knowledge. In this section, I propose the combination of overlay model and Bayesian network so that it is able to take full advantages of strong points of both of them. First, survey of Bayesian inference and Bayesian network is discussed in III.1.1.1. After that main method of applying Bayesian network to overlay model, the core of our method, is described in III.1.1.2. Section III.1.1.3 is the proof of additional probability formula that specifies Bayesian network parameters. Evaluation of combination between Bayesian network and overlay model is mentioned in III.1.1.4. The sub-model created by combining overlay model and Bayesian network is called **Bayesian overlay model** or Bayesian model in brief.

III.1.1.1. Bayesian network

Bayesian rule

Bayesian inference, a form of statistical method, is responsible for collecting evidences to change the current belief in given hypothesis. The more evidences are observed, the higher degree of belief in hypothesis is. First, this belief was assigned an initial probability. 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 Bayesian rule with some special aspects:

$$\Pr(H \mid E) = \frac{\Pr(E \mid H) * \Pr(H)}{\Pr(E)} \quad (\textit{Formula III.1.1})$$

- *H* is probability variable denoting a hypothesis existing before evidence.
- *E* is also probability variable denoting an observed evidence.
- *Pr*(*H*) is prior probability of hypothesis. It is also hypothesis' initial value.
- *Pr*(*H* /*E*), conditional probability of *H* with given *E*, is called posterior probability. It tells us the changed belief in hypothesis when occurring evidence.
- *Pr*(*E* /*H*) is conditional probability of occurring evidence *E* when hypothesis was true. In fact, likelihood ratio is *Pr*(*E* | *H*) / *Pr*(*E*) but *Pr*(*E*) is constant value. So we can consider *Pr*(*E* | *H*) as likelihood function of *H* with fixed *E*.
- *Pr*(*E*) is probability of occurring evidence *E* together all mutually exclusive cases of hypothesis. If *H* and *E* are discrete, $\Pr(E) = \sum_{H} \Pr(E \mid H) * \Pr(H)$, otherwise $f(e) = \int f(e \mid h) f(h) dh$ with *h* and *e* being continuous, *f* denoting probability density function. Because of being sum of products of prior probability and likelihood function, *Pr*(*E*) is called marginal probability.

Note: *H*, *E* must be random variables according to statistical theory.

Bayesian network

Bayesian network (BN) [Neapolitan 2003] 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). Each node represents a random variable which can be an evidence or hypothesis in Bayesian inference. Each arc reveals the cause-effect 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). If variables are discrete, CPD is simplified as conditional probability table (CPT). 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.

E.g., in figure III.1.1, event "cloudy" is cause of event "rain" which in turn is cause of "grass is wet" [Pearl 1988] [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.

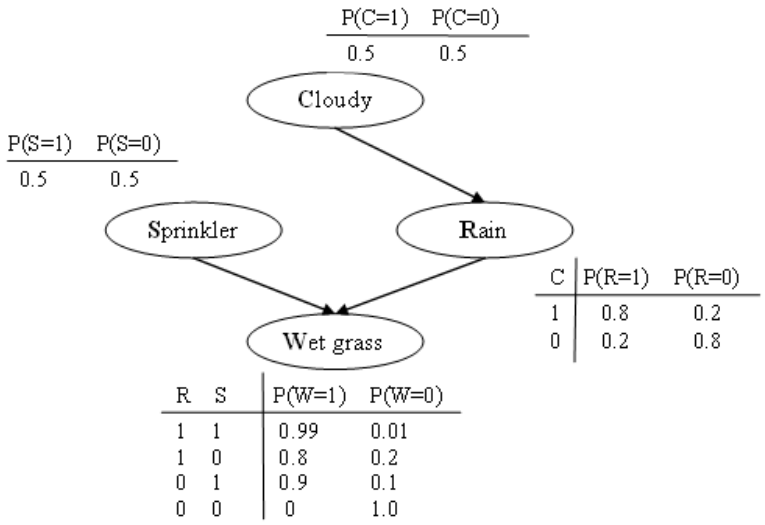


Figure III.1.1. Bayesian network (a classic example about "wet grass")

Suppose we use two letters x_i and $pa(x_i)$ to name a node and a set of its parent, correspondingly. Let X be vector which was constituted of all x_i, $X = (x_1, x_2,..., x_n)$. The **G**lobal **J**oint **P**robability **D**istribution (GJPD) $Pr(X)$ being product of all local CPDs or CPT(s) is formulated as:

$$\Pr(X) = \Pr(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} \Pr(x_i \mid pa(x_i)) \quad (\textit{Formula III.1.2})$$

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

$$\Pr(X) = \Pr(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} \Pr(x_i \mid \Omega_i) \quad (\textit{Formula III.1.2'})$$

Note that $Pr(x_i \mid \Omega_i)$ is the CPT of x_i. According to Bayesian rule, given E the posterior probability of variables x_i is computed as below:

$$\Pr(x_i \mid E) = \frac{\Pr(E \mid x_i) * \Pr(x_i)}{\Pr(E)} \quad (\textit{Formula III.1.3})$$

Where $Pr(x_i \mid E)$ is *prior probability* of random variable x_i and $Pr(E|x_i)$ is conditional probability of occurring E when x_i was true and $Pr(E)$ is probability of occurring E together all mutually exclusive cases of X. Applying (1) into (2) we have:

$$\Pr(x_i \mid E) = \frac{\sum_{X/\{x_i \cup E\}} \Pr(x_1, x_2, ..., x_n)}{\sum_{X/E} \Pr(x_1, x_2, ..., x_n)} \quad \text{(Formula III.1.3')}$$

In figure III.1.1, according to formula III.1.2':

$$\Pr(C, R, S, W) = \Pr(C) * \Pr(R \mid C) * \Pr(S \mid C) * \Pr(W \mid C, R, S)$$

Applying formula III.1.2', $Pr(S \mid C) = Pr(S)$ due to the conditional independence assertion about variables S and C. Furthermore, because S is intermediate node between C and W, we should remove C from $\Pr(W \mid C, R, S)$, hence $Pr(W \mid C, R, S) = Pr(W \mid R, S)$. In short, applying formula III.1.2' we have:

$$\Pr(C, R, S, W) = \Pr(C) * \Pr(S) * \Pr(R \mid C) * \Pr(W \mid R, S) \quad \text{(Formula III.1.4)}$$

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 Bayesian network.

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

$$\Pr(R = 1 \mid W = 1) = \frac{\sum_{C,S} \Pr(C, R = 1, S, W = 1)}{\sum_{C,R,S} \Pr(C, R, S, W = 1} \quad \text{(Formula III.1.5)}$$

$$\Pr(S = 1 \mid W = 1) = \frac{\sum_{C,R} \Pr(C, R, S = 1, W = 1)}{\sum_{C,R,S} \Pr(C, R, S, W = 1} \quad \text{(Formula III.1.6)}$$

In fact, formulas III.5 and III.6 are expansion of formula III.1.1. Applying (4) to (5) & (6), we have:

Pr(R=1|W=1)=0.4475/0.7695=0.581< Pr(S=1|W=1)=0.4725/0.7695=0.614

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

III.1.1.2. Applying Bayesian network to overlay model

The basic idea of overlay modeling is that the user model is the subset of domain model. Straightforward, the domain is decomposed into a set of knowledge elements and the overlay model (namely, user model) is simply a set of masteries over those elements. Suppose that the mastery of each element varies from *0* (not mastered) to *1* (mastered), which is considered as the weight of such element. The relationship of element A to element B is often prerequisite relationship, so, we can deduce that user must comprehend A before learning B. 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.

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:

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 prerequisite relationships between knowledge elements are known as the conditional dependence assertions in BN. Accordingly, each node has a CPT.

3. All knowledge elements are defined as hidden variables (hypothesis). Other learning objects or events (such as tests, exams, exercises, user's feedback, user's activities) which are used to assess or evaluate user's performance in learning process are consider as evidence variables. We must add them to Bayesian network along with determining the conditional dependence relationship between them and remaining hidden variable, namely, specifying their $CPT_{(s)}$. Inferring user's knowledge is to compute posterior probability of hidden variables (according to formula III.1.3') when evidence variables change their values. This process can be known as knowledge diagnosis.

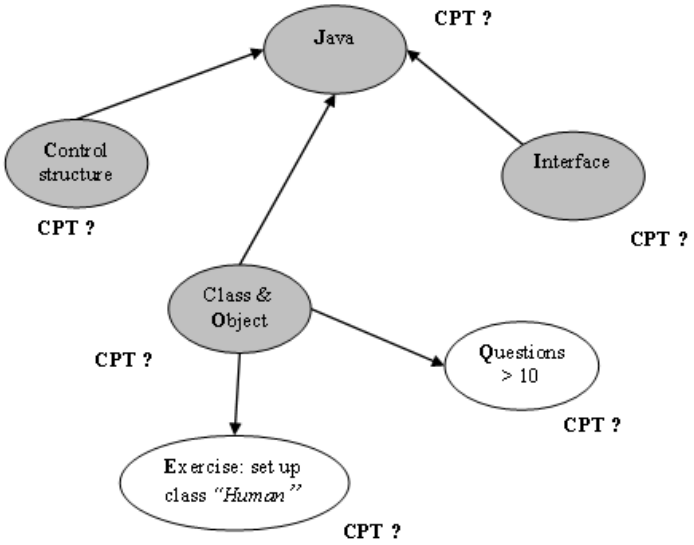


Figure III.1.2. Combination of BN and overlay model
(*evidence nodes are unshaded, otherwise, hidden nodes are shaded*)

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 such as teachers, lecturers, supervisors… or by learning algorithms.
- Specifying the important parameters which are $CPT_{(s)}$ of all variables. This task called development of quantitative model is described right now.

Specifying $CPT_{(s)}$ of variables

Suppose Java course is constituted of four concepts considered as hidden variables whose links are prerequisite relationships. Additionally, there are two evidence variables: "Questions > 10" and "Exercise: set up class *Human*". That learner asks more than 10 questions is to tell how much her/his amount of knowledge. Like that, evidence "Exercise: set up class *Human*" proves whether or not he/she understands concept "Class & Object". The number (in range *0…1*) that measures the relative importance of each prerequisite or evidence 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/from each child/parent node in case of hidden/evidence variable should be *1*. It means that each weight is normalized.

Your attention please, the relationship between hidden variable (*H*) and evidence variable (*E*) must be from *H* to *E* 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:

- *Prerequisite relationships* among hidden variables.

- *Diagnostic relationships* of hidden variables to evidences. The mastery of concepts (hidden) 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.
-

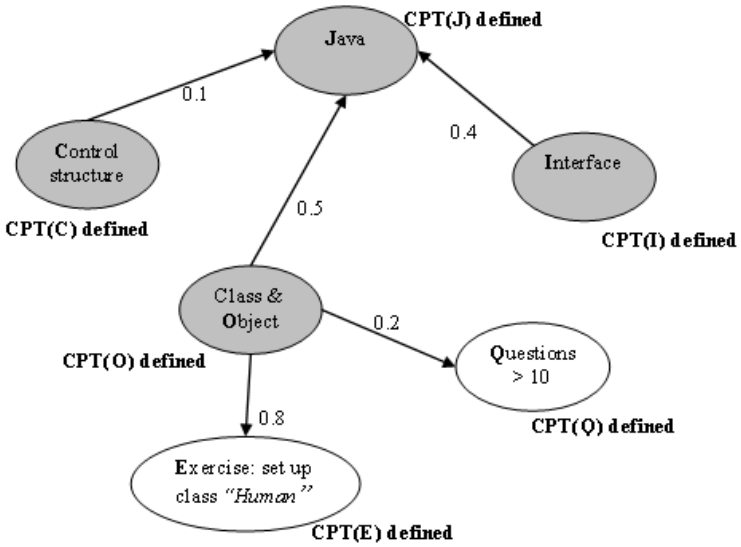


Figure III.1.3. Bayesian overlay model and its parameters in full

In this example, node *J* (Java) has three parents: *C* (Control structure), *O* (Class & Object), and *I* (Interface) which in turn are corresponding to three weights of prerequisite relationship: $w_1=0.1$, $w_2=0.5$, $w_3=0.4$. Conditional probability of *J* is computed as follows:

$Pr(J \mid C, O, I) = w_1*h_1 + w_2*h_2 + w_3*h_3$

where $h_1 = \begin{cases} 1 & if\ C = J \\ 0 & otherwise \end{cases}$ $h_2 = \begin{cases} 1 & if\ O = J \\ 0 & otherwise \end{cases}$ $h_3 = \begin{cases} 1 & if\ I = J \\ 0 & otherwise \end{cases}$

Note: {*J, C, O, I*} is complete set of mutually exclusive variables (of course, each also variable is random and binary). Generalizing about formula III.1.7, it is that:

$$\Pr(X = 1 \mid Y_1, Y_2, ..., Y_n) = \sum_{i=1}^{n} w_i * h_i \quad (Formula\ III.1.7)$$

where $h_i = \begin{cases} 1 & if\ Y_i = X \\ 0 & otherwise \end{cases}$ with given random binary variables *X*, Y_i.

Obviously, $Pr(not\ X \mid Y_1, Y_2, \ldots, Y_n) = 1 - Pr(X \mid Y_1, Y_2, \ldots, Y_n)$.

The formula III.1.7 so-called Union-gate inference is proven in the next section III.1.1.3. Applying formula III.1.7, the CPT (s) of *J, E* and *Q* are determined below:

T_1		
C O I	**Pr(J = 1)**	**Pr(J = 0)** *1- p(J = 1)*
1 1 1	**1.0** (*0.1*1 + 0.5*1 + 0.4*1*)	**0.0**
1 1 0	**0.6** (*0.1*1 + 0.5*1 + 0.4*0*)	**0.4**
1 0 1	**0.5** (*0.1*1 + 0.5*0 + 0.4*1*)	**0.5**
1 0 0	**0.1** (*0.1*1 + 0.5*0 + 0.4*0*)	**0.9**
0 1 1	**0.9** (*0.1*0 + 0.5*1 + 0.4*1*)	**0.1**
0 1 0	**0.5** (*0.1*0 + 0.5*1 + 0.4*0*)	**0.5**
0 0 1	**0.4** (*0.1*0 + 0.5*0 + 0.4*1*)	**0.4**
0 0 0	**0.0** (*0.1*0 + 0.5*0 + 0.4*0*)	**1.0**

T_2		
E	**Pr(E = 1)**	**Pr(E = 0)** *1- Pr(E = 1)*
1	**0.8** (*0.8*1*)	**0.2**
0	**0.0** (*0.8*0*)	**1.0**

T_3		
Q	**Pr(Q = 1)**	**Pr(Q = 0)** *1- Pr(Q = 1)*
1	**0.2** (*0.2*1*)	**0.8**
0	**0.0** (*0.2*0*)	**1.0**

Table III.1.1, III.1.2, III.1.3. $CPT_{(s)}$ of *J, E, Q* namely T_1, T_2, T_3

Because concepts *C, O, I* has no prerequisite knowledge for understanding, their $CPT_{(s)}$ are specified as prior probabilities obeying uniform distribution (assigned medium value *0.5* in most cases).

PR(C=1)	**PR(C=0)**
0.5	0.5
PR(O=1)	**PR(O=0)**
0.5	0.5
PR(I=1)	**PR(I=0)**
0.5	0.5

Table III.1.4, III.1.5, III.1.6. $CPT_{(s)}$ of *C, O, I* namely T_4, T_5, T_6

Inferring user's knowledge

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

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

where *Pr*(*J, C, O, I, E, Q*) is global joint probability distribution, *Pr*(*J, C, O, I, E, Q*) = *Pr*(*C*) * *Pr*(*O*) * *Pr*(*I*) * Pr(*E*|*O*) * *Pr*(*Q*|*O*) * Pr(*J*|*C,O,I*).

Applying all $CPT_{(s)}$ in table III.1.1, III.1.2, III.1.3, III.1.4, III.1.5, III.1.6, it is able to determined *Pr*(*J, C, O, I, E, Q*). After that, we compute *Pr*(*J = 1 | C, O, I , E = 1, Q = 1*) to answer above question.
Note, the set of all parents of a hidden node is the complete set of mutually exclusive hidden variables and the set of all evidence nodes which are children of a hidden node is the complete set of mutually exclusive evidence variables.

III.1.1.3. Union-gate inference

This section is the proof of formula III.1.7. Suppose every node is binary, Union-gate inference in Bayesian network simulates is based on three assumptions:

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

$$I = 0 \Leftrightarrow I \text{ turned OFF}$$
$$I = 1 \Leftrightarrow I \text{ turned ON}$$

- *Inhibition independence*: Inhibitions are mutually independent. For example inhibition I_1 of X_1 is independent from inhibition I_2 of X_2.
- *Union condition*: Suppose we have a set of cause-effect relationships in which *Y* is the effect of many causes X_1, X_2,…, X_n (see following figure). Let I_i be the inhibition of X_i. The event (effect) *Y* is the union of all causes X_i (s).

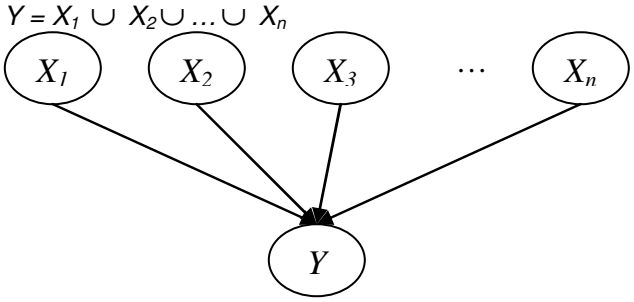


Figure III.1.4. Cause-effect relationships

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

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

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

Applying "*Union condition*", the condition probability of *Y* is the probability of union of all A_i (s).

$$\Pr(Y) = \Pr(\bigcup_i A_i)$$

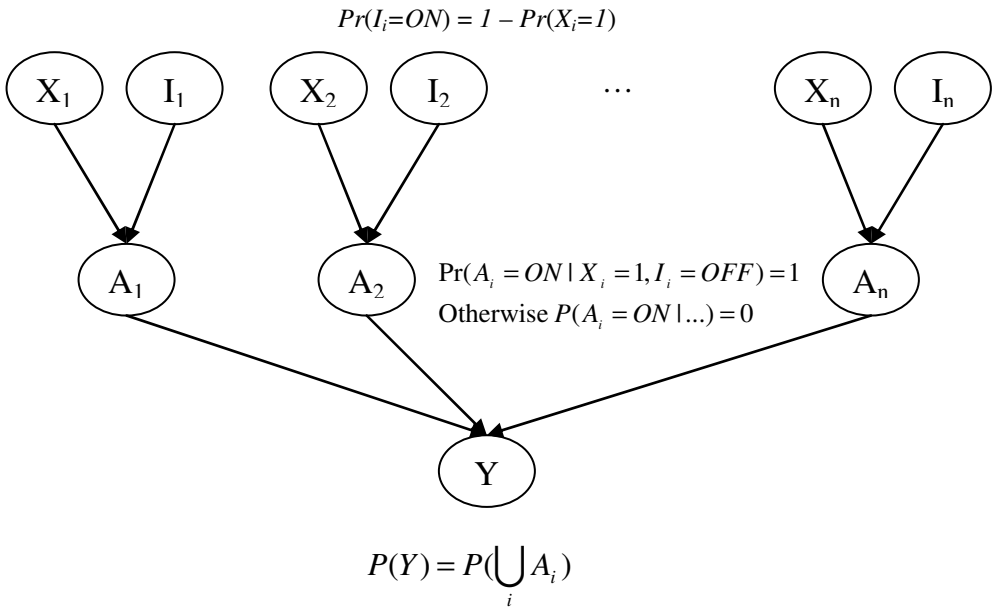


Figure III.1.5. Union-gate model

Now the strength of each cause-effect relationship $X_i \rightarrow Y$ is quantified by the conditional probability table $Pr(Y|X_i)$. Suppose causes $(X_1, X_2, \ldots, X_i, \ldots, X_n)$ become evidences having values $(x_1, x_2, \ldots, x_i, \ldots, x_n)$. Let $Pr(X_i=1) = p_i$ be the probability of $X_i = 1$. The probability of X_i 's inhibition is the inverse:

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

$Pr(I_i=OFF) = Pr(X_i=1) = p_i$
Let K be the set of such i that $X_i = 1$.
$\forall i \in K, X_i = 1$

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

$$\Pr(Y \mid X_1, X_2, ..., X_n) = \Pr(\bigcup_i A_i \mid X_1, X_2, ..., X_n) \quad \text{(due to Union condition)}$$

$$= \sum_i \Pr(A_i \mid X_1, X_2, ..., X_n) \quad \text{(Because } A_i \text{ (s) are mutually independent)}$$

$$= \sum_i \Pr(A_i \mid X_i) \quad \text{(Because } A_i \text{ is only independent on } X_i\text{)}$$

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

$$\Pr(Y = 1 \mid X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$$

$$= \sum_i \Pr(A_i = ON \mid X_i = x_i)$$

$$= \sum_i (\Pr(A_i = ON \mid X_i = x_i, I_i = ON) \Pr(I_i = ON) + \Pr(A_i = ON \mid X_i = x_i, I_i = OFF) \Pr(I_i = OFF))$$

(Applying the law of total probability)

$$= \sum_{i \in K} (\Pr(A_i = ON \mid X_i = 1, I_i = ON) \Pr(I_i = ON) + \Pr(A_i = ON \mid X_i = 1, I_i = OFF) \Pr(I_i = OFF)) +$$

$$\sum_{i \notin K} (\Pr(A_i = ON \mid X_i = 0, I_i = ON) \Pr(I_i = ON) + \Pr(A_i = ON \mid X_i = 0, I_i = OFF) \Pr(I_i = OFF))$$

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

$$= \sum_{i \in K} p_i$$

In conclusion, we have

$Pr(Y=1|X_1,X_2,\ldots, X_n) = \sum_{i \in K} p_i$

$Pr(Y=0|X_1,X_2,\ldots, X_n) = \sum_{i \notin K} (1 - p_i)$

Example:
Given cause-effect relationship shown in following figure. Given prior probabilities of causes X_1, X_2, X_3 are *0.2, 0.7, 0.1*, respectively. We need to compute the conditional probability of effect $Pr(Y=1|X_1=1, X_2=1, X_3=0)$.

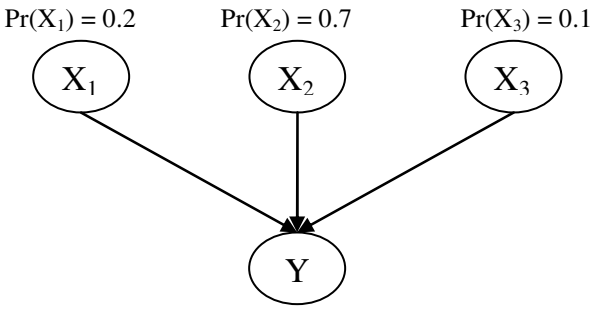


Figure III.1.6. Union-gate inference example

Applying Union-gate inference, we have:
$P(Y=1| X_1=1, X_2=1, X_3=0) = 1*0.2 + 1*0.7 = 0.9$

III.1.1.4. Evaluation

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

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

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

As already discussed, the structure and parameters ($CPT_{(s)}$) in our model are fixed and specified by experts. However, they must be evolved after each occurrence of evidence. When learning machine 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.1.4, III.1.5. The knowledge sub-model created by combining Bayesian network and overlay model is called as Bayesian overlay model or Bayesian model 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.1.2.

III.1.2. Incorporate Bayesian inference into adaptation rules

Adaptive Hypermedia System (AHS) aims to provide users the adaptation effect based on their characteristics. In other words, AHS ensures that the links that are offered and the content of the information pages are adapted to each individual user. AHA! [De Bra 1998] developed by Debra 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 Reference Model [Wu 2002] have 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 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 about AHA! in I.2.3.3 but now we should survey it in detailed. The AHA! engine [De Bra, Smits, Stash 2006] manipulates three main models as below:

1. *Domain model* [De Bra, Houben, Wu 1999].

The domain model consists of concepts which are topics in the application domain. Each concept has:

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

The sequence of anchors describes sub-structures within a concept. These can be used as anchor point for links such as the source or destination of links. The presentation specification navigates the runtime layer how to display concept's content. The attribute-value pairs are arbitrary. Each of them tells us optional information of concept and depends on idea of experts. Some usual attributes are shown in below:

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

2. *User model* [De Bra, Houben, Wu 1999].

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, user model is implicated as overlay model.

3. *Adaptation model* [De Bra, Houben, Wu 1999].

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

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

In the overlay model, with each concept in domain model, there is corresponding concept in user model with the same name. So the expression "*concept.attribute*" refers both the domain model attribute and user model attribute. The adaptation rules whose purpose is to update user model are called *event-condition-action* (ECA) [De Bra, Houben, Wu 1999] rules because:

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

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

Event: access(*C*)
Condition: C.visited = true
Action: C.knowledge = 100%

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

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

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

Condition: C.knowledge > 50
Action: True status: providing advanced subjects about C
False status: providing basic explanations about C

This rule can be interpreted in XML form

```
<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:

- Suppose a student is recommended to learn concept "Class and Object" before concept "Interface and Package" in Java course. There is the prerequisite relationship in which "Class and Object" is the precondition of "Interface and Package". Namely, the *"recommended"* attribute of "Interface and Package" become *true* if and only if the *"knowledge"* attribute of "Class and Object" reaches 50%. Whenever the *"knowledge"* attribute changes its values, the *"recommended"* attribute is checked and can be re-assigned new value (*"true"*). This technique is called forward reasoning because the *"recommended"* attribute is determined as soon as possible and thus before it is actually needed, regardless of user requirement.
- Otherwise, the *"recommended"* attribute is only determined when students require learning "Interface and 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:

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

Dummy attribute and backward reasoning

Suppose Java course is constituted of three concepts considered as knowledge variables whose links are dependency relationships. Additionally, there are two evidence variables: "*Questions > 10*" and "*Exercise: set up class Human*". That learner asks more than 10 questions is to tell how much her/his amount of knowledge. Like that, evidence "Exercise: set up class Human" proves whether or not he/she understands concept "Class & Object". The number (in range *0...1*) that measures the relative importance of each prerequisite or evidence is defined as the weight of arc from parent node to child node. All weights concerning the child variable are normalized and used to build up its CPT.

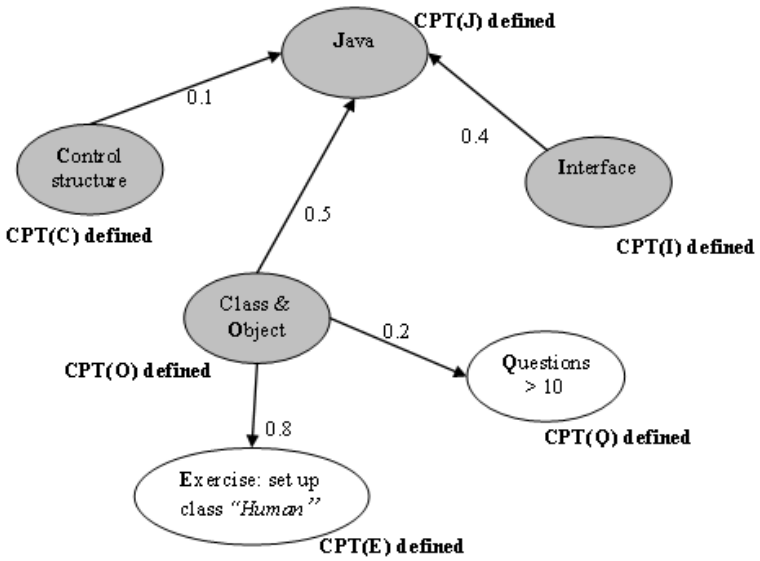


Figure III.1.7. Bayesian overlay model and its parameters in full.

Note that five concepts *O, I, J, E, Q* denote knowledge variables and evidences: *Class & **O**bject, **J**ava, **I**nterface, "**E**xercise: set up class Human"* and *"**Q**uestions > 10"*, respectively. Concepts *E*, *Q* have the new attribute *is_evidence* indicating whether *E*, *Q* are fit to become evidences or not. Concepts *O, I, J* have the new dummy attribute *do$bayes$infer*. It called "dummy" because attribute *do$bayes$infer* does not exists actually in user model. It is only used as the denotation for backward reasoning with Bayesian network inference. Namely, *J.do$bayes$infer* is the posterior probability tell us how mastered leaner is over the concept *J*.
I defined two ECA rules for updating evidences "*Questions > 10*", "*Exercise: set up class Human*" and one CA rule setting presentation specification.

The 1st ECA rule: If user asks more than 10 questions then the attribute *Q. is_evidence* is set to *true*.
The 2nd ECA rule: If user does exercise "Exercise: set up class Human" the attribute E. is_evidence is set to true.
The 3rd 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. do$bayes$infer*

	Event	Condition	Action
Rule 1st	access(Q)	Q.visited > 10	Q.is_evidence = true
Rule 2nd	access(E)	E.visited > 10	E.is_evidence = true
Rule 3rd		J. do$bayes$infer ≥ 0.5	- *True status*: providing advanced subjects about Java *False status*: providing basic explanations about Java

The 3rd rule is translated in XML form:

<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 *3rd* 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 *1st* and *2nd* rules are executed, concepts *Q* and *E* become actual evidences.

$$\Pr(J=1 \mid Q=1, E=1) = \frac{\sum_{O,I} \Pr(O, I, J=1, E=1, Q=1)}{\sum_{O,I,J} \Pr(O, I, J, E=1, Q=1)}$$

J.do$bayes$infer =

Where $\Pr(O, I, J, E, Q) = \Pr(O) * \Pr(I) * \Pr(J \mid O, I) * \Pr(E \mid O) * \Pr(Q \mid O)$ is the global joint probability distribution and *J.do$bayes$infer* is posterior probability that represents user's knowledge about *J*.
Our method 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 our method into other inferences such as neural network, hidden Markov model, etc; hence, there is no need to change the main technique and what we do is to add new dummy attributes to domain element.

III.1.3. Evolution of Bayesian overlay model

Adaptive learning systems require well-organized user model along with solid inference mechanism. Overlay modeling is the method in which the domain is decomposed into a set of elements and the user model is simply a set of masteries over those elements. The combination between overlay model and 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 section III.1.1. Thus it is compulsory to pre-define parameters, namely, Conditional Probability Tables ($CPT_{(s)}$) in BN but no one ensured absolutely the correctness of these $CPT_{(s)}$. This section III.1.3 discusses about how to enhance parameters' quality in Bayesian overlay model, in other words, this is the evolution of $CPT_{(s)}$.

As known, user model is the core of almost adaptive learning systems. There are some effective modeling methods such as stereotype, overlay, plan recognition but overlay model is proven soundness due to two its properties: flexible graphic structure and reflecting comprehensibly the domain knowledge in education course. The basic ideology of overlay model is to represent user knowledge as subset of domain model. The combination between overlay model and BN (see section III.1.1) will make use of each method's strong points and restraints drawbacks.

- The structure of overlay model is translated into BN, each user knowledge element becomes an node in BN.
- Each prerequisite relationship between domain element in overlay model becomes an conditional dependence assertion signified by CPT of each node in BN.
- Domain elements are defined as hidden nodes and other learning objects which are used to assess user's performance are consider as evidence nodes in BN.

Such model is called Bayesian overlay model. 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 $CPT_{(s)}$. Section III.1.3.1 discusses about main subject "learning parameters or the evolution of parameters". Section III.1.3.2 gives an example of parameter evolution. 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.

III.1.3.1. Learning parameters in Bayesian model

Dummy variables and augmented BN

In continuous case, the CPT of each node is replaced by the probability density function (PDF). 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 $\beta(f; a, b)$ or $Beta(f; a, b)$ with parameters $a, b, N=a+b$ where a, b should be integer number greater than 0.

$$\beta(f) = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} f^{a-1}(1-f)^{b-1} \qquad (Formula\ III.1.8)$$

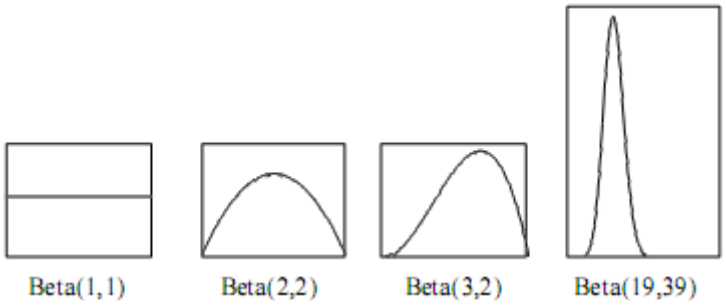


Figure III.1.8. Beta functions

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

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

The integral will converges if *x>0*, at that time, $\Gamma(x) = (x-1)!$. Of course, we have $\frac{\Gamma(x+1)}{\Gamma(x)} = x$ (*Formula III.1.9*).

From formula III.1.8, we have:

$$\int_0^1 f^a (1-f)^b df = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \quad (\textit{Formula III.1.10})$$

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

Proof,

$$= \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \int_0^1 \beta(f; a+1, b+1) df \quad \text{(due to formula III.1.8)}$$

$$= \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \Pr(0 \le f \le 1) = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)}$$

$$(\text{due to } \Pr(0 \le f \le 1) = 1)$$

Suppose there is one binary variable *X* in network and the probability distribution of *X* is considered as relative frequency having values in [*0, 1*] which is the range of variable *F*. We add a dummy variable *F* (whose space consists of numbers in [*0, 1*], of course) which acts as the parent of *X* and has a beta density function *β*(*f; a, b*), so as to:
Pr(*X=1*|*f*) = *f*, where *f* denotes values of *F*.
X and *F* constitute a simple network which is referred as augmented BN. So *X* is referred as real variable (hypothesis) opposite to dummy variable.

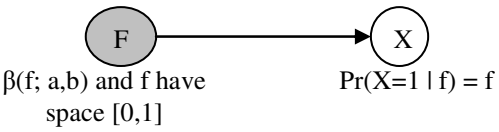


Figure III.1.9. The simple augmented BN with only one hypothesis node *X*

Obviously, *P*(*X=1*) = *E*(*F*) where *E*(*F*) is the expectation of *F*.
Proof, owing to the law of total probability

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

Due to *F* is beta function, $E(F) = \frac{a}{N}$, so, $\Pr(X=1) = \frac{a}{N}$ (*Formula III.1.11*)

Proof,

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

$$= \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \int_0^1 f^a (1-f)^{b-1} = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+1)\Gamma(b)}{\Gamma(N+1)} \text{ (due to formula III.1.10)}$$

$$= \frac{a}{N} \quad \text{(applying formula III.1.9)}$$

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 $Pr(X^{(u)} = 1 \mid f) = f$. So, all $X^{(u)}$ are conditionally dependent on *F*. The probability of variable *X*, $Pr(X=1)$ is learned by these evidences.

We denote the vector of all evidences as $E = (X^{(1)}, X^{(2)}, ..., X^{(M)})$ which is also called the sample of size *M*. Given this sample, $\beta(f)$ is called the prior density function, and $Pr(X^{(u)} = 1) = a/N$ (due to formula III.1.11) is called prior probability of $X^{(u)}$. *It is necessary to determine the posterior density function* $\beta(f|E)$ *and the posterior probability of X, namely* $Pr(X|E)$. *The nature of this process is the parameters learning*. Note that $Pr(X|E)$ is referred as $Pr(X^{(M+1)} \mid E)$.

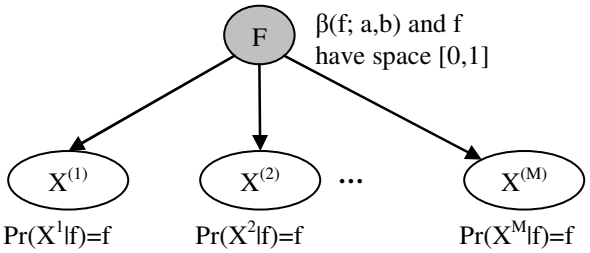


Figure III.1.10. The sample $E=(X^{(1)}, X^{(2)},..., X^{(M)})$ size of *M*

We only surveyed in the case of binomial sample, in other words, *E* having binomial distribution is called binomial sample and the network in figure III.1.8 becomes a binomial augmented BN. Then, suppose *s* is the number of all evidences $X^{(i)}$ which have value *1* (success), otherwise, *t* is the number of all evidences $X^{(j)}$ which have value *0* (failed). Of course, $s + t = M$.

Owing the law of total probability, we have

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

$$= \int_0^1 f^s(1-f)^t \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} f^{a-1}(1-f)^{b-1}df \quad \text{(applying formula III.1.8)}$$

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

$$= \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a+b+s+t)} \quad \text{(due to formula III.1.10)}$$

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

(*Formula III.1.12*)

And,

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

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

(*Formula III.1.13*)

Computing posterior density function

Now, we need to compute the posterior density function $\beta(f|E)$ and the posterior probability $Pr(X=1|E)$. It is essential to determine the probability distribution of X.

$$\beta(f \mid E) = \frac{\Pr(E \mid f)\beta(f)}{\Pr(E)} \qquad \text{(Bayes' law)}$$

$$= \frac{f^s(1-f)^t \beta(f)}{E(f^s(1-f)^t)}$$

$$\left(\text{due to } \Pr(\mathrm{E} \mid \mathrm{f}) = \prod_{i=1}^{M} \Pr(X^i \mid f) = f^s(1-f)^t \quad \text{and apply formula III.1.13}\right)$$

$$= \frac{f^s(1-f)^t \dfrac{\Gamma(N)}{\Gamma(a)\Gamma(b)} f^{a-1}(1-f)^{b-1}}{\dfrac{\Gamma(N)}{\Gamma(N+M)} \dfrac{\Gamma(a+s)\Gamma(b+t)}{\Gamma(a)\Gamma(b)}} \qquad \text{(apply formula III.1.8, III.1.12)}$$

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

(*Formula III.1.14*)

Then the posterior density function is $\beta(f; a+s, b+t)$ where the prior density function is $\beta(f; a, b)$. According to formula III.1.11, the posterior probability $Pr(X=1|E) = E(\beta(f|E)) = \dfrac{a+s}{a+s+b+t} = \dfrac{a+s}{N+M}$ (*Formula III.1.15*)

In general, you should merely remember the formula III.1.8, III.1.11, III.1.14, III.1.15 and the way to recognize prior density function, prior probability of X and posterior density function, posterior probability of X, respectively.

Expanding augmented BN with more than one hypothesis node

Suppose we have a BN with two binary random variables and there is conditional dependence assertion between these nodes. See the network and $CPT_{(s)}$ in following figure:

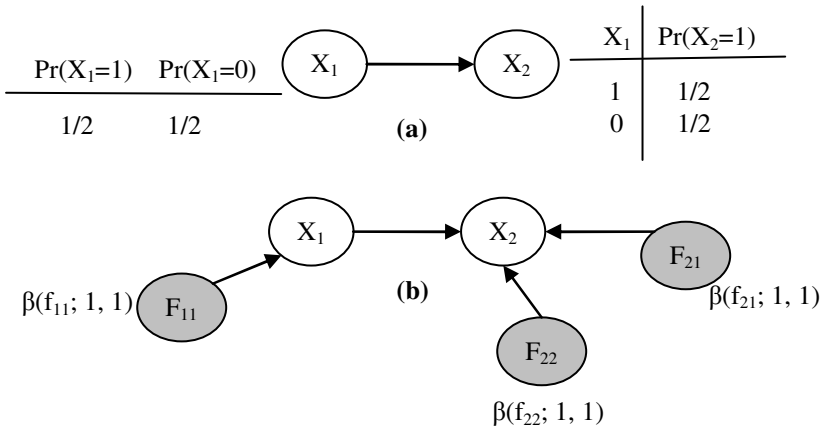


Figure III.1.11. BN (a) and expended augmented BN (b)

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

- If X_i has no parent (not conditionally dependent on any others), we add only one dummy variable denoted F_{i1} having the probability density function $\beta(f_{i1}; a_{i1}, b_{i1})$ so as to: $Pr(X_i=1|f_{i1})= f_{i1}$
- If X_i has a set of k_i parents and each parent pa_{il} ($l=\overline{1,k_i}$) is binary, we add a set of $c_i=2k_i$ dummy variables $F_i = \{f_{i1}, f_{i2},\ldots, f_{ic_i}\}$, in turn, instantiations of parents $PA_i= \{pa_{i1}, pa_{i2}, pa_{i3},\ldots, pa_{ic_i}\}$. In other words, c_i denotes the number of instantiations of the parents PA_i. We have $Pr(X_i=1|pa_{ij},f_{i1},\ldots,f_{ij},\ldots, f_{ic_i})=f_{ij}$ where $\beta(f_{ij}) = \frac{\Gamma(N_{ij})}{\Gamma(a_{ij})\Gamma(b_{ij})} f^{a_{ij}-1}(1-f_{ij})^{b_{ij}-1}$

All f_{ij} have no parent and are mutually independent, so, $\beta(f_{i1}, f_{i2},\ldots, f_{ic_i}) = \beta(f_{i1})\ \beta(f_{i2})\ldots\ \beta(f_{ic_i})$. Besides this local parameter independence, we have the global parameter independence if reviewing all variables X_i (s), such below:
$\beta(F_1, F_2,\ldots, F_n)= \beta(f_{11}, f_{12},\ldots, f_{ic_{in}})= \beta(f_{i1})\ \beta(f_{i2})\ldots\ \beta(f_{ic_{in}})$

All variables X_i and their dummy variables form the expended augmented BN representing the trust BN in figure III.1.9. In the trust BN, the conditional probability of variable X_i with the instantiation of its parent pa_{ij}, in other words, the ij^{th} conditional distribution is the expected value of F_{ij} as below:

$Pr(X_i=1|\ pa_{ij}=1) = E(F_{ij})=\frac{a_{ij}}{N_{ij}}$ (*Formula III.1.15*)

$$\Pr(X_i=1|\ pa_{ij}=1)=\int_0^1\ldots\int_0^1 \Pr(X_i=1|\ pa_{ij}=1, f_{i1},\ldots, f_{ic_i})\beta(f_{i1})\ldots\beta(f_{ic_i})df_{i1}\ldots df_{ic_i}$$

Proof, $=\int_0^1\ldots\int_0^1 f_{ij}\beta(f_{i1})\ldots\beta(f_{ic_i})df_{i1}\ldots df_{ic_i} = E(F_{ij})$

$(\text{due to } F_{ij}\ (s) \text{ are mutually independent and}$

$\Pr(X_i=1|\ pa_{ih_i}=1, f_{i1},\ldots, f_{ic_i}) = \Pr(X_i=1|\ pa_{ij}=1, f_{ij}) = f_{ij})$

Suppose we perform M trials of random process, the outcome of u^{th} trial which is BN like figure III.1.9 is represented as a random vector $X^{(u)} = \begin{pmatrix} X_1^{(u)} \\ \ldots \\ X_n^{(u)} \end{pmatrix}$ containing all evidence variables in network. $X^{(u)}$ is also called evidence vector (or evidence, briefly). M trials constitute the sample of size M which is the set of random vectors denoted as $E=\{X^{(1)}, X^{(2)},\ldots, X^{(M)}\}$. E is also called evidence matrix. We review only in case of binomial sample, it means that E is the binomial BN sample of size M. For example, this sample corresponding to the network in figure III.1.9 is shown below:

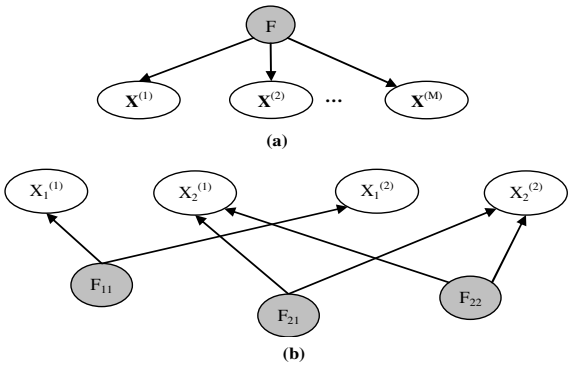


Figure III.1.12. Expanded binomial BN sample of size M

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

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

Where

- c_i is the number of instances of $X_i^{(u)}$'s parents. In binary case, each $X_i^{(u)}$'s parent has two instances/values, namely, *0* and *1*.
- s_{ij}, respective to f_{ij}, is the number of all evidences that variable $X_i = 1$ and $pa_{ij} = 1$.
- t_{ij}, respective to f_{ij}, is the number of all evidences that variable $X_i = 1$ and $pa_{ij} = 0$.

We have,

$$\Pr(E \mid F_1,...,F_n)=\prod_{i=1}^{n}\prod_{u=1}^{M}\Pr(X_i^{(u)} \mid pa_i, F_i)=\prod_{i=1}^{n}\prod_{j=1}^{c_i}(f_{ij})^{s_{ij}}(1-f_{ij})^{t_{ij}} \quad (\textit{Formula III.1.16})$$

$$\Pr(E)=\prod_{i=1}^{n}\prod_{j=1}^{c_i}E(f_{ij}{}^{s_{ij}}(1-f_{ij})^{t_{ij}}) \quad (\textit{Formula III.1.17})$$

$$\Pr(E)=\prod_{i=1}^{n}(\int_{F_i}\prod_{u=1}^{M}\Pr(X_i^{(u)} \mid pa_i, F_i)\beta(F_i)dF_i)$$

(due to the law of total probability and the joint probability distribution)

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

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

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

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

There is the question "how to determine $E(f_{ij}{}^{s_{ij}}(1-f_{ij})^{t_{ij}})$". Applying formula III.1.12, we have:

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

(*Formula III.1.18*)

Updating posterior density function $\beta(f_{ij}|E)$

$$\beta(f_{ij} \mid E)=\frac{(f_{ij})^{s_{ij}}(1-f_{ij})^{t_{ij}}\beta(f_{ij})}{E(f_{ij}{}^{s_{ij}}(1-f_{ij})^{t_{ij}})}=beta(f_{ij};a_{ij}+s_{ij},b_{ij}+t_{ij}) \quad (\textit{Formula III.1.19})$$

Proof,

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

$$= \frac{\left(\int_0^1 ... \int_0^1 \Pr(\mathrm{E} \mid F_1, F_2, ..., F_n) \prod_{ij \neq mn} \beta(f_{ij}) df_{ij}\right)\beta(f_{mn})}{\Pr(E)} \quad \text{(law of total probabilit y)}$$

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

(apply formula III.1.16, III.1.17)

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

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

(expansion of $\beta(f_{mn})$ and applying formula III.1.12 to $E(f_{mn}{}^{s_{mn}}(1-f_{mn})^{t_{mn}})$)

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

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

According to formula III.1.15 and III.1.19,

$$Pr(X_i=1 \mid pa_{ij}=1, E) = E(F_{ij}) = E(\beta(f_{ij}/E) = \frac{a_{ij}+s_{ij}}{a_{ij}+s_{ij}+b_{ij}+t_{ij}} = \frac{a_{ij}+s_{ij}}{N_{ij}+M_{ij}} \quad (\textit{Formula III.1.20})$$

In short, in case of binomial distribution, if we have the real/trust BN embedded in the expanded augmented network such as figure III.1.9 and each dummy node F_{ij} has a prior beta distribution $\beta(f_{ij}; a_{ij}, b_{ij})$ and each hypothesis node X_i has the prior conditional probability $Pr(X_i=1 \mid pa_{ij}=1) = E(\beta(f_{ij})) = \frac{a_{ij}}{N_{ij}}$, the parameter learning process based on a set of evidences is to update the posterior density function $\beta(f_{ij}/E)$ and the posterior conditional probability $Pr(X_i=1 \mid pa_{ij}=1, E)$. Indeed, $\beta(f_{ij} \mid E) = beta(f_{ij}; a_{ij}+s_{ij}, b_{ij}+t_{ij})$ and $Pr(X_i=1 \mid pa_{ih_i}=1, E) = E(\beta(f_{ij}/E)) = \frac{a_{ij}+s_{ij}}{N_{ij}+M_{ij}}$

Example

Suppose we have the set of 5 evidences $E=\{X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}\}$ owing to network in figure III.1.9.

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

Table III.1.7. Set of evidences *E* corresponding to *5* trials (sample of size *5*)

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

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

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

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

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

Network in figure III.1.11 changed as follows:

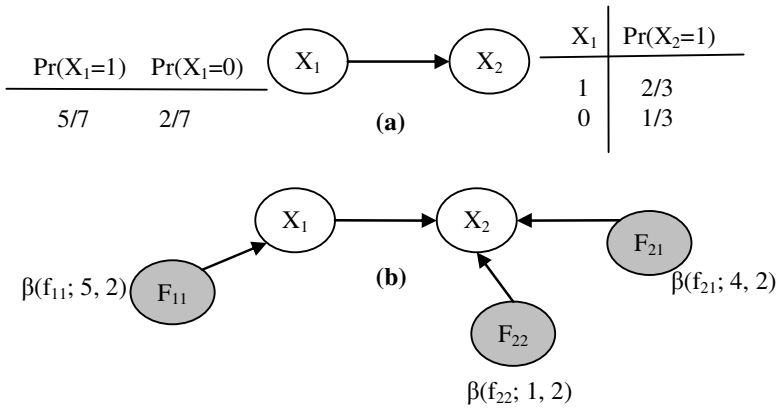


Figure III.1.13. Updated version of BN (a) and augmented BN (b) in figure III.1.9

III.1.3.2. Learning parameters in case of data missing

In practice there are some evidences in E such as $X^{(u)}$ (s) which lack information and thus, it stimulates the question "How to update network from data missing". We must address this problem by artificial intelligence techniques, namely, expectation maximization (EM) algorithm – a famous technique solving estimation of data missing. Like above example, we have the set of *5* evidences $E=\{X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}\}$ along with network in figure III.1.9 but the evidences $X^{(2)}$ and $X^{(5)}$ have not data yet.

	x_1	x_2
$X^{(1)}$	$X_1^{(1)} = 1$	$X_2^{(1)} = 1$
$X^{(2)}$	$X_1^{(2)} = 1$	$X_2^{(2)} = v_1?$
$X^{(3)}$	$X_1^{(3)} = 1$	$X_2^{(3)} = 1$
$X^{(4)}$	$X_1^{(4)} = 1$	$X_2^{(4)} = 0$
$X^{(5)}$	$X_1^{(5)} = 0$	$X_2^{(5)} = v_2?$

Table III.1.8. Set of evidences E (for network in figure III.1.9) with data missing

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

- $X_1^{(2)}=1$ and $X_2^{(2)}=1$, it is easy to infer that:

$$v_1=Pr(X_2^{(2)}=1|X_1^{(2)}=1) = E(\beta_{21})=\frac{a_{21}}{a_{21}+b_{21}} = ½$$

- $X_1^{(2)}=1$ and $X_2^{(2)}=0$, it is easy to infer that:

$$v_2=Pr(X_2^{(2)}=1|X_1^{(2)}=0) =E(\beta_{22})=\frac{a_{22}}{a_{22}+b_{22}} = ½$$

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

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

Table III.1.9. New split evidences *E'* for network in figure III.1.9

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

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

$$Pr(X_2=1|\ X_1=1)=E(\beta(f_{21}|E))=\frac{7/2}{7/2+5/2}=\frac{7}{12}$$

$$Pr(X_2=0|\ X_1=1)=E(\beta(f_{22}|E))=\frac{3/2}{3/2+3/2}=\frac{1}{2}$$

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

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

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

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

III.1.3.3. An example about parameter evolution

Suppose Java course is constituted of three concepts considered as knowledge variables whose links are prerequisite relationships. Additionally, there are two evidence variables: "*Questions > 10*" and "*Exercise: set up class Human*". That learner asks more than 10 questions is to tell how much her/his amount of knowledge. Like that, evidence "Exercise: set up class Human" proves whether or not he/she understands concept "Class & Object". The number (in range 0...1) that measures the relative importance of each prerequisite 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. This process is called parameter evolution.

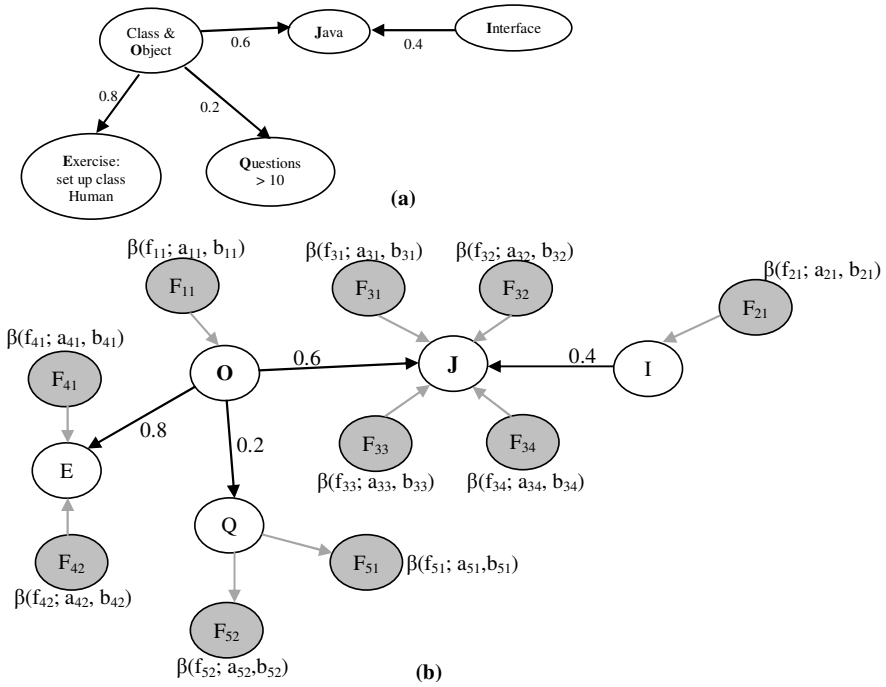


Figure III.1.14. BN (a) and augmented BN (b) of Java course

(*Nodes O, I, J denote knowledge variables Class & Object, Java, Interface respectively. Nodes E, Q denote evidence variables "Exercise: set up class Human" and "Questions > 10" respectively*)
In this example, node *J* (Java) has two parents: *O* (Class & Object) and *I* (Interface) which in turn are corresponding to two weights of prerequisite relationship: $w_1=0.6$, $w_2=0.4$. Conditional probability of *J* is computed as follows:

$p(J \mid C, O, I) = w_1*h_1 + w_2*h_2 + w_3*h_3$

where $h_1 = \begin{cases} 1 & if\ C = J \\ 0 & otherwise \end{cases}$ $h_2 = \begin{cases} 1 & if\ O = J \\ 0 & otherwise \end{cases}$ $h_3 = \begin{cases} 1 & if\ I = J \\ 0 & otherwise \end{cases}$

Note: {*J, C, O, I*} is complete set of mutually exclusive variables; of course, each also variable is random and binary. We have:

$$\Pr(X = 1 \mid Y_1, Y_2, ..., Y_n) = \sum_{i=1}^{n} w_i * h_i$$

where $h_i = \begin{cases} 1 & if\ Y_i = X \\ 0 & otherwise \end{cases}$ with given random binary variables X, Y_i.

Obviously, $Pr(not\ X \mid Y_1, Y_2, \ldots, Y_n) = 1 - Pr(X \mid Y_1, Y_2, \ldots, Y_n)$.

Table III.1.10. All variables and their density functions, prior probabilities

Real Variable	Dummy Variable	Density Function	Prior Probability
O	F_{11}	$\beta(f_{11}; a_{11}, b_{11})$	$Pr(O=1) = a_{11}/(a_{11}+b_{11}) = 0.5$
I	F_{21}	$\beta(f_{21}; a_{21}, b_{21})$	$Pr(I=1) = a_{21}/(a_{21}+b_{21}) = 0.5$
J	F_{31}	$\beta(f_{31}; a_{31}, b_{31})$	$Pr(J=1\|O=1, I=1) = a_{31}/(a_{31}+b_{31}) = 0.6*1+0.4*1 = 1$
J	F_{32}	$\beta(f_{32}; a_{32}, b_{32})$	$Pr(J=1\|O=1, I=0) = a_{32}/(a_{32}+b_{32}) = 0.6*1+0.4*0 = 0.6$
J	F_{33}	$\beta(f_{33}; a_{33}, b_{33})$	$Pr(J=1\|O=0, I=1) = a_{33}/(a_{33}+b_{33}) = 0.6*0+0.4*1 = 0.4$
J	F_{34}	$\beta(f_{34}; a_{34}, b_{34})$	$Pr(J=1\|O=0, I=0) = a_{34}/(a_{34}+b_{34}) = 0.6*0+0.4*0 = 0$
E	F_{41}	$\beta(f_{41}; a_{41}, b_{41})$	$Pr(E=1\|O=1) = a_{41}/(a_{41}+b_{41}) = 0.8*1 = 0.8$
E	F_{42}	$\beta(f_{42}; a_{42}, b_{42})$	$Pr(E=1\|O=0) = a_{42}/(a_{42}+b_{42}) = 0.8*0 = 0$
Q	F_{51}	$\beta(f_{51}; a_{51}, b_{51})$	$Pr(Q=1\|O=1) = a_{51}/(a_{51}+b_{51}) = 0.2*1 = 0.2$
Q	F_{52}	$\beta(f_{52}; a_{52}, b_{52})$	$Pr(Q=1\|O=0) = a_{52}/(a_{52}+b_{52}) = 0.2*0 = 0$

Equivalent sample size N

That $Pr(O=1)$ equals 0.5 and $Pr(I=1)$ equals 0.5 is due to uniform distribution. Before specifying parameters, should glance over the concept "*equivalent sample size*" 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 it exists the number N so that:

$a_{ij} + b_{ij} = Pr(pa_{ij}) * N$

Given $Pr(pa_i)$ is probability of an certain instance of an X_i 's parent according to f_{ij}. Recall in the case of a root, PA_i is empty and $Pr(pa_{ij})=1$. Then the network is called to have *equivalent sample size N*. For example, reviewing figure III.1.12, given $\beta(f_{11}; 2, 2)$, $\beta(f_{21}; 1,1)$, $\beta(f_{22}; 1,1)$, we have:

$4 = a_{11} + b_{11} = 1*4 = 4$ ($Pr(pa_{ij})=1$ because X_1 has no parent)

$2 = a_{21} + b_{21} = Pr(X_1=1) *4 = ½*4 = 2$

$2 = a_{22} + b_{22} = Pr(X_1=0) *4 = ½*4 = 2$

So, this network has equivalent sample size 4. For all i and j, if

$a_{ij} = Pr(X_i=1|pa_{ij}) * Pr(pa_{ij}) * N$

$b_{ij} = Pr(X_i=0|pa_{ij}) * Pr(pa_{ij}) * N$

(*Formula III.1.21*)

then the resultant augmented BN has equivalent sample size N

Proof, $a_{ij}+b_{ij} = Pr(X_i=1|pa_{ij}) * Pr(pa_{ij}) * N + Pr(X_i=0|pa_{ij}) * Pr(pa_{ij}) * N$

$= Pr(pa_{ij}) * N * (Pr(X_i=1|pa_{ij}) + Pr(X_i=0|pa_{ij}))$

$= Pr(pa_{ij}) * N * (Pr(X_i=1|pa_{ij}) + (1 - Pr(X_i=1|pa_{ij}))) = Pr(pa_{ij}) * N$

Back to Java course, we choose size $N = 100$. Applying formula III.1.21, all parameters a_{ij} and b_{ij} are shown in table and figure below:

Table III.1.11. All parameters of prior density functions

Density Functions	Parameters
$\beta(f_{11}; a_{11}, b_{11})$	$a_{11} = Pr(O=1)*1*10 = 0.5*100 = \mathbf{50}$
	$b_{11} = Pr(O=0)*1*10 = 0.5*100 = \mathbf{50}$
$\beta(f_{21}; a_{21}, b_{21})$	$a_{21} = Pr(I=1)*1*10 = 0.5*100 = \mathbf{50}$
	$b_{21} = Pr(I=0)*1*10 = 0.5*100 = \mathbf{50}$
$\beta(f_{31}; a_{31}, b_{31})$	$a_{31} = Pr(J=1\|O=1,I=1)*Pr(O=1, I=1)*100$
	$= Pr(J=1\|O=1,I=1)*Pr(O=1)*Pr(I=1)*100$
	$= 1*0.5*0.5*100 = \mathbf{25}$
	$b_{31} = Pr(J=0\|O=1,I=1)*Pr(O=1, I=1)*100 = \mathbf{0}$
$\beta(f_{32}; a_{32}, b_{32})$	$a_{32} = Pr(J=1\|O=1, I=0)*Pr(O=1, I=0)*100$
	$= 0.6*0.5*0.5*100 = \mathbf{15}$
	$b_{32} = Pr(J=1\|O=1, I=0)*Pr(O=1, I=0)*100 = \mathbf{10}$
$\beta(f_{33}; a_{33}, b_{33})$	$a_{33} = Pr(J=1\|O=0, I=1)*Pr(O=1, I=1)*100$
	$= 0.4*0.5*0.5*100 = \mathbf{10}$
	$b_{33} = Pr(J=0\|O=0, I=1)*Pr(O=1, I=1)*100 = \mathbf{15}$
$\beta(f_{34}; a_{34}, b_{34})$	$a_{34} = Pr(J=1\|O=0, I=0)*Pr(O=0, I=0)*100 = \mathbf{0}$
	$b_{34} = Pr(J=0\|O=0, I=0)*Pr(O=0, I=0)*100$
	$= 1*0.5*0.5*100 = \mathbf{25}$
$\beta(f_{41}; a_{41}, b_{41})$	$a_{41} = Pr(E=1\|O=1)*Pr(O=1)*100 = \mathbf{40}$
	$b_{41} = Pr(E=0\|O=1)*Pr(O=1)*100 = \mathbf{10}$
$\beta(f_{42}; a_{42}, b_{42})$	$a_{42} = Pr(E=1\|O=0)*Pr(O=0)*100 = \mathbf{10}$
	$b_{42} = Pr(E=0\|O=0)*Pr(O=0)*100 = \mathbf{40}$
$\beta(f_{51}; a_{51}, b_{51})$	$a_{51} = Pr(Q=1\|O=1)*Pr(O=1)*100 = \mathbf{10}$
	$b_{51} = Pr(Q=0\|O=1)*Pr(O=1)*100 = \mathbf{40}$
$\beta(f_{52}; a_{52}, b_{52})$	$a_{52} = Pr(Q=1\|O=0)*Pr(O=0)*100 = \mathbf{40}$
	$b_{52} = Pr(Q=0\|O=0)*Pr(O=0)*100 = \mathbf{10}$

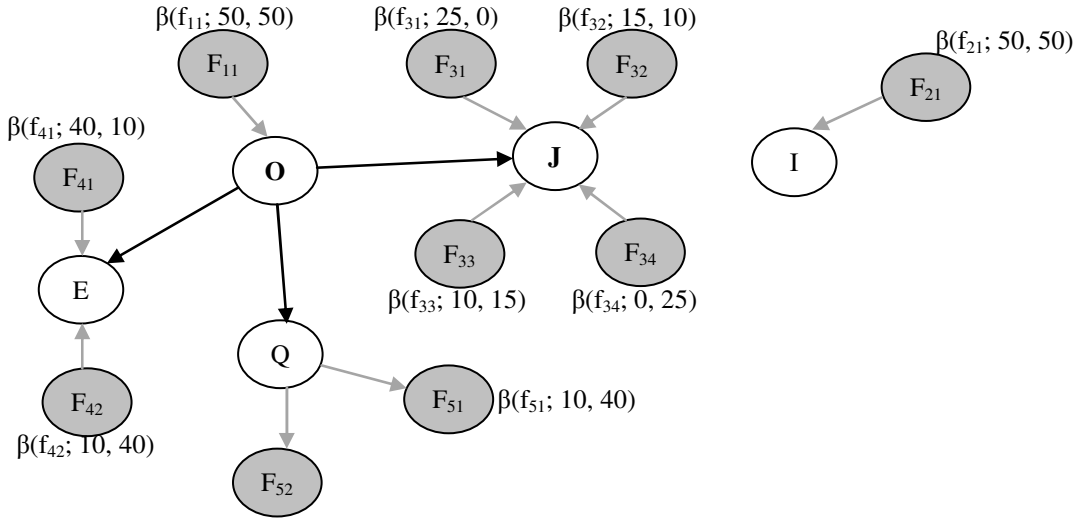


Figure III.1.15. Augmented BN with initial parameters in full

Evolution of parameters

Suppose we have 2 evidences $X^{(1)}=(E^{(1)}=1, Q^{(1)}=1)$ and $X^{(2)}=(E^{(2)}=1, Q^{(2)}=0)$.
The first observation: User does well the "*Exercise: set up class Human*" and asks more than *10 question* in course.

- The second: User does well the "*Exercise: set up class Human*" but asks less than *10 questions.*

But it lacks information about other concepts such as *O, J, I*. So there have not data yet, we split $X^{(1)}$, $X^{(2)}$ into many $X^{(1)}_i$, $X^{(2)}_i$ (s) for data missing cases.

Table III.1.12. Evidences and missing data needed to estimate

	$O^{(1)}$	$I^{(1)}$	$J^{(1)}$	$E^{(1)}$	$Q^{(1)}$
$X^{(1)}_1$	O=1, **$v(O^{(1)}_1)$?**	I=1, **$v(I^{(1)}_1)$?**	O=1,I=1,J=1 **$v(J^{(1)}_1)$?**	E=1 **$v(E^{(1)})$=1**	Q=1 **$v(Q^{(1)})$=1**
$X^{(1)}_2$			O=1, I=1,J=0 **$v(J^{(1)}_2)$?**		
$X^{(1)}_3$			O=1,I=0,J=1 **$v(J^{(1)}_3)$?**		
$X^{(1)}_4$			O=1,I=0,J=0 **$v(J^{(1)}_4)$?**		
$X^{(1)}_5$	O=0, **$v(O^{(1)}_2)$?**	I=0, **$v(I^{(1)}_2)$?**	O=0,I=1,J=1 **$v(J^{(1)}_5)$?**		
$X^{(1)}_6$			O=0, I=1,J=0 **$v(J^{(1)}_6)$?**		
$X^{(1)}_7$			O=0,I=0,J=1 **$v(J^{(1)}_7)$?**		
$X^{(1)}_8$			O=0,I=0,J=0 **$v(J^{(1)}_8)$?**		
	$O^{(2)}$	$I^{(2)}$	$J^{(2)}$	$E^{(2)}$	$Q^{(2)}$
$X^{(2)}_1$	O=1, **$v(O^{(2)}_1)$?**	I=1, **$v(I^{(2)}_1)$?**	O=1,I=1,J=1 **$v(J^{(2)}_1)$?**	E=1 **$v(E^{(2)})$=1**	**$Q^{(2)}$=0** **$v(Q^{(2)})$=1**
$X^{(2)}_2$			O=1, I=1,J=0 **$v(J^{(2)}_2)$?**		
$X^{(2)}_3$			O=1,I=0,J=1 **$v(J^{(2)}_3)$?**		
$X^{(2)}_4$			O=1,I=0,J=0 **$v(J^{(2)}_4)$?**		
$X^{(2)}_5$	O=0, **$v(O^{(2)}_2)$?**	I=0, **$v(I^{(2)}_2)$?**	O=0,I=1,J=1 **$v(J^{(2)}_5)$?**		
$1X^{(2)}_6$			O=0, I=1,J=0 **$v(J^{(2)}_6)$?**		
$X^{(2)}_7$			O=0,I=0,J=1 **$v(J^{(2)}_7)$?**		
$X^{(2)}_8$			O=0,I=0,J=0 **$v(J^{(2)}_8)$?**		

Missing data $\{v(O^{(1)}_i), v(I^{(1)}_i), v(J^{(1)}_i)\}$ and $\{v(O^{(2)}_i), v(I^{(2)}_i), v(J^{(2)}_i)\}$ are computed by expected value of f_{ij}. For example, $v(J^{(1)}_3) = Pr(J=1|PA(J)) = Pr(J=1|O=1, I=0) = E(f_{32}) = a_{32}/(a_{32}+b_{32}) = 15/(15+10) = 3/5$.

Table III.1.13. Estimating missing data

$v(O^{(1)}_1)=E(f_{11})=1/2$	$v(I^{(1)}_1)=E(f_{21})=1/2$	$v(J^{(1)}_1)=E(f_{31})=1$	$v(E^{(1)})=1$	$v(Q^{(1)})=1$
		$v(J^{(1)}_2)=1- v(J^{(2)}_1)=0$		
		$v(J^{(1)}_3)=E(f_{32})=3/5$		
		$v(J^{(1)}_4)=1- v(J^{(2)}_3)=2/5$		
$v(O^{(1)}_2)=1- v(O^{(1)}_1)=1/2$	$v(I^{(1)}_2)=1- v(I^{(1)}_1)=1/2$	$v(J^{(1)}_5)=E(f_{33})=2/5$		
		$v(J^{(1)}_6)?=1- v(J^{(2)}_5)=3/5$		
		$v(J^{(1)}_7)=E(f_{34})=0$		
		$v(J^{(1)}_8)=1- v(J^{(2)}_7)=1$		
$v(O^{(2)}_1)=E(f_{11})=1/2$	$v(I^{(2)}_1)=E(f_{21})=1/2$	$v(J^{(2)}_1)=E(f_{31})=1$	$v(E^{(2)})=1$	$v(Q^{(2)})=1$
		$v(J^{(2)}_2)=1- v(J^{(2)}_1)=0$		
		$v(J^{(2)}_3)=E(f_{32})=3/5$		
		$v(J^{(2)}_4)=1- v(J^{(2)}_3)=2/5$		
$v(O^{(2)}_2)=1- v(O^{(1)}_1)=1/2$	$v(I^{(2)}_2)=1- v(I^{(1)}_1)=1/2$	$v(J^{(2)}_5)=E(f_{33})=2/5$		
		$v(J^{(2)}_6)?=1- v(J^{(2)}_5)=3/5$		
		$v(J^{(2)}_7)=E(f_{34})=0$		
		$v(J^{(2)}_8)=1- v(J^{(2)}_7)=1$		

Suppose s_{ij}, t_{ij} which in turn denote the number of successful and failed evidences according to f_{ij} will be specified. For example, $s_{32}= v(J^{(1)}_3)+ v(J^{(2)}_3)=6/5$, $t_{32}= v(J^{(1)}_4)+ v(J^{(2)}_4)=4/5$

Table III.1.14. All s_{ij} and t_{ij}

$s_{11}=v(O^{(1)}_1)+ v(O^{(2)}_1)=0.5$	$t_{11}=v(O^{(1)}_2)+ v(O^{(2)}_2)=0.5$
$s_{21}=v(I^{(1)}_1)+ v(I^{(2)}_1)=0.5$	$t_{21}=v(I^{(1)}_2)+ v(I^{(2)}_2)=0.5$
$s_{31}=v(J^{(1)}_1)+ v(J^{(2)}_1)=2$	$t_{31}=v(J^{(1)}_2)+ v(J^{(2)}_2)=0$
$s_{32}=v(J^{(1)}_3)+ v(J^{(2)}_3)=6/5$	$t_{32}=v(J^{(1)}_4)+ v(J^{(2)}_4)=4/5$
$s_{33}=v(J^{(1)}_5)+ v(J^{(2)}_5)=4/5$	$t_{33}=v(J^{(1)}_6)+ v(J^{(2)}_6)=6/5$
$s_{34}=v(J^{(1)}_7)+ v(J^{(2)}_7)=0$	$t_{34}=v(J^{(1)}_8)+ v(J^{(2)}_8)=2$
$s_{41}=v(E^{(1)}_1)/2+ v(E^{(2)}_1)/2=1$	$t_{41}= v(E^{(1)}_1)/2+ v(E^{(2)}_1)/2=1$
$s_{42}=v(E^{(1)})/2+ v(E^{(2)})/2=1$	$t_{42}= v(E^{(1)})/2+ v(E^{(2)})/2=1$
$s_{51}=v(Q^{(1)}_1)/2=0.5$	$t_{51}=v(Q^{(2)}_1)/2=0.5$
$s_{52}=v(Q^{(1)}_1)/2=0.5$	$t_{52}= v(Q^{(1)}_1)/2=0.5$

Now we have information enough to compute posterior density functions $\beta(f_{ij}; a'_{ij}, b'_{ij})$

Table III.1.15. All posterior density functions

$\beta(f_{11}; a'_{11}, b'_{11})$	$a'_{11}= a_{11}+s_{11}=50.5$	$b'_{11}= b_{11}+ t_{11}=50.5$
$\beta(f_{21}; a'_{21}, b'_{21})$	$a'_{21}= a_{21}+ s_{21}=50.5$	$b'_{21}= b_{21}+ t_{21}=50.5$
$\beta(f_{31}; a'_{31}, b'_{31})$	$a'_{31}= a_{31}+ s_{31}=27$	$b'_{31}= b_{31}+ t_{31}=0$
$\beta(f_{32}; a'_{32}, b'_{32})$	$a'_{32}= a_{32}+ s_{32}=81/5$	$b'_{32}= b_{32}+ t_{32}=54/5$
$\beta(f_{33}; a'_{33}, b'_{33})$	$a'_{33}= a_{33}+ s_{33}=54/5$	$b'_{33}= b_{33}+ t_{33}=81/5$
$\beta(f_{34}; a'_{34}, b'_{34})$	$a'_{34}= a_{34}+ s_{34}=0$	$b'_{34}= b_{34}+ t_{34}=27$
$\beta(f_{41}; a'_{41}, b'_{41})$	$a'_{41}= a_{41}+ s_{41}=41$	$b'_{41}= b_{41}+ t_{41}=11$
$\beta(f_{42}; a'_{42}, b'_{42})$	$a'_{42}= a_{42}+ s_{42}=11$	$b'_{42}= b_{42}+ t_{42}=41$
$\beta(f_{51}; a'_{51}, b'_{51})$	$a'_{51}= a_{51}+ s_{51}=10.5$	$b'_{51}= b_{51}+ t_{51}=40.5$
$\beta(f_{52}; a'_{52}, b'_{52})$	$a'_{52}= a_{52}+ s_{52}=40.5$	$b'_{52}= b_{52}+ t_{52}=10.5$

If we continue to apply EM algorithm whenever observed evidences are raised, the posterior density functions are updated and become more accurate after many iterations because the $\lim_{k\to\infty}\frac{a_{ij}+s_{ij}^{(k)}}{a_{ij}+s_{ij}^{(k)}+b_{ij}+t_{ij}^{(k)}}$ will gains certain value.

BN is a powerful mathematical tool for reasoning but it is restricted by unimproved initial parameters. This section III.1.3 suggests the approach to parameter evolution that uses the EM algorithm for *beta* functions. Note that the particular features of beta function make this suggestion feasible because it is possible to compute the expectation of *beta* function which is the conditional probability in BN. Whether the EM converges quickly or not depends on how to pre-define the parameters. So, I specify the initial parameters (a_{ij}, b_{ij}) by weights of arcs.
However, the qualitative model (graph structure) is now fixed. It is more creative to apply learning machine algorithms to enhance entirely the structure of BN. That is learning structure process which will be represented in next section.

III.1.4. Improving knowledge sub-model by using dynamic Bayesian network

Normal Bayesian network (BN) described in previous section is effective approach to make reasoning on knowledge model but dynamic Bayesian network (DBN) [Neapolitan 2003] is more robust than BN for modeling users' knowledge when DBN allows monitoring user's process of gaining knowledge and evaluating her/his knowledge. 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.

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. Our 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)".

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. Before describing the way to improve Bayesian overlay model by using dynamic Bayesian network in III.1.4.2, we should glance over what dynamic Bayesian network is in III.1.4.1. Section III.1.4.3 is the evaluation.

III.1.4.1. Dynamic Bayesian network

BN provides a powerful inference mechanism based on evidences but it cannot model temporal relationships between variables. It only represents 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; 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 variable representing the value of x_i at time t where $0 \leq t \leq T$. Let $X[t]$ be the temporal random vector denoting the random vector X at time t, $X[t] = \{x_1[t], x_2[t], \ldots, x_n[t]\}$. A DBN is defined as a BN containing variables that comprise T variable vectors $X[t]$ and determined by following specifications:

- An initial BN $G_0 = \{X[0], Pr(X[0]\}$ at first time $t = 0$
- A transition BN is a template consisting of a transition DAG $G_{\rightarrow}$ containing variables in $X[t] \cup X[t+1]$ and a transition probability distribution $Pr_{\rightarrow}(X[t+1] \mid X[t])$.

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 **D**istributed **G**lobal **J**oint **P**robability **D**istribution of DBN so-called DGJPD is product of probability distribution of G_0 and product of all $Pr_{\rightarrow}$ (s) valuated at all time points, which is denoted following:

$$Pr(X[0], X[1], \ldots, X[T]) = Pr(X[0]) * \prod_{t=0}^{T-1} \Pr_{\rightarrow}(X[t+1] \mid X[t]) \quad (\textit{Formula III.1.22})$$

Note that the transition (temporal) probability can be considered the transition (temporal) dependency.

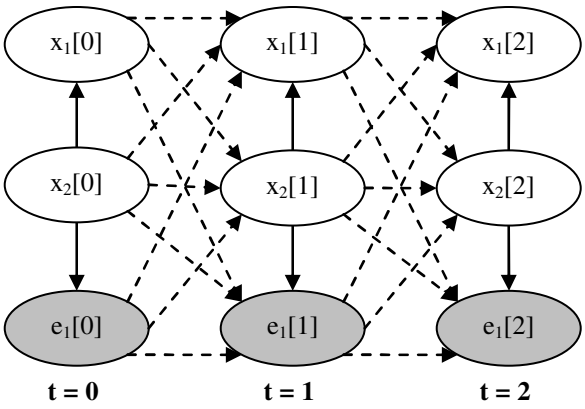


Figure III.1.16. DBN for $t = 0, 1, 2$.
(*Non-evidence variables are not shaded; otherwise, evidence variables are shaded. Dash lines - - - denotes transition probabilities or transition dependencies of $G_{\rightarrow}$ between consecutive points in time*)

Drawbacks of inferences in DBN

Formula III.1.22 is considered as extension of formula III.1.2; so, the posterior probability of each temporal variable is now computed by using DGJPD in formula III.1.22 which is much more complex than normal GJPD in formula III.1.2.

Whenever the posterior of a variable evaluated time point *t* needs to be computed, all temporal random vectors *X*[*0*], *X*[*1*],…, *X*[*t*] must be included for executing Bayesian rule because DGJPD is product of all transition $Pr_{\rightarrow}$ (s) valuated at *t* points in time. Suppose the initial DAG has *n* variables ($X[0] = \{x_1[0], x_2[0],\ldots, x_n[0]\}$), 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 $X[0] \cup X[1] \cup \ldots \cup X[t]$. 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 $x_i[t] \in X[t]$ and variables $x_i[t+1] \in X[t+1]$. Through *t* time points, there are $n*n*t$ transition dependencies. So it is impossible to compute effectively the transition probability distribution $Pr_{\rightarrow}(X[t+1] \mid X[t])$ and the DGJPD in formula III.22.

III.1.4.2. Using dynamic Bayesian network to model user's knowledge

To overcome drawbacks of DBN, I propose the new algorithm that both the size of DBN and the number of CPT(s) in DBN are kept intact (not changed) when the process continues for a long time. However we should glance over some definitions before discussing our method. Given $pa_i[t+1]$ is a set of parents of x_i at time point *t+1*, namely parents of $X_i[t+1]$, the transition probability distribution is computed as below:

$$Pr_{\rightarrow}(X[t+1] \mid X[t]) = \prod_{i=1}^{n} \Pr_{\rightarrow}(x_i[t+1] \mid pa_i[t+1]) \quad (\textit{Formula III.1.23})$$

Applying (5) for all *X* and for all *t*, we have:
$Pr_{\rightarrow}(X[t+1] \mid X[0], X[1],\ldots, X[t]) = Pr_{\rightarrow}(X[t+1] \mid X[t])$ (*Formula III.1.24*)
If the DBN meets fully formula III.1.24, it has Markov property, namely, given the current time point *t*, the conditional probability of next time point *t+1* is only relevant to the current time point *t*, not relevant to any past time point (*t-1, t-2,…,0*). Furthermore, the DBN is stationary if $Pr_{\rightarrow}(X[t+1] \mid X[t])$ is the same for all *t*.

A new algorithm for modeling and inferring user's knowledge by using DBN.

Suppose DBN is stationary and has Markov property. Each time there are occurrences of evidences, DBN is re-constructed and the probabilistic inference is done by six following steps:
- *Step 1*: Initializing DBN.
- *Step 2*: Specifying transition weights.
- *Step 3*: Re-constructing DBN.
- *Step 4*: Normalizing weights of dependencies.
- *Step 5*: Re-defining CPT (s).
- *Step 6*: Probabilistic inference.

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 converge 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 course in which the domain model has four knowledge elements x_1, x_2, x_3, e_1. The item e_1 is the evidence that tells us how learners are mastered over x_1, x_2, x_3. This domain model is represented as a BN having three non-evidence variables x_1, x_2, x_3 and one evidence variable e_1. The weight of an arc from parent variable to child variable represents the strength of dependency among them. In other word, when x_2 and x_3 are prerequisite of x_1, knowing x_2 and x_3 have causal influence in knowing x_1. For instance, the weight of arc from x_2 to x_1 measures the relevant importance of x_2 in x_1. This BN regarded as an example for our algorithm is shown in following figure.

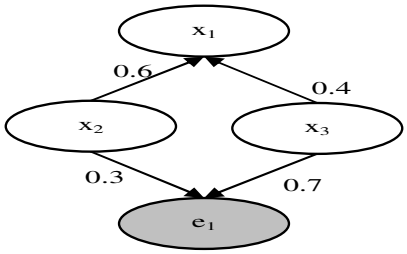


Figure III.1.17. The BN sample

Step 1: Initializing DBN

If *t > 0* then jumping to step *2*. Otherwise, all variables (nodes) and dependencies (arcs) among variables of initial BN G_0 must be specified. The strength of dependency is considered as weight of arc.

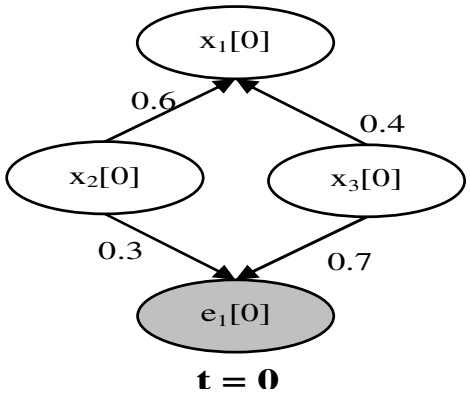


Figure III.1.18. Initial DBN derived from BN in figure III.1.15

Step 2: Specifying transition weight

Given two factors: *slip* and *guess* where *slip* (*guess*) factor expresses the situation that user does (doesn't) know a particular subject but there is solid evidence convincing that she/he doesn't (does) understand it; this evidence just reflects a temporary slip (or lucky guess). *Slip* factor is essentially probability that user has known concept/subject *x* before but she/he forgets it now. Otherwise *guess* factor is essentially probability that user hasn't known concept/subject *x* before but she/he knows it knows. Suppose *x*[*t*] and *x*[*t+1*] denote the user's state of knowledge about *x* at two consecutive time points t_1 and t_2 respectively. Both *x*[*t*] and *x*[*t+1*] are temporal variables referring the same knowledge element *x*.

slip = Pr(*not x*[*t+1*] | *x*[*t*])
guess = Pr(*x*[*t+1*] | *not x*[*t*])
(where $0 \le guess, slip \le 1$)

So the conditional probability (named *a*) of event that user knows *x*[*t+1*] given event that she/he has already known *x*[*t*] has value *1-slip*. Proof,

a = Pr(*x*[*t+1*] | *x*[*t*]) = 1 − *Pr*(*not x*[*t+1*] | *x*[*t*]) = *1 − slip*

The bias *b* is defined as differences of an amount of knowledge user gains about *x* between *t* and *t+1*.

$$b = \frac{1}{1 + \Pr(x[t+1] \mid \overline{x[t]})} = \frac{1}{1 + guess}$$

Now the weight *w* expressing strength of dependency between *x*[*t*] and *x*[*t+1*] is defined as product of the conditional probability *a* and the bias *b*.

$$w = a * b = (1 - slip) * \frac{1}{1 + guess} \quad (\textit{Formula III.1.25})$$

Expanding to temporal random vectors, *w* is considered as the weight of arcs from temporal vector *X*[*t*] to temporal vector *X*[*t+1*]. Thus the weight *w* implicates the conditional transition probability of *X*[*t+1*] given *X*[*t*].

$w \Leftrightarrow Pr_{\rightarrow}(X[t+1] \mid X[t]) = Pr_{\rightarrow}(X[t] \mid X[t-1])$

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 $w = (1-0.3)*\frac{1}{1+0.2} = 0.58$

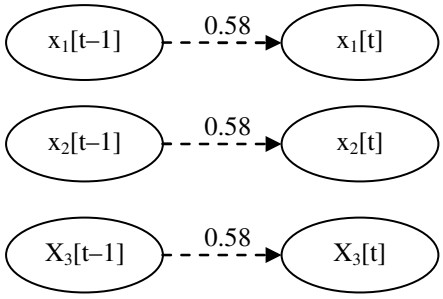


Figure III.1.19. Transition weights

Step 3: Re-constructing DBN

Because our DBN is stationary and has Markov property, 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 augmented BN at time point *t*. The set of such variables is denoted *Y*.

$Y = X[t-1] \cup X[t] / E[t-1] = \{x_1[t-1], x_2[t-1],\ldots, x_n[t-1], x_1[t], x_2[t],\ldots, x_n[t]\} / \{e_1[t-1], e_2[t-1],\ldots, e_k[t-1]\}$ where $E[t-1]$ is the set of evidences at time point $t-1$

A very important fact to which you should pay attention is that all conditional dependencies among variables in $X[t-1]$ are removed from $G[t]$. It means that no arc (or CPT) in $X[t-1]$ 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 weight *w* specified in formula III.1.25. Hence every $x_i[t]$ in $X[t]$ has a parent which in turn is a variable in $X[t-1]$ and the temporal relationship among them are weighted. Vector $X[t-1]$ becomes the input of vector $X[t]$.

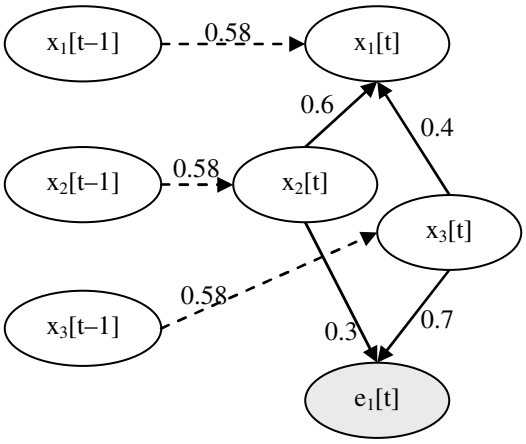


Figure III.1.20. Augmented DBN at time point *t*.

(*Dash lines - - - denotes transition dependencies. The augmented DBN is much simpler than DBN in figures III.1.14.*)

Step 4: Normalizing weights of dependencies

Suppose $x_1[t]$ has two parents $x_2[t]$ and $x_3[2]$. The 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 dependencies inside random $X[t]$.

$w_2 + w_3 = 1$

Now in augmented DBN, the transition weight of temporal arc from $x_1[t-1]$ to $x_1[t]$ is specified according to formula III.1.25

$$w_1 = a * b = (1 - slip) * \frac{1}{1 + guess}$$

The weights w_1, w_2, w_3 must be normalized because sum of them is larger than *1*, $w_1 + w_2 + w_3 > 1$

$w_2 = w_2 * (1-w_1)$, $w_3 = w_3 * (1-w_1)$ (*Formula III.1.26*)

Suppose S is the sum of w_1, w_2 and w_3, we have:

$S = w_1 + w_2 * (1-w_1) + w_3 * (1-w_1) = w_1 + (w_2+w_3)(1-w_1) = w_1 + (1-w_1) = 1$.

Expending formula III.1.26 on general cases, suppose variable $x_i[t]$ has $k-1$ weights $w_{i2}, w_{i3},\ldots, x_{ik}$ 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:

$w_{i2}=w_{i2}*(1-w_{i1}), w_{i3}=w_{i3}*(1-w_{i1}),\ldots, w_{ik}=w_{ik}*(1-w_{i1})$ (*Formula III.1.27*)

After normalizing weights following formula III.1.27, transition weight w_{i1} is kept intact but other weights w_{ij} ($j > 1$) get smaller. So the meaning of formula III.1.27 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 weights relevant to a variable $x_i[t]$, we have:

$W_i[t] = \{w_{i1}, w_{i2}, w_{i3},\ldots, w_{ik}\}$ where $w_{i1} + w_{i2} +\ldots+ w_{ik} = 1$

Table III.1.16. The weights relating $x_i[t]$ are normalized

	w_{11}	w_{12}	w_{13}
$x_1[t]$	0.58	0.6	0.4
$x_1[t]$ (*normalized*)	0.58	0.252	0.168

The following figure shows the variant of augmented DBN (in figure III.1.18) whose weights are normalized.

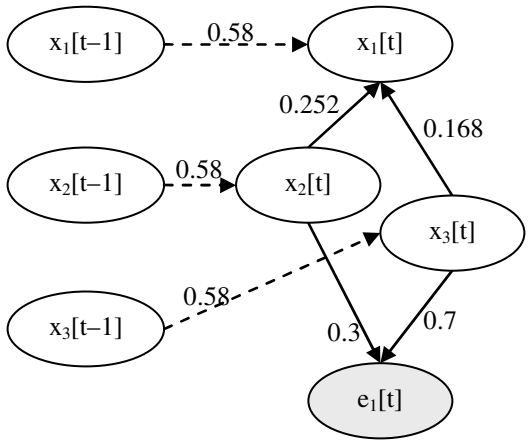


Figure III.1.21. Augmented DBN whose weights are normalized

Step 5: Re-defining CPT(s)

There are two random vectors $X[t-1]$ and $X[t]$. So defining CPT(s) of DBN includes: determining CPT for each variable $x_i[t-1] \in X[t-1]$ and re-defining CPT for each variable $x_i[t] \in X[t]$.

1. *Determining CPT*(*s*) *of* $X[t-1]$. The CPT of $x_i[t-1]$ is the posterior probabilities which were computed in step *6* of previous iteration.

$$\Pr(x_i[t-1] \mid E[t-1]) = \frac{\sum_{X/\{x_i \cup E\}} \Pr(x_1[t-1], x_2[t-1],..., x_n[t-1])}{\sum_{X/E} \Pr(x_1[t-1], x_2[t-1],..., x_n[t-1])} \quad (\textit{see step 6})$$

Table III.1.17. CPT of $x_1[t\text{-}1]$.

Pr(x_1[t-1]=1)	**Pr(x_1[t-1]=0)**
α_1: *the posterior probability of x_1 computed at previous iteration*	**$1 - \alpha_1$**

Table III.1.18. CPT of $x_2[t\text{-}1]$.

Pr(x_2[t-1]=1)	**Pr(x_2[t-1]=0)**
α_2: *the posterior probability of x_2 computed at previous iteration*	**$1 - \alpha_2$**

Table III.1.19. CPT of $x_3[t\text{-}1]$.

Pr(x_3[t-1]=1)	**Pr(x_3[t-1]=0)**
α_3: *the posterior probability of x_3 computed at previous iteration*	**$1 - \alpha_3$**

2. *Re-defining CPT(s) of X[t]*. Suppose $pa_i[t] = \{y_1, x_2, \ldots, x_k\}$ is a set of parents of $x_i[t]$ at time point t and $W_i[t] = \{w_{i1}, w_{i2}, \ldots, w_{ik}\}$ is a set of weights which expresses the strength of dependencies between x_i and such $pa_i[t]$. 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 $Pr(x_i[t] \mid pa_i[t])$. So $Pr(x_i[t] \mid pa_i[t])$ represents the CPT of $x_i[t]$.

$$\Pr(x_i[t] = 1 \mid pa_i[t]) = \sum_{j=1}^{k} w_{ij} * h_{ij} \text{ where } \mathrm{h_{ij}} = \begin{cases} 1 \text{ if } \mathrm{y_{ij}} = x_i[t] = 1 \\ 0 \text{ otherwise} \end{cases}$$

$Pr(x_i[t]=0 \mid pa_i[t]) = 1 - Pr(x_i[t]=1 \mid pa_i[t])$

Table III.1.20. CPT of $x_1[t]$.

x_1[t-1]	**x_2[t]**	**x_3[t]**	**Pr(x_1[t]=1)**	**Pr(x_1[t]=0)**
1	1	1	**1.0** (*0.58*1+0.252*1+0.168*1*)	**0.0**
1	1	0	**0.832** (*0.58*1+0.252*1+0.168*0*)	**0.168**
1	0	1	**0.748** (*0.58*1+0.252*0+0.168*1*)	**0.252**
1	0	0	**0.58** (*0.58*1+0.252*0+0.168*0*)	**0.42**
0	1	1	**0.42** (*0.58*0+0.252*1+0.168*1*)	**0.58**
0	1	0	**0.252** (*0.58*0+0.252*1+0.168*0*)	**0.748**
0	0	1	**0.168** (*0.58*0+0.252*0+0.168*1*)	**0.832**
0	0	0	**0.0** (*0.58*0+0.252*0+0.168*0*)	**1.0**

Table III.1.21. CPT of $x_2[t]$.

x_2[t-1]	**Pr(x_2[t]=1)**	**Pr(x_2[t]=0)**
1	**0.58** (*0.58*1*)	**0.42**
0	**0.0** (*0.58*0*)	**1.0**

Table III.1.22. CPT of $x_3[t]$.

$x_3[t-1]$	$Pr(x_3[t]=1)$	$Pr(x_3[t]=0)$
1	**0.58** (*0.58*1*)	**0.42**
0	**0.0** (*0.58*0*)	**1.0**

Table III.1.23. CPT of $e_1[t]$.

$Pr(e_1[t]=1)$	$Pr(e_1[t]=0)$
0.5 (*use uniform distribution*)	**0.5** (*use uniform distribution*)

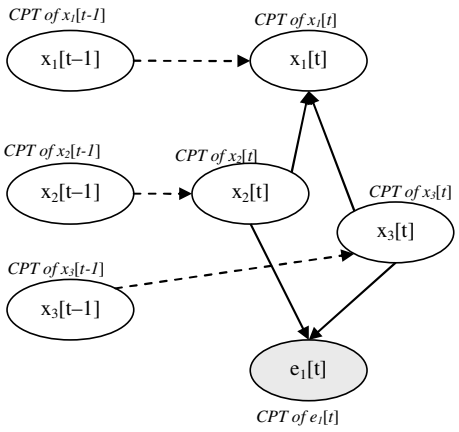


Figure III.1.22. Augmented DBN and its CPT (s)

Step 6: Probabilistic inference

The probabilistic inference in our augmented DBN can be done similarly to normal Bayesian network by using the formulas III.1.5, III.1.6. It is essential to compute the posterior probabilities of non-evidence variable in $X[t]$. This decrease significantly expense of computation regardless of a large number of variables in DBN for a long time. At any time point, it is only to examine *2*n* variables if the DAG has *n* variables instead of including *2*n*t* variables and *n*n*t* transition probabilities given time point *t*. Each posterior probability of $x_i[t] \in X[t]$ is computed below.

$$Pr(x_i[t]) = \Pr(x_i[t] \mid E[t]) = \frac{\sum_{X/\{x_i \cup E\}} \Pr(x_1[t], x_2[t], ..., x_n[t])}{\sum_{X/E} \Pr(x_1[t], x_2[t], ..., x_n[t])}$$ where $E[t]$ is a set of evidences occurring at time point *t*.

Such posterior probabilities are also used for determining CPT (s) *of DBN in step 5 of next iteration*. For example, posterior probabilities of $x_1[t]$, $x_2[t]$ and $x_3[t]$ are α_1, α_2 and α_3 respectively. Note that it is not required to compute the posterior probabilities of $X[t-1]$. If the posterior probabilities are the same as before (previous iteration) then DBN converges when all posterior probabilities of variables $x_i[t]$ gain stable values at any time. If so we can stop algorithm; otherwise turning back step 1.

Table III.1.24. The results of probabilistic inference – *posterior probabilities* are used for determining CPT(s) of DBN in step *5* of next iteration.

$Pr(x_1[t])$	α_1
$Pr(x_2[t])$	α_2
$Pr(x_3[t])$	α_3

III.1.4.3. Evaluation

Our basic idea is to minimize the size of DBN and the number of transition probabilities in order to decrease expense of computation when the process of inference continues for a long time. Suppose DBN is stationary and has Markov property, I define two factors: *slip* & *guess* to specify the same weight for all transition relationships (temporal relationship) among time points instead of specifying a large number of transition probabilities. The augmented DBN composed at given time point *t* has just two random vectors *X*[*t–1*] and *X*[*t*]; so , it is only to examine *2*n* variables if the DAG has *n* variables instead of including *2*n*t* variables and *n*n*t* transition probabilities. That specifying *slip* factor and *guess* factor will solve the problem of temporary slip and lucky guess.

The process of inference including six steps is done in succession through many iterations, the result of current iteration will be input for next iteration. After t^{th} iteration DBN will converge when the posterior probabilities of all variables $x_i[t]$ gain stable values regardless of the occurrence of a variety evidences.

Instead of using DBN to re-constructing network, another approach described in next section 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.

III.1.5. Specifying prior probabilities

Bayesian network provides the solid inference mechanism when convincing the hypothesis by collecting evidences. Bayesian network is instituted of two models: qualitative model quantitative model. The qualitative model is its structure and the quantitative model is its parameters, namely conditional probability tables (CPT) whose entries are probabilities quantifying the dependences among variables in network. 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 our method. The problem of defining prior probabilities is involved in how to estimate parameters in beta distribution. In this report, I propose the formula for estimating beta distribution's parameters by applying the Maximum Likelihood Estimation (MLE) technique. Such optimal parameters that we must find out are called parameter estimators. It is slightly unfortunate when the equations whose solutions are parameter estimators are differential equations and it is too difficult to solve them. Thus I also propose the algorithm so as to find out the approximate solutions of these equations.

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 the dependence relationships among nodes. The strength of dependence relationship is quantified by conditional probabilities. Each node owns a conditional probability table (CPT) that measures the impact of all its parents on it. Such CPT (s) are called the parameters of BN. Note that each entry in a CPT is a conditional probability. The problem which must be solved is how to initialize these parameters so as to be optimal. It means that we should specify prior probability.

Suppose every node *X* in BN obeys beta distribution with two parameters *a* and *b,* we have:

$$f(X,a,b)=\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}X^{a-1}(1-X)^{b-1}$$

Where *X* is random variable and *Γ* denotes the gamma function described below:

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

The integral will converges if *x>0*, at that time, $\Gamma(x)=(x-1)!$. Of course, we have $\frac{\Gamma(x+1)}{\Gamma(x)}=x$. We should consider gamma function carefully because it relates to parameter estimation.

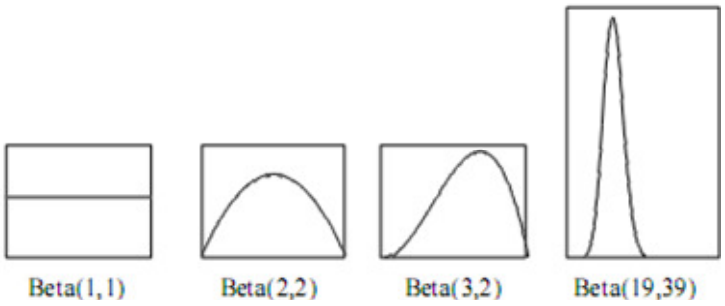


Figure III.1.23. Beta functions

It means that, there are "*a*" successful outcomes (for example, *X =1*) in "*a+b*" trials. Higher value of "*a*" is, higher ratio of success is, so, the graph leans forward right. Higher value of "*a+b*" is, the more the mass concentrates around $\frac{a}{a+b}$ and the more narrow the graph is. The expectation and variance of beta distribution are defined as below:

$$E(X) = \frac{a}{a+b}$$

$$Var(X) = \frac{ab}{(a+b)^2(a+b+1)}$$

The reason I choose beta function as the probability distribution for every node (variable) in BN is that the prior probability of *X* is the expectation of beta function and this value is very easy to compute:

$$\Pr(X=1) = E(X) = \frac{a}{a+b}$$

We need to compute the posterior probability of *X* denoted as *Pr*(*X=1*|*E*) where *E* 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*.

$$\Pr(X=1 \mid E) = E(X \mid E) = \frac{a+s}{a+b+s+t} = \frac{a+s}{N+M}$$

Where *N=a+b* and *M=s+t*

I recognize that the beta distribution provide us some 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, $\Pr(X=1) = E(X) = \frac{a}{a+b}$. The following sections will discuss about the maximum likelihood estimation (MLE) technique and how to use it to estimate two parameters of beta distribution.

III.1.5.1. Maximum likelihood estimation

Let *θ* and *X* be the hypothesis and observation variable, respectively. Suppose $x_1, x_2, \ldots, x_n$ are instances of variable *X* in training data and they are observed independently. According multiplication rule in probability theory, the likelihood function *L*(*θ*) is the joint probability which is the product of condition probabilities of instances x_i, given hypothesis variable

$$L(\theta) = \Pr(X \mid \theta) = \prod_{i=1}^{n} \Pr(x_i \mid \theta)$$

Where *Pr*(x_i|*θ*) is the conditional probability of instance x_i given the hypothesis.

Suppose $\theta=\{\theta_1, \theta_2,\ldots, \theta_k\}$ is the vector of parameters specifying the distribution *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*. The parameter vector that maximizes likelihood function is called optimal parameter vector denoted as $\widehat{\theta}$.

$$\widehat{\theta} = \arg\max_{\theta} L(\theta) = \arg\max_{\theta} \prod_{i=1}^{n} \Pr(x_i \mid \theta)$$

Because it is too difficult to work with the likelihood function in the form of product of condition probabilities, we should take the logarithm of $L(\theta)$ so that the log function converts the repeated multiplication to repeated addition. The logarithm of $L(\theta)$ so-called log-likelihood is denoted $LnL(\theta)$.

$$LnL(\theta) = Ln\prod_{i=1}^{n} \Pr(x_i \mid \theta) = \sum_{i=1}^{n} \ln \Pr(x_i \mid \theta)$$

$$\widehat{\theta} = \arg\max_{\theta} LnL(\theta) = \arg\max_{\theta} \sum_{i=1}^{n} \ln \Pr(x_i \mid \theta)$$

The essence of maximizing the likelihood function is to find the peak of the curve of $LnL(\theta)$. 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 reason is that the slope of the curve $LnL(\theta)$ equals *0* at its peak. The number of equations corresponds with the number of parameters. If all parameters are found, in other words, the optimal parameter vector $\widehat{\theta}=\{\widehat{\theta}_1, \widehat{\theta}_2,\ldots, \widehat{\theta}_k\}$ is defined then the optimal distribution $f(\widehat{\theta})$ is known clearly. Each $\widehat{\theta}_i$ is also called a parameter estimator.

Another important issue is how to determine the standard deviation in distributed *f* when we have already computed all parameters. It is very fortunate when the second-order derivative of the log-likelihood function denoted $\frac{\partial^2 LnL}{\partial\theta\partial\theta^T}$ can be computed and it is used to determine the variances of parameters. If distribution *f* has only one parameter, the second-order derivative $\frac{\partial^2 LnL}{\partial\theta\partial\theta^T}$ is scalar, otherwise it is a matrix so-called Hessian matrix. The negative expectation of Hessian matrix is called the information matrix which in turn is inverted so as to construct the matrix containing the variances of the parameters on its diagonal, and the asymptotic co-variances of the parameters in the off-diagonal positions. Such matrix is called variance matrix. The standard deviation is the root of variance.

$$Var(\theta) = (-E(\frac{\partial^2 LnL}{\partial\theta\partial\theta^T}))^{-1}$$

$$\sigma(\theta) = \sqrt{Var(\theta)}$$

III.1.5.2. Beta likelihood estimation

As discussed, the beta distribution is defined as below:

$$f(X,a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} X^{a-1}(1-X)^{b-1} \quad \text{where } \Gamma(x) = \int_0^{\infty} t^{x-1}e^{-t}dt \quad (\textit{Formula III.1.28})$$

The beta function denoted $B(x, y)$ is a special function defined as below:

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \quad (\textit{Formula III.1.29})$$

The first-order partial derivative of $B(x, y)$ is determined:

$$\frac{\partial}{\partial x}B(x,y) = B(x,y)(\frac{\Gamma^{'}(x)}{\Gamma(x)} - \frac{\Gamma^{'}(x+y)}{\Gamma(x+y)})$$

Let $\psi(x) = \frac{\Gamma^{'}(x)}{\Gamma(x)}$, we have: $\frac{\partial}{\partial x}B(x,y) = B(x,y)(\psi(x) - \psi(x+y))$

(*Formula III.1.30*)

The $\psi(x)$ is also called digamma function. Applying formula III.1.29, the beta distribution is re-written:

$$f(X,a,b) = \frac{1}{B(a,b)}X^{a-1}(1-X)^{b-1} \quad \text{(Formula III.1.31)}$$

Now we specify the likelihood function of beta distribution as below:

$$L(\theta) = \prod_{i=1}^{n} f(x_i,a,b) = \prod_{i=1}^{n} \frac{1}{B(a,b)} x_i^{a-1}(1-x_i)^{b-1} = \frac{1}{B^n(a,b)} \prod_{i=1}^{n} x_i^{a-1} \prod_{i=1}^{n} (1-x_i)^{b-1}$$

Take the logarithm of $L(\theta)$, we have the log-likelihood function:

$$LnL(f) = n\ln\frac{1}{B(a,b)} + \sum_{i=1}^{n}(a-1)\ln x_i + \sum_{i=1}^{n}(b-1)\ln(1-x_i)$$

$$= n\ln\frac{1}{B(a,b)} + (a-1)\sum_{i=1}^{n}\ln x_i + (b-1)\sum_{i=1}^{n}\ln(1-x_i) \quad \text{(Formula III.1.32)}$$

$$= -n\ln B(a,b) + (a-1)\sum_{i=1}^{n}\ln x_i + (b-1)\sum_{i=1}^{n}\ln(1-x_i)$$

There are two parameters *a* and *b* which we must specify so as to maximize the log-likelihood function. Thus, two first-order partial derivatives corresponding to two parameters must be taken.

$$\frac{\partial LnL}{\partial a} = -n\frac{1}{B(a,b)}\frac{\partial}{\partial a}B(a,b) + \sum_{i=1}^{n}\ln x_i = -n(\psi(a) - \psi(a+b)) + \sum_{i=1}^{n}\ln x_i$$

(*Formula III.1.33*)

$$\frac{\partial LnL}{\partial b} = -n\frac{1}{B(a,b)}\frac{\partial}{\partial b}B(a,b) + \sum_{i=1}^{n}\ln(1-x_i) = -n(\psi(b) - \psi(a+b)) + \sum_{i=1}^{n}\ln(1-x_i)$$

(*Formula III.1.34*)
(By applying formula III.1.30 and III.1.32)

Setting two partial derivatives equal *0* so as to find out two parameters *a* and *b*, we have set of equations whose two solutions are the values of *a* and *b*:

$$\begin{cases} \frac{\partial LnL}{\partial a} = 0 \\ \frac{\partial LnL}{\partial b} = 0 \end{cases} \Leftrightarrow \begin{cases} -n(\psi(a) - \psi(a+b)) + \sum_{i=1}^{n} \ln x_i = 0 \\ -n(\psi(b) - \psi(a+b)) + \sum_{i=1}^{n} \ln(1-x_i) = 0 \end{cases}$$

(*Formula III.1.35*)

$$\Leftrightarrow \begin{cases} \psi(a) - \psi(a+b) = \frac{1}{n}\sum_{i=1}^{n} \ln x_i \\ \psi(b) - \psi(a+b) = \frac{1}{n}\sum_{i=1}^{n} \ln(1-x_i) \end{cases}$$

This is the set of differential equations; so we will focus on the existence of solutions of such equations later. Suppose two solutions of this set of equations are $\widehat{a}$ and $\widehat{b}$ which are estimators of *a*, *b* in beta distribution. So $\widehat{a}$ and $\widehat{b}$ are optimal parameters. The optimal beta distribution becomes as below:

$$f(X, \widehat{a}, \widehat{b}) = \frac{1}{B(\widehat{a}, \widehat{b})} X^{\widehat{a}-1}(1-X)^{\widehat{b}-1}$$ (*Formula III.1.36*)

Now we must determine the second-order partial derivative matrix so-called Hessian matrix.

$$H(\theta) = \begin{bmatrix} \frac{\partial LnL}{\partial^2 a} & \frac{\partial LnL}{\partial a \partial b} \\ \frac{\partial LnL}{\partial b \partial a} & \frac{\partial LnL}{\partial^2 b} \end{bmatrix}$$ (*Formula III.1.37*)

Applying formula III.1.33 and III.1.34, we can determine four second-order partial derivatives.

$$\frac{\partial LnL}{\partial^2 a} = \frac{\partial}{\partial a}\psi(a+b) - n\psi'(a)$$

$$\frac{\partial LnL}{\partial a \partial b} = \frac{\partial}{\partial b}\psi(a+b)$$

$$\frac{\partial LnL}{\partial b \partial a} = \frac{\partial}{\partial a}\psi(a+b)$$

$$\frac{\partial LnL}{\partial^2 b} = \frac{\partial}{\partial b}\psi(a+b) - n\psi'(b)$$

Suppose $\frac{\partial LnL}{\partial^2 a}, \frac{\partial LnL}{\partial a \partial b}, \frac{\partial LnL}{\partial b \partial a}, \frac{\partial LnL}{\partial^2 b}$ are denoted as $h_1(a,b)$, $h_2(a,b)$, $h_3(a,b)$, $h_4(a,b)$, respectively. The Hessian matrix is re-written:

$$H(\theta) = \begin{bmatrix} h_1(a,b) & h_2(a,b) \\ h_3(a,b) & h_4(a,b) \end{bmatrix}$$

The information matrix which is the negative expectation of Hessian matrix is determined as below:

$$H(\theta) = \begin{bmatrix} -h_1(a,b) & -h_2(a,b) \\ -h_3(a,b) & -h_4(a,b) \end{bmatrix}$$

Suppose the subtraction $h_1(a,b)*h_4(a,b) - h_2(a,b)*h_3(a,b)$ does not equal *0*, the inverse of $H(\theta)$ containing the variances of parameters exists.

$$H(\theta)^{-1} = \begin{bmatrix} \dfrac{-h_4(a,b)}{h_1(a,b)*h_4(a,b)-h_2(a,b)*h_3(a,b)} & \dfrac{-h_2(a,b)}{h_2(a,b)*h_3(a,b)-h_1(a,b)*h_4(a,b)} \\ \dfrac{h_3(a,b)}{h_1(a,b)*h_4(a,b)-h_2(a,b)*h_3(a,b)} & \dfrac{h_1(a,b)}{h_2(a,b)*h_3(a,b)-h_1(a,b)*h_4(a,b)} \end{bmatrix}$$

The roots of diagonal elements are the standard deviations (or standard errors) of parameter estimates. Let σ($\widehat{a}$) and σ($\widehat{b}$) be the standard errors of optimal parameters $\widehat{a}$ and $\widehat{b}$, respectively. Suppose:

$\forall a$, $\dfrac{-h_4(a,b)}{h_1(a,b)*h_4(a,b)-h_2(a,b)*h_3(a,b)}$ >0 and

$\dfrac{h_1(a,b)}{h_2(a,b)*h_3(a,b)-h_1(a,b)*h_4(a,b)}$ >0,

We have:

$$\sigma(\widehat{a}) = \sqrt{\frac{-h_4(\widehat{a},\widehat{b})}{h_1(\widehat{a},\widehat{b})*h_4(\widehat{a},\widehat{b})-h_2(\widehat{a},\widehat{b})*h_3(\widehat{a},\widehat{b})}}$$

$$\sigma(\widehat{b}) = \sqrt{\frac{h_1(\widehat{a},\widehat{b})}{h_2(\widehat{a},\widehat{b})*h_3(\widehat{a},\widehat{b})-h_1(\widehat{a},\widehat{b})*h_4(\widehat{a},\widehat{b})}}$$

III.1.5.3. Algorithm to solve the equations whose solutions are parameter estimators

As discussed the parameter estimators $\widehat{a}$ and $\widehat{b}$ are solutions of two equations:

$$\begin{cases} \psi(a)-\psi(a+b) = \dfrac{1}{n}\sum_{i=1}^{n}\ln x_i \\ \psi(b)-\psi(a+b) = \dfrac{1}{n}\sum_{i=1}^{n}\ln(1-x_i) \end{cases} \quad \text{where } \psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$$

Obviously, these equations are differential functions whose solutions are families of function. Because it is too difficult to solve them, I propose the algorithm so as to find out the approximate solutions of these equations. The digamma function $\psi(x)$ is written as below:

$$\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)} = \int_0^{+\infty}\left(\frac{e^{-t}}{t} - \frac{e^{-xt}}{1-e^{-t}}\right)dt$$

$$\left(\text{due to } \Gamma(x) = \int_0^{+\infty}(t^{x-1}e^{-t})dt\right)$$

Supposing *a* and *b* are positive whole number. Expanding the expression $\psi(a) - \psi(a+b)$, we have:

$$\psi(a)-\psi(a+b)=\int_0^{+\infty}(\frac{e^{-t}}{t}-\frac{e^{-at}}{1-e^{-t}})dt-\int_0^{+\infty}(\frac{e^{-t}}{t}-\frac{e^{-(a+b)t}}{1-e^{-t}})dt$$

$$=\int_0^{+\infty}\frac{e^{-at}(e^{-bt}-1)}{1-e^{-t}}dt=e^a\int_0^{+\infty}(e^{-bt}-1)\frac{e^{-t}}{1-e^{-t}}dt$$

$$=e^a\int_{-\infty}^{0}((1-e^x)^b-1)dx \qquad (\text{where } x=\ln(1-e^{-t}))$$

$$=e^a\int_{-\infty}^{0}(\sum_{k=1}^{b}(-1)^k C_b^k e^{kx})dx=e^a\sum_{k=1}^{b}((-1)^k C_b^k\int_{-\infty}^{0}e^{kx}dx)$$

$$=e^a\sum_{k=1}^{b}\frac{(-1)^k}{k}C_b^k$$

Where C_n^k is the combination taken *k* of *n* elements, $C_n^k=\frac{n!}{k!(n-k)!}$

The equations whose solutions are parameter estimators becomes two following equations:

$$\begin{cases}e^a\sum_{k=1}^{b}\frac{(-1)^k}{k}C_b^k=\frac{1}{n}\sum_{i=1}^{n}\ln x_i\\ e^b\sum_{k=1}^{a}\frac{(-1)^k}{k}C_a^k=\frac{1}{n}\sum_{i=1}^{n}\ln(1-x_i)\end{cases}\Leftrightarrow\begin{cases}F_1(a,b)=L_1\\F_2(a,b)=L_2\end{cases}\quad(\textit{Formula III.1.38})$$

Where $F_1(a, b) = e^a\sum_{k=1}^{b}\frac{(-1)^k}{k}C_b^k$, $F_2(a, b)= e^b\sum_{k=1}^{a}\frac{(-1)^k}{k}C_a^k$, $L_1= \frac{1}{n}\sum_{i=1}^{n}\ln x_i$ and $L_2= \frac{1}{n}\sum_{i=1}^{n}\ln(1-x_i)$

The number of solutions of equations in formula III.1.38 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 F_1, F_2 at each iteration. Two biases $\Delta_1=F_1(a_i, b_i)-L_1$ and $\Delta_2=F_2(a_i, b_i)-L_2$ are computed. The normal bias is the root of sum of the second power Δ_1 and the second power of Δ_2: $\Delta=\sqrt{\Delta_1^2+\Delta_2^2}$. The pair ($\widehat{a}$, $\widehat{b}$) whose normal bias Δ is minimum are chosen as the parameter estimators. The algorithm is described as below:

min Δ = +∞ ;
$\widehat{a}=\widehat{b}=1$ *(uniform distribution)*
For a = 1 to n do
For b=1 to n do
$\Delta_1=F_1(a, b)-L_1$
$\Delta_2=F_2(a, b)-L_2$
$\Delta=\sqrt{\Delta_1^2+\Delta_2^2}$
If Δ<min Δ then
min Δ= Δ
$\widehat{a}=a$
$\widehat{b}=b$
End If
End for a
End For b
($\widehat{a}$ *and* $\widehat{b}$ *are optimal parameters)*

Note that the factorial functions occurring in F_1 and F_2 becomes 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, $Pr(X) = E(beta(a,b)) = \frac{a}{a+b}$. 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.

III.1.5.4. An example of how to specify prior probabilities

Suppose there is the BN having two variables X_1, X_2 and one arc which links them together. Variables X_1 and X_2 obey beta distribution. We need to specify the prior CPT (s), namely the prior conditional probabilities $Pr(X_1=1)$, $Pr(X_2=1|X_1=1)$ and $Pr(X_2=1|X_1=0)$.

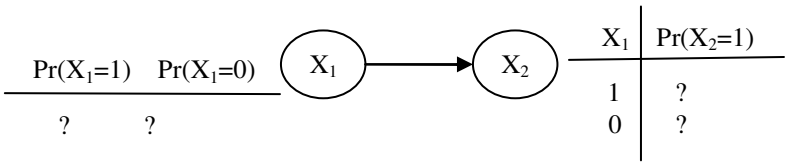


Figure III.1.24. Bayesian network without CPT (s)

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 $Pr(X_1=1)$, $Pr(X_2=1|X_1=1)$ and $Pr(X_2=1|X_1=0)$. We have:

$$Pr(X_1=1) = E(\beta_1(a_1,b_1)) = \frac{a_1}{a_1+b_1}$$

$$Pr(X_2=1|X_1=1) = E(\beta_2(a_2,b_2)) = \frac{a_2}{a_2+b_2}$$

$$Pr(X_2=1|X_1=0) = E(\beta_3(a_3,b_3)) = \frac{a_2}{a_2+b_2}$$

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, β_3, respectively. Suppose we perform *5* trials of a random process, the outcome of i^{th} trial denoted $E^{(i)}$ is considered as an evidence in which X_1 and X_2 obtains value *0* or *1*. So we have the vector of *5* evidences $E = (E^{(1)}, E^{(2)}, E^{(2)}, E^{(3)}, E^{(4)}, E^{(5)})$

Table III.1.25. The evidences corresponding to *5* trials

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

Let L_{ij}, F_{ij}, Δ_{ij}, Δ_i be the values of L_j, F_j, Δ_j, Δ with respect to β_i ($i = \overline{1,3}$, $j = \overline{1,2}$). We have:

$L_{11}= \frac{1}{n}\sum_{i=1}^{n}\ln x_i$ and $L_{12}= \frac{1}{n}\sum_{i=1}^{n}\ln(1-x_i)$ where x_i is the instances of X_1

$L_{21}= \frac{1}{n}\sum_{i=1}^{n}\ln x_i$ and $L_{22}= \frac{1}{n}\sum_{i=1}^{n}\ln(1-x_i)$ where x_i is the instances of X_2 given $X_1=1$

$L_{31}= \frac{1}{n}\sum_{i=1}^{n}\ln x_i$ and $L_{32}= \frac{1}{n}\sum_{i=1}^{n}\ln(1-x_i)$ where x_i is the instances of X_2 given $X_1=0$

$F_{11}(a_1, b_1) = e^{a_1}\sum_{k=1}^{b_1}\frac{(-1)^k}{k}C_{b_1}^k$ and $F_{12}(a_1, b_1)= e^{b_1}\sum_{k=1}^{a_1}\frac{(-1)^k}{k}C_{a_1}^k$

$F_{21}(a_2, b_2) = e^{a_2}\sum_{k=1}^{b_2}\frac{(-1)^k}{k}C_{b_2}^k$ and $F_{22}(a_2, b_2)= e^{b_2}\sum_{k=1}^{a_2}\frac{(-1)^k}{k}C_{a_2}^k$

$F_{31}(a_3, b_3) = e^{a_3}\sum_{k=1}^{b_3}\frac{(-1)^k}{k}C_{b_3}^k$ and $F_{32}(a_3, b_3)= e^{b_3}\sum_{k=1}^{a_3}\frac{(-1)^k}{k}C_{a_3}^k$

$\Delta_{11}=F_{11} - L_{11}$, $\Delta_{12}=F_{12} - L_{12}$, $\Delta_1= \sqrt{\Delta_{11}^{\ 2} + \Delta_{12}^{\ 2}}$

$\Delta_{21}=F_{21} - L_{21}$, $\Delta_{22}=F_{22} - L_{22}$, $\Delta_2= \sqrt{\Delta_{21}^{\ 2} + \Delta_{22}^{\ 2}}$

$\Delta_{31}=F_{31} - L_{31}$, $\Delta_{32}=F_{32} - L_{32}$, $\Delta_3= \sqrt{\Delta_{31}^{\ 2} + \Delta_{32}^{\ 2}}$

Due to $\ln(0) \to -\infty$, *ln*(*0*) is represented as a extremely large number, namely -1000; so $\ln(0) \approx -1000$. Expending the range of function *ln*(*X*) when *X* is binary variable, if *X =0* then *ln*(*X*) ≈–1000, otherwise *ln*(*X*) ≈+1000. From these evidences, it is easy to compute L_{ij}

Table III.1.26. The values of L_{ij}

β_1	β_2	β_3
L_{11} = 600	L_{21} = 500	L_{31} = –1000
L_{12} = –600	L_{22} = –500	L_{32} = 1000

Suppose the range of all parameters is from *1* to *4*. Applying the algorithm proposed, the normal biases of all possible values of (a_1, b_1) with respect to β_1 are shown in following table.

Table III.1.27. The normal biases of (a_1, b_1) with respect to β_1

a_1	b_1	F_{11}	F_{12}	Δ_{11}	Δ_{12}	Δ_1
1	1	-2.72	-2.72	-602.72	597.28	848.54
1	2	-7.39	-4.08	-607.39	595.92	850.91
1	3	-20.09	-4.98	-620.09	595.02	859.39
1	4	-54.6	-5.66	-654.6	594.34	884.16
2	1	-4.08	-7.39	-604.08	592.61	846.23
2	2	-11.08	-11.08	-611.08	588.92	848.67
2	3	-30.13	-13.55	-630.13	586.45	860.81
2	4	-81.9	-15.39	-681.9	584.61	898.19
3	1	-4.98	-20.09	-604.98	579.91	838.04
3	2	-13.55	-30.13	-613.55	569.87	837.37
3	3	-36.82	-36.82	-636.82	563.18	850.12
3	4	-100.1	-41.84	-700.1	558.16	895.36
4	1	-5.66	-54.6	-605.66	545.4	815.04
4	**2**	**-15.39**	**-81.9**	**-615.39**	**518.1**	**804.45**
4	3	-41.84	-100.1	-641.84	499.9	813.55
4	4	-113.75	-113.75	-713.75	486.25	863.64

The normal biases of all possible values of (a_2, b_2) with respect to β_2 are shown in the following table.

Table III.1.28. The normal biases of (a_2, b_2) with respect to β_2

a_2	b_2	F_{21}	F_{22}	Δ_{21}	Δ_{22}	Δ_2
1	1	-2.72	-2.72	-502.72	497.28	707.12
1	2	-7.39	-4.08	-507.39	495.92	709.49
1	3	-20.09	-4.98	-520.09	495.02	718
1	4	-54.6	-5.66	-554.6	494.34	742.93
2	1	-4.08	-7.39	-504.08	492.61	704.81
2	2	-11.08	-11.08	-511.08	488.92	707.28
2	3	-30.13	-13.55	-530.13	486.45	719.49
2	4	-81.9	-15.39	-581.9	484.61	757.26
3	1	-4.98	-20.09	-504.98	479.91	696.65
3	2	-13.55	-30.13	-513.55	469.87	696.07
3	3	-36.82	-36.82	-536.82	463.18	709.02
3	4	-100.1	-41.84	-600.1	458.16	755
4	1	-5.66	-54.6	-505.66	445.4	673.85
4	**2**	**-15.39**	**-81.9**	**-515.39**	**418.1**	**663.66**
4	3	-41.84	-100.1	-541.84	399.9	673.44
4	4	-113.75	-113.75	-613.75	386.25	725.17

The normal biases of all possible values of (a_3, b_3) with respect to β_3 are shown in the following table.

Table III.1.29. The normal biases of (a_3, b_3) with respect to β_3

a_3	b_3	F_{31}	F_{32}	Δ_{31}	Δ_{32}	Δ_3
1	1	-2.72	-2.72	997.28	-1002.72	1414.22
1	2	-7.39	-4.08	992.61	-1004.08	1411.9
1	3	-20.09	-4.98	979.91	-1004.98	1403.65
1	4	-54.6	-5.66	945.4	-1005.66	1380.27
2	1	-4.08	-7.39	995.92	-1007.39	1416.58
2	2	-11.08	-11.08	988.92	-1011.08	1414.3
2	3	-30.13	-13.55	969.87	-1013.55	1402.83
2	**4**	**-81.9**	**-15.39**	**918.1**	**-1015.39**	**1368.92**
3	1	-4.98	-20.09	995.02	-1020.09	1425
3	2	-13.55	-30.13	986.45	-1030.13	1426.27
3	3	-36.82	-36.82	963.18	-1036.82	1415.17
3	4	-100.1	-41.84	899.9	-1041.84	1376.69
4	1	-5.66	-54.6	994.34	-1054.6	1449.44
4	2	-15.39	-81.9	984.61	-1081.9	1462.86
4	3	-41.84	-100.1	958.16	-1100.1	1458.86
4	4	-113.75	-113.75	886.25	-1113.75	1423.33

From above tables, we recognize that when **(a_1,b_1)=(*4,2*), (a_2,b_2)=(*4,2*)** and **(a_3,b_3)=(*2,4*)**, the normal biases of distributions β_1, β_2 and β_3, respectively become minimum. So the parameter estimators $(\widehat{a}_1, \widehat{b}_1)$, $(\widehat{a}_2, \widehat{b}_2)$ and $(\widehat{a}_3, \widehat{b}_3)$ corresponding to distributions β_1, β_1 and β_3 are (*4,2*), (*4,2*) and (*2,4*), respectively. So the conditional probabilities $Pr(X_1=1)$, $Pr(X_2=1|X_1=1)$ and $Pr(X_2=1|X_1=0)$ are determined:

$$\Pr(X_1 = 1) = \frac{\widehat{a}_1}{\widehat{a}_1 + \widehat{b}_1} = \frac{4}{4+2} = 0.66$$

$$\Pr(X_2 = 1 \mid X_1 = 1) = \frac{\widehat{a}_2}{\widehat{a}_2 + \widehat{b}_2} = \frac{4}{4+2} = 0.66$$

$$\Pr(X_1 = 1 \mid X_1 = 0) = \frac{\widehat{a}_3}{\widehat{a}_3 + \widehat{b}_3} = \frac{2}{2+4} = 0.33$$

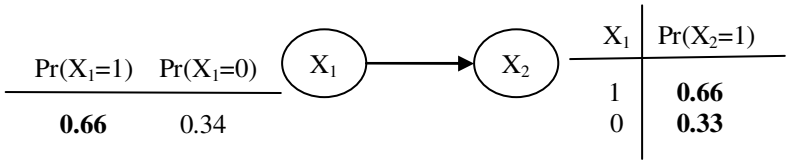


Figure III.1.25. Bayesian network with CPT (s)

III.1.5.4. Evaluation

The basic idea of MLE is to solve the equation formed by setting the first-order derivation of log-likelihood function equal *0*. I apply MLE into beta distribution so as to determine the equations for computing two parameters of beta distribution. Although beta distribution is more complex to estimate its parameters than other distributions like binominal distribution and normal distribution, I proof that these equations exist. Because it is too difficult to solve such equations, I propose the algorithm so as to find out their approximate solutions. This is iterative algorithm in which a number of parameters are surveyed and the parameter whose bias with respect to the actual solution is minimum is the approximate solution. It is impossible to find out the precise solution but it is easy and feasible to implement our algorithm as computer program.

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 ends up the comprehensive description of knowledge sub-model – the apex of Triangular Learner Model (TLM), which represents domain-specific user information. The next section describes learning style – another apex of TLM, which represents domain-independent information.

III.2. Learning style sub-model

Adaptive learning systems are developed rapidly in recent years and the "heart" of such systems is user model. User model is the representation of information about an individual that is essential for an adaptive system to provide the adaptation effect, i.e., to behave differently for different users. There are some main features in user model such as knowledge, goals, learning styles, interests, background… but knowledge, learning styles and goals are features attracting researchers' attention in adaptive e-learning domain. Contrary to knowledge model described in previous section focusing on domain-specific information about users, learning styles represent domain-independent information. Learning styles were surveyed in psychological theories but it is slightly difficult to model them in the domain of computer science because learning styles are too unobvious to represent them and there is no solid inference mechanism for discovering users' learning styles now. Moreover, researchers in domain of computer science will get confused by so many psychological theories about learning style when choosing which theory is appropriate to adaptive system.

In this section 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.

In conclusion, learning style sub-model is one among three sub-models that constituting Triangular Learner Model (TLM). It and knowledge sub-model are managed by belief network engine (BNE).

III.2.1. What learning styles are

People have different views upon the same situation, the way they perceive and estimate the world is different. So their responses to around environment are also different. For example, look at the way students prefers to study a lesson. Some have a preference for listening to instructional content (so-called *auditory* learner), some for perceiving materials as picture (*visual* learner), some for interacting physically with learning material (*tactile kinesthetic* learner), some for making connections to personal and to past learning experiences (*internal kinesthetic* learner). Such characteristics about user cognition are called learning styles but learning styles are wider than what we think about them.

Learning styles are defined as the composite of characteristic cognitive, affective and psychological factors that serve as relatively stable indicators of how a learner perceives, interacts with and responds to the learning environment. 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 following criteria:

- Their theoretical importance.
- Their wide spread use.
- Their influence on other learning style models.

Learning style models are organized within the families such as

- Constitutionally based learning styles and preferences (Dunn and Dunn).
- The cognitive structure (Witkin, Riding).
- Stable personality type (Myers-Briggs).
- Flexibly learning preferences (Kolb, Honey-Mumford, Felder-Silverman, Pask and Vermunt model).

In section III.2.2, we discuses about such learning style families. In general, learning styles are analyzed comprehensively in theory of psychology but there are few of researches on structuring learning styles by mathematical tools to predict/infer users' styles. Former researches often give users questionnaires and then analyze their answers in order to discover their styles but there are so many drawbacks of question-and-answer techniques, i.e., not questions enough, confusing questions, users' wrong answers… that such technique is not a possible solution. It is essential to

use another technique that provides more powerful inference mechanism. So, I propose the new approach which uses hidden Markov model to discover and represent users' learning styles in section III.2.4, III.2.5. Section III.2.6 is the evaluation. We should pay attention to some issues of providing adaptation of learning materials to learning styles concerned in section III.2.3.

III.2.2. Learning style families

III.2.2.1. Constitutionally based learning styles and preferences

Learning styles in this family are fixed and difficult to change. This family has the famous model "Dunn and Dunn model" developed by authors Rita Dunn and Kenneth Dunn [Dunn, Dunn 2003]. With Dunn and Dunn model, learning style is divided into *5* major strands:

- 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, tactile, etc.
- Psychological: focusing on user's psychological traits namely incorporates the information-processing elements of global versus analytic and impulsive versus reflective behaviors.

The psychological strand classifies learning styles into modalities such as

- *Auditory*: Preference to listen to instructional content.
- *Visual (Picture)*: Preference to perceive materials as pictures.
- *Visual (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 physiological strand classifies learning styles into modalities such as

- *Impulsive*: Preference to try out new material immediately.
- *Reflective*: Preference to take time to think about a problem.
- *Global*: Preference to get the 'big picture' first, details second.
- *Analytical*: Preference to process information sequentially: details first, working towards the 'big picture'.

III.2.2.2. The cognitive structure

In this family, learning styles are considered as structural properties of cognitive system itself. So styles are linked to particular personality features, which implicates that cognitive styles are deeply embedded in personality structure. There are two models in this family: Witkin model and Riding model.

Witkin model

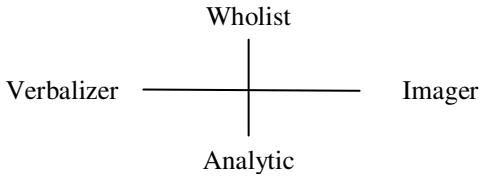
The main aspect in Witkin model [Witkin, Moore, Goodenough, Cox 1997] is the bipolar dimensions of *field-dependence / field-independence* (FD/FI) in which:

- *Field-dependence* (FD) person process information globally and attend to the most salient cues regardless of their relevance. 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. 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*.

- *Wholist-Analytic* dimension expresses how an individual cognitively organize information either into whole or parts. *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, either as text or picture. *Verbalizer* prefers to text. *Imager* prefers to picture.



III.2.2.3. Stable personal type

The models in this family have a common focus upon learning style as one part of the observable expression of a relatively stable personality type. We will glance over the famous model in this family: Myers-Briggs Type Indicator.

Myers-Briggs Type Indicator

This model involves four different pairs of opposite preferences for how person focus and interact with around environment:

- 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.
- 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.
- 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.
- How does a person manage her/his life?
a. *Judger*: organized, set and follow agendas, make decisions quickly.
b. *Perceiver*: disorganized, adapt to change environment, gather more information before making a decision.

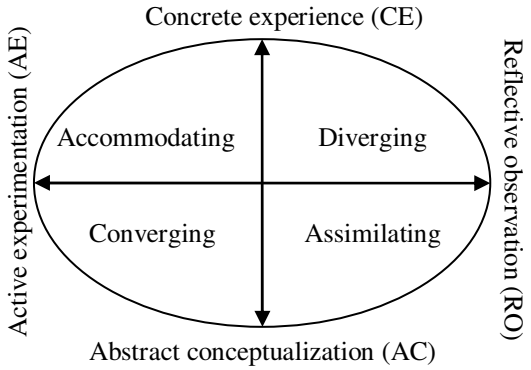
III.2.2.4. Flexible stable learning preference

With models in this family, learning style is not a fixed trait but is a differential preference for learning, which changes slightly from situation to situation. There are three typical models in this family: Kolb's Learning Style Inventory, Honey and Mumford, Felder-Silverman

Kolb Learning Style Inventory

According to Kolb [Kolb 1999], the author of this model: "learning is the process whereby knowledge is created through the transformation of experience. Knowledge results from the combination of grasping experience and transforming it". The center of Kolb 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). 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) himself.
3. He conceptualizes what he watches (observations) into abstract concepts (*AC* - thinking).
4. He experiments actively such concepts and gets the new experience (*AE* - doing). The cycle repeats again.



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.

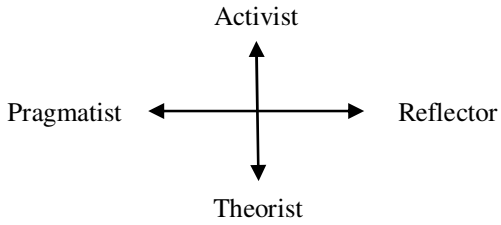
- *Accommodating* (CE/AE): emphasizes concrete experience and active experimentation. Learners prefer to apply learning material in new situations so that they solve real problems. A typical question for this style is "What if?"
- *Assimilating* (AC/RO): prefers abstract conceptualization and reflective observation. 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. 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. 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 1992], the authors of this model, there are four learning styles:

- *Activist*: learners are open-mined 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.



Felder-Silverman model

This model developed by Felder and Silverman [Felder, Silverman 1988] involves following dimensions:
- *Active/Reflective*. Active students understand information only if they discussed it, applied it. Reflective students think thoroughly about things before doing any practice.

- *Sensing/Intuitive*. Sensing students learn from concrete tasks related to problems and facts that could be solved by well-behaved methods. They are keen on details. Intuitive students discover alternate possibilities and relationships by themselves, working with abstractions and formulas.
- *Verbal/Visual*. Verbal students like learning materials in text form. Otherwise visual student prefer to images, pictures, videos, etc.
- *Sequential/ Global*. Sequential students structure their learning process by logically chained steps, each step following from previous one. Global students prefer to learn in random jumps. They can solve complicated problem but don't know clearly how they did it.

Pask model

Pask model developed by Pask [Pask 1976] states that there are two learning styles:
- Who*list*. 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:
- *Meaning-oriented*. Learners prefer to get theory before go to examples (similar to assimilating style of Kolb model).
- *Application-directed.* Learners prefer to know the purpose of information before get theory (similar to accommodating style of Kolb model).
- *Undirected*: similar to FD style of Wikin model.
- *Reproduction-oriented*: similar to FI style of Wikin model.

III.2.3. Providing adaptation of learning materials to learning styles

III.2. Learning style sub-model

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 (lesson, exercise, test, etc). Such teaching method is called learning strategy or instructional strategy or adaptive strategy. Although there are many learning style models but they share some common features, such as the modality *visual (text)/visual (picture)* in Dunn and Dunn model is similar to *verbalizer/imager* dimension in Riding model and *verbal-visual* dimension in Felder-Silverman model. Strategies are supposed according to common features of model because it is too difficult to describe comprehensively all features of model. Features of all models (learning styles) can be categorized into two groups: perception and understanding which are enumerated together with adaptive strategies as below:

Perception group: This group related learners' perception includes:

- The *visual*(*picture*) / *visual*(*text*) modality in Dunn and Dunn model is similar to the *verbalizer/imager* dimension in Riding model and *verbal-visual* dimension in Felder-Silverman model. Instructional strategy is that the teacher should recommend textual materials to verbalizer and pictorial materials to imager
- The *sensing/intuitive* dimension in Felder-Silverman model is identical to the *sensor/intuitive* dimension in Myer Briggs Type Indicator. Sensing learners are recommended examples before expositions, otherwise, expositions before examples for intuitive learners
- The *perceptive-judging* dimension in Myer Briggs Type Indicator. Perceptive learners are provided rich media such as the integrative use of pictures, tables and diagram. Otherwise, judging learners are provided lean materials.
- The *impulsive/reflective* modality in Dunn and Dunn model is similar to the *activist/reflector* dimension in Honey and Mumford model, the *active/reflective* dimension in Felder-Silverman model and the *extravert/introvert* of Myers-Briggs Type Indicator. Active (also impulsive, extravert) learners are provided activity-oriented approach: showing content of activity and links to example, theory and exercise. Reflective (also introvert) learners are provided example-oriented approach: showing content of example and links to theory, exercise and activity.

- The *theorist/pragmatist* dimension of Honey and Mumford model. Theorists are provided theory-oriented approach: showing content of theory and links to example, exercise and activity. Pragmatists are provided exercise-oriented approach: showing content of exercise and links to example, theory and activity.
- The *accommodating/assimilating* dimension of Kolb model is similar to *application-directed/ meaning-oriented* dimension of Vermunt model. The adaptive strategy for accommodating style is to provide application-based information to learners; otherwise, theory-based information for assimilating style.

Understanding group: This group related to the way learners comprehend knowledge includes:

- The *global/analytical* modality in Dunn and Dunn model is similar to *wholist-analytic* dimension in Riding model, *global/sequential* dimension in Felder-Silverman model, *wholist-serialist* dimension in Pask model. Global (also wholist) learners are provided breadth-first structure of learning material. Otherwise, analytical (also analytic, sequential, serialist) learners are recommended depth-first structure of learning materials. For the breadth-first structure, after a learner has already known all the topics at the same level, other descendant topics at lower level are recommended to her/him. For the depth-first structure, after a learner has already known a given topic T_1 and all its children (topic) at lower level, the sibling topic of T_1 (namely T_2, at same level with T_1) will be recommended to her/him.
- The *FD/FI* dimension in Wikin model is correlated with *undirected/reproduction-oriented* dimension in Vermunt model. FD learners are provided breadth-first structure of materials, guided navigation, illustration of ideas with visual materials, advance organizer and system control. FI learners are provided depth-first structure of materials or navigational freedom, user control and individual environment.

The adaptive strategy (for learning style) is the sequence of adaptive rules which define how adaptation to learning styles is performed. Learning style strategies is classified into three following forms:

- Selection of information: Information (learning materials) is presented in various types such as text, audio, video, graph, picture, etc. 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, stretch text, etc.

- Ordering information or providing different navigation paths: The order in which learning materials is suggested to users is tuned with learning styles. For active learners, learning materials are presented in the order: activity→example→theory→exercise. For reflective learner, this order is changed such as example→theory→exercise→activity. This form is corresponding to link adaptation techniques: direct guidance, link sorting, link hiding, link annotation.
- Providing learners with navigation support tools: Different learning tools are supported to learners according to their learning styles. For example, in Witkin model, FD learners are provided tools such as concept map, graphic path indicator. Otherwise FI learners are provided with a control option showing a menu from which they can choose in any order because they have high self-control.

There are two type of strategy [Stash, Cristea, De Bra 2005]:

- *Instructional strategy* is itself, which contains adaptive rules and is in three above forms.
- *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.

Our approach is an instructional meta-strategy, which applies Markov model to infer users' learning styles. Before discussing about main techniques, it is necessary to glance over hidden Markov model.

III.2.4. Hidden Markov model

There are many real-world phenomena (so-called states) that we would like to model in order to explain our observations. Often, given sequence of observations symbols, there is demand of discovering real states. For example, there are some states of weather: *sunny*, *cloudy*, *rainy*. Based on observations such as wind speed, atmospheric pressure, humidity, temperature…, it is possible to forecast the weather by using hidden Markov model (HMM). Before discussing about HMM, we should glance over the definition of Markov model (MM). First, MM is the statistical model which is used to model the stochastic process. MM is defined as below:

- Given a finite set of state $S=\{s_1, s_2,\ldots, s_n\}$ whose cardinality is n. Let $\prod$ be the *initial state distribution* where $\pi_i \in \prod$ 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 times. The process is denoted as a finite vector $P=(x_1, x_2,\ldots, x_u)$ whose element x_i is a state ranging in space S. Note that $x_i \in S$ is one of states in the finite set S, x_i is identical to s_i. Moreover, the process must meet fully the *Markov property*, namely, given the current state x_k of process P, the conditional probability of next state x_{k+1} is only relevant to current state x_k, not relevant any past state $(x_{k-1}, x_{k-2}, x_{k-3},\ldots)$. In other words, $Pr(x_k \mid x_0, x_1,\ldots, x_{k-1}) = Pr(x_k \mid x_{k-1})$. Such process is called first-order Markov process.

- At each lock time, the process transitions to the next state based upon the *transition probability distribution* a_{ij} which depends only on the previous state. So a_{ij} is the probability that, the process change the current state s_i to next state s_j. The probability of transitioning from any given state to some next state is *1,* we have $\forall s_i \in S, \sum_{s_j \in S} a_{ij} = 1$. All transition probabilities a_{ij} (s) constitute the *transition probability matrix* **A**.

Briefly, MM is the triple $\langle S, A, \prod \rangle$. In typical MM, states are observed directly by users and transition probability matrix is the unique parameters. Otherwise, hidden Markov model (HMM) is similar to MM except that the underlying states become hidden from observer, they are hidden parameters. HMM adds more output parameters which are called observations. Each state (hidden parameter) has the conditional probability distribution upon such observations. HMM is responsible for discovering hidden parameters (states) from output parameters (observations), given the stochastic process. The HMM has further properties as below:

- There is the second stochastic process which produces *observations* correlating hidden states. Suppose there is a finite set of possible observations $\Theta=\{\theta_1, \theta_2,\ldots, \theta_m\}$ whose cardinality is m.

- There is a probability distribution of producing a given observation in each state. Let $b_i(k)$ be the probability of observation θ_k when the second stochastic process is in state s_i. The sum of probabilities of all observations which observed in a certain state is *1*, we have $\forall s_i \in S, \sum_{\theta_k \in \Theta} b_i(k) = 1$. All probabilities of observations $b_i(k)$ constitute the *observation probability matrix* **B**.

Thus, HMM is the *5*-tuple $\Delta = \langle S, \Theta, A, B, \prod \rangle$. Back to weather example, suppose you need to predict how whether is tomorrow: *sun* or *cloud* or *rain* since you know only observations about the humidity: *dry, dryish, damp, soggy*. The HMM is represented following:

S = {*sun, cloud, rain*}, Θ = {*dry, dryish, damp, soggy*}

		weather today		
		sun	*cloud*	*rain*
weather yesterday	*sun*	0.5	0.25	0.25
	cloud	0.4	0.2	0.4
	rain	0.1	0.7	0.2

Transition probability matrix *A*

sun	*cloud*	*rain*
0.5	0.5	0.5

Uniform initial state distribution $\prod$

		humidity			
		dry	*dryish*	*damp*	*soggy*
weather	*sun*	0.6	0.2	0.15	0.05
	cloud	0.25	0.25	0.25	0.25
	rain	0.05	0.1	0.35	0.5

Observation probability matrix B

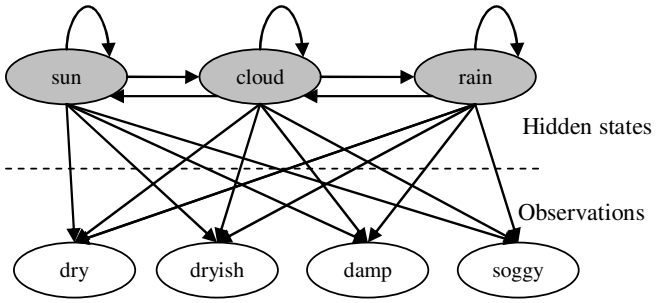


Figure III.2.1. HMM of weather forecast (hidden states are shaded)

Uncovering problem and Viterbi algorithm

Given HMM Δ and a sequence of observations $O = \{o_1 \rightarrow o_2 \rightarrow \ldots \rightarrow o_k\}$ where $o_i \in \Theta$, how to find the sequence of states $U = \{u_1 \rightarrow u_2 \rightarrow \ldots \rightarrow u_k\}$ where $u_i \in S$ so that U is most likely to have produced the observation sequence O. This is the uncovering problem: which sequence of state transitions is most likely to have led to this sequence of observations. It means to maximize the selection of U: $\arg\max_U [\Pr(O \mid \Delta)]$. We can apply brute-force strategy: "go through all possible such O and pick the one with the maximum" but this strategy is infeasible given a very large numbers of states. In this situation, Viterbi algorithm [Dugad, Desai 1996] is the effective solution. Instead of describing details of Viterbi algorithm, I only use it to predict learner's styles given observations about her/him.

III.2.5. Applying Hidden Markov model into building up learning style sub-model

For modeling learning style (LS) using HMM we should determine states, observations and the relationship between states and observations in context of learning style. In other words, we must define five components $S, \Theta, A, B, \prod$. 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. Suppose we choose Honey-Mumford model and Felder-Silverman model as principal models which are presented by HMM. We have three dimensions: *Verbal/Visual, Activist/ Reflector*, *Theorist/ Pragmatist* which are modeled as three HMM(s): Δ_1, Δ_2, Δ_3 respectively. For example, in Δ_1, there are two states: *Verbal* and *Visual*; so S_1={*verbal*, *visual*}. We have:

- $\Delta_1 = \langle S_1, \Theta_1, A_1, B_1, \prod_1 \rangle$.
- $\Delta_2 = \langle S_2, \Theta_2, A_2, B_2, \prod_2 \rangle$.
- $\Delta_3 = \langle S_3, \Theta_3, A_3, B_3, \prod_3 \rangle$.

We are responsible for defining states (S_i), initial state distributions ($\prod_i$), transition probability matrices (A_i), observations (Θ_i), observation probability matrices (B_i) through five steps

1. Defining **states**: each state is corresponding to a leaning style.

S_1={*verbal*, *visual*},
S_2={*activist*, *reflector*},
S_3={*theorist* , *pragmatist*}.

2. Defining **initial state distributions**: Uniform probability distribution is used for each $\prod_i$.

$\prod_1$ = {*0.5, 0.5*}; it means that *Pr*(*verbal*) = *Pr*(v*isual*) = *0.5*
$\prod_2$ = {*0.5, 0.5*}; *Pr*(*activist*) = *Pr*(*reflector*) = *0.5*
$\prod_3$ = {*0.5, 0.5*}; *Pr*(*theorist*) = *Pr*(*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, $Pr(s_i=verbal \mid s_{i-1}=verbal) = 0.7$ is obviously higher than $Pr(s_i=verbal \mid s_{i-1}=verbal) = 0.3$.

Table III.2.1. Transition probability matrices: A_1, A_2, A_3

	verbal	***visual***
verbal	0.7	0.3
visual	0.3	0.7

	activist	**reflector**
activist	0.7	0.3
reflector	0.3	0.7

	theorist	**pragmatist**
theorist	0.7	0.3
pragmatist	0.3	0.7

4. Defining **observations**. There is a relationship between learning object learned by users and their learning styles. Three attributes are assigned to each learning object such as lecture, example, etc.

- *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), it raises observations depending on the attributes of learning object. We must account for the values of the attributes selected. For example, if a student selects a LO which has

format attribute being *text*, *type* attribute being *theory*, *interactive* attribute being *low*, there are considerable observations: *text*, *theory*, *low* (interaction). So, it is possible to infer that she/he is a theorist.

The dimension *Verbal/Visual* is involved in format attribute. The dimensions *Activist/ Reflector* and *Theorist/ Pragmatist* relate to both *type* attribute and *interactive* attribute. So we have:

- Θ_1 = { *text, picture, video* }
- Θ_2 = { *theory, example, exercise, puzzle, low* (interaction)*, medium* (interaction)*, high* (interaction) }
- Θ_3 = { *theory, example, exercise, puzzle, low* (interaction)*, medium* (interaction)*, high* (interaction) }

5. Defining **observation probability matrices**. Different observations (attributes of LO) effect on states (learning styles) in different degrees. Because the "weights" of observation vary according to states, there is a question: "How to specify weights?". If we can specify these "weights", it is easy to determine observation probability matrices.

In the Honey-Mumford model and Felder-Silverman model, verbal students prefer to text material and visual students prefer to pictorial materials. The weights of observations: *text, picture, video* on state *Verbal* are in descending order. Otherwise, the weights of observations: *text, picture, video* on state *Visual* are in ascending order. Such weights themselves are observation probabilities. We can define these weights as below:

- *Pr*(*text* | *verbal*) = *0.6*, *Pr*(*picture* | *verbal*) = *0.3*, *Pr*(*video* | *verbal*) = *0.1*
- *Pr*(*text* | *visual*) = *0.2*, *Pr*(*picture* | *visual*) = *0.4*, *Pr*(*video* | *visual*) = *0.4*

There are some differences in specifying observation probabilities of dimensions *Activist/Reflector* and *Theorist/ Pragmatist*. As discussed, active learners are provided activity-oriented approach: showing content of activity (such as puzzle, game…) and links to example, theory and exercise. Reflective learners are provided example-oriented approach: showing content of example and links to theory, exercise and activity (such as puzzle, game…). The weights of observations: *puzzle, example, theory, exercise* on state *Activist* are in descending order. The weights of observations: *example, theory, exercise, puzzle* on state *Reflector* are in descending order. However, activists tend to learn high interaction materials and reflectors prefer to low interaction materials. So the weight of observations: *low* (interaction), *medium* (interaction), *high* (interaction) on state *Activist* get values: *0, 0, 1,* respectively. Otherwise, the weight of observations: *low* (interaction), *medium* (interaction), *high* (interaction) on state *Reflector* get values: *1, 0, 0,* respectively. We have:

- *Pr*(*puzzle* | *activist*) = *0.4*, *Pr*(*example* | *activist*) = *0.3*, *Pr*(*theory* | *activist*) = *0.2*, *Pr*(*exercise* | *activist*) = *0.1*
  Pr(*low* | *activist*) = *0*, *Pr*(*medium* | *activist*) = *0*, *Pr*(*high* | *activist*) = *1*.
- *Pr*(*example* | *reflector*) = *0.4*, *Pr*(*theory* | *reflector*) = *0.3*, *Pr*(*exercise* | *reflector*) = *0.2*, *Pr*(*puzzle* | *reflector*) = *0.1*
  Pr(*low* | *reflector*)=*1*, *Pr*(*medium* | *reflector*) = *0*, *Pr*(*high* | *reflector*)= *0*.

Because the sum of conditional probabilities of observations on each state equals *1*, we should normalize above probabilities.

- *Pr*(*puzzle* | *activist*) = *0.4*4/7 = 0.22*, *Pr*(*example* | *activist*) = *0.3*4/7 = 0.17*, *Pr*(*theory* | *activist*) = *0.2*4/7 = 0.11*, *Pr*(*exercise* | *activist*) = *0.1*4/7 = 0.05*
  Pr(*low* | *activist*) = *0*3/7 = 0*, *Pr*(*medium* | *activist*) = *0*3/7 = 0*, *Pr*(*high* | *activist*) = *1*3/7 = 0.42*
- *Pr*(*example* | *reflector*) = *0.4*4/7 = 0.22*, *Pr*(*theory* | *reflector*) = *0.3*4/7 = 0.17*, *Pr*(*exercise* | *reflector*) = *0.2*4/7 = 0.11*, *Pr*(*puzzle* | *reflector*) = *0.1*4/7 = 0.05*
  Pr(*low* | *reflector*) = *1*3/7 = 0.42*, *Pr*(*medium* | *reflector*) = *0*3/7 = 0*, *Pr*(*high* | *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.

Table III.2.2. Observation probability matrices: B_1, B_2, B_3

	text	***picture***	***video***
verbal	0.6	0.3	0.1
visual	0.2	0.4	0.4

	theory	***example***	***exercise***	***puzzle***	***low***	***medium***	***high***
activist	0.11	0.17	0.05	0.22	0	0	0.42
reflector	0.17	0.22	0.11	0.05	0.42	0	0

	theory	***example***	***exercise***	***puzzle***	***low***	***medium***	***high***
pragmatist	0.11	0.17	0.22	0.05	0.04	0.08	0.3
theorist	0.22	0.17	0.11	0.05	0.3	0.08	0.04

Now three HMM (s): Δ_1, Δ_2, Δ_3 corresponding to three dimensions of learning styles: *Verbal/Visual*, *Activist/Reflector*, *Pragmatist/Theorist* are represented respectively in figure III.2.2.

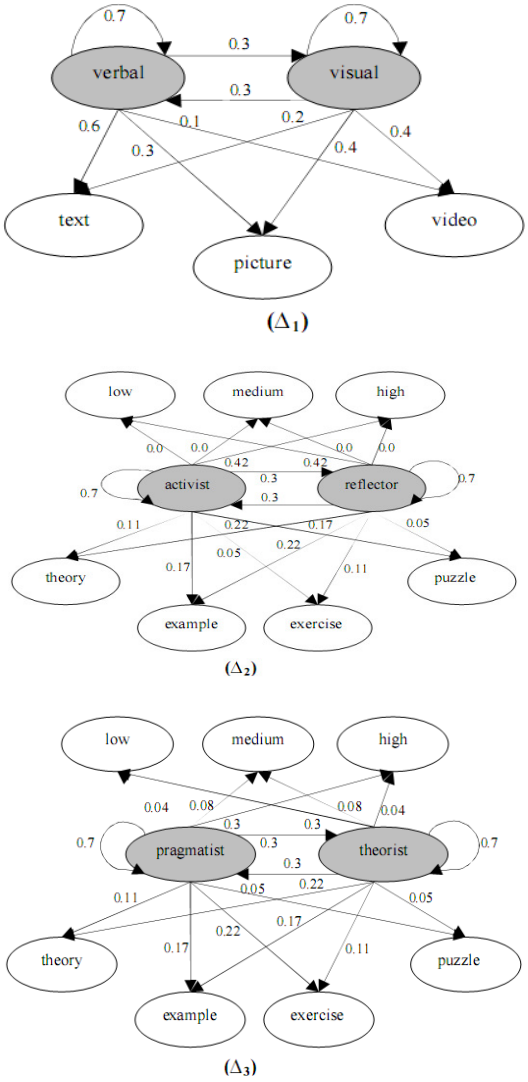


Figure III.2.2. HMM (s) of learning styles (hidden states are shaded)

An example for inferring student's learning styles

Suppose the learning objects that a student selects in session 1, 2 and 3 are LO_1, LO_2 and LO_3 respectively.

Table III.2.3. Learning objects selected

	Format	Type	Interactive
LO_1	picture	theory	not assigned
LO_2	text	example	not assigned
LO_3	text	not assigned	low

It is easy to recognize the sequence of user observations from the attributes *format, type, interactive*

Table III.2.4. Sequence of student observations

HMM – Dimension	Sequence of observations
Δ_1: Dimension Verbal/Visual	picture → text → text
Δ_2: Dimension Activist/Reflector	theory → example → low
Δ_3: Dimension Pragmatist/Theorist	theory → example → low

Using Viterbi algorithm for each HMM, it is possible to find corresponding sequence of state transitions that is most suitable to have produced such sequence of observations.

Table III.2.5. Sequence of state transitions

HMM - Dimension	Sequence of observations	Sequence of state transitions	Student style
Δ_1	picture → text → text	visual → verbal	verbal
Δ_2	theory → example → low	reflector → reflector → reflector	reflector
Δ_3	theory → example → low	theorist → theorist → theorist	theorist

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

III.2.6. Evaluation

HMM and Viterbi algorithm 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 our approach into other learning style models such as Witkin, Riding, Kolb… and there is no need to alter main techniques except that we should specify new states correlating with new learning styles and add more attributes to learning objects.

This section 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 and learning style sub-model. Such information is fine information analyzed or extracted from more essential information that is manipulated by the third sub-model so-called learning history sub-model which is described in next section.

III.3. Learning history sub-model

Learning history sub-model is the basic sub-model among three sub-models constituting the Triangular Learner Model (TLM). 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, etc. This sub-model has four main functions:

1. *Providing necessary information for two remaining sub-models*: learning style sub-model and knowledge sub-model described in previous sections 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.
2. *Supporting learning concept recommendation*.
3. Mining learners' educational data in order to *discover other learners' characteristics* such as interests, background, goals, etc.
4. Supporting collaborative learning through constructing learner groups.

This model is managed by mining engine (**ME**) which almost always uses mining techniques. ME is very important for collecting learners' data, monitoring their actions, structuring and updating TLM. The first responsibility is discussed in previous chapters about knowledge sub-model and learning style sub-model. So I have just described in this chapter the approaches that learning history sub-model applies to perform recommendation tasks and discover another learners' characteristic, namely user interests. 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 model. So this model is very necessary for extending Triangular Learner Model (see figure II.7).

This section includes three following sub-sections such as learning concept recommendation, discovering user interests and constructing user groups that correspond to functions *2, 3, 4* of this sub-model.

III.3.1. Learning path and learning concept recommendation based on mining learning history

Sequential pattern mining is new trend in data mining domain with many useful applications, especially commercial application but it also results surprised effect in adaptive learning. Suppose there is an adaptive e-learning website, a student accesses learning material / does exercises relating domain concepts in sessions. Her/his learning sequences which are lists of concepts accessed after total study sessions construct the learning sequence database *S*. *S* is mined to find the sequences which are expected to be learned frequently or preferred by student. Such sequences called sequential patterns are use to recommend appropriate concepts / learning objects to students in his next visits. It results in enhancing the quality of adaptive learning system. This process is sequential pattern mining. In this section, I also suppose an approach to break sequential pattern $s=\langle c_1, c_2,\ldots, c_m\rangle$ into association rules including left-hand and right-hand in form $c_i \rightarrow c_j$. Left-hand is considered as source concept, right-hand is treated as recommended concept available to students.

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.
- Content-based filtering system recommends items that are considered appropriate to user information in his profile.
- 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.

Sequential pattern mining belongs to collaborative filtering family. User does not rate explicitly items but his series of chosen items are recorded as sequences to construct the sequence database which mined to find frequently repeated

patterns he can choose in future. In learning context, items can be domain concepts / learning objects which students access or learn. First, we should glance over basic concepts and what is sequential pattern mining along with its application, especially in adaptive learning.

Suppose $I = \{i_1, i_2, i_3,\ldots, i_n\}$ is a set of all items. An itemset is a subset of I, denoted $x_i=(i_u,\ldots, i_v)$, if x_i has only one item i_k, it can be denoted $x_i=i_k$, omitting the brackets. An sequence is an ordered list of itemsets, denoted $s=\langle x_1, x_2,\ldots, x_m\rangle$,

where x_i is an item set, x_i is also called an element in sequence s. An item has only been once in an element of sequence but can occur multiple times in different elements in sequences. The number of instances of items occurring in sequence is the length of sequence. Sequence with length l is called *l-sequence*.

Sequence $s_1=\langle x_1, x_2,\ldots, x_m\rangle$ is called sub-sequence of $s_2=\langle y_1, y_2,\ldots, y_m\rangle$ denoted $s_1 \subseteq s_2$ or $s_1 \supseteq s_2$ if there is the ordered series of indexes $k_1, k_2,\ldots, k_n$ so that $1 \le k_1 \le k_2 \le \ldots. \le k_n \le m$ and $x_1 \subseteq y_{k_1}$, $x_2 \subseteq y_{k_2}$,..., $x_n \subseteq y_{k_n}$. 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 $x_1=i_1$, $x_2=(i_1,i_2)$, $x_3=(i_1,i_2,i_3)$, $x_4=(i_1,i_2,i_3,i_4)$, $x_5=i_3$, $x_6=(i_1,i_2)$
- 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:

- 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
- s_3 is sub-sequence of s_2 due to $x_5 \subseteq x_2$ and $x_6 \subseteq x_3$, 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 the fraction of total records containing α, denoted $support(\alpha)=|\{\langle sid, s\rangle \mid \alpha \subseteq s\}|$. 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 $support(\alpha) \ge min\_sup$ then α is called frequent or large 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.

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.

Issue of learning concepts / objects recommendation

Suppose there are compulsory concepts (subjects) in Java course: **d**ata type, **p**ackage, class & **O**OP, **s**election structure, **v**irtual machine, **l**oop structure, **c**ontrol structure, and inter**f**ace which in turn denoted as *d, p, o, s, v, l, c, f*. At our e-learning website, students access learning material relating such domain concepts in sessions, each session contains only one item set and is ordered by time. The student's learning sequence is constituted of item sets accessed in all his sessions.

Table III.3.1. learning sessions → learning sequences

Student	Session	Concept accessed	ID	Learning sequences	Length
1	Aug 5 10:20:01	*o*	1	⟨*of*⟩	2
1	Aug 5 10:26:12	*f*			
2	Aug 6 08:20:01	*d, p*	2	⟨(*dp*)*o*(*slc*)⟩	6
2	Aug 6 14:15:01	*o*			
2	Aug 6 15:00:00	*s, l, c*			

3	Aug 7 12:30:00	*o, v, c*	3	⟨(*ovc*)⟩	3
4	Aug 8 07:14:20	*o*			
4	Aug 8 07:40:25	*s, c*	4	⟨*o*(*sc*)*f*⟩	4
4	Aug 8 10:17:20	*f*			
5	Aug 8 10:26:15	*f*	5	⟨*f*⟩	1

Students accessed learning material in their past sessions, how system recommends appropriate domain concepts to student for next visits. It issues the application of mining sequential patterns. In e-learning context, (learning) sequential pattern is a sequence of concepts / learning materials that students prefer to study or access regularly. Our solution includes two steps as following:

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 learning sequence patterns in III.3.1.1. The approach to break patterns into concepts is proposed in III.3.1.2. Section III.3.1.3 is the evaluation.

III.3.1.1. Approaches of learning sequential pattern mining

There are two main approaches of sequential pattern mining:

- Candidate generation-and-test approach based on algorithm Apriori is also classified into two categories: horizontal and vertical data format methods. This approach is essentially an extension of associate rule discovering algorithm Apriori satisfying the statement "*every non-empty sub-sequence of a frequent sequence is a frequent sequence"* (and vice versa) considered as downward closure property. This approach has two trends: horizontal and vertical data format based sequential pattern with respect to three algorithms: AprioriAll [Agrawal, Srikant 1995], GSP [Srikant, Agrawal 1996] and SPADE [Zaki, 2000].

- Pattern-growth approach based on pattern-growth mining of frequent patterns in transaction database. Huge candidate sets generated in AprioriAll, GSP, SPADE are caused by the large number of elements in seed set. Mining separately frequent sequences with disjointed database for the purpose of reduce the number of elements is idea of FreeSpan algorithm. FreeSpan [Han, Pei, Mortazavi-Asl, Chen, Dayal, Hsu 2000] 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.

III.3.1.2. A proposal of breaking learning sequential pattern

Suppose we discovered the sequential pattern ⟨*osc*(*sc*)⟩ which means:
*"class and **O**OP"–"**s**election structure"–"**c**ontrol structure"–"**s**election structure, **c**ontrol structure"*
and some other patterns as results in algorithm AprioriAll. We accept that these patterns like the learning "routes" that student preferred or learned often in past but in the next time if a student chooses the concept "***c**ontrol structure*", the adaptive learning system should recommend which next concepts in above patterns? So the patterns should be broken into association rules with their confidence. For example, breaking above pattern ⟨*osc*(*sc*)⟩ follows steps:

1. Breaking entire ⟨*osc*(*sc*)⟩ into litemsets such as *o, s, c,* (*sc*) and determining all possible large *2*-sequences which are *2*-arrangment of all litemsets following the criterion: order of *2*-sequences must comply with the order of sequential pattern. There are six large *2*-sequences: ⟨*os*⟩*,* ⟨*oc*⟩*,* ⟨*o*(*sc*)⟩*,* ⟨*sc*⟩*,* ⟨*s*(*sc*)⟩*,* ⟨*c*(*sc*)⟩. Thus, we have six rules derived from these large *2*-sequences in form: "*left-hand litemset → right-hand litemset"*, for example, rule "*s→c*" derived from *2*-sequence ⟨*sc*⟩.

2. Computing the confidences of such rules and sorting them according to these measures. The confidence of a rule is the ratio of the support of *2*-sequences and the support of left-hand litemset, *confidence(x → y) = support*(⟨*xy*⟩) / *support*((*x*)). The rules whose confidence is less than threshold *min_conf* is removed in order to ensure that remains are strong rules. We called these rules as **sequential rules** and these confidences as sequential confidences.

Table III.3.2. Sequential rules

Sequential rules	Sequential confidences
o→s	40%
o→c	40%
o→sc	40%
s→c	0%
s→sc	0%
c→sc	0%

Now, if student choose the concept (itemset) *x*, system will find whole rules broken from all sequential patterns and the

left-hand litemset of each rule must contain *x*. Then, these rules are sorted by their confidences in descending order. Final outcome is an ordered list of *right-hand litemsets* (concepts) of rules, which are recommended to students. The top concept (litemset) in such list is referred as the most necessary concept. For example, as results in table 8, if users choose concept "*class & **O**OP*", we recommend them other concepts as below:

Concepts	Rates
"selection structure"	40%
"control structure"	40%

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.

III.3.1.3. Evaluation

Although sequential pattern mining is applied in e-commercial for customer purchase but the extracted patterns can be used to predict user's learning "route", which is fundamental of learning object recommendation. There are two sequential pattern mining approaches: candidate generation-and-test, pattern-growth. The first which is essentially a refinement of the Apriori-like is easy to implement but causes a problem of large candidate set and so, leads to performance downfall and requirement of both complex computation and huge storage. Especially, in e-learning environment, there are thousands of students; speed and performance are very important factors. The second which is a divide-and-conquer solution intends to reduce the number of candidate sequences. So, it is more efficient.

Last, I propose the technique to break sequential patterns into rules containing recommendable concepts / learning materials. The ideology of this approach is derived from association rule mining but it is more efficient than association rules.

Learning concept recommendation is useful extended function of Zebra, which is supported by mining engine (ME) and learning history sub-model. Another extended function discovering user interests is described in next section, which is also supported by ME and learning history sub-model.

III.3.2. Discovering user interests by document classification

III.3. Learning history sub-model

User interest is one of personal traits attracting researchers' attention in user modeling and user profiling. 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 (lesson, example, exercise, test…) to user interests. I propose a new approach for discovering 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. There are two new points of view:

- The series of user access in his/her history are modeled as documents. So user is referred indirectly to as "document".
- User interests are classes such documents are belong to.

Our approach includes four following steps:

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 decision tree or support vector machine or neural network. Classification rules (weight vectors $\vec{W}$) are drawn from decision tree (support vector machine). They are used as classifiers.
3. Mining user's access history to *find maximum frequent itemsets*. Each itemset is considered an *interesting document* and its member items are considered as terms. Such interesting documents are modeled as vectors.
4. Applying classifiers (see step 2) into these interesting documents (see step 3) order to choose which classes are most suitable to these interesting documents. *Such classes are user interests*.

This approach bases on document classification but it also relates to information retrieval in the manner of representing documents. Hence section III.3.2.1 discusses about vector model for representing documents. Support vector machine, decision tree and neural network on document classification are mentioned in section III.3.2.2, III.3.2.3, III.3.2.4. Main technique to discover user interest is described in section III.3.2.5. Section III.3.2.6 is the evaluation

III.3.2.1. Vector model for representing documents

Suppose our corpus **D** is the composition of documents $D_i \in \mathbf{D} = \{D_1, D_2, \ldots, D_m\}$. Every document D_i contains a set of key words so-called *terms*. The number of times a term occurs in a document is called *term frequency*. 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 as below:

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{ik}}$$

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* is a measure of general importance of the term. It is used to decrease the weight of terms occurring frequently and increase the weight of terms occurring rarely. The inverse document frequency of term t_j is the ratio of the size of corpus to the number of documents that t_j occurs.

$$idf_j = \log \frac{|corpus|}{|\{D: t_j \in D\}|}$$

Where |*corpus*| is the total number of documents in corpus and |{*D:* $t_j \in D$}| is the number of documents containing term t_j. The *log* function is used to normalize idf_j so that it is less than or equals *1*.

The weight of term t_j in document D_i is defined as product of tf_{ij} and idf_i
$w_{ij} = tf_{ij} * idf_i$

This weight measure 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 *n* terms {t_i, t_2,…, t_n}, each document D_i is modeled as the vector which is composed of weights of such terms.
$D_i = (w_{i1}, w_{i2}, w_{i3},\ldots, w_{in})$
Hence the corpus becomes a matrix *m* x *n*, which have *m* rows and *n* columns with respect to *m* document and *n* 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. In this section I apply three methods namely support vector machine, decision tree and neural network for document classification. There methods are described in successive sections.

III.3.2.2. Document classification based on support vector machine

Support vector machine

Support vector machine (SVM) [Cristianini, Shawe-Taylor, 2000] is a supervised learning algorithm for classification and regression. Given a set of *n*-dimensional vectors in vector space, SVM finds the separating hyper-plane that splits vector space into sub-set of vectors; each separated sub-set (so-called data set) is assigned one class. There is the constraint

for this separating hyper-plane: “it must maximize the margin between two sub-sets”.

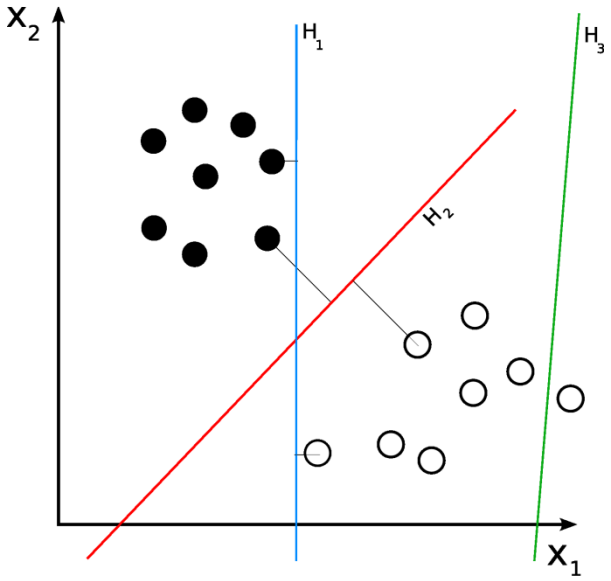


Figure III.3.1. Separating hyper-planes

Suppose we have some *n*-dimensional vectors; each of them belongs to one of two classes. We can find many *n-1* dimensional hyper-planes that classify such vectors but there is only one hyper-plane that maximizes the margin between two classes. In other words, the nearest between a point in one side of this hyper-plane and other side of this hyper-plane is maximized. Such hyper-plane is called maximum-margin hyper-plane and it is considered as maximum-margin classifier.
Let $\{X_1, X_2,\ldots, X_n\}$ be the training set of vectors and let $y_i = \{1, -1\}$ be the class label of vector X_i. It is necessary to determine the maximum-margin hyper-plane that separates vectors belonging to $y_i=1$ from vectors belonging to $y_i = -1$. This hyper-plane is written as the set of points satisfying:

$W^T \otimes X_i + b = 0$ (*Formula III.3.1*)

Where $\otimes$ denotes the scalar product and W is a weight vector perpendicular to hyper-plane and b is the bias. W is also called perpendicular vector or normal vector. It is used to specify hyper-plane.

The value $\frac{b}{|W|}$ is the offset of the hyper-plane from the origin along the weight vector W.

To calculate the margin, two parallel hyper-planes are constructed, one on each side of the maximum-margin hyper-plane. Such two parallel hyper-planes are represented by two following equations:

$W^T \otimes X_i + b = 1$

$W^T \otimes X_i + b = -1$

To prevent vectors falling into the margin, all vectors belonging to two class $y_i=1$, $y_i=-1$ have two following constraints respectively:

$W^T \otimes X_i + b \geq 1$ (*for X_i of class $y_i=1$*)

$W^T \otimes X_i + b \leq -1$ (*for X_i of class $y_i=-1$*)

These constraints can be re-written as:

$Y_i (W^T \otimes X_i + b) \geq 1$ (*Formula III.3.2*)

For any new vector X, the rule for classifying it is computed as below:

$f(X_i) = \text{sign}(W^T \otimes X_i + b) \in \{\leq -1, \geq 1\}$ (*Formula III.3.3*)

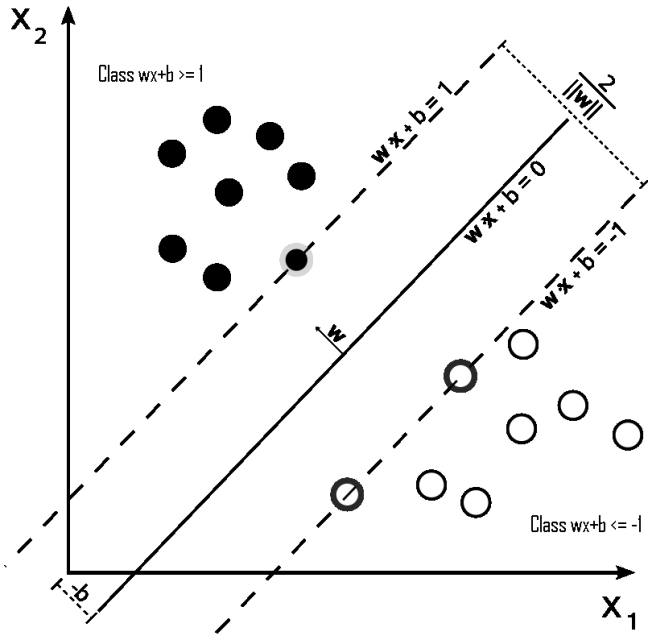


Figure III.3.2. Maximum-margin hyper-plane

Lagrange dual method [Cristianini, Shawe-Taylor 2000] is used to find out optimal weight vector W^* of separating hyper-plane. The bias *b* is computed as below:

$b^* = y_i - W^{*T} \otimes X_i$ (*Formula III.3.4*)

The rule for classification in (3) becomes:

$f(X_i) = R = \text{sign}(W^{*T} \otimes X_i + b^*)$ (*Formula III.3.5*)

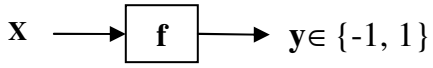


Figure III.3.3. Classification function

It means that whenever we need to determine which class a new vector X_i belongs to, it is only to substitute X_i into $W^{*T} \otimes X_i + b^*$ and check the value of this expression. If the value is less than or equals *-1* (≤ -1) then X_i belongs to class $y_i = -1$. Otherwise, if the value is greater than or equals *1* (≥ 1) then X_i belongs to class $y_i = 1$. Hence the function ($W^{*T} \otimes X_i + b^*$) is called classification function or classification rule.

The Lagrange multipliers are non-zero when $W^T \otimes X_i + b$ equals *1* or *-1*, vectors X_i in this case are considered support vectors which are closest to the maximum-margin hyper-plane. These vectors lie on parallel hyper-planes. So this approach is called support vector machine.

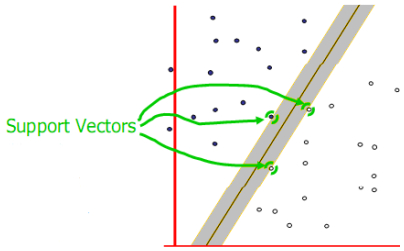


Figure III.3.4. Support vectors

Applying support vector machine into document classification

Give corpus **D** = {D_1, D_2, D_3,…, D_m} in which every document D_i is modeled by a *tf-idf* weight vector. Suppose there are n terms {t_1, t_2,…, t_n}, we have:
$D_i = (d_{i1}, d_{i2},\ldots, d_{in})$
where d_{ij} is product of term frequency and inverse document frequency $d_{ij} = tf_{ij} * idf_j$
If the index of document is ignored, document D is represented as below:
$D = (d_1, d_2,\ldots, d_n)$

Given k classes {l_1, l_2,…, l_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, -1* for y_i=*+1, -1* respectively. So we need to determine unique hyper-plane referred to as *two-class* classifier. It is possible to extend *two-class* classification to *k-class* classification by constructing k *two-class* classifier. In means that we must specify k couple 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 l_i. The process of finding (W_i^*, b_i^*) in training corpus **D** is described in the section of support vector machine.

Table III.3.3. k couple (W_i^*, b_i^*) corresponds with k class {l_1, l_2,…, l_k}

Class	Weight vector	Bias	Classification rule
l_1	W_1^*	b_1^*	$R_1 = W_1^{*T} \otimes X + b_1^*$
l_2	W_2^*	b_2^*	$R_2 = W_2^{*T} \otimes X + b_2^*$
…	…	…	…
l_k	W_k^*	b_k^*	$R_k = W_k^{*T} \otimes X + b_k^*$

For example, classifying document D= (d_1, d_2,…, d_n) is described as below:

1. For each classification rule $R_i = W_i^{*T} \otimes X + b_i^*$, substituting each D into such rule. It means that vector X in such rule is replaced by document D.
$Expression_i = W_i^{*T} \otimes D + b_i^*$

2. Suppose there is a sub-set of rules {R_{h+1}, R_{h+2},…, R_{h+r}} that the value of expression $Expression_i = W_i^{*T} \otimes D + b_i^*$ is greater than or equals *1*. We can conclude that document D is belongs to r classes {l_{h+1}, l_{h+2},…, l_{h+r}} where {l_{h+1}, l_{h+2},…, l_{h+r}} $\subseteq$ {l_1, l_2,…, l_k}

Table III.3.4. Classifying document D

Classification Expression	Value
$W_1^{*T} \otimes D + b_1^*$	≥ 1 : $D \in l_1$
	≤ -1 : $D \notin l_1$
$W_2^{*T} \otimes D + b_2^*$	≥ 1 : $D \in l_2$

	≤ -1 : $D \notin l_2$
...	...
$W_k^{*T} \otimes D + b_k^*$	≥ 1 : $D \in l_k$
	≤ -1 : $D \notin l_k$

III.3.2.3. Document classification based on decision tree

Given a set of classes **C** = {*computer science, math*}, a set of terms **T** = {*computer, programming language, algorithm, derivative*} and the corpus **D** = {*doc1.txt, doc2.txt, doc3.txt, doc4.txt, doc5.txt*}. The training data is shown in following table in which cell (*i, j*) indicates the number of times that term *j* (column *j*) occurs in document *i* (row *i*).

Table III.3.5. Term frequencies of documents

	computer	***Programming language***	***algorithm***	***derivative***	***class***
doc1.txt	5	3	1	1	computer
doc2.txt	5	5	40	5	math
doc3.txt	20	5	20	55	math
doc4.txt	20	55	5	20	computer
doc5.txt	15	15	4	0.3	math
doc6.txt	35	10	45	10	computer

Table III.3.6. Normalized term frequencies

	computer	***Programming language***	***algorithm***	***derivative***	**class**
doc1.txt	0.5	0.3	0.1	0.1	computer
doc2.txt	0.05	0.05	0.4	0.5	math
doc3.txt	0.2	0.05	0.2	0.55	math
doc4.txt	0.2	0.55	0.05	0.2	computer
doc5.txt	0.15	0.15	0.4	0.3	math
doc6.txt	0.35	0.1	0.45	0.1	computer

Because the expense of real number computation is so high, all term frequencies are changed from real number into nominal value:

1. $0 \leq frequency <$ 0.2: *low*
2. $0.2 \leq frequency <$ 0.5: *medium*
3. $0.5 \leq frequency$: *high*

Table III.3.7. Nominal term frequencies

	computer	***Programming language***	***algorithm***	***derivative***	**class**
doc1.txt	high	medium	low	low	computer
doc2.txt	low	low	medium	high	math
doc3.txt	medium	low	medium	high	math
doc4.txt	medium	high	low	medium	computer
doc5.txt	low	low	medium	medium	math
doc6.txt	medium	low	medium	low	computer

The basic idea of generating decision tree [Mitchell 1997] is to split the tree into two sub-trees at the most informative node. Such node is chosen by computing its entropy or information gain. Following figure shows the decision tree generated from our training data.

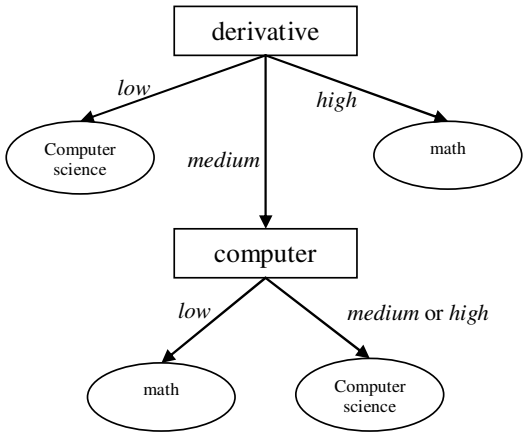


Figure III.3.5. Decision tree

We can extract classification rules from this decision tree:

Table III.3.8. Classification rules deriving from decision tree induction

Rule 1	If frequency of term "derivative" is *low* then document belongs to class *computer science*
Rule 2	If frequency of term "derivative" is *medium* and frequency of term "computer" is *medium* or *high* then document belongs to class *computer science*
Rule 3	If frequency of term "derivative" is *medium* and frequency of term "computer" is *low* then document belongs to class *math*.
Rule 4	If frequency of term "derivative" is *high* then document belongs to class *math*

Suppose the numbers of times that terms *computer, programming language, algorithm* and *derivative* occur in document *D* are *5, 1, 1,* and *3*, respectively. We need to determine which class document *D* is belongs to. *D* is normalized as term frequency vector.
D = (*0.5, 0.1, 0.1, 0.3*)

Changing real number into nominal value, we have:
D = (*high, low, low, medium*)
According to rule *2* in above table, *D* is *computer science* document because in document vector *D*, frequency of term "derivative" is *medium* and frequency of term "computer" is *high*.

III.3.2.4. Document classification based on neural network

Artificial neural network

Artificial neural network (ANN) [Rojas 1996] is the mathematical model based on biological 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. Such processing units are also called neurons or cells 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. However each unit also adjusts the weights of connections. There are three types of units:

- 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.
-

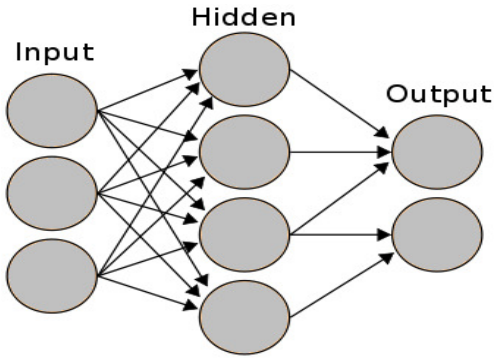


Figure III.3.6. Neural network

Applying neural network into document classification

Given a set of classes **C** = {*computer science, math*}, a set of terms **T** = {*computer, programming language, algorithm, derivative*}. Suppose all input variables (units) are binary or Boolean, every document is represented as a set of input variables. Each term is mapped to an input variable in which value *1* indicates the existence of this term in document

and otherwise value *0* indicates the lack of this term in document. 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"*. These units (variables) are also binary or Boolean. The output layer has only one unit named *"document class"* which is binary or Boolean (*0* – documents belong to *computer science* class and *1* – documents belong to *math* class). The evaluation function used in network is sigmoid function. Our topology is feed-forward neural network in which the weights can be initialized arbitrarily. Note that feed-forward neural network shown in figure III.3.7 is the one that has no cycle in its model.
Note that we denote Boolean value as *0* and *1* (instead of *true* and *false*) for convenience when representing neural network which only accepts numeric value for units.

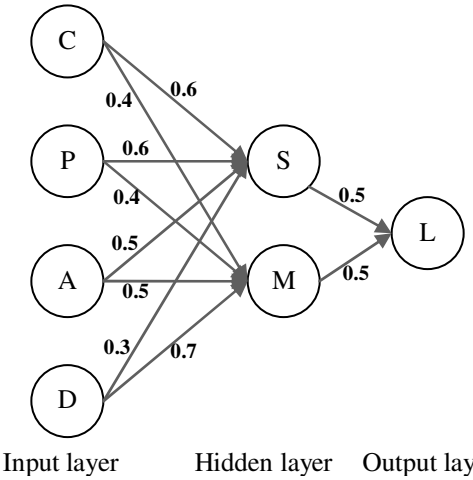


Figure III.3.7. The neural network for document classification

Note that *C, P, A* and *D* denote "*computer", "programming language", "algorithm"* and *"derivative"* respectively. *S* and *M* denote "*computer science*" and "*math*" respectively. *L* denotes *"doc class"*.
Given corpus **D** = {*doc1.txt, doc2.txt, doc3.txt, doc4.txt, doc5.txt*}. The training data is shown in following table in which cell (*i, j*) indicates the number of times that term *j* (column *j*) occurs in document *i* (row *i*).

Table III.3.9. Term frequencies of documents

	computer	***Programming language***	***algorithm***	***derivative***	***class***
doc1.txt	5	3	1	1	computer
doc2.txt	5	5	40	5	math
doc3.txt	20	5	20	55	math
doc4.txt	20	55	5	20	computer
doc5.txt	15	15	4	0.3	math
doc6.txt	35	10	45	10	computer

Table III.3.10. Normalized term frequencies

	computer	***Programming language***	***algorithm***	***derivative***	**class**
doc1.txt	0.5	0.3	0.1	0.1	computer
doc2.txt	0.05	0.05	0.4	0.5	math
doc3.txt	0.2	0.05	0.2	0.55	math
doc4.txt	0.2	0.55	0.05	0.2	computer
doc5.txt	0.15	0.15	0.4	0.3	math
doc6.txt	0.35	0.1	0.45	0.1	computer

Given threshold *α = 0.4*, if the frequency of a term *j* in document *i* is greater than or equals *α*, we consider that term *j* exists in document *i*. Otherwise there is no existence of term *j* in document *i*. So each document is represented as a Boolean vector. Each element in such vector has two values: *1* and *0* (*1* – the respective term occurs in document and *0* – otherwise). So each Boolean vector is the manifest of the occurrences of terms in a document. Corpus **D** becomes a set of Boolean vectors.

Table III.3.11. Boolean document vectors

	computer	***Programming language***	***algorithm***	***derivative***	**class**
doc1.txt	1	0	0	0	computer
doc2.txt	0	0	1	1	math
doc3.txt	0	0	0	1	math
doc4.txt	0.2	1	0	0	computer
doc5.txt	0.15	0	1	0	math
doc6.txt	0.35	0	1	0	computer

Such vectors are fed to our neural network in figure III.3.7 for supervised learning. Back-propagation algorithm is used to train network; thus Boolean document vectors are considered training tuples. Suppose that the weights in neural network are changed as in figure III.3.8 after training process.

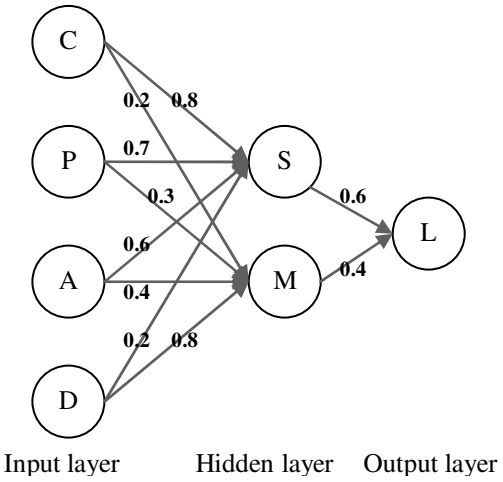


Figure III.3.8. Trained neural network

III.3.2.5. Discovering user interests based on document classification

Suppose in some library or website, user *U* does his search for his interesting books, documents, etc. There is demand of discovering his interests so that such library or website can provide adaptive documents to him whenever he visits in the next time. This is an adaptation process in which system tailors documents to each individual. Given there is a set of key words or terms {*computer, programming language, algorithm, derivative*} that user *U* often looking for, the his searching history is shown in following table:

Table III.3.12. User's searching history

Date	**Keywords (terms) searched**
Aug 28 10:20:01	computer, programming language, algorithm, derivative
Aug 28 13:00:00	computer, programming language, derivative, algorithm
Aug 29 8:15:01	computer
Aug 30 8:15:06	computer

This history is considered as training dataset for mining maximum frequent itemsets. The keywords are now considered items. A itemset is constituted of some items. The support of itemset *x* is defined as the fraction of total transaction which containing *x*. Given support threshold *min_sup*, the itemset *x* is called *frequent itemset* if its support satisfies the support threshold (≥ *min_sup*). Moreover x is *maximum frequent itemset* if *x* is frequent itemset and all super-itemsets of

x are not frequent. Note that *y* is super-itemset of x if $x \subset y$. The item set that has *k* items is called *k-itemset*. Tabel III.3.19 shows the supports of *1*-itemsets.

Table III.3.13. 1-itemsets

1-itemset	***support***
computer	4
programming language	2
algorithm	2
derivative	2

Applying algorithm Apriori, it is easy to find maximum frequent itemsets given *min_sup* = 2. The maximum frequent itemset that user searches are shown in below table:

Table III.3.14. The maximum frequent itemset that user searches

N_o	**itemset**
1	computer, programming language, algorithm, derivative

I propose the new point of view: "*The maximum frequent itemsets are considered as documents and the classes of such documents are considered as user interests*". Such documents may be called interesting documents. Which classes such interesting documents belong to are user interests. It means that discovering user's interests involves in classifying interesting documents. Suppose we have a set of classes **C** = {*computer science, math*}, a set of terms **T** = {*computer, programming language, algorithm, derivative*} and the set of classification rules in table III.3.14. 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. Note that the supports of such items are shown in table III.3.19.

Table III.3.15. Interesting document vector

N_o	**vector**
1	(computer=4, programming language=*2*, algorithm=*2*, derivative=*2*)

Table III.3.16. Interesting document vector is normalized

N_o	***vector***
1	(computer=*0.4*, programming language=*0.2*, algorithm=*0.2*, derivative=*0.2*)

Table III.3.17. **Nominal interesting document vector**

N_o	**vector**
1	(computer=*medium*, programming language=*medium*, algorithm=*medium*, derivative=*medium*)

It is possible to use SVM or 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 in section III.3.2.3. Otherwise we must determine the weight vector W^* if applying SVM approach. SVM approach is more powerful than decision tree with regard to document classification in case of huge training data.
Applying classification rule *2* in table III.3.14, the interesting document belongs to class *computer science* because the frequency 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, interesting document vector is specified as Boolean document vector (or Boolean user vector). For example, given threshold $\alpha = 0.4$, if the frequency of term *j* in document *i* is greater than or equals α, we consider that term *j* exists in document *i*. Otherwise there is no existence of

term *j* in document *i*. We have a Boolean vector. So user *U* is modeled as a Boolean document vector. Such vector is also called Boolean user vector: ***U* = (1, 0, 0, 0)**. According to table III.3.19, we have

Table III.3.18. Boolean document vector (or Boolean user vector)

Term	Existence
computer	1
programming language	0
algorithm	0
derivative	0

The Boolean user vector is considered as a document and the classes of such document are considered as user interests. Now document (user Boolean vector) *U* becomes a data tuple which is fed to trained neural network in figure III.3.8. It is easy to know the class of document *U* by checking the value of output unit in trained neural network. Suppose such output value is *0*, we can infer that document *U* belongs to class "*computer science*". So the interest of user *U* is "*computer science*".

III.3.2.6. Evaluation

Our approach includes following steps:
- Documents are represented as vectors
- Classifying documents by using decision tree or support vector machine or neural network
- Mining user's access history to find maximum frequent itemsets. Each itemset is considered an interesting document
- Applying classifiers 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 access in his/her history are modeled as documents. So user is referred indirectly to as document.
- User interests are classes to which such documents are belong.

The technique of constructing vector model for representing document is not important to this approach. There are some algorithms of text segmentation for specifying all terms in documents. From this, it is easy to build up document vectors by computing term frequency and inverse document frequency. However the concerned techniques of document classification such as SVM, decision tree and neural network influence extremely on this approach. SVM and neural network is more effective than decision tree in case of huge training data set but it is not convenient for applying classifiers (weight vector W^*) into determining the classes of documents. Otherwise it is easy to use classification rules taken out from decision tree for this task.

III.3.3. Constructing user groups or user communities

Remind that user model is the representation of personal traits or characteristics about user such as demographic information, knowledge, learning style, goal, interest, etc. 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. 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 logins system, she/he will be classified into a group and initial information of his model is assigned to common features in such group.
- In the collaborative learning, users need to learn or discuss together. It is very useful if the collaborative learning is restricted in a group of similar users. Therefore, it is convenient for users that have common characteristics (knowledge, goal, interest, etc) to learn together because they do not come up against an obstacle when interacting together.

The problem that needs to be solved now is how to determine user groups. This relates to clustering techniques so as to cluster user models because a group is considered as a cluster of similar user models. Section III.3.1.1 and III.3.1.2 discuss about user modeling clustering technique, namely *k*-means algorithm for vector model, overlay model, Bayesian network. In section III.3.1.2, I propose the formulas so as to compute the dissimilarity of two overlay models (or two Bayesian network). The *k*-medoids algorithm and similarity measures such as cosine similarity measure and correlation coefficient are discussed in section III.3.1.3. Section III.3.1.4 is the conclusion.

In general, these sections focus on how to construct user groups which is a function supported by mining engine (ME) and learning history sub-model, along with other functions such as learning concept recommendation and discovering user interests aforementioned previous sections.

III.3.3.1. User modeling clustering method

Suppose user model U_i is represented as vector $U_i = \{u_{i1}, u_{i2},\ldots, u_{ij},\ldots, u_{in}\}$ whose elements are numbers. For instance, if U_i represents user knowledge then U_i is considered as a knowledge vector in which the j^{th} component of this vector is the conditional probability expresing how user master knowledge item j^{th}. Note that U_i is called user model or user vector or user model vector.

Suppose there is a collection of users $\{U_1, U_2,\ldots, U_m\}$ and we need to find out *k* groups so-called *k user model clusers*. A user model cluster is a set of similar user models, so user models in the same cluster is dissimilar to ones in other clusters. The dissimilarity of two user models is defined as Euclidean distance between them.

$$dissim(U_1,U_2) = dis\tan ce(U_1,U_2) = \sqrt{(u_{11}-u_{21})^2+(u_{12}-u_{22})^2+...+(u_{1n}-u_{2n})^2}$$

The less $dissim(U_1, U_2)$ is, the more similar U_1 and U_2 are. Applying *k*-means algorithm [Han, Kamber 2006], we partition a collection of user models into *k* user model clusters. The *k*-means algorithm includes three following steps:

1. It randomly selects *k* user models, each of which initially represents a cluster mean. Of course, we have *k* cluster means. Each mean is considered as the "representative" of one cluster. There are *k* clusters.
2. For each remaining user model, the dissimilarities between it and *k* cluster means are computed. Such user model belongs to the cluster which it is most similar to; it means that if user model U_i belong to cluster C_j, the dissimilarity measure $dissim(U_i, C_j)$ is minimal.
3. After that, the means of all clusters are re-computed. If stopping condition is met then algorithm is terminated, otherwise returning step *1*.

This process is repeated until the stopping condition is met. For example, the stopping condition is that the square-error criterion is less than a pre-defined threshold. The square-error criterion is defined as below:

$$Err = \sum_{i=1}^{k} \sum_{U \in C_i} dissim(U - m_i)$$

Where C_i and m_i is cluster i and its mean, respectively; $dissim(U, m_i)$ is the dissimilarity between user model U and the mean of cluster C_i.

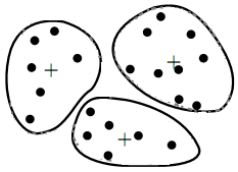


Figure III.3.9. User model clusters (means are marked by sign "+")

The mean m_i of a 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.

$$m_i = (\frac{1}{n_i}\sum_{j=1}^{n_i} u_{j1}, \frac{1}{n_i}\sum_{j=1}^{n_i} u_{j2}, ..., \frac{1}{n_i}\sum_{j=1}^{n_i} u_{jn})$$

Where n_i is the number of user model in cluster C_i.

III.3.3.2. Clustering overlay model

If user is modeled in a vector, the dissimilarity measure in k-mean algorithm is Euclidean distance. However there is a question: "how to compute such measure in case that user model is an overlay model which is in form of domain graph". In this situation, the domain is decomposed into a set of elements and the overlay model is simply a set of masteries over those elements. So overlay model is the subset of domain model.

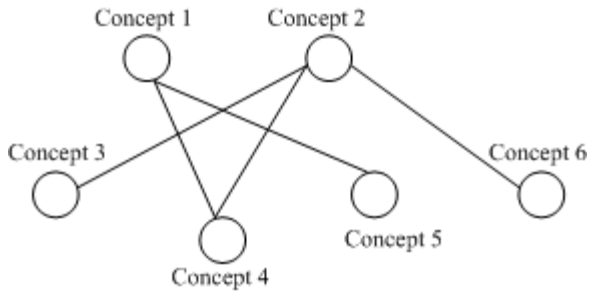


Figure III.3.10. Overlay model

It is essential that overlay model is the graph of domain; so overlay model is also called as graph model. Each node (or vertex) in graph is a knowledge item represented by a number indicating how user masters such knowledge item. Each edge (or arc) reveals the relationship between two nodes. It is clear to say that the dissimilarity measure needs changing so as to compare two overlay models. Note that the terms "user model", "overlay model", "graph model" are the same in this context. Suppose two user overlay models U_1 and U_2 are denoted as below:

$U_1=G_1=<V_1,E_1>$ and $U_2=G_2=<V_2,E_2>$

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.

$V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$

Suppose graph model is in form of tree in which each directed arc represents the prerequisite relationship of two nodes. If there is an arc from node *A* to node *B*, user must master over *A* before learning *B*.

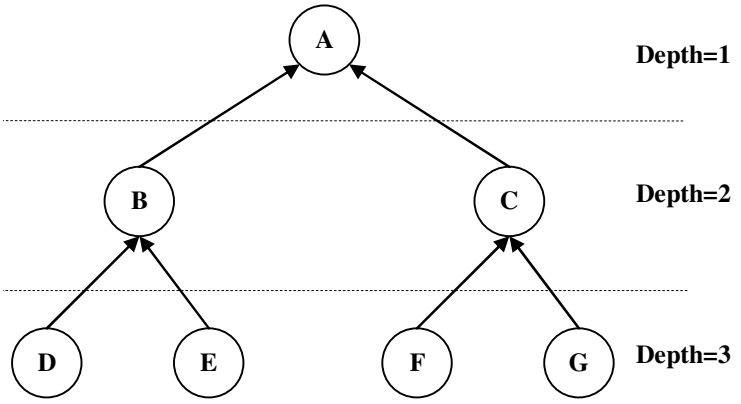


Figure III.3.11. Graph model in form of tree and prerequisite relationships

Let $depth(v_{ij})$ is the depth level of node *j* of graph model G_i. Note that the depth level of root node is *1*.

$$depth(v_{root}) = 1$$

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 as below:

$$dissim(G_1, G_2) = dis\tan ce(G_1, G_2) = \sum_{j=1}^{n} \left| \frac{v_{1j} - v_{2j}}{depth(v_{1j})} \right|$$

The meaning of this formulation is: "the high level concept (node) is the aggregation of low level (basic) concepts". The depth levels of j^{th} nodes of all graph models are the same because the structure of graphs is kept intact.

$$\forall a, b, j, depth(v_{aj}) = depth(v_{bj})$$

For example, there are three graph models whose structures are the same to the structure shown in figure III.3.13. The values of their nodes are shown in following table:

Table III.3.25. Values of graph nodes

	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

The dissimilarity (or distance) between G_3 and G_1, G_2, respectively are computed as below:

$$dissim(G_1, G_3) = \left|\frac{2-2}{1}\right| + \left|\frac{1-1}{2}\right| + \left|\frac{1-1}{2}\right| + \left|\frac{0-1}{3}\right| + \left|\frac{3-4}{3}\right| + \left|\frac{2-5}{3}\right| + \left|\frac{1-4}{3}\right| = 2.66$$

$$dissim(G_2, G_3) = \left|\frac{1-2}{1}\right| + \left|\frac{1-1}{2}\right| + \left|\frac{0-1}{2}\right| + \left|\frac{1-1}{3}\right| + \left|\frac{4-4}{3}\right| + \left|\frac{5-5}{3}\right| + \left|\frac{4-4}{3}\right| = 1.5$$

So G_2 is more similar to G_3 than G_1 is.

III.3.3.2.1. In case that arcs in graph are weighted

In case that each arc is assigned a weight representing the strength of prerequisite relationship between parent node and child node, the following figure shows this situation:

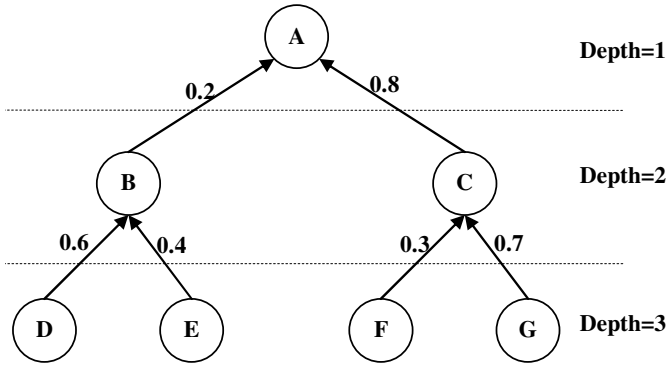


Figure III.3.12. Graph model and weighted arcs

The dissimilarity (or distance) of two graph models G_1 and G_2 is re-defined as below:

$$dissim(G_1, G_2) = dis\tan ce(G_1, G_2) = \sum_{j=1}^{n} \left| \frac{v_{1j} - v_{2j}}{depth(v_{1j})} * weight(v_{1j}) \right|$$

Let $weight(v_{ij})$ be the weight of arc from node j (of graph model i) to its parent. 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*.

$$\sum_{v_{ij} \text{ have the same parent}} weight(v_{ij}) = 1$$

The weight at root node equals *1*

$weight(v_{root})=1$.

The weight at j^{th} nodes of all graph models are the same because the structure of graphs is kept intact.

$$\forall a, b, j, weight(v_{aj}) = weight(v_{bj})$$

For example, there are three graph models whose structures are the same to the structure shown in figure III.3.14. The values of their nodes are shown in table III.3.25. The dissimilarity (or distance) between G_3 and G_1, G_2, respectively are computed as below:

$$dissim(G_1, G_3) =$$

$$\left|\frac{2-2}{1}\right| + \left|\frac{1-1}{2} * 0.2\right| + \left|\frac{1-1}{2} * 0.8\right| + \left|\frac{0-1}{3} * 0.6\right| + \left|\frac{3-4}{3} * 0.4\right| + \left|\frac{2-5}{3} * 0.3\right| + \left|\frac{1-4}{3} * 0.7\right| = 1.33$$

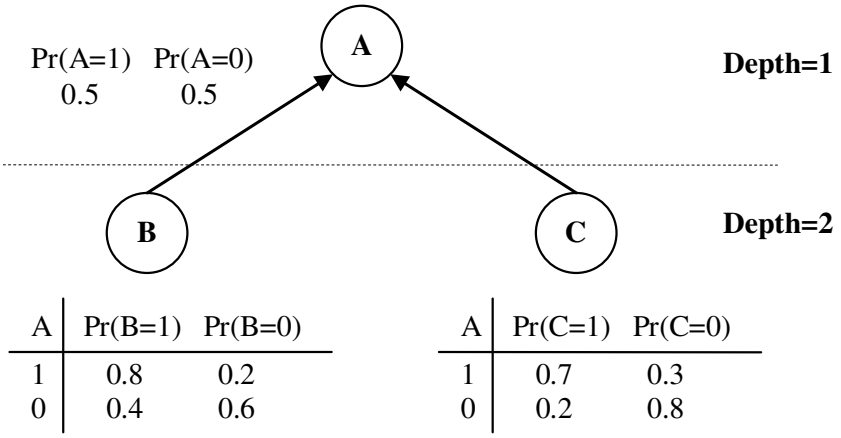
$$dissim(G_2, G_3) =$$

$$\left|\frac{1-2}{1}\right| + \left|\frac{1-1}{2} * 0.2\right| + \left|\frac{0-1}{2} * 0.8\right| + \left|\frac{1-1}{3} * 0.6\right| + \left|\frac{4-4}{3} * 0.4\right| + \left|\frac{5-5}{3} * 0.3\right| + \left|\frac{4-4}{3} * 0.7\right| = 1.4$$

So G_1 is more similar to G_3 than G_2 is.

III.3.3.2.2. In case that graph model is Bayesian network

Bayesian network is the directed acrylic graph (DAG) [Nguyen, Do 2009] constituted of a set of nodes and a set of directed arc. Each node is referred as a binary variable and each arc expresses the dependence relationship (namely, prerequisite 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.
Suppose the structure of Bayesian network is in form of tree, the following figure shows our considered Bayesian network.



A	Pr(B=1)	Pr(B=0)
1	0.8	0.2
0	0.4	0.6

A	Pr(C=1)	Pr(C=0)
1	0.7	0.3
0	0.2	0.8

Figure III.3.13. Bayesian network and its CPT (s)

Let G_1 and G_2 be two Bayesian networks of user *1* and user *2*, respectively.
$G_1=<V_1,E_1>$ and $U_2=G_2=<V_2,E_2>$

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 as below:

$$dissim(G_1,G_2) = dis\tan ce(G_1,G_2) = \sum_{j=1}^{n}\left|\frac{\Pr(v_{1j}) - \Pr(v_{2j})}{depth(v_{1j})}\right|$$

Where $Pr(v_{ij})$ is the marginal probability of node *j* in network *i*. The inference technique to compute marginal probability is discussed in [Nguyen, Do 2009].

III.3.3.3. Similarity measures for clustering algorithm

Although dissimilarity measure considered as the physical distance between them are used to cluster user models, we can use another measure so-called similarity measure so as to cluster user models. There are two typical similarity measures:

- Cosine similarity measure
- Correlation coefficient

Suppose user model U_i is represented as vector $U_i = \{u_{i1}, u_{i2},\ldots, u_{ij},\ldots, u_{in}\}$, the cosine similarity measure of two user models is the cosine of the angle between two vectors.

$$sim(U_i, U_j) = \cos(U_i, U_j) = \frac{U_i \bullet U_j}{|U_i| . |U_j|} = \frac{\sum_{k=1}^{n} u_{ik} * u_{jk}}{\sqrt{\sum_{k=1}^{n} u_{ik}^2} \sqrt{\sum_{k=1}^{n} u_{jk}^2}}$$

The range of cosine similarity measure is from *0* to *1*. If it equals *1*, two users are totally different. If it equals *0*, two users are identical. For example, the following table shows four user models.

Table III.3.26. 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

The cosine similarity measures of user *4* and users *1*, *2*, *3* are computed as below:

$$sim(user_4, user_1) = \frac{1*1+2*2+1*0}{\sqrt{1^2+2^2+0^2}\sqrt{1^2+2^2+1^2}} = 0.9$$

$$sim(user_4, user_2) = \frac{2*1+1*2+1*0}{\sqrt{1^2+2^2+0^2}\sqrt{2^2+1^2+2^2}} = 0.6$$

$$sim(user_4, user_3) = \frac{4*1+1*2+5*0}{\sqrt{1^2+2^2+0^2}\sqrt{4^2+1^2+5^2}} = 0.4$$

Obviously, user *1* and user *2* are more similar to user *4* than user *3* is.

On other hand, correlation coefficient which is the concept in statistics is also used to specify the similarity of two vectors. Let $\overline{U_i}$ be the expectation of user model vector U_i, so $\overline{U_i} = \frac{1}{n}\sum_{k=1}^{n} u_{ik}$

The correlation coefficient is defined as below:

$$sim(U_i, U_j) = correl(U_i, U_j) = \frac{\sum_{k=1}^{n} (u_{ik} - \overline{U_i})(u_{jk} - \overline{U_j})}{\sqrt{\sum_{k=1}^{n} (u_{ik} - \overline{U_i})^2} \sqrt{\sum_{j=1}^{n} (u_{jk} - \overline{U_j})^2}}$$

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:

$$\overline{U_4} = \frac{1+2+0}{3} = 1,\ \overline{U_1} = \frac{1+2+1}{3} = 1.33,\ \overline{U_2} = \frac{2+1+2}{3} = 1.66,\ \overline{U_3} = \frac{4+1+5}{3} = 3.33$$

$$sim(user_4, user_1) = \frac{(1-1)(1-1.33)+(2-1)(2-1.33)+(0-1)(1-1.33)}{\sqrt{(1-1)^2+(2-1)^2+(0-1)^2}\sqrt{(1-1.33)^2+(2-1.33)^2+(1-1.33)^2}} = 0.86$$

$$sim(user_4, user_2) = \frac{(1-1)(2-1.66)+(2-1)(1-1.66)+(0-1)(2-1.66)}{\sqrt{(1-1)^2+(2-1)^2+(0-1)^2}\sqrt{(2-1.66)^2+(1-1.66)^2+(2-1.66)^2}} = -0.86$$

$$sim(user_4, user_2) = \frac{(1-1)(4-3.33)+(2-1)(1-3.33)+(0-1)(5-3.33)}{\sqrt{(1-1)^2+(2-1)^2+(0-1)^2}\sqrt{(4-3.33)^2+(1-3.33)^2+(5-3.33)^2}} = -0.96$$

Obviously, user *1* and user *2* are more similar to user *4* than to user *3* is.

If cosine measure and correlation coefficient are used as the similarity between two user vectors, the serious problem will occurs. That is impossible to specify the mean of each cluster because cosine measure and correlation coefficient have different semantic from Euclidean distance. Instead of using *k*-mean algorithm, we apply *k*-medoid algorithm 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.

1. It randomly selects *k* user models as *k* medoids. Each medoid is considered as the "representative" of one cluster. There are *k* clusters.
2. For each remaining user model, the similarities between it and *k* medoids are computed. Such user model will belongs to cluster C_i if the similarity measure of this user model and the medoid of C_i is the highest.
3. After that, *k* medoids are selected again so that the average similarity of each medoid to other user models in the same cluster is maximal. If *k* medoids are not changed or the absolute-error criterion is less than a pre-defined threshold, the algorithm is terminated; otherwise returning step *1*.

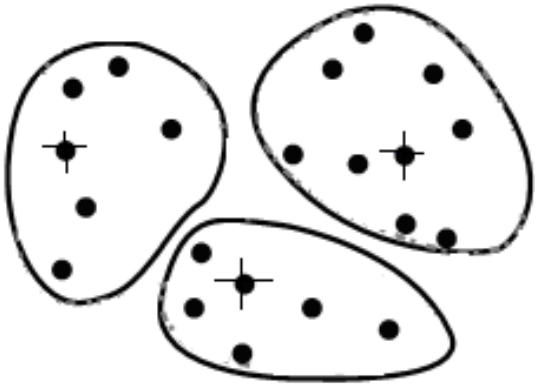


Figure III.3.14. *k-medoid* algorithm (user model vectors considered as medoid are marked by sign "+")

The absolute-error criterion which is the typical stopping condition is defined as below:

$$Err = \sum_{i=1}^{k} \sum_{U \in C_i} \left(1 - sim(U - medoid_i)\right)$$

Where C_i and $medoid_i$ is cluster *i* and its medoid, respectively; $sim(U, medoid_i)$ is the similarity of user model *U* and the medoid of cluster C_i. Note that $|sim(U, medoid_i)| \leq 1$.

III.3.3.4. Evaluation

We discussed two clustering algorithms: *k*-means and *k*-medoids. It is concluded that the dissimilarity measures considered as distance between two user models are appropriate to *k*-means algorithm and the similarity measure such as cosine similarity measure and correlation coefficient are fit to *k*-medoids 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.

Now the basic content of this thesis was presented to you with the full of detailed description focused on how to triangular learner model (TLM) is constructed and how to the user modeling system Zebra is built up and manipulates TLM. The next section is the conclusion and evaluation on my research.

Evaluation of Triangular Learner Model

Chapter I is the state-of-art of user model and user modeling system. Chapters II, III are essential chapters which focus on main works of research including how to triangular learner model (TLM) is constructed and how to the user modeling system Zebra is built up and manipulates TLM. As a result, TLM is constituted of three sub-models such as knowledge, learning styles and learning history.

In general, this research is fundamental research; thus, approaches, models and mathematical formulas are proposed as a perfect whole methodology. This research is not experimental research and testing technique on testing data is not appropriate to evaluate this research. So there are three ways to evaluate this research:

- The correctness of formulas is proven by mathematical tools and logical reasoning.
- The feasibility of model and architecture is authenticated via the software associated with Zebra and TLM. Moreover, the adaptive learning system that interacts to Zebra is also implemented as e-learning web-based software.
- The effectiveness of adaptive learning is proven via approaches described in this chapter.

This chapter focuses on evaluating learner model TLM and user modeling system Zebra with regard to their effectiveness. Section IV.1 is the evaluation on knowledge sub-model. Section IV.2 is the evaluation on the effectiveness of adaptive learning model, especially, the whole TLM and modeling system Zebra.

IV.1. Evaluation of knowledge model

Bayesian network is the most important component of inference mechanism in Zebra when it and hidden Markov model constitutes the *belief network engine* in the core engine. Note that the core of Zebra is the inference engine having two sub-engines: belief network engine and mining engine. Bayesian network is used to assess user knowledge and mining engine is aimed to discover new assumptions about user and to support personal recommendation. The attempt to improve Bayesian network is the same to enhancing belief network engine.

IV.1.1. Evaluation of Bayesian network

There are three methods of building up Bayesian network user model: expert centric, efficiency centric and data centric. These methods are distinguished based on how to construct Bayesian network 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 Bayesian network 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. Bayesian network 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, the evaluation time, etc. Bayesian network 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 learning machine algorithms. Bayesian network user model built up by this method is called data-centric model. This method achieves some advantages when the number of variables may be smaller than expert centric and efficiency centric method because the network is deduced from actual data and so there is no redundant variables. It is easy to evaluate the quality of network by applying network into the testing data. Both observed and hidden variables are regarded in data centric method. However there is a critical drawback of data-centric model when the complexity of learning algorithms decreases the performance of user modeling tasks in run time. It takes more time to do inference in data-centric model than expert-centric model or efficiency-centric model.

The main technique used in Bayesian model of Zebra can belong to efficiency centric method in which the criterion of constructing user model is to map Bayesian network 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 strength of causal (or prerequisite) relationship among variables is 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 causal relationship between *B* and *A* is larger than the weight of causal relationship between *C* and *A*. *This technique achieves the same result to data-centric method in learning context because the expert in learning context is a teacher. She/he masters over the curriculum and so quality of structure of Bayesian network 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.*

Moreover this thesis also introduces two other techniques to improve the conditional probability tables (or parameters) and the structure of Bayesian network.

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 expectation maximum (EM) algorithm for beta function. Suppose the density function (distribution function) is the beta function $\beta(f_{ij}; a_{ij}, b_{ij})$. Suppose variable X_i has the prior conditional probability $Pr(X_i=1|pa_{ij}=1)=E(\beta(f_{ij}))=\frac{a_{ij}}{N_{ij}}$ where $N_{ij}=a_{ij}+b_{ij}$, the parameter learning process based on a set of evidences is to update the posterior density function $\beta(f_{ij}|E)$ and the posterior conditional probability $Pr(X_i=1|pa_{ij}=1,E)$. Indeed, $\beta(f_{ij} \mid E) = beta(f_{ij}; a_{ij}+s_{ij}, b_{ij}+t_{ij})$ and $Pr(X_i=1|pa_{ih_i}=1,E) = E(\beta(f_{ij}|E)) = \frac{a_{ij}+s_{ij}}{N_{ij}+M_{ij}}$. EM algorithm is implemented in offline process so as not to decrease the system performance.

The second technique is to aim to improve the structure of Bayesian network. 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 thesis suggests a new way to keep the number of variables stable by obeying the Markov property, namely, given the current time point *t*, the conditional probability of next time point *t+1* is only relevant to the current time point *t*, not relevant to any past time point (*t-1, t-2,…,0*). Every time the evidences occur, the DBN is re-constructed according to six steps:

- *Step 1*: Initializing DBN
- *Step 2*: Specifying transition weights
- *Step 3*: Re-constructing DBN
- *Step 4*: Normalizing weights of dependencies
- *Step 5*: Re-defining CPT (s)
- *Step 6*: 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 this thesis 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 step *2* and *3*.

In conclusion, the method used to build up Bayesian network 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).

IV.1.2. Assessment of Bayesian network

Besides providing information about users and deducing new assumptions about them, the ultimate of Bayesian (network user) model is to support the adaptive application like ITS, AHS [Brusilovsky, 1994] in order to give user the adaptation effects such as personal learning content, course recommendation, adaptive representation, adaptive navigation in domain space, curriculum sequence, hint generation, etc. So it is very necessary to assess the overall competence of user in a domain. For example, the level of knowledge user gains must be measured in Bayesian model so that it is possible to answer the question: how much user masters over such knowledge. This process is called the exploitation or assessment of Bayesian model. Firstly, we should glance over some approaches for assessment.

Secondly, there is a future trend of the assessment in Zebra, towards to the Computerized Adaptive Testing (CAT).

IV.1.2.1. Overview of assessment approaches

There are three approaches to perform assessment tasks: *alternate*, *diagnostic* and *decision-theoretic.*

Alternate approach

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, hint generation, etc. For example, if such posterior probability is high, user will received an advanced exercise. This is the simplest approach.

Diagnostic approach

In this approach Bayesian network includes two kinds of variables: hidden and observed. Hidden variables are knowledge items that need to be assessed how much user masters it. Observed variables so-called evidences are questions, exercises which are used to test user's knowledge. It is possible to imagine that hidden variables represent the diseases and observed variables are symptoms. By surveying symptoms, it is able to diagnose respective diseases. It means that when evidences have been observed, the posterior probabilities of other hidden variables are computed via Bayesian updating. The conditional relationship between a hidden variables and evidence is represented as a directed arc whose direction is always from hidden variable to evidence and the reversed direction is not permitted.

Decision-theoretic approach

Given a set of action $D = \{d_1, d_2,\ldots, d_m\}$ and a set of possible outcomes $X = \{x_1, x_2,\ldots, x_n\}$ and a conditional probability distribution $Pr(X|D)$. 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 following figure.

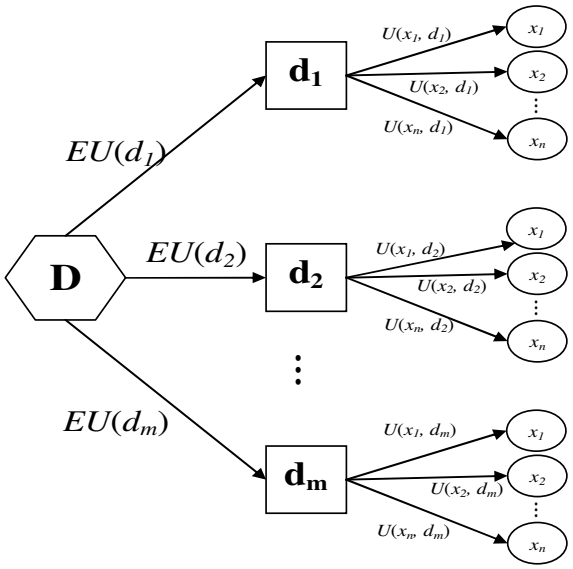


Figure IV.1.1. Decision-theoretic tree

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 nodes named $EU(d_i)$ represents the expected utility of an action d_i. Every outcome x_j of action d_i has a conditional probability $Pr(X=x_j|D=d_i)$ and a utility value $U(X=x_i, D=d_i)$. Of course $Pr(X = x_j| D = d_i)$ and $U(X=x_i, D = d_i)$ are values of conditional probability distribution $Pr(X|D)$ and utility function $U(X, D)$, respectively. The expected utility of an action is defined as the probabilistic weighted sum of utility values of all outcomes of such action:

$$EU(d_i) = \sum_{x_j \in X} \Pr(x_j \mid d_i) U(x_j, d_i)$$

This formula can be generalized as below:

$$EU(D) = \sum_{X} \Pr(X \mid D) U(X, D)$$

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, visiting learning web pages, etc.

In conclusion, the *belief network engine* in the core of Zebra uses the first approach to assess the Bayesian network. It simple to compute the posterior probabilities of knowledge items (variables in network) and match such probability with adaptive rules, for example, if the posterior probability of knowledge item *I* learned by user *A* is high then the advanced content of *I* is provided to user *A* because user *A* is mastered over item *I*. In the future, Zebra aims to apply the Computerized Adaptive Testing (CAT) technique into determining how users master over knowledge items when these items are tests or examinations. CAT based on the Item Response Theory (IRT) is introduced in following section.

IV.1.2.2. Towards the Computerized Adaptive Testing

The computer-based tests have more advantages than the traditional paper-based tests when there is the boom of internet and computer. Computer-based testing allows students to perform the tests at any time and any place and the testing environment becomes more realistic. Moreover, it is very easy to assess students' ability by using the computerized adaptive testing (CAT). The CAT is considered as the branch of computer-based testing but it improves the accuracy of test core when CAT systems try to choose items which are suitable to students' abilities; such items are called adaptive items.

The important problem in CAT is how to estimate students' abilities so as to select the best items for students. There are some methods to solve this problem such as maximization likelihood estimation but I apply the Bayesian approach into computing ability estimates. In this section, I suggest the stopping criterion for CAT algorithm: 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.

IV.1.2.2.1. Overview of Computerized Adaptive Testing (CAT)

Item Response Theory

Item Response Theory (IRT) [Baker 2001] is defined as a statistical model in which examinees can be described by a set of one or more ability scores that are predictive, through mathematical models, linking actual performance on test items, item statistics, and examinee abilities. Note that the term "*item*" indicates test or exam. Given examinee *j* and item *i,* IRT is modeled as a function of a true ability of examinee *j* (denoted θ_j) and three parameters of item *i* (denoted a_i, b_i, c_i). This function so-called Item Response Function (IRF) or Item Characteristic Curve (ICC) function computes the probability of a correct response of examinee *j* to item *i*.

$$IRF(\theta_j) = \Pr(\theta_j) = c_i + \frac{1 - c_i}{1 + e^{a_i(\theta_j - b_i)}}$$

ICC, a variant of logistic function, is plotted in following figure with $a_i=2.0$, $b_i=0$, $c_i=0.25$

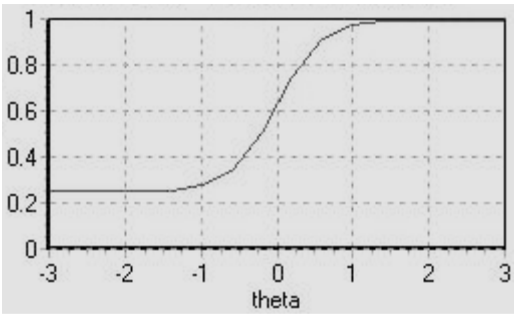


Figure IV.1.2. Item Characteristic Curve

The horizontal axis θ is the scale of examinee's ability, ranging from -3.0 to +3.0. The vertical axis is the probability of correct response to this item specified by three parameters: a_i*=2.0,* b_i*=0,* c_i*=0.25*. The left-hand of curve shows an easy item when the probability of correct response is high for low-ability. The center of curve shows a medium-difficulty item when the probability of correct response is around the middle of the ability scale. The right-hand of curve shows a difficult item when the probability of its correct response is low for most of ability scale. The lower asymptote at c_i*=0.25* indicates the probability of correct response for examinee with lowest ability and otherwise for the upper asymptote at *1.0*.

ICC measures examinee's proficiency based on her/his ability and some properties of item. Every item *i* has three parameters a_i, b_i, c_i which are specified by experts or statistical data.

- The a_i parameter so-called *discriminatory parameter* 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.
- The b_i parameter so-called *difficult parameter* 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 so-called *guessing parameter* 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.

Computerized Adaptive Testing

Computerized Adaptive Testing (CAT) [Linden and Glas, 2002] is the iterative algorithm which begins providing examinee an (test) item so as to be best to her/his initial ability; after that the ability is estimated again and the process of item suggestion is continued until a stopping criterion is met. This algorithm aims to make a series of (test) items which are evaluated to become chosen items that suitable to examinee's ability. The set of items from which system picks ones up is called as (item) pool. The items having chosen and given to examinee compose the adaptive test. CAT includes following steps:

1. The initial ability of examinee must be defined and the items (in the pool) that have not yet been chosen are evaluated to choose the best one which is the most suitable to examinee's current ability estimate
2. The best next item is chosen to give to examinee and the examinee responds. Such item is changed from pool to adaptive test.
3. A new ability estimate is computed based on the responses to all of the chosen items.
4. Steps 1 through 3 are repeated until a stopping criterion is met.

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, etc.

In step 1, there is the question: "how to evaluate the items to choose the best one". So each item *i* is qualified by the amount of information or entropy at given ability θ; such entropy is denoted $I_i(\theta)$. The best next item is the one that provides the most information or has highest value of $I_i(\theta)$.

$$I_i(\theta) = \frac{\Pr_i'(\theta)^2}{\Pr_i(\theta)(1-\Pr_i(\theta))}$$

Where $Pr_i(\theta)$ is the ICC function and $Pr_i'(\theta)$ is the first-order derivative of $Pr_i(\theta)$
The entropy $I_i(\theta)$ reflects how much the item *i* matches examinee's ability. The item should not be too easy or too difficult. Given ability θ, the sum of entropies over items in test which tells the qualification of such pool at ability θ is denoted $I(\theta)$.

$$I(\theta) = \sum_i I_i(\theta)$$

In step 3, there are some methods to compute the ability estimate such as maximum-likelihood, weighted likelihood [Warm 1989], etc. But I aim to apply Bayesian approach into specifying ability estimate according to [Bock, Mislevy 1988].

IV.1.2.2.2. Bayesian approach for CAT in Zebra

CAT applied into Zebra also includes four steps as above but we should redefine some concepts in order to take advantage of Bayesian rule. Suppose the indexes of items in pool are denoted $i = \overline{1, I}$ and the indexes of these items in the adaptive test are denoted $k = \overline{1, K}$ and so the index of item *i* in the pool administered as the item *k* in the test is denoted i_k. Suppose $S_k = \{i_1, i_2, i_{k-1}\}$ is the set of previous *k-1* items that are administered and in a test now; of course they correspond with a set of *k-1* responses denoted as $U_k = \{u_{i_1}, u_{i_2}, \ldots, u_{i_{k-1}}\}$. The set of items in the pool remaining after chosen items is denoted as $R_k = \{1, \ldots, I\} \setminus U_k$. The ICC function is written in general:

$$\Pr_i(\theta) = c_i + (1-c_i)\frac{e^{a_i(\theta-b_i)}}{1+e^{a_i(\theta-b_i)}}$$

Where θ is the examinee's ability and a_i, b_i and c_i are discriminatory parameter, difficult parameter and guessing parameter, respectively.
The likelihood function associated with the responses on the first *k–1* items is denoted as below:

$$L(\theta \mid u_{i_1}, \ldots, u_{i_{k-1}}) = \prod_{j=1}^{k-1} \frac{(e^{a_{i_j}(\theta-b_{i_j})})^{u_{i_j}}}{1+e^{a_{i_j}(\theta-b_{i_j})}}$$

The second-order derivative of the likelihood reflects the curvature of the observed likelihood function at θ. The observed information measure is defined the negative of such second-order derivative:

$$J_{u_{i_1},\ldots,u_{i_{k-1}}}(\theta) = -\frac{\partial}{\partial\theta^2}\ln L(\theta \mid u_{i_1}, \ldots, u_{i_{k-1}})$$

The expectation of observed information measure is expected information measure.

$$I_{u_{i_1},\ldots,u_{i_{k-1}}}(\theta) = E(J_{u_{i_1},\ldots,u_{i_{k-1}}}(\theta))$$

Suppose the prior for the unknown value of examinee's ability is assumed as $g(\theta)$. The function $g(\theta)$ is also called as the prior distribution of θ. According to Bayesian rule, the posterior distribution of ability estimate is computed as below:

$$g(\theta \mid u_{i_1}, \ldots, u_{i_{k-1}}) = \frac{L(\theta \mid u_{i_1}, \ldots, u_{i_{k-1}})g(\theta)}{\int L(\theta \mid u_{i_1}, \ldots, u_{i_{k-1}})g(\theta)d\theta}$$

In step 3, after the responses to *k-1* items, it is necessary to determine the ability estimate denoted $\widehat{\theta}_{u_{i_1},\ldots,u_{i_{k-1}}}$ or $\widehat{\theta}$ in brief. According to [Bock, Mislevy 1988], $\widehat{\theta}$ is the expectation of the posterior distribution $g(\theta \mid u_{i_1}, \ldots, u_{i_{k-1}})$

$$\widehat{\theta} = \widehat{\theta}_{u_{i_1},\ldots,u_{i_{k-1}}} = E(\theta \mid u_{i_1}, \ldots, u_{i_{k-1}}) = \int \theta g(\theta \mid u_{i_1}, \ldots, u_{i_{k-1}})d\theta$$

This technique based on the posterior distribution of θ has an advantage when the ability estimate $\widehat{\theta}$ always exists and

is easy to compute. Another method used to compute ability estimate is to determine the value $\widehat{\theta}$ that maximizes the likelihood function in over all possible values of θ. Such $\widehat{\theta}$ is considered as the ability estimate.

$$\widehat{\theta} = \widehat{\theta}_{u_{i_1},...,u_{i_{k-1}}} = \arg\max_{\theta}(L(\theta \mid u_{i_1},...,u_{i_{k-1}}))$$

This method is not optimal because maybe the maximizer $\widehat{\theta}$ is not found out and the essence of how to determine $\widehat{\theta}$ comes back the hazard of solving the expectation maximization (EM) problem. It is not asserted that there is always solution to such problem. In general, ability estimation is the most important task including three stages:

- At the beginning of the item-selection procedure, we need to specify the prior distribution $g(\theta)$, for example, Gaussian density function.
- During the test, we determine the entropy of each item and compute the ability estimate by using the posterior distribution (Bayesian rule).
- At the end of the test, the final estimate is determined reflecting the examinee's mastery over the test.

IV.1.2.2.3. Suggested stopping criterion in CAT algorithm

In normal, the stopping criterion in CAT algorithm is often the number of (test) items, for example, if the test has ten items then the examinee's final estimate is specified at *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 the 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 standard deviation σ is used to assess the saturation of learner's knowledge. The standard deviation is the root of the variance of the posterior distribution of θ.

$$\mathrm{var}(\theta) = E(\theta - E(\theta \mid u_{i_1},...,u_{i_{k-1}}))^2 = \int (\theta - E(\theta \mid u_{i_1},...,u_{i_{k-1}}))^2 g(\theta \mid u_{i_1},...,u_{i_{k-1}}) d\theta$$

$$\sigma(\theta) = \sqrt{\mathrm{var}(\theta)}$$

Given threshold ξ, if the standard deviation σ is less than ξ then the CAT algorithm stops. There is no restriction for the number of (question) items in exercise.

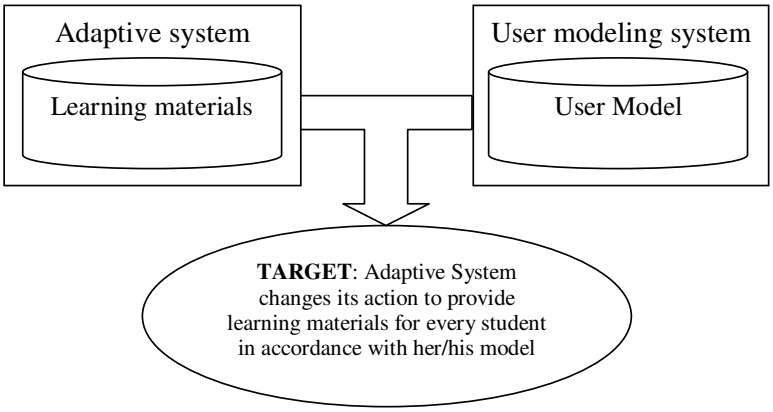
We recognized that CAT gives us the excellent tools for assessing student's ability. The CAT algorithm includes four steps in which step 3 is the most important when student's ability estimate is determined. There is an advantage of Bayesian approach when the ability estimate which is the expectation of posterior probability always exists and is easy to compute. However, the quality of posterior probability depends on the prior probability which may be pre-defined by experts. In the future trend, I intend to find out the technique for learning training data so as to specify precisely prior probabilities.

Moreover I propose the stopping criterion for CAT algorithm in which given threshold ξ, if the standard deviation of student's ability estimate is less than ξ then the CAT algorithm stops. 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.

IV.2. Evaluation of adaptive learning model

As aforementioned in section I.1, 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 [Brusilovsky, Millán 2007] such as knowledge, goal, experience, interest, background, etc when these characteristics vary from person to person. The description of learners' individual information/characteristics

is learner model or user model. Adaptive learning system takes advantages of learner model to improve the quality of adaptation task but it does not build up or manipulate learner model. User modeling system is responsibility for gathering information to create and update learner model. In other words, user modeling system manages user model and provide necessary information about user to adaptive system. Following figure describes the 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]. User modeling system is the heart of adaptive learning system. There are a lot of theories and practical methods including methodologies and approaches in this research to build up adaptive system and user modeling system. Each method has strong points and drawbacks and so it is very useful to evaluate these methods in order to determine which model is appropriate to which situation because each method tailors to concrete conditions and contexts. For example, studying via internet website is very different from studying at a course with support of network. This section 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. We can consider the corporation between adaptive system and user modeling system as an integrated model so-called adaptive learning model. Thus, this section has two goals:

- Firstly, research proposes criterions to evaluate adaptive learning model.
- Secondly, research gives some scenarios as an example that applies criterions above into performing evaluation task in concrete situations.

In other words, this section gives criterions to evaluate TLM, Zebra and WOW. Note that all concepts relating to term "learning" in this research refers to learning via internet or distance learning or e-learning if there is no additional explanation.

IV.2.1. Evaluation criterions

This research proposes three criterions of evaluation:

- Criterion α so-called system criterion tells us how adaptive learning system works with/without user modeling system. For example, when modeling server applies Bayesian network into build up learner model, criterion 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 so-called academic criterion tells us how well modeling server help users to study. This criterion surveys users' study result. The higher criterion β is, the better study result is.
- Criterion γ so-called adaptation criterion or satisfaction criterion measures the quality of adaptation function of learning system with the support of modeling server. After every student gives feedbacks or comments on adaptive system, these feedbacks are collected and analyzed; hence, criterion 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 criterions. Note that in this research, the default user model is Bayesian network model if there is no additional explanation.

IV.2.1.1. Calculating criterion α

There are two ways to calculate 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 $K_U = \{k_1^U, k_2^U, \ldots, k_n^U\}$ 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:
$K_i = \{k_i^U, k_i^U, \ldots, k_i^U\}$ has sample variance s_i^2.
$K_j = \{k_j^U, k_j^U, \ldots, k_j^U\}$ has sample variance s_j^2.
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 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 is used to test two variances.

$$F = \frac{s_i^2}{s_j^2}$$

If $F < f_{0.95, n-1, n-1}$ then the null hypothesis is rejected, we can include that group *A* with support of user modeling system improve adaptive learning system much more than group *B*. So, criterion *α* get Boolean value *true*, 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, 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, etc are represented as a set of random variable are $X_1, X_2, X_3, \ldots, X_n$. Let *Y* represents the total knowledge that users gain over whole course like Java course, Oracle course, etc. We try to find out the linear function of random variables X_i (s) so that *Y* is the expected value of such function.
$Y = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + \ldots + a_nX_n$

Let a_i (s) be regression coefficients. Therefore linear function is determined, it is applied back to each user in both group *A* and *B* so as to predict her/his knowledge so-called *estimated knowledge*. Such knowledge is compared to *real knowledge* of users. The deviation of *estimated knowledge* and *real knowledge* is called *prediction error*. The square sum of all *prediction error* reflects the measure 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 mean least square. Note that because we have two linear functions for group *A* and *B*, there are two sets of regression coefficients, each set for one group.
2. Secondly, let ek_i^A and ek_i^B be estimated knowledge of user i^{th} in group *A* and *B*, respectively. Note that ek_i^A and ek_i^B are calculated by applying linear function determined in the first step.
$ek_i^A = Y^A = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + \ldots + a_nX_n$ (in group A)
$ek_i^B = Y^B = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + \ldots + a_nX_n$ (in group B)
3. Thirdly, let k_i^A and k_i^B be the real knowledge of user i^{th} in group *A* and *B* from database, respectively. Let err_i^A and err_i^B be the prediction error of user i^{th} in group *A* and *B*, respectively.
$err_i^A = | ek_i^A - k_i^A |$
$err_i^B = | ek_i^B - k_i^B |$
4. Finally, the measure α is simple inverse of square sum of all prediction errors.

$$\alpha_A = \frac{1}{\sum_{i \in A} (err_i^A)^2}$$

$$\alpha_B = \frac{1}{\sum_{i \in B} (err_i^B)^2}$$

The larger the measure α is, the better the level of precision of inference method is.

IV.2.1.2. Calculating 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 is the average grades of students in group *A* and *B* over concept *i*, respectively.
$K_A = (k_1^A, k_2^A, \ldots, k_n^A)$ has sample variance s_A^2 and sample mean $\bar{A}$.
$K_B = (k_1^B, k_2^B, \ldots, k_n^B)$ has sample variance s_B^2 and sample mean $\bar{B}$.
The measure β for each group is computed as accumulative probability of assumption user in such group has mastered over course.

$$\beta_A = 1 - \Phi\left(\frac{0.5 - \bar{A}}{s_A/\sqrt{n}}\right)$$

$$\beta_B = 1 - \Phi\left(\frac{0.5 - \bar{B}}{s_B/\sqrt{n}}\right)$$

Let Φ is cumulative function for standard normal distribution. Note that Φ should be accumulative function for *t*-distribution with $n - 1$ degree of freedom for more accurate. Here we use standard normal distribution as example for convenience. The higher criterion β is, the better study result is because the academy criterion α is measured as the probability of event that student's grade is higher than or equal to *0.5*.

IV.2.1.3. Calculating criterion γ

Suppose a questionnaire is built up by expert and it is composed of *n* questions $Q = (q_1, q_2, \ldots, q_n)$. Users in each group rate on each question where rating value may be binary satisfied and unsatisfied. By the simplest way, criterion γ is defined as the ratio of the number of satisfied users to the whole number of users.

$$\beta = \frac{The\ number\ of\ satisfied\ users}{The\ whole\ number\ of\ users}$$

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.
$\gamma_A = |\mu_A|$
$\gamma_B = |\mu_B|$
There are three steps to compress rating matrix and to calculate γ:
1. Firstly, rating matrix is "shrunk" by projecting it onto its eigenvectors. The number of columns is $k << n$. These columns represent essential questions.

Table IV.2.1. Rating matrix as collection of users' feedbacks

a_{11}	…	a_{1j}	…	a_{1n}
…	…	…	…	…
a_{i1}	…	a_{ij}	…	a_{in}
…	…	…	….	…
a_{m1}	…	a_{mj}	…	a_{mn}

Table IV.2.2. Rating matrix is shrunk by projecting it onto its eigenvectors

a_{11}	…	a_{1k}
…	…	…
a_{i1}	…	a_{ik}
…	…	…
a_{m1}	…	a_{mk}

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.

$$\mu = (\mu_1, \mu_2, \ldots, \mu_k)$$

$$\gamma = |\mu| = \sqrt{\mu_1^2 + \mu_2^2 + \cdots + \mu_k^2}$$

In general, the higher criterion *γ* is, the better quality of adaptation is.

IV.2.2. An evaluation scenario

Evaluation scenario is the example for demonstrating how to calculate and apply aforementioned criterions into evaluating the quality of adaptive learning model. E-learning cannot replace face-to-face teaching and it should exist parallel and support traditional education. Thus, this scenario makes the comparison between face-to-face learning manner and distance learning manner. This evaluation scenario is divided into three main acts in which students and teacher play the roles of actors.

1. *Study act*: Teacher teaches and students learn in both face-to-face manner and e-learning manner via website. Suppose students are classified into three groups *A, B* and *C*. Group *A* and *B* represent face-to-face manner and e-learning manner via website, respectively. Especially, group *C* represents e-learning manner with support of user model, namely Bayesian network.
2. *Feedback act*: Students give feedbacks to teacher and teacher collects and analyzes them.
3. *Evaluation act* is done by teacher; thus, criterions *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 criterions.

Study act has *5* scenes:

1. Teacher builds up school's curriculums and set up adaptive e-learning website with/without the support of user modeling system.
2. Teacher teaches and students in group *A, B* and *C* learn by face-to-face manner.
3. Students in group *B* and *C* go on website and study by themselves. Teacher monitors them and put up important notice.
4. Students in group *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.
- 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 group *A, B* and *C* go to class to study in face-to-face manner.
- Students in group *B* and *C* learn themselves on adaptive learning web sites. Note that website / learning materials are adapted to each student based on their knowledge and characteristics.
- Students in group A, B and C do tests / exercise via website.

Feedback act has *3* scenes:

1. Teacher creates the questionnaire to survey students' feeling about both adaptive learning website and curriculum such as very satisfied, satisfied and not satisfied.
2. Students answer or rate on such questions online.
3. Teacher collects students' feedbacks and analyzes them.

Evaluation act has *2* scenes:
1. Teacher calculates three criterions based on students' feedback and test results.
2. Teacher makes the decision about the quality of face-to-face teaching manner and e-learning manner with/without support of user modeling system.

For example, there are two classes *A* and *B*. Class *A* is only taught in face-to-face manner, otherwise students in class *B* study both in face-to-face manner and adaptive learning environment. Study results and students feedback are collected from both two classes. Each class has the same number of students, namely *10*. Let G_A and G_B be the average study results of classes *A* and *B*, respectively.
$G_A = \{4, 5, 3, 6, 2, 8, 3, 5, 8, 6\}$
$G_B = \{6, 8, 9, 10, 6, 7, 9, 6, 9\}$
Suppose there are *4* students in class *A* and *2* students in class *B* who don't satisfy teaching curriculum. Sample mean and deviation of class *A* are $\bar{G}_A$=5.0 and $s_A = \sqrt{38/9} = 2.05$. Sample mean and deviation of class *B* are $\bar{G}_A$= 7.0 and $s_A = \sqrt{38/9} = 1.66$. Let α_A and α_B be academy criterions of class *A* and class *B*, respectively.

$$\alpha_A = 1 - \Phi\left(\frac{5.0 - 5.0}{\frac{2.05}{\sqrt{10}}}\right) = 1 - \Phi(0) = 0.5$$

$$\alpha_B = 1 - \Phi\left(\frac{5.0 - 7.0}{\frac{1.66}{\sqrt{10}}}\right) = 1 - \Phi(3.8) = 0.99$$

Let β_A and β_B be satisfaction criterions of class *A* and class *B*, respectively, we have:

$$\beta_A = \frac{10 - 4}{10} = 0.6$$

$$\beta_B = \frac{10 - 2}{10} = 0.8$$

Suppose the weights of criterion α and criterion β are *0.7* and *0.3*, respectively. Let $eval_A$ and $eval_B$ be the total evaluations on group A and group B, respectively.
$eval_A = 0.7*0.5 + 0.3*0.6 = 0.53$
$eval_B = 0.7*0.99 + 0.3*0.8 = 0.93$
When $eval_A$ is greater than $eval_B$, it is possible to conclude that the teaching method in class *B* is more effective than the one in class *A* because class *B* takes advantages of adaptive learning method.
This section is finished with some comments on evaluation criterions. As aforementioned, there are three criterions such as system criterion α, academy criterion β and adaptation criterion γ. That two of three criterions, 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 criterions, indicates that study is lifelong process for everyone and so, classes and courses are short movies in this lifelong process. Both students and teachers are actors and their roles can mutually interchange, for example, teaching is the best way to learn and student is the best teacher of teacher. This section ends up the main contents of this research and gives you the comprehensive and detailed description about thesis and so the next section is the conclusion and future trend.

CONCLUSION AND FUTURE TREND

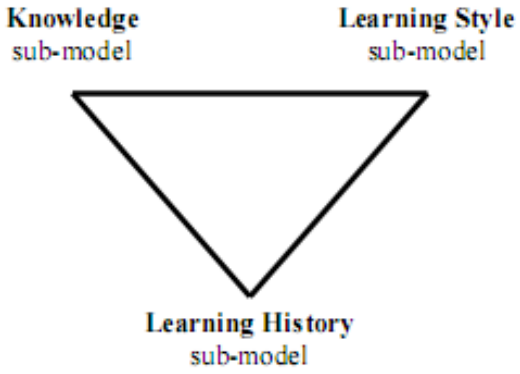
Conclusion

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 higher is the standard of living, the more important are adaptive IT systems which have ability of change their behaviors so as to be in accordance users' characteristics. Note that this thesis focuses on e-learning and so the term "user" implicates "learner" or "student" in learning context. The representation of such characteristics is called user model but there is too much information about individuals to model all users' characteristics; so it is necessary to choose essential characteristics from which a solid architecture of user model is built. Each modeling method is only appropriate to respective characteristic like knowledge or learning style or goal, etc. There is no modeling method fit all characteristics. On the other hand, some domain-independent user modeling systems are too generic to "cover" all learners' characteristics in e-learning context, which may cause unpredictable bad consequences in adaptation process. Moreover user modeling systems require effective inference techniques in their modeling tasks but this is impossible if we can't recognize which individual characteristics are important.
To overcome these obstacles, I propose the new learner model that contains three most important characteristics of user: knowledge (**K**), learning styles (**LS**) and learning history (**LH**). Such three characteristics form a triangle; so our model is called Triangular Learner Model (TLM). **TLM** with three underlying characteristics will cover the whole of user's information required by learning adaptation process. The reasons for such assertion are:

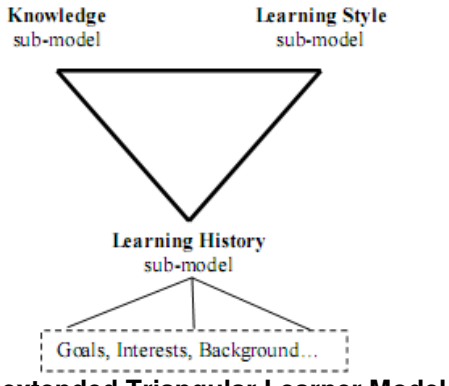
- Knowledge, learning styles and learning history are prerequisite for modeling learner.
- While learning history and knowledge change themselves frequently, learning styles are relatively stable. The combination of them ensures the integrity of information about learner.
- User knowledge is domain specific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

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

So TLM is constituted of three sub-models: *Knowledge, Learning Style* and *Learning History*.



Triangular Learner Model (TLM)



extended Triangular Learner Model

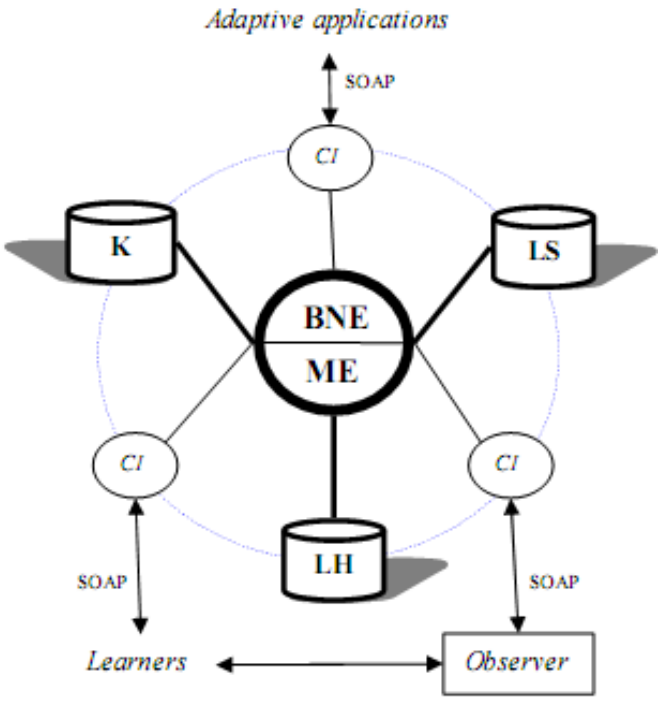
The TLM can be extended to interpret more detailed about learner by attaching more learners' characteristics such as interests, background, goals… into the sub-model *Learning History*.
I also introduce the architecture of the user modeling system so-called Zebra that realizes the TLM. The core of Zebra is the composition of two engines: *mining engine* (**ME**) and *belief network engine* (**BNE**).

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

network engine. Mining engine almost always uses mining techniques. It has three other important functionalities that are to discover some other characteristics (beyond knowledge and learning styles) such as interests, goals, learning context, etc and to support learning concept recommendation and to support collaborative learning.

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

Zebra provides *communication interfaces* (**CI**) that allows users and adaptive systems to see or modify restrictedly TLM. Adaptive applications also interact with Zebra by these interfaces.



The architecture of Zebra

This research is fundamental research, thus, the methodology to build up TLM is specified and proven via mathematic tools. The feasibility of TLM and architecture is authenticated via the user modeling software names Zebra associated with TLM. Moreover, the adaptive learning system that interacts with Zebra is also implemented as e-learning web-based software. Another strong point of TLM is ability of extension, other user information such as user interests and goals can be discovered or extracted from TLM. Researchers can use the methodology, models and mathematical formulas in this research to build up their own user model and user modeling system. They can also develop TLM and Zebra by extending advanced functions such as discovering user goals, context-aware adaptation, ubiquitous modeling, mobile service, etc. Next section discussing the future trend is an example of TLM extension. This research contributes to user modeling and adaptive learning community a methodology for constructing user model and a whole perfect learner model.

The limitation of this research is lack of privacy in learner model although external applications only access TLM via communication interface (see 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 and inference speed is reduced.

V.2. Towards ubiquitous user modeling

Nowadays there is a need for users to interact with many IT systems at anywhere, for examples, users withdraw cash by ATM card or book airplane ticket online or play games on mobile phones. 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.
Ubiquitous user modeling [Heckmann, 2005] is considered as the intersection of three domains: user modeling, ubiquitous computing and semantic web.

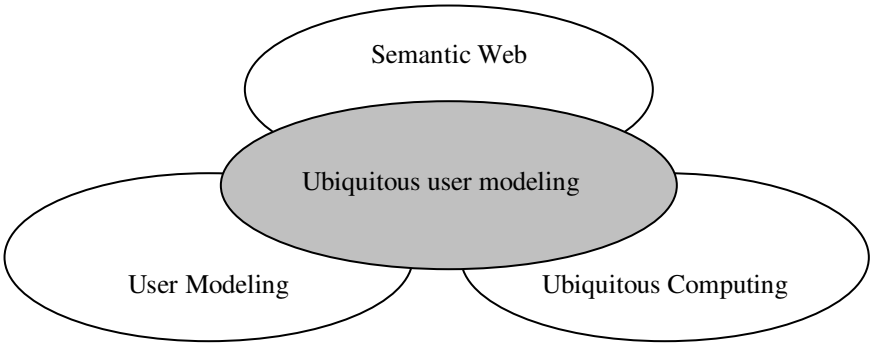


Figure V.2.1. Ubiquitous user modeling

V.2.1. Ubiquitous User Model Service

Ubiquitous user model service is responsible for manipulating ubiquitous user model. It is essentially a ubiquitous user modeling system. According to [Heckmann, 2005], 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.

V.2.1.1. Architecture of ubiquitous user model service

The architecture of ubiquitous user model service [Heckmann 2005] consists of three boxes: *Distributed Services box*, *Application box*, *Distributed Statements box*, *Distributed Ontologies box*, *Interfaces and Exchange box*

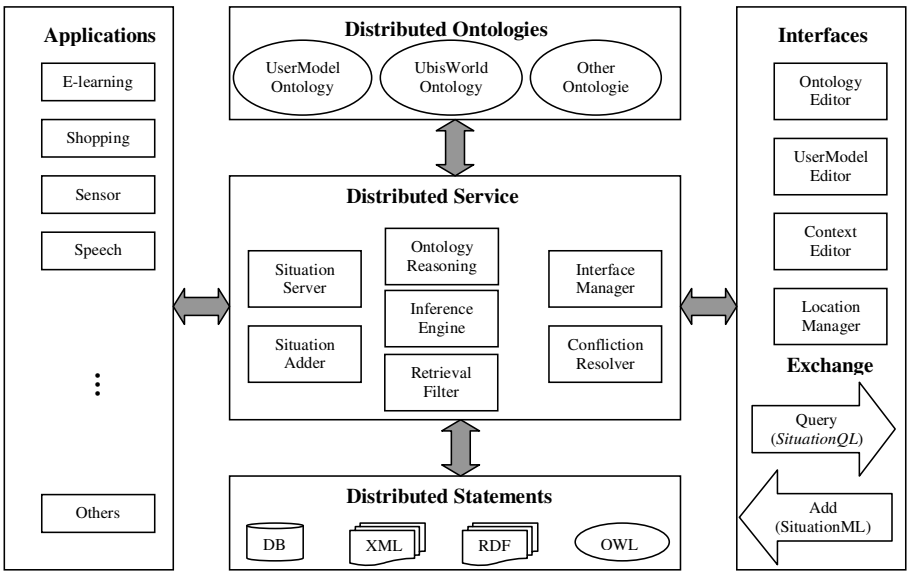


Figure V.2.2. Architecture of ubiquitous user model service

Distributed Services box

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.

- 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

Application box lists applications that possibly cooperate with distributed services such as e-learning, sensors, airport service, mobile phone….

Distributed Statements box

Distributed Statements box 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.

Distributed Ontologies box

Distributed ontologies box 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 and Exchange box

Interfaces & Exchange box 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. 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.

V.2.1.2. 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:

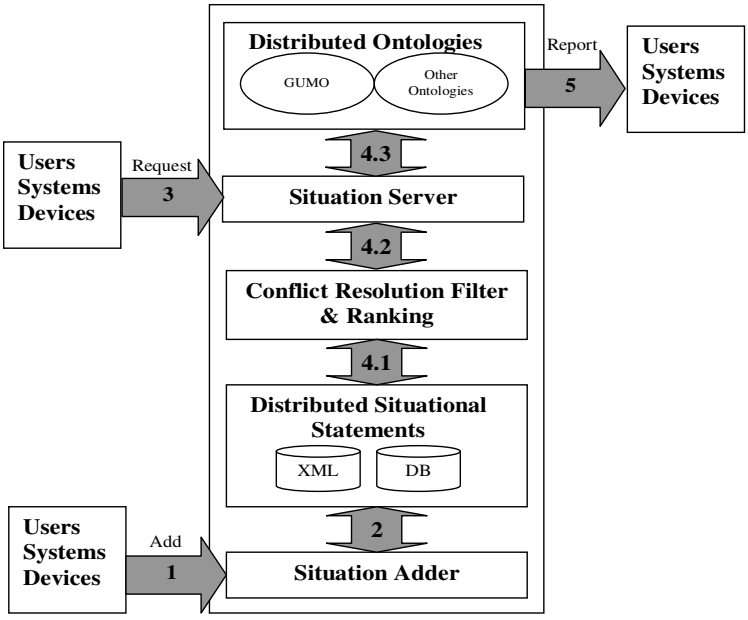


Figure V.2.3. The work flow of ubiquitous user model service

The work flow includes below steps.

- 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)**.

V.2.2. 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* & *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. 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, sequential pattern, etc are very different from statement repositories in ubiquitous service.
- *Secondly*, almost techniques in Zebra like data mining, artificial intelligence, learning machine 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, new personal recommendations… is represented in the form which is incompatible with knowledge representation of situation model in ubiquitous service such as ontologies, RDF, etc.

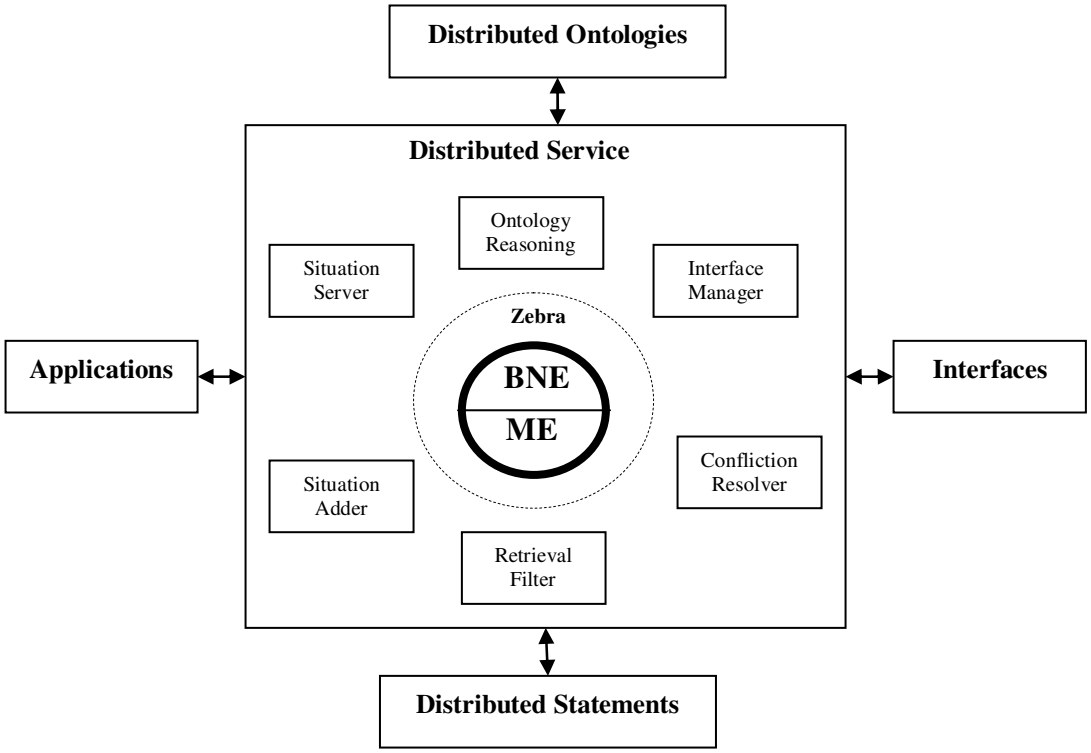


Figure V.2.4. Incorporating Zebra into ubiquitous service

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 four 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* & *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 in form of *SituationML* which is readily understandable for ubiquitous service and sends it to ubiquitous service.

(6)

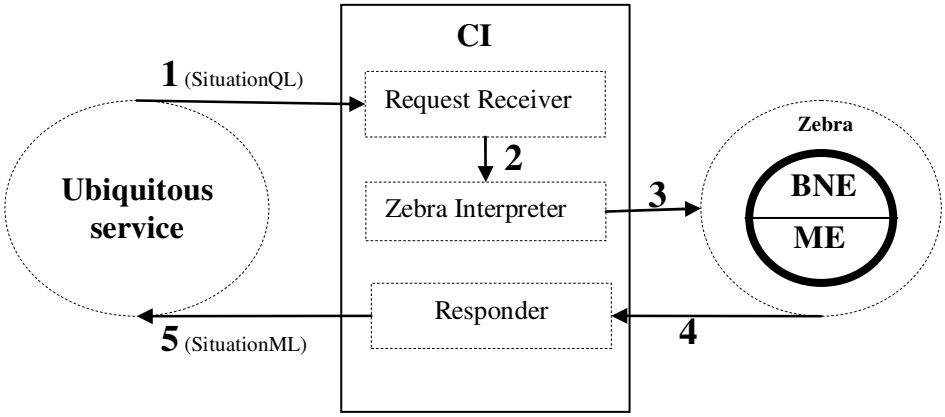


Figure V.2.5. Request-Response communication protocol between Zebra and ubiquitous service

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. Ubiquitous service interacts with the Zebra in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

ACKNOWLEDGEMENT

I would like to show my gratitude to all those who gave me the possibility to complete this thesis. I am grateful to Pro. Dr. Dong Thi Bich Thuy for providing invaluable help and encouragement in my work. If it were not for your help, I would not complete my thesis in time. I have furthermore to thank the Department of Information System – Faculty of Information Technology – University of Science for conveniences during the time of research.
I also want to express my warmest thanks for my family and friends who have supported me in difficult times

REFERENCE

Agrawal R, Srikant R(1994). Fast algorithms for mining association rules. In Proc. 1994 Int. Conf. Very Large DataBases (VLDB94), Pp. 487-499, Santiago, Chile, Sept. 1994.

Agrawal R, Srikant R(1995). Mining sequential patterns. In Proc. 1995 Int. Conf. Data Engineering (ICDE95), Pp.314, Taipei, Taiwan, Mar.1995.

Ahmed K Elmagarmid, Amit P Sheth(2008). Privacy Preserving Data Mining: Models and Algorithms. Advances in Database Systems, Volume34, Series Editors. Springer | 2008-06-20 | ISBN: 0387709916 | 535 pages.

Alexandre Evfimievski(2002). Randomization in privacy preserving data mining. ACM SIGKDD Explorations Newsletter 2002.

Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, Johannes Gehrke(2002). Privacy Preserving Mining of Association Rules. SIGKDD 2002 Edmonton, Alberta, Canada.

Alfred Kobsa (1991). First experiences with the SB-ONE knowledge representation workbench in natural-language applications. ACM SIGART Bulletin. 2(3):70-76.

Alfred Kobsa(1990). User Modeling in Dialog Systems: Potentials and Hazards. AI and Society , 4 (Jul./Sep. 1990), Pp. 214 - 231.

Alfred Kobsa(1993). User Modeling: Recent Work, Prospects and Hazards. Department of Computer Science, Columbia University, New York, USA.

Alfred Kobsa, Josef Fink(2006). An LDAP-Based User Modeling Server and its Evaluation. User Modeling and User-Adapted Interaction 2006 (UMUAI-2006).

Alfred Kobsa, Wolfgang Pohl(1995). The User Modeling Shell System BGP-MS. User Modeling and User-Adapted Interaction 4(2):59-106. (UMUAI-1995).

Alfred Kobsa. Generic User Modeling Systems. User Modeling and User-Adapted Interaction 2006 (UMUAI-2006).

Anders Hald(1999). On the History of Maximum Likelihood in Relation to Inverse Probability and Least Squares. Statistical Science 14 (2), May 1999

Andreas Jedlitschka, Klaus-Dieter Althoff(2001). Using Case-Based Reasoning for User Modeling in an Experience Management System. "Proc. Workshop ""Adaptivität und Benutzermodellierung in Interaktiven Systemen (ABIS'01)"", GI-Workshop-Woche ""Lernen - Lehren - Wissen - Adaptivität (LLWA'01)"", Universität Dortmund, 8.-12. Okt. 2001".

Anthony Jameson(1995). Logic is not enough: Why reasoning about another person's beliefs is reasoning under uncertainty. A. Laux and H. Wansing (eds.): Knowledge and Belief in Philosophy and Artificial Intelligence. Berlin: Akademie Verlag. 1995.

Anthony Jameson, Wolfang Hoeppner, Wolfgang Wahlster(1980). The Natural Language System HAM-RPM as a Hotel Manager: Some Representational Prerequisites. R. Wilhelm (ed.): GI - 10. Jahrestagung Saarbrücken. Berlin: Springer. 1980.

Bock RD, Mislevy RJ(1988). Adaptive EAP estimation of ability in a microcomputer environment. Applied Psychological Measurement, 6: 431-444.

Brajnik G, Tasso C(1994). A Shell for Developing Non-monotonic User Modeling Systems. International J. Human-Computer Studies 40: 31-62. DOI: 10.1006/ijhc.1994.1003

Carla Lane(2000). Implementing Multiple Intelligences and Learning Styles in Distributed Learning/IMS Projects. Technical report, The Education Coalition (TEC). http://www.tecweb.org/styles/imslsindl.pdf.

Christian Wolf(2002). iWeaver: Towards an Interactive Web-Based Adaptive Learning Environment to Address Individual Learning Styles. The Interactive Computer Aided Learning Workshop (ICL2002), Villach, Austria.

Christian Wolf(2003). iWeaver: Towards "Learning Style"-based e-Learning in Computer Science Education. Australasian Computing Education Conference (ACE2003), Adeleida, Australia. Conferences in Research and Practice in Information Technology, Vol.20.

Christian Wolf(2007). Construction of an Adaptive E-learning Environment to Address Learning Styles and an Investigation of the Effect of Media Choice. PhD by Project: RMIT University - Design and Social Context Portfolio, School of Education (Research) - Melbourne - Australia (January 2007).

Christoph Fröschl(2005). User Modeling and User Profiling in Adaptive E-learning Systems. Masters Thesis at Graz University of Technology 2005. An approach for a service-based personalization solution for the research project AdeLE (Adaptive e-Learning with Eye-Tracking).

Christos Papatheodorou(2001). Machine Learning in User Modeling. In G. Paliouras, V. Karkaletsis, C. Spyropoulos eds., "Machine Learning and Applications", Lecture Notes in Artificial Intelligence (LNAI), No 2049: Springer-Verlag. Pp. 286-294.

Cristian Berg(2004). Integral Representation of Some Functions Related to the Gamma Function. Mediterranean Journal of Mathematics, 433–439, 1660-5446/040433-7, DOI 10.1007/s00009-004-0022-6. Birkhäuser Verlag Basel/Switzerland 2004

Cristianini N, Shawe-Taylor J (2000). An Introduction to Support Vector Machines. Cambridge University Press, 2000.

Cristina Conati, Abigail Gertner and Kurt Vanlehn(2002). Using Bayesian Networks to Manage Uncertainty in Student Modeling J.User Modeling and User-Adapted Interaction. 12(4): 371 - 417. ISSN:0924-1868

Cristina Conati, Abigail S. Gertner, Kurt VanLehn, Marek J. Druzdzel(1997). On-line student modelling for coached problem solving using Bayesian Networks. Proceedings of the 6th International Conference on User Modelling UM'97. Wien, New York: Springer Verlag.

Dakshi Agrawal, Charu C. Aggarwal(2001). On the design and quantification of privacy preserving data mining algorithms. In Proceedings of the 20th ACM Symposium on Principles of Database Systems (2001), 247-255.
Daniel Billsus, Michael J(2000). Pazzani. User Modeling for Adaptive News Access. User Modeling and User-Adapted Interaction 10: 147-180, 2000. © 2000 Kluwer Academic Publishers. Printed in the Netherlands.
David Hauger, Mirjam Köck(2007). State of the Art of Adaptivity in E-Learning Platforms. Proceedings of the 15th Workshop on Adaptivity and User Modeling in Interactive Systems (ABIS 2007). ISBN: 978-3-86010-907-6 (2007).
David Heckerman(1996). A Tutorial on Learning With Bayesian Networks. Technical Report MSR-TR-95-06. Microsoft Research Advanced Technology Division, Microsoft Corporation.
David N Chin(1983). A Case Study of Knowledge Representation in UC. Proceedings of the Eighth International Joint Conference on Artificial Intelligence. 1: 388-390. Karlsruhe, West Germany, August 1983.
David N Chin(1989). KNOME: Modeling What the User Knows in UC. A. Kobsa and W. Wahlster (eds.): User Models in Dialog Systems. Berlin, Heidelberg: Springer. Pp. 74-107.
David N Chin(1993). Acquiring user models. Artificial Intelligence Review 7:185-197, 1993. © 1993 Kluwer Academic Publishers. Printed in the Netherlands.
David Poole(1987). A Logical Framework for Default Reasoning. Logic Programming and Artificial Intelligence Group, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L3G1, (519)888-4443.
De Bra P, Houben GJ, Wu H(1999). AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In: Proceedings of 10th ACM Conference on Hypertext and hypermedia (Hypertext'99), Darmstadt, Germany, February 21 - 25, 1999, ACM Press, Pp. 147-156.
De Bra, David Smits, Natalia Stash(2006). The Design of AHA!. Proceedings of the ACM Hypertext Conference, Odense, Denmark, August 23-25, 2006. Pp. 133.
De Bra, Stash Smits(2005). Creating Adaptive Web-Based Applications. Tutorial at the 10th International Conference on User Modeling, Edinburgh, Scotland, July 25, 2005.
Demetris E Kyriacou(2005). Life-long User Modeling. A progress report submitted for continuation towards a PhD thesis. School of Electronics and Computer Science; Faculty of Engineering, Sciences and Mathematics; University of Southampton; Learning Societies Lab.
Dempster AP, Laird NM, Rubin DB(1977). Maximum likelihood from incomplete data via the em algorithm. J. tRoyal stat. Society, Series B, 39:1-38.
Dominikus Heckmann(2005). Ubiquitous User Modeling. PhD Thesis at Universität des Saarlandes. Volume 297 Dissertationen zur Künstlichen Intelligenz.
Dugad R, Desai UB(1996). A tutorial on Hidden Markov models. Signal Processing and Artificial Neural Networks Laboratory, Dept of Electrical Engineering, Indian Institute of Technology, Bombay Technical Report No.: SPANN-96.1.
Elaine Rich(1979). User Modeling via Stereotypes. COGNITIVE SCIENCE 3:329-354
Elaine Rich(1983). Users are individuals: individualizing user models. Int. J. Man-Machine Studies. 18:199-214.
Eric Horvitz, Jack Breese, David Heckerman, David Hovel, Koos Rommelse(1998). The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, July 1998, Pp. 256-265. Morgan Kaufmann: San Francisco.
Eric J Horvitz, John Breese, Max Henrion(1988). Decision Theory in Expert Systems and Artificial Intelligence. International J. Approximate Reasoning 2:247-302
Eva Millán, Jose Luis Pérez-de-la-Cruz (2002). A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation. User Modeling and User-Adapted Interaction 12: 281-330, 2002. © 2002 Kluwer Academic Publisher. Printed in the Netherlands.
Felder RM, Silverman LK(1988). Learning and Teaching Styles in Engineering Education. J. Eng. Edu. Provide paper numbers.
Felix Mödritscher, Victor Manuel Garcia-Barrios, Christian Gütl(2004). The Past, the Present and the future of adaptive E-Learning. In Proceedings of the International Conference Interactive Computer Aided Learning (ICL2004), 2004. http://www.iicm.tugraz.at/www/Research/Publications/2004/_id188a8_/adaptive_e-learning/adaptiv_e-learning.pdf.
Flavia Sparacino(2003). Sto(ry)chastics: a Bayesian Network Architecture for User Modeling and Computational Storytelling for Interactive Spaces. Lecture Notes in Computer Science.
Frank B. Baker(2001). The basics of item response theory. Published by the ERIC Clearinghouse on Assessment Evaluation 2001
Frank Dellaert(2002). The Expectation Maximization Algorithm. College of Computing, Georgia Instituteof Technology. Technical Report number GIT-GVU-02-20, February 2002.
Frank G Halasz, Mayer Schwartz(1990). The Dexter Hypertext Reference Model. Presented at the NIST Hypertext Standardization Workshop, Gaithersburg, MD, January 16-18, 1990
Frank Wittig(1999). Learning Bayesian Networks With Hidden Variables for User Modeling. 7-th International Conference on User Modeling, June 20-24, 1999, Banff Centre, Banff, Canada.
Gerhard Fischer(2000). User Modeling in Human-Computer Interaction. "Contribution to the 10th Anniversary Issue of the Journal ""User Modeling and User-Adapted Interaction (UMUAI).
Han J, Pei J, Yin Y(2000). Mining frequent patterns without candidate generation. In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD00), Pp. 112, Dallas, TX, May 2000.
Ian H Beaumont(1994). User modeling in the interactive anatomy tutoring system ANATOM-TUTOR. In User Models and User Adapted Interaction, 4(1):21–45.
Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, Mei-Chun Hsu(2000). FreeSpan: frequent pattern-projected sequential pattern mining. KDD '00 Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Pp. 355-359, ACM New York, NY, USA ©2000, ISBN:1-58113-233-6, doi>10.1145/347090.347167.
Jiawei Han, Michelline Kamber(2006). Data Mining: Concepts and Techniques. Second Edition. © 2006 by Elsevier Inc.
Jie Cheng, David A Bell, Weiru Liu(1997a). An Algorithm for Bayesian Belief Network Construction from Data. Proceedings of the 6th International Workshop on AI and Statistics (AI and Stat. '97).
Jie Cheng, David A Bell, Weiru Liu(1997b). Learning Belief Networks from Data: An information theory based approach. Proceedings of Sixth International Conference on Information and Knowledge Management (CIKM) :325-331, ACM Press. November 1997, USA.
Jie Cheng, David A Bell, Weiru Liu(1998). Learning Bayesian Networks from Data: An Efficient Approach Based on Information Theory. Technical report, University of Alberta, Canada 1998.
Jonathan L Orwant(1991). Doppelgänger: A User Modeling System. Bachelor thesis. Masschusetts institute of Technology June 1991.
Jonathan L Orwant(1993). Doppelgänger Goes To School: Machine Learning for User Modeling. Master's thesis, Massachusetts Institute of Technology, 1993.
Jonathan L Orwant(1995). Heterogeneous Learning in the Doppelgänger User Modeling System. User Modeling and User-Adapted Interaction 4

(3):197-226. UMUA 1995.
Jörg Schreck(2003). Security and Privacy in User Modeling. Dordrecht: Kluwer Academic Publishers. Fachbereich Mathematik und Informatik der Universität – Gesamthochschule – Essen
Josef Fink Gutachter, Alfred Kobsa, Rainer Unland(2004). User Modeling Servers - Requirements, Design, and Evaluation. Universität Duisburg-Essen, Standort Essen, Fachbereich 6 Mathematik. Amsterdam, Netherlands: IOS Press.
Josef Fink, Alfred Kobsa(2002). User Modeling for Personalized City Tours. Artificial Intelligence Review 18, 2002, Pp. 33-74, Kluwer Academic Publishers.
Joseph Y Halpern. Reasoning about Knowledge. http://www.cs.cornell.edu/home/halpern/topics.html#rak
Joseph Y Halpern. Reasoning about Uncertainty. http://www.cs.cornell.edu/home/halpern/topics.html#rau
Judea Pearl (1988). Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann Publish, San Mateo, California, 1988. ISBN-13: 978-1558604797.
Judy Kay(1995). The um Toolkit for Cooperative User Modelling. User Modeling and User-Adapted Interaction, 4(3):149-196, 1995 (UMUA 95).
Judy Kay, Bob Kummerfeld, Piers Lauder(2002). Personis: A server for user models. Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2002), Pp. 201-212.
Julita Vassileva, Gordon I McCalla, Jim Greer(2003). Multi-Agent Multi-User Modeling in I-Help. User Modeling and User-Adapted Interaction 13: 179-210, 2003 (UMUA 03). © 2003 Kluwer Academic Publishers. Printed in the Netherlands.
Kevin P. Murphy(1998). A Brief Introduction to Graphical Models and Bayesian Networks at http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html. Department of Computer Science, University of British Columbia.
Kolb DA(1999). The Kolb Learning Style Inventory. Version 3. Boston: Hay Group.
Loc Nguyen, Phung Do(2009). Combination of Bayesian Network and Overlay Model in User Modeling. Proceedings of 4th International Conference on Interactive Mobile and Computer Aided Learning (IMCL 2009) 22-24 April 2009. Princess Sumaya University for Technology, Amman-Jordan.
Lora Aroyo. User Modelling and Recommender Systems. KBS course lecture (slides from Vania Dimitrova). 2006.
Michael John Mayo(2001). Bayesian Student Modeling and Decision-Theoretic Selection of Tutorial Action in Intelligent Tutoring System. PhD Thesis, University of Centerbury, 2001.
Michael John Mayo, Antonija Mitrovic(2001). Optimising ITS behaviour with Bayesian networks and decision theory. International J. Artificial Intelligence in Edu. 12(2001):124-153.
Michael Mayo, Antonija Mitrovic(2000). Using a Probabilistic Student Model to Control Problem Difficulty. In Gauthier G., Frasson C., and VanLehn K. (Eds.), Proc. of 5th International Conference on Intelligent Tutoring Systems, Springer-Verlag, Pp. 524-533
Miha Grčar, Dunja Mladenić, Marko Grobelnik(2005). User Profiling for Interest-focused Browsing History. Department of Knowledge Technologies, Jozef Stefan Institute. 2005.
Miroslav Bures, Ivan Jelinek(2006). Automatic Generation of User Model from Non-trivial Hypermedia in Adaptive E-learning Hypermedia System. Proceedings of the Fifth IASTED International Conference, WEB-BASED EDUCATION, January 23-25, 2006, Pierto Vallarta, Mexico.
Mitchell T(1997). Machine Learning. McGraw-Hill International, 1997
Mohammad Alrifai, Peter Dolog, Wolfgang Nejdl(2006). Learner Profile Management for Collaborating Adaptive eLearning Applications. APS'2006: Joint International Workshop on Adaptivity, Personalization and the Semantic Web at at the 17th ACM Hypertext'06 conference, Odense, Denmark, August 2006. ACM Press.
Mohammed J Zaki(2000). SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning, 0, 1-31 (2000). Copyright © 2000 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Natalia Stash, Alexandra Cristea, Paul De Bra(2005). Explicit Intelligence in Adaptive Hypermedia: Generic Adaptation Languages for Learning Preferences and Styles. In Proceedings of HT2005 CIAH Workshop, Salzburg, Austria, 2005.

Natalia Stash, Alexandra I. Cristea, Paul De Bra(2007). Adaptation languages as vehicles of explicit intelligence in Adaptive Hypermedia. International J. Cont. Engineering Education and Life-Long Learning 2007.

Nicola Henze(2000). Adaptive Hyperbooks: Adaptation for Project-Based Learning Resources. PhD Thesis at University of Hannover (Supervisors: Prof. Dr. W. Nejdl, Prof. Dr. U. Lipeck).

Nicola Henze, Wolfgang Nejdl(1999). Bayesian Modeling for Adaptive Hypermedia Systems. Knowledge Based Systems Group, University of Hannover, Lange Laube 3, 30159 Hannover, Germany. 1999.

Nicola Henze, Wolfgang Nejdl(1999). Student modeling for KBS Hyperbook system using Bayesian networks. Technical report, University of Hannover (1999).

Nicola Henze, Wolfgang Nejdl(2004). A Logical Characterization of Adaptive Educational Hypermedia. New Review of Hypermedia and Multimedia (NRHM), 10 (1):77-113.

Ok-choon Park, Jung Lee(2003). Handbook of Research for Educational Communications and Technology, chapter Adaptive Instructional Systems, Pp. 651–660. Association for Educational Communications and Technology.

Owen Conlan(2003). State of the Art: Adaptive Hypermedia. M-Zones State of the Art Paper, soa paper 05/03, Ireland. 2003.

Paiva A, Self J(1994ª). A Learner Model Reason Maintenance System. ECAI94. 11th European Conference on Artificial Intelligence Edited by A.Cohn, Published in 1994 by John Wiley and Sons, Ltd.

Paiva A, Self J(1994b). TAGUS: A User and Learner Modeling System. Proceedings of the Fourth International Conference on User Modeling. Hyannis, MA (1994). Pp. 43-49.

Pask G(1976). Styles and Strategies of Learning. British J. Edu. Psychol.

Paul De Bra, Licia Calvi(1998). AHA! An open Adaptive Hypermedia Architecture. The New Review of Hypermedia and Multimedia. 4: 115-139, Taylor Graham Publishers.

Peter Brusilovsky(1994). The Construction and Application of Student Models in Intelligent Tutoring Systems. J. Computer and System Sci. International. 32(1):70-89.

Peter Brusilovsky(1998). Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 6, 1996, 87-129. (Reprinted in Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers. Pp. 1-43

Peter Brusilovsky(2004). KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. Thirteen International World Wide Web Conference, WWW 2004 (Alternate track papers and posters), New York, NY (2004) 104-113, 10.1145/1013367.1013386.

Peter Brusilovsky, Mark T Maybury(2002). From adaptive hypermedia to the adaptive web. Communications of the ACM. 45(5):30–33.

Peter Brusilovsky, Sergey Sosnovsky, Michael Yudelson(2005). Ontology-based Framework for User Model Interoperability in Distributed Learning Environments. World Conference on E-learning, E-learn 2005, Vancouver, Canada (2005). Pp.2851-2855.

Peter Brusilovsky, Sergey Sosnovsky, Olena Shcherbinina(2005). User Modeling in a Distributed E-Learning Architecture. School of Information Sciences, University of Pittsburgh, Pittsburgh PA 15260, USA.

Peter Dolog, Michael Schäfer(2005). Learner Modeling on the Semantic Web. In Proc. of PerSWeb-2005 Workshop: Personalization on the Semantic Web at User Modeling 2005: 10th International Conference, July, 2005, Edinburgh, UK.

Peter Honey, Alan Mumford(1992). The Manual of Learning Styles. Maidenhead: Peter Honey Publications, 1992

Pontus Johansson(2002). User Modeling in Dialog Systems. St. Anna Report: SAR 02-2.

Pythagoras Karampiperis, Demetrios Sampson(2005). Adaptive Learning Resources Sequencing in Educational Hypermedia Systems. Educational Technology and Society, 8 (4):128-147.

Rakesh Agrawal, Ramakrishnan Srikant(2000). Privacy-Preserving Data Mining. In Proceedings of the ACM SIGMOD Conference on Management of Data 2000, Pp. 439-450.

Raul Rojas(1996). Neural Networks - A Systematic Introduction. Springer-Verlag, Berlin, New-York, 1996 (502 p.,350 illustrations). Website: http://page.mi.fu-berlin.de/rojas/neural/index.html.html.

Ricardo Conejo, Eduardo Guzmán, Jose Luis Pérez-de-la-Cruz, Eva Millán(2005). Introducing adaptive assistance in adaptive testing. en: Chen-Kit Looi, Gord McCalla, Bert Bredeweg, Joost Breuker (Eds.), Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology (IOS Press, Amsterdam, pp. 777-779.

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

Riding R, Rayner S(1998). Cognitive Styles and Learning Strategies: Understanding Style Differences in Learning Behaviour. London: David Fulton Publishers Ltd, 1998.

Rita Dunn, Kenneth Dunn(2003). The Dunn and Dunn Learning Style Model and Its Theoretical Cornerstone. St John's University, New York, 2003

Rober B. Allen(1990). User Models: Theory, Method, and Practice. International J. Man-Machine Studies. 32:511-543.

Robert J Mislevy, Drew H Gitomer(1996). The Role of Probability-Based Inference in an Intelligent Tutoring System. User Modeling and User-Adapted Interaction, 5:253-282.

Robert Kass, Tim Finin(1988). Modeling the user in natural language systems. Journal: Computational Linguistics. Date: January 23, 1988.

Robert Wilensky(1987). Some Problems and Proposals for Knowledge Representation. Computer Science Division, University of California, Berkely, Report No. UCB/CSD 87/351.

Robert Wilensky, David N. Chin, Marc Luria, James Martin, James Mayfield, Dekai Wu(1988). The Berkeley Unix Consultant Project. Computational Linguistics. 14: 4.

Robert Wilensky, James Mayfield, Anthony Albert, David N. Chin, Charles Cox, Marc Luria, James Martin, Dekai Wu(1986). UC - A Progress Report. Computer Science Division, University of California, Berkely, Report No. UCB/CSD 89/303.

Roberto Tedesco, Peter Dolog, Wolfgang Nejdl, Heidrun Allert(2006). Distributed Bayesian Networks for User Modeling. ELEARN 2006 : World Conference on E-Learning in Corporate, Government, Health Care, and Higher Education.

Ronald Fagin, Joseph Y Halpern(1994). Reasoning about knowledge and probability. ACM 41, 2, 1994, Pp. 340-367. Preliminary version appeared in Second Conference on Theoretical Aspects of Reasoning about Knowledge, ed. M. Y. Vardi, Morgan Kaufmann, 1988, Pp. 277-293. Corrigendum: J. ACM 45, 1, Jan. 1998, p. 214.

Ronald J Brachman, Hector J Levesque(2003). Knowledge Representation and Reasoning. ©2003 Ronald J. Brachman and Hector J. Levesque. CMPT 411/882 Course Home Page, Spring 2005.

Scott M Lynch(2003). Maximum Likelihood Estimation. Soc. 504, February 2003

Shariq J Rizvi, Jayant R Haritsa(2002). Maintaining data privacy in association rule mining. In Proceedings of the 28th VLDB Conference, Hong Kong

, China, August 2002.
Srikant R, Agrawal R(1996). Mining sequential patterns: Generalizations and performance improvements. In Proc. 5th Int. Conf. Extending Database Technology (EDBT96). Pp. 317, Avignon, France, Mar. 1996
Tewodros Amdeberhan, Olivier Espinosa and Victor H. Moll (provide year of publication of article). The Laplace Transform of The Digamma Function: An Integral due to Glasser, Manna and Oloa
Tim Finin, David Drager(1986). GUMS: A General User Modeling System. Proceedings of the Canadian Society for Computational Studies of Intelligence 1986 (CSCSI-86).
Tomáš Kubeš (2007). Application of Hypermedia Systems in e-Learning. Master Thesis in the Department of Computers, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic in the May 2007.
Tomáš Kubeš. Overview of Existing Adaptive Hypermedia e-Learning Systems. Published in Proceedings of the 6th seminar Technologies for e-learning, TPEV.
Tomoyosi Akiba, Hozulni Tanaka(1994). A Bayesian Approach for User Modeling in Dialogue Systems. COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics.
Toñi Rios, Eva Millán, Mónica Trella, Jose Luis Pérez-de-la-Cruz, Ricardo Conejo(2001). Internet Based Evaluation System. en: Lajoie,S., Vivet, M. (Eds.). Artificial Intelligence in Education, Open Learning Environments: New Computational Technologies to Support Learning, Exploration and Collaboration. (Le Mans, France) (IOS Press, Amsterdam. Pp. 387-394.
Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, Yannis Theodoridis(2004). State-of-the-art in Privacy Preserving Data Mining. SIGMOD Record, Vol. 33, No. 1, March 2004.
Vergara H(2006). PROTUM: A Prolog Based Tool for User Modeling. WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, WIS-Report 10, (1994)
Vermunt JD(1996). Metacognitive, Cognitive and Affective Aspects of Learning Styles and Strategies: a Phenomenon graphic Analysis. Higher Education.
Warm TA(1989). Weighted likelihood estimation of ability in item response theory with tests of finite length. Psychometrika, 54:427-450.
Wenger E(1987). Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge. Los Altos, CA: Morgan Kaufmann Publishers, Inc.
Wikipedia(2007). Default Logic. http://en.wikipedia.org/wiki/Default_logic
Wim J van der Linden, Gees AW Glas(2002). Computerized Adaptive Testing: Theory and Practice. Kluwer Academic Publishers ©2002. ISBN: 0-7923-6425-2
Witkin HA, Moore CA, Goodenough DR, Cox PW(1977). Field-dependent and Field-independent Cognitive Styles and Their Educational Implications. Review of Educational Research.
Wolfgang Pohl(1999). Logic-Based Representation and Reasoning for User Modeling Shell Systems. User Modeling and User-Adapted Interaction, 9(3): 217 – 283.
Wolfgang Pohl, Jörg Höhle(1997). Mechanisms for flexible representation and use of knowledge in user modeling shell systems. Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.), User Modeling: Proceedings of the Sixth International Conference, UM97. Vienna, New York: Springer Wien New York. © CISM, 1997.
Wolfgang Wahlster, Alfred Kobsa(1986). Dialog-Based User Models. Journal Proceedings of the IEEE, Special Issue on Natural Language Processing, July 1986, Pp. 948-960.
Wolfgang Wahlster, Alfred Kobsa(1989). User Models in Dialog Systems. A. Kobsa and W. Wahlster (eds.): User Models in Dialog Systems. Berlin, Heidelberg: Springer, Pp. 4-34.
Wu HA(2002). Reference Architecture for Adaptive Hypermedia Applications. PhD thesis, Eindhoven University of Technology, ISBN 90-386-0572-2, 2002.
Yehuda Lindell, Benny Pinkas(2000). Privacy Preserving Data Mining. In advances in Cryptology – CRYPTO. Pp. 36-54.

Appendix

A. List of Acronyms

AC	Abstract Conceptualization
	Adaptive Component
AE	Active Experimentation
AEHS	Adaptive Educational Hypermedia System
AES	Adaptive Education System
AI	Artificial Intelligence
AHA	Adaptive Hypermedia for All
AHAM	Adaptive Hypermedia for All Model
AHS	Adaptive Hypermedia System
AEWBS	Adaptive Educational Web-Based System
AEIWBS	Adaptive Educational Intelligent Web-Based System
AIWBS	Adaptive Intelligent Web-Based System
ALS	Adaptive Learning System
AM	Adaptive Model
ANN	Artificial Neural Network
ATI	Aptitude-Treatment Interactions System
AWBS	Adaptive Web-based System
BN	Bayesian Network
BNE	Belief Network Engine
CA	Condition Action (rule)
CAI	Computer Assisted Instructional
	Computer Aided Instructional
CAT	Computerized Adaptive Testing
CBM	Constraint-Based Modeling
CE	Concrete Experience
CI	Communication Interfaces
CPD	Conditional Probability Distribution
	Conditional Probability Density
CPT	Conditional Probability Table
Cr	Relevance Condition
Cs	Satisfaction Condition
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network
DGJPD	Dynamic Global Joint Probability Distribution
dissim	dissimilarity measure
DM	Domain Model
	Data Mining
ECA	Event Condition Action (rule)
EM	Expectation Maximization
FD	Field Dependence
FI	Field Independency
FOL	First-Order Logic
FreeSpan	FREquEnt pattern-projected SEquential PAtterN
GJPD	Global Joint Probability Distribution
GSP	Generalized Sequential Pattern
GUMO	General User Model Ontology
GUMS	General User Modeling System

HMM	Hidden Markov Model
ICC	Item Characteristic Curve
idf	inverse document frequency
IRF	Item Response Function
IRT	Item Response Theory
itemset	Item Set
ITS	Intelligent Tutoring System
JPD	Joint Probability Distribution
K	Knowledge
KI	Knowledge Item
KN-IPCMS	KoNstanz Inter-Process Communication Management System
LCMS	Learning Content Management System
LH	Learning History
LnL(θ)	Log-Likelihood Function
L(θ)	Likelihood Function
litemset	large itemset (frequent itemset)
LJPD	Local Joint Probability Distribution
LMS	Learning Management System
LO	Learning Object
LS	Learning Style
ME	Mining Engine
ML	Machine Learning Maximum Likelihood
MLE	Maximum Likelihood Estimation
MM	Markov Model
NN	Neural Network
OOP	Object-Oriented Programming
OWL	Ontology Web Language
PDF	Probability Density Function
Pr	Probability
PROTUM	PROlog based Tool for User Modeling
QP	Quadratic Programming
RDF	Resource Description Framework
RO	Reflective Observation
sim	similarity measure
SituationML	Situation Makeup Language
SituationQL	Situation Query Language
SPADE	Sequential PAttern Discovery using Equivalent
SVM	Support Vector Machine
tf	term frequency
TLM	Triangular Learner Model
UbisWorld	Ubiquitous World
UM	User Model
UMS	User Modeling Shell/System/Server/ Service
UMT	User Modeling Tool
UserML	User Makeup Language
UserQL	User Query Language
β(a,b)	Beta distribution with two parameters *a, b*
Γ, ψ	Gamma function, Digamma function