# Chapter 13
# Student Modeling

Beverly Park Woolf

Center for Knowledge Communication, 140 Governor's Drive,
University of Massachusetts Amherst, MA 01003-4610
bev@cs.umass.edu

**Abstract.** This chapter describes how to build student models for intelligent tutors and indicates how knowledge is represented, updated, and used to improve tutor performance. It provides examples of how to represent domain content and describes evaluation methodologies. Several future scenarios for student models are discussed. For example, we envision that student models will support assessment for both formative issues (the degree to which the student has learned how to learn – for the purposes of improving learning capacity and effectiveness) and summative considerations (what is learned– for purposes of accountability and promotion). We envision that student models will track when and how skills were learned and what pedagogies worked best for each learner. Moreover, they will include information on the cultural preferences of learners, their personal interests, learning goals, and personal characteristics. Ultimately, student model servers will separate student models from tutors and will be a part of wide area networks, serving more than one application instance at a time.

## 13.1 Introduction

Student models in intelligent tutoring systems represent student competencies and learning achievements. Modeling may involve techniques to represent content skills (e.g., mathematics, art history), knowledge about learning (e.g., metacognitive knowledge), and affective characteristics (e.g., emotional state). Although students' general knowledge might be determined quickly from quiz results, their learning style, attitudes, and emotions are less easily determined and need to be inferred from long-term observations. Models may be used for assessment by measuring changes in the student in any or all three of these areas. Student models generally represent inferences about users (e.g. their level of knowledge, misconceptions, goals, plans, preferences, beliefs), relevant characteristics of users (stereotypes) and users' records, particularly past interactions with the system.

A student model in an intelligent tutor observes student behavior and creates a qualitative representation of her cognitive and affective knowledge. This model partially accounts for student performance (time on task, observed errors) and reasons about adjusting feedback to the student. By itself, the student model achieves very little; its purpose is to provide knowledge that is used to determine the

R. Nkambou et al. (Eds.): Advances in Intelligent Tutoring Systems, SCI 308, pp. 267–279.
springerlink.com © Springer-Verlag Berlin Heidelberg 2010

conditions for adjusting feedback and it supplies data to other tutor modules, particularly the teaching module. One long-term goal of the field of AI and Education is to support learning for students with a range of abilities, disabilities, interests, backgrounds, and other characteristics (Shute et al. 2005).

This chapter describes how to build student models for intelligent tutors and indicates how knowledge is represented, updated, and used to improve tutor performance. The chapter first clarifies how to build a student model and then positions student models in the context of tutor system research. The chapter concludes with a series of open questions about the future of student models.

## 13.2 Motivation for Building Student Models

Human teachers learn about students through years of experience. Master teachers often use secondary learning features, (e.g., a student's facial expressions, body language, and tone of voice) to augment their understanding of students' learning. They also learn which pedagogical strategies work best with which students. Classroom teachers support student learning in many ways, e.g., by patiently repeating material, recognizing misunderstandings, and adapting feedback. Teachers adjust teaching strategies and customize their responses to an individual's learning needs. Interactions between students and teachers provide critical data about student goals, skills, motivation, and interests. Similarly, intelligent tutors make inferences about presumed student knowledge and store them the student model. A primary reason to build a student model is to ensure that the system has principled knowledge about each student so it can respond effectively, engage students ' interest and promote learning. Customized feedback is pivotal to producing learning. Instruction tailored to students' preferred learning style increases their interest in learning and enhances learning, in part, because tutors can support weak students' knowledge and develop strong students' strengths. Master human teachers are particularly astute at adapting material to students' cognitive and motivational characteristics. In mathematics, for example, using more effective supplemental material strongly affects learning at the critical transition from arithmetic to algebra and achievement of traditionally underperforming students (Beal 1994). Students show a surprising variety of preferred media; given a choice, they select many approaches to learning (Yacci 1994). Certain personal characteristics (gender and spatial ability) are known to correlate with learning indicators such as mathematics achievement (Arroyo et al. 2004) and learning methods (Burleson 2006). Characteristics such as proficiency with abstract reasoning also predict responses to different interventions. Thus, adding more detailed student models of cognitive characteristics may greatly increase tutor effectiveness.

Student models typically represent student behavior, which includes student answers (to questions or problems), actions (writing a program), results of actions (written programs), intermediate results (scratch work), and verbal protocols. Student behavior is assumed to reflect student knowledge as well as common misconceptions. Student models are typically qualitative (neither numeric nor physical); they describe objects and processes in terms of spatial, temporal, or causal relations (Clancey 1986; Sison and Shimura 1998). These models are also approximate and

possibly partial (not fully accounting for all aspects of student behavior). In other words, tutor development focuses on computational utility rather than on cognitive fidelity (Self 1994). A more accurate or complete student model is not necessarily better, because the computational effort needed to improve accuracy or completeness might not be justified by any extra pedagogical leverage obtained. Two issues need to be considered when building student models: representing and updating student knowledge.

### *13.2.1 Representing Student Knowledge*

Resenting student knowledge takes many forms, from simple numeric rankings about student mastery to complex plans or networks explaining student knowledge (Brusilovsky 1994; Eliot 1996) and the models may encode many types of knowledge (topics, misconceptions and bugs, affective characteristics, student experience, and stereotypes). Knowledge representation is foundational to artificial intelligence. It is a methodology for encoding concepts (e.g., objects, procedures) within a computer and providing efficient operations on these concepts so computers can reason about concepts and begin to appear intelligent (Brachman and Levesque 2004). A closely related knowledge category is misconceptions, which includes well-understood errors, or incorrect or inconsistent facts, procedures, concepts, principles, schemata, or strategies that result in behavioral errors (Sison and Shimura 1998). Not every error in student behavior is due to incorrect or inconsistent knowledge; behavioral errors can also result from insufficient knowledge.

Affective characteristics, e.g., student emotions and attitudes, are also represented in student models (see chapters 10 and 17 in this book). Confusion, frustration, excitement, boredom also motivation, self-confidence, and fatigue have been represented. Affective computing typically involves emotion detection or measuring student emotion, using both hardware (pressure mouse, face recognition camera, and posture sensing devices) and software technology (e.g., machine learning), and then providing interventions to address negative affect. Stereotypes have been used, specifically collections of default characteristics about groups of students that satisfy the most typical description of a student from a particular class or group (Kay 1994). For example, default characteristics may include physical traits, social background, or computer experience. Stereotypes may represent naïve, intermediate, and expert students (Rich 1983).

### *13.2.2 Updating Student Knowledge*

Updating student knowledge is the second issue to consider in building student models. Updating is used to infer the student's current knowledge and frequently rules are used to compare the student's answers with comparable expert answers or sequences of actions. Student knowledge, as initially represented in the student model, is not usually equivalent to knowledge of the domain as represented in the domain model. The hope is that students' knowledge improves from that of a

naïve student toward that of an expert over several sessions. Conceptually these two data structures are distinct, but practically they may be very similar. Student models typically miss some knowledge contained in domain models or have additional knowledge in terms of misconceptions. Comparison methods are used to update knowledge in the two models, assuming an overlay model. In some cases, the tutor generates models of faulty behavior by using slightly changed rules (malrules) to reproduce the results produced by a student with misconceptions. Student knowledge can be updated by plan recognition or machine learning techniques, which uses data from the problem domain and algorithms to solve problems given to the student. Plan recognition might be used to determine the task on which a student is currently working, for example by predicting student behavior, refined stereotypes, and using plan recognition techniques to recognize which planning behaviors were relevant for updating the student model. If the student is pursuing plans or a recognizable set of tasks, plan recognition techniques construct the student model and compare student behavior to expert procedures to indicate on which plan the student is working. For example, the Andes tutor used updated Bayesian belief networks to infer which new topics the student might know but had not yet demonstrated (see chapter 14 in this part of the book).

## 13.3 Alternative Methods for Building Student Models

Knowledge in student models can take many forms, from simple numeric rankings about student mastery to complex plans or networks. Different techniques work better or worse for different academic disciplines. Given this variety of knowledge, many techniques are needed to update student knowledge. This section describes techniques based on either cognitive science or artificial intelligence methods used to represent student knowledge. Cognitive science techniques include model-tracing and constraint-based methods, where as AI techniques include formal logic, expert systems, plan recognition and Bayesian belief networks. This classification is not meant to be exclusionary; techniques from one category might be used in conjunction with those from the other (e.g., adding a Bayesian belief network to a model-tracing tutor).

In some cases, a student model can be quite different from the domain model based on topics. For example, affective knowledge of a student (engagement, frustration and boredom) is independent of the domain knowledge and is typically inferred and recorded in a student model. Alternatively, a procedural model might use a specially developed expert model and compare the expert solution at a finer level of granularity, at the level of subtopic or subgoals. Such models have stronger diagnostic capabilities than overlay models. Procedural models overlap with generative bug models when they use algorithms divided into stand-alone portions, corresponding to pieces of knowledge that might be performed by students (Self 1994).

Student knowledge can be updated by *plan recognition* or *machine learning* techniques, which use data from the problem domain and algorithms to solve problems given to the student. Analysis involves structuring the problem into actions to be considered. Plan recognition might be used to determine the task on

which a student is currently working. If the student is pursuing plans or a recognizable set of tasks, plan recognition techniques construct the student model and compare student behavior to expert procedures to indicate on which plan the student is working. The Andes tutor used updated Bayesian belief networks to infer which new topics the student might know but had not yet demonstrated (ref). The student model in Wayang Outpost used a Bayesian belief network to infer hidden affective characteristics (Woolf 2008). Open Student Models reflect the student's right to inspect and control the student model and participate in its creation and management. The aim of open modeling (also called overt, inspectable, participative, cooperative, collaborative, open learner, and learner-controlled modeling) is to improve the student modeling enterprise. Open user model refers to the full set of tutor beliefs about the user, including modeling student knowledge as well as preferences and other attributes. One aim of such a model is to prompt students to reflect on their knowledge (including lack of knowledge and misconceptions) and to encourage them to take greater responsibility for their learning. Learners are believed to enjoy comparing their knowledge to that of their peers or to the instructor's expectations for the current stage of their course. See chapter 15 in this part of the book for more details about open learner models.

### *13.3.1 Cognitive Science Methods*

Two cognitive science techniques, model-tracing and constraint satisfaction, are briefly described in this section. The reader can found more details in chapters 3 and 4 of this book. Each technique is based on viewing learning as a computational process. For example, model-tracing tutors (or Cognitive Tutors) are grounded in cognitive psychology theory based on ACT-R cognitive model and assumes that human learning processes can be modeled by methods similar to information processing, e.g., rules or topics that will be learned by students. These tutors provide an underlying model of the domain to interpret students' actions and follow their solution path through the problem space. Model tracing assumes that student's actions—e.g., steps in solving mathematics problems, can be identified and explicitly coded through topics, steps, or rules (if-then rules in the case of a cognitive tutor) (Anderson and Reiser 1985). The student model uses these rules or steps to represent the knowledge. The tutor then traces students' implicit execution of these rules, assuming that students' mental model state (or knowledge level) is available from their actions (Anderson et al. 1987). The tutor is mostly silent, working in the background, yet when help is needed, it reasons about student knowledge and infers whether they traveled down a path encoded by the production rules. Comparison of student actions with execution by the domain model yields error diagnoses. After a student's action, the tutor might suggest which rule or set of rules the student used to solve the problem. The Andes physics student model is an example of a system that modeled students' steps while solving physics problems, was on the edge between simple and complex knowledge (references). It required a separate model for each physics problem (several hundred problems over two semesters) and inferred a new Bayesian network for each student. Building such models required a great deal of work, even with mature authoring tools.

The operative principle for model-tracing tutors is that humans (while learning) and student models (while processing student actions) are input-output equivalents of similar processes (i.e., both humans and the model have functionally identical architectures). Cognitive methods place a premium on the empirical fit of student actions to psychological data. At each opportunity for students to employ a cognitive rule, simple learning and performance assumptions are employed (encoded as production rules) to update an estimate of the probability that the student has learned the rule (Corbett and Anderson 1995). Cognitive tutors for algebra and geometry tutors have had a large impact on educational practice and are used by more than 500,000 students in over 1400 school districts across the U.S.

Whereas model-tracing assumes that human learning processes can be accurately modeled by computer techniques, constraint-based modeling (CBM) assumes the opposite. CBM methods are based on the assumption that learning cannot be fully recorded and only errors (breaking constraints) can be recognized by a computational system. For example, constraints in the field of adding fractions might check that all denominators in the problem and solution are equal before a student adds fractions. Thus the tutor checks that in adding fractions, students submit answers where the numerator equals the sum of operand numerators, and the constraint is satisfied when all denominators (a, d, and n) are equal. CBM methods are particularly powerful for intractable domains, in which students' knowledge cannot be exactly articulated, student approaches cannot be sufficiently described, and misconceptions cannot be fully specified. Many disciplines are intractable, e.g., programming languages, music composition, legal reasoning. In many domains, student input is limited to a few steps (graph lines, variable names, and equations) and this input might include a great deal of noise (student actions unrelated to learning because of a lack of concentration or tiredness) (Mitrovic 1998). Constraint-based methods are based on a psychological theory of learning that asserts that procedural learning occurs primarily when students catch themselves (or are caught by a third party) making mistakes (Ohlsson 1996, 1994; Self 1990). Students often make errors even though they know what to do because their minds are overloaded with many things, hindering them from making the correct decision. In other words, they may already have the necessary declarative knowledge, but a given situation presents too many possibilities to consider when determining which one currently applies (Martin 2001). Thus, merely learning the appropriate declarative knowledge is not enough; students must internalize that knowledge and know how to apply it before they can master the chosen domain.

Constraints represent the application of declarative knowledge to a current situation. Each constraint is an ordered pair of conditions that reduce the solution space. These conditions are the relevance condition (relevant declarative knowledge) and satisfaction condition (when relevant knowledge has been correctly applied): IF *relevance condition* is true THEN *satisfaction condition* will also be true. The *relevance condition* is the set of problem states for which the constraint is relevant, and the *satisfaction condition* is the subset of states in which the constraint is satisfied. If the constraint is violated, the student does not know this concept and requires remedial action.

When constraints are violated, an error is signaled that translates into a student's incomplete or incorrect knowledge. CBM reduces student modeling to pattern matching or finding actions in the domain model that correspond to students' correct or incorrect actions. In the example above the tutor might say: Do you know that denominators must be equal in order to add numerators? If the denominators are not equivalent, you must make them equal. Would you like to know how to do that?

Constraint-based models are radically different from model-based tutors in both underlying theory and resulting modeling systems. Although the underlying theories of model-tracing (Anderson 1983) and of Ohlsson's performance errors (Ohlsson 1996) may be fundamentally different in terms of implementing intelligent tutors, the key difference is level of focus. Model-tracing tutors focus on the procedures carried out and faithfully model procedures to be learned, whereas performance error–based tutors are concerned only with pedagogical states and domain constraints and represent just the pedagogical states the student should satisfy, completely ignoring the path involved (Martin 2001). CBM tutors represent only basic domain principles, through constraints, not all domain knowledge (Mitrovic 1998; Ohlsson 1994). They detect and correct student errors and do model the whole domain. Thus no expert model or a bug library is needed; the process is computationally efficient and neutral with respect to pedagogy (Mitrovic et al. 2004).

Several constraint-based intelligent tutors were developed for university students learning database programming (Mitrovic 1998; Mitrovic and Ohlsson 1999). One system teaches structured query language (SQL, pronounced "Seguel"), and another teaches database techniques, e.g., design and databases (Mitrovic 1998). Web-enabled versions of the Database Tutors have been available at DatabasePlace since 2003, and tens of thousands of students have used them. In addition to the SQL-Tutor, two other database tutors were evaluated: NORMIT, a tutor to normalize a database, and Entity-Relationship (EER), a tutor on database design for generating a database schema. Remote students have used the database tutors on the Internet completely independently from the courses in which they were enrolled. These students demonstrated equivalent learning even though they had no human teacher in the loop.

### *13.3.2 Artificial Intelligence Methods*

Whereas cognitive science methods for building student models are based on the assumption that human learning (or errors) can be modeled, artificial intelligence (AI) methods are agnostic with regard to this assumption. These methods, e.g., including plan recognition and machine learning techniques, are not based on any attempt to model human learning. They are simply techniques that are successful at reasoning about knowledge. Using such methods, intelligent tutors have been able to induce student models, extend their knowledge and infer students' learning strategies. For example, machine learning (ML) paradigms enable intelligent tutors to extend themselves by learning new knowledge rather than by being programmed with that knowledge. Many ML methods have been used in tutors, e.g., Bayesian belief networks, reinforcement learning, hidden Markov models, fuzzy

logic and decision theory, see Woolf (2009) for a description. This section first describes reasoning under uncertainty, which is an underlying principle for many ML techniques and then describes, Bayesian belief networks.

ML techniques describe the probability of an event occurring. *Probability theory* is used to reason about student knowledge and to predict future action by use of data techniques based on prior or current data. ML techniques enable tutors to reason about the *probability* of events as a way to draw useful conclusions about students.

We describe Bayesian belief networks as used in intelligent tutors. Bayesian theory can roughly be boiled down to one principle: To see the future, one must look at the past (Leonhardt 2001). Bayesian methods reason about the probability of future events, given their past and current probabilities. They are based on the understanding that the world is rife with uncertainty and often not suited to clean statistical tests. Bayesian belief networks (BBNs) enable computers to combine new data with prior beliefs about data, make subjective decisions about how strongly to weigh prior beliefs, and provide a policy for keeping new information in the proper perspective (Leonhardt 2001). They provide a graphical method to design probabilistic models based on conditional probabilities and the Bayes formula.

BBNs are used in intelligent tutors to support classification and prediction, model student knowledge, predict student behavior, make tutoring decisions, and (combined with data about student's proficiencies) determine on which steps students will need help and their probable method for solving problems (Mayo and Mitrovic 2001). They represent curriculum sequencing, e.g., skills in a domain. Tutors decide among alternatives, within a probabilistic model of student knowledge and goals, which problem to present next. They seek out propositions that are both part of a solution path and ones that students are likely to know. A basic formulation of BBNs represents causal networks among hidden skills and observed actions. Building a BBN in an intelligent tutor begins by recognizing that human teachers have uncertain knowledge of students and learning and they have only explicit knowledge about observed student actions (problems solved, equations submitted, or questions answered). Like most ML techniques, BBNs begin with observed actions and infer the probability of unobserved (hidden) skills (e.g., topics that students know).

Defining the structure of a BBN begins with a statement of probability:

*P (student_knows (S) | student_knows (R)) _ .95*

which says that most students who know S also know R. is a graphical representation of the probability

*P (Answer Problem*044 | *Skilla ) _ .95*

and means that the probability is high that people who know Skill a are very likely to answer Problem 044 correctly. The BBN represents the observed variable as well as the unobserved variable (Skill a). An arc lists the probability that one variable can be inferred from another (e.g., skill from answer). If an arc joins two nodes, it means the probability of all possible values for the pointed-at-node depends on the value of the previous node. If no arc joins two nodes, it means that

the values for these nodes do not influence each other. Bayesian belief networks involve supervised learning techniques and rely on the basic probability theory and data methods. Graphical models are directed acyclic graphs with only one path through each (Pearl 1988). In intelligent tutors, such networks often represent relationships between prepositions about the student's knowledge and tutoring decisions. Nodes often represent the state of the teaching world (topics, level of skill, or correctness of questions).

Many BBNs have been developed to model student knowledge in intelligent tutors. One category of student model BBN is *expert-centric* or networks and conditional probabilities specified either directly or indirectly by experts (Mayo and Mitrovic 2001). Experts create the network structure or topology, draw the arcs, and defi ne the conditional probability of the arcs. Consider a naïve representation of student knowledge of physics, which states that student success on Problem 023 indicates understanding of Newton's law and that understanding Newton's law may result from reading the text.

A second example expert-centric student model is HYDRIVE (Mislevy and Gitomer 1996), which used a highly abstract method similar to Andes. HYDRIVE trained personnel to troubleshoot aircraft hydraulics involved in flight control, landing gear, and aerial refueling. It simulated features of troubleshooting by presenting students with a video sequence problem in which a pilot described the aircraft malfunction to technicians (e.g., " The rudders do not move during the preflight check"). The student performed troubleshooting procedures by accessing video images of aircraft components.

A second category of student model BBNs is *data-centric* models or networks that are specified from real-world data (e.g., log data) (Mayo and Mitrovic 2001). The structure and conditional probabilities are learned primarily from data collected from real-world evaluations of the tutor. This involves mining data from previous users of the system and classifying relationships (e.g., between solution of a problem and inferred student skill). Confirmation that a student really has a particular skill might come from evaluating student performance on simple problems that require only this skill. Several tutors used a data-centric approach. In one tutor, the hidden parameters of a Bayesian network were inferred using expectation maxima, an ML technique that deals with missing data (Ferguson et al. 2006). In another system, much of the student model was inferred from data (Johns and Woolf 2006).

## 13.4 Discussion and Future Research on Student Models

Student model have been enormously successful, enabling tutors to predict student performance in a number of fields. A variety of student models have been built including open and affective models. This chapter outlined the nature of student models and described how to build and update these models.

Many improvements are needed to fully represent and reason about students. We clearly need socio-technical solutions, recognizing the need for solutions that have large social components. In this section, we suggest some technology areas that provide promising developments. Such technology predictions are based on

current research trends in student models and speculative at best. No one can know the future of student models nor accurately specify solutions for their development due, in part to the rapidly changing nature of software, languages, networks and hardware. Yet, we list considerations about likely future capabilities for student models and identify technologies that seem promising for their future development.

We envision that future student models will be *complex*, not only representing what students know, probably do and have abilities for, but other factors too. For instance, student models will probably track when and how skills were learned and what pedagogies worked best for each learner (Bredeweg et al. 2009). Moreover, student models might include information on the cultural preferences of learners, their personal interests, learning goals, and personal characteristics and will be able to select the optimal mix of learning environments, pedagogy, visualizations, and contexts that maximize engagement, motivation and learning outcomes for each individual. When the learner is part of a group, the model should provide the best estimate among the individuals who are part of the group, to attribute authorship.

We also envision that future student models will *support assessment* for both formative issues (the degree to which the student has learned how to learn – for the purposes of improving learning capacity and effectiveness) and summative considerations (what is learned–for purposes of accountability and promotion). In this regard, approaches to student modeling are needed that lead to valid and reliable inferences about student learning that are both diagnostic and predictive. Such a perspective concurs with the view that assessment should be dynamic over time.

We expect that in the future *privacy issues* in educational student models will be adequately addressed. Student privacy concerns and national and international privacy legislation have a considerable impact on what education applications may do. Strict privacy enhancing software tools and Internet services are needed. Generic student modeling systems will facilitate compliance with such regulations, as well as support privacy-enhancing services.

Currently student modeling techniques are developed and encoded into each individual educational program. For example, to measure a specific construct (e.g., algebra skills, persistence, help-seeking behavior) requires a substantial amount of effort to construct the relevant conceptual and statistical models (Shute et al. 2009). The construction cost of such models is about one year's time for a graduate student. The current approach does not scale to the increasing numbers of electronic learning environments that should have student models. We suggest that future student models will be developed as *shells* that exist independent of the instructional software and attached to the software only after they have been activated (Kobsa 2007). The term "shell" is borrowed from the field of expert systems and describes environments containing the basic components of expert systems (Nii 2009). Associated with each shell is a prescribed method for building a student model by configuring and instantiating encoded components. Shells support construction of knowledge bases through use of inference engines. Instead of building a student model for each instructional system, generic models will define their basic functionality and then be further constructed during development time.

These generic student models will serve as separate components and include a representation system for expressing the particular domain knowledge (e.g., logic formalism, rules, or simple attribute-value pairs) and a reasoning mechanism for deriving assumptions about users from existing models.

Most likely, future *student model servers* will be readily available for education. Servers are similar to generic student models in that they are separate from the application and will not run as part of it (Kobsa 2007). Student model servers will probably be part of local or wide area networks and serve more than one application instance at a time.

It is likely that educational data mining (EDM) and machine learning (ML) techniques will play larger role in augmenting student models automatically. These new approaches are presented in chapter 16 of this book. ML refers to a system's ability to acquire and integrate new knowledge through observations of users and to improve and extend itself by learning rather than by being programmed with knowledge (Shapiro 1992). These techniques organize existing knowledge and acquire new knowledge by intelligently recording and reasoning about data. For example, observations of students' past behavior will be used to provide training examples that will form a model designed to predict future actions (Webb et al. 2001). These techniques have been used to acquire models of individual students interacting with educational software and group them into communities or stereotypes with common interests. ML techniques are promising in cases where very large sets of usage data are available, like educational software on the Web (Kobsa 2007). These techniques improve teaching by repeatedly observing how students react and generalizing rules about the domain or student. These paradigms enable tutors to adapt to new environments, use past experience to inform present decisions, and infer or deduce new knowledge. Teaching environments will use ML techniques to acquire new knowledge about students and predict their learning (Arroyo and Woolf 2005; Johns and Woolf 2006).

One last prediction is that student models will probably adapt to new student populations. Obviously students have a variety of learning needs (e.g., exceptional students learn beyond their age group, special needs students require accommodations). Yet educational software is often built for the average student, expecting all students to be at the same academic level and be ready to learn. This is not so. Students are at various levels, have different skills and different learning abilities. No single instructional method works for all students and all disciplines. ML techniques can help enable software to acquire knowledge about distinct student groups and add that information to the tutor. Techniques can make decisions based on experience with prior populations and enable software to reason "outside" the original variables that made up the system.

## References

Anderson, J.R.: The Architecture of Cognition. Harvard University Press, Cambridge (1983)

Anderson, J.R., Reiser, B.: The Lisp Tutor. BYTE 10, 159–175 (1985)

Anderson, J.R., Boyle, C.F., Farrell, R., Reiser, B.J.: Cognitive principles in the design of computer tutors. In: Morris, P. (ed.) Modelling Cognition. Wiley, Chichester (1987)

Arroyo, I., Woolf, B.: Inferring Learning and Attitudes from a Bayesian Network of Log File Data. In: Looi, C.K., McCalla, G., Bredeweg, B. Breuker, J. (eds.) Twelfth International Conference on Artificial Intelligence in Education, Amsterdam (2005)

Arroyo, I., Beal, C.R., Murray, T., Walles, R., Woolf, B.P.: Web-Based Intelligent Multimedia Tutoring for High Stakes Achievement Tests. In: James, R.M.V., Lester, C., Paraguaçu, F. (eds.) ITS 2004. LNCS, vol. 3220, pp. 468–477. Springer, Heidelberg (2004)

Beal, C.R.: Boys and Girls: The Development of Gender Roles. McGraw Hill, New York (1994)

Brachman, R., Levesque, H.: Knowledge Representation and Reasoning. Morgan Kaufmann (part of Elsevier's Science and Technology Division) (2004)

Bredeweg, B., Arroyo, I., Carney, C., Mavrikis, M., Timms, M.: Intelligent Environments. In: Global Resources for Online Education Workshop, Brighton, UK (2009)

Brusilovsky, P.: The Construction and Application of Student Models in Intelligent Tutoring Systems. Journal of computer and systems sciences international 32(1), 70–89 (1994)

Burleson, W.: Affective Learning Companions: Strategies for Empathetic Agents with Real-Time Multimodal Affective Sensing to Foster Meta-Cognitive and Meta-Affective Approaches to Learning, Motivation, and Perseverance. MIT PhD Thesis. (2006), http://affect.media.mit.edu/publications.php

Clancey, W.: Qualitative Student Models. Annual Review of Computer Science 1, 381–450 (1986)

Corbett, A.T., Anderson, J.R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction 4, 253–278 (1995)

Eliot, C.: An Intelligent Tutoring System Based Upon Adaptive Simulation, Computer Science Department. University of Massachusetts, Amherst (1996)

Ferguson, K., Arroyo, I., Mahadevan, S., Woolf, B., Barto, A.: Improving Intelligent Tutoring Systems: Using EM to Learn Student Skill Levels. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 453–462. Springer, Heidelberg (2006)

IJAIED 2009 special issue on Authoring IEEE-TLT 2009 (2009)

Johns, J., Woolf, B.: A Dynamic Mixture Model to Detect Student Motivation and Proficiency. In: Conference on Artificial Intelligence (AAAI 2006), pp. 2–8. AAAI Press, Menlo Park (2006)

Kay, J.: Lies, Damned Lies and Stereotypes: Pragmatic Approximations of Users. In: Proceedings of the Fourth International Conference on User Modeling, pp. 175–184 (1994)

Kobsa, A. (ed.): Adaptive Web 2007. LNCS, vol. 4321. Springer, Heidelberg (2007)

Koedinger, K., Corbett, A.: Cognitive Tutors. In: Sawyer, K. (ed.) The Cambridge Handbook of the Learning Sciences. Cambridge University Press, Cambridge (2006)

Leonhardt, D.: Adding Art to the Rigor of Statistical Science. The New York Times, New York (2001)

Martin, B.: Intelligent Tutoring Systems: The Practical Implementations of Constraint-Based Modeling. Computer Science. University of Cantebury, Christchurch (2001), http://coscweb2.cosc.canterbury.ac.nz/research/reports/PhdTheses/2003/phd_0301.pdf

Mayo, M., Mitrovic, A.: Optimizing ITS Behavior with Bayesian Networks and Decision Theory. International Journal of Artificial Intelligence in Education 12, 124–153 (2001)

Mislevy, R.J., Gitomer, D.H.: The role of probability-based inference in an intelligent tutoring system. User Modeling and User Adapted Interaction 5(3-4), 253–282 (1996)

Mitrovic, A.: Experience in Implementing Constraint-Based Modeling in SQL-Tutor. In: Goettl, B.P., Halff, H.M., Redfield, C.L., Shute, V. (eds.) ITS 1998. LNCS, vol. 1452, pp. 414–423. Springer, Heidelberg (1998)

Mitrovic, A., Ohlsson, S.: Evaluation of a Constraint-Based Tutor for a Database Language. Artificial Intelligence in Education 10(3-4), 238–256 (1999)

Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A.: DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. Journal of Interactive Learning Research (JILR) 15(4), 409–432 (2004)

Nii, H.P.: Expert Systems Building Tools: Definitions (2009), http://www.wtec.org/loyola/kb/c3_s2.htm

Ohlsson, S.: Constraint-Based Student Modeling. In: Greer, J.E., McCalla, G. (eds.) Student Modeling: The Key to Individualized Knowledge-Based Instruction, pp. 167–189 (1994)

Ohlsson, S.: Learning from performance errors. Psychological Review 103, 241–262 (1996)

Pearl, J.: Probabilistic Reasoning in Intelligent Systems Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)

Razzaq, L., Patvarczki, J., Almeida, S., Vartak, M., Feng, M., Heffernan, N., Koedinger, K.: The ASSISTment Builder: Supporting the Life Cycle of Tutoring System Creation. IEEE Transaction on Learning Technologies 2(2), 157–166 (2009)

Rich: Users are Individuals: Individualizing User Models. International Journal of Man-Machine Studies 18, 199–214 (1983)

Self, J.A.: Bypassing the intractable problem of student modelling. In: Frasson, C., Gauthier, G. (eds.) Intelligent tutoring systems: at the crossroads of artificial intelligence and education. Ablex Publishing, Norwood (1990)

Self, J.A.: Formal Approaches to Student Modeling. In: McCalla, G., Greer, J.E. (eds.) Student Models: The Key to Individual Educational Systems. Springer, New York (1994)

Shapiro, S.: Encyclopedia of Artificial Intelligence, 2nd edn. John Wiley & Sons, Chichester (1992)

Shute, V.J., Graf, E.A., Hansen, E.: Designing adaptive, diagnostic math assessments for individuals with and without visual disabilities. In: PytlikZillig, L., Bruning, R., Bodvarsson, M. (eds.) Technology-based education: Bringing researchers and practitioners. Information Age Publishing, Greenwich (2005)

Shute, V.J., Zapata, D., Kuntz, D., Levy, R., Baker, R., Beck, J., Christopher, R.: Assessment: A Vision, Global Resources for Online Education (GROE), Tempe Arizona (2009)

Sison, R., Shimura, M.: Student Modeling and Machine Learning. International Journal of Artificial Intelligence in Education 9, 128–158 (1998)

Webb, G., Pazzani, M., Billsus, D.: Machine Learning for User Modeling in User Modeling and User-Adapted Interaction, Netherlands 11, 19, 29 (2001)

Woolf, B.: Building Intelligent Interactive Tutors: Student-centered strategies for revolutionizing e-learning, vol. 480. Morgan Kaufmann, Burlington (2008)

Yacci: A Grounded Theory of Student Choice in Information-Rich Learning Environments. Journal of Educational Multimedia and Hypermedia 3(3-4), 327–350 (1994)