Pre-publication draft of a paper which appeared in the Proceedings of the Third Australian Joint Conference on Artificial Intelligence (AI 89), pp 195-205.

A machine learning approach to student modelling

Geoffrey I Webb

Department of Computing and Mathematics, Deakin University, Geelong 3217

Abstract

This paper describes an application of established machine learning principles to student modelling. Unlike previous machine learning based approaches to student modelling, the new approach is based on attribute-value machine learning. In contrast to many previous approaches it is not necessary for the lesson author to identify all forms of error that may be detected. Rather, the lesson author need only identify the relevant attributes both of the tasks to be performed by the student and of the student's actions. The values of these attributes are automatically processed by the student modeler to produce the student model.

Keywords and phrases:

Student modelling, intelligent tutoring systems, machine learning.

1. Introduction

The development of knowledge-based systems has paved the way for intelligent tutoring systems (ITS) - systems whose purpose is the communication to learners of the knowledge in an internal knowledge-base (Wenger, 1987.)

An important requirement of any such system is that it be able to evaluate the student's mastery of the knowledge to be communicated. Without the ability to evaluate the student's mastery of a domain, an ITS will be unable to coordinate effectively its attempts at communication.

Most intelligent tutoring systems maintain a record of their on-going evaluation of the student's mastery of the subject matter. This record is known as the *student model*. Previous approaches to student modelling have described the student's mastery of the domain as expert systems (Clancey, 1987; Sleeman, 1984; Reiser, Anderson & Farrell, 1985; Goldstein, 1979), scripts (Stevens. Collins & Goldin, 1982) and procedures (Brown & Burton, 1978.)

An ITS has three potential sources of knowledge from which to construct a student model. On the one hand, it has prior knowledge of the domain in the form of its knowledge-base. A second knowledge source is prior knowledge about likely forms of error with regard to a domain. This can take the form of a collection of possible errors (such as the bug libraries of Brown & Burton, 1978) or of constraints on the possible forms of error that may occur (VanLehn, 1982.) Finally, the system has its observations of the student's performance during the lesson.

The modelling system must use induction, guided by the domain knowledge and knowledge of likely forms of error, to produce a student model that explains the observed behaviour.

Although the discipline of *machine learning* has conducted extensive formal research into theories and methodologies for automated induction, few student modelling systems have taken advantage thereof. This paper examines how established machine learning principles can be applied to student modelling.

2. Previous uses of Machine Learning in Student Modelling

Two previous student modelling systems have incorporated general machine learning principles and techniques - ACM (Langley, Ohlsson & Sage, 1984) and LMS (Sleeman, 1984.)

ACM diagnoses student understanding in the domain of elementary subtraction. The student is presented with a simple subtraction problem and her/his answer is analysed.

Analysis of each answer has two stages. First, the student's solution path is inferred. To do this it is necessary to pre-define the problem space in which the student is working. This is achieved by specifying a set of operators which the student may use. A constrained search is conducted to determine the most plausible path through the search space to the observed answer.

When ACM has a record of a number of such paths it is able to apply the second stage. A discrimination net is constructed for each operator that specifies the conditions under which it is used. This enables the detection of differences between the correct restrictions on the use of the operator and the restrictions placed by the student.

LMS operates in the domain of elementary algebra. It has a library of rules and mal-rules. Rules represent correct production rules for solving problems in the domain. Mal-rules are erroneous versions of the correct rules. The mal-rules used by the system have been identified by the domain expert. A model of the domain is specified by a set of rules and mal-rules and an ordering for the application of those rules.

Before interactions with a student LMS generates a set of all models that it will consider. This set of models is restricted by both domain specific and pragmatic heuristics. Problems are generated that discriminate between models. The student's responses are used to select which of the possible models best describes the student.

A problem faced by both of these systems is that they rely on prior identification of operators that the student may apply. Extensive studies of the domain of elementary subtraction have shown that even in such a simple domain students typically use greatly varied approaches (VanLehn, 1982.) Neither of the above systems will be able to produce sensible models of a student who has a different *viewpoint* (Wenger, 1987) of the domain from that of the system.

3. Comparison with attribute-value Machine Learning

One of the most highly developed areas in machine learning is induction from attribute-value data (Hunt, Marin & Stone. 1966.) Examples are presented to the machine learning system described in terms of a vector of attribute values and a class. The system develops a mapping from combinations of attribute values to classes based on the available examples. When applied to a case (described in terms of a vector of attribute values) the mapping will predict a class for that case.

Two main formalisms have been adopted for representing such a mapping. Decision trees (Quinlan, 1986) are trees whose internal nodes represent tests on attribute values and whose leaves represent classes. When a decision tree is applied to the vector of attribute values representing a case the tree is traversed starting from the root and ending at a leaf. At each internal node the relevant test is applied. The result determines a branch to follow leading to another node. The leaf that is reached specifies the class for the case.

The other major formalism that has been adopted is the *characteristic description* (Michalski, 1984.) A characteristic description (in the context of attribute-value machine learning) can be any form of description of arrays of attribute values. Characteristic descriptions are associated with classes. When a case is examined, the characteristic description that best describes the array of attribute values for that case is selected and the associated class is assigned to the case. Any characteristic description that is true of a case is said to *cover* that case.

Characteristic descriptions can be reformulated as production rules. To achieve, this, each characteristic description is treated as the antecedent of a rule. The consequent of each rule specifies the class of any case whose vector of attribute values satisfies the antecedent. This is determined from the class which is associated with the characteristic description that formed the antecedent of the rule. When applied to a

case, the rule is selected whose antecedent is best satisfied by the cases attribute values. The consequent of this rule assigns a class to the case.

Characteristic descriptions can be partially ordered on generality (Mitchell, 1977.) A characteristic description $\propto$ is a *generalisation* of another characteristic description b iff it is necessarily true that $\propto$ covers all cases that b covers and that $\propto$ may cover some cases that b does not. If $\propto$ is a generalisation of b then b is a *specialisation* of $\propto$. For example, **X is red** is a generalisation of **X is red and X is rectangular** because the former must cover all cases covered by the latter and may cover cases not covered by the latter (namely, any red non-rectangular cases.)

Attribute-value machine learning differs greatly from the forms of machine learning used by ACM and LMS. Attribute-value machine learning can be used to produce a description of what decision will be made in a particular situation without regard for the cognitive processes that would occur in a human making that decision. By contrast, both ACM and LMS attempt to develop a model of the procedures and strategies that a student adopts in order to reach a decision. Clearly the latter is the harder of the two tasks.

4. Feature Based Modelling

FBM (Feature-Based Modelling) is an approach to student modelling whose core is the application of attribute-value machine learning techniques to student modelling. This paper concentrates on this core aspect of FBM. The approach is described in full by Webb (1989.)

FBM describes the tasks that a student tackles as vectors of attribute values. The student's actions are described as classifications of those problems.

FBM does not restrict the description of each student's action to a single classification. Rather, like the description of the tasks being tackled, the description of the student actions can take the form of a vector of attribute values. Standard machine learning techniques can then be applied separately with each of these attributes being treated as the target of classification.

An FBM model is able to describe the regularities between task attributes and action attributes. It achieves this without having to produce a detailed model of the internal operations of the student's cognitive system.

For consistency with the ITS for which FBM was originally developed (Webb, 1988), the attributes that FBM considers are called *feature choices* and attribute values are called *features.*

There are two types of features. *Task features* describe the tasks that the student tackles. It is intended that these features should provide a very broad description of the tasks, describing not only the formal problem solving features of the task but also any relevant features of the particular environment in which the task is tackled. This is important as it will frequently not be possible to interpret a student's actions correctly except in the context of factors external to the formal task being undertaken. To provide an extreme example, if one student is copying another student's solutions then it will nor be possible to create an accurate model of the former student without reference to details of the behaviour of the latter student.

Action features describe the student's performance when tackling a task. Any aspects of the student's behaviour which can be observed and which may be relevant to understanding their mastery of the domain may be included as an action feature.

To clarify these concepts we will refer to a tutoring system in which this methodology is imbedded. The Unification Tutor examines the unification of terms from the Prolog programming language. This has been chosen as a simple yet non-trivial problem solving task.

Each task is specified by two Prolog terms. The student must either provide a most general unifier for the terms or state that the terms cannot be unified. Table 1 lists some tasks and their task features.

Term 1	Term 2	Task Features
x(X,X)	x(a,Z)	Two Compound Terms
		The Terms are Different,
		The Functors Have Identical Names,
		The Functors Have Identical Arity,
		The Terms Contain Variables,
		A Variable Appears More Than Once,
		No Variable Has Inconsistent Bindings,
		No Variable Opposes Itself,
		All Arguments Unify.
x(X,X)	X(a,b)	Two Compound Terms
		The Terms are Different,
		The Functors Have Different Names,
		The Functors Have Identical Arity,
		The Terms Contain Variables,
		A Variable Appears More Than Once,
		A Variable Has Inconsistent Bindings,
		No Variable Opposes Itself,
		Not All Arguments Unify.

Table 1: Some unification tasks and their task features.

Table 2 provides some examples of tasks, the student's responses and a selection of the action features exhibited by those responses.

Term 1	Term 2	Answer	**Action Features**
x(X,X)	x(a,Y)	{X=a, Y=a}	The Answer is Correct
			The Terms Unify,
			No Variable has Multiple Bindings.
x(X,X)	x(a,Y)	none	The Answer is Not Correct,
			The Terms Do Not Unify.
x(X,X)	x(a,Y)	{X=a, X=Y}	The Answer is Not Correct,
			The Terms Unify,
			A Variable has Multiple Bindings.
x(X,X)	x(a,b)	none	The Answer is Correct,
			The Terms Do Not Unify.
x(X,X)	x(a,b)	{X=a, X=b}	The Answer is Not Correct,
			The Terms Unify,
			A Variable has Multiple Bindings.

Table 2: Some responses and their action features.

The features used by the Unification Tutor and their meanings are described by Webb & Van der Klooster (1989.)

The action features that describe appropriate actions for the student to perform in response to a task are called the *appropriate action features* for that task. The current implementations of FBM assume that only one action will be appropriate for a task and hence that only one consistent set of actions features will he appropriate. However, this is not a necessary restriction on the methodology.

Associations are the target of the machine learning component of FBM. An association can be viewed as a production rule. The antecedent is a set of task features and the consequent is a single action feature. Each association represents the system's determination that the given set of task features, whenever present, lead the student to act in the manner represented by the action feature.

An association can be viewed alternatively as a characteristic description of the tasks for which a particular feature is present in the student actions.

The aim of FBM is to produce a student model consisting of associations such that for any set of task features describing a task for the student, the system is able to predict all of the action features that describe the student's response to the task.

FBM uses generalisation to construct the set of associations from the observed relationships between sets of task features and individual action features. In principle, the set of associations is the set of all most general characteristic descriptions $C_1 \wedge C_2 ... C_n \Rightarrow A$ such that for every task that exhibited task features $C_1 \wedge C_2 ... C_n$, the student's actions exhibited feature A.

This differs significantly from most approaches to characteristic description machine learning which attempt to select a minimal set of characteristic descriptions that are capable of assigning the correct class to every example. Using all most general characteristic descriptions increases the chance of forming multiple characteristic descriptions that apply to a single case but are associated with different classes. On the other hand, it decreases the chance of relevant characteristic descriptions being excluded from the model on the grounds that a simpler model can be constructed.

An even greater advantage of using all most general characteristic descriptions is that it decreases the probability that a new example will force a radical revision of the model. If a minimal set of most general characteristic descriptions was used, once a choice between alternative characteristic descriptions was subsequently demonstrated to be incorrect, the inappropriate characteristic description would have to be removed from the model and another characteristic description inserted in its place. This would result in frequent dramatic revisions to the student model.

By contrast, when the set of all most general characteristic descriptions are included in the student model, most changes to the model will take the form of generalisations or specialisations of existing associations. An association will only be deleted when either the student's treatment of the domain changes and the association is no longer supported, or if additional observations of the student allow the association to be specialised to a characteristic description which is itself a specialisation of another association in the model. Totally new associations are only added when the student's treatment of the domain alters or when there have not previously been sufficient examples to warrant their inclusion. This strategy leads to more gradual change in the student model which is more conducive to consistent and comprehensible interactions with the student.

5. Allowance for noise

A major problem encountered in student modelling is the presence of noise in the system's observations of the student. That is, in many cases the observable behaviour of the student will not be consistent with her/his actual comprehension and mastery of the domain. Examples of such situations are slips, such as accidentally pressing the wrong key on the keyboard, and inattentiveness causing failure to take in and or process all of the relevant details.

As a result of noise it is possible that the associations that should be in student model will be either more generalised or more specialised than those that fully describe the observed behaviour.

FBM addresses this problem by placing a requirement of sufficient evidence upon the associations that it will accept. This allows associations to he accepted under some circumstances in which there are counter-examples in the observed behaviour, and rejects some associations for which there are examples in the observed behaviour.

The current criterion for sufficient evidence for accepting a hypothesis is stated in terms of two quantities – *P*, the number of positive cases consistent with the hypothesis, and *N*, the number of cases inconsistent with the hypothesis. There is considered to be sufficient evidence for accepting an hypothesis if $P + N \geq 4$ and $\frac{P}{P+N} > .8$. The first condition prevents the system from accepting associations from too few examples. The second condition allows the system to accept associations for which less than 20% of the relevant evidence is negative. The limits 4 and .8 used in these conditions are the result of informal experimentation.

On the face of it, the possibility of accepting associations for which almost 20% of the relevant evidence is negative suggests that the system will tend to accept overly general associations. However, there is an additional guard against this possibility. If an association $a \Rightarrow b$ is overly general then it should be the case that there are regularities in the counter-examples that it covers. If the student has tackled such counter-example tasks then the FBM model will contain an association between a specialisation of a and a different action feature from the feature choice for b. Under these circumstances, the overly general association is rejected enabling it to be replaced by one or more of its specialisations.

6. Erroneous associations

It is important to recognise that many associations will be *appropriate* associations. For example, it is appropriate to associate the task features `The Terms Do Not Contain Variables` and `The Terms are Identical` with the action feature `The Terms Unify`. Whenever confronted by the task of finding a most general unifier for two identical terms that do not contain variables the student should provide a response that indicates that they unify.

In general it is important to identify the erroneous aspects of the student's comprehension of the domain. To this end, erroneous associations are identified and placed in a separate partition of the student model. It is these associations that are most commonly used by the tutoring system.

An association is identified as erroneous only if the student has tackled a task covered by the task features of the association for which the action feature was inappropriate and for which the action feature described the student's actions. That is, the association must have been observed to apply in a situation in which it should not have applied.

This measure also suppresses overly general associations that would otherwise be accepted solely because no counter-examples have been observed. Until the association has been observed in an inappropriate context it is not held to be erroneous and so will not be acted upon by the tutor.

7. Current implementations

FBM has been implemented in three separate systems.

The first implementation is an off-line student modelling sub-system for the DABIS knowledge-based tutoring system (Webb, 1988.) This sub-system takes a record of a student's performance and produces a student model. Unfortunately, there is no facility for this model to be accessed by the tutoring sub-system. Thus, it cannot be used to aid interactions with the student. Its primary function is to provide intelligent student evaluation for use by teachers.

This system has been used in a number of informal trials, primarily in the context of a lesson on English word classes for Linguistics students. Experience gained from these trials has helped plan the integration of the student modelling facilities into an interactive tutoring environment.

Amato & Tsang (1988) have implemented a piano scale tutor that utilises FBM. This system uses seven feature choices to describe the tonic, hand motion, number of octaves, touch and tone of a scale. These feature choices contain only action features. They describe the features of an attempt to play the scale. The tutor requests that the student play a specified scale. The analysis compares the appropriate action features for a task with the features observed in the students attempt to play the scale. The results of the analysis are used to select further scales for practice.

A third system has been created to provide a test bed for the interactive use of FBM student modelling. It is a computer-based tutor for the domain of the unification of terms from the Prolog programming language. This tutor consists of seven sub-systems, a *feature set selector,* a *task generator,* a *student interface,* an *action analyser*, a *task adviser*, an *FBM student modeler* and a *model-based adviser*.

The feature set selector consults the FBM student model and selects a set of task features that are either -

• not associated with the action feature `The Answer Is Correct`; or

• have a subset erroneously associated with any action feature

and have all prerequisite feature sets associated with the action feature `The Answer Is Correct` (prerequisite feature sets are specified by the lesson author.)

This ensures that the student is only presented with tasks that the system does not yet have sufficient evidence to believe that s/he has mastered and is not presented with tasks that are too advanced.

The task generator generates a task with the features in the feature set output by the feature set selector. This task is presented to the student by the student interface which also handles input of the student's response. The response is passed to the action analyser which produces a set of action features that describe the student's actions.

The first source of feedback is managed by the task adviser which takes as input the task features of the task and the action features describing the student's actions. From these it generates suitable *domain model driven* advice and comments. The task adviser does not refer to the student model when generating comments. Rather it acts on general domain based assumptions as to the likely basis of any observed behaviour.

The second source of feedback is managed by the model-based adviser. This sub-system scans the student model for a suitable association on which to comment. The selected association is described to the student and it is suggested that s/he reconsider how s/he tackles the tasks described by the association. The advice from this sub-system is *student model driven*.

An association is considered suitable for comment if it has not previously been commented upon and it is erroneously demonstrated by the student's immediately preceding action. An association is erroneously demonstrated by an action if its task features describe the task on which the student was engaged, its action feature describes the student's action, and the action feature was not appropriate to the task. These conditions ensure that the model-based adviser's advice is both salient and pertinent.

In future implementations of the Unification Tutor, the system will offer to demonstrate how the selected association is erroneous and how it would tackle the relevant class of tasks.

Figure 1 shows an interaction with the Unification Tutor which includes comments generated by both the task adviser and the model-based adviser. The model-based adviser's comment is based on an association that has been detected over an extended period of interaction between the task feature `A Variable Appears More Than Once` and the action feature `A Variable has Multiple Bindings`.

It should be noted that the Unification Tutor does not assume that the student has adopted any particular viewpoint of the domain. It will operate effectively with any approach to problem solving in the domain.

Consider the following two terms

Term (B, variable (B))

Term (Atom, Atom)

Enter the most general, unifier for these terms or type none, help or exit.

=>{Atom=B, Atom=variable(b)}

A substitution should never contain the same variable on the left of more than one pair. Your answer has **Atom** on the left of more than one pair.

It appears to me that whenever you examine two terms that have the one variable appearing more than once you give an incorrect answer.

Perhaps you should reconsider how you tackle such problems.

Press Space to continue.

The student interface presents the task to the student. The student's answer is underlined. The first comment is provided by the task adviser. The boxed comment is provided by the model-based adviser.

Figure 1: The unification tutor in action.

8. Scope and Limitations

As demonstrated by the range of domains for which successful implementations exist, FBM has very general application. The major limitation is the requirement that it be possible to identify relevant features of the student's problem solving behaviour.

The identification of a student's problem solving behaviour can be performed at a number of different levels of detail. The most simple characterisation is a simple identification of whether it is appropriate for the current task. More detailed characterisations are necessarily domain specific in nature. Lesson authors must provide the means of identifying the relevant action features.

The use in the student model of all most general characterisations, rather than a minimal set, entails major computational overheads. The modelling system maintains a record for every possible association. This results in an exponential increase in computational complexity as the number of task features increases. This limits the number of features that may be used.

However, the limits imposed are not too severe. The Unification Tutor is able to provide satisfactory interactive FBM modelling using 22 task features and 14 action features. During use a La Trobe University in September 1989 the average CPU time spent on student modelling after a task was 4.19 CPU seconds (11.45 seconds real time) on a heavily loaded Pyramid 90 mx. As modelling occurs while the student is reading the feedback provided by the Task Advisor, the student rarely experiences a significant delay due to modelling. Further, there is considerable room for optimisation of this modelling system, which suggests that it should be able to provide acceptable performance in considerably more complex domains

9. Conclusion

FBM is an approach to student modelling based on standard attribute-value machine learning techniques. It has been successfully implemented in three separate computer-based tutoring systems The domains to which it has been applied range from analytic (English word classes) to skill acquisition (piano scales) to problem solving (unification.)

FBM provides an approach to student modelling whose success does not depend upon the lesson author identifying all forms of mistakes that the student may make. Nor need the lesson author specify the problem solving operators that the student may use. Rather, the lesson author need only specify what features of a task and of a student's response to a task may be relevant to student modelling. This greatly simplifies the demands placed upon the lesson author and increases the versatility of the modelling system.

10. References

Amato, N.H. & Tsang, C. P. (1988) *Student Modelling in a Scale Tutoring System*, Tech. Rep. 88/5, Department of Computer Science. University of Western Australia, Nedlands, WA.

Brown, J. S. & Burton, R, R. *Diagnostic models for procedural bugs in basic mathematical skills*, Cognitive Science, Vol. 2(1978), pp. 153-192.

Clancey, W J. (1987) *Knowledge-Based Tutoring: The GUIDON Program*, MIT Press, Cambridge, Mass.

Goldstein, I. P. *The genetic-graph: A representation for the evolution of procedural knowledge* International J. Man-Machine Studies, Vo 11 (1979), pp. 51-77.

Hunt. E. B., Marin, J & Stone, P. J. (1966) *Experiments in induction*, Academic Press, New York.

Langley P., Olhsson, S.. & Sage. S. (1984) *A Machine Learning Approach to Student Modelling*, Tech. Rep, CMU-RI-TR847, The Robotics institute, Carnegie University.

Michalski, R. S. (1984) *A theory and methodology of inductive learning*, in R. S. Michalski, J. G. Carbonell & T. M, Mitchell (Eds.) Machine Learning: An Artificial intelligence Approach, Springer-Verlag, Berlin, pp 83- 129.

Mitchell, T. M. (1977) *Version spaces: A candidate elimination approach to rule learning*, Proceedings of the Fifth international Joint Conference on Artificial Intelligence, pp. 305-310.

Quinlan, .J. R. *Induction of decision trees*, Machine Learning, Vol. 1 (1986), pp. 81-106.

Reiser, B.J., Anderson, J, R. & Farrell, R. G. (1985) *Dynamic student modelling in an intelligent tutor for Lisp programming*, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, pp. 8-14.

Sleeman, D.H. (1984) *Inferring student models for intelligent computer-ajded instruction*, in R S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.) Machine Learning: An Artificial Intelligence Approach, Springer-Verlag. Berlin, pp. 483-510.

Stevens, A. L., Collins, A. & Goldin, S. E. (1982) *Misconceptions in students' understanding*, in D. H. Sleeman & J. S. Brown (Eds.) Intelligent Tutoring Systems, Academic Press, London, pp. 13-24

VanLehn, K Bugs are not enough: Empirical studies of bugs, impasses, and repairs in procedural skills, Journal of Mathematical Behaviour, Vol. 3(1982), pp. 3-72.

Webb, G. I. *A knowledge based approach to computer-aided learning*, International J. Man- Machine Studies. Vol 29 (1988), pp. 257-285.

Webb, G.I. (1989) *Feature-Based Cognitive Diagnosis*, EXCALIBUR Tech. Rep. 14, Department of Computer Science, La Trobe University, Bundoora, Australia.

Webb, G. I. & Van der Klooster, R. (1989) *Inside the Unification Tutor*, EXCALIBUR Tech. Rep. 18, Department of Computer Science, La Trobe University Bundoora, Australia

Wenger, E (1987) *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann, Los Altos, CA.