Pre-publication draft of paper which appeared in the Proceedings of the First Machine Learning for User Modeling Workshop UM'97
Authors: B.C. Chiu and G.I. Webb

Using C4.5 as an Induction Engine for Agent Modelling: An Experiment of Optimisation

Patrick Chiu

chiu@deakin.edu.au
School of Computing and Mathematics, Deakin University, Australia
Fax: +61 3 52272028

Abstract: Input-Output Agent Modelling (IOAM) is an approach to modelling an agent in terms of relationships between the inputs and outputs of the cognitive system. This approach, together with one of the leading inductive learning algorithm, C4.5, has been adopted to build a C4.5-IOAM subtraction modeller, which aims to model students' competencies on elementary subtraction skills. Results showed that C4.5-IOAM could achieved reasonably high predictive power for this purpose. Very little attempt has been made for optimising the current system that improvement of its performance could be achieved by employing strategies and techniques for this purpose. This paper reports an experiment that studied how the system's performance could be improved with techniques of confining training examples and resolving conflicting predictions. Results show that these strategies improve the system's performance in the aspects of capturing more student errors and achieving higher prediction rate.

1 Introduction

The use of inductive learning for student modelling has been studied previously (for example, Desmoulins and Van Labeke, 1996; Gilmore and Self, 1988). Yet, it is still underrepresented. One of the reasons may be that a modelling system requires extra efforts for implementation if its inductive engine is tightly linked to the cognitive aspects of an agent. Input-Output Agent Modelling (IOAM) provides an approach that it allows a system treats the operation of the cognitive system as a black box and models an agent in terms of the relationships between the inputs and outputs of the system. By describing the capabilities rather than capturing internal mechanisms of the cognitive system, this approach reduces the system's inter-modular complexity and allows different inductive engines to plug-in for generating different languages to describe an agent. Previous IOAM approaches include Feature Based Modelling (FBM) (Webb and Kuzmycz, 1996), Relational Based Modelling (Kuzmycz, 1995), FFOIL-IOAM and C4.5-IOAM (Chiu et al., 1997). Among them, C4.5-IOAM uses C4.5 (Quinlan, 1993), a well-known and general-purpose learning algorithm, as its induction engine to model a student's competency of subtraction skills. Comparative evaluations of C4.5-IOAM against FBM (Geoff et al., 1997) and FFOIL-IOAM (Chiu et al., 1997) have shown that the use of C4.5 increased the number of predictions made without significantly altering the accuracy of those predictions. There was very little attempt has been made for optimising C4.5-IOAM that its prediction performance could be improved in the aspects of student error prediction and the overall prediction rate. This paper reports an experiment that studied how this objective could be achieved with the techniques of confining training examples and resolving conflicting predictions.

2 An overview of C4.5-IOAM

In the C4.5-IOAM subtraction modeller, the ways of manipulating context and action features of the problem domain were adopted from by FBM (Webb & Kuzmycz 1996). Context features describe the problems with which a student is faced while action features describe aspects of a student's actions for a particular problem. C4.5-IOAM manipulates a n-digit subtraction problem by treating it as n separate column problems. It uses eleven decision trees to build a student model. They correspond to the action features *Result=M-S*, *Result=M-S-1*, *Result=10+M-S*, *Result=10+M-S-1*, *Result=M*, *Result=S*, *Result=zero*, *Result=M-S-2*, *Result=10+M-S-2*, *Result=S-M* and *Result=correct*, where *M* and *S* stand for Minuend and Subtrahend respectively. Each decision tree can be regarded as a model of a student's action (behavior). The context features of a unit problem are described by 12 attributes. The first four attributes, M*_is_0*, *S_is_0*, *S_is_9* and *S_is_BK* (*BK* stands for blank), are self-explained, while the rest of them are listed below with their meanings where N stands for Not Available.

- *M_vs_S* : {G,L,E}, the Minuend is greater or less than, or equal to the Subtrahend.
- *M_L_is_0* : {T,F,N}, the Minuend digit in the column to the left is zero.
- *M_L_is_1* : {T,F,N}, the Minuend digit in the column to the left is one.
- *M_R_is_0* : {T,F,N}, the Minuend digit in the column to the right is zero.
- *S_R_is_9* : {T,F,N}, the Subtrahend digit in the column to the right is nine.
- *M_S_R* : {G,L,E,N}, similar to *M_vs_S*, but it describes the column to the right.
- *M_S_2R* : {G,L,E,N}, similar to *M_vs_S*, but it describes two columns to the right.
- *Column* : {L,I,R}, the current column is left-most, inner or right-most.

Figure 1 illustrates how training examples are formed for a data file which is used to build a corresponding decision tree.

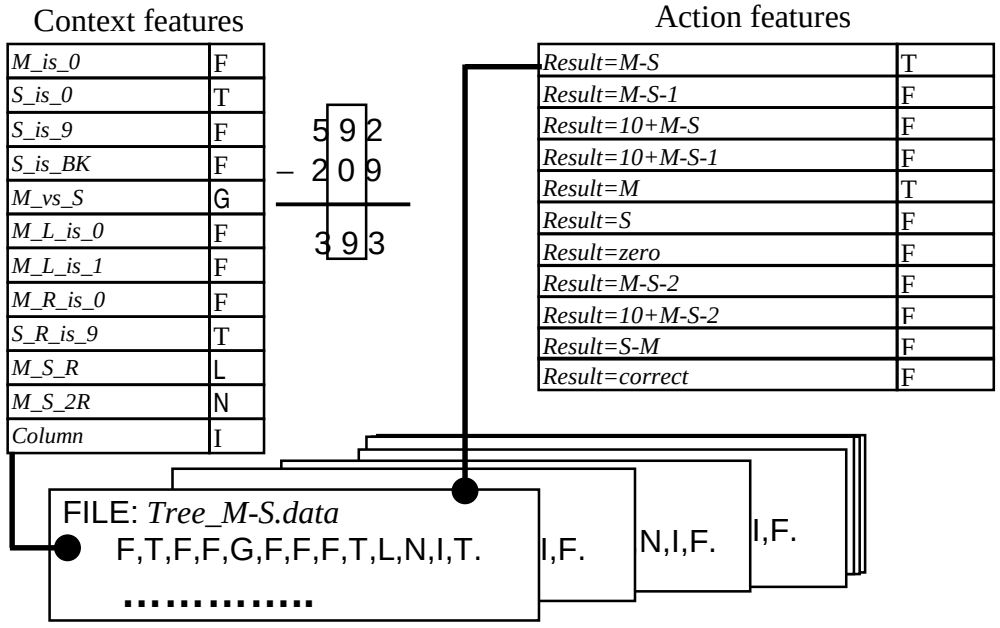

Figure 1. Formation of a column's training examples for decision trees.

Figure 2 shows a sample theory inferred by C4.5-IOAM. Decision trees with only one leaf labelled F predict the student will not exhibit the corresponding actions. *Tree_M* predicts that if the subtrahend is zero, the student will assign the minuend as the answer.

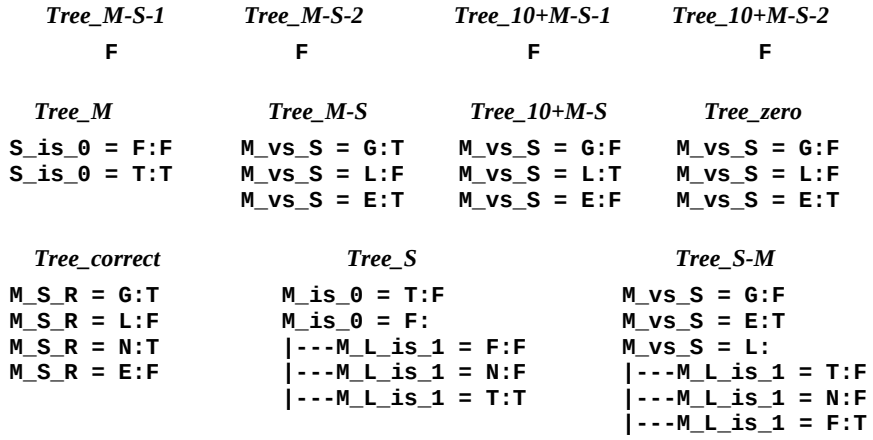

Figure 2. A theory inferred by C4.5-IOAM.

3 Techniques for improving prediction performance

The current version of C4.5-IOAM makes no prediction whenever there exist conflicting predictions made by decision trees. This reflects that there is still scope for the current system to be improved. This section will explore methods that aim to improve the prediction rate or prediction accuracy of the current system.

3.1 Confining training examples to erroneous answers

A decision tree which serves to predict a student's erroneous action should be trained by sufficient evidences of related examples. If the set of relevant examples is incomplete or erroneous cases share a very small proportion, a decision tree might predict no erroneous action. However, improvement in predicting students' errors could still be possible if some of the decision trees are specially trained for this purpose. Consider a unit problem m – 0 that a student gave a correct answer of m. This might not be a case to support that the student set the digit m as the answer; it was more likely to support that the student had exhibited an action: *Result=M-S*. On the other hand, if m was an incorrect answer, this case would be an evidence to support an erroneous action: *Result=M*. By confining training examples to erroneous answers only, the decision tree *Tree_M* would be more effective to capture a student's erroneous action. This treatment can be applied to decision trees *Tree_S*, *Tree_S-M* and *Tree_zero* for improving the system's performance in error predictions. However, it may cause some decision trees to be over reactive. They would tend to incorrectly predict that a student will exhibit erroneous actions. The risk of this prediction error could be reduced by introducing reliability measures for a prediction that will be discussed in the next two subsections.

3.2 Associating an estimated error rate to a decision tree

The prediction rate of the system can be improved by resolving conflicting predictions. Technique like voting is not suitable to the system because the decision trees predict different aspects of a student's actions. By associating decision trees with estimated error rates, and consulting them in a ranked order, the system could make more predictions without dropping the prediction accuracy significantly. We have considered employing stratified ten-fold cross-validation (Kohavi, 1995) for estimating the error rate of each decision tree. For each action feature, the training examples are equally divided into ten partitions. Each partition, which preserves the original class distribution, is used in turn as test data for the decision trees trained by the remaining nine partitions. The total numbers of correct and incorrect predictions of these tests are then used to estimate the error rate of the decision tree trained by the whole training set. The system's prediction is based on an action (a prediction T from a decision tree) which links to a digit. If a decision tree predicts no action, the system will consult the others in a preference order until a positive response is obtained or the error rate of the current decision tree exceeds a threshold limit.

3.3 Retrieving an error measure at a leaf node

The method of estimating error rate for each decision tree mentioned above provides an overall measure of prediction quality of a decision tree. The leaf node of a path of a decision tree may also provide an estimated error measure of a prediction. C4.5 takes the class of the majority at a leaf node as the leaf label and gives that label as prediction when a test example matches the decision path. The reliability of this prediction can be estimated by examining the distribution of the classes at the leaf node. Since we are only concerned with the leaf labeled as T for predicting that a student will exhibit a particular action, the proportion of examples with class label F at that leaf node could be used as the estimated error rate of a prediction. Whenever there exist conflicting predictions, the system might adopts the prediction of a decision tree which leaf node is associated with a lower estimated error rate.

4 Experiment

The same data set, which has been used to evaluate C4.5-IOAM and other IOAM based subtraction modellers (Geoff et al., 1997; Chiu et at., 1997), was used to evaluate the techniques mentioned in Section 3. The data came from 73 primary school students who were administered with five rounds of subtraction-problem tests. For each student, a modelling system used all data from prior rounds to build a student model and used the current round data to test the student model. The performance of the current version of C4.5-IOAM was used as a base line. The symbol +X was used to denote a version that was implemented by introducing a technique X to the current system. We used the keys LIMIT, TQTY and LQTY to represent limiting training examples to erroneous answers, associating estimated error rates to decision trees, and retrieving an error measure at a leaf node respectively. New versions that were created by implementing more than one technique were also evaluated. For example, the version +TQTY+LQTY makes predictions based on the information of the prediction qualities of the decision tree and the leaf node of a decision path. The system would adopt a prediction of a decision tree in which the leaf node bears lower error rate despite the decision tree is not the most preferable one.

4.1 Experimental results

Table 1 summarizes the prediction performance of four decision trees built with default (Full) training set with that trained by confining training examples to erroneous answers (Limit). As expected, these decision trees improved their performance (indicated in bold fonts) in error prediction, with the expense of poorly predicting students' correct actions.

Table 1. The performance of decision trees fed by full and bias training set.

	Tree_M		*Tree_S-M*		*Tree_S*		*Tree_zero*	
	Full	Limit	Full	Limit	Full	Limit	Full	Limit
Number of predictions of action that led to correct answer	3242	2045	3415	1939	804	1824	3600	2120
Number of predictions that were correct	2850	1063	3065	1143	377	523	2844	699
Accuracy	87.9%	52.0%	89.8%	58.9%	46.9%	28.7%	79.0%	33.0%
Number of predictions of action that led to incorrect answer	468	**544**	673	**720**	350	**415**	390	**432**
Number of predictions that were correct	343	**430**	536	**583**	269	**325**	232	**275**
Accuracy	73.3%	**79.0%**	79.6%	**81.0%**	76.9%	**78.3%**	59.5%	**63.7%**

The effects of implementing techniques to the current system described in Section 3 are shown in Table 2. The treatment of confining training examples to four decision trees increased the number of erroneous answer predictions while the overall prediction rate was lower because the system confronted more conflicting predictions. The introduction of quality measures for decision trees and leaf nodes exhibited its positive effect in resolving the problem of conflicting predictions. All new versions with this kind of treatment achieved higher prediction rates without significantly dropping the overall prediction accuracy.

Table 2. Performance of new versions created by three kinds of treatment.

	C4.5-IOAM	+LIMIT	+LQTY	+LQTY +LIMIT	+TQTY	+TQTY +LIMIT	+LQTY +TQTY	+LQTY +TQTY +LIMIT
Number of predictions made	28700	25911	30093	30208	30994	29635	30078	299924
Prediction rate	94%	85%	99%	99%	99%	97%	99%	98%
Number of predictions that were correct	26507	23945	27543	27599	27496	27261	27528	27501
Prediction accuracy	92%	92%	92%	91%	91%	92%	92%	92%
Number of error predictions made	1999	2106	2173	2416	2300	2201	2160	2001
Prediction rate	55%	58%	60%	67%	61%	61%	60%	55%
Number of error predictions that were correct	1347	1393	1426	1536	1463	1426	1417	1365
Prediction accuracy	67%	66%	66%	64%	64%	65%	66%	68%

5 Conclusions

We evaluated three techniques for improving the prediction performance of inductive based subtraction skill modelling systems. We have seen how different techniques can be usefully employed for this purpose. The method of confining training examples to erroneous answers enables some decision trees to capture more students' erroneous actions. It could be applied to situations where the proportion of student errors is low.

Techniques of employing quality measures on decision trees and leaf nodes in resolving conflicting predictions have been shown effective for this purpose. These two methods cover two aspects of resolving conflicts: adopting a decision from a point of view at global level and considering the judgement based on local experience. It is quite similar to consulting human experts. While an engineer might have good reputation of knowledge but lacks sufficient experience for a particular case, an ordinary people could have encountered numerous similar examples and could be an expert for that case.

C4.5-IOAM employs machine learning techniques for agent modelling. It uses decision trees to model different aspects of a student's response to a subtraction problem. Although the techniques explored in this paper help the system to improve the general prediction performance, these techniques do not learn how to resolve conflicting predictions. This suggests a possible exploration on applying inductive learning in resolving conflicting predictions. Stacked generalization (Wolpert, 1992), which is a method that learns the outputs of others inductive learners, could be one of the considerations. We will see whether this method works in future studies.

References

Chiu, B., Webb, G. I., and Kuzmycz, M. (1997). A comparison of First-order and Zeroth-order induction for Input-Oput Agent Modelling. Manuscript submitted for publication.

Desmoulins, C., and Van Labeke, N. (1996). Towards student modelling in geometry with inductive logic programming. In Brna, P., Paiva, A., and Self, J., eds., *Proceedings of the European Conference on Artificial Intelligence in Education*. Manuscript submitted for publication.

Gilmore, D., and Self, J. (1988). The application of machine learning to intelligent tutoring systems. In Self, J., ed., *Artificial Intelligence and Human Learning: Intelligent Computer-aided Instruction.* London: Chapman and Hall. 179-196.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of 14th International Joint Conference on Artificial Intelligence*, 1137-1143.

Kuzmycz, M. (1994). A dynamic vocabulary for student modelling. *Proceedings of the Fourth International Conference on User Modelling*, 185-190.

Webb, G. I., Chiu, B., and Kuzmycz, M. (1997). A comparative evaluation of the use of C4.5 and Feature Based Modelling as induction engines for Input/Output Agent Modelling. Manuscript submitted for publication.

Webb, G. I., and Kuzmycz, M. (1996). Feature Based Modelling: A methodology for producing coherent, dynamically changing models of agent's competencies. *User Modeling and User-Adapted Interaction* 5(2):117-150.

Wolpert, D. (1992). Stacked generalization. *Neural Networks* 5:241-259.