Predicting Correctness of Problem Solving from Low-level Log Data in Intelligent Tutoring Systems

Suleyman Cetintas[1], Luo Si[1], Yan Ping Xin[2] and Casey Hord[2]
[1]{scetinta, lsi}@cs.purdue.edu, [2]{xin, thord}@purdue.edu
[1] Department of Computer Sciences, [2]Department of Educational Studies
Purdue University, West Lafayette, IN, 47906, USA

Abstract. This paper proposes a learning based method that can automatically determine how likely a student is to give a correct answer to a problem in an intelligent tutoring system. Only log files that record students' actions with the system are used to train the model, therefore the modeling process doesn't require expert knowledge for identifying domain specific skills that are needed to solve the problem or students' possible solution methods etc. The model utilizes a set of performance features, problem features, time and mouse movement features and is compared to i) a model that utilizes performance and problem features, ii) a model that uses performance, problem and time features. In order to address data sparseness problem, a robust Ridge Regression algorithm is designed to estimate model parameters. An extensive set of experiment results demonstrate the power of using multiple types of evidence as well as the robust Ridge Regression algorithm.

1 Introduction

Increasing trend in computers' utilization for teaching has led to the development of many intelligent tutoring systems (ITS). It has been shown that ITSs improve students' learning by providing individualized guidance by means of modeling students' cognitive skills while they solve problems [6, 11]. This approach, i.e. building a detailed model of students' domain specific cognitive skills, and updating them accordingly as students proceed is known as "model-tracing" [1] and have been followed by several tutors such as ANDES [5], SlideTutor [7], PAT [10] etc. For instance, the ANDES system utilizes a Bayesian network representation which is constructed from potential solutions of a current problem and is updated after each student action to determine when a student may need help or what method may potentially be used by the student to solve the problem. Building models that require construction of domain specific skills, and following students' progress at the skill level is an accurate way of student modeling; however it is time-consuming and requires experts' knowledge of the domain.

Instead of explicitly modeling students' cognitive skills with the help of a domain expert, it is possible to utilize the detailed low-level log data that ITS collect during the student-tutor interaction, to help the decision making process of a tutor. Robinet et al. have recently proposed a method to discover high-level student behaviors from low-level traces of students in a problem solving environment [14]. Their system uses domain dependent context-action-outcome (CAO) triplets extracted from the low-level log data and clusters them into high-level abilities (HLA) that can later be used for generating or selecting sets of exercises. Although their CAO triplets are domain dependent, their method is at a higher level than the level of the student resolution. Beck et al. note that it is not always obvious how to map the low level cognitive information to higher level teaching actions and propose a machine learning agent that directly learns to predict the probability of whether the student's response will be correct and how long it will take to generate that response [3]. They also note that it's possible to explicitly model how students generate their answers but is time-consuming. Instead, they use 4 groups of model inputs describing the current state of a student (using student, topic, problem, and context related features) and use linear regression to produce their two outcomes: i) whether the student will be able to solve the problem correctly, ii) how much time it will take him to solve it. They claim that any machine learning method doing function approximation will (in theory) work and what is important is the model inputs and outputs.

To the best of our knowledge, there is no prior research on the automatic detection of whether a student will be able to correctly answer a question with a high-level student model (i.e. without using any expert knowledge of the domain), utilizing mouse movement data. Prior work mainly focuses on the model-tracing approach [5, 7, 10] which is a quite different task than high-level student modeling. De Vicente and Pain have human participants use mouse movement data as well as data from student-tutor interaction for motivation diagnosis [8]. In a recent work, Cetintas et al. [4] also utilize mouse movement data along with performance and time features to automatically detect the off-task behaviors of students while they work with the tutoring system. However both of these works focus on tasks that are also very different than predicting the correctness of problem solving via a high-level student model. The works that focus on high-level student modeling utilize only combinations of performance, problem, context, topic, action, outcome related features (that can all be extracted from the low-level logs of user-system interaction) but ignore mouse movement data [3, 14]. Although these features are quite important to improve the effectiveness of student modeling, mouse movement data is another important type of data that can also be incorporated into high-level student modeling to improve the prediction accuracy. Similar to all other features that have been mentioned so far, mouse movement data can also easily be stored and retrieved from low-level user-system logs in every intelligent tutoring system.

This paper proposes a machine learning method that can automatically predict whether a student will be able to correctly answer a problem in a problem solving environment by utilizing multiple types of evidence including performance, problem, time and mouse movement features that are extracted from the log files of students' actions within tutoring software. To address data sparseness problem, the proposed model utilizes a robust Ridge Regression technique to estimate model parameters. The proposed model is compared to i) a model that utilizes performance and problem features; ii) a model that uses performance, problem and time features and iii) all models when data sparseness problem is not addressed (i.e. when model parameters are not learned with Ridge Regression). We show that utilizing multiple types of evidence as well as the robust Ridge Regression technique improves the effectiveness of the student model.

2 Data

Data from a study conducted in 2008 in an elementary school was used in this work. The study was conducted in mathematics classrooms using a math tutoring software (that has

Table 1. Details of the problem solving worksheets. The information of whether problems of a worksheet include i) diagram boxes can be seen under the *Include Diagram Boxes* column; ii) equation boxes can be seen under the *Include Equation Boxes* column; iii) unknown number to be solved for can be seen under the *Include Unknown Number* column. *Shows Correct Answer* column shows whether the correct answer to a question is shown to students after they submit their answer.

Worksheet	Includes Diagram Boxes	Includes Equation Boxes	Includes Unknown Number	Shows Correct Answer
EG/MC Worksheet 1	Yes	No	No	Yes
EG/MC Worksheet 2	Yes	Yes	Yes	Yes
EG/MC Worksheet 3	No	Yes	Yes	Yes
EG/MC Worksheet 4	No	Yes	Yes	No
Mixed Worksheet 1	No	Yes	Yes	Yes
Mixed Worksheet 2	No	Yes	Yes	No
Mixed Worksheet 3	No	Yes	Yes	No

Table 2. Details of the problems collected out of the problem solving sessions. Number of correctly solved problems for training, text splits can be seen under the *# of Correctly Solved Problems* column; number of incorrectly solved problems can be seen under the *# of Incorrectly Solved Problems* column and the total number of problems for training and test splits can be seen under the *Total* column. The percentages in the parenthesis indicate the ratio of positive and negative data for training and test splits as well as the total.

	# of Correctly Solved Problems	# of Incorrectly Solved Problems	Total
Training	713 (65.3%)	379 (34.7%)	1092
Test	578 (66.6%)	290 (33.4%)	868
Total	1291 (65.8%)	669 (34.1%)	1960

been developed by the authors). The tutoring software teaches problem solving skills for Equal Group (EG) and Multiplicative Compare (MC) problems. These two problem types are a subset of the most important mathematical word problem types that represent about 78% of the problems in a fourth grade mathematics textbook [12]. In the tutoring system; first, a conceptual instruction session is studied by a student followed by problem solving sections to test their understanding. Both of conceptual instruction and problem solving parts require students to work one-on-one with the tutoring software and if students fail to pass a problem solving session, they have to repeat the corresponding conceptual instruction and the problem solving session. Space limitations preclude discussing in detail but the tutoring software has a total of 4 conceptual instruction sessions and 11 problem solving worksheets that have 12 questions each (4 for Equal Group worksheets, 4 for Multiplicative Compare worksheets, 3 Mixed worksheets each of which include 6 EG & 6 MC problems) and is supported with animations, audio (with more than 500 audio files), instructional hints, exercises etc.

In a problem solving worksheet, a problem is counted as correctly solved only if all question boxes for the problem are filled correctly. Question boxes for a problem check students' ability to find the correct solution of the problem as well as to fulfill some partial skills that are needed for the solution of a problem when they can't give a full answer. Such partial skills for a problem include answers to i) diagram boxes which check student's mapping of the information given in a problem into an abstract model; ii) equation box which checks whether a student can form a correct equation from the information given in a problem; iii) final answer box which checks whether a student can solve the asked unknown in a problem correctly. Details about which worksheets include which groups of boxes are shown in Table 1. In some of the worksheets, after the student answers the question, the correct answer is also shown to the student after each question in the worksheet (regardless of whether the student's answer is correct or not) to make the student learn from his/her mistake. Details about which worksheets show correct answers are also shown in Table 1.

The study with the tutoring system included 8 students which include 3 students with learning disabilities, 1 student with emotional disorder and 1 student with emotional disorder combined with mental retardation. Students used the tutor for several 30 minute class sessions (on average 18.1255 sessions per student with standard deviation of 3.4408 sessions) during which their interaction with the tutoring system was logged in a centralized database. A total of 1960 problems that were solved, 1291 of which were correctly solved and 669 of which were incorrectly solved. The average number of correctly solved problems per student is 161.37 (with a standard deviation of 42.07) and the average number of incorrectly solved problems per student is 83.62 (with a standard deviation of 27.07). Data from 4 students were used as training data to build the models for making predictions for the other 4 students (who are used as the test data). Details about the training and test splits are given in Table 2.

3 Methods: Least Squares and Ridge Regression

Data sparseness is an important problem which is caused by using limited training data to learn parameters of a model and leads to the common problem of *over-fitting* [9]. Over-fitting as the name implies is the problem of having an excellent fit to the training data which may not be a precise indicator of future test data especially in the case of data sparseness. Regularization is a technique to control the over-fitting problem by setting constraints on model parameters in order to discourage them from reaching large values that lead to over-fitting. We will briefly discuss the Least Squares technique followed by Ridge Regression technique that controls over-fitting [9].

The simplest linear model for regression involves a linear combination of input variables as follows:

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D = \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x} \qquad (1)$$

where $\boldsymbol{x} = (1, x_1, \ldots, x_D)^T$ is an instance of training data of D+1 dimensions and $\boldsymbol{w} = (w_0, w_1, \ldots, w_D)^T$ are model coefficients (w_0 is the bias parameter). For such a model, the sum of squares error between predictions $y(\boldsymbol{x}_n, \boldsymbol{w})$ for each data point $\boldsymbol{x}_n$ and the corresponding target values t_n is as follows:

$$E_D(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} \{t_n - y(\boldsymbol{x}_n, \boldsymbol{w})\}^2 = \frac{1}{2}\sum_{n=1}^{N} \{t_n - \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}_n\}^2 \qquad (2)$$

which can be minimized with a maximum likelihood solution that gives *the Least Squares* solution of the model parameters as follows:

$$\boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi t} \tag{3}$$

where $\boldsymbol{\Phi}$ is an N*D design matrix whose elements are given by $\boldsymbol{\Phi_{nj}} = \boldsymbol{x_{nj}}$ (i.e. jth dimension of the nth training instance). Ridge Regression adds a quadratic regularization penalty of $E_W(\boldsymbol{w}) = \boldsymbol{1/2}\,\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$ to the data-dependent error (i.e. Eq. (2)) with which the total error function becomes:

$$E_{TOTAL}(\boldsymbol{w}) = E_D(\boldsymbol{w}) + \lambda E_W(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \boldsymbol{w}^{\boldsymbol{T}}x_n\}^2 + \frac{\lambda}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} \tag{4}$$

where λ is the regularization coefficient that controls the relative importance of data-dependent error $E_D(\boldsymbol{w})$ and the regularization term $E_W(\boldsymbol{w})$. The regularization coefficient in this work is learned with cross validation in the training phase (i.e. splitting the training data into smaller training and test datasets). The exact minimizer of the total error function can be found in closed form as follows:

$$\boldsymbol{w}_{RIDGE} = (\lambda\boldsymbol{I} + \boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi t} \tag{5}$$

which is *the Ridge Regression* solution of the parameters of the model.

4 Modeling Approaches

This section describes the models that are used for evaluation: i) a model that considers performance and problem features, ii) another modeling approach that considers time features as well as performance and problem features, and finally iii) a more advanced model that incorporates mouse movement features with performance, problem and time related features.

4.1 Performance and Problem Based Modeling (PerfProb_Mod)

Using performance and problem based features has been shown to be a useful approach for student modeling in the prior work [3, 14]. The idea of using performance features is quite intuitive since students' performance up to a certain problem is a good indicator of their performance for that problem. Similarly problem related features such as problem difficulty or number of sub skills (types of question boxes in this work) required etc., are very informative to see whether a current student can correctly answer a problem or not.

In this work; 4 performance features are used. The set of 4 performance features are used as a measure of the probability that the student knew the skills asked in a question. The first feature is the *# of correct answers so far* in a problem solving worksheet. Each problem solving worksheet consists of 12 math word problems and a problem is counted as correct only if all question boxes for the problem are filled correctly. The number of correctly solved problems up to a current problem in a worksheet is a good indicator for student's success for the current problem. Second, third and fourth performance features help to assess student's partial skills that are needed for the solution of a problem when they can't give a full answer. Such partial skills for a problem include the abilities to give answers to, as mentioned before, i) diagram boxes which check student's mapping of the information given in a problem into an abstract model; ii) equation box which checks whether a student can form a correct equation from the information given in a problem; iii) final answer box which checks whether a student can solve the asked unknown in a problem correctly. The corresponding features are *percentage of correct diagram answers so far*, *percentage of correct equation box answers so far, percentage of final answers so far* in a problem solving worksheet. They provide the percentage of correct answers given by a student for the associated partial skill boxes of all the solved problems of a current worksheet up to the current problem.

In addition to the 4 performance features, 11 problem features are also used indicating which problem solving worksheet the current problem belongs to. In our model, there are 11 binary variables corresponding to 11 worksheets. If a current problem belongs to 5th worksheet (i.e. MC Worksheet 1), then 5th binary variable will be 1 and all others will be 0. This encoding approach enables the model to associate each problem with the different characteristics of different worksheets. This encoding scheme is also mentioned in Beck's work as "one hot" encoding [3].

Performance and problem based modeling in this work serves as the baseline for all other models and will be referred as PerfProb_Mod.

4.2 Performance, Problem and Time Based Modeling (PerfProbTime_Mod)

Performance and problem based modeling approach is useful in many situations however there are lots of other possible data that can be good indicators of students' success for a current problem such as the time that a student spends while solving a problem. Although not all the prior work used time related features [14], it has been used as a feature by Beck [3].

In addition to the 15 performance and problem features mentioned before, this modeling approach also incorporates the time feature for student modeling. The time feature in this work is defined as the time a student spends while solving a problem.

Performance, problem and time based modeling approach will be referred as PerfProbTime_Mod.

4.3 Performance, Problem, Time and Mouse Tracking Based Modeling (PerfProbTimeMouseT_Mod)

Incorporation of the time feature into the performance and problem based modeling is an effective way of improving student modeling; however there is still more room to improve. Both performance & problem based modeling and performance, problem & time based modeling approaches ignore an important data source with which students are almost always in interaction while they are solving problems in a problem solving environment: the mouse. As far as we know there is no prior research on student modeling that utilize mouse tracking data. More details about the prior work on this

Table 3. Results of PerfProbTimeMouseT_Mod method is shown in comparison to PerfProb_Mod and PerfProbTime_Mod methods for high level student modeling to detect whether a student can correctly solve a given problem. Note that the results for each model for the technique of least squares are shown under the *Least Squares* column, and the results for each model for the technique of Ridge Regression are shown under the *Ridge* Regression column. The percentages in the parenthesis show the relative improvements of each method with respect to the Least Squares version of the PerfProb_Mod model. The performance is evaluated by the F_1 measure.

Methods	Technique	
	Least Squares	Ridge Regression
PerfProb_Mod	0.4632	0.6749 (+45.70%)
PerfProbTime_Mod	0.5090 (+09.89%)	0.6805 (+46.91%)
PerfProbTimeMouseT_Mod	0.5249 (+13.32%)	0.6894 (+48.85%)

modeling approach as well as utilizing mouse movement data can be found in the Introduction section.

In addition to the 4 performance related features, 11 problem related features and 1 time feature that have been mentioned; this modeling approach incorporates 3 more features as mouse tracking data. The first feature is the *maximum mouse off time* in a problem which provides the knowledge of the biggest time interval (in seconds) in which mouse is not used for a current problem. Second and third mouse tracking features are the *average x movement* and *average y movement* respectively. They basically assess average number of pixels the mouse is moved along the x and y axes in 0.2 second intervals.

Performance, problem, time and mouse tracking based modeling that we propose will be referred as PerfProbTimeMouseT_Mod.

5 Experimental Methodology: Evaluation Metric

To evaluate the effectiveness of the off-task behavior detection task, we use the common F_1 measure, which is the harmonic mean of precision and recall [2,13]. Precision (p) is the ratio of the correct categorizations by a model divided by all the categorizations of that model. Recall (r) is the ratio of correct categorizations by a model divided by the total number of correct categorizations.

$$F_1 = \frac{2pr}{p+r} \tag{6}$$

6 Experiment Results

This section presents the experimental results of the methods that are presented in Methods section. All the methods were evaluated on the dataset described in Data section.

An extensive set of experiments are conducted to address the following questions:

- How effective is the PerfProbTime_Mod method that utilizes performance, problem and time features with respect to PerfProb_Mod method that utilizes performance and problem features?
- How effective is the PerfProbTimeMouseT_Mod method that utilizes mouse tracking data as well as performance, problem and time features with respect to PerfProb_Mod and PerfProbTime_Mod methods?
- How effective is the approach of utilizing the Ridge Regression technique to estimate the model parameters?

6.1 The Performance of Performance, Problem and Time Based Modeling (PerfProbTime_Mod)

The first set of experiments was conducted to measure the effect of including the time feature in the PerfProb_Mod model. The details about this approach are given in detail in Section 4.1.

More specifically, PerfProbTime_Mod model is compared with PerfProb_Mod and their performances are shown in comparison to each other in Table 3. It can be seen that the PerfProbTime_Mod model outperforms PerfProb_Mod model. The lesson to learn from this set of experiments is that time feature is helpful when it is combined with performance and problem related features for the task of predicting whether a student will be able to correctly answer a current problem. This explicitly demonstrates the power of incorporating the time feature into the performance and problem related based modeling.

6.2 The Performance of Performance, Problem, Time and Mouse Tracking Based Modeling (PerfProbTimeMouseT_Mod)

The second set of experiments was conducted to measure the effect of including the mouse tracking data in the PerfProbTime_Mod model. The details about this approach are given in detail in Section 4.2.

More specifically, PerfProbTimeMouseT_Mod method is compared with the other two models and its performance is shown in comparison to the other two models in Table 3. It can be seen that the PerfProbTimeMouseT_Mod model outperforms both PerfProbTime_Mod and PerfProb_Mod models. This set of experiments show that mouse movement features are helpful when they are combined with performance and problem related features along with the time feature for high level student modeling. This explicitly demonstrates the power of incorporating the mouse tracking features into performance, problem and time based modeling.

6.3 The Performance of Utilizing the Robust Ridge Regression Technique

The last set of experiments was conducted to measure the effect of utilizing the technique of Ridge Regression for learning the model parameters for each of the models. The details about this approach are given in detail in Section 3.

More specifically, Ridge Regression learned models are compared to Least Squares learned models. The performance of Ridge Regression versions of each model is shown in comparison to Least Squares versions in Table 3. It can be seen that the Ridge Regression version of each model outperforms Least Squares versions with its regularization framework. This confirms that Ridge Regression models better solve the data sparseness problems in this application.

7 Conclusion and Future Work

This paper proposes a novel machine learning method for high-level student modeling (that doesn't require any expert knowledge of the domain to extract skills, or possible solutions that students may follow) to detect if a student can correctly solve a current problem in a problem solving environment while using an intelligent tutoring system. This model relies only on the low-level log data that is available from the log files from students' actions within the software. The proposed model makes use of a set of evidence such as performance, problem, time and mouse movement features and is compared to i) a model that utilizes performance and problem related features, ii) a model that uses performance, problem and time features together. To address data sparseness problem, the proposed model utilizes a robust Ridge Regression technique to estimate model parameters.

An extensive set of empirical results show that the proposed method that automatically detects whether a student will be able to correctly answer a problem substantially outperforms the model that uses performance and problem related features as well as the model that utilizes performance, problem and time features together. Furthermore empirical results show that the proposed model attains a better performance by utilizing the technique of Ridge Regression over the standard Least Squares Regression technique.

There are several possibilities to extend the research. For example, different students have different types of characteristics for solving problems (e.g. using more or less time to solve the problems; having difficulties with particular types of questions and/or problems or different mouse usage types etc.). Therefore, personalized models tend to provide more accurate detection results than a single model for all students. Future research work will be conducted mainly in this direction.

Acknowledgements

This research was partially supported by the NSF grants IIS-0749462 and IIS-0746830. Any opinions, findings, conclusions, or recommendations expressed in this paper are the authors', and do not necessarily reflect those of the sponsor.

References

[1] Anderson, J.R., Boyle, C.F., Corbett, A., Lewis, M. Cognitive Modeling and Intelligent Tutoring. *Artificial Intelligence*, 1990, 42, p. 7-49.

[2] Baeza-Yates, R. & Ribeiro-Neto, B. Modern Information Retrieval. *Addison Wesley*, 1999.

[3] Beck, J.E., Woolf, B.P. High Level Student Modeling with Machine Learning. *Proceedings of the Intelligent Tutoring Systems Conference,* 2000, p. 584-593.

[4] Cetintas, S., Si, L., Xin, Y. P., Hord, C. & Zhang, D. Learning to Identify Students' Off-task Behavior in Intelligent Tutoring Systems. *Proceedings of the 14th International Conference on Artificial Intelligence in Education,* 2009, to appear.

[5] Conati, C., Gertner, A.S., VanLehn, K. Using Bayesian Networks to Manage Uncertainty in Student Modeling. *User Model, User Adapt, Interact,* 2002, 12(4), p. 371-417.

[6] Corbett, A.T. and Anderson, J.R. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User Adapted Interaction,* 1995, 4, p. 253-278.

[7] Crowley, R., Medvedeva, O. and Jukic, D. SlideTutor: A model-tracing Intelligent Tutoring System for teaching microscopic diagnosis. *Proceedings of the 11th International Conference on Artificial Intelligence in Education,* 2003.

[8] De Vicente, A. and Pain, H. Informing the detection of the students' motivational state: an empirical study. *Proceedings of the Intelligent Tutoring Systems Conference,* 2002, p. 933-943.

[9] Hastie, T., Tibshirani, R. & Friedman, J. The Elements of Statistical Learning. *Springer,* 2001.

[10] Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A. Intelligent Tutoring Goes to School in the Big City. *Intelligent Journal of Artificial Intelligence in Education,* 1997, 8.

[11] Koedinger, K.R., Corbett, A.T., Ritter, S., Shapiro, L.J. Carnegie Learning's Cognitive Tutor: Summary of Research Results. *White Paper, Carnegie Learning,* 2000, Pittsburg, PA.

[12] Maletsky, E. M. et al. Harcourt Math, *Indiana Edition*, *2004,* Chicago: Harcourt.

[13] Rijsbergen, C. J. v. Information Retrieval*, 2nd edition,* 1979*,* University of Glasgow.

[14] Robinet, V., Bisson, G., Gordon, M. B., Lemaire, B. Inducing High-Level Behaviors from Problem-Solving Traces Using Machine-Learning Tools. *IEEE Intelligent Systems,* 2007, 22, 4, p. 22-30.