Using HMMs and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset

Zachary A. Pardos ZPARDOS@WPI.EDU
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Rd. #3213
Worcester, MA 01609

Neil T. Heffernan NTH@WPI.EDU
Academic Adviser
Worcester Polytechnic Institute

Abstract

This article describes the user modeling, feature extraction and bagged decision tree methods that were used to win 2nd place student prize and 4th place overall in the ACM's 2010 KDD Cup.
Keywords: User modeling, Bayesian networks, Random forests, EDM, KDD Cup

1 Introduction

The datasets for the 2010 KDD Cup came from Intelligent Tutoring Systems (ITS) used by thousands of students over the course of the 2008-2009 school year. This was the first time the ACM used an educational data set for the competition and also marked the largest dataset the competition has hosted thus far. There were 30 million training rows and 1.2 million test rows in total occupying over 9 gigabytes on disk. The competition consisted of two datasets from two different algebra tutors. One came from the Carnegie Learning Algebra system; this dataset was simply called "Algebra". The other came from the Bridge to Algebra system whose dataset was aptly called "Bridge to Algebra". The task was to predict if a student answered a given math step correctly or incorrectly. Predictions between 0 and 1 were allowed and were scored based on RMSE. In addition to the two challenge datasets, three datasets were released prior to the start of the official competition. Two datasets were from the two previous years of the Carnegie Learning Algebra tutor and one was from the previous year of the Bridge to Algebra tutor. These datasets were referred to as the development datasets. Full test labels were given for these datasets so that competitors could familiarize themselves with the data and test various prediction strategies before the official competition began. These datasets were also considerably smaller, roughly 1/5th the size of the competition datasets. A few anomalies in the 2007-2008 Algebra dataset were announced early on, so that dataset will not be analyzed in this article.

1.1 Summary of methods used in the final prediction

The final prediction was a combination of Bayesian Hidden Markov Models (HMMs) and bagged decision trees. The HMM used was a novel Bayesian model developed based on work by Pardos & Heffernan (2010) that predicts the probability of knowledge for each student at each opportunity as well as a prediction of probability of correctness on each step. The model learns individualized student specific parameters (learn rate, guess and slip) and then uses these parameters to train skill specific models. The resulting model that considers the composition of user and skill parameters outperforms models that only take into account parameters of the skill. The Bayesian model was used in a variant of ensemble selection (Caruana and Niculescu-Mizil,

2004) and also to generate extra features for the decision tree classifier. The bagged decision tree classifier was the primary classifier used and was developed by Leo Breiman (2001).

1.2 The Anatomy of the Tutor

While the two datasets came from different tutors, the format of the datasets and underlying structure of the tutors was the same. A typical use of the system would be as follows; students would start a math curriculum determined by their teacher. The student would answer math question on the tutor and would be given tutorial help if they answered incorrectly. The tutorial help would often involve asking a series of additional questions that broke the problem into sub steps. A student could also request a hint but requesting a hint would mark the student as getting the step wrong in the system.

The largest curriculum component in the tutor is a unit. Units contain sections and sections contain problems. Problems are the math questions that the student tries to answer and can consist of multiple steps. Each row in the dataset represented a student’s answer to a single step in a problem. Determining whether or not a student answers a problem step correctly on the first attempt was the prediction task of the competition.

Students’ advancing through the tutor is based on their mastery of the skills involved in the pedagogical unit they are working on. If students do not master all the skills in a unit, they cannot proceed on their own, however, a teacher may intervene and skip them ahead.

1.3 Missing data in the test sets

Seven columns in the training sets were intentionally omitted from the test sets. These columns either involved time, such as timestamp and step duration or information about performance on the question, such as hints requested or number of incorrect attempts at answering the step.

In the development datasets, the chronology of the steps in the test rows with respect to the training rows could be determined by the row ID column, however, in the challenge set the row ID of the test rows was reset to 1. The test row chronology was inferred based on the unit in which the student answered problem steps in. A student’s rows for a given unit in the test set were assumed to come directly after their rows for that unit in the training set. While there may have been exceptions, this was a safe assumption to make given the organizers description of how the test rows were selected which was that an arbitrary problem was selected from each of the units a student completed. The steps in that problem became the test rows and all steps prior to those became part of the training set.

2 Pre-processing

The Algebra dataset contained 24 columns while the Bridge to Algebra dataset contained 20. The first step to being able to work with the dataset was to convert the ASCII text fields of the columns into strictly numeric values. This was done using perl to hash text values such as anonymized usernames and skill names into integer values. The timestamp field was converted to epoc and the problem hierarchy field was parsed into separate unit and section values.

Special attention was given to the step duration column that describes how long the student spent answering the step. This column had a high percentage of null and zero values making it very noisy. For the rows in which the step during value was null or zero, a replacement to the step duration value was calculated as the time elapsed between the current row’s timestamp and the next row’s timestamp for that same user. Outlier values for this recalculated step time were possible since the next row could be another day that the student used the system. It was also the case that row ID ordering did not strictly coincide with timestamp ordering so negative step duration values occurred periodically. Whenever a negative value or value greater than 1,000 seconds was encountered, the default step duration value of null or zero was kept. The step duration field was used for feature generation described in the Random forests section.

2.1 Knowledge Component columns in the dataset

The Knowledge Component (KC) columns in the dataset described the skill or skills involved in the row's problem step. Different KC columns used a different group of skills to describe a problem step. The KCs are used in Cognitive Tutoring to track student learning over the course of the curriculum. A KC skill tagging that more accurately represents the student's knowledge at that time will also more accurately predict future performance. Because of this it was important to explore which KC columns most accurately fit the data for each dataset.

2.1.1 Rows of data where a KC column had no value

There were a large percentage of rows (~20-25%) in both the training and test sets in which one or more KC columns had no value. That is, no skill was associated with the problem step. The Bayesian model needs skill associations to predict performance so this issue needed to be addressed. The solution was to treat null KC values as a separate skill with ID 1, called the NULL skill. A skill that appears in a separate unit is considered a separate skill so there were as many null ID skills as there were units. These null skill steps were predicted with relatively low error (RMSE ~0.20). In personal communication with Carnegie Learning staff after the competition, it was suggested that the majority of the null steps were most likely non math related steps such as closing an application window or clicking a button.

2.1.2 Handling of KC values with multiple skills

There can be one or more skills associated with a step for any of the KC columns. Modeling multiple skills with Knowledge Tracing is significantly more complex and is not currently done within the Cognitive Tutor. To avoid having to model multiple skills per step, the KC values with multiple skills were collapsed into one skill. Two ways of collapsing the values were tried, creating two separate versions of the KC columns. The first way was to keep only the most difficult skill. This approach is based on the hypothesis that skills compose conjunctively in an ITS. Difficulty was calculated based on percent correct of all rows in the training set containing that skill. The second way of collapsing multiple skill values was to treat a unique set of skills as a completely separate skill. The result of this processing was the generation of two skill models for each KC column for each challenge set. For the development datasets, only the "treat a set of skills as a separate skill" strategy was used.

2.1.3 Creating internal testing and training sets

The competition's test sets represent student responses that chronologically directly follow the responses in the training set for each student per unit. We could have arbitrarily split the training set allowing for a 2-fold cross validation, however arbitrary selecting rows for internal training and testing would not respect the temporal aspect of the data. Instead, the internal test set was created by taking the last problem responses per student per unit in the training set. The internal training set then became what was left after removing those last responses. The intuition behind this organization was that the internal predictions would be more accurate at forecasting challenge test set predictions if the internal training and test sets were symmetric to the challenge training and test sets. This method also allowed for features relating to recent student performance to be appended to the test sets. In order to test the effectiveness of ensemble selection and of the Random forest classification, without going to the public leader board, a second internal test and training set was created from the first internal training set[1]. While the training sets were used to train the Bayesian models, the test sets, using feature extraction, were good enough samples of the dataset to serve as highly accurate training sets for the challenge test set.

[1] In comments about the method in submissions to the leaderboard, the names ECO and PRE were used to referred to the two internal Bayesian test sets that served as training sets for the Random forests.

3 Bayesian Networks Approach

Bayesian Networks were used to model student knowledge over time. A simple HMM with one hidden node and one observed node has been the standard for tracking student knowledge in ITS and was introduced to the domain by Corbett and Anderson (1995). In this model, known as knowledge tracing, a student's incorrect and correct responses to questions of a particular skill are tracked. Based on the parameters of the HMM for that skill and the student's past responses, a probability of knowledge is inferred. In the Cognitive Tutor, students who know a skill with 95% probability, according to the HMM, are considered to have mastered that skill. There are four parameters of the HMM and they are typically learned from the data using Expectation Maximization (EM). A max iteration of 100 and was used for EM parameter learning. EM will also stop if the log likelihood fit to the data increases by less than 1e-5 between iterations.

3.1 The Prior Per Student Model (Simple Model)

Standard knowledge tracing has four parameters that are learned per skill. The parameters capture the concepts of learning rate, prior knowledge, guess rate and slip rate. Work by Pardos and Heffernan (2010) has shown that specifying a separate prior per student in the training and predictions steps can increase the accuracy of the learned parameters and of the prediction accuracy. In that work, simulated datasets created from a known distribution were analyzed by the standard knowledge tracing model and by one that allowed for a prior per student based on the student's first response. The prior per student model resulted in more accurate predictions as well as convergence to the ground truth parameter values regardless of initial parameter values for EM parameter learning. The standard knowledge tracing model, however, was very sensitive to initial parameter values in converging to the ground truth parameters.

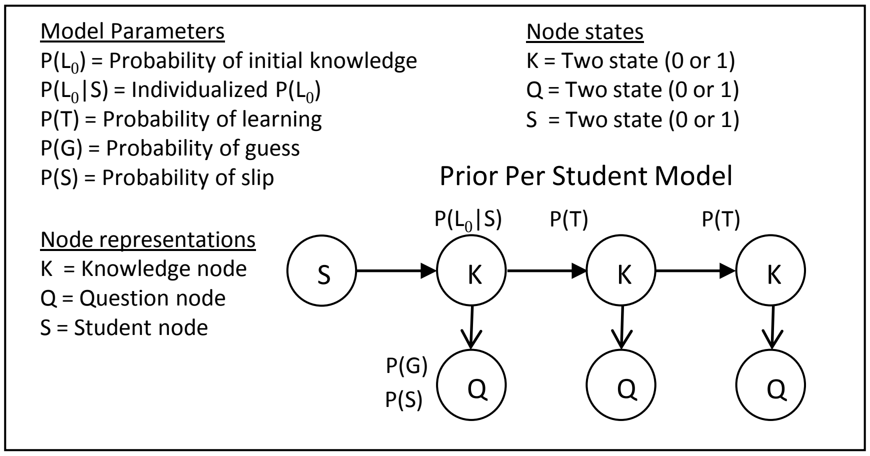


Figure 1. Simple HMM: Prior per student model

Figure 1 describes the Prior Per Student (PPS) model. In this model the student node can take on any value from 1 to N where N is the number of students, however, we have found that using a student's first response to seed the prior is an effective heuristic. We refer to this as the cold start heuristic. If a student answers the first observed step incorrectly, they are assigned a prior of 0.10, if they answer the step correctly; they are assigned a prior of 0.85. These values were chosen *ad-hoc* based on experimentation with this and other datasets. One alternative to the *ad-hoc* setting is to let the two prior seeding values be adjusted and learned from data. These values may be capturing guess and slip probabilities so another alternative is to have the prior seeding values be the same as the guess and slip values. We tested these three strategies with the two development datasets and found the following results, shown in Table 1.

Algebra (development)

	Strategy	**RMSE**
1	adjustable	0.3659
2	guess/slip	0.3660
3	*Ad-hoc*	0.3662

Bridge to Algebra (development)

	Strategy	**RMSE**
1	guess/slip	0.3227
2	adjustable	0.3228
3	*Ad-hoc*	0.3236

Table 1. Results of prior seeding strategies on the two development datasets

Table 1 shows that for the algebra (development) datasets, the difference between the *ad-hoc* and adjustable strategy was 0.0003. This appeared to be a small benefit at the time and the extra free parameters of the adjustable strategy added to the compute time of the EM runs. While the guess/slip strategy added less compute time than the adjustable strategy, the *ad-hoc* value strategy was chosen to be used going forward with all models used for the competition datasets because of the small difference in RMSE and because this strategy had already been more carefully studied in past work (Pardos & Heffernan, 2010). Another reason *Ad-hoc* was chosen is because it appeared to be the best strategy in the bridge to algebra dataset when initially calculated. Upon closer inspection for this article, the *Ad-hoc* prediction was short around 250 rows compared to the other strategy predictions. After correcting this, the guess/slip strategy appears favorable.

3.1.1 Limiting the number of student responses used

The EM training for skills with high amounts of student responses would take up over 8gigs of memory on the compute machines. This was too much as the machines used to run these models had only 8 gigs and reaching into swap memory caused the job to take consderiably longer to finish. The skills with high amounts of data often had over 400 responses by one student. To alleviate the memory strain, limits were placed on the number of most recent responses that would be used in training and prediction. The limits tested were 5, 10, 25, 150 and none.

Algebra (development)

	Limit	**RMSE**
1	25	0.3673
2	150	0.3675
3	none	0.3678
4	10	0.3687
5	5	0.3730

Bridge to Algebra (development)

	Limit	**RMSE**
1	10	0.3220
2	25	0.3236
3	5	0.3239
4	none	0.3252
5	150	0.3264

Table 2. Results of limiting the number of most recent student responses used

Table 2 shows the prediction RMSE on the development sets when limiting the number of most recent student responses used for training and prediction. A surprising result was that very few responses were needed to achieve the same or better results as using all data. In the algebra (development) set, 25 was the best limit of the limits tried and was the second best limit in the bridge to algebra (development) set. This prediction improvement was a welcomed bonus to eliminating the memory issue which would have been compounded when working with the much larger competition set. A limit of 25 would be used for all subsequent models.

3.1.2 Distribution of skill parameters

Using the model in Figure 1, learn, guess and slip rates were learned from the data for all 387 skills in the algebra (development) set and 442 skills in the bridge to algebra (development) set. The distribution of the values of those parameters for each dataset is shown in Figure 2.

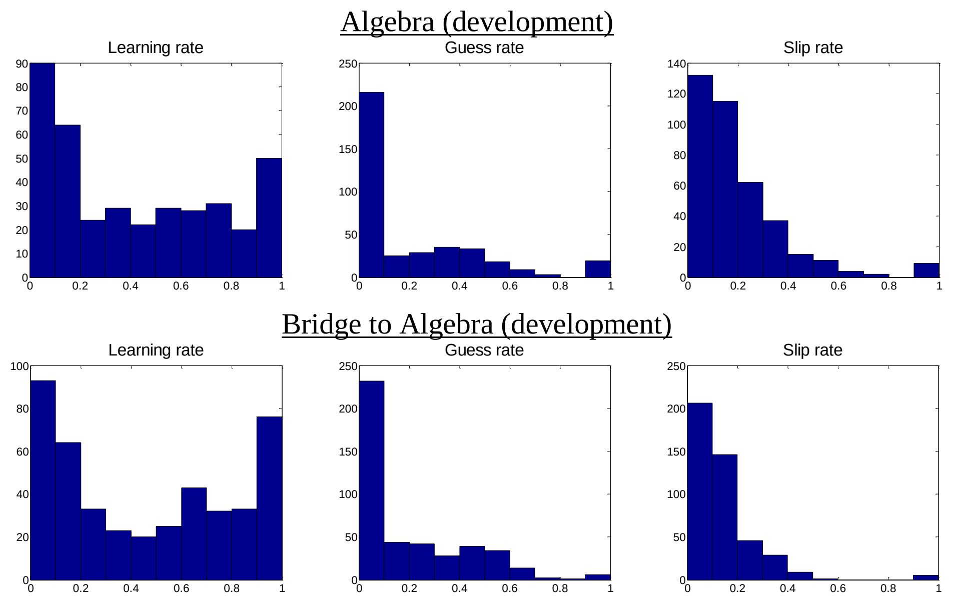


Figure 2. Distribution of skill parameters in the algebra and bridge to algebra development sets

Figure 2 shows that both datasets are populated with skills of various learning rates with a higher frequency of skills that are either very hard or very easy to learn. Both datasets have a high frequency of skills that are both hard to guess and hard to slip on. The Algebra (development) set appears to have slightly more skills with higher slip rates than bridge to algebra (development).

3.1.3 Prediction performance of the KC models in the challenge datasets

Unlike the development sets, the challenge datasets had multiple KC columns which gave different skill associations for the step. The bridge to algebra set had two KC columns while the algebra set had three. As described in section 3.2, two versions of each KC model were created, each using a different strategy for converting multi skill step representations to a single skill. The results in Table 3 describe the KC model and RMSE. KC model "2-1", for instance, refers to the 2nd KC column for that dataset using "use the hardest skill" for multiple skill steps while KC model "2-2" refers to the 2nd KC column using "treat a set of skills as a separate skill".

Algebra (challenge)

	KC model	**RMSE**
1	3-1	0.2834
2	3-2	0.2835
3	1-1	0.3019
4	1-2	0.3021
5	2-2	0.3049
6	2-1	0.3050

Bridge to Algebra (challenge)

	KC model	**RMSE**
1	1-1	0.2858
2	1-2	0.2860
3	2-1	0.2870
4	2-2	0.2871

Table 3. Prediction accuracy of the KC models in both challenge datasets

The most significant observation from Table 3 is the considerably superior performance of the third KC model in the algebra set. The different of 0.0185 between the algebra KC models 3-1 and 1-1 is greater than the RMSE difference between the first and tenth overall finisher in the competition. The difference between multiple skill approach 1 and 2 was negligible.

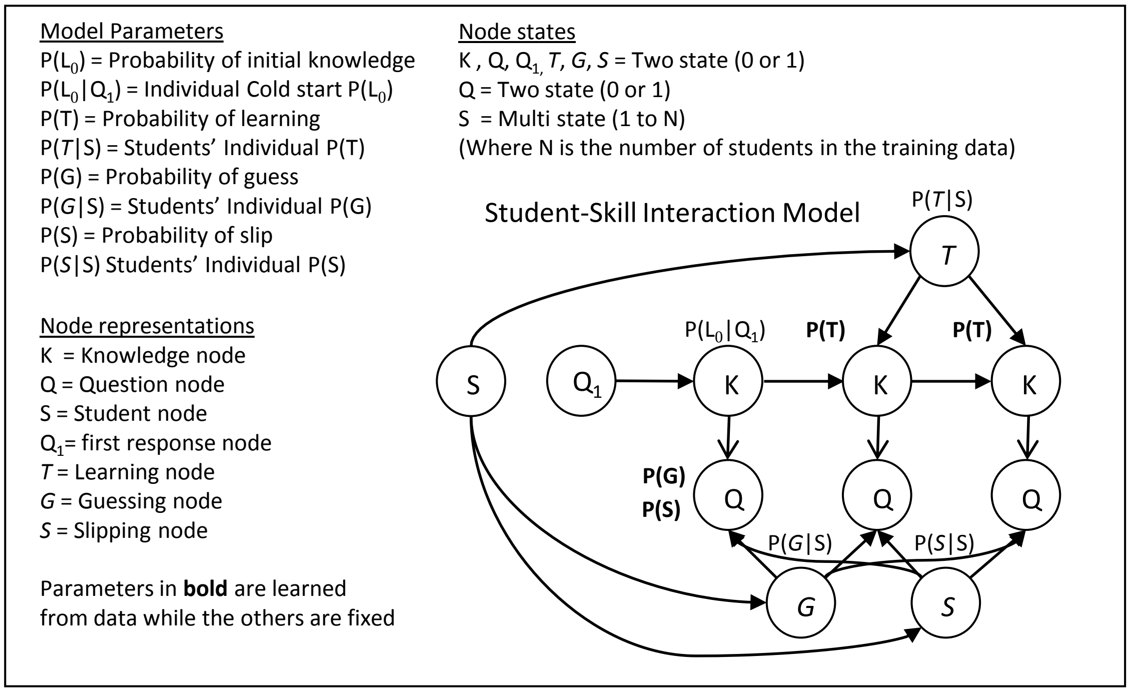
It is important to note that the Bayesian models only made predictions when there existed previous responses by the student to the skill being predicted. If no prior skill data existed no prediction was made. This occurred in a significant portion of the test data (~10%). Therefore, the RMSE scores shown in Table 3 represent the RMSE only for the predicted rows and not the entire test set. It was also the case that total number of predicted rows for each KC model differed by ~1,200, likely due to a Bayesian skill prediction job not finishing or other processing anomaly. While 1,200 rows only constitutes 0.2% of the total algebra test rows it was a significant enough difference to cause the algebra 3-2 KC model to appear to have a lower RMSE than 3-1 and for the bridge to algebra KC model 1-2 to appear to have a lower RMSE than 1-1. Because of this, all subsequent models were created using 3-2 and 1-2. The RMSE scores in Table 3 were calculated based only on the test rows that all the KC model predictions had in common which was 435,180/508,912 (86%) rows for algebra and 712,880/774,378 (92%) rows for bridge to algebra. The additional prediction rows were filled in by Random forests for the final submission.

3.2 The Student-Skill Interaction Model (Complex Model)

The more complex model expanded on the simple model considerably. The idea was to learn student specific learn, guess and slip rates and then use that information in training the parameters of skill specific models. The hypothesis is that if a student has a general learning rate trait then it can be learned from the data and used to benefit inference of how quickly a student learns a particular skill and subsequently the probability they will answer a given step correctly.

The first step in training this model was to learn student parameters one student at a time. Student specific parameters were learned by using the simple model shown in Figure 1 but only training on the data of the individual student. The rows of the data were skills answered by the student and the columns were responses to steps of those skills. All responses per skill started at column 1. Some skills spanned more columns than others due to more responses on those skills. EM is able to work with this type of sparsity in the training matrix.

The second step was to embed all the student specific parameter information into the complex model, called the Student-Skill Interaction (SSI) Model, shown in Figure 3.



Model Parameters
P(L$_0$) = Probability of initial knowledge
P(L$_0$|Q$_1$) = Individual Cold start P(L$_0$)
P(T) = Probability of learning
P(*T*|S) = Students' Individual P(T)
P(G) = Probability of guess
P(*G*|S) = Students' Individual P(G)
P(S) = Probability of slip
P(*S*|S) Students' Individual P(S)

Node representations
K = Knowledge node
Q = Question node
S = Student node
Q$_1$= first response node
T = Learning node
G = Guessing node
S = Slipping node

Parameters in **bold** are learned from data while the others are fixed

Node states
K , Q, Q$_1$, *T*, *G*, *S* = Two state (0 or 1)
Q = Two state (0 or 1)
S = Multi state (1 to N)
(Where N is the number of students in the training data)

Figure 3. Student-Skill Interaction Model

There is an SSI model for each skill but each SSI model is instantiated with the same student specific parameter data. For example, the list of student learning rates is placed into the CPT of the *T* node. The parameters that are learned in the SSI model are the guess, slip and learn rate for each skill. The effect of the student parameter nodes is to inform the network which students have high learn, guess or slip rates and allow the skill parameters to be learned conditioning upon this information. For example, two learning rates will be learned for each skill. One learning rate for if the student is a fast learner (described in the *T* node) and one learning rate for if the student is a not a fast learner. The same is done for the skill's guess and slip parameters. These values can be different for each skill but they are conditioned upon the same information about the students. While a student may have a high individual learn rate, the fast-student learn rate for a difficult skill like Pythagorean Theorem may be lower than the fast-student learn rate for subtraction. The model also allows for similar learn rates for both fast and slow student learners. Results of SSI vs. PPS are shown in Table 4. The improvement is modest but significant.

Algebra (challenge)

	Bayesian model	**RMSE**
1	SSI (KC 3-2)	0.2813
2	PPS (KC 3-2)	0.2835

Improvement: 0.0022

Bridge to Algebra (challenge)

	Bayesian model	**RMSE**
1	SSI (KC 1-2)	0.2824
2	PPS (KC 1-2)	0.2856

Improvement: 0.0032

Table 4. Results of complex SSI model vs. the simple PPS model.

3.2.1 Distribution of student parameters

Student specific learn, guess and slip rates were learned from the data for all 575 student in the algebra (development) set and 1,146 student in the bridge to algebra (development) set. The distribution of the values of those parameters for each dataset is shown in Figure 4.

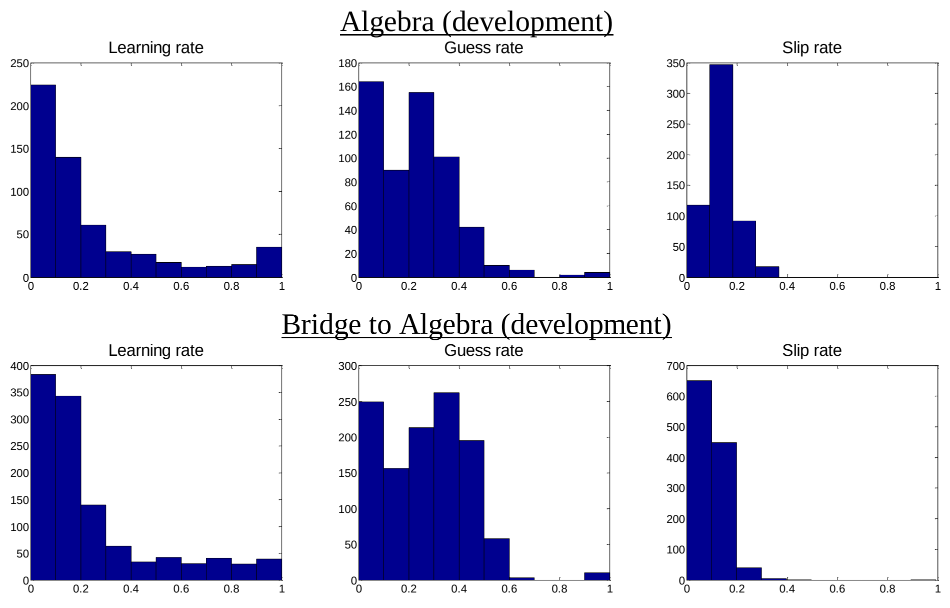


Figure 4. Distribution of student parameters in the algebra and bridge to algebra development sets

Figure 4 shows that users in both datasets have low learning rates but that a small portion of students posses learning rates in each range. Moderate guessing and low slipping existed among students in both datasets.

4 Random Forests Classifier

Leo Breiman Random Forests© (Breiman , 2001) were used to make predictions based on a rich set of features from the training and testing sets. Random Forests is also known as bagged decision trees since it trains an ensemble of decision tree classifiers and uses bagging to combine predictions. Using feature extraction, only a fraction of the data needed to be used for training.

4.1 Parameters of the Random Forest algorithm

MATLAB's TreeBagger implementation of bagged decision trees was used. Regression mode was used so that the end prediction would be a value between 0 and 1 representing the probability of the binary class. The number of features for each tree to sample was left at its default of 1/3 the number of features. The two parameters that were modified were MinLeaf and NumTrees. MinLeaf is the minimum number of observations per tree leaf. This is recommended to be set at 1 for classification and 5 for regression, however, the optimal values for this parameter were often between 15 and 65. The NumTrees parameter is the number of bagged decision trees trained. The rule of thumb is to use a value of 200 or greater. Values between 50 and 800 were tried. For some of the feature sets a randomly chosen 50,000 rows were used for training and 50,000 for testing in order to do a grid search of the MinLeaf parameter. MinLeaf was searched from 1 to 100 in increments of 1 and NumTrees was set at 50 for this parameter search. NumTrees did not appear to affect the optimal MinLeaf value chosen, however this was not tested thoroughly. It is possible that there is a different optimal MinLeaf value depending on NumTrees.

4.2 Feature Extraction

Feature sets for random forest training and prediction were created. Some were created based on KCs while others were based on user and problem properties. Values for the test set features were created from the training set. For instance, population-percent-correct for a particular problem in the test set was calculated by taking the percent correct of that problem in the training set. The same procedure was applied to create features for the internal test set. The internal test set, with all features added, then became the training set for the challenge test set.

4.2.1 Percent correct features

For each skill, the percent correct of steps associated with that skill was calculated for each section, problem and step the skill was associated with including the overall percent correct for steps of that skill. This was done for each of the skill models in each of the challenge datasets. Percent correct was also calculated for each student by unit, section, problem, step and overall percent correct. These features were joined into the test sets that will be used as training sets. The joining looks at the user, skill, unit, section, problem and step of the row in the test set and adds the appropriate ten percent correct features to it, five from user and five from skill.

4.2.2 Student progress features

These features were based upon previous performance of a student in the training set prior to answering the test row. Many of these features were adopted from work on gamming the system (Baker et al., 2008) which is a type of behavior a student can exhibit when he or she is no longer trying to learn or solve the problem but instead is clicking through help and hints in the problem. Features of student progress that were generated included the following:

- The number of data points: [today, on the first day of using the tutor, since starting the tutor, on the first day of starting the current unit]
- The number of correct answers among the last [3, 5, 10] responses
- The percent correct among the last [3, 5, 10] responses
- The number of steps out of the last 8 in which a hint was requested
- The mean number of hints requested in the last 10 steps

- The mean number of incorrect attempts in the last 10 steps
- The number of days since [starting the tutor, starting the unit]
- The sum of the last [3, 10] z-scores for [step duration, hints requested, incorrect attempts]

Z-scores were calculated by first calculating the mean and standard deviation of step duration, hints requested and incorrect attempts on a step for each skill. A z-score for step duration, for instance, was calculated by taking the step duration of a student on the last step and subtracting the mean step duration for that skill and then dividing that by the standard deviation step duration for that skill. The sum of the last three such z-scores constituted a feature. In addition to the features listed above, identical features were generated specific to the skill associated with the test row. For example, the feature "number of data points today" would become "number of data points of skill X today" where skill X is the still associated with the test row that the feature value is being generated for. There was often not enough past data for a particular skill to calculate the feature. Because of this, the skill specific version of student progress feature set covered fewer test rows than the non skill specific version.

4.2.3 Bayesian HMM features

The SSI model, which was run for each skill in the KC model of a dataset, generated various outputs that were treated as features for the Random forests. The features generated included:

- The predicted probability of correct for the test row
- The inferred probability of knowing the skill
- The absolute value of the inferred probability of knowing the skill subtracted by the predicted probability of correct
- The number of students used in training the parameters
- The number of data points used in training the parameters
- The final EM log likelihood fit of the parameters divided by the number of data points
- The total number of steps in the predicted test problem
- The problem step number
- The problem step number divided by the total number of steps in the test problem

Similar to the skill specific student progress features, the Bayesian HMM features required that prior skill data for the student be available. If such data was not available, no features were created for that test row. Because of this the Bayesian feature set did not cover all the test rows.

4.3 Random forest prediction results

After generating features for the three test sets (two internal sets and the challenge test set) based on the top KC models, Random forests were trained on the internal test set 2 to predict 1 and trained on internal test set 1 to predict 2. The means of the RMSE results for the two are shown bellow for the best performing Random forest parameter combinations per feature set. Coverage percentage is also included indicating what percentage of the total test rows were predicted by the feature set.

Algebra (challenge)

	Feature set	RMSE	Coverage
1	All features	0.2762	87%
2	Percent correct+	0.2824	96%
3	All features (fill)	0.2847	97%

Bridge to Algebra (challenge)

	Feature set	RMSE	Coverage
1	All features	0.2712	92%
2	All features (fill)	0.2791	99%
3	Percent correct+	0.2800	98%

Table 5. Random forest prediction results of different feature sets

Table 5 shows that with all features, Random forests predict 92% of the bridge to algebra test set with an RMSE of 0.2712. This is outstanding prediction accuracy given that the top RMSE for this dataset on the leaderboard was 0.2777. The problem is that the remaining 8% of the test rows represent students who do not have past data for the skill being predicted and this group is particularly difficult to predict. The "All features (fill)" is the simplest attempt to fill in the missing values of "All features" by taking the mean value for a column when it doesn't exist in a skill feature set that covers less than the other feature sets. The improvement is significant over only using the percent correct feature sets but the accuracy decreased heavily from the original. An improvement on this would have been to take the mean value for the column amongst rows of the same skill. A step further still would have been to predict or impute the missing values using Random forests. Time ran out in the competition so these last two steps became a matter for future investigation.

5 Ensemble selection

A variant of ensemble selection (Caruana and Niculescu-Mizil, 2004) was used to blend the various model predictions. Because of the varying number of predictions between models, a special ensemble initialization technique was created whereby the best model was chosen first based on lowest RMSE and subsequent models were chosen based on the RMSE of the predicted rows excluding the rows already added to the initialized ensemble. This allowed for models to be used for the portions of the test set in which they excelled. For instance, the rows of the test set containing skills not yet seen by the user were best predicted by a model that was not a top predicting model overall.

After the initialization, all models were blended with the current ensemble to determine which resulted in the best improvement to RMSE. The processes stopped when no blending of models would improve RMSE. Only three models were chosen in the blending stage for the bridge to algebra set and two for the algebra set. In this ensemble selection procedure, the internal test set RMSE is minimized and the same actions are performed on the challenge test predictions as on the internal ones. Since two internal test sets had been made, we were able to confirm that this ensemble selection procedure decrease the RMSE of the hill climbing set as well as the external test set. Table 6 shows the models chosen during the initialization processes and what percent of the test rows were covered after adding the prediction's rows to the ensemble. There were 76 models for ensemble selection of the algebra set and 81 for the bridge to algebra set. This included the Bayesian model predictions and Random forest predictions with various parameters.

Algebra (challenge)

	Prediction file	**RMSE**	**Coverage**
1	Rf600m35_allFeat	0.2762	87%
2	SSI_KC_3-2	0.2758	91%
3	Rf100m15_hints	0.2839	99%
4	Rf100m15_pctCor	0.2840	100%

RMSE after blending (2 models): 0.2834

Bridge to Algebra (challenge)

	Prediction file	**RMSE**	**Coverage**
1	Rf500m15_allFeat	0.2712	92%
2	SSI_KC_1-2	0.2719	94%
3	Rf800m15_pctCor2	0.2775	99%
4	Rf250m15_pctCor	0.2785	100%

RMSE after blending (3 models): 0.2780

Table 6. Ensemble selection procedure and RMSE improvement on the hill climbing set

6 Conclusion

Combining user features with skill features was very powerful in both modeling and classification approaches. Prediction error was very low for rows that had sufficient data to compile a complete user and skill feature set however error was very high for rows were the user did not have sufficient skill data. In order to increase prediction accuracy for these rows, imputing missing features could be very beneficial. What to do with these rows is a worthy area of future study since, while small, they significantly impacted overall RMSE.

Acknowledgements

We would like to thank the National Sciences Foundation and U.S. Department of Education for their funding including NSF equipment grant (CNS CRI 0551584). We would also like to thank the WPI Computer Science Department for their funding and Professors Baker, Beck, and Ruiz at WPI who gave talks relating to the competition. The first author would also like to thank academic advisor Neil Heffernan for his excellent support throughout the degree and for his guidance in designing the complex Bayesian models[2]. And, of course, thank you to everyone in the Educational Data Mining community.

References

R.S.J.d. Baker, Albert T. Corbett, Ido Roll, Kenneth R. Koedinger. Developing a Generalizable Detector of When Students Game the System. *User Modeling and User-Adapted Interaction*, 18(3):287-314, 2008.

L. Breiman. Random forests. *Machine Learning*, 45(1):5-32, 2001.

Rich Caruana and Alexandru Niculescu-Mizil. Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning* (ICML'04), 2004.

Albert T. Corbett, John R. Anderson. Knowledge tracing: modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4, 253–278, 1995.

Zachary A. Pardos and Neil T. Heffernan. Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. In *Proceedings of the 18th International Conference on User Modeling, Adaptation and Personalization*. Hawaii, 2010.

Zachary A. Pardos and Neil T. Heffernan. Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm. In *Proceedings of the 3rd International Conference on Educational Data Mining*. Pittsburg, 2010.

APPENDIX

Notes on other machine learning techniques attempted: Neural networks with 1-3 hidden layers were tried but with predictive performance far below that of bagged decision trees. SVMs were also tried with both linear and non-linear kernels. The linear kernel SVM parameters were explored using a coarse grid search and then a higher resolution search around the areas of low RMSE found in the first search. This approach resulted in prediction accuracies comparable to the neural network predictions.

Notes on hardware and software used: A 30 node rocks cluster with 4 CPUs per node and a 6 node rocks cluster with 8 CPUs per node were used to train the ~1,500 Bayesian skill models for each dataset and to generate the feature sets. One 16 core and one 8 core machine with 32gigs of RAM each were used to run the bagged decision tree classification using MATLAB's TreeBagger function. The Parallel Computing Toolbox was used to parallelize the training of the Random forests naïve decision tree classifiers over 8 processor cores. All skills for a KC model could be run in 2 days with the complex model. Random forests prediction took 2 to 14 hours.

[2] In keeping with the spirit of a student team, minimal assistance was sought during the official competition.