JMLR

Pardos, Z.A., Heffernan, N. T.: Using HMMs and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset. To appear in the *Journal of Machine Learning Research W & CP*, In Press

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 Knowledge Discover and Data Mining 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 Association for Computing Machinery (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 made by Carnegie Learning. One came from the Algebra Cognitive Tutor system; this dataset was simply called "Algebra". The other came from the Bridge to Algebra Cognitive Tutor 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 given information about the step and the students past history of responses. Predictions between 0 and 1 were allowed and were scored based on root mean squared error (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 development dataset were announced early on; therefore that dataset was not analyzed for this article.

1.1 Summary of methods used in the final prediction model

The final prediction model was an ensemble of Bayesian Hidden Markov Models (HMMs) and Random Forests (bagged decision trees with feature and data re-sampling randomization). One of the HMMs used was a novel Bayesian model developed by the authors, built upon prior work (Pardos & Heffernan, 2010a) 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 outperformed 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 (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; a student would start a math curriculum determined by her teacher. The student would be given multi step problems to solve often consisting of multiple different skills. The student could make multiple attempts at answering a question and would receive feedback on the correctness of her answer. The student could ask for hints to solve the step but would be marked as incorrect if a hint was requested. Once the student achieved "mastery" of a skill, according to the system, the student would no longer need to solve steps of that skill in their current curriculum, or unit.

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 which 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' advancement through the tutor curriculum is based on their mastery of the skills involved in the pedagogical unit they are working on. If a student does not master all the skills in a unit, they cannot advance to the next lesson on their own; however, a teacher may intervene and skip them ahead.

1.3 Format of the datasets

The datasets all contained the same features and the same format. Each row in a dataset corresponded to one response from a student on a problem step. Each row had 18 features plus the target, which was "correct on first attempt". Among the features were; unit, problem, step and skill. The skill column specified which math skill or skills were associated with the problem step that the student attempted. A skill was associated with a step by Cognitive tutor subject matter experts. In the development datasets there were around 400 skills and around 1,000 in the competition datasets. The Algebra competition set had two extra skill association features and the Bridge to Algebra set had one extra. These were alternative associations of skills to steps using a different bank of skill names (further details were not disclosed). The predictive power of these skill associations was an important component of our HMM approach.

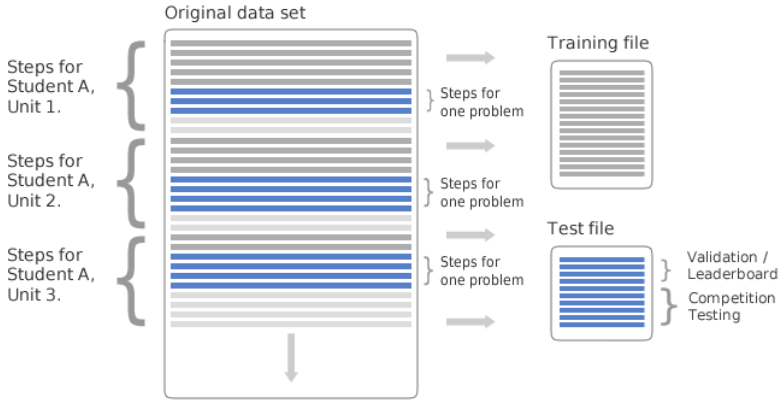

Figure 1. The test set creation processes as illustrated by the organizers

The organizers created the competition training and test datasets by iterating through all the students in their master dataset and for each student and each unit the student completed, selecting an arbitrary problem in that unit and placing into the test set all the student's rows in that problem. All the student's rows in that unit prior to the test set problem were placed in the training set. The rows following the selected problem were discarded. This process is illustrated in Figure 1 (compliments of the competition website).

1.4 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. Competition organizers explained that these features were omitted from the test set because they made the prediction task too easy. In internal analysis we confirmed that step duration was very predictive of an incorrect or correct response and that the value of the hints and incorrects column completely determined the value of the target, "correct on first attempt". This is because the tutor marks the student as answering incorrect on first attempt if they receive help on the question, denoted by a hint value of greater than 0. The incorrects value specified how many times the student answered the step incorrectly.

In the development datasets, valuable information about 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 therefore 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, as described in section 1.3.

2 Data preparation

The first step to being able to work with the dataset was to convert the categorical, alphanumeric fields of the columns into 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. Rows were divided out into separate files based on skill and user for training with the Bayes Nets.

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 duration 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 Creating an internal validation dataset

An internal validation dataset was created in order to provide internal scoring of various prediction models. Besides using the scoring to test the accuracy of the Bayesian Networks and Random Forests methods it was also used to test various other approaches such as neural networks, linear regression and SVMs (see appendix). A validation dataset was created for each of the competition datasets from the training datasets by taking all the rows in the last problem of each student's units and placing them in the validation set and the remaining data into an internal

training set. This process was meant to mirror the processes used by the organizers to create the official test set, described in section 1.3. The only difference was that the last problem in a unit was selected instead of an arbitrary problem in a unit. The missing features from the official test sets were also removed from the created validation sets. By fashioning the validation sets after the official test set, a high correlation between validation and test set results should be achieved. A second validation set was also created so that ensemble methods could be tested internally. This set was created from the training rows that were not placed into the first validation set. The second validation set constituted rows from students' second to last problem in each of their units.

2.2 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 Tutors to track student learning over the course of the curriculum. KC skill associations that more accurately correlated with 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.

2.2.1 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 clicking a button or other interface related interactions.

2.2.2 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 a standard practice in student modeling. To avoid having to model multiple skills per step, the KC values with multiple skills were collapsed into one skill. Two strategies for collapsing the values were tried for each KC column. The first 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. KC models applying this strategy will be labeled with "-hard" throughout the text. The second way of collapsing multiple skill values was to treat a unique set of skills as a completely separate skill. Therefore, a step associated with "Subtraction" and "Addition" skills would be merged into the skill of "Subtraction-Addition". KC models applying this strategy will be labeled with "-uniq" throughout the text. The result of this processing was the generation of two additional skill models for each KC column for each challenge set. All of the development dataset analysis in this paper uses only the unique strategy, for brevity.

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 (Corbett & 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 can be fit to the data using Expectation Maximization (EM) or a grid search of the parameter space. We used EM with a max iteration of 100. EM will also stop if the log likelihood fit to the data increases by less than 1e-5 between iterations. While this simple HMM was the basis of our Bayesian Networks approach, additional models which utilized the parameters learned by the simpler models were utilized for prediction.

3.1 The Prior Per Student Model (Simple Model)

Standard knowledge tracing has four parameters. A separate set of parameters are fit for each skill based on students' sequences of responses to steps of that skill. The intuition is that students will learn a skill over time. The latent represents knowledge of that skill and the two transition probabilities for the latent are prior knowledge and learning rate. Prior knowledge is the probability that students knew the skill prior to working on the tutor. Learning rate is the probability that students will transition from the unlearned to the learned state between opportunities to answer steps of that skill. The probability of transitioning from learned to unlearned (forgetting) is fixed at zero since the time between responses is typically less than 24 hours. Forgetting is customarily not modeled in Knowledge Tracing; however, it certainly could be occurring given a long enough passage of time between opportunities. The two emission probabilities are the guess and slip rate. Guess is the probability of observing a correct response when the student is in the unlearned state. Slip is the probability of observing an incorrect response when the student is in the learned state. Prior work by the authors has shown that modeling a separate prior per student in the training and prediction steps can increase the accuracy of the learned parameters (Pardos & Heffernan, 2010b) as well as prediction accuracy (Pardos & Heffernan, 2010a). In parameter analysis 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 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.

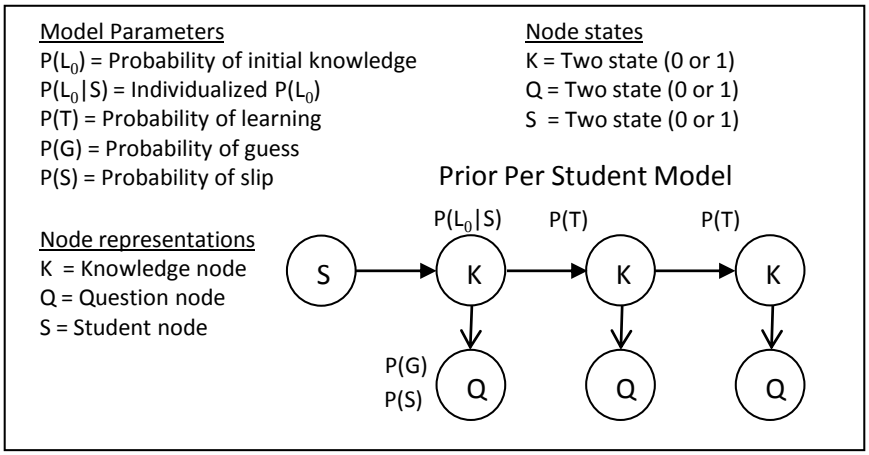

Figure 2. Prior Per Student (PPS) model parameters and topology

Figure 2 shows the Prior Per Student (PPS) model topology. In this model the student node acts as a unique student identifier with values that range from 1 to N where N is the number of students in the dataset; however, we have found that modeling only two distinct priors and assigning a student to one of those priors based on their first response 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, 2010b). 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 missing 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 occupy over 8GB of virtual memory on the compute machines. This was too much as the machines used to run these models had only 8GB and reaching into swap memory caused the job to take considerably 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 for EM training

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 in addition to eliminating the memory issue which would have been compounded when working with the much larger competition sets. A limit of 25 would be used for all subsequent models.

3.1.2 Distribution of skill parameters

Using the PPS model; 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 is shown with histograms in Figure 3.

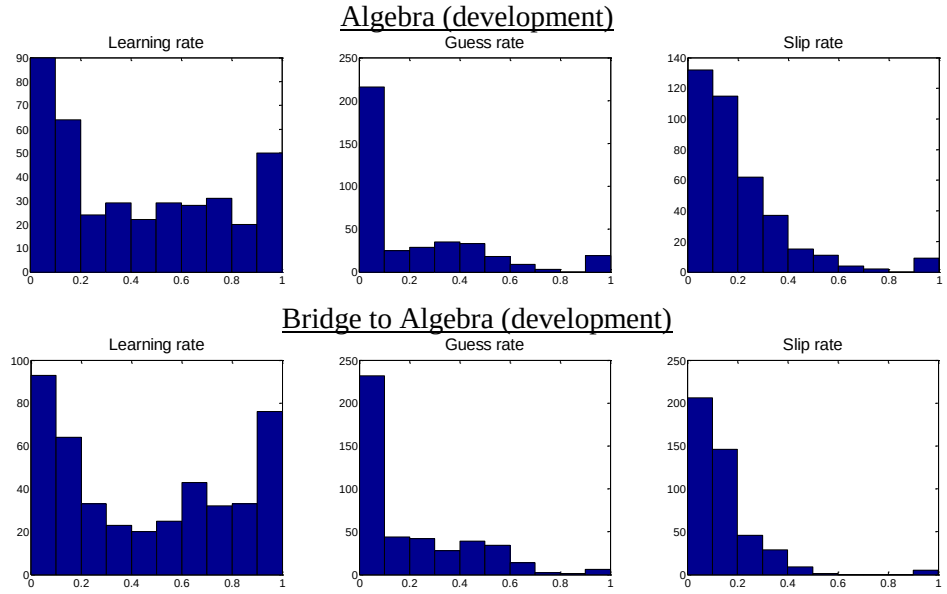

Figure 3. Distribution of skill parameters in the algebra and bridge to algebra development sets

The X axis of the histograms in Figure 3 is the value of the parameter and the Y axis is the occurrence of that parameter value among the skills in the dataset. These parameters were learned from the data using EM with the prior per student model (cold start heuristic). Figure 3 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 each step. The bridge to algebra set had two KC columns while the algebra set had three. As described in section 2.2.2, 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-hard", for instance, refers to the 2nd KC model for that dataset with "use the hardest skill" applied for multiple skill steps while KC model "2-uniq" refers to the 2nd KC model using "treat a set of skills as a separate skill".

Algebra (challenge)

	KC model	# Skills	RMSE
1	3-hard	2,359	0.2834
2	3-uniq	2,855	0.2835
3	1-hard	1,124	0.3019
4	1-uniq	2,207	0.3021
5	2-uniq	845	0.3049
6	2-hard	606	0.3050

Bridge to Algebra (challenge)

	KC model	# Skills	RMSE
1	1-hard	1,117	0.2858
2	1-uniq	1,809	0.2860
3	2-hard	920	0.2870
4	2-uniq	1,206	0.2871

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

The most significant observation from Table 3 is the considerably better performance of the third KC model in the algebra set. The different of 0.0185 between the algebra KC models 3-hard and 1-hard is greater than the RMSE difference between the first and tenth overall finisher in the competition. The differences between the multiple skill approaches were negligible. Table 3 also shows the number of skills in each competition datasets per KC model with the hard and unique multi-skill reduction strategy applied. The unique strategy always created more rules but the difference is most prominent for KC column 1. The table also shows how the various KC models differ in skill granularity; Algebra model 2-hard has only 606 skills used to associated with steps while Algebra model 3-hard used 2,359 skills to associate with those steps. Among the "–hard" models, the more skills the KC model had, the better it performed.

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. No previous skill information for a student was available 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-uniq KC model to appear to have a lower RMSE than 3-hard and for the bridge to algebra KC model 1-uniq to appear to have a lower RMSE than 1-hard in our preliminary RMSE calculations. Because of this, all subsequent models run during the competition were created using 3-uniq and 1-uniq. The RMSE scores in Table 3 are the corrected calculations 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. This model was created during the competition and has not been described previously in publication.

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 PPS model by training on all skill data of an individual student one at a time. The rows of the data were skills encountered by the student and the columns were responses to steps of those skills. All responses per skill started at column 1 in the constructed training set of responses. 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 4. Parameters were then learned for the SSI model given the student specific parameter values. After the parameters were trained the model could be used to predict unseen data given past history of responses of a student on a skill. Depending on the learning rate of the skill and the learning rate of the user, the model would forecast the rate of acquiring knowledge and give predictions with increasing probability of correct on each subsequent predicted response for a student on steps of a particular skill.

The limitation of the model is that it requires that a plentiful amount of data exists for the student in order to train their individual parameters. The format of the competition's data was ideal for this model since the students in the training set also appeared in the test set and because student data was available in the training set for a variety of skills.

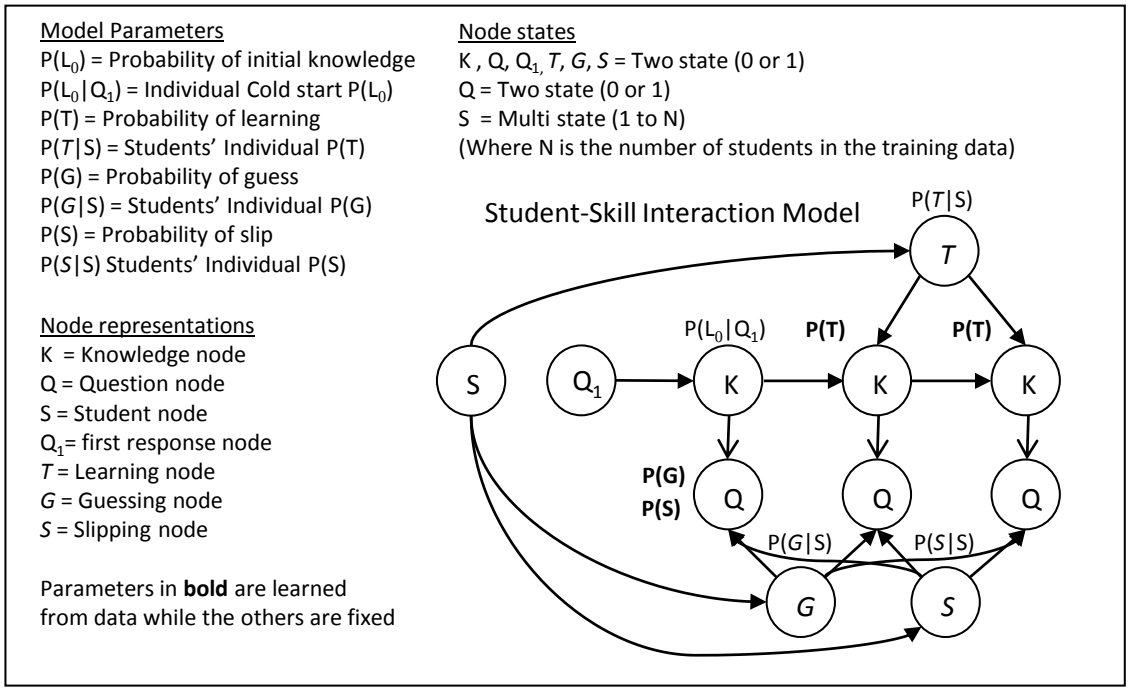

Figure 4. Student-Skill Interaction (SSI) model parameters and topology

There was an SSI model trained for each skill but each SSI model was fixed with the same student specific parameter data. For example, the list of student learning rates is placed into the conditional probability table of the *T* node. There are six parameters that are learned in the SSI model. The effect of the student parameter nodes is to inform the network which students have high or low 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 high learner (described in the *T* node) and one learning rate for if the student is a low 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 was greater than the difference between 1st and 3rd place overall in the competition. The difference between SSI and PPS squared errors were significant for both datasets at the $p << 0.01$ level using a paired t-test.

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 the SSI model vs. the PPS model.

3.2.1 Distribution of student parameters

Individual student 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 5.

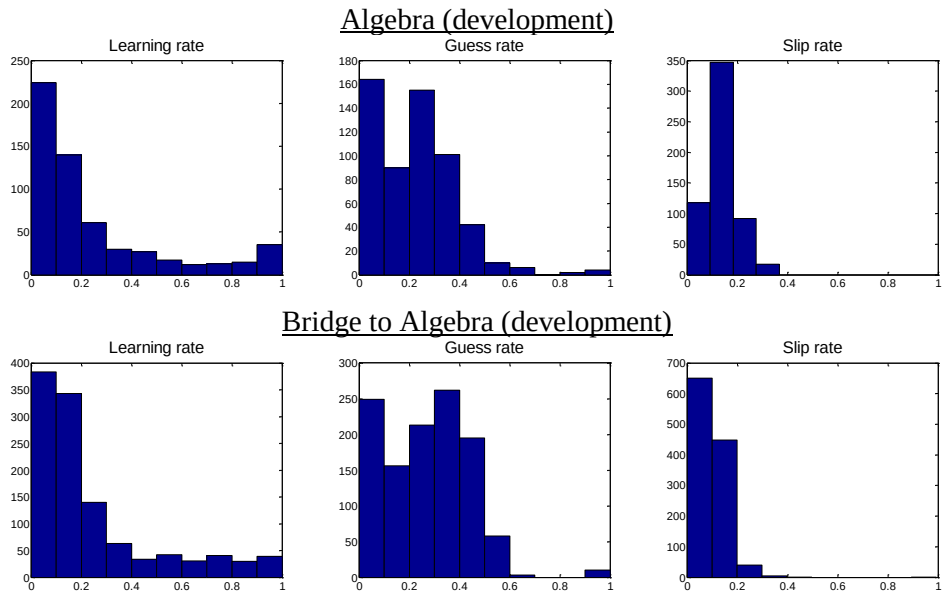

Figure 5. Distribution of student parameters in the algebra and bridge to algebra development sets

Figure 5 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. The majority of the parameters learned fell within plausible ranges.

4 Random Forests Classifier

Leo Breiman's Random Forests© (Breiman, 2001) wer used to make predictions based on a rich set of features from the training and testing sets. Random Forests is a variant on bagged decision trees. Random Forests trains an ensemble of decision tree classifiers or regression trees and uses bagging to combine predictions. Each tree selects a portion of the features at random and a random resampling of the data to train on. This approach required feature engineering and feature extraction as opposed to the HMM approach which required student responses grouped by skill.

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 for regression mode; $1/3^{rd}$ the number of features. The two parameters that were modified were MinLeaf and NumTrees. MinLeaf is the minimum number of observations needed 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 based on testing with a small validation set. The NumTrees parameter is the number of random decision trees trained. The rule of thumb is to use a value of 50 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 parameter search of the optimal 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. Each tree trained by Random Forests resamples from the training data, with replacement. The size of the resampled training data can be set, however, this was left at its default value which was to create a resampled set the same size as the original training set.

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. The largest set contained 146 featured. The features created for training had to also be created for the official test sets, which contained missing features that were not missing in the official training sets. With this in mind, the strategy was to aggregate existing features from the training set into the test set. An example of this would be creating a feature called "average step duration of student on skill". The feature of "step duration" exists in the official training set but not the test set. So, in order to add this feature to the test set, the average step duration of student 10 on skill B, for example, was calculated for the data in the training set and every row in the test set that contained student 10 and skill B was given this aggregated value for the "average step duration of student on skill" feature column. This process was used to create many of the features for the test set. The training dataset had to contain the same features as the test set in order for the test set features to be of use. In order to accomplish this, the internal validation sets were utilized. Since the validation set was of the same format as the test set, the same feature creation procedure was run on the validation set using the remainder of the training set data not used in the validation set. Figure 6 depicts the portions of the dataset that were used for generating the feature rich datasets that the Random Forests ultimately trained on.

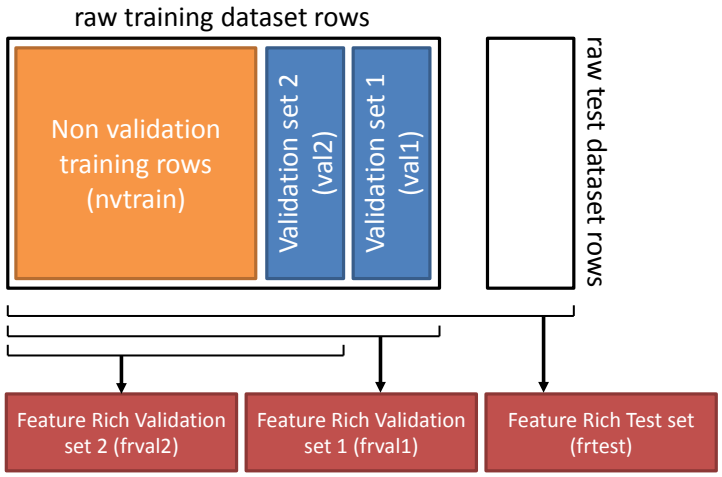

Figure 6. Diagram of the segments of the dataset that were used for generating the various feature rich datasets used for Random Forests training and testing

Figure 6 shows, for instance, how the non-validation training rows (nvtrain) in addition to validation set 2 (val2) were used to generate features for the feature rich validation set 2 (frval2). Only nvtrain was used to generate missing test set related features, such as "average step duration of student on skill", for frval2; however, val2 could still be used to generate features that were available in the official test set, such as "number of questions answered by student in this problem".

Random Forests was trained on frval1 to predict frval2 and trained on frval2 to predict frval1 as part of a 2-fold internal cross-validation. The datasets frval1 and frval2 were combined when training the Random Forests models to make predictions on the official test set. The cross-validated frval1 and frval2 predictions were combined as the validation set for Ensemble selection. The Bayesian network SSI model was also used to generate features as well as its predictions for the feature rich sets. To produce features for frval2, only nvtrain was used to train parameters for the model. To produce features for frval1, only data from nvtrain+val2 were used and to produce features for the official test set, data from the entire training set were used (nvtrain+val1+val2).

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 gaming 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 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 number of steps completed thus far in the predicted test problem

- The number of steps completed 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 were 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 datasets (two validation sets and the official test set) based on the top KC models, Random forests were trained on the two validations sets. The RMSE results for the validation sets are shown in Table 5 for the best performing Random Forest parameter combination for the full feature set. Coverage percentage is also included indicating what percentage of the total validation rows were predicted by the feature set. Prediction using only basic percent correct features to train on is also included as a baseline.

Algebra (challenge)

	Feature set	RMSE	Coverage
1	All features	1.4942	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

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 winning RMSE for this dataset on the leaderboard was 0.2777. The problem was that the remaining 8% of the test rows represent students who do not have past data for the skill being predicted and this group was particularly difficult to predict. The "All features (fill)" was an attempt to fill in the missing values of "All features" by using the mean value of a column in place of its missing values and retraining on this "filled in" dataset. This approach provided some benefit over using just percent correct features to train the Random Forests with the bridge to algebra set but performed worse than the percent correct features in the algebra set. An improvement on this would have been to take the mean value for the column among the 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 approaches for future investigation.

4.4 Feature set importance

The importance of the various feature sets was calculated by turning on the Random Forests out of bag permuted variable error calculation. This feature allowed the model to permute the value of each feature and then observe the change in mean standard error among tree predictions. A higher positive change in error indicates more importance.

Algebra (challenge)

	Feature set	Importance
1	Student progress	1.4942
2	Percent correct (Skill)	1.3615
3	Student progress (Skill)	1.3094
4	Percent correct (User)	1.2732
5	SSI model features	0.9993

Bridge to Algebra (challenge)

	Feature set	Importance
1	Percent correct (User)	2.1831
2	Student progress	2.0989
3	Student progress (Skill)	1.8118
4	SSI model features	1.6436
5	Percent correct (Skill)	1.5950

Table 6. Random Forests average variable importance for each feature set

A Random Forests model was trained on frval2 with the all features dataset with 200 trees and min leaf of 15. The average importance of each variable within a feature set was calculated to produce the results in Table 6. The table shows that the student progress feature set was highly important in both datasets. The percent correct features of the user were most important in the bridge to algebra set; however, the percent correct features of skill were the least important. Inversely, in the algebra set, the percent correct features of skill were more important than percent correct feature of the user. The importance of user features on bridge to algebra is perhaps one reason why the user and skill oriented SSI model showed a greater improvement over the skill only PPS model on bridge to algebra. The SSI model features added value but made the least impact on error on average. This was the only feature set containing features and a prediction from another classifier. This characteristic could have made it difficult for the decision trees to find additional exploitable patterns in these features.

5 Ensemble selection

A variant of ensemble selection (Caruana & Niculescu-Mizil, 2004) was used to blend the collection of Random Forests and Bayesian networks generated predictions. Because of the varying number of test rows covered by the predictions of each model, a special ensemble initialization technique was created whereby the best model was chosen first based on lowest validation set 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 data in which they excelled. For instance, the rows of the test set containing skills sparsely seen by the user were best predicted by a model that was not a top predicting model overall.

After the initialization, all models were averaged with the current ensemble to determine which resulted in the best improvement to RMSE. The processes stopped when no averaging of models would improve RMSE with respect to the validation set. Only three models were chosen in the averaging stage for the bridge to algebra set and two for the algebra set. In this ensemble selection procedure, the validation set RMSE is minimized and the same actions are performed on the official test predictions as on the validation predictions. Since two validation sets had been made, we were able to confirm that this ensemble selection procedure decreased the RMSE on a hold out set and confirmed the benefit on the official test set through feedback from the leaderboard. Table 7 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-uniq	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-uniq	0.2719	94%
3	Rf800m15_pctCor2	0.2775	99%
4	Rf250m15_pctCor	0.2785	100%

RMSE after blending (3 models): 0.2780

Table 7. Ensemble selection procedure and RMSE improvement on the hill climbing set

Table 7 shows that the most accurate model chosen for both datasets was a Random Forests model. The second model chosen was the Bayesian SSI model illustrating that the Bayesian model captured variance not captured by the Random Forests models. This was likely due to the Bayesian model's ability to competently model the temporal nature of the data.

6 Conclusion

Combining user features with skill features was very powerful in both Bayesian and Random Forests 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. Handling these rows is a worthy area of future study since prediction error of these rows substantially increased overall RMSE. Feature selection would likely have also improved prediction and a closer study of individual features importance is an important open question for future work.

The strong performance of the Knowledge Tracing based PPS and SSI models demonstrated the power of the HMM assumption of learning in educational datasets. Only using the students' past sequence of correct and incorrect responses by skill, the HMM model's predictions rivaled that of the Random Forests approach which required substantial feature engineering. The Random Forests predictions, however, were able to increase the level of prediction accuracy by leveraging features not included in the HMMs. Random forests has not been previously used in the ITS community, however, given its standalone performance and performance in concert with HMMs, they would be a valuable option to consider for future research in student performance prediction.

Acknowledgements

We would like to thank the National Sciences Foundation (NSF) and U.S. Department of Education for their funding including NSF equipment grant CNS CRI 0551584 and NSF "Graduates in K-12 Education" (GK-12) Fellowship award DGE0742503. We would also like to thank the ACM and Worcester Polytechnic Institute's Computer Science Department for its funding and Professors Ryan Baker, Joseph Beck and Carolina Ruiz at WPI who gave talks relating to the competition. The first author would also like to thank academic advisor Neil Heffernan for his support throughout the degree and guidance in designing the complex Bayesian models. And, of course, thank you to the Pittsburg Science of Learning Center (PSLC) and 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 with layer node sizes iterated between 2 and 100. The predictive performance of the NNs was 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 ~10,00 Bayesian skill models for the competition and to generate the feature sets. All skills for a KC model could be run in 2 days using the SSI model and 12 hours using the PPS model. Kevin Murphy's Bayes Net Toolbox for MATLAB was used to construct and train the Bayesian Networks models. One 16 core and one 8 core machine with 32gigs of RAM each were used to run the Random Forests classification using MATLAB's TreeBagger function. The Parallel Computing Toolbox was used to parallelize the training of the Random forests decision tree classifiers over 8 processor cores. Random forests prediction took 2 to 14 hours depending on the number of trees specified (50-800).