From Predictive Models to Instructional Policies

Joseph Rollinson
Computer Science Department
Carnegie Mellon University
jtrollinson@gmail.com

Emma Brunskill
Computer Science Department
Carnegie Mellon University
ebrun@cs.cmu.edu

ABSTRACT

At their core, Intelligent Tutoring Systems consist of a student model and a policy. The student model captures the state of the student and the policy uses the student model to individualize instruction. Policies require different properties from the student model. For example, a mastery threshold policy requires the student model to have a way to quantify whether the student has mastered a skill. A large amount of work has been done on building student models that can predict student performance on the next question. In this paper, we leverage this prior work with a new when-to-stop policy that is compatible with any such predictive student model. Our results suggest that, when employed as part of our new predictive similarity policy, student models with similar predictive accuracies can suggest that substantially different amounts of practice are necessary. This suggests that predictive accuracy may not be a sufficient metric by itself when choosing which student model to use in intelligent tutoring systems.

1. INTRODUCTION

Intelligent tutoring systems offer the promise of highly effective, personalized, scalable education. Within the ITS research community, there has been substantial work on constructing student models that can accurately predict student performance (e.g. [6, 3, 15, 5, 10, 9, 14, 7]). Another key issue is how to improve student performance through the use of instructional policy design. There has been significant interest in cognitive models used for within activity design (often referred to as the inner-loop) and even authoring tools developed to make designing effective activities easier (e.g. CTAT [1]). However, there has been much less attention to outer-loop (what problem to select or when to stop) instructional policies (though exceptions include [5, 12, 17]).

In this paper we focus on a common outer-loop ITS challenge, adaptively deciding when to stop teaching a certain skill to a student given correct/incorrect responses. Somewhat surprisingly, there are no standard policy rules or algorithms for deciding when to stop teaching for many of the student models introduced over the last decade. Bayesian Knowledge Tracing [6] naturally lends itself to mastery teaching, since one can halt when the student has mastered a skill with probability above a certain threshold. Such a mastery threshold has been used as part of widely used tutoring systems, but typically in conjunction with additional rules since a student may never reach a sufficient mastery threshold given the available activities.

We seek to be able to directly use a wide range of student models to create instructional policies that halt both when a student has learned a skill and when the student seems unlikely to make any further progress given the available tutoring activities. To do so we introduce an instructional policy rule based on change in predicted student performance.

Our specific contributions are as follows:

- We provide a functional interface to student models that captures their predictive powers without knowledge of their internal mechanics (Section 3).
- We introduce the *predictive similarity policy*: a new when-to-stop policy that can take as input any predictive student-model (Section 4) and can halt both if students have successfully acquired a skill or do not seem able to do so given the available activities.
- We analyze the performance of this policy compared to a mastery threshold policy on the KDD dataset and find our policy tends to suggest similar or a smaller number of problems than a mastery threshold policy (Section 5).
- We also show that our new policy can be used to analyze a range of student models with similar predictive performance (on the KDD dataset) and find that they can sometimes suggest very different numbers of instructional problems. (Section 5).

Our results suggest that predictive accuracy alone can mask some of the substantial differences among student models. Polices based on models with similar predictive accuracy can make widely different decisions. One direction for future work is to measure which models produce the best learning policies. This will require new experiments and datasets.

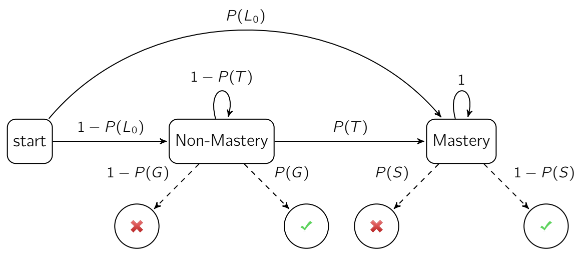

Figure 1: BKT as a Markov process. *Mastery* and *Non-Mastery* are hidden states. Arrow values represent the probability of the transition or observation.

2. BACKGROUND: STUDENT MODELS

Student models are responsible for modeling the learning process of students. The majority of student models are *predictive models* that provide probabilistic predictions of whether a student will get a subsequent item correct. In this section we describe two popular predictive student models, *Bayesian knowledge tracing* and *latent factor models*. Note that other predictive models, such as Predictive State Representations (PSRs), can also be used to calculate the probability of a correct response [7].

2.1 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) [6] tracks the state of the student's knowledge as they respond to practice questions. BKT treats students as being in one of two possible hidden states: *Mastery* and *Non-Mastery*. It is assumed that a student never forgets what they have mastered and if not yet mastered, a new question always has a fixed static probability of helping the student master the skill. These assumptions mean that BKT requires only four trained parameters:

$P(L_0)$	Initial probability of mastery.
$P(T)$	Probability of *transitioning* to mastery over a single learning opportunity.
$P(G)$	Probability of *guessing* the correct answer when the student is not in the mastered state.
$P(S)$	Probability of *slipping* (making a mistake) when the student is in the mastered state.

After every response, the probability of mastery, $P(L_t)$, is updated with Bayesian inference. The probability that a student responds correctly is

$$P_{\mathrm{BKT}}(C_t) = (1 - P(S))P(L_t) + P(G)(1 - P(L_t)). \quad (1)$$

Prior work suggests that students can get stuck on a particular activity. Unfortunately, BKT as described above assumes that students will inevitably master a skill if given enough questions. As this is not always the case, in industry BKT is often used together with additional rules to make instructional decisions.

2.2 Latent Factor Models

Unlike BKT models, Latent Factor Models (LFM) do not directly model learning as a process [3]. Instead, LFMs assume that there are latent parameters of both the student and skill that can be used to predict student performance. These parameters are learned from a dataset of students answering questions on multiple skills. The probability that the student responds correctly to the next question is calculated by applying the sigmoid function to the linear combination of parameters p and features f.

$$P_{\mathrm{LFM}}(C) = \frac{1}{1 - e^{-f \cdot p}} \quad (2)$$

Additive Factor Models (AFM) [3] are based on the assumption that student performance increases with more questions. A student is represented by an aptitude parameter (α_i) and a skill is represented by a difficulty parameter (β_k) and learning rate (γ_k). AFM is sensitive to the number of questions the student has seen, but ignores the correctness of student responses. The probability that student i will respond correctly after n responses on skill k is

$$P_{\mathrm{AFM}}(C) = \frac{1}{1 - e^{-(\alpha_i + \beta_k + \gamma_k n)}}. \quad (3)$$

Performance Factor Models (PFM) [15] are an extension of AFMs that are sensitive to the correctness of student responses. PFMs separate the skill learning rate into success and failure parameters, μ_k and ρ_k respectively. The probability that student i will respond correctly after s correct responses and f incorrect responses on skill k is

$$P_{\mathrm{PFM}}(C) = \frac{1}{1 - e^{-(\alpha_i + \beta_k + \mu_k s + \rho_k f)}}. \quad (4)$$

LFMs can easily be extended to capture other features. For example, the instructional factors model [5] extends PFMs with a parameter for the number of tells (interactions that do not generate observations) given to the student. To our knowledge there is almost no work on using LFMs to capture temporal information about the order of observations. Unlike BKT, LFMs are not frequently used in instructional policies.

Though structurally different, BKT models, AFMs and PFMs tend to have similar predictive accuracy [9, 15]. This raises the interesting question of whether instructional policies that use these models are similar.

3. WHEN-TO-STOP POLICIES

We assume a simple intelligent tutoring system that teaches students one skill at a time. All questions are treated the same, so the system only has to decide when to stop providing the student questions. In this section, we provide a general framework for the when-to-stop problem. In particular, we describe an interface that abstracts out the student model from instructional policies, which we will use to define the MASTERY THRESHOLD policy and use in the next section as the foundations of a model-agnostic instructional policy.

3.1 Accessing Models

Policies require a mechanism for getting values from student models to make decisions. We describe this mechanism as a state type and a set of functions. A student model consists of two types of values: immutable parameters that are learned on training data and mutable state that changes over time. For example, the parameters for BKT are $(P(L_0), P(T), P(G), P(S))$ and the model state is the probability of mastery $(P(L_t))$. Policies treat the state

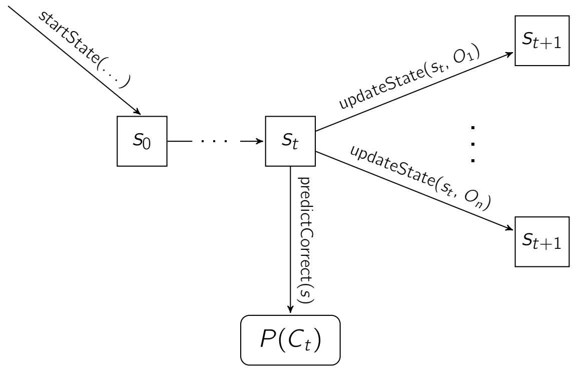

Figure 2: Model process with functional interface

as a black box, which they pass to functions. All predictive student models must provide the following functions. **startState**(...) returns the model state given that the student has not seen any questions. **updateState**(state, obs) returns an updated state given the observation. For this paper, observations are whether the student got the last question correct or incorrect. Finally, predictive student models must provide **predictCorrect**(state), which returns the probability that the student will get the next question correct. The function interfaces for BKT models and PFM are provided in table 1. Under this abstraction, when-to-stop policies are functions **stop**(state) that output true if the system should stop providing questions for the current skill and false if the system should continue providing the student with questions.

3.2 Mastery Threshold Policy

The MASTERY THRESHOLD POLICY halts when the student model is confident that the student has mastered the skill. This implies that we want to halt when the student masters the skill. Note that if the estimate of student mastery is based solely on a BKT[1] then a mastery threshold policy implicitly assumes that every student will master the skill given enough problems. Mathematically, we want to stop at time t if $P(L_t) > \Delta$, where Δ is our mastery threshold. The MASTERY THRESHOLD policy function can be written as:

$$\mathbf{stop}_M(\text{state}) = \mathbf{predictMastery}(\text{state}) > \Delta. \quad (5)$$

The MASTERY THRESHOLD policy can only be used with models that include **predictMastery**(state) in their function set. BKT models are compatible, but LFMs are not. By itself, the MASTERY THRESHOLD does not stop if the student has no chance of attaining mastery in the skill with the given activities. Students on poorly designed skills could be stuck learning a skill indefinitely.

4. FROM PREDICTION TO POLICY

In educational data mining, a large emphasis is put on building models that can accurately predict student observations. Our goal is to build a new when-to-stop policy that will work with any predictive student model.

[1]In practice, industry systems that use mastery thresholds and BKTs often use additional rules as well.

Our new instructional policy is based on a set of assumptions. First, students working on a skill will eventually end in one of two hidden end-states. Either, they will master the skill, or they will be unable to master the skill given the activities available. Second, once students enter either end-state, the probability that they respond correctly to a question stays the same. Third, if the probability that a student will respond correctly is not changing, then the student is in an end-state. Finally, we should stop if the student is in an end-state.

From these assumptions it follows that if the probability that the student will respond correctly to the next question is not changing, then we should stop. In other words, we should stop if it is highly likely that showing the student another question will not change the probability that the student will get the next question correct by a significant amount. We propose to stop if

$$(P(|P(C_t) - P(C_{t+1})| < \epsilon)) > \delta \quad (6)$$

where $P(C_t)$ is the probability that the student will get the next question right. This can be thought of as a threshold on the sum of the probabilities of each observation that will lead to an insignificant change in the probability that a student will get the next question correct, which can be written as

$$\sum_{o \in O} P(O_t = o)\mathbb{1}(|P(C_t) - P(C_{t+1}|O_t = o)| < \epsilon) > \delta \quad (7)$$

where $P(C_{t+1}|O_t = o)$ is the probability that the student will respond correctly after observation o, O_t is the observation at time t, and $\mathbb{1}$ is an indicator variable. In our case $O = \{C, \neg C\}$. This expression is true in the following cases:

1. $P(C_t) > \delta$ and $|P(C_t) - P(C_{t+1}|C_t)| < \epsilon$
2. $P(\neg C_t) > \delta$ and $|P(C_t) - P(C_{t+1}|\neg C_t)| < \epsilon$
3. $|P(C_t) - P(C_{t+1}|C_t)| < \epsilon$ and
   $|P(C_t) - P(C_{t+1}|\neg C_t)| < \epsilon$

First, if a student is highly likely to respond correctly to the next question and the change in prediction is small if the student responds correctly, then we should stop. Second, if a student is highly unlikely to respond correctly to the next question and the change in prediction is small if the student responds incorrectly, then we should stop. Third, if the change in prediction is small no matter how the student responds, then we should stop. All terms in these expressions can be calculated from the predictive student model interface as shown in equations 8 and 9. We call the instructional policy that stops according to these three cases the PREDICTIVE SIMILARITY policy. The function for the PREDICTIVE SIMILARITY policy is provided in algorithm 1

$$P(C_t) = \mathbf{predictCorrect}(s) \quad (8)$$

$$P(C_{t+1}|O_t) = \mathbf{predictCorrect}(\mathbf{updateState}(s, O_t)) \quad (9)$$

5. EXPERIMENTS & RESULTS

We now compare the PREDICTIVE SIMILARITY policy to the MASTERY THRESHOLD policy and see if using different student models as input to the predictive SIMILARITY POLICY yields quantitatively different policies.

Table 1: Functional interfaces for BKT and PFM

	BKT	PFM
startState(...)	$P(L_0)$	$(\alpha_i + \beta_k, \mu_k, \rho_k, 0, 0)$
updateState(s, o)	$P(L_{t+1} \| P(L_t), O_{t+1} = o)$	$\begin{cases} (w, \mu, \rho, s+1, f) & \text{if } o = C \\ (w, \mu, \rho, s, f+1) & \text{if } o = \neg C \end{cases}$
predictCorrect(s)	$P(\neg S)P(L_t) + P(G)(1 - P(L_t))$	$\left(1 + e^{-(w+s\mu+f\rho)}\right)^{-1}$
predictMastery(s)	$P(L_t)$	—

Algorithm 1 PREDICTIVE SIMILARITY policy stop function

```
1:  function STOP(state)
2:      $P(C_t) \leftarrow$ predictCorrect(state)
3:      total $\leftarrow 0$
4:      if $P(C_t) > 0$ then
5:          state$' \leftarrow$ updateState(state, correct)
6:          $P(C_{t+1}|C_t) \leftarrow$ predictCorrect(state$'$)
7:          if $|P(C_t) - P(C_{t+1}|C_t)| < \epsilon$ then
8:              total $\leftarrow$ total $+ P(C_t)$
9:      if $P(C_t) < 1$ then
10:         state$' \leftarrow$ updateState(state, incorrect)
11:         $P(C_{t+1}|\neg C_t) \leftarrow$ predictCorrect(state$'$)
12:         if $|P(C_t) - P(C_{t+1}|\neg C_t)| < \epsilon$ then
13:             total $\leftarrow$ total $+ (1 - P(C_t))$
14:     return total $> \delta$
```

Algorithm 2 Expected Number of Learning Opportunites

```
1:  function EXPOPS(startState)
2:      function EXPOPS$'$(state, $P$(path), len)
3:          if $P$(path) < pathThreshold then
4:              return 0
5:          if len $\geq$ maxLen then
6:              return 0
7:          if stop(state) then
8:              return 0
9:          $P(C) \leftarrow$ predictCorrect(state)
10:         $P(W) \leftarrow 1 - P(C)$
11:         expOpsSoFar $\leftarrow 0$
12:         if $P(C) > 0$ then
13:             $P$(path + c) $\leftarrow P$(path) $* P(C)$
14:             state$' \leftarrow$ updateState(state, $C$)
15:             ops $\leftarrow$ EXPOPS$'$(state$'$, $P$(path + c), len + 1)
16:             expOpsSoFar $\leftarrow$ expOpsSoFar + (ops $* P(C)$)
17:         if $P(W) > 0$ then
18:             $P$(path + w) $\leftarrow P$(path) $* P(W)$
19:             state$' \leftarrow$ updateState(state, incorrect)
20:             ops $\leftarrow$ EXPOPS$'$(state$'$, $P$(path + w), len + 1)
21:             expOpsSoFar $\leftarrow$ expOpsSoFar + (ops $* P(W)$)
22:         return 1 + expOpsSoFar
23:     return EXPOPS$'$((startState, 1, 0))
```

5.1 ExpOps

In order to better understand the differences between two instructional policies we will measure the expected number of problems to be given to students by a policy using the ExpOps algorithm. The ExpOps algorithm allows us to summarize an instructional policy into a single number by approximately calculating the expected number of questions an instructional policy would provide to a student. A naive algorithm takes in the state of the student model and returns 0 if the instructional policy stops at the current state or recursively calls itself with an updated state given each possible observation as shown in equation 10. It builds a synthetic tree of possible observations and their probability using the model state. The tree grows until the policy decides to stop teaching the student. This approach does not require any student data nor does it generate any observation sequences. However, this algorithm may never stop, so ExpOps approximates it by also stopping if we reach a maximum length or if the probability of the sequence of observations thus far drops below a path threshold as shown in algorithm 2. In this paper, we use a path threshold of 10^{-7} and a maximum length of 100.

$$E[Ops] = \begin{cases} 0 & \text{if } \textbf{stop}(s) \\ 1 + \sum_{o \in O} P(O_t = o) E[Ops|o] & \text{otherwise} \end{cases} \quad (10)$$

Lee and Brunskill first introduced this metric to show that individualized models lead to significantly different policies than the general models [12].

5.2 Data

For our experiments we used the Algebra I 2008–2009 dataset from the KDD Cup 2010 Educational Data Mining Challenge [18]. This dataset was collected from students learning algebra I using Carnegie Learning Inc.'s intelligent tutoring systems. The dataset consists of 8,918,054 rows where each row corresponds to a single step inside a problem. These steps are tagged according to three different knowledge component models. For this paper, we used the SubSkills knowledge component model. We removed all rows with missing data. We combined the rows into observation sequences per student and per skill. Steps attached to multiple skills were added to the observation sequences of all attached skills. We removed all skills that had less than 50 observation sequences. Our final dataset included 3292 students, 505 skills, and 421,991 observation sequences.

We performed 5-fold cross-validation on the datasets to see how well AFM, PFM, and BKT models predict student performance. We randomly separated the dataset into five folds with an equal number of observation sequences per skill in each fold. We trained AFM, PFM, and BKT models on four of the five folds and then predicted student performance on

Table 2: Root Mean Squared Error on 5 Folds

Fold	BKT	PFM	AFM
0	0.353	0.364	0.368
1	0.359	0.367	0.371
2	0.358	0.368	0.371
3	0.366	0.369	0.374
4	0.353	0.365	0.368

the leftover fold. We calculated the root mean squared error found in Table 2. Our results show that the three models had similar predictive accuracy, agreeing with prior work.

5.3 Model Implementation

We implemented BKT models as hidden Markov models using a python package we developed. We used the Baum-Welch algorithm to train the models, stopping when the change in log-likelihood between iterations fell below 10^{-5}. For each skill, 10 models with random starting parameters were trained, and the one with the highest likelihood was picked. Both AFM and PFM were implemented using scikit-learn's logistic regression classifier [16]. We used L1 normalization and included a fit intercept. The tolerance was 10^{-4}. We treated an observation connected to multiple skills as multiple observations, one per skill. It is also popular to treat them as a single observation with multiple skill parameters. In the interest of reproducibility, we have published the models used as a python package.[2]

5.4 Experiment 1: Comparing policies

The MASTERY THRESHOLD policy is frequently used as a key part of deciding when to stop showing students questions. However without additional rules, it does not stop if students cannot learn the skill from the current activities. In this experiment we compare the PREDICTIVE SIMILARITY policy to the MASTERY THRESHOLD policy to see if the PREDICTIVE SIMILARITY policy acts like the MASTERY THRESHOLD policy when students learn and stops sooner when students are unable to learn with the given tutoring. We based both policies on BKT models.

We ran ExpOps on each skill for both policies. For the MASTERY THRESHOLD policy, we used the community standard threshold of $\Delta = 0.95$. For the PREDICTIVE SIMILARITY policy, we decided that the smallest meaningful change in predictions is 0.01 and that our confidence should be 0.95, so we set $\epsilon = 0.01$ and $\delta = 0.95$. We then split the skills into those where the BKT model trained on them had semantically meaningful parameters and the rest. A BKT model was said to have semantically meaningful parameters if $P(G) \leq 0.5$ and $P(S) \leq 0.5$. 218 skills had semantically meaningful parameters and 283 did not.[3].

[2]The packages are available at `http://www.jrollinson.com/research/2015/edm/from-predictive-models-to-instructional-policies.html`.

[3]We found similar results for both experiments using BKT models trained through brute force iteration on semantically meaningful values. These results can be found at `http://www.jrollinson.com/research/2015/edm/from-predictive-models-to-instructional-policies.html`

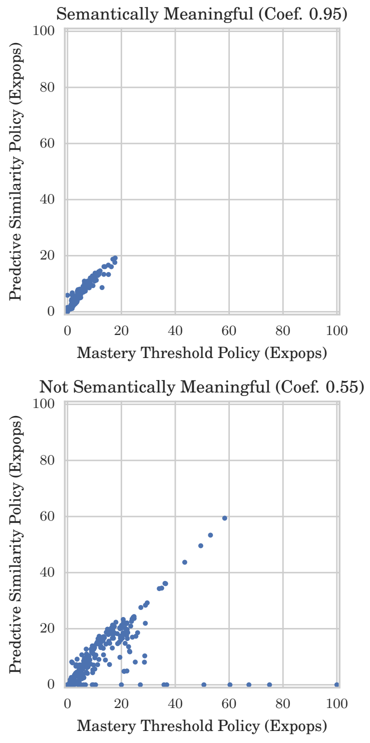

Figure 3: ExpOps using the mastery threshold policy and the predictive similarity policy on skills with and without semantically meaningful parameters.

The Pearson correlation coefficient between ExpOps values calculated using the two policies on skills with semantically meaningful parameters was 0.95. This suggests that the two policies make very similar decisions when based on BKT models with semantically meaningful parameters. However, the Pearson correlation coefficient between ExpOps values calculated using the two policies on skills that do not have semantically meaningful parameters was only 0.55. To uncover why the correlation coefficient was so much lower on skills that do not have semantically meaningful parameters, we plotted the ExpOps values calculated with the MASTERY THRESHOLD policy on the X-axis and the ExpOps values calculated with the PREDICTIVE SIMILARITY policy on the Y-axis for each skill as shown in figure 3. This plot shows that the PREDICTIVE SIMILARITY policy tends to either agree with the MASTERY THRESHOLD policy or have a lower ExpOps value on skills with parameters that are not semantically meaningful. This suggests that the PREDICTIVE SIMILARITY policy is stopping sooner on skills that students are unlikely to learn. The mastery policy does not give up on these skills, and instead teaches them for a long time.

5.5 Experiment 2: Comparing models with the predictive similarity policy

The previous experiment suggests that the PREDICTIVE SIMILARITY policy can effectively mimic the good aspects MASTERY THRESHOLD policy when based on a BKT model. We now wish to see how using models with similar predictive accuracy, but different internal structure will affect it. LFMs

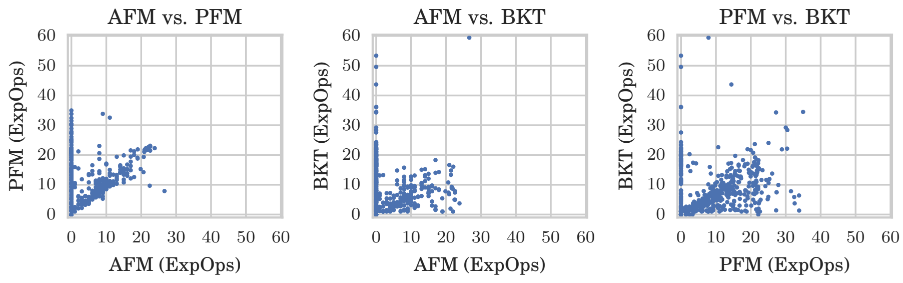

Figure 4: ExpOps plots for the predictive similarity policy using BKT, AFM, and PFM.

Table 3: Correlation coefficients on ExpOps values from policies using BKT, AFM, and PFM.

Models	Coefficient with all skills	Coefficient with skills not stopped immediately
AFM vs. PFM	0.32	0.72
AFM vs. BKT	-0.06	0.44
PFM vs. BKT	0.16	0.46

and BKT models have vastly different structure making them good models for this task. Our earlier results also found that AFM, PFM, and BKT models have similar predictive accuracy. We ran ExpOps on each skill with the PREDICTIVE SIMILARITY policy based both on AFM and PFM. AFM and PFM require a student parameter, which we set to the mean of their trained student parameters. This is commonly done when modeling a student that has not been seen before. We compared the ExpOps values for these two models with the values for the BKT-based PREDICTIVE SIMILARITY policy calculated in the previous experiment.

We first looked at how many skills the different policies immediately stopped on. We found that the BKT-based policy stopped immediately on 31 (6%) of the skills, whilst PFM stopped immediately on 130 (26%) and AFM stopped immediately on 295 (59%).

We calculated the correlation coefficient between each pair of policies on all skills as well as just on skills in which both policies did not stop immediately as shown in table 3. We found that AFM and PFM had the highest correlation coefficient. For each pair of policies, we found that removing the immediately stopped skills had a large positive impact on correlation coefficient. The BKT-based policy had a correlation coefficient of 0.44 with the AFM-based policy and 0.46 with the PFM-based policy on skills that were not immediately stopped on. This suggests that there is a weak correlation between LFM-based and BKT-based policies.

We plotted the ExpOps values for each pair of policies, shown in figure 4. The AFM vs. PFM plot reiterates that the AFM-based and PFM-based policies have similar ExpOps values on skills where AFM does not stop immediately. The BKT vs. PFM plot shows that the PFM-based policy either immediately stops or has a higher ExpOps value than the BKT-based policy on most skills.

To understand why the PFM-based policy tends to either stop immediately or go on for longer than the BKT-based policy, we studied two skills. The first skill is 'Plot point on minor tick mark — integer major fractional minor' on which the BKT-based policy has an ExpOps value of 7.0 and the PFM-based policy has an ExpOps value of 20.7. The second skill is 'Identify solution type of compound inequality using and' on which the BKT-based policy has an ExpOps value of 11.4 and the PFM-based policy immediately stops. We calculated the predictions of both models on two artificial students, one who gets every question correct and one who gets every question incorrect. In figure 5, we plot the prediction trajectories to see how the predictions of the two models compare. In both plots, the PFM-based policy asymptotes slower than the BKT-based policy. Since LFMs calculate predictions with a logistic function, PFM predictions asymptote to 0 when given only incorrect responses and 1 when given only correct responses, whereas the BKT model's predictions asymptote to $P(G)$ and $1-P(S)$ respectively. In the first plot, the PFM-based policy learns at a slower rate than the BKT-based policy, but the predictions do begin to asymptote by the 20^{th} question. In the second plot, the PFM-based policy learns much more slowly. After 25 correct questions, the PFM-based policy's prediction changes by just over 0.1, and after 25 incorrect questions, the PFM-based policy's predictions changes by less than 0.03. In contrast, the BKT-based policy asymptotes over 10 questions to $1 - P(S) = 0.79$ when given correct responses and $P(G) = 0.47$ when given incorrect responses.

This figure also shows how the parameters of a BKT model affect decision making. $P(L_0)$ is responsible for the initial probability of a correct response. $P(S)$ and $P(G)$ respectively provide the upper and lower asymptotes for the probability of a correct response. $P(T)$ is responsible for the speed of reaching the asymptotes. For the PREDICTIVE SIMILARITY policy, the distance between the initial probability of a correct response and the asymptotes along with the speed of reaching the asymptotes is responsible for the number of questions suggested.

6. DISCUSSION

Our results from experiment 1 show that the PREDICTIVE SIMILARITY policy performs similarly to the MASTERY THRESHOLD policy on BKT models with semantically meaningful parameters and suggests the same or fewer problems

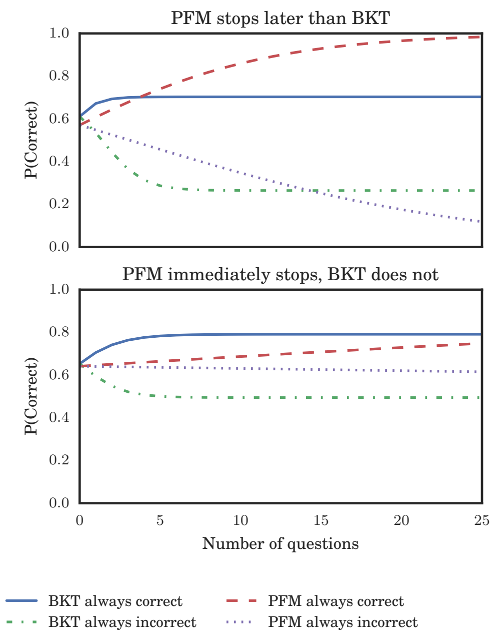

Figure 5: Predictions of BKT models and PFMs if given all correct responses or all incorrect responses on two skills.

on BKT models without semantically meaningful parameters. Thus, this experiment suggests that the two instructional policies treat students successfully learning skills similarly. The lower ExpOps values for the PREDICTIVE SIMILARITY policy provide evidence that the PREDICTIVE SIMILARITY policy does not waste as much student time as the MASTERY THRESHOLD policy on its own. Fundamentally, the MASTERY THRESHOLD policy fails to recognize that some students may not be ready to learn a skill. The PREDICTIVE SIMILARITY policy does not make the same error. Instead, the policy stops either when the system succeeds in teaching the student or when the skill is unteachable by the system. In practice MASTERY THRESHOLD policies are often used in conjunction with other rules such as a maximum amount of practice before stopping. A comparison of such hybrid policies to the PREDICTIVE SIMILARITY policy is an interesting direction for future work. However, it is important to note that such hybrid policies would still require the underlying model to have a notion of mastery, unlike our predictive similarity policy.

The PREDICTIVE SIMILARITY policy can be used to uncover differences in predictive models. Experiment 2 shows that policies based on models with the same predictive power can have widely different actions. AFMs had a very similar RMSE to both PFMs and the BKT models, but immediately stopped on a majority of the skills. An AFM must provide the same predictions to students who get many questions correct and students who get many questions incorrect. To account for this, its predictions do not change much over time. One may argue that this suggests that AFM models are poor predictive models, because their predictions hardly change with large differences in state. Both AFMs and PFMs have inaccurate asymptotes because it is likely that students who have mastered the skill will not get every question correct and that students who have not mastered the skill will not get every question incorrect. This means that these models will attempt to stay away from their asymptotes with lower learning rates. One possible solution would be to build LFMs that limit the history length. Such a model could learn asymptotes that are not 0 and 1.

7. RELATED WORK

Predictive student models are a key area of interest in the intelligent tutoring systems and educational data mining community. One recent model incorporates both BKT and LFM into a single model with better predictive accuracy than both [10]. It assumes that there are many problems associated with a single skill, and each problem has an item parameter. If we were to use such a model in a when-to-stop policy context, the simplest approach would be to find the problem with the highest learning parameter for that skill, and repeatedly apply it. However, this reduces Khajah et al.'s model to a simple BKT model, which is why we did not explicitly compare to their approach.

Less work has been done on the effects of student models on policies. Fancsali et al. [8] showed that when using the MASTERY THRESHOLD policy with BKT one can view the mastery threshold as a parameter controlling the frequency of false negatives and false positives. This work focused on simulated data from BKT models. Since BKT assumes that students eventually learn, this work did not consider wheel-spinning. Rafferty et al. [17] showed that different models of student learning of a cognitive matching task lead to significantly different partially observable Markov decision process policies. Unlike our work which focuses on deciding when-to-stop teaching a single activity type, that work focused on how to sequence different types of activities and did not use a standard education domain (unlike our use of KDD cup). Mandel et al. [13] did a large comparison of different student models in terms of their predicted influence on the best instructional policy and expected performance of that policy in the context of an educational game; however, like Rafferty et al. their focus was on considering how to sequence different types of activities, and instead of learning outcomes they focused on enhancing engagement. Chi et al. [5] performed feature selection to create models of student learning designed to be part of policies that that would enhance learning gains on a physics tutor; however, the focus again was on selecting among different types of activities rather than a when-to-stop policy. Note that neither BKT nor LFMs in their original form can be used to select among different types of problems, though extensions to both can enable such functionality. An interesting direction of future work would be to see how to extend our policy to take into account different types of activities.

Work on when-to-stop policies is also quite limited. Lee and Brunskill [12] showed that individualizing student BKT models has a significant impact on the expected number of practice opportunities (as measured through ExpOps) for a significant fraction of students. Koedinger et al. [11] showed that splitting one skill into multiple skills could significantly improve learning performance; this process was done by human experts and leveraged BKT models for the policy design. Cen et al.[4] improved the efficiency of student learning by noticing that AFM models suggested that some skills were significantly over or under practiced. They created new BKT parameters for such skills and the result was a new tutor that helped students learn significantly faster. However, the authors did not directly use AFM to induce policies, but rather used an expert based approach to transform the models back to BKT models, which could be used with existing mastery approaches. In contrast, our approach can be directly used with AFM and other such models.

Our policy assumes that learning is a gradual process. If you were to instead subscribe to an all-at-once method of learning, you could possibly use the moment of learning as your stopping point. Baker et al. provide a method of detecting the moment at which learning occurs [2]. However, this work does not attempt to build instructional policies.

8. CONCLUSION & FUTURE WORK

The main contribution of this paper is a when-to-stop policy with two attractive properties: it can be used with any predictive student model and it will provide finite practice both to students that succeed in learning a given skill and to those unable to do so given the presented activities.

This policy allowed us for the first time to compare common predictive models (LFMs and BKT models) in terms of their predicted practice required. In doing so we found that models with similar predictive error rates can lead to very different policies. This suggests that if they are to be used for instructional decision making, student models should not be judged by predictive error rates alone. One limitation of the current work is that only one dataset was used in the experiments. To confirm these results it would be useful to compare to other datasets.

One key issue raised by this work is how to evaluate instructional policy accuracy. One possible solution is to run trials with students stopping after different numbers of questions. The student would take both a pre and post-test, which could be compared to see if the student improved. However, such a trial would require many students and could be detrimental to their learning.

There is a lot of room for extending this instructional policy. First, we would like to incorporate other types of interactions, such as dictated information ("tells") or worked examples, into the PREDICTIVE SIMILARITY policy. This would give student models more information and hopefully lead to better predictions. Second, the PREDICTIVE SIMILARITY policy is myopic, and we are interested in the effects of expanding to longer horizons. Third, we are excited about extending this instructional policy to choosing between skills. Instead of stopping when there is a high probability of predictions not changing, the instructional policy could return either the skill that had the highest chance of a significant change in prediction, or the skill with the highest expected change in prediction.

9. REFERENCES

[1] V. Aleven, B. M. McLaren, J. Sewall, and K. R. Koedinger. The cognitive tutor authoring tools (ctat): preliminary evaluation of efficiency gains. In *ITS*. Springer, 2006.

[2] R. S. Baker, A. B. Goldstein, and N. T. Heffernan. Detecting learning moment-by-moment. *IJAIED*, 21(1), 2011.

[3] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *ITS*. Springer, 2006.

[4] H. Cen, K. R. Koedinger, and B. Junker. Is over practice necessary?-improving learning efficiency with the cognitive tutor through educational data mining. *FAIA*, 158, 2007.

[5] M. Chi, K. R. Koedinger, G. J. Gordon, P. W. Jordan, and K. VanLehn. Instructional factors analysis: A cognitive model for multiple instructional interventions. In *EDM*, 2011.

[6] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *UMAP*, 4(4), 1994.

[7] M. H. Falakmasir, Z. A. Pardos, G. J. Gordon, and P. Brusilovsky. A spectral learning approach to knowledge tracing. In *EDM 2013*, 2010.

[8] S. E. Fancsali, T. Nixon, and S. Ritter. Optimal and worst-case performance of mastery learning assessment with bayesian knowledge tracing. In *EDM*, 2013.

[9] Y. Gong, J. E. Beck, and N. T. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *ITS*, 2010.

[10] M. Khajah, R. M. Wing, R. V. Lindsey, and M. C. Mozer. Integrating latent-factor and knowledge-tracing models to predict individual differences in learning. *EDM*, 2014.

[11] K. R. Koedinger, J. C. Stamper, E. A. McLaughlin, and T. Nixon. Using data-driven discovery of better student models to improve student learning. In *AIED*. Springer, 2013.

[12] J. I. Lee and E. Brunskill. The impact on individualizing student models on necessary practice opportunities. In *EDM*, 2012.

[13] T. Mandel, Y.-E. Liu, S. Levine, E. Brunskill, and Z. Popovic. Offline policy evaluation across representations with applications to educational games. AAMAS, 2014.

[14] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *UMAP*. Springer, 2010.

[15] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger. Performance factors analysis–a new alternative to knowledge tracing. 2009.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 12, 2011.

[17] A. Rafferty, E. Brunskill, T. Griffths, and P. Shafto. Faster teaching by POMDP planning. In *AIED*, 2011.

[18] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Algebra 1 2008-2009. challenge data set from kdd cup 2010 educational data mining challenge. find it at http://pslcdatashop.web.cmu.edu/kddcup/downloads.jsp.