Discovery of Student Strategies using Hidden Markov Model Clustering

Benjamin Shih
Machine Learning Department
Carnegie-Mellon University
Pittsburgh, PA 15213
shih@cmu.edu

Kenneth R. Koedinger
HCI Institute
Carnegie-Mellon University
Pittsburgh, PA 15213
koedinger@cmu.edu

Richard Scheines
Department of Philosophy
Carnegie-Mellon University
Pittsburgh, PA 15213
scheines@cmu.edu

Abstract

Students interacting with educational software generate data on their use of software assistance and on the correctness of their answers. This data comes in the form of a time series, with each interaction as a separate data point. This data poses a number of unique issues. In educational research, results should be interpretable by domain experts, which strongly biases learning towards simpler models. Educational data also has a temporal dimension that is generally not fully utilized. Finally, when educational data is analyzed using machine learning techniques, the algorithm is generally off-the-shelf with little consideration for the unique properties of educational data. We focus on the problem of analyzing student interactions with software tutors. Our objective is to discover different strategies that students employ and to use those strategies to predict learning outcomes. For this, we utilize hidden Markov model (HMM) clustering. Unlike some other approaches, HMMs incorporate the time dimension into the model. By learning many HMMs rather than just one, the result will include smaller, more interpretable models. Finally, as part of this process, we can examine different model selection criteria with respect to the models predictions of student learning outcomes. This allows further insight into the properties of model selection criteria on educational data sets, beyond the usual cross-validation or test analysis. We discover that the algorithm is effective across multiple measures and that the adjusted-R^2 is an effective model selection metric.

1 Introduction

Educational software is an increasingly important part of human education. Many schools use educational software as a major component in classroom curricula and individuals are using specialized software for diverse purposes such as second-language acquisition and extracurricular tutoring. Likewise, the analysis of data from educational software is also a growing field. Individuals interacting with an educational system generate sizable quantities of time-stamped data, ranging in granularity from individual mouse movements to attempted solutions. This data offers insight into an individual's underlying cognitive processes and has the potential to guide future educational interventions.

However, the temporal-sequential aspect of educational data is frequently underutilized. In brief, the usual approach to analyzing educational data is to compute a set of features, e.g. average number of attempts, and to then input those features into an off-the-shelf machine learning algorithm in an attempt to predict learning between separately administered pre-tests and post-tests. These features usually do not incorporate a significant temporal aspect aside from the student's response time, i.e. the time between a stimulus, such as a problem statement, and the response, such as a solution

attempt. By not incorporating the entirety of the data, particularly the ordering of actions, such analyses fail to realize the data's full potential.

In a computer tutoring system, the log data may be treated as a time-series with variable intervals of observation. If the performance on each task is conditionally independent given the student, i.e. solving a math problem step does not require successful solutions to prior steps, then each task can be treated as a separate sequence of observations. Thus, each step or sequence can be considered a segment of a time-series. For example, if a student requests help at the beginning of the step and then attempts solutions until they solve the step, that is likely a different strategy than if a student attempts to solve the step and, upon failing, requests help.

In this paper, the concept of student strategies is instantiated by hidden Markov models (HMMs). HMMs are graphical models which treat observed data as an ordered sequences of symbols. HMMs will be discussed in more detail in the Background section; however, the primary observation is, by learning many different HMMs from educational data, each HMM can be treated as a model of a different student strategy. Prior work in educational data mining has largely focused on learning single, complicated models that describe all possible student behaviors. The advantages of collections of HMMs are four-fold: they have disjoint observations, the observations are ordered, they are much easier to interpret, and they provide extremely accurate predictions. Further, the algorithm we propose offers several advantages over standard HMM clustering algorithms: it has adaptive parameters, biases strongly towards smaller models, and can incorporate external measures.

The remainder of this paper is divided into several sections. The Background section covers the relevant machine learning literature. The Method section describes a number of unique properties to our method, including data preprocessing. The Data section describes the two data sets used in this study. The Results section includes both model predictions and interpretations, reported across multiple parameterizations and data sets.

2 Background

A hidden Markov model (HMM) is a set of unobserved states governed by the Markov property where the relationship between an unobserved state and the actual observations derives from a probability distribution. In short, an HMM is a probabilistic function of an unobserved Markov chain.[3] More specifically, graphical models are a series of discrete states with the transitions between states determined by a probability matrix. The Markov property requires that the state of a model at time t is exlusively dependent on the state of the model at time $t-1$. No prior states are relevant. An HMM additionally requires that the probability of observing a symbol at time t depends only on the model's state at time t. For this paper, symbols will correspond to types of student actions in a computer tutoring system. Observed symbols are called emissions in the HMM literature and, in this paper, all HMMs will use discrete emissions.

In general, the following parameters uniquely describe an N-state discrete HMM M: the states S_i ($0 \leq i < N$); the initial probability of M starting in state S_i, written as π_i; the transition probability from state S_i to S_j, t_{ij}; the alphabet of symbols Σ; and the emission probability of a symbol σ_k for state i ($\sigma_k \in \Sigma$, $0 \leq k < |\Sigma|$). A detailed introduction to HMMs is available in Rabiner [8] and a thorough treatment of inference in HMMs is available in Cappé et. al.[4]

For this paper, a series of observed symbols will be called a sequence. The Baum-Welch algorithm allows for efficient estimation of the parameters of an HMM given a set of observed sequences.[2] In short, given that a set of student sequences are associated with an HMM, the Baum-Welch algorithm can relearn the parameters of that HMM to better fit the observed data. Given an HMM and an observed sequence, the Forward-Backward algorithm can calculate the probability of that observed sequence conditioned on the parameters of the HMM. These two algorithms are the core of any HMM clustering algorithm.

2.1 Clustering

The Baum-Welch algorithm only learns the parameters for a single HMM, but the extension to learning sets of HMMs with clustering is trivial. First, define a set of HMMs M as a collection C. Given a set of observed sequences Q where each sequence $q_i \in Q$ is a series of observed symbols

$\sigma \in \Sigma$, a partition of the observations Q for a fixed number of partitions K is $P = (P_1, \ldots, P_K)$, where $\forall q_i \in Q \exists k$ s.t. $q_i \in P_k$. The goal of a clustering algorithm is to maximize the objective function:

$$F(P) = \prod_{k=1}^{K} \prod_{i \in P_k} Pr(q_i|M_k) \tag{1}$$

where $Pr(q_i|M_k)$ is the probability of observing a sequence q_i given the paramaterization of a model M_k. Replacing the probability with the likelihood and taking the log of the objective function, the result is a standard clustering objective function:

$$f(P) = \sum_{k=1}^{K} \sum_{i \in P_k} l(q_i|M_k) \tag{2}$$

Objective functions of this form have long been optimized with Expectation-Maximization (E-M) algorithms and, for HMM clustering, given an initial set of HMMs, the usual algorithm is of the form:

Input: sequence set Q, model count K, initial collection C^0 of models M_k^0, $0 \leq k < K$
Output: collection C
iteration $t = 0$;
while *termination criteria not satisfied* **do**
  iteration $t = t + 1$;
  create partition sets P_k, $0 \leq k < K$;
  foreach *sequence* $q_i \in Q$ **do**
    find the best model $k = \arg\max_k l(q_i|M_k^{t-1})$;
    assign sequence q_i to partition P_k^t;
  end
  foreach $M_k^{t-1} \in C^{t-1}$ **do**
    M_k^t = Baum-Welch(M_k^{t-1},P_k^t);
    assign M_k^t to C^t
  end
end
return C^t;

Algorithm 1: HMM-Cluster

Possible termination criteria include any of the following being below some threshold ϵ:

- Change in log-likelihood: $\Delta \sum_{q_i \in Q} l(q_i|C)$
- Number of changed labels: $\sum_{q_i \in Q} \delta(q_i \in P_k^{t-1}, q_i \in P_l^t, k \neq l)$
- Maximum change in the size of a partition: $\max_{0 \leq k < K} \Delta |P_k|$

HMM-Cluster converges to a local maximum because Baum-Welch converges and, for any given clustering iteration, both relearning the parameters and repartitioning the observed sequences are monotonic operations with respect to the likelihood. Further, *HMM-Cluster* will never change the size of the collection C or the number of states per model N. In general, if seeded with a collection of initial models (chosen randomly or with a heuristic), the basic structure of each model will remain constant; only the parameters and partitions will change.

There have been many prior uses of similar E-M HMM clustering algorithms. One of the earliest uses was by Rabiner et. al. for word recognition. [9] More recently, the predominant domain has been gene expression data. [11] Others have also looked at HMM clustering algorithms for analyzing text cohesion [5]. There have been many improvements suggested, including processing sequences with dynamic time warping, [7] more principled ways of generating candidate HMMs,[12] and

Student	Step	Action	Duration
S01	ARCS-3 ARC-EG-MEASURE	Attempt	11.446
S01	ARCS-3 ARC-EG-MEASURE	Attempt	4.847
S01	ARCS-3 ARC-EG-MEASURE	Attempt	19.588
S01	ARCS-3 ARC-EG-MEASURE	Attempt	6.179
S01	ARCS-3 ARC-EG-MEASURE	Attempt	10.535

Table 1: Example Tutor Step

using spectral clustering instead of partition-based clustering. [6] Jebara et. al.'s work on spectral clustering with HMMs is especially important as a potential avenue for future work. [6]

Some of the prior work on E-M HMM clustering uses fixed values for K and for the number of states (N) per initial model[7]. Other examples use fixed initial values for K, but allow the merging or splitting of clusters. For example, Schliep uses "model surgery", which merges and splits clusters based on the total size of each cluster[11]. However, it is unclear which merge/split criteria are optimal. We will instead use *HMM-Cluster* as a subroutine for another algorithm, and so will limit it to fixed values of K and N.

3 Data

We consider two data sets extracted from log files of the Geometry Cognitive Tutor. In the tutor, students are presented with a geometry problem and several empty text fields. A step in the problem requires filling in a text field. The fields are arranged systematically on each problem page and might, for example, ask for the values of angles in a polygon or for the intermediate values required to calculate the circumference of a circle.

Both data sets originate in earlier experimental studies, though only the control groups for each study will be used.

In each data set, a problem is defined as a series of steps and each step as a series of transactions. A student transaction is defined by the following four-tuple: $\langle$ Student,Step,Action,Duration $\rangle$. An action can be either an "Attempt" or "Help Request". Each data set consists of a series of these transactions, categorized by step and student. An example step is shown in Table 1.

*02*This data set originates in an experiment published in 2002. [1] The control condition includes 21 students and 57204 actions divided into 3740 steps.

*06*This data set originates in an experiment published in 2006. [10] The control condition includes 16 students and 16374(FIX) actions divided into 5217(FIX) steps.

Both data sets are similar in that they cover the same geometry units and use the same general interface, though there are some differences in both domain content and interface layout. The most important differences in the data lie in the students' distribution of actions and steps. In the 06 data, students exhibit far fewer actions per step, which complicates any direct comparison between results for the two data sets.

4 Method

A student action is defined by the following four-tuple: $\langle$ Student,Step,Action,Duration$\rangle$. Once actions are conditioned on students and steps, what remains is the tuple $\langle$ Action,Duration$\rangle$. While it is technically possible to directly analyze the data in this two-dimensional, partially continuous space, the results are difficult to interpret. Instead, consider a threshold of τ seconds which divides actions into "fast" and "slow" actions. There exists a mapping from the bivariate $\langle$Action,Duration$\rangle$ tuple to a single four-category variable, shown in Table 2.

Guessing and Trying are fairly self-explanatory: a guess is a suspected attempt to solve using the system's correctness-feedback while a try is a suspected attempt to solve using actual problem-solving techniques. A drill is rapidly requesting hints, probably without reading them, either to get

	Attempt	Help Request
Fast	*G*uess	*D*rill
Slow	*T*ry	*R*eason

Table 2: Mapping from $\langle$ Action,Duration$\rangle$ to one variable

a more concrete hint or to reach the solution (final hint). A reasoned hint request is when the student is assumed to read the hint. These four categories will be abbreviated G, D, T, and R.

Define a sequence $q_i^s \in Q$ to be the i-th step performed by a student s. Each sequence is then a series of observations from $\Sigma = \{G, D, T, R\}$. After applying some learning algorithm, e.g. a randomly-seeded *HMM-Cluster*, Q will be partitioned such that each q_i^s is assigned to one model M_k. Let $q_k = \{q_i^s | q_i^s \in M_k\}$ be the set of all sequences in partition k. There are then several approaches for estimating the relationship between the partition P and external measures of learning (denoted by G):

- Maximum Correlation with the Absolute Count (per model): $\max_{0 \leq k < K} \rho(q_k, G)$
- Maximum Correlation with the Ratio (per model): $\max_{0 \leq k < K} \rho(\frac{q_k}{(\sum_{0 \leq l < K} q_l)}, G)$
- Linear Regression, i.e. fitting a regression model and calculating R^2 or adjusted R^2

For this paper, all learning gain measures are pre-test to post-test learning gain, by student. There is, however, an additional subtlety with respect to G: it can be measured in absolute terms $(\text{post} - \text{pre})$, in terms of adjusted gain ($\frac{(\text{post}-\text{pre})}{(1-\text{pre})}$), or in terms of z-scores. This paper will focus on the adjusted gain, as the data sets used are significantly non-normal (floor effects) and absolute learning gain is not an interpretable concept between curricula.

Within this framework, *HMM-Cluster* has several problems:

- *HMM-Cluster* gets trapped in local maxima that can be significantly worse than the global maximum, potentially requiring many random restarts to approximate the best fit.
- The choice of values for K and N will determine the effectiveness of *HMM-Cluster*. If they are too large, C will overfit Q; if they are too small, no collection C will fit Q well.
- By default, direct measures of fit will bias towards larger values of K and N. The resulting collections will not only overfit, but be very difficult to interpret.
- K and N are fixed and non-adaptive.

In principle, a better algorithm would search over values of K and N with a bias towards smaller and fewer models, allowing for higher test-set accuracy and easier interpretation of models. It would also do this in a way that would allow the algorithm to incrementally find better local maxima. One such algorithm is *Stepwise-HMM-Cluster*, which is to *HMM-Cluster* what stepwise regression is to normal regression. Let *New-HMMs*(k,n) be shorthand for a function that returns k new HMMs with n states each.

A critical step in *Stepwise-HMM-Cluster* is the selection of "good models" from a collection C. This choice is determined with regression. In this paper, we use forward stepwise linear regression. The number of sequences classified by each model for each student is used as the input (along with the total sequences per students) and the learning gain is used as the output. For example, if M_0 was the best fitting model on 20 sequences for Student 0 and 10 sequences for Student 1, and M_1 was the best fitting model on 15 sequences for Student 0 and 25 sequences for Student 1, the regression approximations would be:

$$\begin{aligned} G_0 &\approx \beta_0 + \beta_1 * 20 + \beta_2 * 10 \\ G_1 &\approx \beta_0 + \beta_1 * 15 + \beta_2 * 25 \end{aligned}$$

Input: sequence set Q,student set S, student learning gains G
Output: collection C
iteration $t = 0$;
models $K = 2$;
states $N = 2$;
collection C^0 = New-HMMs(K,N);
while *termination criteria not satisfied* **do**
  iteration $t = t + 1$;
  relearn C^{t-1} = HMM-Cluster(Q,K,C^{t-1});
  create partition sets P_k^s, $0 \leq k < K$, $0 \leq s < S$;
  foreach *sequence* $q_i \in Q$ **do**
    find the best model $k = \arg\max_k l(q_i|M_k^{t-1})$;
    let $s_i \in S$ bet the student acting in sequence q_i;
    assign sequence q_i to partition P_k^s;
  end
  significant models R = Regression(G,[P_0,...,P_K]);
  foreach $M_k^{t-1} \in C^{t-1}$ **do**
    if $M_k^{t-1} \in R$ **then**
      assign M_k^{t-1} to C^t;
    end
  end
  if *model count criteria satisfied* **then**
    $K = K + 1$;
  end
  if *state count criteria satisfied* **then**
    $N = N + 1$;
  end
  $C^t = C^t \cup$ New-HMMs($K - |C^t|$,N);
end
return C^t;

Algorithm 2: Stepwise-HMM-Cluster

τ	MAC(Train)	MAC(Test)	MRC(Train)	MRC(Test)	Adj-R^2(Train)	R^2(Test)	Adj-R^2T(Train)	R^2T(Test)
6	0.62	0.73	0.7	0.73	0.29	0.33	0.29	0.33
8	0.66	0.72	0.76	0.67	0.31	0.25	0.35	0.25
10	0.67	0.72	0.75	0.66	0.33	0.29	0.37	0.29

Table 3: Best possible metric scores for 02

There are a variety of possible termination criteria, model count incrementing criteria, and state count incrementing criteria. In this paper, we will use the simplest: given a constant limit on consecutive iterations ι, a constant limit on the number of models κ, and a constant limit on the number of states per model ν, if the overall regression fit (adjusted R^2) does not improve significantly after at least I consecutive iterations, increment the number of states N used by *New-HMMs*. If $N = \nu$ already, increase the number of allowed states K and reset $N = 2$. Terminate the algorithm when there have been ι consecutive non-significant iterations, but both $K = \kappa$ and $N = \nu$.

In general, *New-HMMs* can generate candidate HMMs using heuristics or through an active learning paradigm, but for our purposes, candidate HMMs are generated with randomly chosen parameters. We will use a value of $\iota = 2$, allowing for one extra "bad" result before continuing to the next level of complexity.

5 Results

There are five metrics we will consider for estimating the quality of a collection:

- Bayesian Information Criterion (BIC)
- Maximum Absolute Correlation (MAC): $\max_{0 \leq k < K} \rho(q_k, G)$
- Maximum Ratio Correlation (MRC): $\max_{0 \leq k < K} \rho(\frac{q_k}{(\sum_{0 \leq l < K} q_l)}, G)$
- (Adjusted-)R^2 of a linear regression without total count (R2)
- (Adjusted-)R^2 of a linear regression with total count (R2T)

The first metric (BIC) forms a baseline. BIC offers a cheap, off-the-shelf method for controlling for overfitting, but it also has known biases and limitations. However, it is a standard method that still sees frequent use, and is included here for comparison.

The correlations, MAC and MRC, describe the best prediction of any single HMM in a collection. They are useful for three reasons. First, it is easier to interpret a single HMM, rather than interpreting a mathematical function over a set of models (such as with a regression). Second, it is easier to construct an intervention from only one model at a time. Finally, as will be shown later, in most cases, two (almost linearly dependent) model-partitions dominate other model-partitions in the same collection. When two models classify the majority of observed sequences, there is little difference between the maximum correlation and the best R^2 value. However, MAC and MRC also "cheat". While they are ostensibly based on the correlation of only one HMM with learning gain, that one HMM is part of a greater collection. It classifies a subset of sequences because other HMMs classify some sequences better; without the other models, even a high-correlating HMM is useless as its partition would contain all observed sequences. The general interpretation of these collections will be that two models are usually the "best" classifiers, but that other models serve to remove specific, non-useful sequence types from those models' partitions.

R^2 and R^2T are the metrics that best match the construction process for collections in *Stepwise-HMM-Cluster*. Linear regression is the simplest method of incorporating information from all models in a collection into a single prediction. In general, R^2T will provide the best fit on training data, in part because *Stepwise-HMM-Cluster* uses R^2T to select optimal collections. R^2 and R^2T are adjusted for the number of parameters when reported on training data, but are not adjusted when reported on test data. In general, the adjustment will not be noted here after, except in table column headers. Instead, the difference will be indicated by R^2(Train) versus R^2(Test), and similarly for R^2T.

τ	Models	Max States	MAC(Train)	MAC(Test)	MRC(Train)	MRC(Test)	Adj-R^2(Train)	R^2(Test)	Adj-R^2T(Train)	R^2T(Test)
6	5	3	0.61	0.58	0.7	0.71	0.29	0.24	0.29	0.24
8	5	4	0.58	0.51	0.73	0.59	0.31	0.18	0.31	0.11
10	4	4	0.57	0.54	0.71	0.53	0.325	0.23	0.3125	0.25

Table 4: Best collections for 02, chosen by R^2(Train)

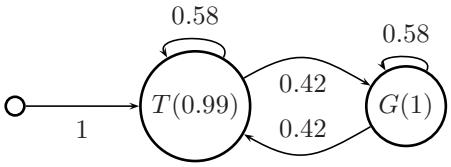

Figure 1: Dominant model for $\tau = 6$, 02 data

Table 3 is taken from the 02 data and details the maximum values for each gain metric across different values of the time threshold τ, for both the training data and withheld (20%) test data. Each row represents one run of *Stepwise-HMM-Cluster* at a fixed value of τ.

The primary purpose of Table 3 is to show the optimal, best-case metrics, especially for R^2(Test) and R^2T(Test). In practice, we have to choose a single collection for future predictive purposes, and cannot select using our test measure. However, the table shows our optimal test-set result, if we could choose perfectly, is about 0.33.

Table 4, also taken from 02 data, shows the results from choosing the "best" collections, i.e. the collection, per choice of τ, with the highest adjusted R2(Train) score. It also shows the number of models in the best collection and the maximum number of states per model in that collection. The R^2(Test) and R^2T(Test) columns are particularly important as they represent test-set validity. For $\tau = 6$ and $\tau = 8$, the R^2(Test) results represent at least 80% of the best possible test-set prediction.

Additionally, where R^2(Train) and R^2T(Train) are equal, there was no significant improvement from adding the total count to the stepwise regression. The table shows that, for the 02 data, this is usually the case. However, when $\tau = 10$, R^2(Train) and R^2T(Train) are not identical. From empirical tests, it is generally true that on the 02 data, selecting the best collection with R^2(Train) is more effective than selecting with R^2T(Train); this will not hold for the 06 data.

The test-set R^2 and R^2T fits are all reasonable, with fairly predictive models for $\tau = 6$ and $\tau = 10$. A 0.25 R^2 value is equivalent to a 0.5 correlation, which is very respectable for predicting overall unit pre-post learning gain using only logged tutor data. Additionally, this methodology uses no domain knowledge, which makes the result more impressive and generalizable.

To better understand the results, it's worth noting that in all three optimal collections shown, there is a single dominant HMM that classifies at least 40% of the sequences and has a 0.5 or better correlation with learning gain (when computed on the training data). In all three collections, the model generates predominantly Tries and Guesses. Shown below are two examples of dominant HMMs for $\tau = 6$ and $\tau = 8$. The emissions are shown as *Symbol* (Probability), e.g. $T(0.8)$ emits a Try symbol with 80% probability. Only symbols emitted with at least a 20% probability are shown.

In both cases, the models emit Tries and Guesses with high probability and emits both symbols equally often (over the course of many sequences). One possible explanation is that these models actually select short sequences where the student already knows the answer and thus, where they can solve the step in one try. This would indicate that the student has learned the material. However, a quick correlation between the frequency of first-try-correct sequences and learning gain nets a -0.24 correlation. Instead, an alternative interpretation of these models is that they represent a persistence-trait. Students who attempt to solve repeatedly are more likely to learn the material than those that rely on hints. This is borne out by the duration-agnostic nature of the model, which emits both Tries and Guesses, as well as the lack of hints as a probable emission.

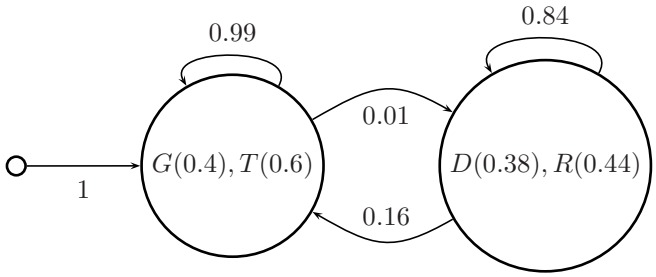

Figure 2: Dominant model for $\tau = 8$, 02 data

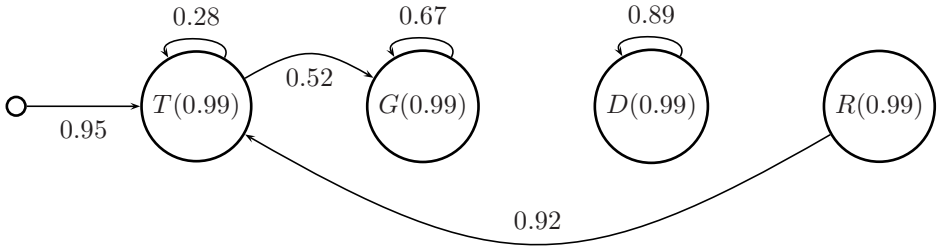

Figure 3: Try-Guess model for $\tau = 10$, 02 data

However, this raises a conflict between the results and common sense. The models shown in Figure 1 and Figure 2 have a high probability of emitting a sequence of type TGGGGG, i.e. a single Try followed by many Guesses. This is, in the educational literature, considered a very poor learning behavior. Intuitively, it represents a failed attempt to solve followed by repeated, unthinking guessing. This disagreement can be resolved by noting that no single model in any collection can be interpreted alone. Each model exists only as part of the entire collection, and as such, other models in the collection can remove specific degenerate sequences. Take the $\tau = 10$ collection as an example. It contains a model, shown below, that has a high probability of emitting Try-Guess type sequences.

The general interpretation of these results is that students learn more when using persistence-type strategies, so long as they don't just guess repeatedly. Interestingly, this is largely independent of the choice of threshold τ. This suggests that, at least for this range of τ and for persistent-type strategies, the duration is not relevant *so long as* students do not engage in repeated guessing. This issue will be further addressed in the Conclusions section.

It's possible, however, that selecting with a different metric, such as BIC, would yield better collections. Figure 4 shows the BIC scores for collections on 02 data for $\tau = 6$, plotted against the Adjusted-R^2(Train) and R^2(Test) metrics. Figure 4 clearly shows that the BIC score has a non-linear relationship with learning gain prediction, on training or test data, and that thus BIC is a poor metric for choosing collections of HMMs. There may exist better metrics than BIC or R^2(Train) for this task, but it's clear that R^2(Train)'s success is non-trivial.

Still, this result may be data-set dependent, so an experiment on a second data set is essential. For this, there is the 06 data, which, compared to the 02 data, has different students in different classrooms using a different version of the Geometry Cognitive Tutor. Unfortunately, a direct application of the above method is ineffective: if the best collections are chosen using the R^2(Train) metric, none of them have decent test-set results. In fact, none of the top 5 R^2(Train) collections for each choice of τ have test-set performance on par with the 02 data results.

Practically, there are reasons to suspect R^2T(Train) is a better metric for the 06 data, regardless of the test-set performance. There is more variation in the number of sequences between students in the

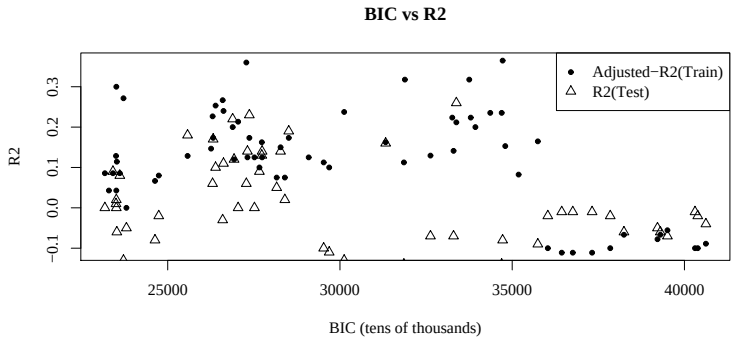

Figure 4: BIC scores versus R^2(Train) and R^2(Test), $\tau = 6$, 02 data

τ	Models	Max States	Adj-R^2(Train)	R^2(Test)	Adj-R^2T(Train)	R^2T(Test)
6	3	8	0.26	0.26	0.35	0.26
8	3	5	0.18	0.19	0.28	0.22
10	6	5	0.44	0.19	0.49	0.17

Table 5: Best collections for 06, chosen by R^2T(Train)

06 data than in the 02 data, which suggests a greater importance for the total number of sequences in any regression. Unfortunately, at first glance, selecting by R^2T(Train) does not improve results. The top collections still have poor test-set performance, in part because they have 7 or 8 models apiece. However, within the top 3 collections for each choice of τ, there is a collection with a nearly 0.2 R^2 score on test-data. These collections all have one thing in common: fewer models. They are shown in Table 5.

For $\tau = 6$ and $\tau = 8$, the other R^2T(Train) collections have at least 7 models, so if a stricter complexity penalty was used, it would likely select the 3-model collections. For $\tau = 10$, this is not as clearly evident. However, putting aside the choice of collection, these best collections exhibit the same overall form as was seen in the 02 collections. For example, the dominant model for $\tau = 6$ is shown in Figure 5. The interpretation of this model is the same as the models from Figure 1 and Figure 2, namely that of persistence. The presence of this model in the best collections across data sets and values of τ is highly suggestive that persistent-attempts is a very important strategy.

In general, across all collections shown, hint-heavy strategies are negatively associated with learning. However, this should not be interpreted too broadly. Many of the more complex collections (4 or 5 models) contain a ”noise” model which generates most possible sequences with nearly uniform probability. Thus, the hint-heavy strategies are actually very focused on specific types of hint re-

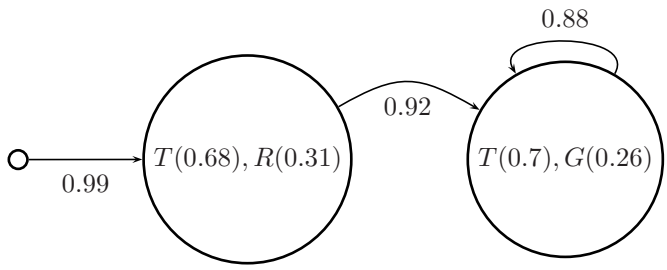

Figure 5: Dominant model for $\tau = 6$, 06 data

quests, usually Drill requests. Some positive-learning models involve hints as well. For example, in Figure 5, there is actually a 30% chance of generating a Reason action as the first action, before a series of attempts.

6 Conclusions

Using a traditional HMM clustering algorithm with fixed values of K, the number of models, and N, the number of states per model, it is possible to find collections of HMMs that predict learning. These models not only predict learning, but because the HMMs are relatively small, they are human-interpretable as classes of student strategies. However, this basic learning algorithm requires many random restarts, and it's unclear how to prevent the algorithm from "fishing" for results and thus overfitting.

An alternative approach is to iteratively increase the values of K and N, keeping at each iteration an optimal collection of HMMs from prior iterations. This approach, called *Stepwise-HMM-Cluster*, requires fewer clusterings to converge to a highly predictive model. Further, it avoids pre-hoc choices for K and N, biases strongly towards smaller models, provides better test-set predictions, and incorporates external measures of learning gain.

We showed that using *Stepwise-HMM-Cluster* found collections with high training-set prediction accuracy, even after adjusting for the number of models in a collection. Further, for the 02 data, withholding part of the data as a test-set still resulted in accurate predictions, on the order of a 0.5 correlation. For the other data set, a more heavily penalized selection criterion also gave similar correlations. This algorithm satisfies the primary goals of an educational data mining method: it produces interpretable models, provides good fits across data sets, and not only fits the tutor data, but predicts actual learning outcomes.

Additionally, generalization from a learning sciences perspective is not a simple matter of successful predictions on test data: it requires the production of general learning principles that can be applied independently of any given parametric model. *Stepwise-HMM-Cluster* produced such a general principle. Our results provide a strong argument that hint-scaffolding as it is presently used is not actually very effective and that most learning results from persistent attempts to solve. This suggests a new paradigm for tutoring system design that emphasizes attempts and provides hints or worked examples only when strictly necessary to keep the student advancing in the curriculum.

There are several directions for future work. First, prior research has shown that spectral clustering of HMMs tends to outperform E-M clustering of HMMs. Spectral clustering had an additional advantage of returning only a single result. However, it is not clear how to constrain spectral clustering algorithms to produce interpretable models. Second, for *Stepwise-HMM-Cluster*, the choice of τ is a difficult one. A more principled method for choosing the threshold value would be useful, but more importantly, an understanding of the role of time in student-tutor action sequences would be invaluable. For example, it's possible that there are three types of actions ("Fast","Medium", and "Slow"), or that there are different optimal thresholds for different types of actions. One approach to resolving this question would be to learn models that use continuous distributions for emitting durations and to use those distributions to set the thresholds. Thresholding the duration is still essential in the long-run to produce general learning principles.

There are also questions of generalization. First, does this method or these models generate to other domains? Can a single collection be learned such that it can be used (perhaps after some data preprocessing) to predict learning in other data sets? And, of course, can the principle of persistence be applied to either future tutor designs or to the creation of educational interventions?

References

[1] Vincent Aleven and Kenneth R. Koedinger. An effective meta-cognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive Science*, page 147179, 2002.

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 1970.

[3] C. J. Burke and M. Rosenblatt. A markovian function of a markov chain. *The Annals of Mathematical Statistics*, 1958.

[4] Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer, 2005.

[5] Pascale Fung, Grace Ngai, and Chi-Shun Cheung. Combining optimal clustering and hidden markov models for extractive summarization. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering*, pages 21–28, 2003.

[6] Tony Jebara, Yingbo Song, and Kapil Thadani. Spectral clustering and embedding with hidden markov models. In *Proceedings of the European Conference on Machine Learning*, 2007.

[7] Tim Oates, Laura Firoiu, and Paul R. Cohen. Using dynamic time warping to bootstrap hmm-based clustering of time series. *Sequence Learning*, pages 35–52, 2001.

[8] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[9] L. R. Rabiner, C. H. Lee, B. H. Juang, and J. G. Wilpon. Hmm clustering for connected word recognition. In *Proceedings of the IEEE ICASSP*, 1989.

[10] Ido Roll, Vincent Aleven, Bruce M. McLaren, Eunjeong Ryu, Ryan S.J.d. Baker, and Kenneth R. Koedinger. The help tutor: Does metacognitive feedback improve students' help-seeking actions, skills and learning? In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 2006.

[11] Alexander Schliep, Alexander Schnhuth, and Christine Steinhoff. Using hidden markov models to analyze gene expression time course data. *Bioinformatics*, 19:255–263, 2003.

[12] Padhraic Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems 9*, 1996.