# Target Sequence Clustering

**Benjamin Shih**

December 2011
CMU-ML-11-103

School of Computer Science
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA

*Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy*

This research was sponsored by the United States Department of Education (grant no. R305B040063) through the Program for Interdisciplinary Education Research and by the National Science Foundation (grant no. SBE-0836012) through the Pittsburgh Science of Learning Center. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

## 1 Abstract

Researchers have discovered many successful algorithms and methodologies for solving problems at the intersection of machine learning and education research. This umbrella category, “educational data mining,” has enjoyed a series of successes that span the research process, from post-hoc data analysis that generates models to the use of those models in successful educational interventions. However, most of these successes have arisen from the use of pre-existing psychological and educational constructs (e.g., guessing) and thus from the use of semi-supervised or fully-supervised machine learning algorithms. Algorithms for novel discovery, also known as unsupervised clustering, have enjoyed significantly fewer successes in this domain, partially because education data exhibit unique, complex structure.

This thesis is a mixture of algorithm development, simulation, and experimentation on real-world data, all designed to define and test a novel paradigm for clustering in education (and a range of other domains). This paradigm, target clustering, revolves around the inclusion of high-level targets, such as student learning from pre-test to post-test. This approach differs from other existing machine learning approaches in that it is designed completely, from the initial concept to the final execution, for solving educational research problems, taking advantage of the structural complexities that are problematic for other algorithms. This thesis includes a range of data sets drawn from a variety of research domains, but does

not include new data from experiments in the psychological sense.[1] However, the thesis includes analysis of methodology, results, and implications from an educational research perspective and relies entirely on education data and research problems.

## 1.1 Novelty and Contributions from a Machine Learning Perspective

Clustering algorithms are powerful methods for understanding the structure of data. However, in many applied domains, clustering algorithms are only useful if they produce clusters that explain external variables. For example, in education research, clusters of student behaviors are only useful if they help predict student learning. Generally, clustering algorithms are not explicitly designed to handle this additional constraint. In this thesis, I show that incorporating these external variables directly into clustering algorithms can improve both cluster relevancy and model generalization. This approach, called target sequence clustering, has produced promising results on multiple data sets. While I will demonstrate the effectiveness of this approach using educational data sets, I will also illustrate how the method can be applied more broadly. The primary contributions of this thesis include:

- The introduction and development of a paradigm for applied machine learn-

[1]This thesis includes many new analyses of existing data sets (experiments in the machine learning sense), but includes no previously unpublished data.

ing research designed specifically to tackle a class of practical, heretofore unaddressed problems.

- Several target clustering algorithms and algorithm variants.
- Connections from target clustering to other, established approaches, and the associated possibilities for future algorithm adaptation and development.

## 1.2 Novelty and Contributions from an Educational Research Perspective

The discovery of new psychological and educational constructs has largely been the domain of researchers, not algorithms. Purely computational approaches, even when blessed with good data, are largely unable to find novel constructs with applicability to interesting educational problems. This difficulty is, in part, because traditional machine learning problems rely on input data $x$ and the associated labels $y$, a framework divorced from the educational research reality of complex interrelationships between constructs and subjects and tasks. Standard algorithms simply do not account for the vertical richness of educational data sets, such as the presence of individual students, individual problems, and overall learning. A few approaches exist in machine learning that allow for the inclusion of additional data and constraints, such as must-link constraints in semi-supervised clustering, but adapting these algorithms to educational data is a non-trivial task. Target clustering, proposed herein, represents an approach built from the ground-up to generate novel constructs from educational data. The primary contributions of this thesis include:

- A new class of algorithms that are simple to understand and implement, but powerful enough to tackle educational data with a wide-range of complexity and structure.
- A variety of novel human-learning constructs generated from existing data. These constructs can provide impetus for additional research or provide evidence supporting existing results.
- Additional algorithms and approaches for other educational research problems.

# Table of Contents

# List of Figures

# List of Tables

# 2 Introduction

The application of machine learning and data mining methods to real world data and real world problems often requires significant intervention and expertise from human researchers. In particular, the application of exploratory methods often requires two model searches: a formal model search embedded within the machine learning algorithm(s) and an informal model search conducted by the machine learning practitioner. For example, a researcher in speech recognition might initially try to parse human speech with a dynamic time-warping approach, then hidden Markov models, and then linear dynamical systems. This process of iteratively testing different possible algorithms and evaluating the resulting models is similar to a stochastic hill-climbing algorithm, except that this model search is conducted by a human researcher.

This process is necessary for several reasons. First, when compared to theoretical simulations and toy data sets, practical problems often involve data with more noise, more corruption, more complex structure, and less fidelity to common assumptions (e.g., independence). Especially given the often novel structure found in real-world data sets, finding an algorithm that is suitable for a particular data set is as much art as science. Second, real-world problems often have goals beyond good classification accuracy or tight, well-defined clusters. For example, in the educational domain, the goal of machine learning and data mining is usually to find models that predict student outcome measures such as the drop-out rate, test scores, etc. Unfortunately, sequence classification algorithms are generally not optimized for predicting outcome measures when they exhibit this type of unusual, semi-hierarchical structure.

There do exist algorithms that can incorporate various types of non-label information, such as the must-link and cannot-link constraints found in semi-supervised clustering or, more generally, the probabilistic relations featured in structured graphical models. These methods will be discussed later.[2] However, most algorithms that operate at the level of educational software trace data cannot directly incorporate measures such as student learning, at least not in the context of sequence clustering. This limitation is due to the distance between the various levels of data: the input might be a series of student interactions at the granularity of clicking or typing *per problem*; the labels-to-learn might be for constructive versus non-constructive types of interactions *per problem*; and the outcome measure might be the amount learned between a pre-test and a post-test, administered and scored *per student*. On top of this, the relationships between levels of data can be quite complicated (e.g., test scores may be associated with the distribution of different types of student actions, as will be assumed later). As a result, these high-level outcome measures serve as a guideline or target, useful for evaluating computational models, but not trivially applicable to inferring parameters or structure. The algorithms discussed herein are designed within a new paradigm that directly incorporates these high-level targets into the sequence learning process, allowing this additional data to improve both model training and model evaluation. Further, the application of clustering algorithms that incorporate data from

[2]See Section 4.2, page 45 and Section 4.1, page 44.

multiple grain sizes is also novel in this domain (educational data).

I call this paradigm *target clustering*.

## 2.1 Formalism

Let $X = \{x_i | i = 1 \ldots n\}$ be the data of interest. $x_i$ could be a point in high-dimensional Euclidean space, a sequence of symbols, a corner of a hypercube, or any other type of data. For the purposes of this thesis, however, $x_i$ will usually be a sequence of symbols, each representing a student action in an educational software environment. Let $Y = \{y_i | i = 1 \ldots n\}$ be the (usually unobserved) labels for $x_i$. Each $y_i$ can be viewed as an assignment for the $i$-th data point into a domain-interpretable cluster. For example, each cluster could represent good-versus-bad actions, an evaluation of student attentiveness, or any other kind of common categorization. For the purposes of this thesis, $y_i \in Y$ will always be discrete. Finally, let $\Lambda = \{\lambda_j\}$ be the set of higher-level targets where $j < n$. For example, if each $\lambda_j$ represents a test score and there are $m$ students, then $1 \leq j \leq m$. Let each $\lambda_j$ be defined in such a way that it maps to a subset of $X$, e.g., each student's test scores map onto his or her low-level actions in an educational software environment. In general, these labels $\lambda_j$ could be anything from student learning gains to truancy rates.[3]

[3]This basic problem structure is common outside of education as well. For example, microbiome research could be viewed in the same context: each $x_i$ would be a gene sequence for one microbe drawn from one test subject, each $y_i$ would be an assignment for a microbe into a cluster, and each $\lambda_j$ would be test subject $j$'s score on some measure (e.g., symptoms of distal cancer).

As a more detailed illustration, consider automated essay grading. For automated grading, each $x_i$ is a student essay, possibly represented as a vector of words or n-grams. Each $y_i$ is a label for that essay such as “pass” or “fail”.[4] This part of target clustering is the same as in standard approaches to essay grading. What target clustering adds are targets (i.e., $\lambda$’s) that represent desirable classifications of students. For example, if the automated essay grading system is for a college entrance examination, then the $\lambda$ values could be college graduation rates. In this case, the $y_i$’s will not simply be representations of “pass” or “fail” essays, but “pass” or “fail” essays classified so that the $y_i$’s predict college graduation rates.

The goal of a target clustering algorithm is, given $X$ and $\Lambda$, to learn $Y$ such that $X$ maps to $Y$ and $Y$ maps to $\Lambda$. This task involves optimizing over two search spaces: the mapping between $X$ and $Y$ and the mapping between $Y$ and $\Lambda$. These relationships can be defined parametrically or non-parametrically, just as with traditional learning problems.

For example, to revisit the automated essay grading example, one simple way to classify essays is by the size of their vocabulary. If a test-taker uses more than 100 distinct words, mark it as “pass”, and if a test-taker uses fewer than 100 words, mark it as “fail”. This algorithm represents a simple way to score exams and may even match well with training labels such as human scoring of the essays. However, if the goal is to score essays such that the essay scores help predict student performance in college, then vocabulary size is probably an

[4]The example still holds with if there are more than two categories for $y_i$.

insufficient criterion. There is thus a constant trade-off between improving the mapping between $X$ and $Y$ (exam and grade) and the mapping between $Y$ and $\Lambda$ (grade and future college performance). This trade-off is a major component of target clustering and can be optimized either iteratively (optimizing one mapping and then the other) or simultaneously.

Another perspective on target clustering is to treat the problem as a function-learning task. In traditional clustering, the goal is to learn a function $f$ such that $f(X) = Y$ minimizes some objective function. Target clustering introduces a second function $g$ that must be optimized such that $g(Y) \approx \Lambda$.

One example target clustering algorithm is Guidepost EM-HMM, to be described later. However, for illustrative purposes, in Guidepost EM-HMM, $f$ is a set of hidden Markov models (HMMs) while $g$ is linear regression. Optimizing $f$ requires learning the parameters of the set of HMMs while optimizing $g$ involves a stepwise regression, the side-effect of which is to perform model selection on the HMMs. This process creates a feedback loop that allows target $\lambda$ values to affect the learning of $f$.

A final perspective on target clustering is to think of clustering as providing an answer to a question; however, in applied domains, unsupervised clustering often amounts to finding an answer and then, post-hoc, trying to link the answer back to the original question. This can illustratively be called "cluster and pray" and is exemplified by the common use of clustering algorithms where the clustered sequences are only post-hoc correlated to the variables of actual interest. There is no particular reason to believe that every question about a data set should be answered by the same partioning of the data. Further, the semi-supervised solution only works if high-quality labels are available. The result is that, in domains with no naturally occuring labels and in which humans are ineffective (or expensive) annotators, there is a significant discrepancy between the methods available and the demands of common research problems.

## 2.2 Overview

This document begins with a discussion of education research and machine learning, particularly focused on intelligent tutoring systems. A discussion of data and data representation follows, along with a brief sojourn into vector space algorithms. The bulk of the document is divided up between two classes of algorithms: those based around hidden Markov models (HMMs) and those based around distance metric learning. These sections consist of a mix of algorithm descriptions, simulation results, and empirical results. The final sections detail practical results for the education domain and general conclusions.

The first class of algorithms involve learning hidden Markov models for all or some of the data, and then adjusting, improving, or adapting the models to provide clusters that are predictive of targets ($\Lambda$'s). The second class of algorithms involves using (or learning) distance metrics between sequences, transforming the problem from a sequential one to a distance-metric clustering problem. Many of these metrics preserve some order information in their measures of distance, and in the case of learned metrics, target information can be readily incorporated.

# 3 Problem, Data, and Representations

## 3.1 Education Research and Machine Learning

Education research has a long and storied history. There are many topics of study in education research, ranging from cognitive models to social dynamics in the classroom [51]. This thesis focuses on the area of metacognitive strategies and behaviors [25]. Metacognition is distinguished from other topics by being domain independent: a metacognitive strategy should be applicable in a variety of disciplines (e.g., physics [31] and geometry [48]) and involves both self-monitoring and self-regulation tasks [25].

Machine learning methods have been used to analyze educational trace data for decades. In particular, for metacognitive strategies and behaviors, some topics that have been explored with machine learning include:

- Guessing [10]
- Gaming the System / Help-Abuse [10]
- Teaching / Tutoring Strategies [18, 29]

- General, High-Level Strategies [10]

In fact, many of these topics have been studied using hidden Markov models (HMMs), including each of the examples cited above. For example, Cooper et al. used both neural networks and HMMs to analyze student interactions as they try to solve ill-defined problems [18]. The neural networks find patterns in the students' choices, which then form strategies. They used HMMs to model the transitions between these strategies. Jeong et al. used HMMs to analyze the steps a student used to "teach" a computer agent about a topic [29]. They broke the HMMs down into separate categories. For example, in their map-building HMM, students usually edited a map, submitted the map, and sometimes used reading resources.

One particularly relevant study, Beal et al., used tutoring system log data to learn HMMs modeling patterns of student behavior [10]. Their study differs from this one in several ways: they define the structure of the HMMs by hand, they use outputs from another algorithm as inputs to their HMMs, they learn one HMM per student, and they perform clustering of students (not tactics) only after learning the HMM parameters. However, their key result is still relevant: HMMs can work as both descriptive and predictive models for student learning behaviors and can find useful patterns without using cognitive models or domain content knowledge.

## 3.2 Intelligent Tutoring Systems

This thesis will focus on one particular subset of educational research focused on intelligent tutoring systems. Intelligent tutoring systems allow students to interact with software that adapts to their cognitive mastery (and possibly metacognitive mastery as well). These adaptations can be as simple as not presenting problems based on skills a student has already mastered, or can be as complex as extra assessment and instruction focused on domain skills and metacognitive feedback. Examples of intelligent tutoring systems include:

1. Andes: Tutoring system for physics.[55]
2. ASSISTments: Web-based tutoring system that addresses curriculum- and test-relevant concepts and skills in middle schools.
3. AutoTutor: Tutors critical thinking and metacognitive skills in the context of physics.[27]
4. Cognitive Tutors: Variants cover a range of math, science, and language curricula, including core algebra and geometry, with full supporting curricula.
5. Wayang Outpost: Multimedia-oriented tutoring system targeting standardized math tests.

Research into all the tutoring systems cited above has included work on metacognition, sometimes in the form of predictive models and sometimes in the form of active interventions. This thesis will focus on predictive models, but it's worth considering, as context, the forms that an educational intervention can take. Sometimes interventions take the form of discouraging or prohibiting poor behaviors, sometimes of encouraging or modeling positive ones, and sometimes they involve a significant redesign aimed at preventing poor behaviors before they occur.

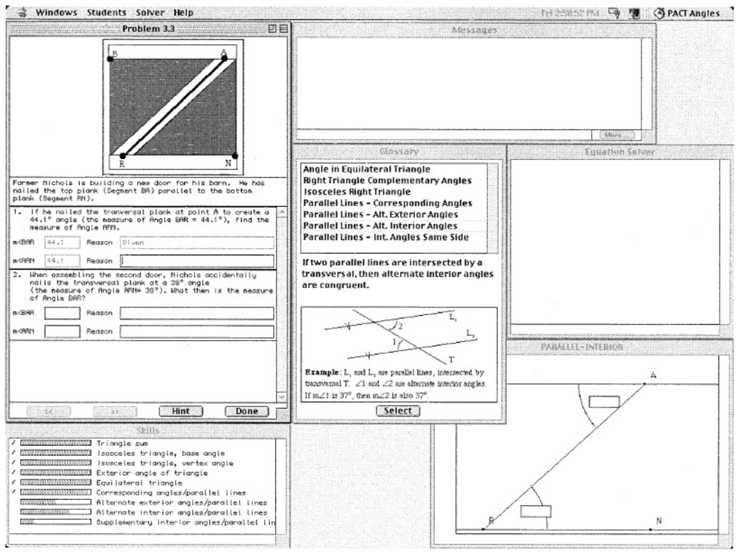
One standard study design used with intelligent tutoring systems is to:

1. Have students take a pre-test to determine their initial mastery.
2. Allow students to interact with the tutoring system.
3. Have students take a post-test to determine their final mastery.

During the second step of this process, students generate large amounts of log data. This trace data can be as detailed as mouse-clicks and eye-tracking or as coarse as time-spent-per-screen. Machine learning algorithms can be applied to the log data, producing various types of models of student behavior. These models can then be used to predict student learning. This analysis step is where target clustering can prove useful.

## 3.3 Data

Student traces drawn from several data sets will be used at varying stages in this document. These are as follows:



Farmer Nichols is building a new door for his barn. He has nailed the top plank (Segment BA) parallel to the bottom plank (Segment RN).

1. If he nailed the tranversal plank at point A to create a 44.1° angle (the measure of Angle BAR = 44.1°), find the measure of Angle ARN.

m<BAR 44.1 Reason Given

m<ARN 44.1 Reason

2. When assembling the second door, Nichols accidentally nails the transversal plank at a 38° angle (the measure of Angle ARN= 38°). What then is the measure of Angle BAR?

m<BAR Reason

m<ARN Reason

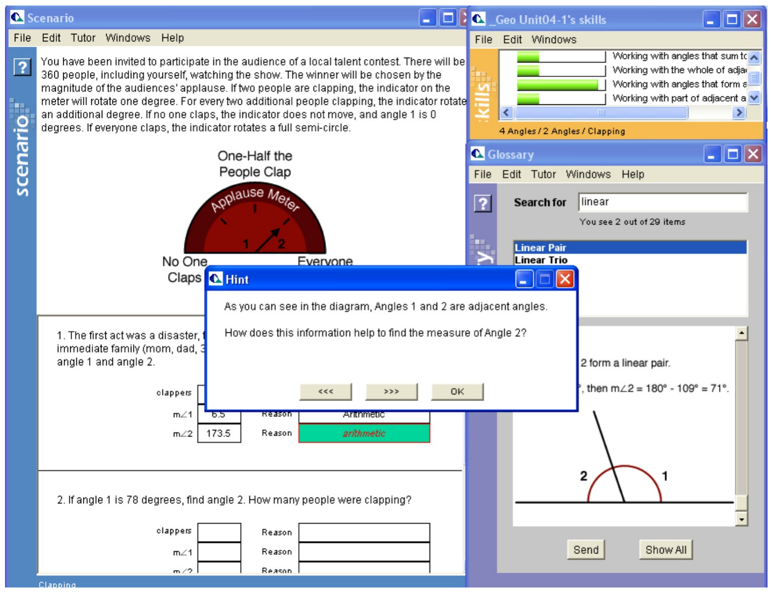
Angle in Equilateral Triangle
Right Triangle Complementary Angles
Isosceles Right Triangle
Parallel Lines - Corresponding Angles
Parallel Lines - Alt. Exterior Angles
Parallel Lines - Alt. Interior Angles
Parallel Lines - Int. Angles Same Side

If two parallel lines are intersected by a transversal, then alternate interior angles are congruent.

**Example**: $L_1$ and $L_2$ are parallel lines, intersected by transversal T. ∠1 and ∠2 are alternate interior angles. If m∠1 is 37°, then m∠2 is also 37°.

✓ Triangle sum
✓ Isosceles triangle, base angle
✓ Isosceles triangle, vertex angle
✓ Exterior angle of triangle
✓ Equilateral triangle
✓ Corresponding angles/parallel lines
Alternate exterior angles/parallel lines
Alternate interior angles/parallel lines
Supplementary interior angles/parallel lin

Figure 1: Screenshot of the Geometry Cognitive Tutor tutor, circa 2002. Original was not black-and-white.



You have been invited to participate in the audience of a local talent contest. There will be 360 people, including yourself, watching the show. The winner will be chosen by the magnitude of the audiences' applause. If two people are clapping, the indicator on the meter will rotate one degree. For every two additional people clapping, the indicator rotate an additional degree. If no one claps, the indicator does not move, and angle 1 is 0 degrees. If everyone claps, the indicator rotates a full semi-circle.

1. The first act was a disaster, immediate family (mom, dad, 3 angle 1 and angle 2.

clappers

m∠1 6.5 Reason Arithmetic

m∠2 173.5 Reason arithmetic

2. If angle 1 is 78 degrees, find angle 2. How many people were clapping?

clappers Reason

m∠1 Reason

m∠2 Reason

As you can see in the diagram, Angles 1 and 2 are adjacent angles.

How does this information help to find the measure of Angle 2?

Working with angles that sum to
Working with the whole of adja
Working with angles that form a
Working with part of adjacent a

4 Angles / 2 Angles / Clapping

Search for linear

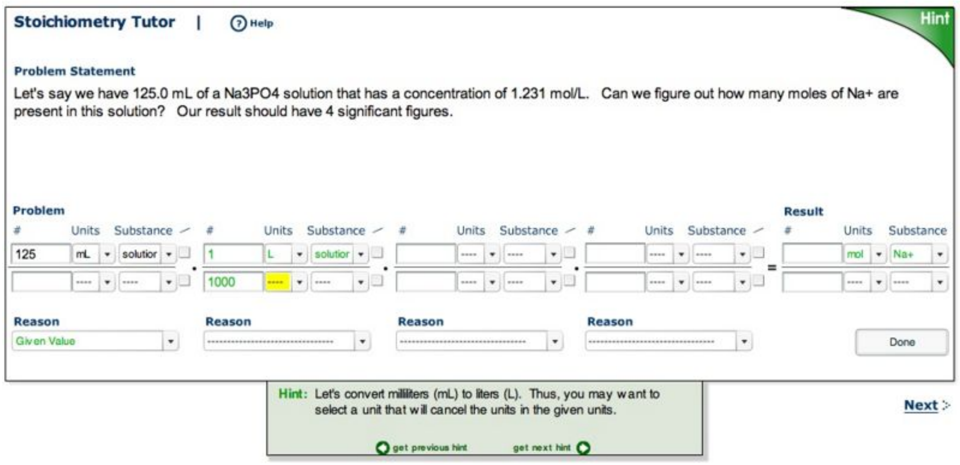
You see 2 out of 29 items

Linear Pair
Linear Trio

2 form a linear pair.

, then m∠2 = 180° - 109° = 71°.

Figure 2: Screenshot of the Geometry Cognitive Tutor tutor, circa 2006.



Problem Statement

Let's say we have 125.0 mL of a Na3PO4 solution that has a concentration of 1.231 mol/L. Can we figure out how many moles of Na+ are present in this solution? Our result should have 4 significant figures.

Hint: Let's convert milliliters (mL) to liters (L). Thus, you may want to select a unit that will cancel the units in the given units.

Figure 3: Screenshot of the Stoichiometry tutor.

- **Geometry 2002** (Geometry02) — There were 21 students in the control condition of this study. They used a relatively early version of the Geometry Cognitive Tutor without worked examples or any controls on hint use [1]. The interface is shown in Figure 1.

- **Geometry 2006** (Geometry06) — There were 16 students in the control condition of this study. The version of the Geometry Cognitive Tutor used in the study restricted hint requests to at most one per two seconds [48]. The interface is shown in Figure 2.

- **Stoichiometry 2006** (Stoich06) — There were 81 students in this study. This tutor introduces math required to solve elementary chemistry problems [40]. The interface is shown in Figure 3.

The bulk of examples and analysis will use Geometry02, which was used as a canonical example in earlier target clustering papers.

## 3.4 Problems versus Steps

In these data sets, students perform a series of actions, such as requesting a hint. Each action is performed for a given step (e.g., finding an angle in a triangle). A problem is made up of a series of related steps, and problems are in turn divided into units.

On any given step, a student's actions can be represented by a sequence $s \in \Sigma^{n_s}$ where $\Sigma$ is the set of possible actions and $n_s$ is a random variable representing the length of a step or problem. Let $\Sigma$ also be known as the alphabet. In the simplest case, a student's attempt to solve a step is represented by a sequence of **A**'s (attempts) and **H**'s (hints). For example, a long series of hints followed by an attempt would be **HHHHHA**. In this example, $\Sigma = \{\mathbf{A}, \mathbf{H}\}$ and $n_s$ is 6.

Generally, the most important actions will be **A** and **H**. However, students can make either a correct attempt or an incorrect attempt, not just a generic attempt. Thus, in some experiments, **C** and **I** will represent correct and incorrect attempts, replacing **A**.

Sequences of actions from multiple steps can be concatenated into sequences of actions for a problem. For example, if a student gets the first two steps correct on the first try, but then makes a mistake and needs hints to solve the last step, the individual step sequences would look like:

- **C**
- **C**
- **I**
- **IHHHC**

As a single problem sequence, this would be: **CCIIHHHC**. In addition, students can switch back and forth between steps. Thus, an additional symbol can be added to represents switches, **X**. With the **X** symbol included, the example problem sequence would be **CXCXIXIHHHCX**.

Generally, these different representations address different questions. In particular, patterns of behavior observed in step sequences might be: guessing repeatedly on a step, asking for a hint, etc. However, patterns of behavior observed in problem sequences can provide additional context and can address more complicated behaviors, such as student problem-solving strategies, e.g., solving the easy steps first versus going through the steps in order. Unfortunately, the problem-level representation has disadvantages as well. Dimensionality is an issue and inference is more difficult for larger alphabets of symbols, which increases the number of parameters to be learned, as well as for problems versus steps, as there are fewer problems and thus fewer examples in a problem-level representation.

|  | **Attempt** | **Hint** |
|---|---|---|
| Fast | a | h |
| Slow | **A** | **H** |

Table 1: Mapping from actions and durations to symbolic representation

## 3.5 Thresholding

Each action also has an associated duration. For example, it may take one student 4.7 seconds to read a hint or it may take another student 6.3 seconds to enter an attempted solution. One way to incorporate this temporal information into the alphabet is to split on a threshold such that fast actions are denoted by one symbol and slow actions are denoted by another. An example of this is shown in Table 1.

In general, all upper-case symbols represent long duration actions (or an action not thresholded), and all lower-case symbols represent short duration actions. Practically, this is limited to attempts (**A**, a), hints (**H**, h), correct attempts (**C**, c) and incorrect attempts (**I**, i). Other actions occur fairly rarely and there is no evidence of improved model performance or improved interpretability when they are thresholded.

I use a simple shorthand to identify the data set, the granularity, and the thresholds used: *SubjectYear-GranularityThresholds*. For example, Geometry02-StepA8H8 indicates that the data set is about geometry, from 2002, uses a-step level representation, and uses a threshold of 8 seconds for both attempts and hints. Similarly, Algebra07-ProblemA9H2 indicates that the data set is about algebra, from 2007, uses a problem-level representation, and uses thresholds of 9 seconds for attempts and 2 seconds for hints. Finally, something like Stoichiometry06-StepMean indicates that the data set is about stoichiometry, from 2006, uses a step-level representation, and uses thresholds for attempts and hints set at their respective mean values (e.g., if the mean duration for attempts is 4.3 seconds, then the threshold for attempts would be 4.3 seconds).

Some of the data sets used will be:

- Geometry02-StepA8H8
- Geometry02-StepMean
- Geometry06-ProblemA8H8

8 seconds is a commonly used threshold as it happens to be an effective choice. Discussion of this phenomenon will be reserved for the Conclusions section.

## 3.6 Sequential and Vector Representations

As was shown before, the basic representation for these log files is sequential. Unfortunately, most existing algorithms relevant to target clustering are defined for vector space representations of data; there are algorithms better suited for sequence learning,[5] but these algorithms tend to require more data than are available

[5]Examples include n-gram methods, sequence kernels, etc.

| Sequence |
|---|
| AAAAAAA |
| HHHHHHA |
| AHHHHHA |

(a) Sequential Representation

| Attempts | Hints |
|---|---|
| 7 | 0 |
| 1 | 6 |
| 2 | 5 |

(b) Vector Representation

Figure 4: Sequential and bag-of-words representations for vector data

in educational data sets.[6] However, many existing algorithms can theoretically be adapted to target clustering. For example, sequential data can be treated as a bag-of-words where each possible symbol type is converted to one dimension in a vector representation. The two tables in Figure 4 correspond to a sequential representation and vector / bag-of-words representation of the same sequences, demonstrating the distinction.

In principle, once this transformation is complete, any vector-space algorithm represents a potential approach to analyzing the data. In practice, this is not the case since many algorithms do not work on vector data with small feature sets. Further, as will be shown later, this transformation (which sacrifices all ordering information) tends to lead to poor results in real data sets of interest.

There are other possible transformations as well. For example, if a practitioner selects an upper bound on the length of any sequence, then all sequences can be standardized to that length (either by truncation or through the addition of blank symbols). Then, each index in the sequence can be treated as a feature, with

[6]For a more detailed example, see Section 3.7, page 41, which discusses an information retrieval analogue to the target clustering problem.

each possible symbol representing a nominal value. This case will not be treated here, but is likely to perform poorly on intelligent tutoring system data where the variability in sequence length contains vital information.

There are, however, several advantages to choosing a vector space representation and the associated algorithms. First, most algorithms related to target clustering are vector algorithms. Second, these algorithms can be easily tested on vector data to determine their viability after adaptation to target clustering, something that cannot be done with most sequential algorithms due to a lack of standard data sets with a known ground truth. Finally, most of these algorithms are variants of classic, robust solutions to unsupervised or semi-supervised learning problems, so decades of research are available for solutions and inspiration.

Unfortunately, these methods also do not work well once the necessary, lossy transformations are applied to the sequential data. For illustration, consider k-means and spectral clustering (with k-means):

**K-Means** is a standard, randomized Expectation-Maximization algorithm for finding clusters in Euclidean space. However, for sequential data transformed into bags-of-words, k-means is unlikely to perform well, both from the loss of sequential information and the presence of overlapping data points.

**Spectral Clustering**, generally, consists of performing k-means clustering after using a singular value decomposition to reduce dimensionality. It is capable of finding clusters along manifolds in the data, but still suffers from the lossy data transformation.

To apply these methods to Geometry02-StepMean requires a few steps:

- Convert the data ($m$ steps) to the discrete, thresholded, symbolic representation.
- Convert each sequence of actions to a bag-of-words representation, resulting in an $m \times |\Sigma|$ matrix $X$.
- (Optional) Normalize each row.
- Separate training and test data.
- Apply either K-Means or Spectral Clustering to the training data, producing $c$ clusters and assigning each row to a cluster.
- Create an $n \times c$ matrix $D$ where $D_{i,j}$ is the probability that a random sequence from the $i$-th student would belong to cluster $j$.
- Apply a regression to the training clusters, attempting to predict student learning ($n \times 1$ vector).
- Keeping the regression coefficients and cluster centers from the training data, compute clusters on the test data. Using the same coefficients as before, compute the $R^2$ fit to student learning scores.

A simplified, graphical representation of the experimental procedure is shown in Figure 5. Table 2 shows how each method performed on Geometry02-Step. The

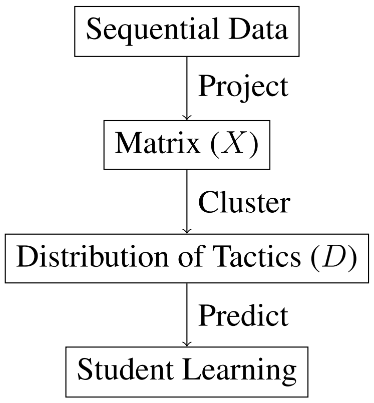


Figure 5: The (simplified) experimental procedure used for vector algorithms applied to a bag-of-words representations

Table 2: Vector algorithm performance

| **Algorithm** | **Average Training $R^2$** | **Best Test $R^2$** |
|---|---|---|
| K-Means | 0.4269 | -28.6939 |
| Spectral | 0.5124 | -15.3498 |

first 80% of steps per student are made available as training data. The first column shows training-set performance averaged across 10 iterations. The second column shows poor test-set performance, suggesting over-fit. It's worth noting that $R^2$, while bounded between 0 and 1 on training data, is unbounded on the downside when applied to other data.

There are two probable reasons that these algorithms do not generate models that predict student learning. First, the algorithms may themselves be bad, e.g., k-means is sensitive to the initial starting centers and choice of $k$. Second, there may be fundamentally too much data loss during the transformation from sequences to vectors.[7]

## 3.7 Information Retrieval Analogy

To better understand why many traditional methods (like K-Means) might fail on these types of problems, it is useful to view target sequence clustering through an analogy to information retrieval in web search. Each student corresponds to a webpage, each sequence of actions corresponds to a word, and learning gain corresponds to some rating of webpage quality. When humans read a word on a webpage, all manner of context comes into play. On a webpage, words form sentences, sentences form paragraphs, and paragraphs develop arguments and lines of reasoning. Similarly, for a student interacting with a tutoring system, actions on steps combine into behaviors on problems, behaviors on problems combine into behaviors for entire sessions, and behaviors during a session are governed by a student's strategies and policies for learning the material.

In this analogy, it's useful to think of the information retrieval target as being relative to a construct: words relevant to a search for sports cars may not be relevant to a search for apple pie recipes. This example is an illustration of why target clustering is important, particularly for education: there is no a priori reason to believe that behaviors relevant to one assessment should necessarily be relevant to

[7]There are other techniques for transforming sequences to vectors. For example, n-grams are a logical extension of bag-of-words representations to n-tuples of multiple consecutive symbols. It is not known if n-grams might improve the above results. An argument to the contrary, however, is presented in the next section.

another.

The classic information retrieval solution, less popular now, was to ignore the context and treat each webpage as a bag-of-words. This simplification was possible (and necessary) in part due to the incredible number of webpages available; properties of any given word could be inferred by using the hundreds of thousands of webpages on which even an obscure word would occur. This dependence on data set size is where target sequence clustering problems differ from information retrieval problems: there may be billions of webpages on the Internet, but most educational data sets with the right characteristics (e.g., sequences of actions, per-student measures, distinct steps, etc) have only a handful of students. A data set with 40 or 50 students per condition is relatively huge!

Due to the scarcity of data, a successful approach to solving target sequence clustering problems in education cannot rely entirely on existing methods. However, there are still many commonalities, and the information retrieval analogy is worth keeping in mind. In later sections, there will be preprocessing methods almost identical to stop-lists, as well as algorithms that use information retrieval metrics such as edit-distance.

Overall, what will be shown is that, by incorporating target information, sequence clustering algorithms can be adapted to learn clusters relevant to selected topics. In particular, these algorithms will perform well on simulated data drawn from expert-developed models, will perform well on randomly generated models under certain conditions, and will perform well on real-world data. However, the most effective algorithms will be slower and more prone to local maxima, while faster, globally optimal algorithms will tend to be less robust. In addition, it will be shown that these results depend both on algorithms that adapt to target information and the presence of high quality target information.

## 4 Background

Target clustering is strongly related to the following problem classes: structured graphical models, semi-supervised clustering, learning with noisy labels, learning with multiple labels, and multiple-instance learning. Many algorithms from each class can be adapted to target clustering tasks. Each area of prior research also provides key motivations and framing for target clustering. There are also several individual algorithms designed to address similar issues or special cases. The particular algorithms closest to target clustering will be treated at the end of this section.

For the purposes of this section, assume that all $\lambda_j \in \Lambda$ are discrete, as most prior educational literature treats $\lambda_j$ as a discrete variable. Examples of discrete $\lambda$ include pass/fail, graduate/not-graduate, good-learner/bad-learner, mastery/non-mastery, and A/B/C/D/F. These discrete $\lambda$ are distinct from continuous $\lambda$ such as pre- to post-test scores, percentage-of-problems-correct, and attendance rates. Though most $\lambda$ used in this thesis are actually continuous, assuming they are discrete simplifies many of the examples in this section.

## 4.1 Structured Graphical Models

The core idea behind structured graphical models is to extend traditional graphical models beyond flat data (i.e., a single table in a relational database) to data where variables can have probabilistic parent relations (e.g., getting a step right or wrong is partially predicted by the student and partially predicted by the step itself). One of the primary approaches is that of probabilistic relational models (PRMs) [26], which, as the name implies, combine probabilistic models with relational database schemas. The power of PRMs is in their generality: they can model uncertainty in variables, the dependencies between variables, and even in the relational schema. However, PRMs, plate models, and other structured graphical models have not been applied to educational data before and it is not clear how they would perform. In particular, educational data often exhibits unusual structure. For example, as shown in the earlier bag-of-words example,[8] I have and will assume that student behaviors belong to clusters and that the distribution of a student's actions across clusters predicts learning; structured graphical models can certainly accommodate this relationship, but the necessary extensions are non-trivial.

However, in addition to being a potential method for directly analyzing educational data, structured graphical models could present an avenue for extending target clustering algorithms. For example, the primary target clustering algorithm in this thesis relies on hidden Markov models (HMMs) to model the observed student behaviors. There has been work on adapting PRMs to work with HMMs [2, 41].

[8] See Section 3.6, page 36.

Meyer-Delius et al., in particular, use HMMs to model observed behaviors and then separate the HMMs from the complex relationships to other variables by using an abstraction layer. This approach will show a marked similarity to the Guidepost algorithm to be presented later.[9]

## 4.2 Semi-supervised Clustering

Semi-supervised learning, in general, is the task of learning labels from a mix of unlabeled and labeled training data, plus constraints. As a formalized problem class, it dates back at least to the classic Blum and Mitchell paper on co-training [15]. However, semi-supervised learning tasks bear a strong relationship to missing data problems as well. Semi-supervised clustering is the special case where labels are learned from a mix of unlabeled training data, labeled training data, and constraints, but where the full set of labels is unknown. Thus, a semi-supervised clustering algorithm can induce new labels not present in the training data. This makes semi-supervised clustering a close relative to target clustering, as both problem classes require label induction under a series of constraints or biases.

There are several paradigms for semi-supervised clustering, but the most common either assume pairwise constraints on learning, e.g., must-link$(x_i, x_j) \rightarrow y_i = y_j$, or assume labeled points with an incomplete set of labels. Call the former semi-supervised constraint clustering and the latter semi-supervised metric clustering.

[9]See Section 6.1, page 63.

Algorithms for constraint clustering are usually based on unsupervised clustering algorithms with a modified distance metric that incorporates some form of cluster purity [23, 56, 8, 30, 13, 17], which are sometimes described as similarity-adapting methods. Algorithms for metric clustering are usually based on graphical methods that treat the links as present or not present edges [57, 6, 11, 5, 37, 34, 33, 22, 45, 44], which are sometimes described as search-based methods. Some methods exist somewhere in-between, such as Xing et al.'s algorithm for learning modified distance metrics using pairwise constraints [58]. This sub-category of metric learning from constraints has since become an active area of research [60]. There have also been attempts to merge the two frameworks [59, 12, 7, 32].

Notably, target clustering and semi-supervised clustering are not identical. Semi-supervised clustering assumes that, while we do not have full information about $f$, we do have partial information in the form of an incomplete label set or constraints on the labels. As a result, semi-supervised clustering involves a reduced search space (some candidate models are disqualified by said constraints) and no $g$ function. However, in target clustering, the search is over both $f$ and $g$.

Of the two main classes of semi-supervised clustering algorithms (excluding, for now, the hybrids), semi-supervised metric clustering algorithms are more promising as inspiration for target clustering algorithms. These methods usually depend on a novel distance metric that combines standard distance metrics with a measure of cluster purity. For example, a standard distance metric is the Euclidean distance between the centroids of two clusters; when considering the assignment of a point

$x$ with label $y$ to a cluster $C$, the Euclidean distance could be penalized by a factor of $\frac{|C_y|}{|C|}$ where $C_y$ is the set of all points in $C$ with label $y$. Measures of cluster purity (including measures like cluster entropy and mutual information) can be adapted for target clustering by allowing for multiple labels and for both discrete and continuous $\Lambda$. For example, consider the simplest measure of cluster purity, the number of majority class instances divided by the cluster size (as shown in the previous example). With discrete $\Lambda$, this can be computed exactly as is for every level of the hierarchy of labels, i.e., for a single point $x$ with $j$ labels $y_j$, $\prod_j \frac{|C_{y_j}|}{|C|}$.

Some semi-supervised constraint clustering algorithms are also applicable to target clustering. Chief among these are algorithms with probabilistic constraints, since they relax the usual requirement of strict pairwise constraints and replace them with fuzzy similarity. Examples of this are CVQE [44] and HMRF-KMEANS [7]. However, even strict pairwise algorithms can theoretically be adapted to target clustering. Particularly promising is hierarchical agglomerative clustering (HAC), which was shown to be viable with constraints by Davidson and Ravi [21]. HAC, which constructs a hierarchy of clusters using pairwise distances, is promising since the levels of a HAC hierarchy could theoretically be constrained by the set of labels $\Lambda$. Later sections will discuss HAC using targeted distance metrics.[10]

[10]See Section 8.3, page 107.

## 4.3 Learning with Noisy Labels

Learning with noisy labels (also known as learning with classification noise) involves learning in the presence of either random or systematic noise in the provided labels and applies to both semi-supervised and fully supervised learning problems. In general, this is a fairly realistic paradigm as both human and automated sources of labels are prone to errors. For example, in education data, most labels are provided by either teachers or field observers who, despite significant training, are still prone to human fallibilities. Learning with classification noise, even random, non-systematic noise, is hard. For example, no convex boosting algorithm can learn a concept class given sufficient noise [36] and even learning halfspaces in the presence of noise is NP-hard [46]. Even small amounts of random noise can lead to rapidly degrading performance in applied cases [63]. This deterioration occurs despite avoiding the largely degenerate case of systematic noise.

Using the noisy learning paradigm, target clustering $\lambda$'s can be thought of as noisy labels. For example, for education data, each student's learning gain can be thought of as a label for all the learning tactics that student uses. A learning tactic could be, for example, guessing repeatedly or asking for all the available hints before answering. Of course, learning gains are *extremely* noisy labels for learning tactics: good students are likely to occasionally use bad tactics and bad students are likely to occasionally use good tactics. Thus, mapping $\lambda$ values to noisy labels may not form an effective solution for general target clustering problems; there are, however, several interesting ideas to draw from the noisy labels literature. An example of how these various problem classes interact is research on learning with noisy constraints, which belongs to both semi-supervised clustering and learning with noisy labels [61].

## 4.4 Multiple-Instance Learning

Multiple-instance learning is defined as learning from bags of examples, where if at least one example in a bag is positive, the entire bag is given a positive label; otherwise, the bag receives a negative label. Multiple-instance learning is an interesting framework and represents many real problems, ranging from pharmacological attribution [24] to web page recommendations [62]. For multiple-instance learning to map to target clustering, it can be imagined that each $\Lambda$ represents a label for a bag of examples. For example, a "positive"-labeled student who has a high pre-post gain would be represented as a bag of actions, but only some of these actions would truly be "positive" even though the student is labeled as a "positive" student. Unfortunately, this is an overly simplistic representation of the richness of target clustering: for example, there are certainly differences between a student who uses one good learning tactic and is otherwise a poor student, as compared to a student who only uses good learning tactics. However, multiple-instance learning algorithms operate under the assumption that one positive example is sufficient. Also, less work has been done on multiple-instance learning with real-valued labels than with discrete labels.

One class of algorithmic solution, however, is particularly salient as a possible solution for target clustering. Diverse Density tries to find the point $t$ that is close to at least one example from every positive bag. Expectation-Maximization with Diverse Density extends this idea by using the point $t$ as a seed point. In the expectation step, it then picks the point $t_i$ from each positive bag $i$ that is closest to the center-point $t$. In the maximization step, it re-estimates $t$ using the new $t_i$'s. A similar algorithm might work for target sequence clustering. For example, finding a single HMM $M_t$ to classify the whole set of sequences is not difficult. The learning task then devolves into a traditional E-M clustering algorithm for HMMs, except that the centers of each cluster are determined by their closeness to $M_t$ and not by the structure of the cluster itself.

It's worth noting that, for independently sampled examples, multiple-instance learning is a special case of learning with noisy labels [14]. However, in target clustering problems, sampling is unlikely to be independent: students share problems, classroom events, and even the weather, all of which may impact student behaviors.

## 4.5 Multilabel Learning

Multilabel learning is defined as learning a mapping function $f$ such that $f(x_i) \approx \{y_{i,1}, \ldots, y_{i,l}\}$ where $l$ specifies the number of labels per example. Sometimes $l$ is assumed to be fixed for all $x_i$ and sometimes it varies between examples. Multi-

$$\begin{tikzcd}
x_1 \arrow[r] & \lambda_1, \lambda_3 \\
x_2 \arrow[r] & \lambda_1, \lambda_3 \\
x_3 \arrow[r] & \lambda_1, \lambda_4 \\
x_4 \arrow[r] & \lambda_2, \lambda_4 \\
x_5 \arrow[r] & \lambda_2, \lambda_4
\end{tikzcd}$$

(a) Multilabel Learning

$$\begin{tikzcd}
x_1 \arrow[r] & y_1 \arrow[rddd] \arrow[rdddd] & \lambda_4 \\
x_2 \arrow[ru] & & \lambda_2 \\
x_3 \arrow[r] & y_2 \arrow[ruu] \arrow[rd] & \\
x_4 \arrow[rd] & & \lambda_1 \\
x_5 \arrow[r] & y_3 \arrow[ruuuu] \arrow[ruuu] & \lambda_3
\end{tikzcd}$$

(b) Target Clustering

Figure 6: Multilabel learning and target-clustering-as-multilabel-learning.

label learning is usually treated as a classification task with known labels. There are two primary types of multilabel learning algorithms: problem transformation and algorithm adaptation methods [54]. An example problem transformation algorithm learns naive binary classifiers for each label and then combines the results. An example algorithm adaptation method is to find the $k$ nearest neighbors for a point $x$ and output the most common $m$ labels [38]. In practice, the main difference between the two classes is whether the problem is transformed externally to the classification algorithm or internally, such as by adjusting the objective function.

Theoretically, multilabel learning can be treated as a special case of target clustering where $g$ is fixed to be injective from $Y$ onto each $\Lambda$. This analogy is illustrated in Figure 6.

A special case of multilabel learning is hierarchical multilabel learning, where each $x_i$ receives one label $y_{i,j}$ from each tier $j$ of a label tree. An example is more likely to be classified by a given label if there is a path between them on the tree. At first glance, this may seem like a close match to target clustering. However, all the limits relevant to general multilabel learning (particularly the injective requirement) are also relevant to hierarchical multilabel learning. In fact, hierarchical multilabel learning constrains the functional forms of $g$ even further than general multilabel learning, which makes it somewhat less useful in the general case of target clustering.

## 4.6 Target Factor Analysis

One particular method deserves special treatment: target factor analysis (also known as target testing). Developed for chemistry studies in the late 70's and early 80's, target factor analysis essentially uses known, empirical results to verify and interpret the results of factor analysis [39]. In general, it involves taking a target vector $t$ and measuring its distance from a subspace found through singular value decomposition. In chemistry, $t$ is usually some compound's pure spectrum, where the compound has desirable properties at the experimental spectral resolution. The presence or non-presence of various target compounds $t$ is then used to verify and interpret data from chemical sensors.

One variation of target factor analysis is iterative target factor analysis [16]. In iterative target factor analysis, rather than testing the data collection method using the test vector $t$, the raw data is used to improve upon or find missing values of $t$. The iterative improvement is performed by updating $t$ with new predictions and using the predicted values as new inputs to the algorithm. Both target factor analysis and its iterative form have an interesting relationship to a class of algorithm to be discussed later: Guidepost algorithms.[11]

## 4.7 Particular Algorithms

Several algorithms developed for biological data are similar to target sequence clustering algorithms or solve similar problems. For example, Lin et al. developed an approach using hierarchies of discriminative HMMs to identify highly discriminative motifs [35]. In general, motifs are a set of sequences with some common feature or ancestry, e.g., protein sequences mapped to certain biological functions. They are usually represented by some probability distribution across symbols at each index in a series, the exact details of the motif representation varying with the researcher's assumptions. Lin et al. demonstrated that their approach can find highly discriminative motifs in sets of protein sequences. However, unlike the target clustering algorithms described in this document, their approach requires predefined sets of distinct classes. For example, mapped into the space of educational data sets, their approach could be used to find one HMM representing a motif common to good students and one HMM representing a motif common to poor students, with the two HMMs providing a high degree of discrimination be-

[11]See Section 6.1, page 63.

tween the two classes. With some modifications, their approach can be adapted to identify a set of highly discriminative HMMs for each class. However, this showcases two key distinctions between this approach and target clustering algorithms: first, their approach derives its power from the existence of discrete classes, but target clustering algorithms are designed to operate without distinct classes; second, target clustering algorithms operate with the possibility, assumed away by the Lin et al. approach, that important, useful patterns can, in fact, be common to both good and bad students. Consider a student that gets a step right on the first try without using hints: this is generally considered good behavior and a sign of understanding and mastery. However, both good students and bad students will get some steps correct on the first try, which would be impossible under the Lin et al. representation. In sum, both approaches are designed for fundamentally different problems, but the discriminative HMM algorithm described later draws most of its inspiration from adapting the Lin et al. approach to target clustering.

# 5 EM-HMM

## 5.1 Hidden Markov Models

Hidden Markov models (HMMs) represent the primary generative model used in this thesis. HMMs were chosen because they provide a good representation of learning tactics. Consider the following simple example of a learning tactic: “The

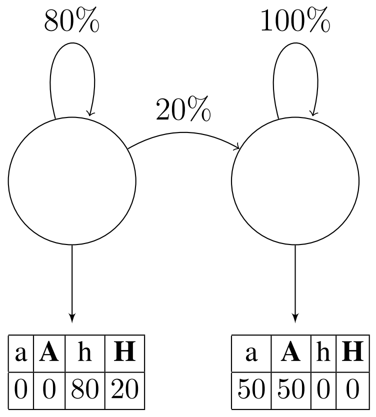


Figure 7: Example HMM — "Click-through-Hints and Answer"

student requests hints quickly, repeatedly, until the tutor provides the solution. The student then enters the solution." From this example, a learning tactic can be generalized to be an observable, predictable, and repeated *pattern* of behavior that is sufficiently abstract to include multiple *observed* instantiations. An HMM is a particularly felicitous representation of a learning tactic, as it includes a set of unobserved states, each state related to observations through a probability distribution. For student-tutor data, the observations are student actions. Figure 7 shows an example HMM that operationalizes the concept of a student clicking quickly through the available hints before answering. Each unobserved state is represented by a circle; each arrow between states or looping back to a state represents a transition; the number above the arrow is a transition probability. The tables below the states show the probabilities of observing an action. When an HMM produces an action symbol, we say it *emits* the symbol.

Let a series of observed actions be a sequence. Sequences can be defined for either all actions in a problem or all actions in a step. Given a set of student sequences associated with an HMM, the Baum-Welch algorithm can relearn the parameters of that HMM to better fit the observed data [9]. Baum-Welch is a standard method for learning the parameters of a single HMM.

## 5.2 EM Clustering for Hidden Markov Models

Let each individual HMM represent a single learning tactic. Discovering learning tactics requires discovering sets of HMMs. Let a set of HMMs be called a *collection*. In a collection, an observed sequence of actions is classified by the HMM most likely to generate it. For example, consider the HMMs shown in Figures 8 and 9, called respectively "Guessing" and "Repeated Attempts". The probability of a length-2 sequence generated by "Repeated Attempts" being **A**a is $50\%$, a very high probability; however, the probability of such a sequence being generated by "Guessing" is $100\%$. Thus, **A**a will be assigned to "Guessing" and not "Repeated Attempts". This classification process results in a partitioning of the set of sequences, with each partition corresponding to one HMM. Each partition thus includes all observed examples of a given tactic.

The Baum-Welch algorithm can only learn parameters for a single HMM, but clustering algorithms can learn sets of HMMs, and thus sets of tactics. The usual objective of an HMM clustering algorithm is to maximize the total likelihood of

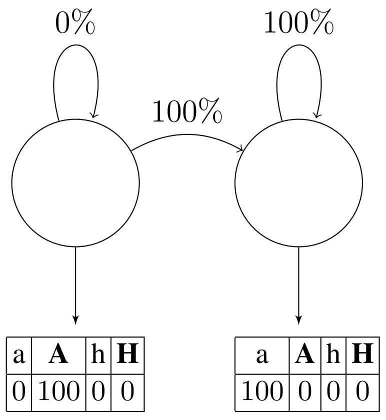


Figure 8: Example HMM — “Guessing”

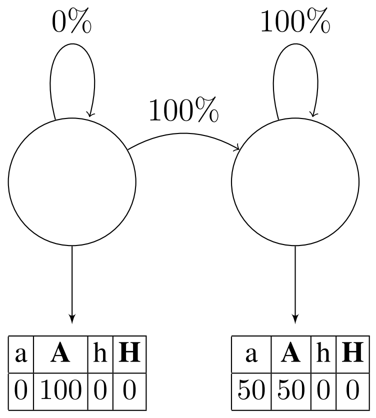


Figure 9: Example HMM — “Repeated Attempts”

generating the observed sequences. This type of problem has historically been tackled with Expectation-Maximization (E-M) algorithms and, for HMM clustering, given an initial set of HMMs, one iteration of an E-M clustering algorithm is:

- Expectation: Assign each sequence to the HMM most likely to generate it.
- Maximization: For each HMM, relearn its parameters with Baum-Welch using the sequences in its partition.[12]

This process begins with initial seed HMMs and repeats until a termination criterion is met, such as when an iteration results in fewer than 10 sequences being reclassified. A collection learned by this algorithm fits the data well if the likelihood of generating the observed sequences is high. This algorithm, here on called *EM-HMM*, is guaranteed to converge to a *local* minimum. Further, *EM-HMM* will never change the number of HMMs in the collection ($k$) or the number of states per HMM ($n$); only the parameters (and thus, the partitioning of the data) will change. A more formal description is given in Algorithm 1.

There have been many prior uses of similar E-M HMM clustering algorithms, beginning with Rabiner et al. for word recognition [47]. While there are newer variants, most HMM clustering is still done with Rabiner's original algorithm. A particularly illustrative study was done by Schliep et al. to analyze gene expression

[12]Since Baum-Welch is itself an E-M algorithm, EM-HMM clustering represents nested E-M algorithms.

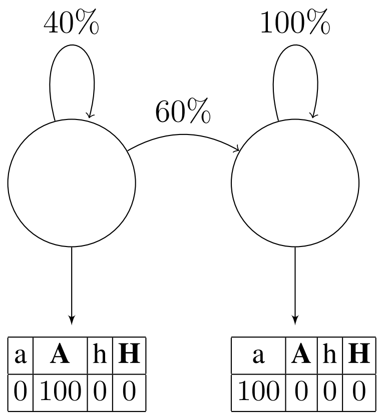
```
**Data**: Initial HMMs $M$ and Data $X$
**Result**: HMMs $M$
**while** *termination criteria not satisfied* **do**
    $M' = \{\}$;
    $\forall i \leq |M|, X_i = \{x | i = \arg\min_j l(x|m_j), m_j \in M\}$;
    **for** $m_i \in M$ **do**
        $m'_i =$ Baum-Welch$(m_i, X_i)$;
        $M' = M' \cup \{m'_i\}$;
    **end**
    $M = M'$;
**end**
return($M$);
```

**Algorithm 1:** E-M HMM

data [50]. The paper discusses, amongst other topics, the expressiveness of the models, the interpretation of results (for genetics), the inclusion of human labels, and the comparison of HMM clusters to other time series models.

HMMs as described so far do not produce sequences of unknown length; rather, the length of the sequence is a parameter for HMM sequence generation. For example, one can query an HMM for a random sequence of length 20, but to get a random sequence of length *less than or equal to* 20, one has to place a prior over the probabilities of each length. Rather than apply a transformation to standardize the length of all sequences of student actions, however, HMMs will be allowed to produce special terminal symbols (denoted here by $\omega$). For example, consider the HMMs shown in Figures 10 and 11.

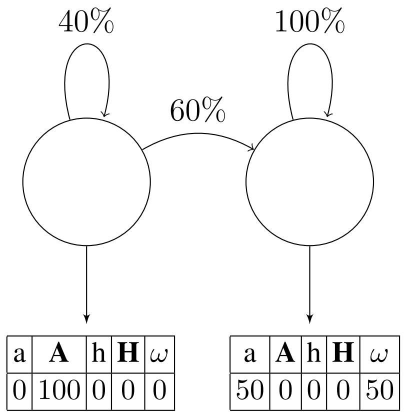
Notably, in Figure 10, the lack of a terminal symbol means that, even after transitioning to the the trap state (state 2, from which no transitions lead out), the



| a | **A** | h | **H** |
|---|---|---|---|
| 0 | 100 | 0 | 0 |

| a | **A** | h | **H** |
|---|---|---|---|
| 100 | 0 | 0 | 0 |

Figure 10: Example HMM without terminal symbol.



| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 0 | 100 | 0 | 0 | 0 |

| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 50 | 0 | 0 | 0 | 50 |

Figure 11: Example HMM with terminal symbol.

$$\begin{tikzcd}\boxed{\text{Sequential Data}} \arrow[d, "\text{Cluster}"] \\ \boxed{\text{Distribution of Tactics}} \arrow[d, "\text{Predict}"] \\ \boxed{\text{Student Learning}}\end{tikzcd}$$

Figure 12: The experimental procedure used for E-M HMM

HMM will continue to produce a's. However, in Figure 11, the presence of $\omega$ allows the HMM to generate sequences such as, for example, $\mathbf{AA}\mathrm{a}\omega$ . By simply removing or truncating the emitted terminal symbols, HMMs with terminal symbols can generate sequences of varying lengths. In practice, terminal symbols are natural to educational data sets, as students transition between steps and between problems.[13]

### 5.2.1 Results

Naive E-M HMM can be tested using the same experimental procedure used for the bag-of-words representations, except without the conversion to a bag-of-words. This experimental procedure is shown in Figure 12.[14] Using this experimental procedure, Figure 13 shows boxplots of $R^2$ results from 10 iterations of

[13]In addition, a terminal symbol at the step-level corresponds to a general-purpose symbol for step-switching at the problem-level. While switching steps terminates a sequence at the step-level, students sometimes switch back-and-forth between steps while solving a problem, thus allowing step-switch symbols to occur many times within a problem.

[14]For a more detailed version, see the bag-of-words experimental procedure shown on page 40.

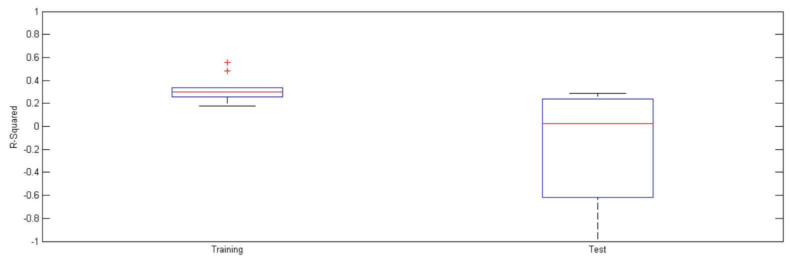


Figure 13: E-M HMM results for Geometry02-StepA8H8 — adjusted-$R^2$ on training data and $R^2$ on test data

E-M HMM on Geometry02-StepA8H8 data. Training performance (left box-plot) is reasonable in the 0.3-0.4 range for adjusted-$R^2$, but test-set performance (right box-plot, measured by $R^2$) is often worse than simply picking the mean value.[15] In this case, the withheld test set is *within student* data, which makes the performance truly abysmal.

This same result holds true across data sets and across parses: E-M HMM produces models that perform well on training data, but are overfit and do not perform well on withheld test data. However, the fact that the naive E-M HMM algorithm can fit reasonably well to training data is promising. As an alternative, I propose an algorithm that can train and select HMMs with the benefit of target information, e.g., student learning gains. I call this algorithm Guidepost EM-HMM.

[15]The mean value corresponds to $R^2 = 0$.

# 6 Guidepost EM-HMM

The following section describes the Guidepost family of HMM-clustering algorithms. This class of algorithms represent a primary contribution of this work. The core idea — embedding target information into a model selection step — is simple, as is the basic algorithm. However, later subsections will introduce important ideas that improve computational efficiency and lead to better results on empirical data sets. In particular, Guidepost Weighted-HMMs will incorporate target information into the learning of HMM parameters.

## 6.1 Guideposting

Intuitively, the main idea behind guideposting is simple. E-M algorithms suffer from local minima, and therefore must be restarted from random starting points in order to effectively explore a search space. Guideposting offers an alternative. At each local minimum, instead of randomly reseeding the E-M algorithm, Guidepost algorithms use the candidate model plus target information to intelligently select a new starting point in the search space.

One way to understand this process is to imagine two hikers trying to find the highest peak in a mountain range. The first hiker, **R**, constantly hikes[16] up the nearest slope until he reaches a mountain peak. Then **R** gets in his car and drives to a random location in the mountain range and starts the process over again.

[16]Hiking implies taking little steps as in Expectation-Maximization.

Now imagine a second hiker, **G**, who also constantly hikes up the nearest slope until he reaches a mountain peak. However, **G**, unlike **R**, then proceeds to look for a guidepost (trail sign) that points him towards another, nearby mountain that probably has a high peak. **G** is generally improving his location across iterations whereas **R** is forgetting all the useful information he learned before. Hiker **R** is an analogue for E-M with random reseeding whereas **G** is an analogue for Guidepost E-M.[17]

In practice, the exact implementation of guideposting is going to depend on the available target information $\Lambda$, some parametric assumptions about the functions $f(X) \rightarrow Y$ and $g(Y) \rightarrow \Lambda$, and the size and structure of the potential search space. However, for the kinds of targets used in education and for the space of models including HMMs, and assuming EMHMM() implements the standard E-M HMM clustering algorithm and Select() implements a model selection algorithm, one reasonable implementation is shown in Algorithm 2.

```
Data: Initial HMMs $M$, Data $X$, and Targets $\Lambda$
Result: HMMs $M$
while termination criteria not satisfied do
    $M'$ = EMHMM($M$,$X$);
    $M$ = Select($M'$,$\Lambda$);
end
return($M$);
```

**Algorithm 2:** Guidepost E-M HMM

[17]Consider another hiker **M** that corresponds to Markov Chain Monte Carlo methods. **M** does not hike all the way up the hill. Instead, **M** chooses a random point nearby and then decides, based on how high that point looks, whether to hike there. In a sense, **M** makes uninformed decisions about where to go, but makes informed decisions about whether to go; **G** makes informed decisions about where to go, but makes uninformed decisions about whether to go.

The subroutine *Select*() can be implemented in a number of ways. The most straight-forward approach is to use stepwise linear regression. The speed and simplicity of stepwise regression allows for more time and computation to be spent on the relatively more interesting underlying HMMs. Further, while linear regression is a fairly strict parametric representation, there is significant modeling flexibility in the HMMs themselves, and linear regression has proven fairly effective in a variety of educational contexts [49].

It is also useful to build in an additional bias towards simpler models and simpler collections. This helps avoid overfit and makes the resulting collections more interpretable. To accomplish this, Guidepost can start with simple models and only replace them when new ones are significantly better. Thus, in practice, an iteration of Guidepost HMM proceeds:

1. Start with a set of HMMs $M$, set of sequences $X$, and set of high-level targets $\Lambda$.

2. Assign each sequences $x \in X$ to an HMM $M_i$ s.t. $i = \arg\max_j l(x|M_j)$. This creates a partition of $X$.

3. Relearn the parameters of each HMM $M$ to maximize likelihood over its associated partition.

4. Create a matrix $S$ such that each column $S_i$ is the distribution, for student $i$, of all that student's sequences across the HMMs $M$. Thus, $S_{ij}$ would be the

probability that student $i$ generated a sequence of actions that is associated with the $j$-th HMM.

5. Solve the regression $\Lambda = S\beta + \epsilon$.

6. Let $C$ be the set of all columns of $S$ deemed significant (are statistically significant predictors of learning gain). Then define $M' = \{m_i | m_i \in M \wedge i \in C\}$.

After a certain number of iterations of Guidepost HMM or after reaching some convergence criteria, increase the number of HMMs and states. Standard convergence criteria are applicable here: $\Delta$ cluster membership, $\Delta$ log-likelihood, etc.[18]

Finally, there is the problem of selecting the best collection. The most effective way to do this is with a withheld validation set. Cross-validation is theoretically better, but withholding the last few sequences works reasonably well in practice. The general protocol for all the remaining algorithms to be discussed is as follows:

- Iteratively learn collections from training data.
- Select the collection with the highest adjusted-$R^2$ metric on withheld validation data.[19]

[18]It is also worth noting that, unlike stepwise regression, many forms of regularization make for a poor fit for this problem. Strict model selection, where a variable is either in-or-out, allows for straight-forward guideposting. More complex regularization also performs shrinkage that may have unintended consequences with regard to later iterations.

[19]The adjusted-$R^2$ is defined as $1 - (1 - R)\frac{n-1}{n-p-1}$ where $n$ is the number of students and $p$ is the number of HMMs.

- Test the selected collection against separate test data, evaluating the collection's performance using unadjusted $R^2$.

The use of adjusted-$R^2$ as a selection metric creates a bias towards simpler collections. Intuitively, Guidepost EM-HMM has two means of biasing its results towards simplicity. First, the parameters of the stepwise regression can and should be set to admit collections with some probability (e.g., p $\leq 0.1$), but to only reject collections when the p-value is quite high (e.g., p $\geq 0.3$). This parameterization biases Guidepost HMM towards keeping older HMMs, which tend to be simpler (2-state HMMs are tried before 3-state HMMs, etc). Further, using the adjusted-$R^2$ to select the best collection biases Guidepost towards collections with fewer HMMs in total. The combination of these two biases will, especially for later algorithms (with some adjustments), lead Guidepost algorithms to simple collections with only a few HMMs, each with only a few states.[20]

## 6.2 Simulations

Simulated data allows for the comparison of models and algorithms under conditions of a known true distribution. For target sequence clustering with HMMs, this allows for the exploration of several key questions:

- For what types of collections of HMMs can target clustering algorithms recover the original (generating) models?

[20]See Appendices for examples.

- Can target clustering algorithms recover the types of HMMs hypothesized by educational domain experts?
- Which target clustering algorithms recover which types of HMMs? How efficiently?
- When target clustering algorithms do not recover the original HMMs, how do they behave?

This section will focus primarily on Guidepost E-M HMM as a representative for EM target clustering algorithms. Issues of algorithm comparison (e.g., the third question above) as well as basic simulation results for each algorithm will be included in the discussion of individual algorithms. The topics for this section are as follows: issues with simulating data from HMMs, results for the recovery of human-defined collections, especially those created by domain-experts or from existing student models; an exploration of the conditions necessary for the recovery of random collections of HMMs; observed differences in the behavior of target clustering algorithms, particularly Guidepost EM, when recovering the generating collection and failing to recover the generating collection.

Figure 14 shows the experimental procedure used with simulated data. Several steps are novel compared to prior experimental procedures. First, note that there is an implicit requirement to determine how many generating HMMs to use, how many students to simulate, and how many actions to sample for each student. Second, note that the the distribution of HMMs (a probability vector for each simu-

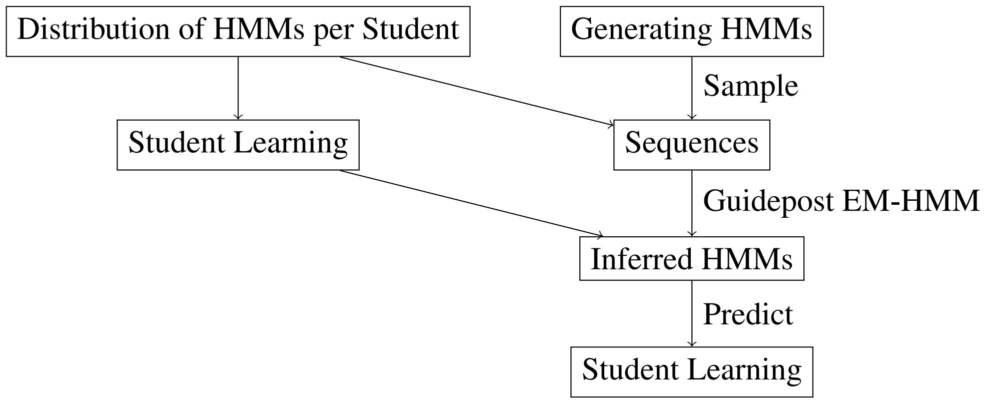


Figure 14: The experimental procedure used for simulated data

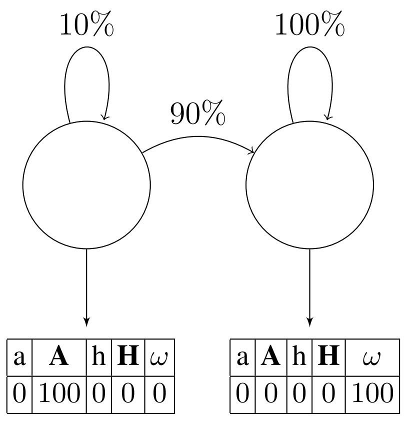
lated student) is generated independently from the HMMs themselves. In practice, both can be generated at random. Third, note that generating the actual step-data requires both a distribution across the HMMs (to determine when to sample from each HMM) and the HMMs themselves. Finally, note that the learning gains are derived from the distribution across HMMs (and some arbitrary regression coefficients) and are thus conditionally independent of the sampled sequences. This separation between the generating processes for the HMMs and the learning gains, plus the noise inherent in sampling from HMMs, results in the best possible predictive performance on simulated data being less than perfect, even when using the original generating models.

A quick note: none of the following results will include any direct structural comparisons between the generating and discovered HMMs. Directly comparing HMM structures is not only difficult, but the fact that each HMM exists in relation to other HMMs means that the structure of one individual HMM does not uniquely determine the sequences in its associated cluster. For example, take a pair of HMMs, one of which ($M_1$) generates sequences consisting of **A**'s and one of which ($M_2$) generates sequences consisting of a's. $M_1$ can have any arbitrary probability of generating a's and **A**'s so long as $P(\mathbf{A}|M_1) > P(\mathbf{A}|M_2)$ and $P(a|M_1) < P(a|M_2)$. This means that, even if $M_1$ has a $49\%$ probability of generating a's, sequences of a's can still be assigned to $M_2$ and not $M_1$. This same argument can extend to arbitrarily complicated HMM structures and alphabets.

### 6.2.1 The Naive Collection

A simple, but easily machine-learned, human-defined collection of HMMs should include distinct, barely-overlapping models representing the most commonly hypothesized student behaviors. Call this collection *Naive*. The Naive collection includes:

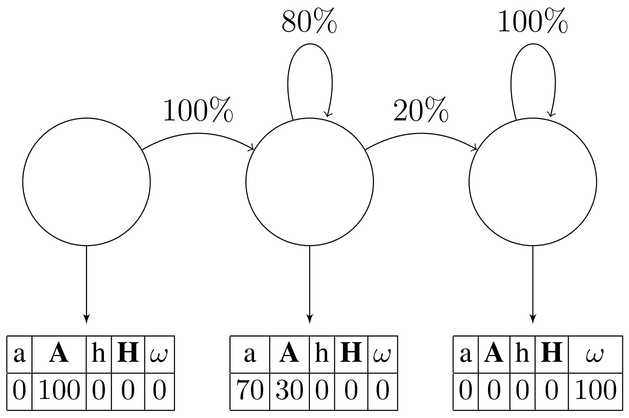
- The Correct model: this HMM generates short sequences of mostly long, correct attempts. This model is shown in Figure 15.
- The Guessing model: this HMM generates long sequences of mostly short, incorrect attempts followed by a correct attempt. This model is shown in Figure 16.
- The Hint-Abuse model: this HMM generates long sequences of mostly short hint requests followed by a correct attempt. This model is shown in Figure 17.



| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 0 | 100 | 0 | 0 | 0 |

| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 100 |

Figure 15: Naive Collection — Correct



| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 0 | 100 | 0 | 0 | 0 |

| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 70 | 30 | 0 | 0 | 0 |

| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 100 |

Figure 16: Naive Collection — Guess

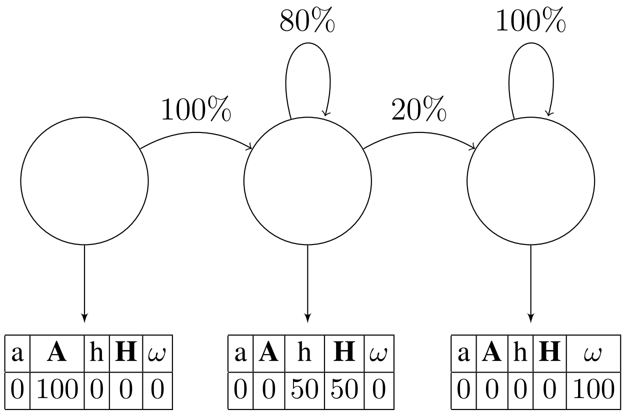


Figure 17: Naive Collection — Hint Abuse

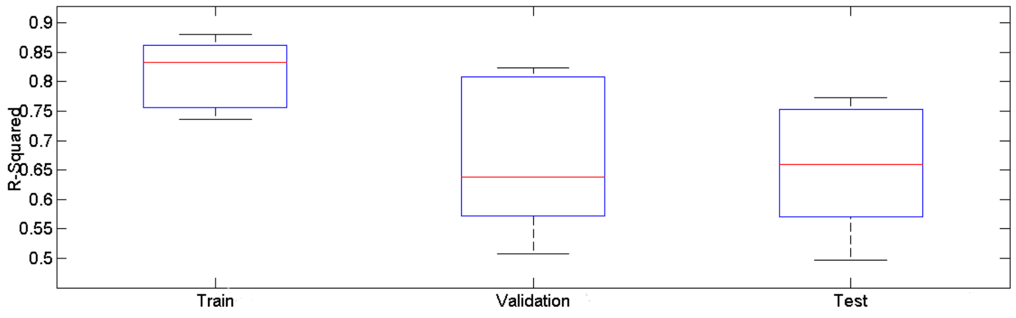


Figure 18: Naive Collection — Train/Test Simulation Results

This generating collection includes no model for correct hint-use because there is no general agreement on correct hint behaviors. Note that this collection can be trivially adapted to any set of emission symbols (e.g., $\{\mathbf{A}, \mathbf{H}\}$, $\{\mathbf{I}, \mathbf{C}, \mathbf{H}\}$).

Let there be 20 simulated students with 100 simulated sequences of actions each (a small dataset). Figure 18 shows the results of running Guidepost EM HMM on 5 separate data sets generated from this collection. Per usual, the y-axis denotes an

$R^2$ value. On the left is the highest training-set adjusted-$R^2$ performance and on the right is the test-set $R^2$ performance for the same collection. For this simple, naive model, achieving a mean $R^2$ of $0.66$ ($\rho \approx 0.81$) on test data is feasible because of the low-noise, minimally-overlapping HMMs.[21]

Figure 19 shows the relationship between the discovered models and the original model. The y-axis shows the misclassification rate[22] and each entry on the x-axis belongs to one collection of HMMs. The collections are sorted so the "best" collections of HMMs are towards the left, and for those collections, the misclassification rate sits near zero, showing that the collections all closely mirror the generating collection. This suggests that, for simple collections, Guidepost E-M reliably performs at or near ceiling. It's also worth noting that Guidepost E-M finds a class of similar collections (the first three bars) that mirror the generating collection, but performance degrades outside of this class.

### 6.2.2 Basic Collection

Unlike the Naive collection, the Basic collection features significant overlap between the sequences represented by each of its models. This overlap comes in two forms: inherent overlap due to the nature of the constructs and noise introduced by replacing structural zeros with small $\epsilon$ probabilities. The inherent overlap better

[21]When applying the original generating models to the generated data, the observed fits ($R^2$ values) fall in the range of $0.71 - 0.96$ due to noise, providing an upper bound on empirical performance.

[22]Defined in detail in Section 7.2.2 on page 96. For now, it's sufficient to note that the misclassification rate measures error and is bounded between 0 and 1.

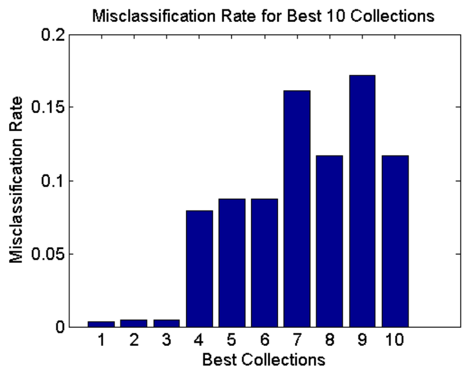
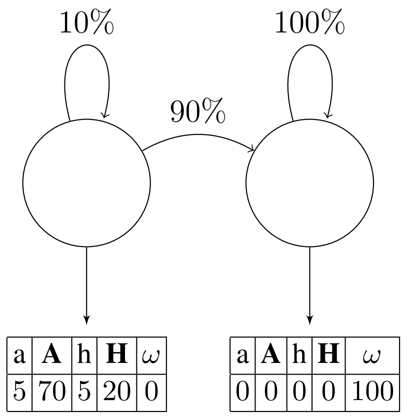


Figure 19: Naive Collection — Misclassification rate for the best discovered collections

represents actual uncertainty about the world, while the noise is simply a natural byproduct of real-world phenomena. The Basic Collection includes:

- The Basic Correct model: this HMM generates short sequences of mostly long, correct attempts. It differs from the Naive Correct model in that it introduces noise. This allows for an inherent overlap since it can now generate sequences similar to a Guessing sequence (plus generally noisy sequences). However, it is still highly biased towards generating **A**. This model is shown in Figure 20.

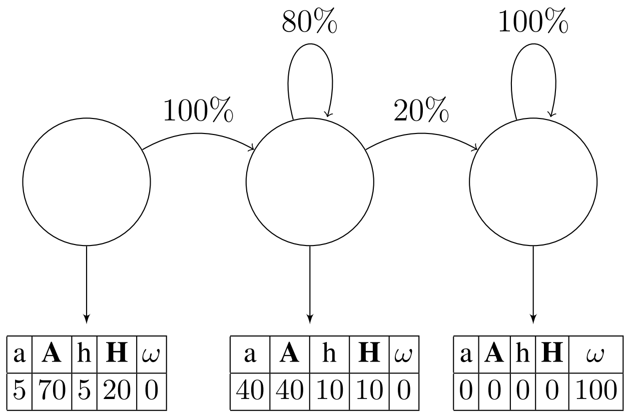
- The Basic Guessing model: this HMM generates long sequences of mostly short, incorrect attempts followed by a correct attempt. It differs from the Naive Guessing model in that there is a probability of emitting sequences like those from Basic Correct, and there is noise present in all non-termination states. This model is shown in Figure 21.



| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 5 | 70 | 5 | 20 | 0 |

| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 100 |

Figure 20: Basic Collection — Correct



| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 5 | 70 | 5 | 20 | 0 |

| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 40 | 40 | 10 | 10 | 0 |

| a | **A** | h | **H** | $\omega$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 100 |

Figure 21: Basic Collection — Guess

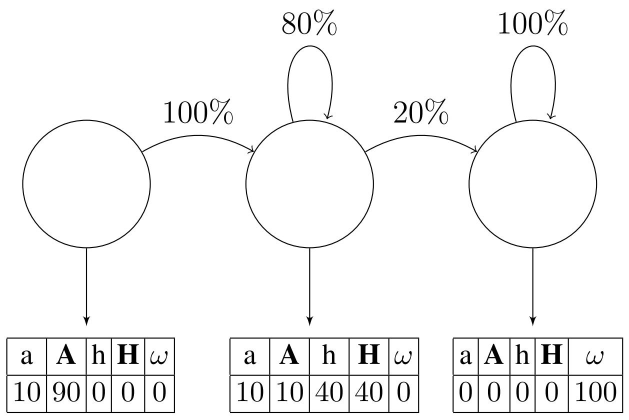


Figure 22: Basic Collection — Hint Abuse

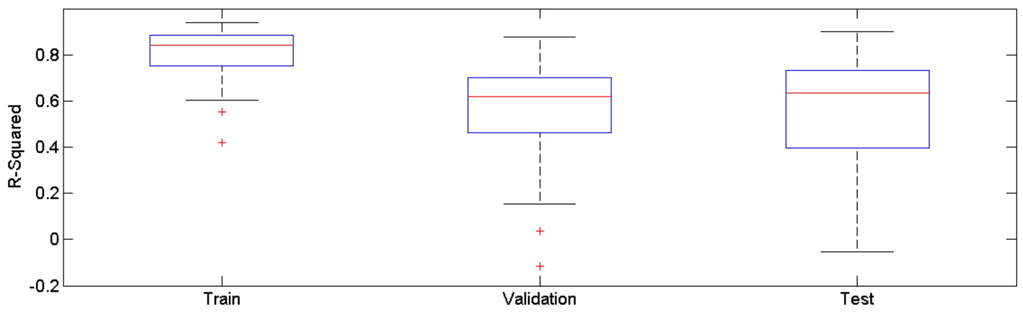


Figure 23: Basic Collection — Train/Test Simulation Results

- The Basic Hint-Abuse model: this HMM generates long sequences of mostly short hint requests followed by a correct attempt. It differs from the Naive Hint-Abuse model by the introduction of noise in its emissions. This model is shown in Figure 22.

Figure 23 mirrors Figure 18, except that it shows results for the Basic Collection. Performance is marginally reduced, but an $R^2$ in the range of $0.4-0.6$ on withheld

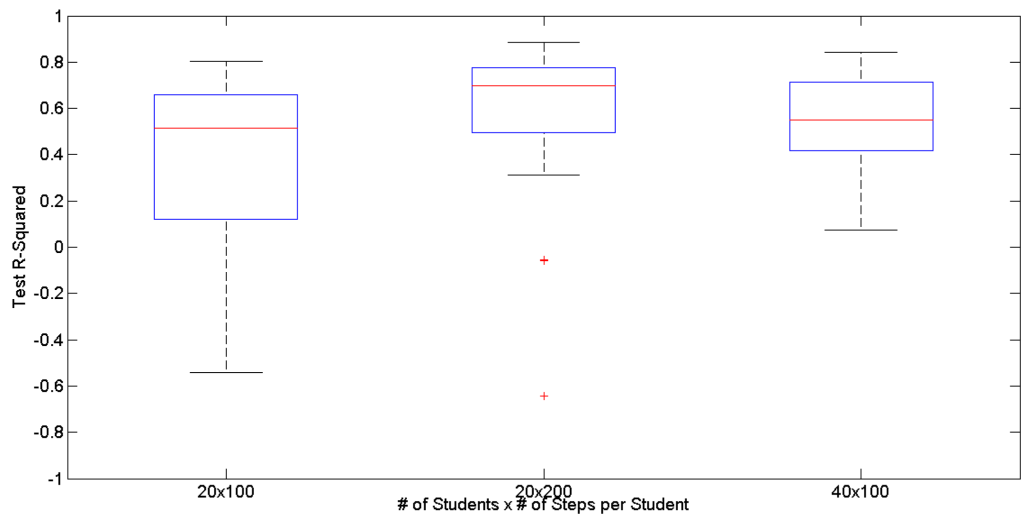


Figure 24: Basic Collection — Test Performance By Size

test data is reasonable.[23] This collection, unlike the Naive collection, is a fairly close facsimile of collections found in actual empirical data, and thus makes a more realistic collection for additional simulations.

One obvious question is: How would an increase in available data affect the performance of the algorithm and its resulting collections? And in particular, which matters more, the number of students or the number of steps / sequences? Figure 24 shows three sets of test $R^2$ results: the first is for 20 simulated students with 100 simulated steps each, the second is for 20 simulated students with 200 simulated steps each, and the last is for 40 simulated students with 100 simulated steps each. The latter two data sets both offer twice the total number of observed steps, and both data sets result in fewer poor results, but there is no practical or

[23]When applying the generating models to the generated data, the observed fits ($R^2$ values) fall in the range of $0.62 - 0.94$, providing an upper bound on empirical performance.

significant difference between the two cases. That is, at least when both students and steps are scarce, increasing the number of either is a valuable exercise.

### 6.2.3 Conditions for the Recovery and Non-Recovery of Generating Collections

However, just because it's possible to recover approximations of the generating models when they are simple, researcher-determined HMMs, does not mean it is possible to recover anything useful for more complex, more arbitrarily-determined HMMs. To address this question, Figure 25 shows the performance (on test data) of Guidepost EM-HMM using simulated data generated under various constraints. This figure is similar to prior ones, except rotated. It shows $R^2$ on the y-axis (left-to-right) with the boxplots representing results for each type of generating collection. The first grouping, for example, consists of test-set performance using completely random HMMs to generate data. Unsurprisingly, it performs poorly. There are several plausible reasons: for random collections, individual HMMs tend to have reasonable likelihoods of producing sequences from other HMMs; the complexity of random HMM transition matrices are simply too complicated; random HMMs lack strongly defined emission profiles; random HMMs have priors over their states that prevent the easy recovery of their structure. Three of these possibilities are easily addressed. When generating random HMMs, the complexity of the transition matrix, emissions matrix, and prior probability vector can be restricted by increasing the number of structural zeroes. Figure 25 shows

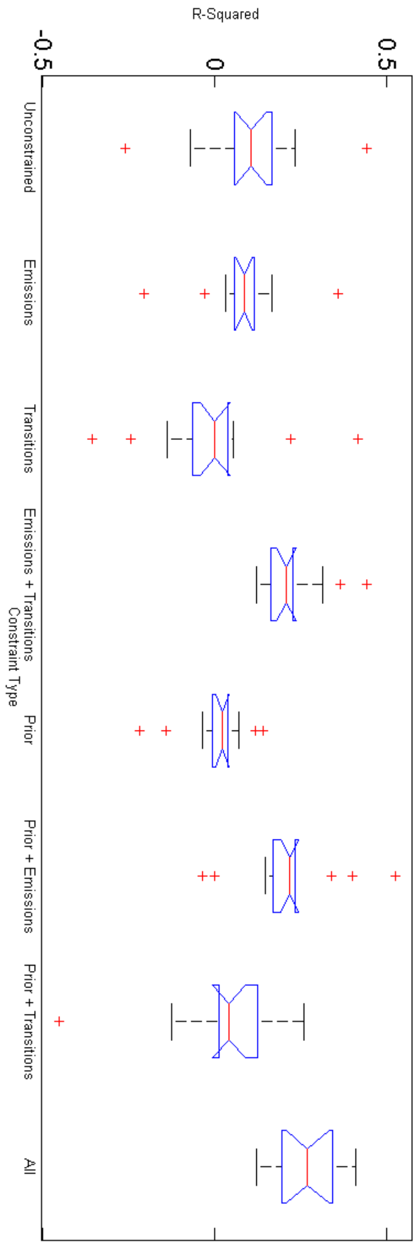


Figure 25: Simulation results for random HMMs under various constraints

the results of simulations using various combinations of these constraints.[24] Each simulation was performed twenty times.

Of the constraint classes shown in Figure 25, only three are significantly better than the others (most lead to Guidepost-learned collections with $R^2$ performance close to zero which is statistically indistinguishable from pure noise). The learnable-collections involved constraining: the emissions matrix and the transitions matrix, the prior probability vector and the emissions matrix, or all of the above. Interestingly, constraining the emissions matrix appears, in this case, to be a necessary, but not sufficient, means to improve test-set performance. These results have implications regarding the types of generating models that can feasibly be found by Guidepost EM-HMM: they must have reasonably well-identified, well-separated states (with a high probability of emitting one or two symbols), and they should have a prior probability highly biased towards a single state. In practice, as will be shown further on, this holds true in empirical data as well.

### 6.2.4 Simulation Summary

The simulation results above lead to some important implications. First, there exist generating collections of HMMs for which Guidepost EM-HMM is an effective algorithm capable of recovering an approximation of the original distribution. Second, there also exist generating collections (e.g., Random) for which Guide-

[24]More precisely, these constraints are: one state with non-zero prior, one symbol emitted per state, and/or half of the transition matrix constrained to zero. The exact form of these constraints was motivated by empirically-discovered models.

Table 3: Geometry02 Fits — Test Data

| $\tau$ | **Problem-Level ($R^2$)** | **Step-Level ($R^2$)** | **# of HMMs** | **Max # of States** |
|---|---|---|---|---|
| 6 | 0.24 | 0.19 | 5 | 3 |
| 8 | 0.17 | 0.25 | 5 | 4 |
| 10 | 0.22 | 0.26 | 4 | 4 |

post HMM will find collections that clearly do not fit the data. Third, it is not trivial to constrain random HMMs to produce Guidepost-learnable HMMs. For example, constraining only the emissions matrices is not sufficient. These implications allow for some belief that, if Guidepost HMM finds a good collection in empirical data, then there is actual, identifiable structure in that data.

## 6.3 Applied Results

However, simulation results are only part of the story. It is even more important that target clustering algorithms can find predictive models in real data sets. In the first application of Guidepost EM-HMM, Shih et al. found that Guidepost E-M for HMMs was far more effective at finding predictive (and somewhat interpretable) student models than naive E-M HMM clustering [53]. Table 3 shows those original test-set results. In the table, $\tau$ represents threshold values. Notably, that study used a single-threshold method where there was one threshold for all actions and with actions grouped into only two categories: Attempts and Hints. The published analysis only used step-level sequences, but the problem-level results are included here for convenience.

As importantly, the results were at least somewhat interpretable. First, as is shown in Table 3, the discovered collections are simple, containing 4-5 HMMs of at most 3-4 states each. Further, the actual HMMs are easy to interpret: Figure 26 shows a hidden Markov model negatively associated with learning gains. It is evidently a model for guessing, as it usually generates one **A** before transitioning to the second state and generating mostly a's. This corresponds to students entering many incorrect answers in a row very quickly. Guessing is both an element of gaming [4] and poor help-seeking [1], both of which are also negatively associated with learning, so the model's association with learning is reasonable as well.

On a related note, it's also reasonable to ask if the adjusted-$R^2$ statistic, computed on validation data, is a good predictor of test data performance. To that end, the correlation between the adjusted-$R^2$ on validation data and the $R^2$ result on test data is 0.956, averaged across many iterations. While it does occasionally make mistakes, in the long-term, over many restarts, the adjusted-$R^2$ computed on validation data is a good predictor of test-set performance.

However, that original analysis was overly simplistic in a number of ways. First, it did not incorporate correctness or step switching. Second, it glossed over problems of stability of results: E-M is notorious for requiring many, many random restarts to achieve good performance. Third, it ignored the possibility that different actions should have different thresholds. Fourth, it glossed over the problem of differences between high-performing collections. Two HMMs were commonly recurring throughout the collections, but the other HMMs were different between

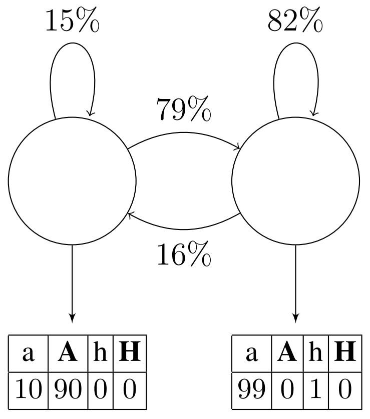


Figure 26: Repeated Guessing HMM for $\tau = 6$, 02 data

"best" collections.

The following section will present a superior algorithm to the basic Guidepost EM-HMM algorithm. This algorithm will be used to address the above three issues.

## 6.4 Limitations

Guidepost HMM produces simple, interpretable models that both predict student learning and cluster their observed behaviors. However, the algorithm has a number of limitations: it finds local minima, it converges slowly, and it's dependent on random seeds. The first limitation, that Guidepost finds local minima, is a general problem for E-M algorithms. Guidepost HMM is worse than most E-M algorithms because of the complexity of the search space and because it uses nested

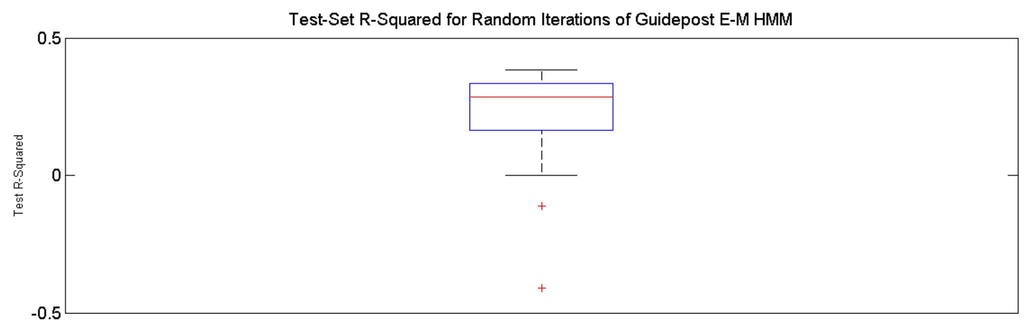


Figure 27: Test-set $R^2$ across iterations of Guidepost HMM

E-M loops;[25] Guidepost is better than most E-M algorithms because it uses $\lambda$ for model selection, and thus has more insight into which models to keep. To see the extent of the problem, consider Figure 27, which shows a boxplot summarizing the result of 30 separate runs of Guidepost HMM. When picking the "best" collection using adjusted-$R^2$ performance on validation data, the "best" collection has a test-set $R^2$ of 0.25, as shown above. However, the variability in the results of individual runs[26] suggests that many, many restarts are needed to confidently find a good solution.

The only way to completely resolve local optima is to shift away from E-M algorithms. However, there is room for improvement even without such a drastic step: Figure 27 shows that Guidepost generates a wide range of candidate collections with performance ranging from poor ($R^2$ ¡ 0) to mediocre (e.g., $R^2 = 0.1$) to great. This diversity of results means that incremental improvement is still worthwhile.

[25]The nested E-M loops result from both HMM parameter learning and HMM clustering being EM-algorithms.

[26]The variability of individual runs is shown by the lower whisker of the boxplot almost reaching zero and by the presence of multiple outliers below zero.

Without abandoning E-M, it is possible to reduce the number of poor candidates by embedding target information into the HMM parameter learning sub-loop.[27]

The second limitation, that Guidepost HMM converges slowly, is also a result of the nested E-M loops. However, the convergence rate of Guidepost E-M could be improved by replacing just one loop. For example, if a different algorithm was used to train the HMMs, then much of the iterative optimization could be avoided.[28]

The final limitation, that Guidepost depends heavily on random seeds and thus requires multiple restarts, is a direct consequence of the Guidepost model selection process. Every iteration of the outer clustering loop leads to a model selection step where some of the HMMs are removed. Afterwards, they are replaced with new, random HMMs. This repeated randomization makes determining the appropriate number of iterations very difficult. Removing this limitation would require moving to an algorithm class other than Guidepost.

[27]See Guidepost Weighted HMMs, Section 7.2, page 86.
[28]For examples of possible replacement algorithms, see Section 11.1, page 163.

# 7 Guidepost Variations

## 7.1 State-Limited Guidepost

One limitation not mentioned above is that the basic Guidepost EM-HMM algorithm searches over an $m$x$k$-matrix of possible numbers of HMMs ($m$) and possible numbers of states ($k$). One simple change to the original algorithm is to restrict the range of possible HMMs and states to the lower-triangular matrix, thus preventing cases where individual HMMs have more states than there are HMMs in total.

The resulting improvement in results is evident in Figure 28, which shows the performance of Guidepost EM-HMM and the state-limited equivalent on the Basic Collection (described earlier). The naive, non-state-limited test-set $R^2$ is shown on the left; the state-limited test-set $R^2$ is shown on the right. Further, the state-limited version of Guidepost EM-HMM more than halves the computational cost. For all additional Guidepost variants, this modification will be assumed.

## 7.2 Guidepost E-M Discriminative and Weighted HMMs

The next Guidepost variant requires more significant background. First, standard hidden Markov models are generative: they learn a model with a high likelihood of generating observed positive examples. This approach is effective for individual HMMs, but for collections of HMMs, the standard generative solution only

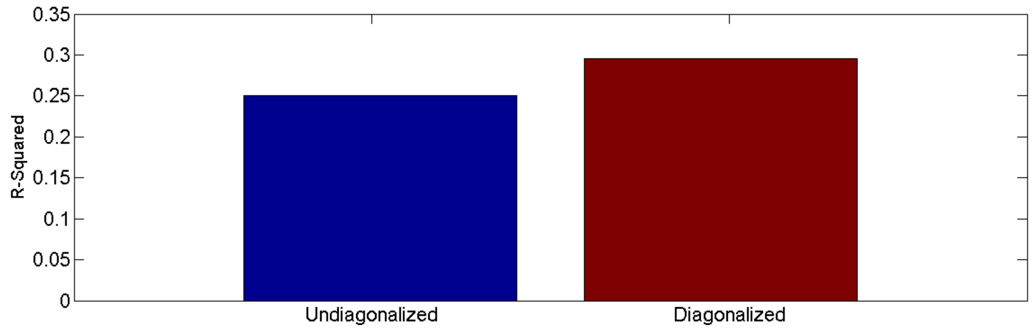


Figure 28: Comparison of Guidepost EM-HMM and State-Limited Guidepost EM-HMM

achieves an optimal solution if:

- There is an infinite amount of training data available.
- The local optimization algorithms for each HMM combine to find the aggregate maximum likelihood solution.
- HMMs are an accurate representation of the original generating model.

None of these criteria hold true in practice. The first issue, that of training data, is standard for parameter optimization with E-M. However, the other two issues are unusually problematic for Guidepost EM-HMM: finding HMM clusters involves embedding an E-M search within another E-M search, aggravating the second issue, and even under the most optimistic assumptions, HMMs are nowhere near a perfect representation of human cognition and learning, aggravating the last issue.

The main reason for this limitation is that, when used in an E-M clustering algorithm, each HMM is only aware of the sequences in its partition and has no knowledge of other partitions. For each HMM, assuming that $x \in P_k$ represents a sequence in the partition for the $k$-th HMM $M_k$, the objective function is:

$$\sum_{i}^{|P_k|} \ln P(x_i|M_k) \tag{1}$$

However, there is an alternative approach. Discriminative HMMs use both positive and negative examples in training. One simple discriminative training algorithm is the Maximum Mutual Information Estimation (MMIE) algorithm [3]. In general, for $n$ sequences $x_i$ and $n$ corresponding HMM-assignments $m_i$, MMIE optimizes the objective function:

$$\sum_{i}^{n} \ln \frac{P(x_i|M_{m_i})P(M_{m_i})}{\sum_m P(x_i|M_m)P(M_m)} \tag{2}$$

Optimizing the MMIE objective function maximizes the probability of a sequence conditional on its being generated by the most probable HMM, but also minimizes the probability of a sequence conditional on its being generated by all other HMMs. This creates a discriminative training process where negative examples can contribute to parameter learning.[29] Unlike the basic E-M HMM algorithm, which maximizes the likelihood for $P(x_i|M_k)$, MMIE maximizes the a-posteriori probability of the collection given the data.[30]

[29]Negative examples being defined, for an HMM $M$, as all sequences not assigned to $M$.

[30]Assuming the probability of the models in the collection are fixed.

In practice, the $P(M)$ values are a critical point. While it's possible to optimize an MMIE function where $P(M)$ is replaced by the empirical estimator, $\hat{P}(M)$, this does not work well in practice.[31] Instead, $P(M)$ is usually estimated from some external data. For example, MMIE is most frequently used in speech recognition, where $P(M)$ is often estimated from the underlying language model.[32]

If $P(M)$ was replaced by $\hat{P}(M)$, the MMIE objective function would be a direct replacement for the standard likelihood function. However, it is possible to incorporate $\Lambda$ information into the objective function directly, allowing targets to be embedded as a lower level of the algorithm. Define, for each sequence $x_i$, $G(x_i)$ as the normalized, re-centered learning gain for the student associated with sequence $i$.[33] Further, define $G(M, x_i)$:

$$G(M, x_i) = \begin{cases} G(x_i) & \text{if } M \text{ is positively associated with learning} \\ 1 - G(x_i) & \text{if } M \text{ is negatively associated with learning} \end{cases}$$

Then, an alternative objective function for HMMs (using MMIE) would be:

[31] $\hat{P}(M)$ would correspond to the percentage of sequences in model $M$'s partition. Since this estimate will change every time $M$ changes, it does not provide effective guidance for learning algorithms.

[32] For speech recognition, each HMM $M$ often corresponds to a single word and $P(M)$ is usually the probability of that word in the language.

[33] That is, if the student who performed sequence $x_i$ had a pre-post learning gain of 0.8, $G(x_i) = 0.5 + 0.8 \cdot 0.5 = 0.9$. Whereas pre-post learning gain falls in the range $[-1, 1]$, $G(x_i)$ falls in the range $[0, 1]$.

$$\sum_{i}^{n} \ln \frac{P(x_i|M_{m_i})G(M_{m_i}, x_i)}{\sum_m P(x_i|M_m)G(M_m, x_i)} \tag{3}$$

The key idea is to replace $P(M)$, which is usually an a priori probability, with something comparable found in education data. $G(M, x_i)$ essentially provides a noisy estimate of $P(M)$. Unlike in speech recognition where there exists a known (or easily estimated) probability for each model (word), education data sets have no equivalent a priori probability. The closest equivalent in these data sets is an estimate of the probability of each learning tactic *given what is believed about how the tactic impacts learning*. This new objective function allows for more weight to be placed on observed sequences originating from more extreme students (e.g., for positive models, extra weight is added to sequences generated by a good learners). One interpretation of $G(M, x_i)$ is as a noisy estimate of the probability that a sequence $x_i$ is an example of a positive behavior (assuming $M$ is an HMM positively correlated with learning). Under this interpretation, $G(M, x_i)$ discriminates between sequences that reflect the HMM's association with learning and those that do not. Admittedly, $G(M, x_i$ exhibits a different type of discrimination than the usual, but one that is just as effective.

To summarize this approach: take the basic Guidepost E-M HMM algorithm, but replace the usual objective function with Equation 3. Let this algorithm be known as Guidepost E-M Discriminative HMM.

If the same idea is applied to the basic E-M HMM objective function, adding the learning gains as a weight, the result is:

$$\sum_{i}^{|P_k|} P(x_i|M_k)G(M_k, x_i) \tag{4}$$

Let this algorithm be known as Guidepost E-M Weighted HMM. The coefficient in the Weighted HMM objective function can be thought of in the same way as the coefficient in the Discriminative HMM objective function, but there is no longer a denominator that allows (or requires) optimizing across the whole set of models. However, both algorithms do allow for the direct incorporation of $\Lambda$ values into the search algorithm by re-weighting each probability by the probability of the associated student exhibiting learning. Thus, HMMs that are positively associated with learning will emphasize sequences in the partition that are frequently used by students with high learning gains, and vice versa for HMMs that are negatively associated to learning.

The advantage of the Weighted version over the fully Discriminative version is that it only uses $\frac{n}{m}$ examples per HMM and so will train faster. Guidepost E-M Weighted HMM still discriminates between positive and negative examples, but unlike Guidepost E-M Discriminative HMM, it discriminates exclusively on whether behaviors predict learning, not on whether they belong to a specific model. In practice, as I show next, Guidepost E-M Weighted HMM performs as well as

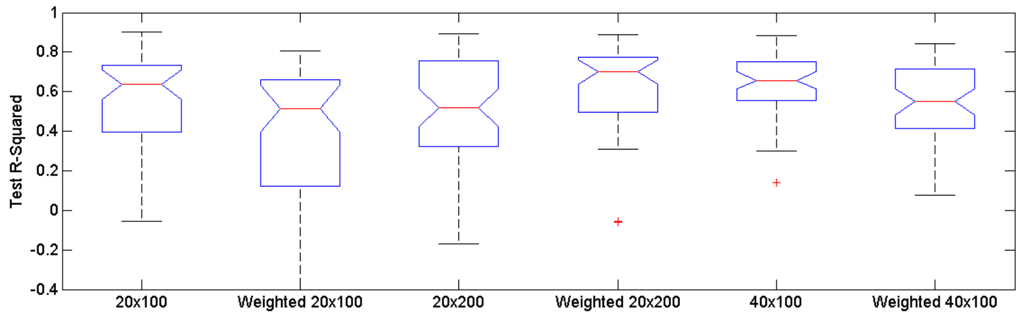


Figure 29: Comparison of fit for Weighted and Unweighted HMMs on various simulated data sets

the fully Discriminative version.[34] However, the root rationale for both algorithms is the same.

### 7.2.1 Simulation Results

Figure 29 shows the performance of Weighted and Unweighted HMMs on the Basic Collection for different data sizes. The y-axis is test $R^2$, per usual, and the x-axis alternates between a set of results for Unweighted Guidepost and a set of results for Weighted Guidepost. Notably, there appears to be no significant difference, with Weighted sometimes slightly outperforming Unweighted and vice versa. Even if this was the final result, Guidepost Weighted HMM would be an improvement, as the weighted solution is state-limited, which makes it train faster, and the weighting itself tends to make the individual HMMs converge even faster

[34]Replacing $P(M)$ with $G(M, x_i)$ means that the MMIE objective function is no longer equivalent to minimizing the Kullback-Leibler divergence for $P(M|x)$, and is thus no longer optimal. This sub-optimality is likely one reason that the Discriminative solution does not out-perform the simpler Weighted solution.

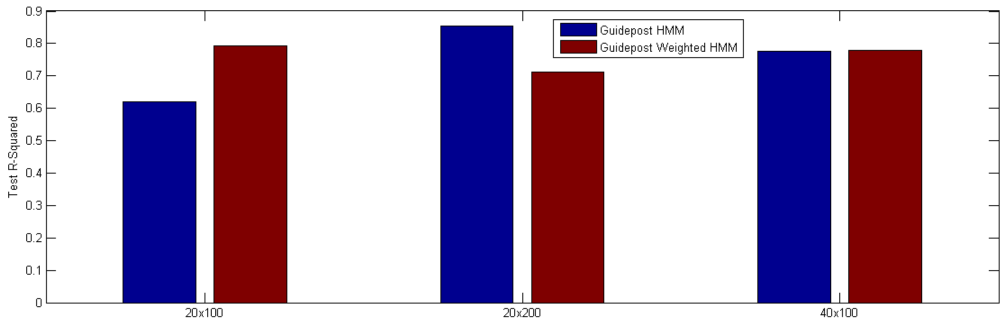


Figure 30: Comparison of best fit for Weighted and Unweighted HMMs on various simulated data sets

than before.

However, Figure 29 somewhat confuses the point. E-M is subject to the fickle randomness of its starting collection; what is really important is, given multiple runs of an E-M clustering algorithm, how good is the result? For that, see Figure 30, which shows the performance of weighted and unweighted HMMs on simulated data when only one collection is selected (by using validation performance) across all runs. Unfortunately, the results are mixed once again, with Weighted Guidepost only outperforming Unweighted Guidepost in the experiment with 20 students and 100 sequences per student.

### 7.2.2 Applied Results

Figure 31 is equivalent to Figure 30 except that it shows the performance of both algorithms on Geometry02-StepMean. Here, Weighted Guidepost finally shines, clearly outperforming Unweighted Guidepost. For reasons unknown, on real-

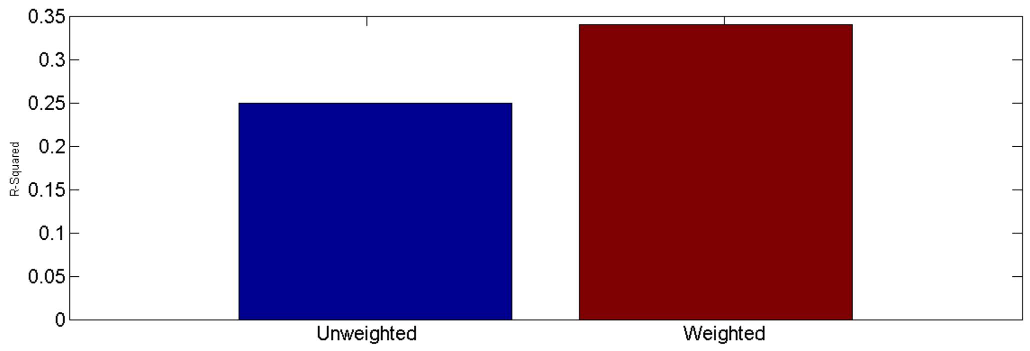


Figure 31: Comparison of Weighted and Unweighted HMMs on Geometry02-StepMean

world data, Guidepost works better with the weighted HMMs than without them. This difference in performance suggests some discrepancy between the simulated data (results shown before) and real data (results shown here).

Guidepost Weighted-HMMs is the final (and most advanced) version of Guidepost to be discussed in this thesis. It thus presents an opportunity to address several lingering questions left unanswered in the Guidepost EM-HMM section. They were:

- Is there any improvement in the results when different actions have different thresholds? For example, attempts and hints could both be thresholded by their mean value, or they could both be thresholded at the same value: is there an improvement when the thresholds are allowed to be distinct?
- Is there any improvement in the results when using a different symbol set, e.g., if correctness is added in as a splitting point?

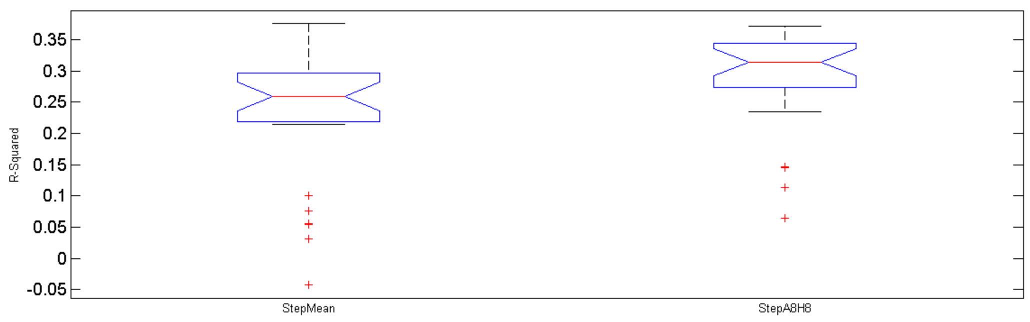


Figure 32: Comparison of test-set performance for StepMean and StepA8H8

- How stable are the resulting HMMs? Is there a substantial difference between the “best” HMM and other “good” HMMs?

For question one, Figure 32 shows that the StepMean parse, which splits at about 7.5 seconds for attempts and a bit over 1 second for hints, performs worse in practice than the StepA8H8 parse. On the other hand, in practice, most choices of thresholds do not perform anywhere near as well as the StepMean. Thus, for a novel data set, the StepMean is a good first choice, but it should be possible to determine better thresholds by other means. Explanations for this phenomenon relate to the actual structure of learned HMMs and will be covered in the final sections.

For question two, Figure 33 shows that there is no reliable difference between splitting on corrects versus not. There are slight differences between performance for the best collections (as selected by validation adjusted-$R^2$) between models with corrects or without, but adding correctness results in only a slight improvement for one set of thresholds and a slight reduction for another set. For this data

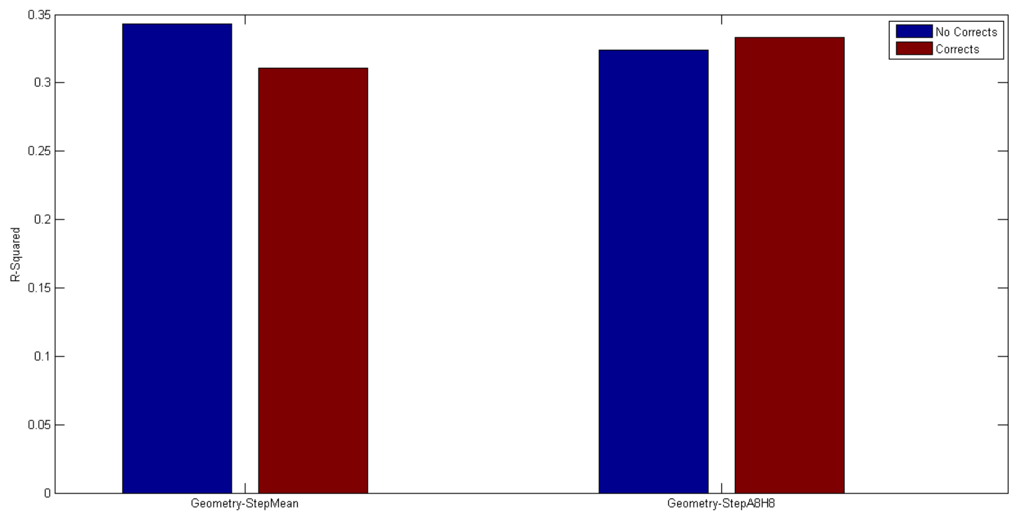


Figure 33: Comparison of test-set performance for StepMean and StepA8H8 with correctness and without

set, at least, adding correctness as a splitting criteria does not help. However, the question of using correctness will be revisited later for another data set with more dramatic results.[35]

For question three, the first step is to define a comparison function for a pair of HMMs. First, directly comparing HMM structures may be counter-productive as these comparisons are not only difficult to define and compute, but the fact that each HMM exists in relation to other HMM clusters means that the structure of one individual HMM does not uniquely determine the sequences that it classifies. For example, take a pair of HMMs, one of which ($M_1$) generates sequences consisting of **A**'s and one of which ($M_2$) generates sequences consisting of a's. $M_1$ can have any arbitrary probability of generating a's and **A**'s so long

[35]See Section 9.4, page 121.

as $P(\mathbf{A}|M_1) > P(\mathbf{A}|M_2)$ and $P(a|M_1) < P(a|M_2)$. This same argument can extend to arbitrarily complicated HMM structures.

That said, there have been a number of approaches to comparing HMMs, such as the Probabilistic Product Kernel [28], but these are designed to operate over the entire space of possible sequences and may not apply to the actual empirical distribution as observed in real-world data. Instead, consider this definition of misclassification: if two HMMs have a high disagreement when applying labels to observed sequences, then they are dissimilar; however, if two HMMs agree on labels for all the observed sequences, then any structural differences between them are unimportant for practical purposes.

Unfortunately, differences in the number of HMMs in each collection make this comparison somewhat complicated. It's possible, for example, for a 5-HMM Collection and a 2-HMM Collection to be almost identical. Call these collections A and B. HMM $A_1$ may match HMM $B_1$, while the other four HMMs $A_2$, $A_3$, $A_4$, and $A_5$ may be subsumed by $B_2$. Then, within the limits of their model complexity, A and B are the same collection. This example illustrates two major difficulties in computing the misclassification rate: the labels for HMMs may be arbitrarily permuted between collections and one collection may have more HMMs than the other.

Without loss of generality, assume A is the collection with more HMMs and B is the collection with fewer. Let $S_i$ be the set of sequences assigned to the $i$-th HMM in collection A and let $L(B, S_i)$ be the most common label applied by collection

Table 4: Mutual misclassification rate for the 5 Best Collections in Geometry02-StepA8H8

| | **Collection 1** | **Collection 2** | **Collection 3** | **Collection 4** | **Collection 5** |
|---|---|---|---|---|---|
| **Collection 1** | 0 | 0.0700 | 0.1060 | 0.0497 | 0.0487 |
| **Collection 2** | 0.0700 | 0 | 0.0550 | 0.0713 | 0.0647 |
| **Collection 3** | 0.1060 | 0.0550 | 0 | 0.0796 | 0.0954 |
| **Collection 4** | 0.0497 | 0.0713 | 0.0796 | 0 | 0.0633 |
| **Collection 5** | 0.0487 | 0.0647 | 0.0954 | 0.0633 | 0 |

B to set $S_i$. Let $Y(j, S_i)$ be the number of sequences in $S_i$ that do *not* have the label $j$ (assigned by B). Let $m$ be the number of HMMs in A. Then

$$\sum_i = 1^m |S_i| - Y(L(B, S_i), S_i) \tag{5}$$

represents an approximation of the misclassification rate. In particular, it assigns to each label in A the most commonly co-occurring label from collection B and computes the misclassification rate assuming those labels match. This relationship can be many-to-one, as in the example given above. Overall, this metric is only an approximation and can be an over- or under-estimate. Call this the *mutual misclassification rate*. To avoid issues of asymmetry, assume that the mutual misclassification rate is the minimum of applying Equation 5 to both (A,B) and (B,A).

Table 4 shows the mutual misclassification rate[36] for the five best collections found by Guidepost Weighted HMMs in Geometry02-StepA8H8. The highest misclas-

[36]Misclassification rate is bounded between 0 and 1.

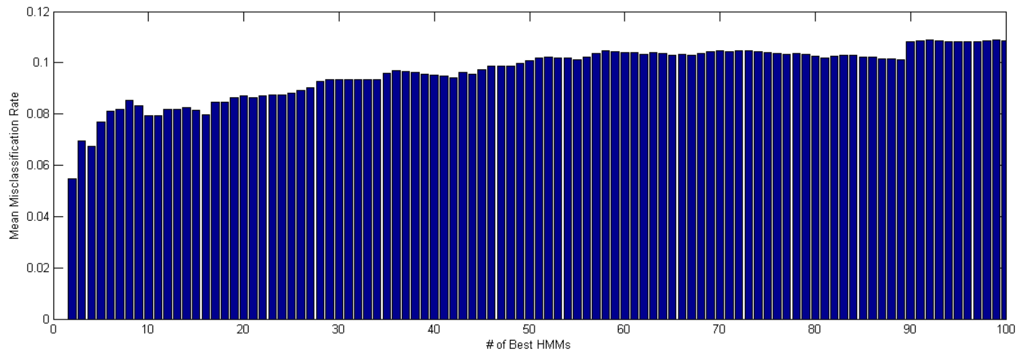


Figure 34: Mean mutual misclassification rates for the best collections in Geometry02-StepA8H8

sification rate is only about $10\%$ and the mean is about $5.6\%$.

Even for larger sets of collections, the mutual misclassification rate is reasonably low. Figures 34 and 35 show the mean mutual misclassification rate for ever larger sets of "best" collections, e.g., the tenth bar shows the mean mutual misclassification rate for the top 10 collections. Until around the $90$-th collection, the averaged mutual misclassification rate stays below $10\%$. This suggests that there exists a class of similar, high-quality HMMs.

This result represents the final instance of general, theoretically-driven results for Guidepost algorithms. Later sections will address the performance of Guidepost algorithms on other data sets, as well as explore the resulting models. However, it is sufficient at this point to note that Guidepost is effective on both empirical and simulated data sets, and exhibits desirable behaviors (e.g., does not recover totally random models).

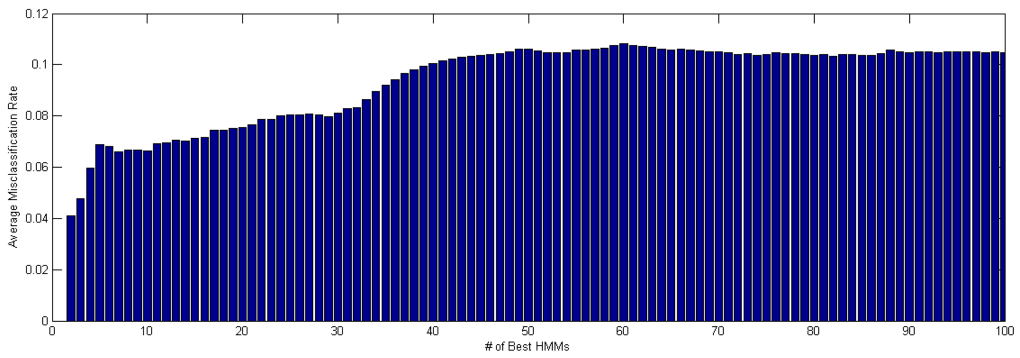


Figure 35: Mean mutual misclassification rates for the best collections in Geometry02-StepMean

## 8 Metric Learning

Target clustering, as a paradigm, extends beyond HMMs and Guidepost algorithms. For example, metric learning presents another opportunity for target clustering. Metric learning algorithms involve a decidedly different approach to classification problems. They have their roots in algorithms that rely primarily on measures of distance or similarity between data points. Example algorithms include agglomerative clustering and spectral clustering. Metric learning algorithms adjust or relearn the distance metric to provide a better solution, whether by whitening or regularizing in the basic cases, or by incorporating additional data into the distance metric itself in more complicated cases.

There are three underlying questions before applying a metric learning algorithm:

- What is the clustering algorithm?

- What is the distance metric?
- How will the distance metric be adjusted?

Target clustering, as a paradigm, only addresses the third question. However, as was done with algorithms like k-means, it is important to show that basic sequence clustering solutions will not work well without adjusting for $\Lambda$ information. This section begins with a discussion of some basic metric learning algorithms, then demonstrates these algorithms using a simple edit-distance metric. Following that is a section detailing theory and results for reweighting distance metrics using $\Lambda$ information. Finally, there are discussions of other potential target clustering algorithms using metric learning.

## 8.1 Basic Algorithms

In this section, the two basic algorithms under consideration are hierarchical agglomerative clustering (HAC) and spectral clustering. There exist many other clustering methods that depend only on measures of similarity or distance between points, but these represent canonical examples. In addition, both make few assumptions about the distance metrics and can thus be easily adapted.

HAC starts with each data point in its own cluster. During each iteration, it groups the two closest clusters together. Over the course of many iterations, HAC produces a dendrogram, which is a graph with a tree-structure that shows the hierar-

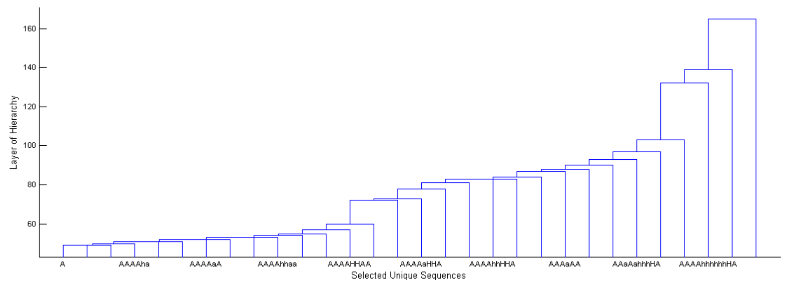


Figure 36: Example of a dendrogram

chy of clusters. An example dendrogram is shown in Figure 36. From bottom-to-top, the graph shows the order in which clusters are merged. HAC is agglomerative because it takes a bottom-up approach, aggregating smaller clusters into larger ones until it has created a single, size-$n$ cluster.

As will hold true for all future dendrograms, Figure 36 collapses across identical sequences, e.g. there is only one data point representing all sequences of the form **AAA**a. In general, there are still too many unique sequences to show in a single graph, so only randomly-selected, but frequently-occurring sequences will be shown.

In practice, given an input matrix $X$, HAC assumes the existence of three functions: a distance function $d(x_i, x_j)$; a distance matrix $D$, where $D_{ij} = d(x_i, x_j)$; and a linkage function $L(c_i, c_j)$ where $c_i$ and $c_j$ are clusters (sets) of points in $X$. HAC then outputs a hierarchy of clusters as represented by a history of cluster merges, $H$. HAC is implemented as follows:

Before this point, I have not defined the linkage function $L(c_a, c_b)$. The purpose

```
Data: $n \times n$ matrix $D$
Result: history of merges $H$
$H = \{\}$;
$C = \{c_i | c_i = \{x_i\}, i \leq n\}$;
while $|H| < n$ do
    Find $a$ and $b$ s.t. $L(a, b) = \min L(c_i, c_j) \forall i, j$;
    Append $(a, b)$ to $H$;
    Let $a = a \cup b$;
end
return($H$);
```

**Algorithm 3:** Hierarchical Agglomerative Clustering

of the linkage function is to define the distance between two clusters (sets of data points) given an existing distance function for individual data points. In practice, there are many possibilities. The two simplest are complete-linkage and single-linkage. Complete-linkage defines $L(c_i, c_j) = \max d(x_i, x_j) \forall x_i \in c_i, x_j \in c_j$; single-linkage defines $L(c_i, c_j) = \min d(x_i, x_j) \forall x_i \in c_i, x_j \in c_j$. For the metrics and data sets discussed herein, there has been no observed empirical difference between the two, and so the simpler single-linkage will be used.

Similar to HAC, spectral clustering depends only on a similarity matrix. The spectrum of the similarity matrix is used for dimensionality reduction. There are a variety of techniques for the dimensionality reduction. The basic outline of the approach [43] is shown in Algorithm 4:

Both HAC and spectral clustering allow for a variety of distance metrics. In practice, target clustering requires adjusting or penalizing the distance metrics to incorporate $\Lambda$ information. However, as a proof of concept, consider the edit distance.

```
**Data**: Similarity matrix $S$, $k$ clusters
**Result**: Clusters $C_1, \ldots, C_k$
Let $L$ be the Laplacian of $S$;
Let $e_1, \ldots, e_k$ be the first $k$ eigenvectors of $L$;
Let $E$ be the matrix with $e_1, \ldots, e_k$ as columns;
Let $y_i$ be the $i$-th row of $E$;
Cluster the points $y_i$ into the $k$ clusters $C$;
return($C$);
```

**Algorithm 4:** Spectral Clustering

## 8.2 Edit Distance

The edit distance, also known as the Levenshtein distance, is simply the number of edits (inserts, substitutions, and deletes) required to transform one sequence into another.[37] For example, **A**aa and **A**a have an edit distance of 1, while **A**aa and **AAA** have an edit distance of 2. Critically, unlike, for example, using the Euclidean distance on a bag-of-words[38] representation, the edit distance preserves order information. This allows HAC and spectral clustering with edit distance to serve as a check on earlier assumptions, namely whether target clustering is strictly necessary for these data sets or whether preserving sequence information is sufficient.

Figure 37 shows an example dendrogram generated by single-linkage HAC on the Geometry02 data set. The dendrogram is highly unbalanced, which suggests that the data may be weighted more heavily towards one type. In the case of Geometry02 data, most sequences are short and consists primarily of either short attempts

[37]The Levenshtein distance is technically only one type of edit distance, but the two terms will be use interchangeably herein.

[38]See Section 3.6, page 36, for an example.

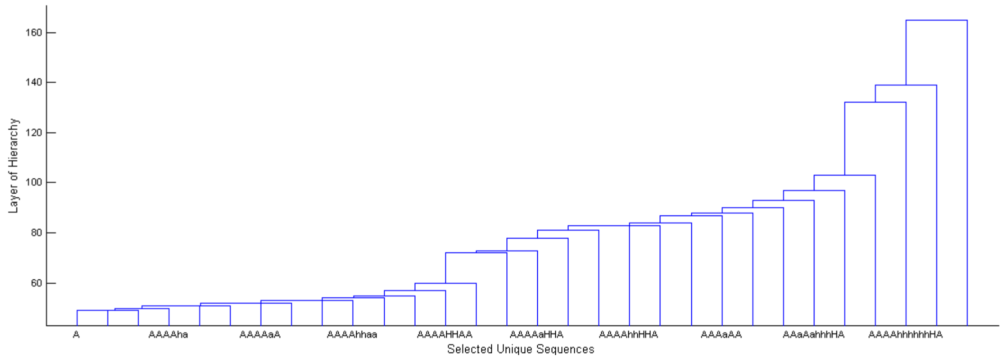


Figure 37: Dendrogram for edit distance

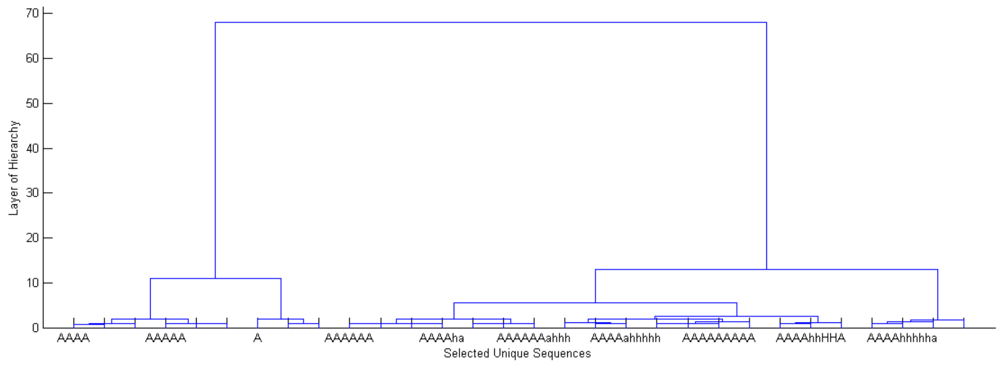


Figure 38: Dendrogram for normalized edit distance

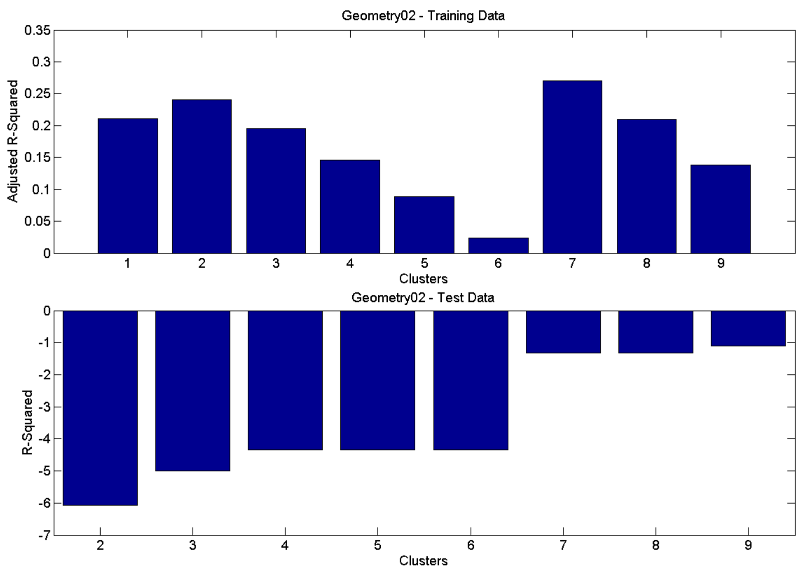


Figure 39: Correlations by clusters for Geometry02-StepA8H8

or long attempts, which creates the imbalance. Figure 38 uses the normalized edit distance, which balances the clusters by dividing the edit distance for a pair of sequences by the length of the longer sequence.

However, despite the balanced clustering, HAC still produces poor results on test data. Figure 39 shows the training and test-set correlations for HAC (with normalized edit distance) applied to Geometry02-StepA8H8. The top graph shows the training-set $R^2$ fits while the bottom graph shows the corresponding test-set $R^2$ fits; along the x-axis are results for different numbers of clusters. The key point is that the test-set results are universally less than zero.

Still, the edit distance is a fairly simple distance metric. In particular, many more complicated distance metrics and sequence kernels have been developed for gene expression data. It is likely that other distance metrics that incorporate sequence information will be as useful as, or perhaps better than, the edit distance.

However, the focus here is on target clustering. When using the metric learning paradigm, Target clustering algorithms assume that there exists a metric to modify, e.g., edit-distance. They then learn a new metric-space derived from that original metric. While the examples will use the edit-distance as the starting metric, it is not strictly necessary. However, the edit-distance has the convenient property of being an L1-norm over its operations (to be detailed in the next section), and many of these methods will use linear operators that result in straight-forward interpretations for the edit-distance.

## 8.3 Reweighted Distances

As mentioned before, the edit distance between two sequences $x_i$ and $x_j$ can be reformulated as the L1-norm of a vector $v_{i,j}$, where $v_{ij} = f(x_i, x_j)$ and where $v_{ij}$ is a vector in the space of possible operations. For example, if the space of operations is defined over $\{$insert, substitute, and delete$\}$, then $v \in \mathbb{N}^3$. This space can be expanded to encompass operations over different symbols $\sigma$ in an alphabet $\Sigma$, e.g., insert($\sigma_a$) and substitute($\sigma_a$,$\sigma_b$). For our canonical four operations (a, h, **A**, and **H**), an appropriate vector $v_{i,j}$ would be:

$$\begin{pmatrix} inserts(\sigma_a) \\ substitutions(\sigma_a, \sigma_h) \\ substitutions(\sigma_a, \sigma_A) \\ substitutions(\sigma_a, \sigma_H) \\ inserts(\sigma_h) \\ substitutions(\sigma_h, \sigma_a) \\ substitutions(\sigma_h, \sigma_A) \\ substitutions(\sigma_h, \sigma_H) \\ inserts(\sigma_A) \\ substitutions(\sigma_A, \sigma_a) \\ substitutions(\sigma_A, \sigma_h) \\ substitutions(\sigma_A, \sigma_H) \\ inserts(\sigma_H) \\ substitutions(\sigma_H, \sigma_a) \\ substitutions(\sigma_H, \sigma_h) \\ substitutions(\sigma_H, \sigma_A) \end{pmatrix}$$

Let there be $p$ such operations. The usual edit distance metric gives each type of operation $o$ an equal unit weight, e.g., two substitutions($\sigma_a$,$\sigma_A$) carries the same weight as two inserts($\sigma_H$). However, if, instead, each operation $o$ is allowed to have a separate weight $\beta_o$, it becomes possible to learn weights over the edit-distance. Learning these weights is similar to the learning task required by most metric learning algorithms, except that most of those algorithms are designed to operate in vector-space while this method is designed to operate with only a distance metric.

In practice, to learn a vector of weights, $\beta$, it is sufficient to solve the linear regression:

$$Y = \alpha + \beta V + \epsilon$$

Here, $Y$ represents target information (to be defined in a moment), $\alpha$ is a constant $p \times 1$ vector, $\beta$ is a $p \times 1$ vector of weights to learn, $\epsilon$ is noise, and $V$ is a matrix of $v_{i,j}$ vectors, as mentioned before. The dimensions of $V$ require some exposition. Let there be $n$ observed sequences in the data set. Of these $n$ sequences, there are actually only $m < n$ distinct sequences. For example, one student may perform **A**aa on one problem and another may perform **A**aa on a different problem, both with slightly different timing information, but as far as the algorithm is concerned, they are both instantiations of the same unique sequence. Thus, $V$ will be an $m(m-1) \times p$ matrix as there are $m(m-1)$ pairs of distinct sequences and, for each such pair, there is a distance that can be defined as a vector $v_{i,j}$.

The remaining question is, what can be used for $Y$? $Y$ should be useful as a target for the algorithm, and should thus contain learning information, but the learning information must be aggregated across instances of all instances of a unique sequence. Underlying any choice of $Y$ is the theoretical assumptions of the method. One assumption required by the following algorithms is that every distinct se-

Table 5: Hypothetical example: the distribution of two sequences across students

| **Aaa** | **Aaah** |
|---|---|
| Student 1 | Student 3 |
| Student 1 | Student 4 |
| Student 2 | Student 4 |
| Student 2 | Student 5 |

Table 6: Hypothetical example: learning gains as noisy labels

| **Aaa** | **Aaah** |
|---|---|
| 0.8 | -0.2 |
| 0.8 | 0.3 |
| 0.6 | 0.3 |
| 0.6 | -0.4 |

quence is unique and any differences arising between observations involving one student and observations involving another student are simply noise. For example, if **A**aa occurs for a student exhibiting a $0.8$ learning gain, but also for a student exhibiting $0.6$ learning gain, both observations represent some underlying truth as to the usefulness of this action sequence, plus some noise. Thus, two simple choices for $Y$ are the differences in the mean and median of learning gains associated with each distinct sequence. Concretely, assume there are two distinct sequences **A**aa and **A**aah, with the observed sequences shown in Table 5. Then the learning gains for each student can be treated as noisy observations of the underlying usefulness of each distinct sequence, resulting in Table 6.

Based on this distribution of learning gains, it would appear that the first sequence (**A**aa) is a more useful sequence than the second sequence (**A**aah). Using the mean as an aggregation function makes the first sequence have a corresponding value of $0.7$ and the second sequence have a corresponding value of $0$. Thus, $Y_{\mathbf{A}aa,\mathbf{A}aah} = Y_{\mathbf{A}aah,\mathbf{A}aa} = 0.7$. Through this method, $Y$ aggregates across all instances of a distinct sequence while preserving the learning gain information in the form of noisy estimates of the usefulness of each sequence.

On the matter of interpretation, solving the regression formula means more weight is applied to sequence pairs generated by more distinct types of students, and thus, the formula weights an operation more heavily if its presence is predictive of a difference in student learning. Before considering the algorithm in detail, there are two important caveats to note. First, it is possible for $\beta$ coefficients to be negative. Imagine that insert($\sigma_a$) has a negative $\beta$ coefficient. Then, the sequences **A** and **A**aaaaaa would have a negative distance. Negative distances are especially problematic because the distance between **A** and **A** would still be zero and thus greater than a negative distance! One potential solution is to take the absolute value of the distance. This solution is counterintuitive: after all, a high negative weight $\beta$ suggests that the edit-operation actually improves the odds of two sequences being from similar types of students, whereas the absolute value would suggest the converse; however, the solution works well in practice and beat other options in a series of trials. The absolute value is probably effective because most negative $\beta$ have small absolute values.

Call this algorithm Linear Distance Reweighting (LDR), as shown in Algorithm 5. One useful property of LDR is that there is a clear and easy interpretation for $\beta$

```
**Data**: Sequences $X$, Gains $G$
**Result**: Clusters $C$
Compute $V_{i,j}$ for each pair of distinct sequences ($X_i$, $X_j$);
Compute $Y_{i,j}$ for each pair of distinct sequences ($X_i$, $X_j$);
Solve $Y = \alpha + \beta V + \epsilon$;
Compute $D$ where $D_{i,j}$ is the distance between distinct sequences ($X_i$,$X_j$);
Calculate clusters $C$ from $D$ using any metric clustering algorithm;
return($C$);
```

**Algorithm 5:** Linear Distance Reweighting

coefficients. A high, positive $\beta$ coefficient implies that the associated action tends to indicate a dramatic shift in the interpretation of a sequence. For example, **H** often receives a high $\beta$ coefficient, which implies that observing an additional quick hint request significantly changes the interpretation of a sequence. This fits with observed behaviors and results (in these data sets): asking for a hint is probably a worse option than not asking for a hint (all else being equal), and since there are only a few actual levels of hints in most systems, each hint request carries great weight.

Vice versa, a low or negative $\beta$ coefficient implies an action has little to no impact on the interpretation of a sequence. For example, quick attempts often receive a low $\beta$ coefficient. This fits with observed behavior: while guessing repeatedly appears to be a poor behavior in these data sets, guessing one more time if the student already guessed before is unlikely to result in any dramatic difference in interpretation.

There are many choices for the set of possible weights. For example, a simple case would be to have one weight for all substitutions and one weight for all insertions and deletions. This option, however, would not allow for the weights to adapt to the importance of various action types. An extreme option in the direction of more complexity would be to allow a weight for each possible insertion, deletion, and substitution, such as is shown above. This corresponds to the value of $V$ shown earlier.

There are several other important choices when it comes to implementing Linear Distance Reweighting. First, as mentioned before, there is the question of the aggregation function. Learning gains can be aggregated with the mean or median or any other aggregate statistic, but, in addition, these learning gains can also be weighted by a number of measures: number of times the sequence is observed, probability of the sequence, etc. For example, if a student uses a sequence $i$ eight times out of 50 total sequences, when computing $Y_i$, the student's learning gain can either contribute eight times, or contribute 0.16 times. Second, there is the related question of normalization for $V$. Visually, each operation contributes to the edit distance just once, independent of sequence length. However, as shown before, clusters tend to be more balanced when distances are normalized; for Linear Distance Reweighting, normalizing by sequence length corresponds to a row-normalization of $V$.

### 8.3.1 Empirical Results

Tables 7 and 8 show the test-set performance of Linear Distance Reweighting applied to Geometry02-StepMean and Geometry02-StepA8H8. Performance on

Table 7: LDR validation / test results on Geometry02-StepMean

| **Clusters** | **Validation-Set Adjusted-$R^2$** | **Test-Set $R^2$** |
|---|---|---|
| 2 | 0.17 | 0.18 |
| 3 | 0.27 | 0.26 |
| 4 | 0.20 | 0.18 |
| 5 | 0.28 | 0.40 |

Table 8: LDR validation / test results on Geometry02-StepA8H8

| **Clusters** | **Validation-Set Adjusted-$R^2$** | **Test-Set $R^2$** |
|---|---|---|
| 2 | 0.29 | -0.55 |
| 3 | 0.27 | -0.18 |
| 4 | 0.20 | -82.47 |
| 5 | 0.17 | -55.20 |

the former is decent (0.18) to fantastic (0.40). However, while Linear Distance Reweighting only shows slightly poorer validation-set performance on Geometry02-StepA8H8, it shows abysmal test-set performance.

The two tables (7 and 8) show the two main problems with Linear Distance Reweighting: inconsistency and a lack of a reliable predictor of test-set performance. In the first case, note how dramatically test-set results vary given a simple change to the duration thresholds; this dramatic shift is not observed in the Guidepost HMM algorithms. Further, there's no clear relationship between validation-set performance and test-set performance, making it impossible to distinguish the degenerate cases, e.g., Geometry02-A8H8, from the cases in which LDR performs well. However, when LDR performs well, it performs extremely well: a 0.4 $R^2$ on test data is solid, and on some data sets, LDR produces training-adjusted-$R^2$

values and test-$R^2$ values of nearly 0.5.

In the end, LDR offers a number of advantages:

- Convergence: LDR is built around a linear regression and a concave metric learning algorithm (in this case, hierarchical agglomerative clustering), and so is guaranteed to converge to the same solution given the same data set.[39]

- Scalability: The dominant computational cost in solving LDR is computing the distances between distinct sequences, which requires $O(m^2)$ time. For large data sets, this makes LDR more efficient than Guidepost HMM algorithms since Guideposting scales in a polynomial of the total number of sequences $n$ (not the number of distinct sequences $m$). Many sequences, such as **A**a, are repeated many times in any given data set, meaning $m \ll n$.

- Interpretation: The $\beta$ coefficients computed during the regression stage are easily interpreted. Large $\beta$ values suggest that adding an action of a certain type is significantly predictive of a change in outcomes. $\beta$ values can thus be used to construct interventions (e.g., intervene if a good student is engaging in actions with large $\beta$ values).

However, LDR, as presently formulated, has a number of significant disadvantages as well:

- Consistency: Discussed above, LDR does not offer consistent performance

[39]Whether the solution is in some sense optimal is not clear.

even on data sets (e.g., Geometry02-StepMean) for which Guidepost HMM algorithms consistently perform well.

- Evaluation: Possibly the worst problem, high validation-set performance for LDR does not necessarily predict high test-set performance. This deficiency is possibly more about the size of the data set, and thus the size of the validation and test sets, than about anything else.

- Interpretation: While the $\beta$ coefficients provide a powerful tool for interpreting LDR models, the models are otherwise difficult to interpret. Collections of HMMs, while not simple to interpret, do contain structural information that can help provide domain-oriented interpretations for actual clusters. For LDR, excepting the example sequences in each cluster, there is no such structural information available.

While LDR and metric learning versions of target clustering algorithms are promising in the long-run, they are not yet capable of consistently high-performance on real data. Thus, when exploring additional data sets in the next section, I will restrict the analyses to Guidepost algorithms. The following section addresses the real-world applicability of these algorithms, including the unfortunate reality of data pre-processing and the detection of degenerate cases.

# 9 Results

Empirically, there are three data sets used in this thesis:

- **Geometry 2002** (Geometry02) — There were 21 students in the control condition of this study. They used a relatively early version of the Geometry Cognitive Tutor without worked examples or any controls on hint use [1].
- **Geometry 2006** (Geometry06) — There were 16 students in the control condition of this study. The version of the Geometry Cognitive Tutor used in the study restricted hint requests to at most one per two seconds [48].
- **Stoichiometry 2006** (Stoichiometry06) — There were 81 students in this study. This tutor introduces math required to solve elementary chemistry problems [40].

Most of the results above have focused on Geometry02. So far, Geometry02 has shown that the choice of threshold and granularity matter in determining the performance of various target clustering algorithms. In general, the most promising option has been Guidepost Weighted HMMs, followed by Reweighted Edit Distance.

Each of these data sets offers something unique: Geometry06 is similar to Geometry02, but also has marked differences in population and some significant differences in the tutor interface, and could thus have similar observed behaviors or different observed behaviors; Stoichiometry06 is a different type of tutor in a distinct domain (mathematics for chemistry), providing a significantly differentiated data set.

## 9.1 Simulation Results

First, it's important to briefly review the results from simulated data sets to provide baseline measures for comparison. There have been three types of simulated data sets considered so far: Naive, Basic, and Random. The Naive model is relatively easy to learn with distinct, rarely-overlapping HMMs. The Basic model is based on models previously learned from empirical data sets; it generates noisier, more realistic data. The Random model refers to any set of randomly generated HMMs.

For the Naive model, Guidepost EM-HMM can find HMM collections, on average, with a test-set $R^2$ of $0.65$ and, at a minimum, with a test-set $R^2$ of $0.5$. For the Basic model, Guidepost EM-HMM can find HMM collections, on average, with a test-set $R^2$ of $0.6$. For the Random model, Guidepost EM-HMM can sometime find HMM collections with a positive test-set $R^2$; for more constrained random HMMs, the test-set $R^2$ rises as high as $0.25$. In general, the more constrained the original HMMs and the more distinctly separated their associated emissions, the easier it is for Guidepost EM-HMM to learn the collections.

In sum, the simulation results indicate that there exist collections of HMMs that can be recovered by Guidepost algorithms, that it is not the case that any random collection of HMMs will lead to discoverable models, and that therefore, if useful models are discovered, it is likely that the models describe some inherent structure in the data.

## 9.2 Geometry02

Aside from the simulation results, most of the results up until this point have been about Geometry02. The main results with respect to Geometry02 are the following:

- Guidepost EM-HMM can find models of student behavior that predict ($R^2 = 0.26$) pre-post learning scores.
- Guidepost EM-HMM works (for Geometry02) on both the step and problem level.
- Guidepost EM-HMM is largely robust to different threshold choices in Geometry02.
- Guidepost Weighted-HMMs reaches similar or better ($R^2 = 0.36$) results, but requires less computation time.
- Both algorithms usually produce multiple candidate collections with high performance, but these collections also tend to be similar.

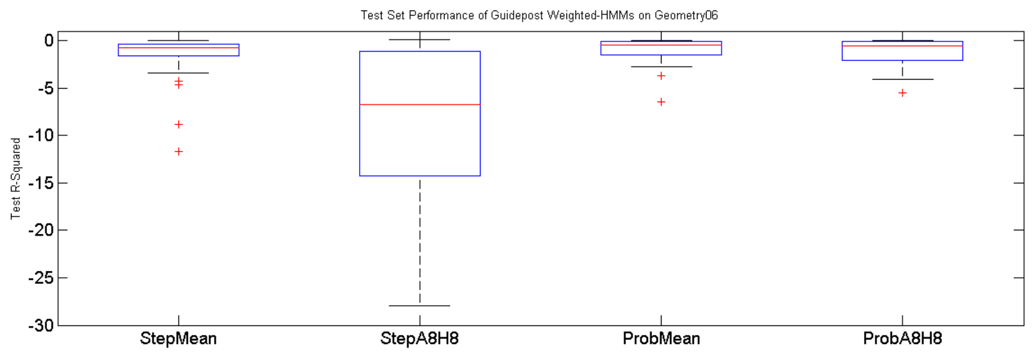


Figure 40: The test-set $R^2$ performance of Guidepost Weighted-HMMs on four different versions of Geometry06 data with overall poor results

## 9.3 Geometry06

The next most similar data set is Geometry06. The Geometry06 data set originates with a different version of the Geometry Cognitive Tutor, covering a different unit (circles instead of angles), and includes a different student population [48]. However, in principle, it is a relatively similar data set that provides a good test case for the empirical properties of Guidepost EM-HMM. Figure 40 shows the test-set performance of Guidepost Weighted-HMMs on Geometry06 data. With an almost consistently negative test-set $R^2$, these results indicate that Guidepost Weighted-HMM did not find any HMM collections that reasonably modeled student behaviors and learning.

There are a number of possibilities to explain the poor performance of Guidepost algorithms on this data set. First, the target data (the pre- and post-test scores) could be completely unrelated to student behavior, violating a core assumption of target clustering. Second, students could behave in ways that cannot be modeled by simple hidden Markov models with only a few states. Student behaviors could also require additional complexity beyond clusters of HMMs; for example, student behaviors on one step could be conditionally dependent on behaviors on previous steps, which cannot be represented by the present collections. Finally, the poor performance could simply be an inference problem to be solved by more computation, by better algorithm and model design, or by more data.

There is strong evidence favoring the first possibility, that the pre- and post-test scores are unassociated with observed behaviors. This evidence will be presented in Section 9.6.1 (page 127), after discussing another relevant data set.

## 9.4 Stoichiometry06

The performance of Guidepost algorithms on Stoichiometry06 data is, at first glance, poor. In fact, it looks as poor or worse than Geometry06. This can be seen in Figure 41. All of the same possible explanations apply, with the additional fact that the stoichiometry tutor and domain content are far removed from Geometry02's tutor and domain content.

These two cases are initially discouraging. However, to understand the root differences requires first understanding the actual behavior of Guidepost algorithms and two distinct forms of failure.

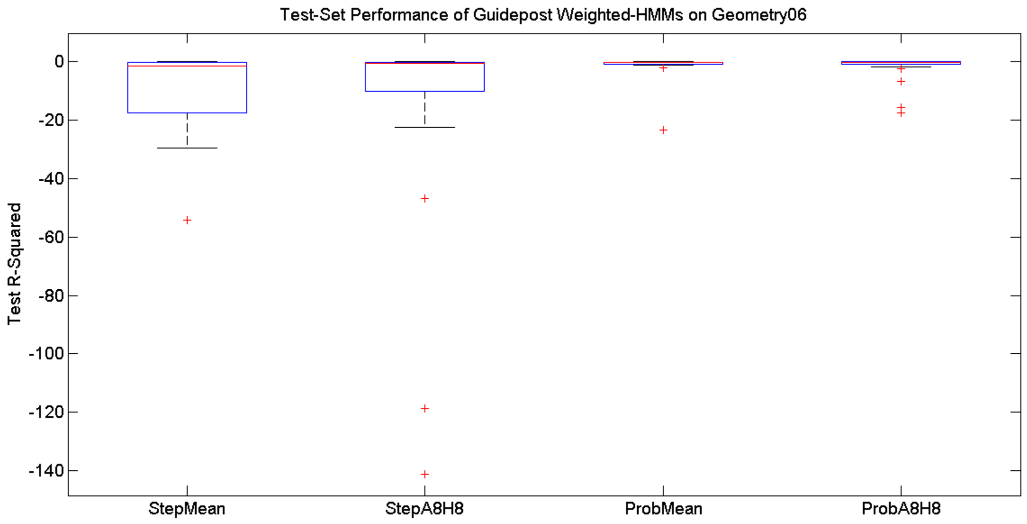


Figure 41: The test-set $R^2$ performance of Guidepost Weighted-HMMs on four different versions of Stoichiometry06 data with overall poor results

## 9.5 Types of Failure

One theory to explain the poor performance of target clustering algorithms on some of these data sets is that the educational assessments may be disconnected from actual student behaviors. This theory is easy to test with both empirical and simulated data sets. By simply randomizing the assignment of $\lambda$ values to students and then trying to learn from those values, it is possible to determine the extent to which poor target information can confuse or cripple a target clustering algorithm.

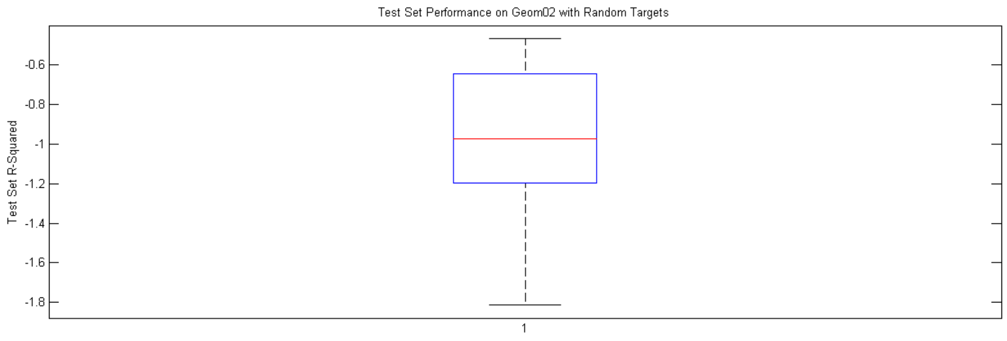


Figure 42: Test-set performance for Geometry02 with randomized targets

### 9.5.1 Geometry02

Since Geometry02 is a data set on which target clustering algorithms perform well, it is a good candidate for randomizing $\lambda$ values. Target information can be randomized by simply reassigning existing $\lambda$ values to random students. The resulting test-set performance is shown in Figure 42. Obviously, the prediction is extremely poor.

A key question is whether it might be possible to detect when the target $\lambda$'s hold little to no relevant information. One possibility lies in the quality of *training*-set fits. Figure 43 shows histograms of the training-set performance for collections of HMMs tested by Guidepost Weighted-HMM during its iterative search. In the case of random $\lambda$'s (top of figure), training-set performance is abysmal, with regressions usually failing to find a significant predictor. In cases of non-random $\lambda$'s (bottom of figure), training-set performance is sometimes poor, but sometimes not, reflecting a search over a set of reasonable candidates. This distinction offers

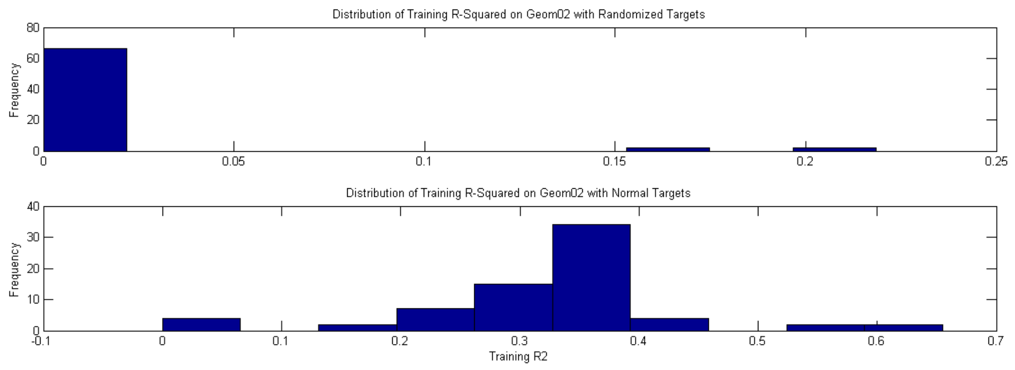


Figure 43: Histograms for Geometry02 with randomized (top) and non-randomized (bottom) targets

a possible heuristic for determining the potential for target clustering on a data set: if collections learned from training data are so poor as to have no significant predictors, target clustering is probably hopeless; however, if collections have a range of fits on training data, but test-set performance is poor, then it is probably worth looking at the underlying modeling assumptions of the algorithm and making adjustments as necessary.

### 9.5.2 Naive

Figure 44 shows the histogram of training-$R^2$ results for data from the Naive Model, both with the normal $\lambda$ values and randomized $\lambda$ values. The same judgments apply: the histogram of training-$R^2$ results is skewed towards 0 in the case of randomized targets, but is more evenly balanced when the $\lambda$ values have actual information. Using the Naive Model makes it possible to test an additional ques-

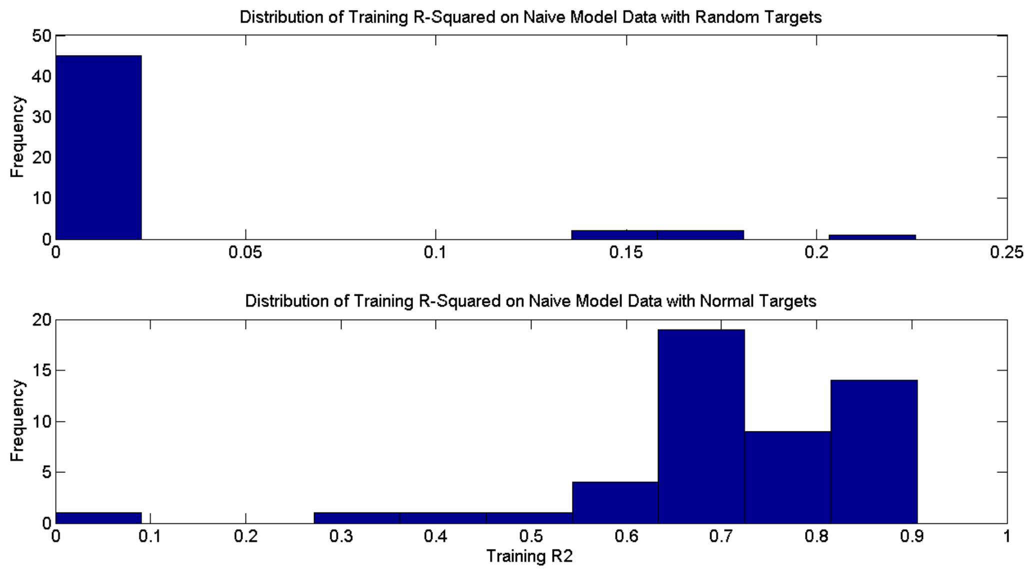


Figure 44: Histograms for the Naive Model with normal and randomized targets

tion: what happens if the fit is poor not because of bad $\lambda$ values, but because of a poor choice of threshold?

One way to simulate a poor choice of threshold is to begin with data generated by the Naive Model and replace half of the **A**'s with a's, randomly. This simulates the case where the threshold for attempts is set too high, resulting in even long attempts being classified as quick ones. The test-set performance is unsurprisingly poor, but more importantly, the training-set performance exhibits patterns more reminiscent of normal targets than randomized targets, as shown in Figure 45. This fact provides a useful heuristic for determining whether the target information available in a data set is inherently poor, or whether the problem lies elsewhere in the preprocessing or modeling assumptions.

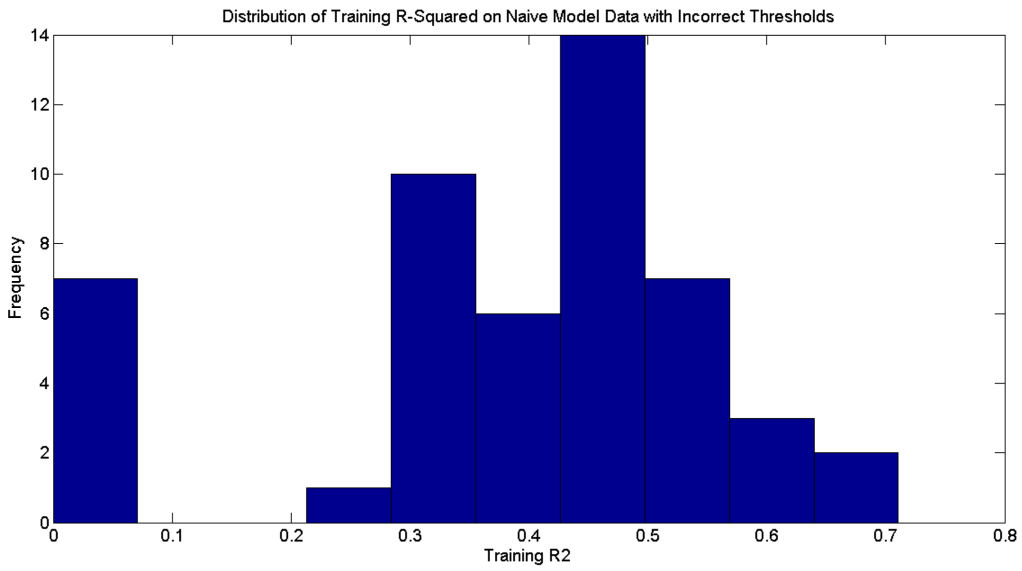


Figure 45: Histograms for the Naive Model with incorrect thresholds

## 9.6 Preprocessing

The last piece of the empirical puzzle relates to the distribution of sequence lengths. Figure 46 shows the distribution of sequence lengths for a number of data sets. The most important differentiation is that, while most sequences are of length 1 (a or **A**) even in Geometry02, there are also many sequences of lengths 2 and 3 (and a long tail of longer sequences). In Geometry06, the tail of longer sequences still exists, but there are significantly fewer sequences of length 2 and 3. This makes it harder for an algorithm to find expressive HMMs since a and **A** dominate. In Geometry06, in particular, sequences of length 1 make up over $75\%$ of the data.

One potential solution is to remove sequences of length 1 from the data set en-

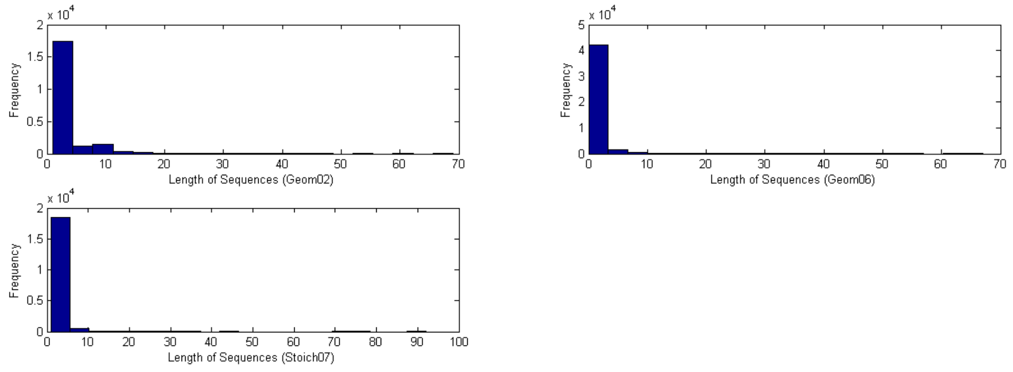


Figure 46: Comparison of the distribution of sequence lengths across datasets

tirely. The vast majority of them are single attempts that lead to correct answers. Since the goal is to find models that predict learning, and not necessarily those that predict knowing, these actions are potentially of limited usefulness. However, their presence results in a biasing of HMM parameters.

### 9.6.1 Geometry06

Figure 47 shows the performance of Guidepost Weighted-HMMs on Geometry06 after removing sequences of length 1. Clearly, no collections performed above chance on test data, as is evident from the y-axis ($R^2$) starting at zero and going down from there. This may, initially, seem to suggest that removing specific sequences as a form of pre-processing is not useful on real data. However, consider Figure 48 which shows the training-set performance of the same collections shown in Figure 47. Most collections found by Guidepost Weighted-HMMs on Geometry06 data have no significant predictors of student learning gain at all,

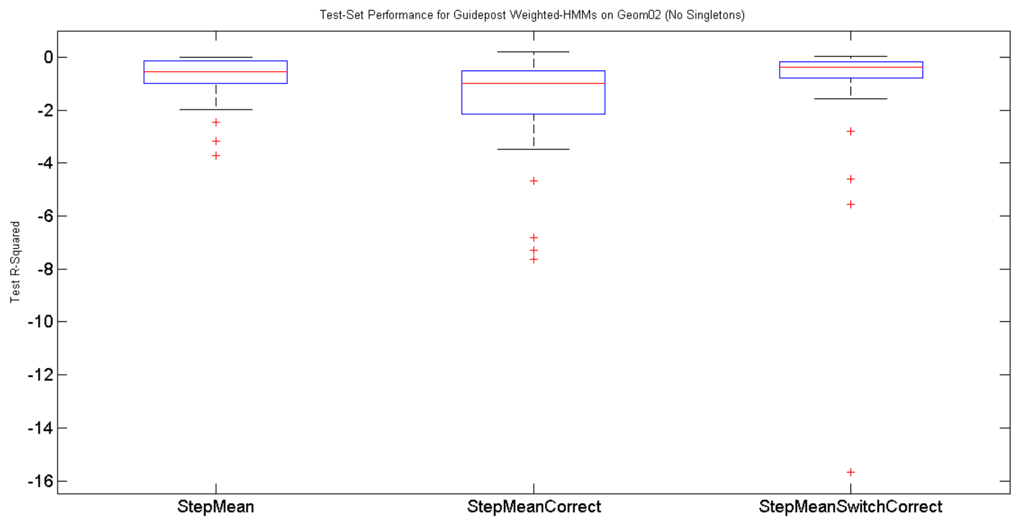


Figure 47: Training and test performance for Geometry06 without length-1 sequences

even on training data. If we accept the empirical result from the previous section, this suggests that the learning gain is not strongly associated with relevant behaviors, much as in the simulations presented before.

### 9.6.2 Stoichiometry06

Returning to the stochiometry data that motivated this digression, it initially appears to exhibit similar patterns to Geometry06. First, the histogram of step lengths is similar, as shown back in Figure 43, with step lengths highly concentrated towards 1. Second, the test-set $R^2$ for discovered collections are poor, as shown back in Figure 41. However, the difference lies in the distribution of training-set $R^2$ results. As shown in Figure 49, Guidepost Weighted-HMMs finds

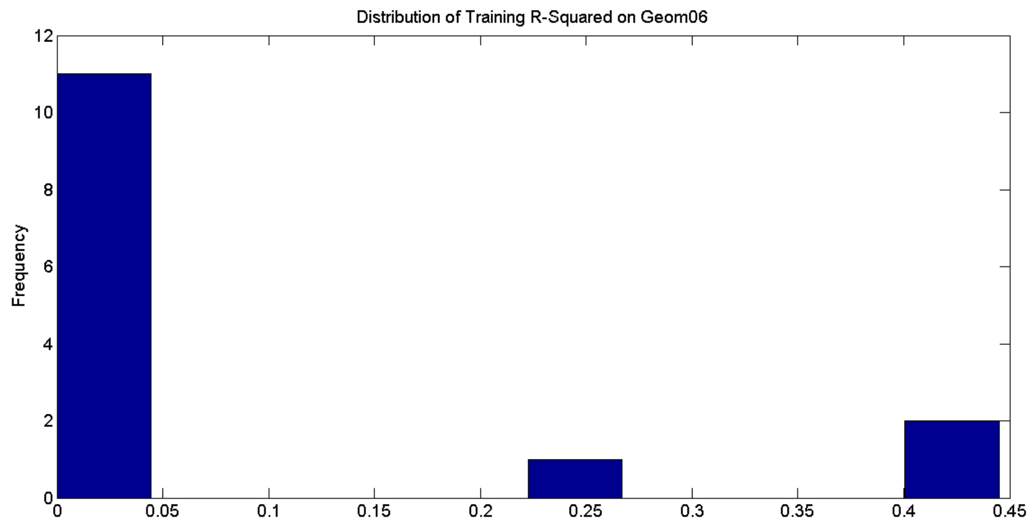


Figure 48: Training-set performance for Geometry06 without length-1 sequences

collections with reasonable (non-zero) training-set performance most of the time. As shown before, a wide distribution of positive training-set $R^2$ fits suggests that there is useful information in the data set, but that the algorithm is having difficulty learning effective collections.

If sequences of length 1 are removed from the data set, the results of Guidepost Weighted-HMMs are much improved. Figure 50 shows the test-set performance of Guidepost Weighted-HMMs on a particular variant on the stoichiometry data, Stoichiometry06-StepMeanSwitchCorrect. For reasons to be explored later, for this data set, this particular set of emission symbols results in better predictions of learning than with the basic set. If the same tweaks (StepMeanSwitchCorrect and removing length-1 sequences) are performed on Geometry02 data, the predictive accuracy is worse ($R^2 = 0.22$, compared to $R^2$ for Geometry02-StepMean in the

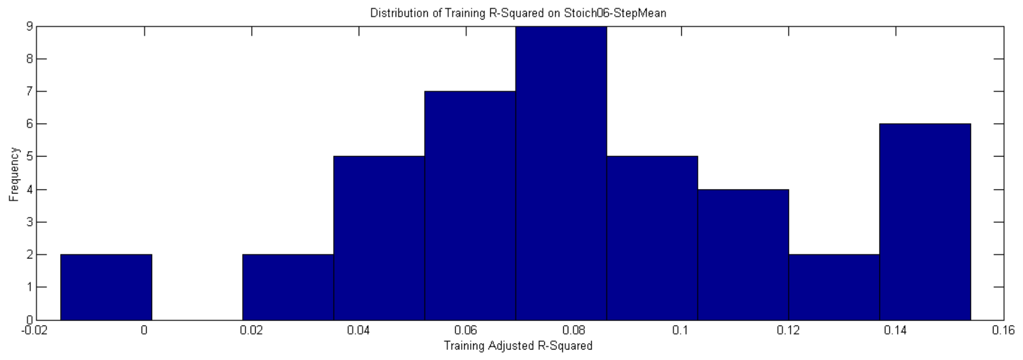


Figure 49: Histogram of training-set $R^2$ for Stoichiometry06-StepMean

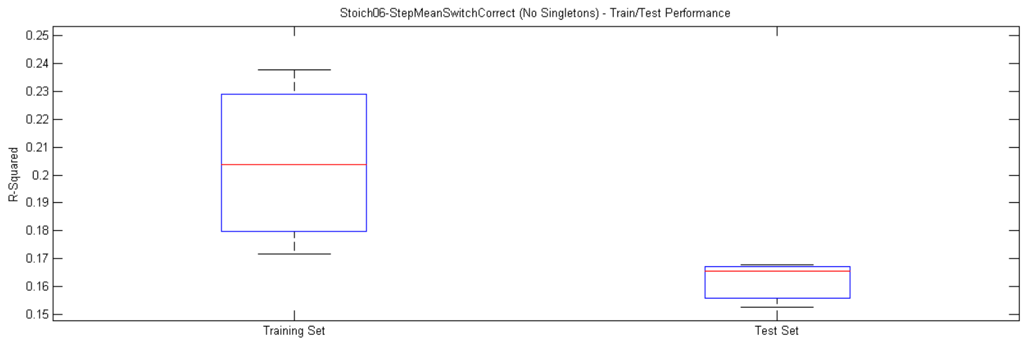


Figure 50: Training- and test-set performance without length-1 sequences for Stoichiometry06-StepMeanSwitchCorrect

$0.3$ range) while there is little interesting change in actual models (as discussed later).

### 9.6.3 Additional Preprocessing Options

There are other applicable techniques for preprocessing data for other problem paradigms that can be useful in target clustering. For example, one of the primary approaches to solving noisy label problems is to pre-process the data. This usually involves either cleaning (removing) or correcting examples that are believed to be noisy. A similar idea can be applied to target clustering problems. For example, in education, getting the first attempt correct on a step of a problem suggests you either knew or learned the skill; this behavior is generally positively correlated with learning. This means starkly positive sequences, like first-attempt-correct, could be detected with straightforward correlations and removed from poor learners; similarly, starkly negative sequences could be detected and removed from good learners. All this could be performed without any inherent modeling assumptions for the $f$ function.

It is also possible, and possibly recommended, to provide a better initialization for Guidepost algorithms. Right now, initial HMMs are determined randomly. Instead, it's possible to initialize HMMs using target $\lambda$'s. For example, the first collection of HMMs could consist of two HMMs, one trained with all sequences drawn from students with positive learning, and the second HMM trained with all sequences drawn from students with negative or poor learning. Further, during each iteration, when an HMM is removed from the collection, it could be replaced with two HMMs, like above, except drawing only from sequences in the departing HMM's partition.

## 9.7 Interpretation

For Geometry02 and Stoichiometry06, there exist collections discovered by Guidepost Weighted-HMMs that perform well on both training and test data. However, to be useful for more than simple prediction of student learning, the HMMs in these collections need to be interpretable by domain experts.

### 9.7.1 Geometry02

Geometry02 yields stable, discoverable HMMs under a variety of thresholds and granularities. As a result, it's important to consider the commonalities between individual collections of HMMs discovered by target clustering on Geometry02. For the following examples, there are five emission symbols.[40] There are four main "types" of HMMs that usually show up in step-level collections learned from Geometry02. These are:

- A "Dominant Model" that includes correct-on-first-try sequences as well as longer sequences of attempts. This model is usually positively associated

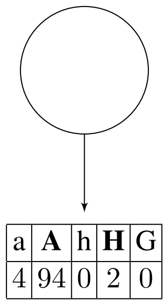
[40] a is a quick attempt, **A** is a long attempt, h is a quick hint, **H** is a long hint, and G is a request to access the tutor's glossary.

with learning. This model is "Dominant" because it contains the most predictive sequence, **A**. That sequence, which indicates a single correct attempt and, thus, that the student probably knows the skill(s) required by the step, is a frequently occurring and important sequence. In addition, the Dominant model usually contains longer sequences of repeated attempts. These longer sequences usually include all sequences of attempts not contained in the more specific "Guessing Model".

- A "Guessing Model", which usually generates a single **A** followed by a string of a's, thus representing an important type of sequence not assigned to the Dominant Model. This model is usually negatively associated with learning. It is called "Guessing" here even though it is not known that the student is actually guessing; however, repeated fast attempts suggests that, at the very least, the student is not thinking his steps through.

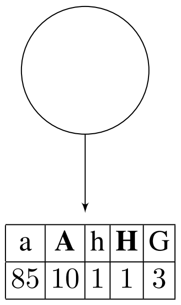
- A "Hints Model", which is usually negatively associated with learning and contains many types of hint behaviors. This model is sometimes merged with a "Miscellaneous Model".

- A "Miscellaneous Model" that simply absorbs all the other sequences not covered above. Since the "Miscellaneous" model usually has little to no correlation to learning, the sequences that are assigned to it are essentially treated as noise.



| a | **A** | h | **H** | G |
|---|---|---|---|---|
| 4 | 94 | 0 | 2 | 0 |

Figure 51: Dominant Model



| a | **A** | h | **H** | G |
|---|---|---|---|---|
| 85 | 10 | 1 | 1 | 3 |

Figure 52: Guessing Model

The last two models, the Hints model and Miscellaneous Model, are often merged in discovered collections. The result is usually a Miscellaneous Model centered around hint-related behaviors.

As an illustration, consider the simple 3-HMM collection shown in Figures 51, 52, and 53. This collection was originally discovered by Guidepost Weighted-HMMs, and while it is not the most powerful collection (test-$R^2$: $0.25$), it is easy to understand.

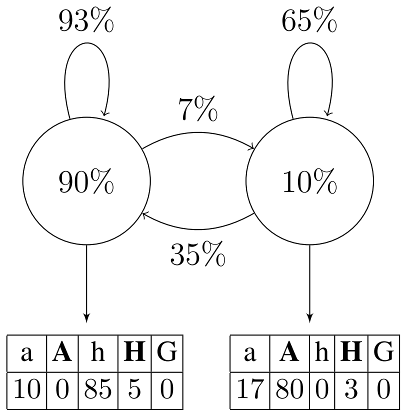


Figure 53: Hints and Miscellaneous

However, examining the structure of individual HMMs gives an incomplete picture of the collection as a whole. After all, a high probability sequence for one HMM can also be a high priority sequence for another HMM, and it is not trivial to determine, at a glance, to which HMM such a sequence would be assigned. Instead, the structure of each HMM can be augmented by information about its partition of the training data to better explain the reasons for its structure and its functionality within the collection. For example, consider the following tables, which show the frequency of the five most common sequences in each HMM's partition of the training data:

- Simple 3-HMM Collection — "Dominant Model"

  Regression Coefficient: $0.66$, Figure 51

  77 unique sequences, 4763 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **A** | **4042** |
| **AA** | **343** |
| **AAA** | **72** |
| **AaA** | **44** |
| **aAA** | **39** |

- Simple 3-HMM Collection — “Guessing Model”

  Regression Coefficient: $-0.66$, Figure 52

  124 unique sequences, 6156 total sequences

  | **Sequence** | **Frequency** |
  |---|---|
  | **a** | **5062** |
  | **Aa** | **259** |
  | **aa** | **257** |
  | **aA** | **220** |
  | **aaa** | **63** |

- Simple 3-HMM Collection — “Hints and Miscellaneous”

  Regression Coefficient: $-1.8$, Figure 53

  935 unique sequences, 2525 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **hhhhhHA** | **117** |
| **hhhhhha** | **97** |
| **hhhhhhha** | **75** |
| **hhhhha** | **65** |
| **hhhHA** | **59** |

Other example collections are available in Appendix 1. These core models, particularly the Dominant Model and the Guessing Model, show up again and again in subtly distinct guises. Sometimes there are more states in a model, and sometimes fewer, but the basic structure and partitioning of the data appear to recur amongst most collections with a high correlation to learning. Other models sometimes appear as well, including ones that suggest good hint use (e.g., hhh**HA**) or poor types of long repeated attempts. However, these models do not appear consistently enough to be considered stable under this particular analysis.

Examining these collections and the commonly-occurring models leads to the following tentative hypotheses:

- Guessing repeatedly is bad for learning.
- Excessive hint use (at least in some forms) is bad for learning.
- Getting the answer correct on the first try is a good indication of mastery.
- Slowly attempting a problem more than once (but not guessing) is a good

predictor of student learning.

### 9.7.2 Stoichiometry06

As discussed before, the stoichiometry data set does not, initially, yield highly predictive collections of HMMs.[41] However, after some tweaking (removing length-1 sequences and adding correctness and switching symbols[42]), Stoichiometry06 yields collections with a test-set $R^2$ as high as $0.2$. To understand why the data set demands these manipulations requires examining the HMMs themselves.

Consider, first, the following two models from one particular Stoichiometry06-StepMeanSwitchCorrect collection:

[41]Discussed in Section 9.4, page 121.

[42]The alphabet for these HMMs contains: i for quick incorrects, **I** for long incorrects, c for quick corrects, **C** for quick incorrects, h for quick hints, **H** for long hints, and **X** for the end of a step.

- A "Repeated Attempts Model" that includes instances of long sequences of long attempts. This model is similar to the "Dominant Model" from Geometry02, except that it does not include the single-correct sequences that have been preprocessed out of the data set.

  Regression Coefficient: $0.13$, Figure 54

  80 unique sequences, 409 total sequences

  | **Sequence** | **Frequency** |
  |---|---|
  | **IcX** | **112** |
  | **IIX** | **65** |
  | **IicX** | **29** |
  | **IiX** | **8** |
  | **IIccX** | **8** |

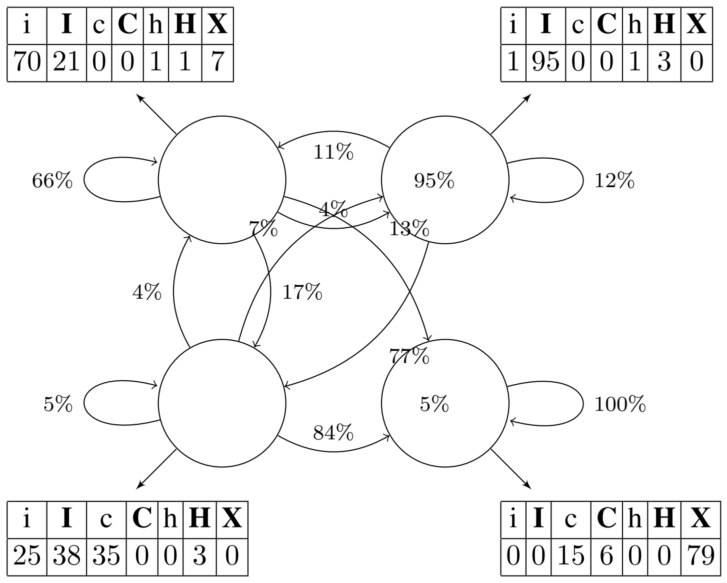
- A "Guessing Model" which is almost identical to the equivalent model in Geometry02.

  Regression Coefficient: $-0.27$, Figure 55

  77 unique sequences, 384 total sequences

  | **Sequence** | **Frequency** |
  |---|---|
  | **iiX** | **121** |
  | **iCX** | **80** |
  | **iIX** | **18** |
  | **iiiiX** | **16** |
  | **iiiX** | **14** |

In addition to these two models, there are also two "Trash" models, as shown in



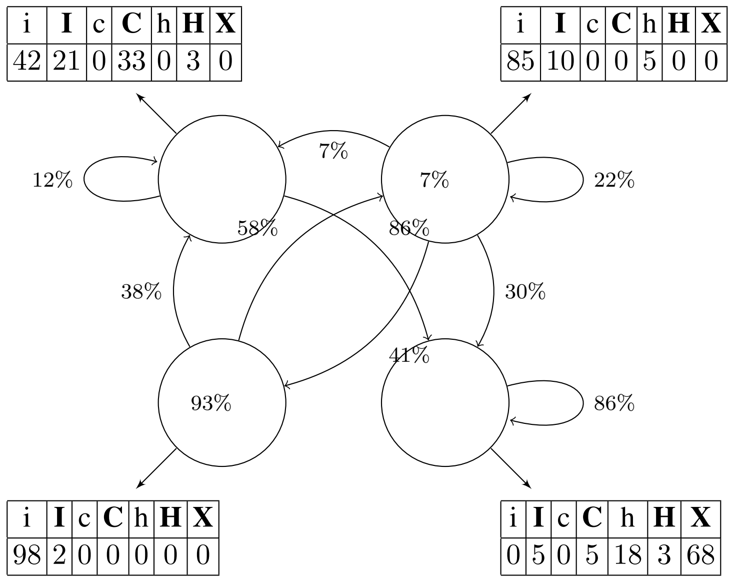
| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 70 | 21 | 0 | 0 | 1 | 1 | 7 |

| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 1 | 95 | 0 | 0 | 1 | 3 | 0 |

| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 25 | 38 | 35 | 0 | 0 | 3 | 0 |

| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 0 | 0 | 15 | 6 | 0 | 0 | 79 |

Figure 54: Quick Click Collection — “Repeated Attempts Model”



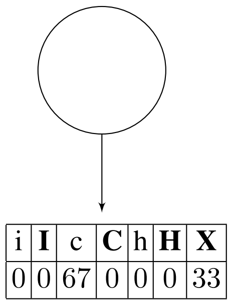
| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 42 | 21 | 0 | 33 | 0 | 3 | 0 |

| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 85 | 10 | 0 | 0 | 5 | 0 | 0 |

| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 98 | 2 | 0 | 0 | 0 | 0 | 0 |

| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 0 | 5 | 0 | 5 | 18 | 3 | 68 |

Figure 55: Quick Click Collection — “Guessing Model”



| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 0 | 0 | 67 | 0 | 0 | 0 | 33 |

Figure 56: Quick Click Collection — “Two Quick Corrects”

Appendix 2. What is interesting is the fifth model (Figure 56), which is negatively associated with learning ($\beta = -0.7$) and contains only one sequence, cc**X**. This sequence mostly happens with combo boxes, an interface widget not found in the Geometry data sets, and clearly indicates an unnecessary double-submission of a correct answer. This suggests an interesting possible interpretation for this highly predictive model: that it indicates students that are not fully attentive to their activites in the tutoring system. This would also explain why removing length-1 sequences, such as c**X** and **CX**, was necessary, as Guidepost Weighted-HMMs would be unlikely to generate HMM collections that could distinguish those short sequences from the particular, and precise, sequence cc**X**. In addition, without the correctness symbols and the switch symbol (which here serves as a terminal), it would be impossible to distinguish between guessing type behaviors (ic**X**) and inattentiveness behaviors (cc**X**), as both would simply be aa.

# 10 Conclusions

## 10.1 Summary of Results

Overall, target clustering algorithms have performed well on data sets, simulated and empirical, even when the data is difficult for non-target-clustering algorithms. However, each individual analysis shown before forms part of a more cohesive picture, illustrating why target clustering is necessary, feasible, robust, consistent, and reliable. The thesis' main results thus cover:

- **Necessity of Target Clustering**: The existence of an applied problem that cannot be solved by traditional methods.
- **Feasibility and Robustness of Target Clustering**: The ability of target clustering to learn predictive models on real and simulated data sets.
- **Consistency of Target Clustering Results**: Target clustering algorithms produce similar models despite different initial conditions, parameters, and even data sets.
- **Reliability of Target Clustering**: Different runs of a target clustering algorithm can reliably recover collections that are functionally similar. Target clustering algorithms cannot recover arbitrary random collections, however, so the results are reliably indicative of underlying structure in the data.
- **Comparison of Various Algorithms**: Guidepost Weighted-HMMs, despite

being E-M based, is a reliable, efficient, and robust algorithm that produces interpretable models. Linear Distance Reweighting and its future algorithmic cousins offer advantages in performance and optimality, but, at present, do not offer comparable reliability.

- **Educational Implications**: Guidepost algorithms have already provided useful evidence in favor of and against several popular educational theories. In addition, it has shown promise as a means to offer feedback on design changes, in-progress studies, and new educational theories. It is also a plausible solution that can be used by many researchers for many problems.

### 10.1.1 Necessity

Most traditional clustering algorithms are designed to maximize fit on input data, but are not designed to adapt to additional, external measures. Thus, there are at least two ways in which the target clustering algorithms discussed in this thesis differ from most traditional clustering algorithms. First, they learn using sequences, not vectors. Second, they incorporate target information, such as student learning gains. As shown on Geometry02 for a variety of vector algorithms, without target information, it is difficult (or impossible) to find models that predict learning gain. For example, both basic K-Means and Spectral Clustering (both of which learn on vector data) found clusters that performed well on training data, but that performed poorly on test data (test-$R^2$ of $-28$ and $-15$, respectively). In fact, even EM-HMM, the basic algorithm on which the Guidepost HMM algorithms are built, similarly failed to predict learning gains on test data despite learning using full sequential data. However, when Guideposting (and thus, target information) was added to the algorithm, performance improved dramatically.

In addition, when target information is randomized, target clustering algorithms no longer work well. When dealing with randomized targets, poor performance is expected if target information is important to the machine learning process on these data sets. This contrast with random targets further suggests that not only is target clustering important for learning useful models on applied data sets, but that without high-quality target information, even target clustering algorithms will fail.

### 10.1.2 Feasibility and Robustness of Target Clustering Algorithms

For both empirical and simulated data sets, Guidepost HMMs, and in particular, Guidepost Weighted-HMMs, can learn collections that are interpretable and predict performance on both training and test data. The algorithms find predictive models when dealing with extremely simple (Naive) generating models, more realistic (Basic) simulation models, and even constrained, randomized models. The algorithms even find predictive models across multiple variations on two data sets. Test-set $R^2$ range from $0.2$ (Stoichiometry) to the high $0.3$'s (Geometry02). However, Guidepost Weighted-HMMs did not produce useful models on at least one data set (Geometry06), and other algorithms, such as Linear Distance Reweighting, were even less consistent (only producing useful models for specific choices of thresholds).

Importantly, however, it is possible to distinguish between different types of failure, at least for the Guidepost algorithms. For both empirical and simulated data, it can be shown (empirically) that randomizing target information leads to different expected training-set performance than having a poor symbol set (e.g., because the thresholds between fast and slow actions are incorrect). For educational data, this means that there is a detectable difference between cases where learning gain information is either unreliable or unassociated with modeled behaviors and cases where fundamental assumptions about the data set are incorrect. To wit, for all tested threshold combinations and for all forms of preprocessing, no target clustering algorithm has been able to find predictive models for the Geometry06 data set. This result was entirely predictable as target clustering algorithms have not been able to find even suitable fits on training data. This makes it possible to distinguish promising cases (e.g., Stoichiometry) from cases where the discrepancies are irresolvable.

### 10.1.3 Consistency

On simulated data, Guidepost HMM algorithms produce collections that are not only consistent across experiments, but consistent with the generating models. On empirical data, separate experiments with Guidepost HMM algorithms using distinct initial collections (and sometimes different parameterizations) find similar collections of HMMs. In fact, across different (reasonable) threshold choices for

Geometry02, the discovered collections are almost identical. In addition, when sequences of length-1 are removed, the discovered collections adapt and look like the collections discovered with length-1 sequences included, except that those sequences are absent; structurally, however, the HMMs are little changed. This observation is reasonably apparent when comparing collections in Appendix 1.

Consistency is more of an issue for Linear Distance Reweighting (LDR), which only performs well for specific data sets with specific thresholds. While it is possible that the algorithm itself is flawed,[43] it is more likely that the targeting solution is the problem. For example, after LDR learns coefficients for different operations, it assumes that all previous distance calculations still hold, only now using the new coefficients. However, the new coefficients can change the minimum cost path between two sequences, and thus their distance.

### 10.1.4 Reliability

Target clustering can recover the original generating models from simulated data when the original models are well-separated. These well-separated models are still expressive, being reflective of both existing models from the educational literature and of models actually discovered by target clustering in empirical data sets. While there are sometimes significant structural differences between models in the generating and discovered collections, when treated as a whole collection,

[43]For example, LDR could simply be more sensitive to small perturbations within the data or could require a more robust targeting solution.

both sets of models usually classify sequences into similar clusters.

In addition to being able to learn some realistic simulated HMMs, target clustering cannot recover arbitrary, random HMMs given limited amounts of data. For example, on simulated data, Guidepost Weighted-HMMs cannot reliably learn predictive collections given fully-random generating collections, even when using thousands of sequences. However, for the same number of sequences, Guidepost algorithms can learn predictive collections when the generating collections are sparse in the prior matrix and observation matrix. The sparsity requirement is actually useful as it indicates that recovered collections are not spurious. For example, if it was possible to recover predictive collections for any random generating collection, then there would be little reason to believe that collections discovered from empirical data were in any sense meaningful or descriptive of real phenomena. Instead, since target clustering can only discover useful collections if the generating collection belongs to a subspace of possible HMM collections, when target clustering does find something useful in empirical data, it is likely to be a meaningful result.

### 10.1.5 Comparison Between Algorithms

The two target clustering algorithms explored in detail used collections of HMMs in one case and an adaptive distance metric in the other. For the HMM-based algorithms, there was a clear progression from EM-HMMs to Guidepost EM-HMMs to Guidepost Weighted-HMMs. The first algorithm, EM-HMM, found collections of HMMs with good training set performance, but could not find collections with good test set performance. The second algorithm, Guidepost EM-HMMs, added target information to the model selection process and, as a result, found collections of HMMs with good fits to training data that were also capable of predicting learning on test data. Guidepost Weighted-HMMs further improved the fits and predictions of the best collections while also improving the running time of the algorithms. It did this by incorporating targets into the fitting of HMMs, in addition to using them for model selection.

The results for metric learning were less clear cut. While offering an easily computed solution with no local optima, metric learning with targets proved unreliable and inconsistent. For certain choices of thresholds on certain data sets, metric learning provided clusters that were competitive with Guidepost-learned collections; on other data sets, metric learning performed worse than simply predicting the mean. It remains to be seen if the problems with target clustering via metric learning lie with the paradigm,[44] with the specific algorithms,[45] or with the way targets are incorporated.[46]

[44]The class of metric learning algorithms described assume that differences in learning are associated with differences between individual sequences, not merely associated with the distribution of behaviors.

[45]One possibility is that the edit distance fails to capture critical properties of the difference between sequences.

[46]Using one learning score per student is problematic for both metric learning and the HMM algorithms, but is likely an especially significant problem with LDR due to the reweighting.

## 10.2 Educational Implications

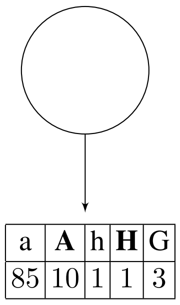
Target clustering is intended to address a fundamental problem in educational research: the generation of plausible, data-driven models (and hypotheses) that relate to actual educational measures. Unsupervised machine learning methods learn interesting models that describe data, but do not necessarily associate well with external measures. Semi-supervised methods produce models that predict external measures and describe learning, but require costly fine-grained labels, and, in addition, cannot generate novel constructs. For example, many researchers have used algorithms to learn models for non-attentive correct answers, e.g. guessing, but their models are constrained by the researcher's original constructs; unless specifically added as a special case, these models would not include, for example, multiple correct inputs on a combo box, which was shown earlier to be highly predictive in the Stoichiometry data. Target clustering has the ability to generate novel models and hypotheses that are also relevant to educational measures, mixing aspects of both unsupervised and semi-supervised methods, all the while using only high-level guidance from targets rather than relying on time-consuming labels for individual sequences.

Since target clustering is designed to generate useful, relevant hypotheses, this section contains arguments for and against various educational hypotheses, as evidenced (or not evidenced) by the results described heretofore. These include arguments in favor and against allowing repeated attempts, in favor and against the inclusion of hints, and against locking answer input widgets. Each of these sections include the discussion of available evidence *as provided by target clustering*, as well as the limitations of that evidence and the algorithms themselves. Finally, there is an extended discussion of target clustering in education generally, both the pros and cons of the algorithms and the approach itself.

### 10.2.1 Against Allowing Repeated Attempts

The most commonly modeled type of repeated-attempt behavior is guessing, the bogeyman of intelligent tutoring system design. Guessing, or at least quickly and repeatedly inputting incorrect answers, is commonly identified by target clustering, appearing in similar guises in every highly predictive collection across every data set. While, technically, the detected behavior consists of many rapid attempts and not necessarily guesses, per se, the model is remarkably consistent across data sets and algorithms. The model is always negatively associated with learning, usually devoid of non-guessing sequences in its partition of the data set, and generally captures sequences of fast incorrect attempts regardless of length. An example was shown in Figure 52 (figure is repeated here for convenience).

When guessing is believed to be a major problem, one solution is to discourage multiple solution attempts on one step. Generally, many tutoring systems discourage guessing either directly, such as by allowing only one submission, or indirectly, such as by visibly lowering student skill estimates after an incorrect answer. The above model provided by target clustering would be supportive of this approach, or possibly a more nuanced one, as it shows that quick, repeated



| a | **A** | h | **H** | G |
|---|---|---|---|---|
| 85 | 10 | 1 | 1 | 3 |

Figure 52: An example guessing HMM from Geometry02-StepA8H8

attempts predicts less learning.

### 10.2.2 For Allowing Repeated Attempts

As shown in the collections learned on Geometry02 and Stoichiometry06, there exist productive behaviors that begin with an incorrect attempt. For example, **IC** is a commonly observed sequence that is generally positively associated with learning. This can be interpreted either as a measure of persistence and engagement or as an indicator of slippage. For the former interpretation, despite making an error on the first attempt, the student persists in rethinking and then solving the step and, eventually, answers it correctly. For the latter interpretation, the student knows the correct answer, makes a small error in its entry, and then corrects it for the second attempt.

An example of a repeated attempts model with a positive association with learning was shown in Figure 51 (figure repeated here).

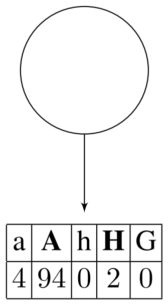


Figure 51: An example repeated attempts HMM from Geometry02-StepA8H8

Regardless of interpretation, the many guessing models do not describe these nuances. However, the collections discovered by target clustering do capture this distinction. Persistence is one of the most commonly observed behaviors and one of the most predictive of learning gains. One hypothesis generated by target clustering, then, is that student learning may improve if certain types of repeated attempts are discouraged while other types are encouraged, allowing for a more nuanced approach to student problem-solving and estimates of student knowledge.

### 10.2.3 Against Hints

Most learning, *as detectable by target clustering*, *appears* to occur at the point of first application, not the time of first explanation. In essence, as described above, attempts are powerful predictors of learning; hints, however, show little to no evidence of being useful for or predictive of learning. The best *positive* predictor of learning found by target clustering is the percentage of steps on which a student either gets the first attempt correct or makes repeated, slow, persistent attempts.

This observation fits well with theories of learning that focus primarily on the application of skills over the reading of instruction or explanations.

Much of the modern educational literature on self-regulated learning and meta-cognition focuses on good help-seeking behaviors and self-explanation. At the heart of these theories is the argument that students learn either in the presence of declarative instruction (e.g., hints) or worked examples (e.g., bottom-out hints). An alternative is a more traditional argument that focuses on each application of a skill as a learning opportunity. For example, in ACT-R, learning of production rules (skills) occurs when they are applied, correctly or incorrectly, rather than when the student reads instruction about the production rule.

Hint-use in the discovered collections for both Geometry02 and Stoichiometry06 falls into two categories: negatively associated with learning and no association with learning. Hint-heavy HMMs negatively associated with learning tend to model sequences with many h, usually one after the other in a long-sequence. This likely indicates drilling down through the layers of hints, looking for either the bottom-out (final) hint or another low-level hint. More neutral[47] hint-heavy HMMs tend to include all other repeated hint behaviors, such as sequences of **H** or sequences mixing hints and attempts. Either way, these models are evidence against the efficacy of hints in the above data sets. They suggest that hints in these tutoring systems are either not helpful or actively detrimental to learning.

[47]Correlation of approximately zero with learning.

### 10.2.4 For Hints

While the discovered collections suggest hint-use is not a useful predictor of learning, it's not clear if students are not learning from the hints at all or if the hints are simply not that useful when students receive direct instruction through other means, such as class instruction and textbooks. It's also not clear if the underlying issue involves the content of the hints, their presentation, or their ordering. It's possible, for example, that showing a worked example instead of layered hints would be more effective.

Further, many theories about self-regulated learning suggest that students require more active support to be productive, self-regulated learners. For example, the self-explanation phenomena, while physically only requiring that students observe worked examples, cognitively requires an active level of engagement with the material. Thus, the lack of an observed hint-effect may be the result of student disengagement. The hints may themselves be useful, but only if students receive an intervention to encourage more engagement.

Finally, there may be useful hint-behaviors that are difficult to detect with these models. One example, still dealing with self-explanation, is prior work detailing a model for productive bottom-out hint use [52]. The core of that model relies on differences in timing between bottom-out hints and non-bottom-out hints, with some adjustments for individual differences in typing time and reading time. That model was positively predictive of learning despite focusing on sequences of actions involving many fast hint requests.

However, Guidepost HMM algorithms cannot necessarily find such a model for several reasons:

- The symbol set does not distinguish bottom-out hints from non-bottom-out hints. Adjusting this would require more data to compensate for the relative infrequency of bottom-out hints.
- Thresholds disguise subtle differences in the timing of individual actions.
- Guidepost HMM algorithms model the distribution of actions per student, not the mean time per action per student.
- Adjusting for individual differences in the Shih et al. bottom-out hint model requires access to the first action in the following steps, something not available to target clustering in its present design.

These arguments all suggest that there may exist useful hint-related behaviors that have simply gone undetected, due to limitations in the modeling assumptions, in the algorithms, or in the data. The latter seems particularly possible as these data sets only have tens of students from whom to learn parameters for three to six HMMs.

### 10.2.5 Against Locking Input Widgets

Though a somewhat silly sequence, cc**X**[48] is one of the most useful, predictive sequences found in any of the above data sets. Given that this sequence occurs mostly with the combo box[49] interface widget, this sequence likely indicates a kind of absent-mindedness or lack of engagement by the student.[50] This sequence, while it does occur in other data sets (e.g., Geometry02), is usually less common. A strong explanation is that combo boxes are not "locked," that is, a student can enter an answer in them more than once; on the other hand, the Geometry Cognitive Tutor will attempt to block further attempts after a step has been solved.[51]

Usually, locking widgets after a correct attempt is considered useful because it decreases slips,[52] making it easy to tell what steps remain incomplete and helping to keep logs clean. However, the cited collections provide an argument against locking solutions, since by not locking them, it may be possible to glean useful data about a student's engagement. This represents an implementation-dependent, but potentially useful model that is both unlikely to be a priori theorized by a researcher and unlikely to be detected by general unsupervised algorithms, making it uniquely a product of target clustering.

[48]As a reminder, cc**X** corresponds to a quick correct, another quick correct, and then a switch to a new step.

[49]A combo box is a drop-down menu showing multiple options available for selection.

[50]Alternative explanation: students engage in cc**X** behaviors due to a lack of impulse control. This hypothesis would have similar implications.

[51]Due to latency between submitting an answer and having the step locked, it is possible to submit an answer twice.

[52]A slip is when a student makes an error despite knowing the skill(s) required to complete a step.

### 10.2.6 Against Target Clustering in Education

For all the interesting results cited above, there are practical problems with target clustering for education data. First, the most effective target clustering algorithms tested to-date are based on Expectation-Maximization (EM). EM algorithms are prone to local optima, which makes it difficult to interpret the models as representing fundamental truths.

Second, the most effective algorithms to-date also rely on hidden Markov models. While HMMs are powerful and interpretable, there is a lack of a strong theoretical foundation for their use in modeling student behaviors. For example, HMMs assume that all necessary information is represented in the state of the HMM and no other past history is required. For sufficiently large state spaces, this is probably true, but for the small number of states used in target clustering, this becomes a problematic assumption.

Third, target clustering as a *paradigm*, independent of implementation, adds additional complexity and sensitivity to the machine learning process. Most algorithms are only dependent on the quality of the raw data; target clustering algorithms are dependent on the quality of the raw data *and* the quality of the targets. If the targets are poor (e.g., an unreliable assessment) then the results of target clustering will be poor as well. In fact, it is possible for the raw data to be high quality and for an assessment to be high quality while betwixt them, in the intersection of the two, is trace data that is uncorrelated with the assessment. This makes target clustering potentially more sensitive to imperfections in available data.

### 10.2.7 For Target Clustering in Education

Each of these arguments also has a counterpoint. First, while EM is hypothetically problematic, in practice, Guidepost HMM algorithms have been robust to changes in initial condition, parameters, and even data sets. Second, while HMMs may include some theoretically unsound assumptions regarding education data, they are also highly effective models not only of student behaviors, but also of behaviors in other domains. Third, while the interaction between data quality and target quality is potentially troublesome, it's clear from earlier results that the lack of targets is a far worse liability.

Thus, despite its limitations, target clustering serves many useful functions. Target clustering algorithms have demonstrated effectiveness in educational data for five purposes:

- Generation of predictive models.
- Exploratory analysis of data sets.
- Evaluation of assessments.
- Evaluation of interface widgets.
- Adaptation to new data sets and systems.

Each of these uses will be covered in turn. Many of them will include a concrete, though hypothetical, example of how target clustering could be applied by an education researcher during the course of a research project.

### Generation of Predictive Models

While not a function unique to target clustering algorithms, the ability to learn predictive models from non-experimental data is exceedingly valuable. In general, target clustering algorithms have proven quite capable at this task, discovering models with an $R^2$ as high as 0.5 ($r \approx 0.7$). Admittedly, 0.7 is not an exceedingly high correlation with learning, but ample, and notably, in the present experiments, target clustering algorithms have no access to skill models, actual student answers, or demographic information.[53] While somewhat limiting, by not depending on skill models or complicated answer comparisons, target clustering algorithms are detail agnostic: essentially, given a data set with proper formatting and a few candidate thresholds, target clustering algorithms can produce predictive models with no significant preprocessing.

A hypothetical example of how this can be useful is described below:

[53]Steps to address these limitations are explored in the next section.

A researcher is running a pilot for a new version of a tutoring system. Part way through the study, the researcher wants to know whether the changes are improving outcomes and, if not, to make further modifications to the system. Rather than take time from the study to administer a formative assessment, the researcher can run target clustering algorithms on data from a prior version of the tutor, then use those models to predict outcomes for the students in the pilot study. Using those predictions, the researchers can compute the probability that this batch of students will have final outcomes better than outcomes for students in the prior study.

**Exploratory Analysis of Data Sets**

Unlike some domains, such as physics or chemistry, the fundamental tenets and principles of education are still uncertain. For example, empirically productive learning behaviors in one domain may be completely ineffective in another, e.g., rote memorization for multiplication facts versus algebra problem solving. There do exist important principles and tenets for education, but the list is far from exhaustive. As a result, when analyzing a new data set, it is insufficient to rely on prior models and theories.

As part of an arsenal of tools, analytical and descriptive, target clustering can help address this problem. Target clustering is not useful for verifying an existing model: for that task, there are many statistical tools available. Target clustering is also not useful for implementing a researcher's hypothesis: for that task, human coding schemes are generally more useful. However, if there does not yet exist a specific model or a clear hypothesis, target clustering is a powerful tool for generating hypotheses (and corresponding models).

A hypothetical example is described below:

A researcher is analyzing a data set for patterns in student guessing. The researcher uses a model from prior work that emphasizes long sequences of repeated, similar answers as indicative of guessing. However, rather than simply relying on one model, the researcher also uses target sequence clustering. As a result, the researcher discovers that there are, in fact, two types of repeated attempts sequences in the data set, and can construct a hypothesis regarding the difference.

## Evaluation of Assessments

As shown for both simulated and empirical data sets, target clustering provides a useful heuristic indicator for the usefulness of target $\lambda$ values. When these $\lambda$ values are the results of educational assessments, it is possible to use this heuristic indicator to evaluate the usefulness of the assessment. For example, if target clustering produces correlations that are predominantly zero, even on training data, then the assessment has little to no discernible correlation with observed behaviors.[54] Such an assessment merits further examination.

[54] In so far as the behaviors can be represented by hidden Markov models.

**Evaluation of Interface Elements**

One interesting feature of target clustering algorithms is the ability to discover interesting behaviors that are associated with a specific interface element. This can be done in one of two ways: either the behavior can be post-hoc associated with certain interface interactions or interactions with a specific interface element can be isolated a priori. The first approach was already illustrated in the stoichiometry data set where an unusual behavior (**CCX**) was found to be highly associated with an interface element (combo boxes). This suggests that something about combo boxes leads to or allows this behavior.

An illustration of the second approach is described below:

A researcher is curious about how students interact with a new graphical input tool for a geometry angles unit. First, the researcher isolates all steps where a student uses the graphical input tool. Second, the researcher applies target clustering algorithms to this data. The researcher is surprised to learn that students who use this tool the fastest, and thus demonstrate the most mastery, are actually learning less. The researcher can then adjust the interface element accordingly.

**Adaptation to new data sets and systems.**

Even in the presence of known interface elements, a strong theoretical foundation, and reliable assessments, a lot can change when existing software is moved into a new classroom or existing principles are applied to new software. In the first case, the new students in a data set can exhibit behaviors that were not observed before. These potential new behaviors present a fantastic opportunity to apply target clustering algorithms that can discover novel behaviors if they associate with learning (i.e., are relevant). In the second case, even the same students using a new tutoring system can exhibit different behaviors or learn differently given the same behaviors. For example, in a tutoring system for complex reasoning and argument, slow answers may be highly predictive of learning. On the other hand, in a tutoring system for multiplication tables, students that answer slowly for a long time are probably not learning. Target clustering can identify these differences before expensive experiments or researcher-intensive modeling is required.

# 11 Future Work

## 11.1 Improvements to Existing Algorithms

The evidence discussed above suggests that Guidepost HMM algorithms are effective at learning models that predict learning.[55] In addition, the learned HMMs fit other desirable criteria such as interpretability and limited model complexity. However, there are many ways Guidepost HMM algorithms can be improved. The most important is eliminating the Expectation-Maximization (EM) searches in the learning process. A particularly elegant solution would be to learn a col-

[55]In particular, see Table 3 on page 81 and Figure 50 on page 130 for illustrations of the effectiveness of Guidepost HMM on real data.

lection of HMMs in a single-step. There are algorithms for learning individual HMMs without EM;[56] there are also many algorithms for learning clusters without EM.[57] Combining these two types of algorithms could create an algorithm that can learn clusters of HMMs without using EM, since clusters of HMMs are equivalent to single HMMs with block-diagonal transition matrices. However, constraining such results to use targets is non-trivial.

In addition, Guidepost HMM would benefit from many simpler tweaks and improvements. For example, algorithms that learn multiple HMMs often feature steps that either split or merge existing HMMs. Initial attempts at integrating these steps into Guidepost HMM algorithms were not promising, but if the splits and merges were properly constrained by target information, the results might be improved. Further, there is room to experiment with substituting a variety of HMM variants for basic, vanilla HMMs, as well as opportunity to try predictor functions besides linear regression.

There are many opportunities for improving the existing metric learning algorithms. LDR, for example, is a basic approach to solving the targeting problem for distance-based clustering algorithms that can be improved with many ideas available from the semi-supervised clustering literature.[58] For example, combin-

[56]Example algorithms for learning HMMs without EM: subspace ID [20], phylogenetic trees [42], and matrix factorization [19]. Initial attempts to embed some of these methods into the Guidepost HMM algorithm did not show promising results, but the causes of this are unknown.

[57]A particularly relevant method for learning clusters of HMMs without EM is to train an HMM for every sequence, compute a distance between the HMMs, and then cluster with the distance matrix. A prime example of this method is Jebara et al.'s work [28], which uses one particular probabilistic kernel to compute the distance between HMMs.

[58]Useful examples from the semi-supervised clustering literature are discussed in Section 4.2,

ing a distance metric like the edit distance with a measure of cluster purity seems like a promising approach, where purity is a measure on the distribution of learning gains represented in the cluster. However, initial attempts at this have not yet yielded productive results. It's possible that a more complex purity measure, or perhaps another similarity-adapting approach in general, will prove more promising.

## 11.2 Hierarchical Target Clustering

Many learning tactics (even amongst those representable by collections of hidden Markov models) cannot be learned from step-level sequences. For example, students sometimes do work on paper first, solving each step in a problem, and then input all the steps at once. Over the course of a problem, this student's behavior generates a markedly different trace than the behavior of a student who, for example, solves many of the steps in order, easily, and in one try, but struggles on the last (and hardest). For the former student, the trace might look like c**X**c**X**c**X**c**X**. For the latter student, the trace might look like c**X**c**X**c**X**ii**ICX**. However, at the step-level, these two students would look $75\%$ similar, making it difficult to distinguish them once the data is aggregated across many problems and students.

Unfortunately, initial attempts to tackle problem-level behaviors have only been mildly effective. While target clustering algorithms can fit descriptive and predictive models for problem-level behaviors, the models are difficult to interpret and

page 45.

inconsistent across runs. One likely cause of both problems is a fundamentally incorrect assumption: rather than treating problems as sequences of step-level behaviors, problem-level sequences simply treat problems as sequences of individual actions. Take the above example. Imagine two HMMs: “Correct”, which has a high probability of generating c**X**, and “Struggle”, which has a high probability of generating ii**ICX**. Then the above problems could be represented by a pair of hierarchical HMMs, one of which generates sequences like CorrectCorrectCorrect-Correct and another that generates sequences like CorrectCorrectCorrectStruggle. In this way, both step-level and problem-level behaviors can be included in the same models.

For hierarchical target clustering to be effective would require not only developing ways to constrain multiple levels of the hierarchy at once, but also to allow optimizations between levels of the hierarchy. For example, the inefficient EM approach would be to try to optimize the step-level HMMs while fixing the problem-level HMMs, and then to optimize the problem-level HMMs while fixing the step-level HMMs. To be truly efficient and effective would require constraining both, and this makes this problem interesting from both a practical and theoretical standpoint.

## 11.3 Multiple Targets

While not yet used in this work, methods from multilabel learning are important to consider. There are many cases in which there exist multiple interesting educational measures, e.g., surveys, tests, etc. It would be useful to be able to search for behaviors related to combinations of these measures, e.g., behaviors related to self-efficacy surveys *and* physics assessments. For this task, research into multilabel learning is a good source for potential methods.

As described in the Background section, problem transformation methods are promising for cases with multiple targets. Let PT1 be the problem transformation where a classifier is trained for each label and the results from each classifier are combined. When adapted to target clustering, this allows any target clustering algorithm for single $\Lambda$ to be used for the multiple $\Lambda$ case, albeit with some tweaks. Also consider PT2, the problem transformation where each set of possible labels is converted into a new label. This approach is also relatively simple and may be promising for problems with discrete $\Lambda$. There are other problem transformations, but they are less readily adapted. In addition, more complicated problem transformations, such as those that involve pruning, will not be addressed as the work proposed herein does not consider cases with many $\Lambda$ or imbalanced sets of $\Lambda$ values.

## 11.4 Semi-Supervised Target Clustering

One of the major limitations of target clustering algorithms at present is their inability to incorporate other labels. For example, an education study might call for the use of a coding scheme whereby human annotators label steps as being “guessing” or “not guessing”. However, to target clustering, these labels may as well not exist as target clustering algorithms are, at present, purely unsupervised algorithms.[59] The ability to incorporate some labels into target clustering would extend into semi-supervised learning.

From an education standpoint, this is equivalent to saying that target clustering can be constrained to include certain behaviors in its model, e.g., guessing. This also indicates one easy approach to this problem: sequences with labels can be a priori separated from the other data and an HMM learned for each category. These HMMs can be used as seeds in each iteration of the learning algorithm.

In practice, however, the above approach is unsatisfactory. A more sound approach would include probabilistic constraints on the models, or perhaps penalize models for containing multiple sequences from the same category. There exists a wide range of literature on approaches to semi-supervised clustering that can be mined for additional ideas. In addition, structured graphical models[60] could provide a solution. In particular, there has been some work on embedding HMMs

[59]Some might argue that the addition of target data makes target clustering algorithms semi-supervised. However, it’s worth remembering that the targets are not what the algorithm is trying to learn, and that it is actually learning cluster membership for individual sequences. None of these clusters or cluster memberships are provided.

[60]See Section 4.1, page 44.

into probabilistic relational models [2, 41]. Combining these approaches with Guidepost could allow for modeling of complex relationships with existing labels and multiple targets, providing a particularly promising avenue for research.

## 11.5 Additional Data

### 11.5.1 Skill Models

Skill models are essentially mappings, often one-to-one but sometimes many-to-one, between skills and steps. For example, one step might require the *Complementary Angles* skill, while another step might require the *Triangle Angles Sum to* $180°$ skill. A good skill model is incredibly powerful: target clustering assumes that learning is a product of behaviors as related to one skill, e.g., “Geometry”; skill models assume that learning occurs for each skill in the model separately, so a student can learn about Complementary Angles and still not know that the angles of a triangle sum to $180°$. This disconnect is one of the main reasons that target clustering is theoretically limited in the accuracy of its models: simply put, it is impossible to fully predict how much a student learns without knowing something about the content of the problems.

There is, however, one approach to this issue that derives directly from the target clustering paradigm. It relies on the probability that a student has mastered a skill, which derives from the skill model and prior student performance. Consider Guidepost Weighted-HMMs, which essentially weights each sequence by the expected target value (e.g., learning gain) amongst the students that exhibit that sequence. Assume without loss of generality that each step has only one associated skill. Instead of weighting sequences by the target values, it is also possible to weight each sequence by the probability the student has mastered a skill. Thus, for a model positively associated with learning, each sequence might be weighted by the probability the student has mastered the skill; for a model negatively associated with learning, each sequence might be weighted by the probability that the student has not mastered the skill. In this way, the skill model is integrated directly into the model training process.

### 11.5.2 Continuous Durations

At present, Guidepost HMM algorithms use thresholds to change continuous times-per-action into discrete ones. This transformation is potentially limiting, as it not only makes the algorithm dependent on a good choice of thresholds, but also potentially hides nuances in the data that depend on the full continuous value. An obvious solution to this problem is to replace discrete HMMs with continuous HMMs. However, this requires either making distributional assumptions about the durations or choosing to use a non-parametric approach. Unfortunately, the former is non-trivial: not only is there no consensus on a general distribution for times-per-action, but every tutoring system is also unique. The second option, using a non-parametric approach, is potentially interesting in a different way. Many non-parametric estimators could interact with Guidepost iterations, e.g., relearn-ing the distribution approximation after each collection is trained to better reflect the distributions for each HMM. This topic is a promising avenue for research, both for its practical application and potential theoretical interest.

## 11.6 Preprocessing

### 11.6.1 Seeding

The Guidepost-HMM algorithms can benefit enormously from seeding (as can many algorithms discussed in Future Work). For target clustering, seeding involves modifying the data or initial models to better reflect targets. For example, there are a number of ways to choose seed HMMs for Guidepost algorithms. One approach is to split students into high-learners and low-learners, and then to learn an HMM for each. Another approach is to learn HMMs for each student and merge the ones that result in the best clusters.

### 11.6.2 By Other Models

An interesting possibility is to use target clustering algorithms to deal with sequences not well-classified by other models. For example, take a non-exhaustive model for student meta-cognition. This model might deal with proper hint use, hint-abuse, and guessing; however, there may also be many sequences that do not fit any of those categories. In that case, the sequences with a low probability of belonging to any of the above categories can be combined into one data set. Target clustering can be applied to this data set to generate hypotheses as to what patterns and behaviors may be represented within.

## Final Thoughts

Target clustering algorithms have shown promising results on several data sets, simulated and empirical. Target clustering algorithms have even produced results of interest from an educational perspective. However, the real excitement for target clustering lies in the future: in improving the existing algorithms, in adapting algorithms into the target clustering paradigm, in combining target clustering with other approaches like probabilistic relational models, in exploring a fully hierarchical form of target clustering, in extending the existing algorithms to support skill models and continuous variables, and, of course, in exploring new data sets and domains. Overall, this work is only the first step in research on informed clustering for educational data sets, but it provides a foundation on which to build and opens up doors to a wide array of future work.

# 12 Appendix 1 — Geometry02 Collections

## 12.1 Geometry02-StepA8H8

### 12.1.1 Simple 3-HMM Collection

Test-Set $R^2$: $0.36$

- Simple 3-HMM Collection — "Dominant Model", Regression Coefficient: $0.66$, Figure 57

  77 unique sequences, 4763 total sequences

  | **Sequence** | **Frequency** |
  |---|---|
  | **A** | **4042** |
  | **AA** | **343** |
  | **AAA** | **72** |
  | **AaA** | **44** |
  | **aAA** | **39** |

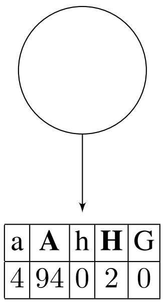


Figure 57: Simple 3-HMM Collection — "Dominant Model"

- Simple 3-HMM Collection — "Guessing Model", Regression Coefficient: $-0.66$, Figure 58

  124 unique sequences, 6156 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **a** | **5062** |
| **Aa** | **259** |
| **aa** | **257** |
| **aA** | **220** |
| **aaa** | **63** |

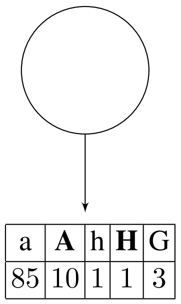


Figure 58: Simple 3-HMM Collection — "Guessing Model"

- Simple 3-HMM Collection — "Hints and Miscellaneous", Regression Coefficient: $-1.8$, Figure 59

  935 unique sequences, 2525 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **hhhhhHA** | **117** |
| **hhhhhha** | **97** |
| **hhhhhhha** | **75** |
| **hhhhha** | **65** |
| **hhhHA** | **59** |

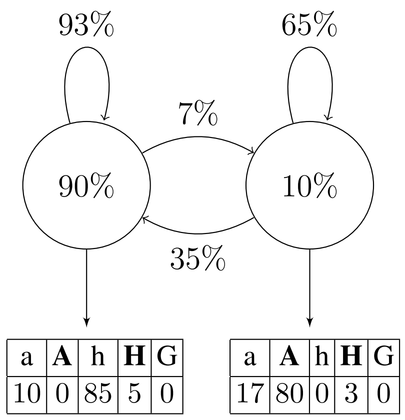


Figure 59: Simple 3-HMM Collection — "Hints and Miscellaneous"

### 12.1.2 Single-State Collection

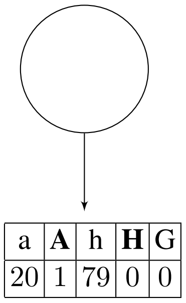
Test-Set $R^2$: 0.25

- “Hint-Abuse Model”

  Regression Coefficient: -1.8, Figure 60

  797 unique sequences, 2335 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **hhhhhHA** | **117** |
| **hhhhhha** | **97** |
| **hhhhhhha** | **75** |
| **hhhhha** | **65** |
| **hhhHa** | **59** |



| a | **A** | h | **H** | G |
|---|---|---|---|---|
| 20 | 1 | 79 | 0 | 0 |

Figure 60: Single-State Collection — “Hint-Abuse Model”

- “Dominant Model”

  Regression Coefficient: 0.75, Figure 61

  22 unique sequences, 4547 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **A** | **4042** |
| **AA** | **343** |
| **AAA** | **72** |
| **AAAA** | **23** |
| **aAAA** | **21** |

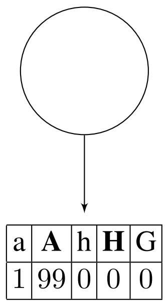


Figure 61: Single-State Collection — “Dominant Model”

- “Guessing Model”

  Regression Coefficient: -0.61, Figure 62

  131 unique sequences, 6267 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **a** | **5062** |
| **Aa** | **259** |
| **aa** | **257** |
| **aA** | **220** |
| **aaa** | **63** |

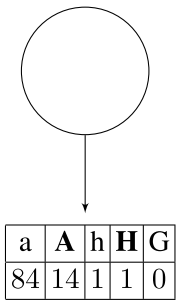


Figure 62: Single-State Collection — “Guessing Model”

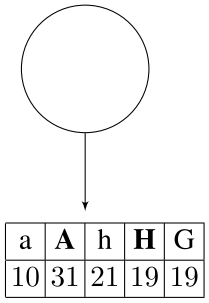
- “Miscellaneous Model”

  Regression Coefficient: 0.39, Figure 63

  186 unique sequences, 295 total sequences[61]

[61] As an example of how varied the sequences assigned to this model actually are, this model’s partition of the data sets includes: **AA**aahh**HHA**.

| **Sequence** | **Frequency** |
|---|---|
| **HA** | **27** |
| **hHA** | **9** |
| **AhHA** | **8** |
| **hhGA** | **8** |
| **GA** | **8** |



| a | **A** | h | **H** | G |
|---|---|---|---|---|
| 10 | 31 | 21 | 19 | 19 |

Figure 63: Single-State Collection — “Miscellaneous”

## 12.2 Geometry02-StepMeanSwitchCorrect

### 12.2.1 Simple Collection

Test-Set $R^2$: $0.16$

- “Hint-Abuse Model”

  Regression Coefficient: -0.72, Figure 64

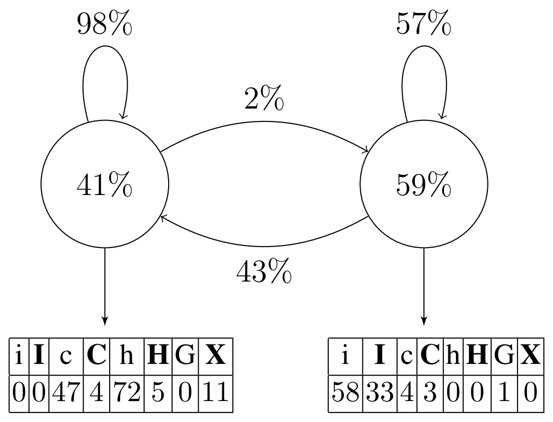
  966 unique sequences, 2381 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **hhhhhhcX** | **131** |
| **hhhhhhhcX** | **85** |
| **hhhhhcX** | **79** |
| **hhhhhHCX** | **72** |
| **hhhhcX** | **45** |

- “Quick Attempts Model”

  Regression Coefficient: -0.15, Figure 65

  65 unique sequences, 710 total sequences

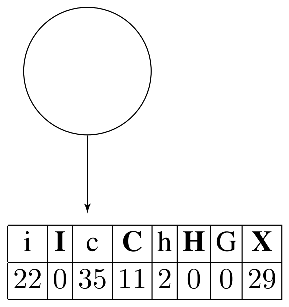


| i | **I** | c | **C** | h | **H** | G | **X** |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 47 | 4 | 72 | 5 | 0 | 11 |

| i | **I** | c | **C** | h | **H** | G | **X** |
|---|---|---|---|---|---|---|---|
| 58 | 33 | 4 | 3 | 0 | 0 | 1 | 0 |

Figure 64: Simple Collection — "Hint-Abuse Model"

| **Sequence** | **Frequency** |
|---|---|
| **icX** | **268** |
| **cCX** | **86** |
| **CcX** | **75** |
| **iicX** | **66** |
| **ccX** | **56** |



| i | **I** | c | **C** | h | **H** | G | **X** |
|---|---|---|---|---|---|---|---|
| 22 | 0 | 35 | 11 | 2 | 0 | 0 | 29 |

Figure 65: Simple Collection — "Quick Attempts Model"

- “Persistent Attempts Model”

  Regression Coefficient: 0.15, Figure 66

  208 unique sequences, 921 total sequences

| **Sequence** | **Frequency** |
|---|---|
| **IcX** | **155** |
| **ICX** | **141** |
| **iCX** | **116** |
| **CCX** | **70** |
| **IIcX** | **26** |

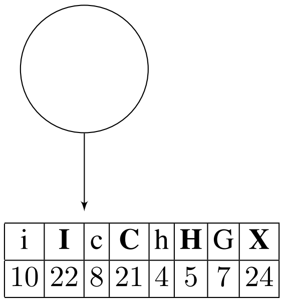
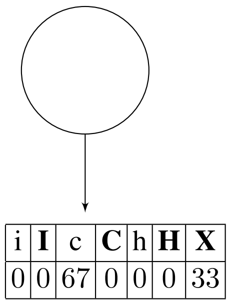


Figure 66: Simple Collection — “Persistent Attempts Model”



| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 0 | 0 | 67 | 0 | 0 | 0 | 33 |

Figure 67: Quick Click Collection — “Two Quick Corrects”

# 13 Appendix 2 — Stoichiometry06 Collections

## 13.1 Stoichiometry6-StepMeanSwitchCorrect

### 13.1.1 Quick Click Collection

Test-Set $R^2$: $0.20$

- “Two Quick Corrects”

  Regression Coefficient: $-0.7$, Figure 67

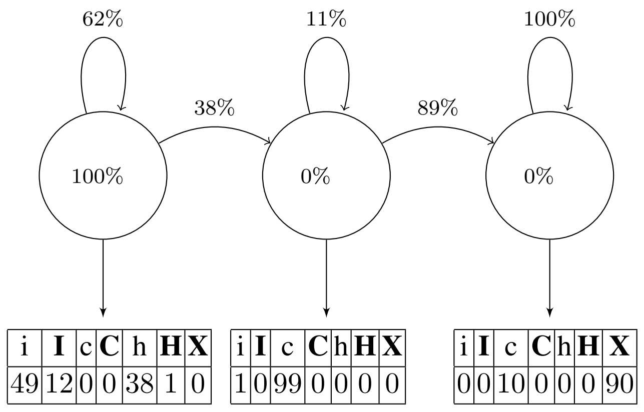
  1 unique sequence, 985 total sequences

  | **Sequence** | **Frequency** |
  |---|---|
  | **ccX** | **985** |

- “Quick Wrongs Model”

  Regression Coefficient: $-0.49$, Figure 68

  128 unique sequences, 937 total sequences



| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 49 | 12 | 0 | 0 | 38 | 1 | 0 |

| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 1 | 0 | 99 | 0 | 0 | 0 | 0 |

| i | **I** | c | **C** | h | **H** | **X** |
|---|---|---|---|---|---|---|
| 0 | 0 | 10 | 0 | 0 | 0 | 90 |

Figure 68: Quick Click Collection — “Quick Wrongs Model”

| **Sequence** | **Frequency** |
|---|---|
| **icX** | **355** |
| **iiccX** | **87** |
| **iicX** | **61** |
| **hcX** | **56** |
| **hhhhhhccX** | **29** |

- “Trash”

  Regression Coefficient: $0.04$, Figure 69

  152 unique sequences, 435 total sequences

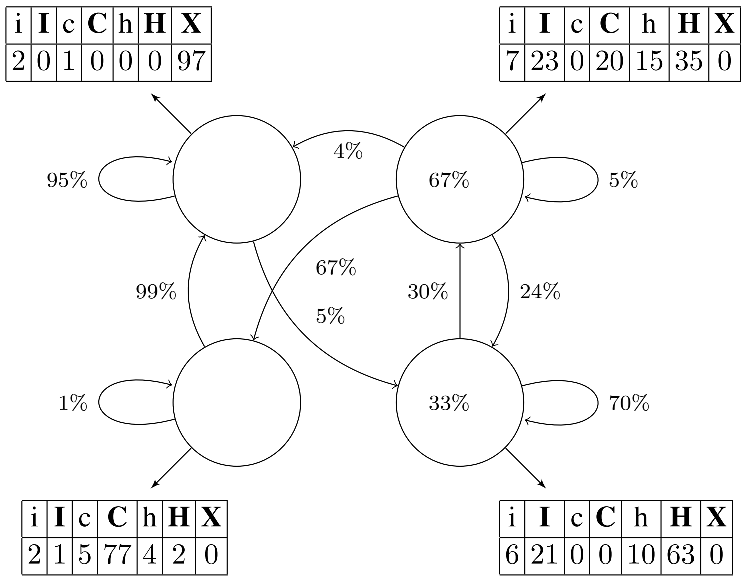


Figure 69: Quick Click Collection — “Trash”

| **Sequence** | **Frequency** |
|---|---|
| **CCX** | **112** |
| **ICX** | **65** |
| **HCX** | **29** |
| **hHCX** | **8** |
| **hhHCX** | **8** |

- “Repeated Attempts Model”

  Regression Coefficient: $0.13$, Figure 70

  80 unique sequences, 409 total sequences

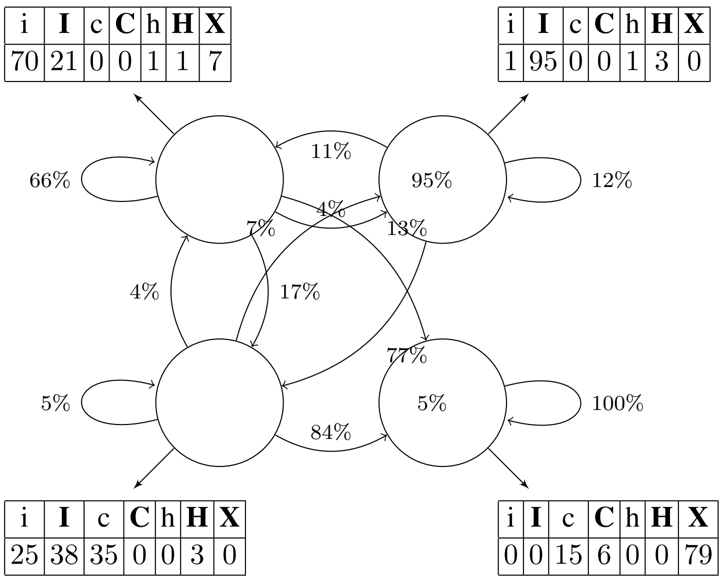


Figure 70: Quick Click Collection — “Repeated Attempts Model”

| **Sequence** | **Frequency** |
|---|---|
| **IcX** | **112** |
| **IIX** | **65** |
| **IicX** | **29** |
| **IiX** | **8** |
| **IIccX** | **8** |

- “Guessing Model”

  Regression Coefficient: $-0.27$, Figure 71

  77 unique sequences, 384 total sequences

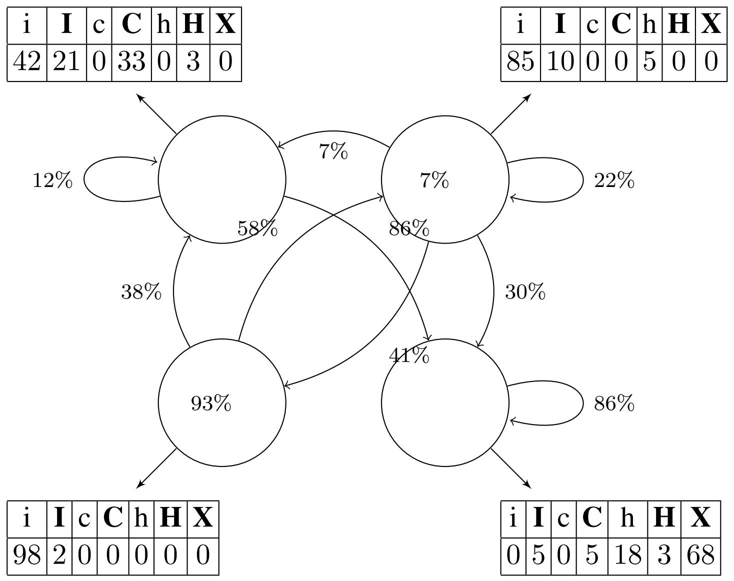


Figure 71: Quick Click Collection — “Guessing Model”.

| **Sequence** | **Frequency** |
|---|---|
| **iiX** | **121** |
| **iCX** | **80** |
| **iIX** | **18** |
| **iiiiX** | **16** |
| **iiiX** | **14** |

# 14 Appendix 3 — Sample Code for Guidepost EM-HMMs

This appendix contains sample Matlab code for Guidepost EM-HMMs. The code has been simplified to the bare minimum for ease of readability, so many secondary options and much of the error checking have been removed. It does, however, represent a fully functional implementation of Guidepost EM-HMMs excepting two blocks of code:

- The code assumes the existence of an hmm class with a constructor, a method learn() for training the HMM, and a method p() for determining the probability of a sequence. Depending on the implementation of these methods, Guidepost can use different flavors of HMMs.
- The code assumes the existence of a stepwisefit method that performs stepwise regression.

## 14.1 tgt_experiment

```
% tgt_experiment - Main class for the target clustering demo code.
%
% For the purposes of this demo code, assume that the train/test sets are
%  already setup.  Assuming n is the number of steps in the training set
%  and m is the number of students, then train is a struct with:
%  train.students = n-dim cell array of strings, one per step
%  train.sequences = n-dim cell array of vector, one per step
%    each vector contains integers corresponding to each action type
%  train.gains is a struct with:
%    train.gains.data = m-dim vector with the learning gains
%    train.gains.textdata = m-dim cell array with corresponding students
%    overall, train.gains is the output of an import.csv on a csv file with
%      2 columns, Student and Lambda
%
%  hmm is assumed to be an implementation of HMMs with the following:
%    hmm(observations,states) -> new random hmm
classdef tgt_experiment
    properties
        nObservations % The # of observation symbols.  For convenience.

        train
        test

        collections
    end

    methods
        function obj=tgt_experiment(nObservations,train,test)
            obj.nObservations = nObservations;
            obj.train = train;
            obj.test = test;
        end

        function obj=train_collections(obj,maxHmmNo,maxStateNo,minLLPct)
            hmms = {};
            for m=2:maxHmmNo
                for s=2:maxStateNo
                    display(strcat(num2str(m),' HMMs, ',num2str(s),' states.'));

                    % Add states
                    for k = 1:(m-length(hmms))
                        hmms = [hmms; {hmm(obj.nObservations,s)}];
                    end
                    % Assume this outputs a struct with:
                    %   temp.hmms = the best collection of HMMs found
                    %   temp.clusters = vector mapping seq to clusters
                    %   temp.nIters = number of iterations required
                    temp = tgt_emhmms(hmms,obj.train.sequences,minLLPct);

                    % Now setup the collection, i.e., hmms & train fit
                    collection={}; % A temporary result holder.
                    collection.hmms = temp.hmms;
                    collection.nHmms = m;
                    collection.nStates = s;
                    collection.train = tgt_regressGains(obj.train.gains,obj.train.students,m,temp.clusters);
                    % collection.train holds the regress fit and coeff

                    % Now apply the collection to test data...
                    temp = tgt_classifyHmms(collection.hmms,obj.test.sequences);

                    % And compute test fits
                    clusters = temp.clusters;
                    collection.test=tgt_regressGains(obj.test.gains,obj.test.students,m,clusters,
                        collection.train.fit.intercept,collection.train.fit.coeffs);

                    obj.collections = [obj.collections {collection}];

                    % Pick survivors
                    inmodel = collection.train.fit.inmodel;
                    hmms = collection.hmms(inmodel);

                    display(strcat(num2str(m - sum(inmodel)),' HMMs removed.'));
                end
            end
        end
    end
end
```

## 14.2 tgt_classifyHmms

```
% tgt_classifyHmms - Classifies a set of sequences using given hmms.
% hmms = cell array of m hmm objects
% sequences = cell array of n vectors (of integers)
% classification.clusters = n-dim vector, each entry from 1..m
% classification.loglik = sum of loglikelihood
function [ classification ] = tgt_classifyHmms( hmms, sequences )

    classification.clusters = [];
    classification.loglik = 0;

    % Just loop through sequences, assigning each to the best-fit hmm
    for i = 1:length(sequences)
        cluster = 1;
        loglik = log(hmms{1}.p(sequences{i}));
        for j = 2:length(hmms)
            ll = log(hmms{j}.p(sequences{i}));
            if ll > loglik
                loglik = ll;
                cluster = j;
            end
        end
        classification.clusters = [classification.clusters; cluster];
        classification.loglik = classification.loglik + loglik;
    end
end
```

## 14.3 tgt_emHmms

```
% tgt_emHmms - Returns a collection of hmms optimized with EM from
%   a given initial set of hmms and sequences
% hmms = cell array of m hmm objects
% sequences = cell array of n vectors (of integers)
% minLLPct = minimum percentage by which loglik must improve each iter
function [ collection ] = tgt_emHmms(hmms,sequences,minLLPct)

    % Arbitrarily set to simplify later logic for the first iteration
    loglik_prev = -900000;
    loglik_present = -800000;

    nIters = 0;

     % Repeat until the new loglikelihood is within minLLPct
    while loglik_prev < loglik_present*(1+minLLPct) && loglik_present < 0
        % Prep the iteration
        nIters = nIters + 1;
        loglik_prev = loglik_present;
        loglik_present = 0;

        % Classify sequences
        collection = tgt_classifyHmms( hmms, sequences );

        % For readability
        clusters = collection.clusters;
        loglik_present = collection.loglik;

        display(strcat('Iter: ', num2str(nIters), ', Total Loglik: ', num2str(loglik_present)));

        % Relearn hmms
        for j = 1:length(hmms)
            % Grab only data in this partition
            data = {};
            for i = 1:length(sequences)
                if clusters(i) == j
                    data = [data sequences{i}];
                end
            end

            % Only try to relearn if the partition is non-empty
            if(~isempty(data))
                hmms{j} = hmms{j}.learn(data);
            end
        end
    end

    collection.clusters = clusters;
    collection.hmms = hmms;
    collection.nIters = nIters;
end
```

## 14.4 tgt_regressGains

```
% tgt_regressGains - Used for both computing a stepwise regression of
%   cluster distribution against gains and for applying an existing set of
%   coefficients to new data
% gains = student learning gains
%  gains.data = m-dim vector with the learning gains
%  gains.textdata = m-dim cell array with corresponding students
%  overall, gains is the output of an import.csv on a csv file with
%      2 columns, Student and Lambda
% students_clusters = n-dim cell array of strings, maps students to sequences
% nClusters = number of clusters
% clusters = n-dim vector of ints, maps sequences to clusters
% varargin = a number (intercept) and an m-dim vector of coefficients
function [ results ] = tgt_regressGains(gains,students_clusters,nClusters,clusters,varargin)

    % Assumes that gains includes a filler column specifying the student.
    students = gains.textdata(1:length(gains.textdata)); % Wipe the 2nd col
    lambda = gains.data;

    students(1) = []; % Delete headers, we know what they are already

    % Aggregate the clusters
    x = zeros(nClusters,length(students));
    for i = 1:nClusters
        indices = (1:length(clusters));
        indices = indices(clusters == i);
        for j = indices
            index = find(ismember(students,students_clusters{j})==1,1);
            x(i,index) = x(i,index) + 1;
        end
    end

    x=x';

    % Remove 0'ed students
    for i = length(lambda):1
        if(sum(x(i,:)) == 0)
            display(strcat('Removing student: ',num2str(i)));
            x(i,:) = [];
            lambda(i) = [];
        end
    end

    x=tgt_normalize(x); % Row normalization of X
    % Clear NANs
    select = arrayfun(@(i) not(isnan(x(i,1))), 1:nrows(x));
    x=x(select,:);
    students=students(select);
    lambda=lambda(select);

    nStudents = length(students);

    % If varargin is length 0, then we're doing a stepwise regression for
    %   model selection
    % If varargin is length > 0, then we're applying existing coefficients
    if(length(varargin)==0)
        [fit.coeffs,fit.se,fit.pvals,fit.inmodel,stats] = stepwisefit(x,lambda,'premove',0.3,'penter',0.1);
        fit.intercept = stats.intercept;
        fit.x = x;
        fit.clusters = clusters;
        statistics.pval = stats.pval;
        statistics.r2 = 1-stats.SSresid/stats.SStotal;
        statistics.adj_r2 = (1.0 - (1.0-statistics.r2)*((nStudents-1)/(nStudents-nClusters)));
    else
        intercept = varargin{1};
        coeffs = varargin{2};
        fit = intercept + x*coeffs;
        sst = sum((lambda - mean(lambda)).^2);
        sse = sum((fit - lambda).^2);
        statistics.r2 = 1-sse/sst;
        statistics.adj_r2 = (1.0 - (1.0-statistics.r2)*(nStudents/(nStudents-nClusters)));
    end

    display(strcat('Clusters: ',num2str(nClusters)));
    display(strcat('R2: ',num2str(statistics.r2)));
    display(strcat('ADJ_R2: ',num2str(statistics.adj_r2)));

    results.fit = fit;
    results.stats = statistics;

return
```

## 14.5 tgt_ normalize

```
% tgt_normalize - Performs a row normalization of matrix
function [new_matrix] = tgt_normalize(matrix)
    new_matrix = matrix;
    for i = 1:nrows(matrix)
        new_matrix(i,:) = matrix(i,:)/sum(matrix(i,:));
    end
end
```

# References

[1] Aleven, V., and Koedinger, K. R. An effective meta-cognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive Science 26* (2002), 147–179.

[2] Altendorf, E., Jorgensen, J., and Inc, C. Probabilistic relational models of on-line user behavior early explorations. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003).

[3] Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proceedings of the 11th International Conference on Acoustics, Speech, and Signal Processing* (1986).

[4] Baker, R. S., Corbett, A. T., and Koedinger, K. R. Detecting student misuse of intelligent tutoring systems. In *Proceedings of the 7th International Conference on International Tutoring Systems* (2004).

[5] Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. Learning a mahalanobis metric with side information. *Journal of Machine Learning Research* (2005).

[6] Bar-Hillel, A., Shental, N., and Weinshall, D. Learning distance functions using equivalence relations. In *In Proceedings of the Twentieth International Conference on Machine Learning* (2003).

[7] BASU, S. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKIDD International Conference on Knowledge Discovery and Data Mining* (2004), pp. 59–68.

[8] BASU, S., BANERJEE, A., AND MOONEY, R. J. Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning* (2002).

[9] BAUM, L. E., PETRIE, T., SOULES, G., AND WEISS, N. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics* (1970).

[10] BEAL, C. R., MITRA, S., AND COHEN, P. R. Modeling learning patterns of students with a tutoring system using hidden markov models. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education* (2007).

[11] BIE, T. D., MOMMA, M., AND CRISTIANINI, N. Efficiently learn the metric with side information. In *Lecture Notes in Artificial Intelligence* (2003), Springer, pp. 175–189.

[12] BILENKO, M., BASU, S., AND MOONEY, R. J. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning* (2004), pp. 81–88.

[13] BILENKO, M., AND MOONEY, R. J. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM*

*SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003).

[14] BLUM, A., AND KALAI, A. A note on learning from multiple-instance examples. *Machine Learning 30* (1998), 23–29.

[15] BLUM, A., AND MITCHELL, T. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory* (1998).

[16] BRAYDEN, T. H., POROPATIC, P. A., AND WATANABE, J. L. Iterative target testing for calculation of missing data points. *Analytical Chemistry 60* (1988), 1154–1158.

[17] COHN, D., CARUANA, R., AND MCCALLUM, A. Semi-supervised clustering with user feedback. Tech. rep., Cornell University, 2003.

[18] COOPER, M. M., SANDI-URENA, S., AND STEVENS, R. Reliable multi method assessment of metacognition use in chemistry problem solving. *Chemistry Education Research and Practice 9* (2008), 18–24.

[19] CYBENKO, G., AND CRESPI, V. Learning hidden markov models using nonnegative matrix factorization. *IEEE Transactions on Information Theory 57* (2011), 3963–3970.

[20] DANIEL HSU, SHAM M. KAKADE, T. Z. A spectral algorithm for learning hidden markov models. In *Proceedings of the 22nd Conference on Learning Theory* (2009).

[21] DAVIDSON, I., AND RAVI, S. S. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *Lecture Notes in Computer Science* (05), Springer, pp. 59–70.

[22] DAVIDSON, I., AND RAVI, S. S. Clustering with constraints: Feasibility issues and the k-means algorithm. In *SIAM International Conference on Data Mining* (2005).

[23] DEMIRIZ, A., BENNETT, K., AND EMBRECHTS, M. J. Semi-supervised clustering using genetic algorithms. In *In Artificial Neural Networks in Engineering (ANNIE-99* (1999), ASME Press, pp. 809–814.

[24] DIETTERICH, T. G., LATHROP, R. H., AND LOZANO-PEREZ, T. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence 89* (1997), 31–71.

[25] FLAVELL, J. H. Metacognitive aspects of problem solving. In *The nature of intelligence*. Hillsdale, NJ: Erlbaum, 1976.

[26] GETOOR, L., FRIEDMAN, N., KOLLER, D., PFEFFER, A., AND TASKAR, B. Probabilistic relational models. In *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

[27] GRAESSER, A., CHIPMAN, P., HAYNES, B. C., AND OLNEY, A. Autotutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions in Education 48* (2005), 612–618.

[28] JEBARA, T., SONG, Y., AND THADANI, K. Spectral clustering and embedding with hidden markov models. In *European Conference of Machine Learning* (2007).

[29] JEONG, H., GUPTA, A., ROSCOE, R., WAGSTER, J., BISWAS, G., AND SCHWARTZ, D. Using hidden markov models to characterize student behaviors in learning-by-teaching environments. In *Proceedings of the 9th international conference on Intelligent Tutoring Systems* (2008).

[30] KLEIN, D., KAMVAR, S., AND MANNING, C. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning* (2002), pp. 307–314.

[31] KOCH, A. Training in metacognition and comprehension of physics texts. *Science Education 85* (2001), 758–768.

[32] KULIS, B., BASU, S., DHILLON, I., AND MOONEY, R. Semi-supervised graph clustering: a kernel approach. In *Proceedings of the 22nd international conference on Machine learning* (New York, NY, USA, 2005), ACM, pp. 457–464.

[33] LANGE, T., LAW, M. H. C., JAIN, A. K., AND BUHMANN, J. M. Learning with constrained and unlabelled data. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), pp. 731–738.

[34] LAW, M. H. C., TOPCHY, A. P., AND JAIN, A. K. Model-based clustering with probabilistic constraints. In *SIAM International Conference on Data Mining* (2005).

[35] LIN, T.-H., MURPHY, R. F., AND BAR-JOSEPH, Z. Discriminative motif finding for predicting protein subcellular localization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2010).

[36] LONG, P. M., AND SERVEDIO, R. A. Random classification noise defeats all convex potential boosters. *Machine Learning 78* (2009), 287–304.

[37] LU, Z., AND LEEN, T. K. Semi-supervised learning with penalized probabilistic clustering. In *Advances in Neural Information Processing Systems 17* (2005).

[38] LUO, X., AND ZINCIR-HEYWOOD, A. N. Evaluation of two systems on multi-class multi-label document classification. In *Foundations of Intelligent Systems, 15th International Symposium* (2005), pp. 161–169.

[39] MALINOWSKI, E. R. *Factor Analysis in Chemistry*. Wiley-Interscience, 1980.

[40] MCLAREN, B. M., LIM, S.-J., GAGNON, F., YARON, D., AND KOEDINGER, K. R. Study the effects of personalized language and worked examples in the context of a web-based intelligent tutor. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (2006).

[41] MEYER-DELIUS, D., PLAGEMANN, C., VON WICHERT, G., FEITEN, W., LAWITZKY, G., AND BURGARD, W. A probabilistic relational model for characterizing situations in dynamic multi-agent systems. In *Proceedings of the 31st Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications* (2007).

[42] MOSSEL, E., AND ROCH, S. Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing* (2005).

[43] NG, A. Y., JORDAN, M. I., AND WEISS, Y. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems* (2001).

[44] PELLEG, D., AND BARAS, D. K-means with large and noisy constraint sets. In *Proceedings of the 18th European Conference on Machine Learning* (2007).

[45] PENG, W., AND LI, T. Music clustering with constraints. In *Proceedings of the 8th International Conference on Music Information Retrieval* (2007).

[46] PITT, L., AND VALIANT, L. Computational limitations on learning from examples. *Jounal of the ACM 35* (1988), 965–984.

[47] RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE* (1989), pp. 257–286.

[48] ROLL, I., ALEVEN, V., MCLAREN, B. M., RYU, E., BAKER, R. S., AND KOEDINGER, K. R. 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).

[49] ROMERO, C., AND VENTURA, S. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications 33* (2007), 135–146.

[50] SCHLIEP, A., SCHNHUTH, A., AND STEINHOFF, C. Using hidden markov models to analyze gene expression time course data. *Bioinformatics 19* (2003), 255–263.

[51] SCHOENFELD, A. H. Beyond the purely cognitive: Belief systems, social cognitions, and metacognitions as driving forces in intellectual performance. *Cognitive Science 7* (1986), 329–363.

[52] SHIH, B., KOEDINGER, K. R., AND SCHEINES, R. A response time model for bottom-out hints as worked examples. In *Proceedings of the 1st International Conference on Educational Data Mining* (2008).

[53] SHIH, B., KOEDINGER, K. R., AND SCHEINES, R. Discovery of learning tactics using hidden markov model clustering. In *Proceedings of the 3rd International Conference on Educational Data Mining* (2010).

[54] TSOUMAKAS, G., KATAKIS, I., AND VLAHAVAS, I. P. *Data Mining and Knowledge Discovery Handbook.* Springer, 2010, ch. Mining Multi-label Data.

[55] VANLEHN, K., LYNCH, C., SCHULZE, K., SHAPIRO, J. A., SHELBY, R. H., TAYLOR, L., TREACY, D. J., WEINSTEIN, A., AND WINTERSGILL, M. C. The andes physics tutoring system: Five years of evaluations. In *Proceedings of the Artificial Intelligence in Education Conference* (2005).

[56] WAGSTAFF, K., AND CARDIE, C. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning* (2000), pp. 1103–1110.

[57] WAGSTAFF, K., CARDIE, C., ROGERS, S., AND SCHROEDL, S. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning* (2001), Morgan Kaufmann, pp. 577–584.

[58] XING, E. P., NG, A. Y., JORDAN, M. I., AND RUSSELL, S. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15* (2002), MIT Press, pp. 505–512.

[59] YAN, R., ZHANG, J., YANG, J., AND HAUPTMANN, A. G. A discriminative learning framework with pairwise constraints for video object classification. *IEEE Transactions in Pattern Analysis and Machine Intelligence 28*, 4 (2006), 578.

[60] YANG, L., AND JIN, R. Bayesian active distance metric learning. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence* (2007).

[61] YANG, T., JIN, R., AND JAIN, A. K. Learning from noisy side information by generalized maximum entropy model. In *International Conference of Machine Learning 2010* (2010).

[62] ZHOU, Z.-H., JIANG, K., AND LI, M. Multi-instance learning based web mining. *Applied Intelligence 22* (2005), 135–147.

[63] ZHU, X., AND WU, X. Class noise vs attribute noise: A quantitative study of their impacts. *Artificial Intelligence Review 22* (2004), 177–210.