Mixture Modeling of Individual Learning Curves

Matthew Streeter
Duolingo, Inc.
Pittsburgh, PA
matt@duolingo.com

ABSTRACT

We show that student learning can be accurately modeled using a mixture of learning curves, each of which specifies error probability as a function of time. This approach generalizes Knowledge Tracing [7], which can be viewed as a mixture model in which the learning curves are step functions. We show that this generality yields order-of-magnitude improvements in prediction accuracy on real data. Furthermore, examination of the learning curves provides actionable insights into how different segments of the student population are learning.

To make our mixture model more expressive, we allow the learning curves to be defined by generalized linear models with arbitrary features. This approach generalizes Additive Factor Models [4] and Performance Factors Analysis [16], and outperforms them on a large, real world dataset.

1. INTRODUCTION

In the mid-1980s, a now-famous study demonstrated the potential impact of adaptive, personalized education: students tutored one-on-one outperformed those taught in a conventional classroom by two standard deviations [3]. Remarkably, subsequent research has achieved similar gains using interactive, computerized tutors that maintain an accurate model of the student's knowledge and skills [6]. In the past few years, widespread access to smartphones and the web has allowed such systems to be deployed on an unprecedented scale. Duolingo's personalized language courses have enrolled over 90 million students, more than the total number of students in all U.S. elementary and secondary schools combined.

A central component of an intelligent tutoring system is the student model, which infers a student's latent skills and knowledge from observed data. To make accurate inferences from the limited data available for a particular student, one must make assumptions about how students learn. How do students differ in their learning of a particular skill or concept? Is the primary difference in the initial error rate, the rate at which error decreases with time, the shape of the learning curve, or something else? The answers to these questions have implications for the choice of model class (e.g., Hidden Markov Model, logistic regression), as well as the choice of model parameters.

Previous approaches to student modeling typically make strong assumptions about the shape of each student's learning curve (i.e., the error rate as a function of the number of trials). Additive Factor Models [4] use the student and the number of trials as features in a logistic regression model, which implies a sigmoidal learning curve with the same steepness for each student, but different horizontal offset. Knowledge Tracing [7] is a two-state Hidden Markov Model where, conditioned on the trial t at which the student first transitions from not knowing the skill to mastering it, the learning curve is a step function.

In empirical studies, it has been observed that aggregate learning curves often follow a power law, a phenomenon so ubiquitous it has been called the *power law of practice* [13]. Later work suggested that, although error rates follow a power law when averaged over an entire population, individual learning curves are more accurately modeled by exponentials [10]. That is, the power law curve observed in aggregate data is actually a mixture of exponentials, with each student's data coming from one component of the mixture.

These observations led us to seek out a more general approach to student modeling, in which individual learning curves could be teased apart from aggregate data, without making strong assumptions about the shape of the curves. Such an approach has the potential not only to make the student model more accurate, but also to explain and summarize the data in a way that can produce actionable insights into the behavior of different subsets of the student population.

This work makes several contributions to student modeling. First, we present models of student learning that generalize several prominent existing models and that outperform them on real-world datasets from Duolingo. Second, we show how our models can be used to visualize student performance in a way that gives insights into how well an intelligent tutoring system "works", improving upon the population-level learning curve analysis that is typically used for this purpose [11]. Finally, by demonstrating that relatively simple mixture models can deliver these benefits, we hope to inspire further work on more sophisticated approaches that use mixture models as a building block.

1.1 Related Work

The problem of modeling student learning is multifaceted. In full generality it entails modeling a student's latent abilities, modeling how latent abilities relate to observed performance, and modeling how abilities change over time as a result of learning and forgetting. For an overview of various approaches to student modeling, see [5, 8].

This work focuses on the important subproblem of modeling error probability as a function of trial number for a particular task. Following the influential work of Corbett and Anderson [7], Knowledge Tracing has been used to solve this problem in many intelligent tutoring systems. Recent work has sought to overcome two limitations of the basic Knowledge Tracing model: its assumption that each observed data point requires the use of a single skill, and its assumption that model parameters are the same for all students. To address the first limitation, Additive Factor Models [4] and Performance Factors Analysis [16] use logistic regressions that include parameters for each skill involved in some trial. The second limitation has been addressed by adapting the basic Knowledge Tracing model to individual students, for example by fitting per-student odds multipliers [7], or by learning per-student initial mastery probabilities [14].

Our work seeks to address a third limitation of Knowledge Tracing: its strong assumptions about the shape of the learning curve. Following Knowledge Tracing, we first attempt to model performance on a task that requires only a single skill. In §4, we generalize this approach to obtain a mixture model that includes both Additive Factor Models and Performance Factors Analysis as special cases, and that outperforms both on a large, real-world dataset.

2. SINGLE-TASK MIXTURE MODEL

In this section we present a simple mixture model that is appropriate for use on datasets with a single task. This model is a viable alternative to the basic (non-individualized) version of Knowledge Tracing, and is useful for exploratory data analysis. In §4, we generalize this model to handle datasets with multiple tasks.

2.1 The Probabilistic Model

A student's performance on a task after T trials can be represented as an error vector $v \in \{0,1\}^T$, where $v_t = 1$ if the student made an error on trial t and is 0 otherwise. Thus a task, together with a distribution over students, defines a distribution over binary error vectors. In this work, we model this distribution as a mixture of K distributions, where each component of the mixture is a *learning curve*, or equivalently a product of Bernoulli distributions (one for each trial).

To formally define this model, define the probability of observing outcome $o \in \{0,1\}$ when sampling from a Bernoulli distribution with parameter p as

$$\mathcal{B}(p,o) = \begin{cases} p & o = 1 \\ 1-p & o = 0 \ . \end{cases}$$

A learning curve $q \in [0,1]^\infty$ specifies, for each trial t, the probability q_t that the student makes an error on trial t. The probability of the error vector v according to learning curve q is $\prod_t \mathcal{B}(q_t, v_t)$. A K-component mixture over learning curves is a set $q^1, q^2, \ldots, q^K$ of learning curves, together with prior probabilities $p^1, p^2, \ldots, p^K$. The probability of an error vector $v \in \{0,1\}^T$ according to the mixture model is

$$\sum_{j=1}^{K} p^j \prod_{t=1}^{T} \mathcal{B}(q_t^j, v_t) \ .$$

Inference in a mixture model consists of applying Bayes' rule to compute a posterior distribution over the K components of the mixture, given an observed error vector. The model parameters can be fit from data using the EM algorithm, pseudo code for which is given in Algorithm 1.

Algorithm 1 EM Algorithm for single-task mixture model

```
**Parameters:** number of components $K$, error vector $v^s$ for each student $s$, prior parameters $\alpha \geq 1$, $\beta \geq 1$.
**Initialize** $p^j \leftarrow \frac{1}{k}\ \forall j$, and $q_t^j \leftarrow \mathrm{Rand}(0,1)\ \forall j, t$.
**while** not converged **do**
    $L_{s,j} \leftarrow p^j \prod_{t=1}^{T} \mathcal{B}(q_t^j, v_t^s)\ \forall s, j$
    $z_{s,j} \leftarrow \frac{L_{s,j}}{\sum_{j'} L_{s,j'}}\ \forall s, j$
    $q_t^j \leftarrow \frac{\alpha - 1 + \sum_s z_{s,j} v_j^s}{\alpha + \beta - 2 + \sum_s z_{s,j}}\ \forall j, t$
    $p^j \leftarrow \frac{\sum_s z_{s,j}}{\sum_s \sum_{j'} z_{s,j'}}\ \forall j$
**end while**
```

To make Algorithm 1 perform well when data is sparse, it is useful to place a Bayesian prior over the set of possible learning curves. In this work we use a product of Beta distributions for the prior: $\mathbb{P}[q] = \prod_t \mathrm{Beta}(\alpha, \beta)(\mathrm{q_t})$. This choice of prior gives a simple closed form for the maximization step of the EM algorithm, which can be thought of computing the maximum-likelihood estimate of q_t^j after "hallucinating" $\alpha - 1$ correct responses and $\beta - 1$ errors (see pseudo code).

2.2 Knowledge Tracing as a Mixture Model

Knowledge Tracing is typically presented as a two-state Hidden Markov Model, where the student's state indicates whether or not they have mastered a particular skill. In this section, we show that if the maximum number of trials is T, Knowledge Tracing can also be thought of as a mixture model with $T + 1$ components, each of which is a step function. Thus, Knowledge Tracing can be viewed as a constrained mixture model, in contrast to the unconstrained model discussed in the previous section.

To see this relationship, recall that in a Knowledge Tracing model, the student makes an error with slip probability p_s if they have mastered the skill, and with probability $1 - p_g$ otherwise, where p_g is the probability of a correct guess. The probability of mastery is p_0 initially, and after each trial, a student who has not yet mastered the skill transitions to the mastered state with probability p_T.

Let V be an error vector, so $V_t = 1$ if the student makes an error on trial t and is 0 otherwise, and let M be the state vector: $M_t = 1$ if the student has mastered the skill at the beginning of trial t and is 0 otherwise. The distribution over

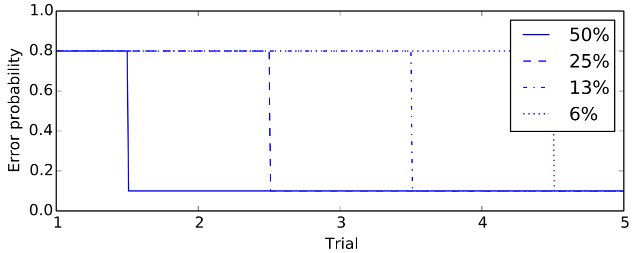

Figure 1: Mixture model representation of a Knowledge Tracing model with guess probability $p_g = 0.2$, slip probability $p_s = 0.1$, transition probability $p_T = 0.5$, and initial mastery probability $p_0 = 0$.

error vectors defined by Knowledge Tracing is given by

$$\mathbb{P}[V = v] = \sum_m \mathbb{P}[M = m]\mathbb{P}[V = v|M = m] \ .$$

Because the student never leaves the mastered state after reaching it, there are only $T + 1$ possibilities for the state vector M. Letting m^j be the jth possibility ($m_t^j = 0$ if $t < j$, 1 otherwise), and letting $p^j = \mathbb{P}[M = m^j]$, we have

$$\mathbb{P}[V = v] = \sum_{j=1}^{T+1} p^j \cdot \mathbb{P}[V = v|M = m^j] \ .$$

Because the components of V are conditionally independent given M,

$$\mathbb{P}[V = v|M = m^j] = \prod_{t=1}^{T} \mathcal{B}(q_t^j, v_t)$$

where

$$q_t^j = \begin{cases} 1 - p_g & t < j \\ p_s & t \geq j \ . \end{cases}$$

Putting these facts together, we see that the probability of a particular error vector under Knowledge Tracing is the same as under a mixture model with $T+1$ components, where each learning curve q^j is a step function with the same initial and final height but a different horizontal offset (see Figure 1).

Because the HMM and the mixture model are both generative models that specify the same distribution over binary vectors, the conditional distributions over binary vectors given a sequence of observations are also the same, and Bayesian inference yields exactly the same predictions when performed on either model.

Viewing Knowledge Tracing in this way, it is natural to consider generalizations that remove some of the constraints, for example allowing the step functions to have different initial or final heights (perhaps students who master the skill earlier are less likely to slip later on). In the model presented in §2.1 we simply remove all the constraints, allowing us to fit a mixture model over learning curves of arbitrary shape.

We note that later work on Knowledge Tracing allowed for the possibility of forgetting (transitioning from the mastered to unmastered state). This version can still be modeled as a mixture model, but with 2^T rather than $T + 1$ components.

2.3 Statistical Consistency

A model is *statistically consistent* if, given enough data, it converges to the ground truth. In this section we show that the "hard" version of EM algorithm 1 is consistent, provided the number of components in the mixture model grows with the amount of available data (the hard EM algorithm is the same as algorithm 1, except that it sets $z_{s,j} = 1$ for the j that maximizes $L_{s,j}$, and $z_{s,j} = 0$ otherwise). For simplicity we assume the number of trials T is the same for all students, but this is not essential. Also, though the data requirements suggested by this analysis are exponential T, in practice we find that near-optimal predictions are obtained using a much smaller number of components.

THEOREM 1. *Consider the "hard" version of EM algorithm 1, and suppose that the number of trials is T for all students. This algorithm is statistically consistent, provided the number of curves K in the mixture model grows as a function of the number of data points n.*

PROOF. Recall that an event occurs *with high probability* (whp) if, as $n \to \infty$, the probability of the event approaches 1. The idea of the proof is to show that, whp, each of the 2^T possible error vectors will be placed into its own cluster on the first iteration of the EM algorithm. This will imply that the EM algorithm converges on the first iteration to a mixture model that is close to the true distribution.

Consider a particular error vector $v^s \in \{0, 1\}^T$, and let j be the index of the likelihood-maximizing curve on the first iteration of the algorithm (i.e., $z_{s,j} = 1$). If $Q \in [0, 1]^T$ is a random curve, the probability that $\prod_{t=1}^T \mathcal{B}(Q_t, v_t^s) > \frac{1}{2}$ is positive. Thus, as $K \to \infty$, whp at least one of the K random curves will satisfy this inequality, and in particular for the likelihood-maximizing curve q^j we have $\prod_{t=1}^T \mathcal{B}(q_t^j, v_t^s) > \frac{1}{2}$, which implies $\mathcal{B}(q_t^j, v_t^s) > \frac{1}{2}$ for all t. For any error vector $v^{s'} \neq v^s$, there must be some t such that $v_t^s \neq v_t^{s'}$, which implies $\mathcal{B}(q_t^j, v_t^{s'}) < \frac{1}{2}$. This means that whp, q^j cannot be the likelihood-maximizing curve for $v^{s'}$, and so each binary vector will have a unique likelihood-maximizing curve.

If each binary vector v has a unique likelihood-maximizing curve q^j, then the M step of the algorithm will simply set $q^j \leftarrow v$, and will set p^j to the empirical frequency of v within the dataset. As $n \to \infty$, this empirical frequency approaches the true probability, which shows that the algorithm is consistent. □

In the worst case, statistical consistency requires a constant amount of data for every possible error vector, hence the data requirements grow exponentially with T. However, this is not as bad as it may seem. In intelligent tutoring systems, it is often the case that T is small enough that even in the worst case we can guarantee near-optimal performance. Furthermore, as we show experimentally in §3.2, near-optimal performance can often be achieved with a much smaller number of components in practice.

2.4 Use in an Intelligent Tutoring System

How should the predictions of a mixture model be used to schedule practice within an intelligent tutoring system? When using Knowledge Tracing, a typical approach is to schedule practice for a skill until the inferred probability of having mastered it exceeds some threshold such as 0.95. With a mixture model, we can no longer take this approach since we don't make explicit predictions about whether the student has mastered a skill. Nevertheless, we can define a reasonable practice scheduling rule in terms of predicted future performance.

In particular, note that another way of formulating the scheduling rule typically used in Knowledge Tracing is to say that we stop practice once we are 95% confident that performance has reached an asymptote. With a mixture model, it is unlikely that the marginal value of practice will be exactly 0, so this precise rule is unlikely to work well (it would simply schedule indefinite practice). However, we can compute the expected marginal benefit of practice (in terms of reduction in error rate), and stop scheduling practice once this drops below some threshold.

Note that when practice scheduling is defined in terms of expected marginal benefit, the practice schedule is a function of the predicted distribution over error vectors, so mixture models that make the same predictions will result in the same practice schedule even if the model parameters are different. This is in contrast to Knowledge Tracing, where multiple globally optimal models (in terms of likelihood) can lead to very different practice schedules, because the inferred probability of mastery can be different even for two models that make identical predictions [2].

2.5 Identifiability

A statistical model is identifiable if there is a unique set of parameters that maximize likelihood. Our mixture model is not identifiable, since in general there are many ways to express a given distribution over binary vectors as a mixture of learning curves. However, as we argued in the previous section, non-identifiability does not pose a problem for practice scheduling if the schedule is defined in terms of the model's predictions rather than its parameters.

3. EXPERIMENTS WITH SINGLE-TASK MODEL

In this section we evaluate the single-task mixture model of §2 on data from Duolingo. These experiments serve two purposes. First, they show that the mixture model can give much more accurate predictions than Knowledge Tracing on real data. Second, inspection of the learning curves produced by the mixture model reveals interesting facts about the student population that are not apparent from conventional learning curve analysis. In §4 we present a more general mixture model that is appropriate for datasets with multiple skills.

3.1 The Duolingo Dataset

We collected log data from Duolingo, a free language learning application with over 90 million students. Students who use Duolingo progress through a sequence of lessons, each of which takes a few minutes to complete and teaches certain words and grammatical concepts. Within each lesson, the student is asked to solve a sequence of self-contained challenges, which can be of various types. For example, a student learning Spanish may be asked to translate a Spanish sentence into English, or to determine which of several possible translations of an English sentence into Spanish is correct.

For these experiments, we focus on *listen challenges*, in which the student listens to a recording of a sentence spoken in the language they are learning, then types what they hear. Listen challenges are attractive because, unlike challenges which involve translating a sentence, there is only one correct answer, which simplifies error attribution. For these experiments we use a simple bag-of-words knowledge component (KC) model. There is one KC for each word in the correct answer, and a KC is marked correct if it appears among the words the student typed. For example, if a student learning English hears the spoken sentence "I have a business card" and types "I have a business car", we would mark the KC *card* as incorrect, while marking the KCs for the other four words correct. This approach is not perfect because it ignores word order as well as the effects of context (students may be able to infer which word is being said from context clues, even if they cannot in general recognize the word when spoken). However, the learning curves generated by this KC model are smooth and monotonically decreasing, suggesting that it performs reasonably well.

Our experiments use data from the Spanish course for English speakers, one of the most popular courses on Duolingo. In this section, we focus on modeling acquisition of a single skill, using data for the KC *una* (the feminine version of the indefinite article "a"). In §4 we consider more general mixture models, and in §5 we evaluate them on datasets with multiple KCs. The full dataset has roughly 700,000 data points (there is one data point for each combination of student, trial, and KC), while the *una* dataset contains around 15,000.

3.2 Prediction Accuracy

To evaluate the mixture model's prediction accuracy, we divided the Duolingo dataset into equal-sized training and test sets by assigning each student to one of the two groups at random. We then ran the EM algorithm on the training data to fit mixture models with various numbers of components, as well as a Knowledge Tracing model, and computed the predictions of these models on the test data. We evaluate prediction accuracy using two commonly-used metrics.

1. *Average log-likelihood.* Log-likelihood measures how probable the test data is according to the model. Specifically, if the dataset D consists of n independent data points $D_1, D_2, \dots, D_n$ (each data point is the binary performance of a particular student on a particular trial), and $p_i = \mathbb{P}[D_i|M]$ is the conditional probability of the ith data point D_i given the model M, then

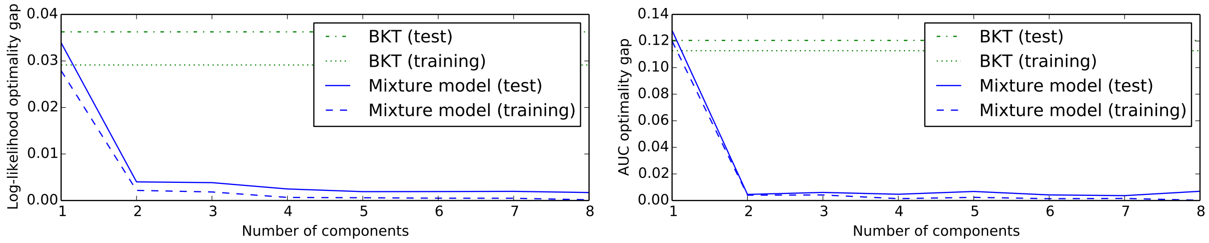

Figure 2: Optimality gaps for log likelihood (left) and AUC (right) as a function of number of components in the mixture model, compared to Knowledge Tracing (horizontal lines). The optimality gap is the absolute difference between the model's accuracy and the maximum possible accuracy on the dataset.

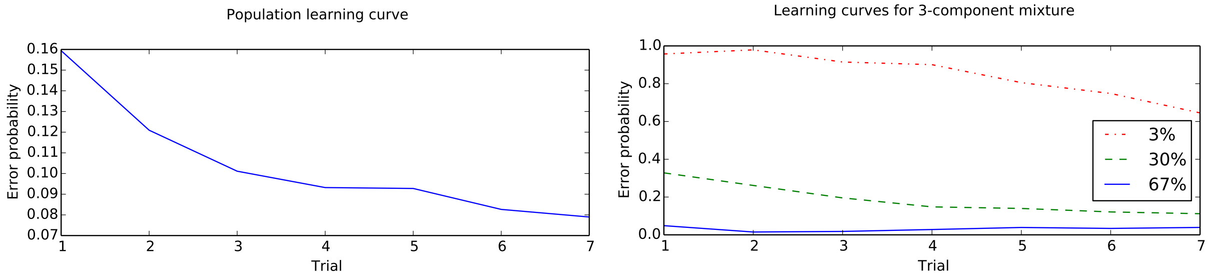

Figure 3: Learning curves for recognizing the Spanish word *una* in a Duolingo listen challenge. The population curve (left) suggests a reasonable rate of learning in aggregate, but the mixture model (right) reveals large differences among different clusters of students.

average log-likelihood is

$$\frac{1}{n}\log \mathbb{P}[D|M] = \frac{1}{n}\log \prod_{i=1}^{n} p_i = \frac{1}{n}\sum_{i=1}^{n} \log p_i \ .$$

Because both the mixture model and Knowledge Tracing are fit using maximum likelihood, it is natural to compare them in terms of this objective function.

2. *AUC.* AUC evaluates the accuracy of the model's predictions when they are converted from probabilities to binary values by applying a threshold. It can be defined as the probability that $p > q$, where p is the model's prediction for a randomly-selected positive example and q is the model's prediction for a randomly-selected negative example. This is equivalent to the area under the ROC curve, which plots true positive rate against false positive rate (both of which vary as a function of the chosen threshold).

Figure 2 presents accuracy on the *una* dataset as a function of the number of components in the mixture model, both on training and held-out test data. To make relative improvements clearer, we plot the optimality gap rather than the raw value of the prediction accuracy metric. For example, the optimality gap for test set log likelihood is the difference between the optimal log likelihood on the test data (which can be computed in closed form) and the model's log likelihood on the test data.

For both AUC and log-likelihood, the improvement in accuracy is largest when going from one component to two, and there are diminishing returns to additional components, particularly in terms of performance on held-out test data. With more than 5 components, log-likelihood on test data gets slightly worse due to overfitting, while performance on training data improves slightly. In practice, the number of components can be selected using cross-validation.

For both metrics, Knowledge Tracing is similar to the one-component model but significantly worse than the two component model in terms of accuracy, both on training and test data. Furthermore, all mixture models with two or more components outperform Knowledge Tracing by an *order of magnitude* in terms of the optimality gap for log-likelihood and AUC, both on training and on held-out test data. We observed very similar results for datasets based on other Spanish words, such as *come* (eat), *mujer* (woman), and *hombre* (man).

3.3 Learning Curve Mixture Analysis

In this section we examine the learning curves that make up the components of the mixture model fit to Duolingo data. This analysis can be viewed as a more general version of learning curve analysis [11], which examines the population learning curve (this is equivalent to the curve for a one-component mixture model).

Figure 3 presents learning curves for the *una* dataset. The left pane of the figure shows the aggregate learning curve, while the right pane shows the curves for a 3-component mixture model fit using the EM algorithm. Examining the right pane, we see that the mixture model clusters students into three quite different groups.

- Around two-thirds of the students belong to a cluster that in aggregate has an error probability around 5% on the first trial, and this error rate does not change with increased trials.
- A second, smaller cluster contains 30% of the students. These students, in aggregate, have an initial error rate of 33% which decreases to around 11% after 7 trials.
- The third cluster contains only 3% of students. These students have a very high initial error rate of 96%, which declines to about 65% after 7 trials.

The existence of this third, high-error-rate cluster surprised us, so we went back to the log data to examine the behavior of students in this cluster in more detail. It turned out that almost all of these students were simply giving up when presented with a listen challenge (although they correctly answered other types of challenges). Further examination of the log data revealed that some of these students skipped all listen challenges, while others would skip all listen challenges for long stretches of time, then at other times would correctly answer listen challenges. We conjecture that the former set of students are either hearing-impaired or do not have working speakers, while the latter do not want to turn their speakers on at certain times, for example because they are in a public place. Duolingo attempts to accommodate such students by offering a setting that disables listen challenges, but not all students realize this is available. As a result of these insights, Duolingo is now exploring user interface changes that will actively detect students that fall into this cluster and make it easier for them to temporarily disable listen challenges.

This analysis shows how mixture modeling can produce valuable insights that are not apparent from examination of the population learning curve alone. We hope this will inspire the use of mixture modeling more broadly as a general-purpose diagnostic tool for intelligent tutoring systems.

4. GENERAL MIXTURE MODEL

The single-task model is appropriate for datasets where there is a single knowledge component (KC) and many students. In an actual intelligent tutoring system, a student will learn many KCs, and prediction accuracy can improved by using student performance on one KC to help predict performance on other, not yet seen KCs. In this section we present a more general mixture model that accomplishes this.

In this more general model, student performance is again modeled as a mixture of K learning curves. However, instead of treating each point on the learning curve as a separate parameter, we let it be the output of a generalized linear model with features that depend on the student, task, and trial number. In particular, for a student s and task i, the probability of a performance vector $v_1, v_2, \ldots, v_T$ is

$$\sum_{j=1}^{k} p^j \prod_{t=1}^{T} \mathcal{B}(q^j(s,i,t;\beta^j), v_t)$$

where

$$q^j(s,i,t;\beta^j) = g^{-1}(\phi_{s,i,t} \cdot \beta^j),$$

where $\phi_{s,i,t}$ is the feature vector for student s, task i, trial t, and g is the link function for the generalized linear model [12]. Our experiments use logistic regression, for which the link function is $g(p) = \text{logit}(p)$.

Note that this model generalizes the single-task mixture model presented in §2. In particular, the single-task model with curve $q^j(t)$ is recovered by setting $\phi_{s,i,t} = e_t$, an indicator vector for trial t, and setting $\beta^j_t = g(q_j(t))$.

As with the single-task model, we can estimate the parameters of this model using the EM algorithm. The main difference is that the maximization step no longer has a closed form solution. However, it is a convex optimization and can still be solved exactly using a number of algorithms, for example stochastic gradient descent.

To define the EM algorithm, first define the likelihood function

$$L^j_{s,i}(\beta) = \prod_{t=1}^{T} \mathcal{B}(q^j(s,i,t;\beta), v_t) \ .$$

For the E step, we define hidden variables $z^j_{s,i}$, which give the probability that the data for student s and task i follows curve j.

$$z^j_{s,i} = \frac{p^j L^j_{s,i}}{\sum_{j'} p^{j'} L^{j'}_{s,i}(\beta)} \ .$$

For the M step, we optimize the coefficient vector for each component j so as to maximize expected log-likelihood.

$$\beta^j = \text{argmax}_\beta \left\{ \sum_s \sum_i z^j_{s,i} \log(L^j_{s,i}(\beta)) \right\} \ .$$

When performing inference for a new student, we solve a similar optimization problem, but we only update the coefficients for that particular student.

4.1 Relationship to Other Models

This mixture model is quite general, and with appropriate choices for the feature function ϕ can recover many previously-studied models. In particular, any modeling approach that is based on a logistic regression using features that depend only on the student, task, and trial number can be recovered by using a single component ($K = 1$), choosing g = logit, and defining ϕ to include the appropriate features. This includes both Additive Factor Models [4] and Performance Factors Analysis [16]. By choosing a larger K, we immediately obtain generalizations of each of these methods that have the potential to more accurately model the behavior of individual clusters of students. Because the trial number (together with the student and task) identifies a unique learning event, we can also include features that depend on the trial type, elapsed time, and previous learning history, as in learning decomposition [1].

Note that for the mixture model to add value over a simple regression, we must define "task" in such a way that we observe multiple trials for a given (student, task) pair. For datasets where each item requires the use of multiple KCs,

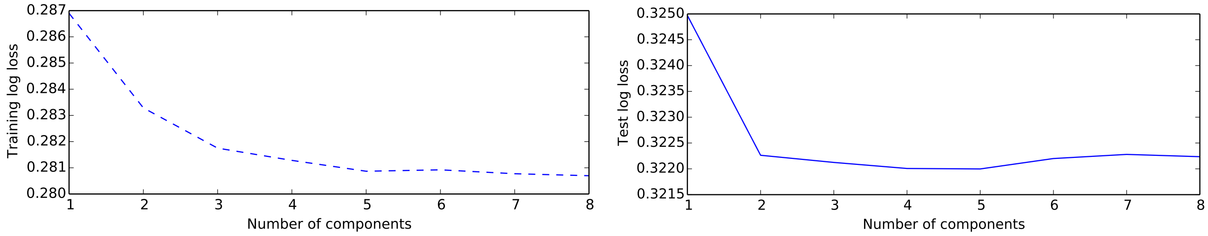

Figure 4: Performance of a mixture of Additive Factor Models on training data (left) and test data (right), as a function of the number of components in the mixture model.

Table 1: Performance on Duolingo dataset

Method	Training log loss	Test log loss	Training AUC loss	Test AUC loss
Knowledge Tracing	0.3429	0.3441	0.3406	0.3460
Performance Factors Analysis	0.3248	0.3285	0.2774	0.2865
Additive Factor Model	0.2869	0.3250	0.1629	0.2789
A.F.M. Mixture (3 components)	**0.2818**	**0.3220**	**0.1598**	**0.2760**

this entails either (a) defining a task for each combination of KCs, or (b) using error attribution to create a dataset in which each example involves only a single KC, and having one task per KC. We use the latter approach in our experiments in §5. This approach is different from the one taken by algorithms such as LR-DBN [17], which make predictions on multiple-KC items directly.

4.2 Parameter Sharing

To make more efficient use of available data when fitting this generalized mixture model, it can be useful for certain coefficient values to be shared across components of the mixture model. To illustrate this issue, consider fitting a mixture of Additive Factor Models. In this case, ϕ includes an indicator feature for each student. If we fit a K component mixture, we must estimate K separate coefficient values for each student, which increases the variance of the estimates compared to the basic Additive Factor Model. For students for whom we do not yet have much data, this can result in larger values of K giving worse performance.

To overcome this difficulty, we allow certain coefficients to be shared across all components of the mixture model, while others have a separate value for each component. This requires only minor changes to the M step of the EM algorithm. Instead of solving K separate optimization problems, we solve a single larger optimization problem of the form:

$$\text{argmax}_{\beta^1,\beta^2,\ldots,\beta^j} \left\{ \sum_j \sum_s \sum_i Z^j_{s,i} \log(L^j_{s,i}(\beta^j)) \right\}$$

subject to

$$\beta^1_z = \beta^2_z = \ldots = \beta^j_z \text{ for all shared } z.$$

Again, for g = logit, this is a weighted logistic regression problem that can be solved using a variety of standard algorithms.

5. EXPERIMENTS WITH GENERALIZED MODEL

In this section, we demonstrate the potential of the generalized mixture model by using it to learn a mixture of Additive Factor Models which models student performance on Duolingo listen challenges.

For these experiments, we use the same Duolingo dataset described in §3.1, but with all knowledge components included (i.e., every time student s completes a listen challenge, there is an example for each word w in the challenge, and the label for the example indicates whether the student included word w in their response). Each KC (i.e., each word) is considered a separate task. Note that although each listen challenge involves multiple KCs, we are using error attribution to create a dataset in which each example involves only a single KC. There is nothing about our methodology that requires this, but it mirrors the problem we wish to solve at Duolingo, and also allows for a cleaner comparison with Knowledge Tracing.

When splitting the data into training and test sets, we put each (student, KC) pair into one of the two groups uniformly at random. When fitting a mixture of Additive Factor Models, we use parameter sharing (see §4.2) for the student and KC indicator features, while allowing the times-seen feature to vary across components.

Figure 4 shows how performance on training and test data varies as a function of the number of components in the mixture model. The leftmost point ($K = 1$) corresponds to a regular Additive Factor Model, which can be fit by running a single logistic regression. Other points correspond to mixture models fit using the EM algorithm, in which each iteration entails solving a weighted logistic regression problem. As can be seen, using more than one component in the mixture model improves accuracy on both training and held-out test data.

Table 1 compares the performance of the Additive Factor

Model, the 3-component mixture of Additive Factor Models, Knowledge Tracing, and Performance Factors Analysis [16] on the same dataset. In this table, we present accuracy in terms of losses (log loss is -1 times log-likelihood, while AUC loss is one minus AUC), so lower values are better. As can be seen, the 3-component mixture gives the best performance of all the methods we considered in terms of both metrics, both on training and test data.

6. CONCLUSIONS

In this work we explored the use of mixture models to predict how students' error rates change as they learn. This led to order-of-magnitude improvements over Knowledge Tracing in terms of prediction accuracy on single-task datasets from Duolingo, as measured by the optimality gaps for both log-likelihood and AUC. Furthermore, examining the curves in the mixture model led us to uncover surprising facts about different groups of students.

We then generalized this mixture model to the multi-task setting, by learning a mixture of generalized linear models. This generalized mixture model offered state of the art performance on a large Duolingo dataset, outperforming Performance Factors Analysis, Additive Factor Models, and Knowledge Tracing on the same data.

There are several ways in which this work could be extended:

1. *Finding a good prior over learning curves.* In the single-task setting, we simply placed a Beta prior over each point on each learning curve. Though this worked well on the Duolingo dataset we considered (which contained around 15,000 data points), it may not give the best bias/variance tradeoff for smaller datasets. A natural way to constrain the algorithm would be to require error probability to be non-increasing as a function of trial number. Restricting to a particular family of curves such as exponentials or APEX functions [10], which generalize power laws and exponentials, may also be reasonable.

2. *Accounting for forgetting.* We have assumed that performance depends only on the trial number, and not on the amount of time elapsed since a particular knowledge component was last seen. For this reason, our model has no way to capture the benefit of spaced repetition [9] over massed practice, which is important for practice scheduling in the context of language learning [15].

3. *Feature exploration in the multi-task setting.* The generalized mixture model from §4 can be used with any set of features ϕ, but our experiments in §4 considered only a few possible choices. It would be interesting to explore other feature sets, and to see whether the features that work best in the usual regression setting ($K = 1$) are also best for larger K.

7. REFERENCES

[1] J. E. Beck. Using learning decomposition to analyze student fluency development. In *ITS 2006*, pages 21–28, 2006.

[2] J. E. Beck and K. Chang. Identifiability: A fundamental problem of student modeling. In *User Modeling 2007*, pages 137–146. Springer, 2007.

[3] B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, pages 4–16, 1984.

[4] H. Cen, K. Koedinger, and B. Junker. Learning Factors Analysis – A general method for cognitive model evaluation and improvement. In *Intelligent Tutoring Systems*, pages 164–175. Springer, 2006.

[5] K. Chrysafiadi and M. Virvou. Student modeling approaches: A literature review for the last decade. *Expert Systems with Applications*, 40(11):4715–4729, 2013.

[6] A. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *User Modeling 2001*, pages 137–147. Springer, 2001.

[7] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[8] M. C. Desmarais and R. S. J. d Baker. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, 22(1-2):9–38, 2012.

[9] H. Ebbinghaus. *Memory: A contribution to experimental psychology.* Dover, New York, 1885.

[10] A. Heathcote, S. Brown, and D. Mewhort. The power law repealed: The case for an exponential law of practice. *Psychonomic Bulletin & Review*, 7(2):185–207, 2000.

[11] B. Martin, A. Mitrovic, K. R. Koedinger, and S. Mathan. Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Interaction*, 21(3):249–283, 2011.

[12] P. McCullagh, J. A. Nelder, and P. McCullagh. *Generalized linear models.* Chapman and Hall London, 1989.

[13] A. Newell and P. S. Rosenbloom. Mechanisms of skill acquisition and the law of practice. In J. R. Anderson, editor, *Cognitive skills and their acquisition*, pages 1–55. Erlbaum, Hillsdale, NJ, 1983.

[14] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a Bayesian networks implementation of knowledge tracing. In *User Modeling, Adaptation, and Personalization*, pages 255–266. Springer, 2010.

[15] P. I. Pavlik and J. R. Anderson. Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied*, 14(2):101, 2008.

[16] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger. Performance factors analysis – A new alternative to knowledge tracing. In *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, pages 531–538, 2009.

[17] Y. Xu and J. Mostow. Comparison of methods to trace multiple subskills: Is LR-DBN best? In *EDM*, pages 41–48, 2012.