Machine Learning Techniques for Personalized Learning

by

Andrew S. Lan

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

Richard G. Baraniuk, Chair
Victor E. Cameron Professor of Electrical and Computer Engineering

Ashok Veeraraghavan
Assistant Professor of Electrical and Computer Engineering

Anshumali Shrivastava
Assistant Professor of Computer Science

Houston, Texas

March, 2016
ABSTRACT

Machine Learning Techniques for Personalized Learning

by

Andrew S. Lan

Recent developments in personalized learning, powered by recent advances in machine learning and big data, have the potential to revamp the “one-size-fits-all” approach in today’s education by delivering a fully personalized learning experience for each student. The key behind these developments is to create a personalized learning system (PLS), which can automatically deliver analytics and feedback on the students’ progress and recommend learning actions for the students to take. A PLS presents a scalable approach to personalized learning by analyzing the data students generate while interacting with learning resources (i.e., textbook sections, lecture videos, assessment questions, etc). Such an approach relies on only minimal human effort and has the ability to scale to applications with millions of students, thousands of learning resources, and hundreds of domains. In this thesis, we develop a series of machine learning techniques for personalized learning, building on our previous work on sparse factor analysis (SPARFA) for learning and content analytics.

To begin with, we develop a new, nonlinear latent variable model that we call the dealbreaker model, in which a student’s success probability is determined by their weakest concept mastery. We develop efficient parameter inference algorithms for this model using novel methods for nonconvex optimization. We demonstrate that the
dealbreaker model excels at prediction and the parameters learned are interpretable: they provide key insights into which concepts are critical (i.e., the “dealbreakers”) to answering a question correctly. We also apply the dealbreaker model to a movie rating dataset, illustrating its broad applicability to applications other than education.

Then, we propose SPARFA-Trace, a new framework for time-varying learning and content analytics. We develop a novel message passing-based, blind, approximate Kalman filtering and smoothing algorithm for SPARFA that jointly traces student concept knowledge evolution over time, analyzes student concept knowledge state transitions (induced by studying learning resources, such as textbook sections, lecture videos, etc., or the forgetting effect), and estimates the content organization and difficulty of the questions in assessments. These quantities are estimated solely from binary-valued (correct/incorrect) graded student response data and the specific actions each student performs (e.g., answering a question or studying a learning resource) at each time instant.

Additionally, we study the problem of automatic grading and feedback generation for the kinds of open response mathematical questions that figure prominently in STEM (science, technology, engineering, and mathematics) courses. Our data-driven framework for mathematical language processing (MLP) leverages solution data from a large number of students to evaluate the correctness of their solutions, assign partial-credit scores, and provide feedback to each student on the likely locations of any errors. MLP takes inspiration from the success of natural language processing for text data and comprises three main steps. First, we convert each solution to an open response mathematical question into a series of numerical features. Second, we cluster the features from several solutions to uncover the structures of correct, partially correct, and incorrect solutions. We develop two different clustering approaches, one that leverages generic clustering algorithms and one based on Bayesian nonparametrics. Third, we automatically grade the remaining (potentially large number of) solutions based on their assigned cluster and one instructor-provided grade per cluster. As a
bonus, we can track the cluster assignment of each step of a multistep solution and determine when it departs from a cluster of correct solutions, which enables us to indicate the likely locations of errors to students.

Furthermore, we study the problem of selecting the best personalized learning action that each student should take next given their learning history; actions could include reading a textbook section, watching a lecture video, interacting with a simulation or lab, solving a practice question, and so on. We first estimate each student’s knowledge profile from their binary-valued graded responses to questions in their previous assessments. We then employ these knowledge profiles as contexts in the contextual (multi-armed) bandits framework to learn a policy that selects the personalized learning actions that maximize each student’s immediate success, i.e., their performance on their next assessment. We develop three algorithms for personalized learning action selection. While one is mainly of theoretical interest, we experimentally validate the other two using real-world educational datasets.

Our proposed set of models and algorithms comprise the basic and most essential components of a PLS, i.e., learning analytics, content analytics, grading and feedback, and scheduling.
Acknowledgement

First of all, I would like to thank my advisor, Richard Baraniuk, for his inspiring advices and motivating guidance throughout this period, without which this work could not be done. His insights on personalized learning is of paramount importance; His enthusiasm and thorough approach towards research has inspired me throughout my studies. But most importantly, I would like to thank him for bringing out the fun in daily work, which has the magic to turn tedious work into fun explorations. The spirit of SPARFA will live on forever!

I would also like to thank my thesis committee members, Ashok Veeraraghavan and Anshumali Shrivastava, and also other faculties at Rice, Genevera Allen and Wotao Yin, for insightful suggestions on my work.

Then I would also like to thank my collaborators, Christoph Studer, Andrew Waters, Divyanshu Vats, Thomas Goldstein, Ryan Ning, and Phillip Grimaldi for countless insightful comments, helpful discussions and alcohol. Also thanks to my ex-officemate Amirali Aghazadeh for a fantastic working environment during most of my time at Rice.


I would also thank all my other friends for their support.

Most importantly, I want to thank my parents, Jiang Lan and Beiyan Zheng, for everything.
If interested, please see our website www.sparfa.com, where you can learn more about our work and purchase SPARFA t-shirts and other merchandise.
# Contents

Abstract ii  
List of Illustrations xi  
List of Tables xv  

1 Introduction 1  
1.1 The Promise of Personalized Learning 1  
1.2 Thesis Organization 2  

2 Background: Sparse Factor Analysis for Learning and Content Analytics 4  

3 Dealbreaker: A Nonlinear Latent Variable Model for Educational Data 9  
3.1 Introduction 9  
3.1.1 Limits of affine student–response models 10  
3.1.2 Contributions 11  
3.2 The Dealbreaker Model 11  
3.3 Inference for the Hard Dealbreaker Model 14  
3.3.1 ADMM algorithm 14  
3.3.2 Proximal operators 16  
3.4 Inference for the Soft Dealbreaker Model 19  
3.5 Experiments 22  
3.5.1 Predicting unobserved student responses 22
3.5.2 Visualizing the dealbreaker model ............................................. 25
3.5.3 Interpreting movie ratings ....................................................... 27
3.6 Conclusions ............................................................................. 28

4 Time-varying Learning and Content Analytics via Sparse Factor Analysis

4.1 Introduction ............................................................................ 30
  4.1.1 SPARFA-Trace: Time-varying learning and content analytics .... 30
  4.1.2 Related work ........................................................................ 32
4.2 Statistical Model for Time-Varying Learning and Content Analytics . 34
  4.2.1 Statistical model for time-varying graded student responses to
       questions ................................................................................. 34
  4.2.2 Statistical model for student knowledge state transitions .......... 36
4.3 Time-Varying Learning Analytics .............................................. 38
  4.3.1 Kalman filtering ..................................................................... 39
  4.3.2 Kalman smoothing ............................................................... 42
  4.3.3 Approximate Kalman filtering for student concept knowledge
       tracing ...................................................................................... 44
4.4 Content Analytics .................................................................... 47
  4.4.1 SPARFA-Trace: An EM algorithm for parameter estimation ... 47
  4.4.2 Estimating the initial student knowledge parameters ............... 48
  4.4.3 Estimating the student concept knowledge state transition
       parameters ................................................................................ 48
  4.4.4 Estimating the question-dependent parameters ....................... 51
4.5 Experimental Results .............................................................. 53
  4.5.1 Predicting responses for new students ..................................... 54
  4.5.2 Predicting unobserved student responses ............................... 56
  4.5.3 Visualizing time-varying learning and content analytics .......... 57
6.2 Problem Formulation .............................................. 95
  6.2.1 Background on bandits ..................................... 97
6.3 Algorithms ............................................................. 98
  6.3.1 CLUB: An algorithm in theory ................................. 98
  6.3.2 A-CLUB: An algorithm in practice ........................... 101
  6.3.3 CPT: A Bayesian algorithm ................................. 103
6.4 Experiments ......................................................... 104
  6.4.1 Personalized cohort selection ............................... 105
  6.4.2 Personalized practice question selection .................... 108
6.5 Conclusions and Future Work ..................................... 110
6.6 Appendix: Proofs ..................................................... 111

7 Conclusions and Future Work ................................... 116

Bibliography ............................................................. 118
Illustrations

2.1 (a) The SPARFA framework processes a (potentially incomplete) binary-valued dataset of graded student–question responses to (b) estimate the underlying questions-concept association graph and the abstract conceptual knowledge of each student (illustrated here by smiley faces for student $j = 3$, the column in (a) selected by the red dashed box).

2.2 (a) Sparse question–concept association graph and (b) most important tags associated with each concept for Grade 8 Earth science with $N = 135$ students answering $Q = 80$ questions. Only 13.5% of all graded student–question responses were observed.

3.1 Visualization of the estimated question difficulty parameters $\mu_{i,k}$. “Warm” colors mean that the question requires the students to have high knowledge on those concepts. For questions testing multiple concepts, we can see that the estimated difficulty parameters clearly show which concept is the “dealbreaker.”
4.1 The SPARFA-Trace framework processes the binary-valued graded student response matrix $Y$ (binary-valued, with 1 denoting a correct response, 0 an incorrect one, and ? indicates an unobserved one) and the student activity matrices $\{R^{(t)}\}$ (binary-valued, with 1 denoting that a student studied a particular learning resource, and 0 otherwise). Upon analyzing this data, SPARFA-Trace jointly traces the student concept knowledge states $c_j^{(t)}$ (a happy face represents high concept knowledge, and a sad face represents low concept knowledge) over time, and estimates the learning resource content organization and quality parameters $D_m$, $d_m$ and $\Gamma_m$, together with question–concept association parameters $w_i$ and question difficulty parameters $\mu_i$.  

4.2 Factor graph message passing algorithm for the estimation of a set of $T$ latent state variables with Markovian transition properties from (possibly noisy) observations.  

4.3 Estimated latent student concept knowledge states for all time instances, for the UG dataset. (a) Student 1’s latent concept knowledge state evolution; (b) Average student latent concept knowledge states evolution.  

4.4 Visualized student knowledge state transition effect of two distinct learning resources for the SS dataset. (a) Student knowledge state transition effect for Learning resource 3; (b) Student knowledge state transition effect for Learning resource 9.  

4.5 (a) Question–concept association graph and concept labels for the SS dataset. (a) Question–concept association graph. Note that for the visualization to be compact, we show only 1/3 of all questions in the dataset; (b) Label of each concept.
5.1 Example solutions to the question “Find the derivative of \((x^3 + \sin x)/e^x\)” that were assigned scores of 3, 2, 1 and 0 out of 3, respectively, by our MLP-B algorithm. 

5.2 Examples of two different yet correct paths to solve the question “Simplify the expression \((x^2 + x + \sin^2 x + \cos^2 x)(2x - 3)\).”

5.3 Illustration of the clusters obtained by MLP-S by applying affinity propagation (AP) on the similarity matrix \(S\) corresponding to students’ solutions to four different mathematical questions (see Table 1 for more details about the datasets and the Appendix for the question statements). Each node corresponds to a solution. Nodes with the same color correspond to solutions that are estimated to be in the same cluster. The thickness of the edge between two solutions is proportional to their similarity score. Boxed solutions are correct; all others are in varying degrees of correctness.

5.4 Graphical model of the generation process of solutions to mathematical questions. \(\alpha, \alpha, \beta, \beta\) are hyperparameters, \(z\) and \(\Phi\) are latent variables to be inferred, and \(Y\) is the observed data defined in (5.1).

5.5 Mean absolute error (MAE) versus the number of instructor graded solutions (clusters) \(K\), for Questions 1–4, respectively. For example, on Question 1, MLP-S and MLP-B estimate the true grade of each solution with an average error of around 0.1 out of a full credit of 3. “RS” represents the random sub-sampling baseline. Both MLP-S methods and MLP-B outperforms the baseline method.

5.6 Demonstration of real-time feedback generation by MLP-B while students enter their solutions. After each expression, we compute both the probability that the student’s solution belongs to a cluster that does not have full credit and the student’s expected grade. An alert is generated when the expected credit is less than full credit.
6.1 A personalized learning schedule. .......................................................... 96

6.2 Average student credit on Assessment 5 vs. number of students used by the algorithms on the college physics dataset. The students’ performance on the follow-up assessment increases as the algorithms have access to more training data. Concretely, using data from 38 students, A-CLUB finds a PLA selection policy for students that is about 10% better than selecting randomly (as computed from the first and last data points on its curve). ......................................................... 107
Tables

3.1 Performance comparison in terms of the prediction accuracy (ACC) for the dealbreaker models (Hard DB and Soft DB) against the DINA, 3PL MIRT, and Rasch models, and also the 1-bit MC algorithm. 21

3.2 Performance comparison in terms of the area under the receiver operating characteristic curve (AUC) for the dealbreaker models (Hard DB and Soft DB) against the DINA, 3PL MIRT, and Rasch models, and also the 1-bit MC algorithm. 21

3.3 Comparison of the dealbreaker (DB) model against the Rasch model in terms of ACC when both models are fitted separately on subsets of the MT dataset. The dealbeaker model performs well on both subsets while the Rasch model does not perform well on questions with diverse response patterns across students. 24

3.4 Prediction performance of the dealbreaker model, the Rasch model, and 1-bit MC on the MovieLens dataset. 28

4.1 Comparisons of SPARFA-Trace against knowledge tracing (KT) on predicting responses for new students using using the UG dataset. SPARFA-Trace outperforms KT on all three metrics. 56

4.2 Comparisons of SPARFA-Trace against SPARFA-M on predicting unobserved student responses for the UG dataset. 58

4.3 Comparisons of SPARFA-Trace against SPARFA-M on predicting unobserved student responses for the SS dataset. 58
5.1 Datasets consisting of the solutions of 116 students to 4 mathematical questions on algebra and signal processing. See the Appendix for the question statements. . . . . . . . . . . . . . . . . . . . . . . . . . . . 85

6.1 Performance comparison of A-CLUB and CPT against other baseline algorithms on personalized cohort selection on the college physics course dataset. A-CLUB outperforms the other algorithms in terms of average student credit on the follow-up assessment (out of a full credit of 5) on both the training and test sets. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 109

6.2 Performance comparison of A-CLUB and CPT against other baseline algorithms on personalized practice question selection on the high school physics course dataset. A-CLUB, CPT, and UCB achieve comparable performance in terms of average student credit on the follow-up assessment (out of a full credit of 3). . . . . . . . . . . . . . . . . . . . . . . . . . . . . 110
Chapter 1

Introduction

Textbooks, lectures, and homework assignments were the answer to the main educational challenges of the 19th century, but they are the main bottleneck of the 21st century. Today’s textbooks are static, linearly organized, time-consuming to develop, soon out-of-date, and expensive. Lectures remain a primarily passive experience of copying down what an instructor says and writes on a board (or projects on a screen). Homework assignments that are not graded for weeks provide poor feedback to students (e.g., students) on their learning progress. Even more importantly, today’s courses provide only a “one-size-fits-all” learning experience that does not cater to the background, interests, and goals of individual students.

1.1 The Promise of Personalized Learning

We envision a world where access to high-quality, personally tailored educational experiences is affordable to all of the world’s students. The key is to integrate textbooks, lectures, and homework assignments into a personalized learning system (PLS) that closes the learning feedback loop by (i) continuously monitoring and analyzing student interactions with learning resources in order to assess their learning progress and (ii) providing timely remediation, enrichment, or practice based on that analysis. See [74], [85], [99], [113], [132], and [142] for various visions and examples.

Some progress has been made over the past few decades on personalized learning; see, for example, the sizable literature on intelligent tutoring systems discussed in [111]. To date, the lionshare of fielded, intelligent tutors have been rule-based systems that are hard-coded by domain experts to give students feedback for pre-defined scenarios
(e.g., [6], [24], [26], [75], and [143]). The specificity of such systems is counterbalanced by their high development cost in terms of both time and money, which has limited their scalability and impact in practice. Moreover, these systems are domain-specific, i.e., the rules specified by domain experts in one domain cannot be applied to other domains.

In a fresh direction, recent progress has been made on applying machine learning algorithms to mine student interaction data and educational content (see the overview articles by [10] and [122]). In contrast to rule-based approaches, machine learning-based PLSs promise to be rapid and inexpensive to deploy, which will enhance their scalability and impact. Indeed, the dawning age of “big data” provides new opportunities to build PLSs based on data rather than rules. We conceptualize the architecture of a generic machine learning-based PLS to have four interlocking components:

- **Learning analytics**: Algorithms that estimate what each student does and does not understand based on data obtained from tracking their interactions with learning content.

- **Content analytics**: Algorithms that organize learning content such as text, video, simulations, questions, and feedback hints.

- **Grading and feedback**: Algorithms that automatically grade student solutions to open-response questions and provide feedback to the student on the likely location, type, and cause of their errors.

- **Scheduling**: Algorithms that use the results of learning and content analytics to suggest to each student at each moment what they should be doing in order to maximize their learning outcomes, in effect closing the learning feedback loop.

### 1.2 Thesis Organization

We begin with a brief overview of our prior work, sparse factor analysis (SPARFA) for learning and content analytics, in Chapter 2. SPARFA uses a linear model for
graded student responses to questions and extracts a set of static knowledge profiles for each student and estimates the content and difficulty of each question. We first introduce the dealbreaker model, a non-linear and better model for learning analytics, in Chapter 3. We then introduce the time-varying SPARFA-Trace framework for time-varying learning analytics and content analytics in Chapter 4. SPARFA-Trace extends the content analytics capability of SPARFA beyond questions and also analyze the content and quality of learning resources (e.g., textbook sections, lecture videos, etc.). We introduce the MLP framework for automatic grading and feedback generation for open-response mathematical questions in Chapter 5. Finally, we introduce an approach to scheduling by introducing a contextual (multi-armed) bandits framework for personalized learning action selection in Chapter 6. We conclude and discuss avenues of future work in Chapter 7.
Chapter 2

Background: Sparse Factor Analysis for Learning and Content Analytics

In this chapter, we briefly review our prior work on the sparse factor analysis (SPARFA) framework, a suite of algorithms for joint machine learning-based learning analytics and content analytics. The SPARFA model represents the probability that a student provides the correct response to a given question in terms of three factors: their knowledge of the underlying concepts, the concepts involved in each question, and each question’s intrinsic difficulty.

Figure 2.1 provides a graphical depiction of the SPARFA framework. As shown in Figure 2.1(a), we are provided with data relating to the correctness of the students’ responses to a collection of questions. We encode these graded responses in a “gradebook,” a source of information commonly used in the context of classical test theory ([104]). Specifically, the “gradebook” is a matrix with entry $Y_{i,j} = 1$ or $0$ depending on whether student $j$ answers question $i$ correctly or incorrectly, respectively. Question marks correspond to incomplete data due to unanswered questions.

Specifically, SPARFA models the binary-valued graded response of student $j$ to question $i$ as a Bernoulli random variable (with $1$ representing a correct answer and $0$ an incorrect one) $Y_{i,j}$, and we have

$$Y_{i,j} \sim Ber(\Phi(Z_{i,j})) \quad \text{with} \quad Z_{i,j} = w_i^T c_j - \mu_i.$$ 

Here, $Z_{i,j}$ is a slack variable governing the probability of student $j$ answering question $i$ correctly or incorrectly, and $\Phi(\cdot)$ is the inverse logit/probit link function. The variable $Z_{i,j}$ depends on three factors: (i) the question–concept association vector $w_i$ which characterizes how question $i$ relates to each abstract concept, (ii) the student

Figure 2.1: (a) The SPARFA framework processes a (potentially incomplete) binary-valued dataset of graded student–question responses to (b) estimate the underlying questions-concept association graph and the abstract conceptual knowledge of each student (illustrated here by smiley faces for student $j = 3$, the column in (a) selected by the red dashed box).

concept knowledge vector $c_j$ of student $j$, and (iii) the intrinsic difficulty parameter $\mu_i$ of question $i$. The question–concept association matrix $W$, which is obtained by stacking the column vectors $w_i$, $i \in \{1, 2, \ldots\}$, can be interpreted as a real-valued variant of the Q-matrix ([11, 124]). The student concept knowledge matrix $C$ and intrinsic difficulty vector $\mu$ are formed similarly. With these definitions, we have the streamlined notation

$$Y \sim Ber(\Phi(Z)) \quad \text{with} \quad Z = WC - \mu,$$

where the inverse link function operates entry-wise on the matrix $Z$.

Armed with this model and given incomplete observations of the graded student–question responses $Y_{i,j}$, SPARFA’s goal is to estimate the factors $W$, $C$, and $M$. Such a factor-analysis problem is ill-posed in general, especially when each student answers only a small subset of the collection of questions (see [62] for a factor analysis overview). In order to overcome this issue, SPARFA makes three key model assumptions based
on observations in real-world student response data. The first key observation that enables a well-posed solution is the fact that typical educational domains of interest involve only a small number of key concepts. Consequently, $W$ becomes a tall, narrow matrix that relates the questions to a small set of abstract concepts, while $C$ becomes a short, wide matrix that relates student knowledge to that same small set of abstract concepts. Note that the concepts are “abstract” in that they will be estimated from the data rather than dictated by a subject matter expert. The second key observation is that each question involves only a small subset of the abstract concepts. Consequently, the matrix $W$ is sparsely populated. The third observation is that the entries of $W$ should be non-negative to ensure that large positive values in $C$ represent strong knowledge of the associated abstract concepts.

Leveraging these observations and the corresponding model assumptions, we proposed a suite of new algorithms for solving the SPARFA problem. Specifically, we developed SPARFA-M, which uses an efficient bi-convex optimization approach to produce maximum likelihood point estimates of the factors. SPARFA also proposes a novel method for incorporating user-defined tags that label the questions, in order to facilitate interpretation of the abstract concepts estimated by the SPARFA algorithms.

As an illustrative example of SPARFA’s efficacy, Fig. 2.2 provides the results for a dataset collected from students using [133], a science curriculum platform. The dataset consists of 145 Grade 8 students from a single school district answering a tagged set of 80 questions on Earth science; only 13.5% of all graded student–question responses were observed. We apply the SPARFA-M algorithm to retrieve the factors $W$, $C$, and $M$ using 5 latent concepts. The resulting sparse matrix $W$ is displayed as a bipartite graph in Fig. 2.2(a); circles denote the abstract concepts and boxes denote questions. Each question box is labeled with its estimated intrinsic difficulty $\mu_i$, with large positive values denoting easy questions. Links between the concept and question nodes represent the active (non-zero) entries of $W$, with thicker links denoting larger values $W_{i,k}$. Unconnected questions are those for which no abstract concept explained
the students’ answer pattern; these questions typically have either very low or very high intrinsic difficulty, resulting in nearly all students answering them correctly or incorrectly. The tags in Fig. 2.2(b) provide interpretation to the estimated abstract concepts.

The information extracted by the SPARFA framework enables a PLS to automatically provide feedback to the students on their strengths and weaknesses, and to prune out potentially problematic questions that are either too hard, too easy, too confusing, or unrelated to the concepts underlying the collection of questions. More importantly, this information allows a PLS to automatically recommend questions the students should practice on in order to improve learning.

Two previous extensions of the SPARFA framework, SPARFA-Tag [80] and SPARFA-Top [79], focus on using the instructor generated question tags and question text to enhance interpretability of the latent “concepts”.
(a) Inferred question–concept association graph.

<table>
<thead>
<tr>
<th>Concept 1</th>
<th>Concept 2</th>
<th>Concept 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changes to land (45%)</td>
<td>Evidence of the past (74%)</td>
<td>Alternative energy (76%)</td>
</tr>
<tr>
<td>Properties of soil (28%)</td>
<td>Mixtures and solutions (14%)</td>
<td>Environmental changes (19%)</td>
</tr>
<tr>
<td>Uses of energy (27%)</td>
<td>Environmental changes (12%)</td>
<td>Changes from heat (5%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept 4</th>
<th>Concept 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties of soil (77%)</td>
<td>Formulation of fossil fuels (54%)</td>
</tr>
<tr>
<td>Environmental changes (17%)</td>
<td>Mixtures and solutions (28%)</td>
</tr>
<tr>
<td>Classifying matter (6%)</td>
<td>Uses of energy (18%)</td>
</tr>
</tbody>
</table>

(b) Most important tags and relative weights for the estimated concepts.

Figure 2.2: (a) Sparse question–concept association graph and (b) most important tags associated with each concept for Grade 8 Earth science with $N = 135$ students answering $Q = 80$ questions. Only 13.5% of all graded student–question responses were observed.
Chapter 3

Dealbreaker: A Nonlinear Latent Variable Model for Educational Data

In this chapter, we propose the dealbreaker model, a non-linear, non-additive model for student responses that admits superior interpretability than the SPARFA model and excels at capturing the complex dynamics behind student responses.

3.1 Introduction

A key problem for the creation of a PLS is student-response modeling, i.e., developing principled statistical models that captures the (possibly complex) dynamics behind student responses and thus accurately predict unobserved responses. A wide range of student-response models have been proposed in the literature, including the Rasch [116], item response theory (IRT) [91], knowledge tracing [37], and factor analysis-based models [16, 28, 32, 56, 81, 108], including our previous work, SPARFA, introduced above.

The Rasch model [116] is simple yet effective for analyzing student-response data. This model characterizes the probability of a correct response as a function of two scalar parameters: the student’s ability and the question’s difficulty. The Rasch model lays the foundation for the IRT model [91], which features additional parameters characterizing the discrimination level of the questions across students and the effect of guessing. The multi-dimensional IRT (MIRT) model [118] and the factor analysis-based models expand upon the IRT model by adding multi-dimensional ability and difficulty parameters (we refer to the model dimensions as “concepts”).
3.1.1 Limits of affine student–response models

A key commonality of all the models described above is that they are affine—they characterize a student’s probability of success on a question as an affine function of the student’s knowledge on underlying concepts. While such models are simple and enable accurate prediction of unobserved student responses, they suffer from a key flaw known as the “explaining away” phenomenon [152]: Affine models allow weak knowledge of a concept to be erroneously covered up by strong knowledge of other potentially unrelated concepts. Affine models also fail to capture more complicated nonlinear dynamics underlying student responses. For instance, it may be impossible for a student to answer a question correctly without mastering a specific concept. Consider the situation where a student tries to solve the problem: “Simplify the expression \( (5x^2 \sin^2 x + 5x^2 \cos^2 x + 10x)/(x + 2) \).” Students that do not know the trigonometric identity \( \sin^2 x + \cos^2 x = 1 \), will be stymied, no matter how strong their knowledge of polynomial division. This kind of nonlinear “dealbreaker” property cannot be captured by an affine model.

Only limited progress has been made in nonlinear student-response models. For example, the deterministic inputs, noisy and-gate (DINA) model [42] posits that a student’s probability of answering a question correctly depends on each specific combination of their binary-valued concept knowledge states (e.g., \( 101 \) means that the student has mastered Concepts 1 and 3, but not 2). While the DINA model enables the characterization of more complex response behavior, such as “students have to master both Concepts 1 and 3 in order to answer this question correctly,” it remains an affine model, because the success probability is modeled as an affine function of the probabilities of the student being in each specific knowledge state. Moreover, the DINA model suffers from the fact that there can be up to \( 2^K \) possible knowledge patterns for each question involving \( K \) concepts; this prevents its use in domains that cover tens or more different concepts.
3.1.2 Contributions

In this chapter, we develop a new statistical framework for student-response modeling, dubbed the *dealbreaker model*, that avoids the drawbacks of existing models. In the dealbreaker model, the probability of a student’s success on a question depends only on their *weakest* concept mastery among all the concepts involved in that question and no others; this prevents the “explaining away” phenomenon. For the example question mentioned above, we say that not knowing the trigonometric identity \( \sin^2 x + \cos^2 x = 1 \) is the “dealbreaker” of the question.

To perform parameter inference for this non-affine model, we develop a novel, nonconvex optimization algorithm as well as a smooth approximation to the dealbreaker model that leads to even more efficient inference.

Using four distinct educational datasets, we demonstrate that the exact and approximate dealbreaker models achieve comparable or better prediction performance on unobserved student responses than state-of-the-art affine models (Rasch, MIRT, and DINA models). Moreover, we showcase the ability of our models to identify the key concept (the so-called “dealbreaker”) that is needed to answer a question correctly. This new functionality could play a significant role in the modern, machine learning-based approach to personalized learning that has been identified as a national priority in the US [1]. Going further, we report preliminary results for a movie rating dataset, which showcase the broader applicability and interpretability advantage of the dealbreaker model to domains outside of education.

3.2 The Dealbreaker Model

Let \( N \) be the total number of students and \( Q \) the total number of questions. Let \( Y_{i,j} \) denote the binary-valued graded response of student \( j \) to question \( i \), where \( Y_{i,j} = 1 \) denotes a correct response and \( Y_{i,j} = 0 \) an incorrect response. Note that some (or many) responses \( Y_{i,j} \) may be unobserved or missing. Let \( K \) be the number of concepts underlying the questions in the dataset, where the concepts are the latent factors that
control the probability of a correct answer. Let $C_{k,j}$ denote the knowledge mastery level of student $j$ on concept $k$ [81]. Also let $\mu_{i,k}$ denote the intrinsic difficulty of question $i$ on concept $k$, which characterizes the level of knowledge required on this concept for a student to answer this question correctly.

The hard dealbreaker model represents the probability that student $j$ answers question $i$ correctly as follows:

$$p(Y_{i,j} = 1) = \Phi\left(\min_{k=1,\ldots,K} (C_{k,j} - \mu_{i,k})\right)$$

$$= \min_{k=1,\ldots,K} \Phi(C_{k,j} - \mu_{i,k}).$$

(3.1)

Here, $\Phi(x)$ is a suitably-chosen link function that maps real values onto the success probability of a Bernoulli random variable in $[0, 1]$. Without loss of generality, we will exclusively use the inverse logit link function defined as $\Phi(x) = (1 + e^{-x})^{-1}$; hence, the second equality in (3.1) follows from the fact that $\Phi(x)$ is non-decreasing in $x$.

We will refer to $\min(x) = \min_k x_k : \mathbb{R}^K \to \mathbb{R}$ as the min function, with the max function defined analogously. The min function is a non-smooth, non-convex function that makes parameter estimation a nontrivial task. As an alternative, we will also use the so-called soft-min function

$$f_\alpha(x) = -\frac{1}{\alpha} \log \sum_{k=1,\ldots,K} e^{-\alpha x_k},$$

which is a smooth approximation to the min function; the parameter $\alpha > 0$ determines the quality of the approximation (larger values correspond to tighter approximations). This soft-min approximation leads to the soft dealbreaker model for graded student responses:

$$p(Y_{i,j} = 1) = \Phi\left(-\frac{1}{\alpha} \log \sum_{k=1,\ldots,K} e^{-\alpha(C_{k,j} - \mu_{i,k})}\right).$$

(3.2)

For $K = 1$, both dealbreaker models (3.1) and (3.2) coincide (trivially) with the classical Rasch model [116].
Intuitively, the two dealbreaker models state that the probability of a student answering a question correctly depends only on their weakest concept mastery that is tested in the question. For example, suppose that geometry and algebra are both involved in a question. The dealbreaker model requires the student to have strong knowledge of both geometry and algebra in order to succeed with high probability. If they have strong knowledge of only geometry but not of algebra, then they are not likely to succeed—literally, algebra is a “dealbreaker” to their success on this question.

Remark 3.1
By defining

\[ p(Y_{i,j} = 0) = \min_{k=1,\ldots,K} \Phi(\mu_{i,k} - C_{k,j}) \]  \hspace{1cm} (3.3)

instead of (3.1), we arrive at an alternative model, which we refer to as the hard dealmaker model; analogously to (3.2), a soft-version can be derived. In contrast to the dealbreaker models, these dealmaker models imply that it is sufficient for student j to master only one concept \( C_{k,j} \) to successfully answer question \( Y_{i,j} \). In what follows, we will focus on the hard and soft dealbreaker models as they (i) better reflect educational scenarios and (ii) achieve superior prediction performance in our experiments on real-world educational datasets. Nevertheless, our proposed inference methods can easily be applied to the dealmaker model. We also note that the dealmaker model may be useful in the analysis of other datasets (e.g., to model single-issue politics in voting).

Remark 3.2
The two dealbreaker models (3.1) and (3.2), as well as the hard dealmaker model in (3.3), are only identifiable in their parameters \( C_{k,j} \) and \( \mu_{i,k} \) up to a constant offset in each concept, i.e., the model predictions remain unchanged if we add an arbitrary constant \( a_k \) to the parameters \( C_{k,j}, \forall j \) and \( \mu_{i,k}, \forall i \). Therefore, parameter estimation for these models is non-unique. We will alleviate this identifiability issue by regularizing the parameters \( C_{k,j} \) and \( \mu_{i,k} \) in Sec. 3.5.1.
3.3 Inference for the Hard Dealbreaker Model

We now develop a computationally efficient parameter inference algorithm for the hard dealbreaker model. We first outline the full algorithm, which employs the alternating direction method of multipliers (ADMM) framework \[20\] for our nonconvex problem. We then detail the proximal operators that are required in our algorithm.

3.3.1 ADMM algorithm

We formulate parameter estimation for the hard dealbreaker model as an optimization problem that minimizes the negative log-likelihood of the observed student responses. Let $\Omega_1 = \{(i, j) : Y_{i,j} = 1\}$, and $\Omega_0 = \{(i, j) : Y_{i,j} = 0\}$. The dealbreaker model decomposes into the form

$$\min_{C_{k,j}, \mu_{i,k}, \forall i,j,k} \sum_{(i,j) \in \Omega_1} \max_k - \log \Phi(Z_{i,j}^k)$$

$$+ \sum_{(i,j) \in \Omega_0} \min_k - \log \Phi(-Z_{i,j}^k),$$

$$\text{subject to } Z_{i,j}^k = C_{k,j} - \mu_{i,k}.$$
The augmented Lagrangian for this problem is as follows:

\[
\begin{align*}
\text{minimize} \quad & C_{k,j}, \mu_{i,k}, \forall (i,j) \\
& \sum_{(i,j) \in \Omega_1} \max_k - \log \Phi(Z^k_{i,j}) \\
& + \sum_{(i,j) \in \Omega_0} \min_k - \log \Phi(-Z^k_{i,j}) \\
& + \frac{\rho}{2} \sum_{i,j,k} (Z^k_{i,j} - C_{k,j} + \mu_{i,k} + \Lambda^k_{i,j})^2,
\end{align*}
\]

where \( \Lambda^k_{i,j} \) is the Lagrange multiplier for the constraint \( Z^k_{i,j} = C_{k,j} - \mu_{i,k} \) and \( \rho \geq 0 \) is a (suitably chosen) scaling parameter.* We randomly initialize the variables \( Z^k_{i,j}, C_{k,j}, \mu_{i,k}, \forall i, j, k \) from the standard normal distribution, and initialize the Lagrange multipliers as \( \Lambda^k_{i,j} = 0, \forall i, j, k \). We then iterate the following steps until convergence is reached.

**Optimize over \( Z^k_{i,j} \):** For each index pair \((i, j) \in \Omega_1\), solve the following proximal problem:

\[
\begin{align*}
\text{minimize} \quad & \frac{1}{2} \sum_k (Z^k_{i,j} - C_{k,j} + \mu_{i,k} + \Lambda^k_{i,j})^2 \\
& + \frac{1}{\rho} \max_k - \log \Phi(Z^k_{i,j}),
\end{align*}
\]

and for each index pair \((i, j) \in \Omega_0\), solve the following proximal problem:

\[
\begin{align*}
\text{minimize} \quad & \frac{1}{2} \sum_k (Z^k_{i,j} - C_{k,j} + \mu_{i,k} + \Lambda^k_{i,j})^2 \\
& + \frac{1}{\rho} \min_k - \log \Phi(-Z^k_{i,j}).
\end{align*}
\]

The details of these two proximal problems are given in the next section.

**Optimize over \( C_{k,j} \):** Solve the following problem:

\[
\begin{align*}
\text{minimize} \quad & \frac{1}{2} \sum_i (Z^i_{i,j} - C_{k,j} + \mu_{i,k} + \Lambda^i_{i,j})^2.
\end{align*}
\]

*Note that we use the *scaled* augmented Lagrangian, in which the Lagrange multiplier appears inside of the least-squares penalty.
The closed-form solution is given by
\[ C_{k,j} = \frac{1}{Q} \sum_i (Z_{i,j} + \mu_{i,k} + \Lambda_{i,j}^k). \]

**Optimize over** $\mu_{i,k}$: Solve the following problem:
\[
\min_{\mu_{i,k}} \frac{1}{2} \sum_j \left( Z_{i,j}^k - C_{k,j} + \mu_{i,k} + \Lambda_{i,j}^k \right)^2.
\]

The closed-form solution is given by
\[ \hat{\mu}_{i,k} = \frac{1}{N} \sum_j (C_{k,j} - Z_{i,j}^k - \Lambda_{i,j}^k). \]

**Update Lagrange multiplier**: Compute
\[ \hat{\Lambda}_{i,j}^k = \Lambda_{i,j}^k + Z_{i,j}^k - C_{k,j} + \mu_{i,k}, \forall i,j,k. \]

### 3.3.2 Proximal operators

In the hard dealbreaker ADMM algorithm, we need to solve the following proximal problems:

\[
P_{\text{max}}: \quad \min_{x} \frac{1}{2} \| y - x \|_2^2 + \max_k g(x_k),
\]
\[
P_{\text{min}}: \quad \min_{x} \frac{1}{2} \| y - x \|_2^2 + \min_k g(-x_k).
\]

Here, $y \in \mathbb{R}^K$ and $g(x) = \frac{1}{\rho} \log(1 + e^{-x})$ is a non-increasing, non-negative convex function on $(-\infty, \infty)$. The following theorem characterizes the solution to $P_{\text{max}}$.

**Theorem 1** Assume that the entries in $y$ are sorted in ascending order. Then, the solution to the proximal problem $P_{\text{max}}$ is given by
\[ x_k = \begin{cases} \hat{\tau} & \text{for } k = 1, \ldots, \hat{K}, \\ y_k & \text{for } k = \hat{K} + 1, \ldots, K, \end{cases} \]

where $\hat{K}$ is the largest integer $M$ such that
\[ My_M - \sum_{k=1}^{M} y_k + g'(y_M) \leq 0 \]

and $\hat{\tau}$ is the solution to $\hat{K}\tau - \sum_{k=1}^{\hat{K}} y_k + g'(\tau) = 0$. 
Proof 1 The problem $P_{\text{max}}$ is equivalent to

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2}\|y - x\|_2^2 + t \\
\text{subject to} & \quad g(x_k) \leq t, \ \forall k.
\end{align*}$$

The Karush-Kuhn-Tucker (KKT) conditions for this problem are as follows:

$$\begin{align*}
x_k - y_k + \gamma_k g'(x_k) &= 0, \quad \forall k, \quad (3.4) \\
\sum_k \gamma_k &= 1, \quad (3.5) \\
\gamma_k (g(x_k) - t) &= 0, \quad \forall k, \quad (3.6)
\end{align*}$$

Here, $\gamma_k$ is the non-negative Lagrange multiplier for the inequality constraint $g(x_k) \leq t$. In the complimentary slackness condition (3.6), we have that if $\gamma_k = 0$, then $g(x_k) \leq t$. In this case, the stationarity condition (3.4) gives $x_k = y_k$. On the other hand, if $\gamma_k > 0$, then $g(x_k) = t$, meaning that $x_k = g^{-1}(t) := \tau$. In this case, (3.4) leads to $x_k = y_k - \gamma_k g'(x_k) \geq y_k$, since $\gamma_k \geq 0$ and $g'(x_k) \leq 0$ because $g(x_k)$ is non-increasing.

As a consequence, we know that the solution to $P_{\text{max}}$ is given by

$$x_k = \max\{y_k, \tau\} \quad (3.7)$$

for some constant $\tau$. Hence, we need only find $\tau$. Since $\frac{1}{2}\|y - x\|_2^2$ is non-decreasing and $\max_k g(x_k)$ is non-increasing as $\tau$ increases, we know that there will be a minimizer for $\tau$. In order to find its value, we note that the analysis above gives

$$\gamma_k = \begin{cases} 
0 & x_k = y_k, \\
\frac{y_k - \tau}{g'(\tau)} & x_k = \tau \geq y_k.
\end{cases}$$

Together with the stationary condition for $t$ (3.5), we have

$$\sum_{k'} \frac{y_{k'} - \tau}{g'(\tau)} = 1 \iff \sum_{k'} (y_{k'} - \tau) - g'(\tau) = 0,$$

where $k'$ corresponds to the indices in $x$ that satisfy $x_k = \tau$. First, we need to identify these indices. By assumption, $y_1 \leq \ldots \leq y_K$. Then, we examine the value
of \( f(\tau) = \sum_{k'} (y_{k'} - \tau) - g'(\tau) \). Note that \( f(\tau) \) is a non-increasing function of \( \tau \) as both \( \sum_{k'} (y_{k'} - \tau) \) and \( -g'(\tau) = \frac{1}{\rho(1+e^\tau)} \) are non-increasing functions of \( \tau \). To find the indices \( k' \), we check \( f(\tau) \) for different values of \( \tau \):

\[ \tau < y_1: f(\tau) = -g'(\tau) > 0, \text{ since we have } x_k = y_k \text{ for } k = 1, 2, \ldots, K \text{ from (3.7)}. \]

\[ y_1 \leq \tau < y_2: f(\tau) = y_1 - \tau - g'(\tau) \text{ since we have } x_1 = \tau \text{ and } x_k = y_k \text{ for } k = 2, \ldots, K, \]

\[ \vdots \]

\[ \tau \geq y_K: f(\tau) = \sum_{k=1}^K y_k + K\tau - g'(\tau) \text{ since we have } x_1 = \ldots = x_K = \tau, \text{ giving } f(y_K) = \sum_{k=1}^K y_k - Ky_K - g'(y_K). \]

According to the analysis above, the number of elements in \( x \) that are equal to \( \tau \) is simply the largest integer \( M \) such that \( My_M - \sum_{k=1}^M y_k + g'(y_M) \leq 0 \).

Once we have found the integer \( \widehat{K} \), the value of \( \tau \) can be found by solving \( f'(\tau) = \widehat{K} \tau - \sum_{k=1}^{\widehat{K}} y_k + g'(\tau) = 0 \). We use Newton's method by initializing \( \tau_0 = y_{\widehat{K}} \) and iteratively performing the following update:

\[
\tau_{\ell+1} = \tau_{\ell} - \frac{\widehat{K} \tau_{\ell} - \sum_{k=1}^{\widehat{K}} y_k + g'(\tau_{\ell})}{\widehat{K} + g''(\tau_{\ell})}
\]

until the sequence \( \{\tau_{\ell}\} \) converges to \( \hat{\tau} \).

In summary, the solution of \( \text{P}_{\text{max}} \) can be written as

\[
x_k = \begin{cases} 
\hat{\tau} & \text{for } k = 1, \ldots, \widehat{K}, \\
y_k & \text{for } k = \widehat{K} + 1, \ldots, K.
\end{cases}
\]

We note that \( \text{P}_{\text{max}} \) is a generalization of the proximal problem for the \( \ell_\infty \)-norm [45, 135], which corresponds to the special case of \( g(x) = |x| \).

The following theorem characterizes the solution to \( \text{P}_{\text{min}} \).
Theorem 2 Assume that the entries in $y$ are sorted in ascending order. Then, the solution to the proximal problem $P_{\min}$ is given by

$$x_k = \begin{cases} \hat{\tau} & \text{for } k = 1, \\ y_k & \text{for } k = 2, \ldots, K, \end{cases}$$

where $\hat{\tau}$ is the solution to $\tau - y_1 + g'(\tau) = 0$.

Proof 2 In this case, the function $g(-x)$ is non-decreasing on $(-\infty, \infty)$. Therefore, the value of $\min_k g(-x_k)$ depends only on the smallest element in $x$, and the other elements of $x$ will simply be equal to their corresponding elements in $y$. We need only solve for the smallest element; it will be given by the solution to the equation $\tau - y_1 + g'(-\tau) = 0$. In our algorithm, we use Newton’s method, analogously to the one used to solve $P_{\min}$ to find $\hat{\tau}$.

3.4 Inference for the Soft Dealbreaker Model

We now develop the inference algorithm for the soft dealbreaker model. As for the hard dealbreaker model, the inference problem minimizes the approximated negative log-likelihood (3.2) of the observed student responses

$$\minimize_{C_{k,j},\mu_{i,k},\forall i,j,k} \sum_{(i,j)\in \Omega_1} - \log \Phi(\min_k (C_{k,j} - \mu_{i,k}))$$

$$+ \sum_{(i,j)\in \Omega_0} - \log \Phi(- \min_k (C_{k,j} - \mu_{i,k}))$$

$$\approx \sum_{(i,j)\in \Omega_1} - \log \Phi\left(- \frac{1}{\alpha} \log \sum_k e^{-\alpha(C_{k,j} - \mu_{i,k})}\right)$$

$$+ \sum_{(i,j)\in \Omega_0} - \log \Phi\left( \frac{1}{\alpha} \log \sum_k e^{-\alpha(C_{k,j} - \mu_{i,k})}\right),$$

where $\alpha \geq 0$ controls how tight the soft-min approximates the hard-min function.

Since the approximate negative log-likelihood function is smooth in the variables $C_{k,j}$ and $\mu_{i,k}$, we can use the fast adaptive shrinkage/thresholding algorithm (FASTA) framework [55] to efficiently find a locally optimal solution to this problem.
We start by initializing the variables as for the hard dealbreaker model. To reduce the chance of getting stuck in a local optimum, we initialize $\alpha$ to a small positive value (e.g., $\alpha = 0.1$) that ensures smoothness of the initial objective function. We also initialize the stepsize $s$ to a small positive value. Then, in each iteration, we perform the following steps until convergence is reached.

**Gradient step on $C_{k,j}$ and $\mu_{i,k}$:** Calculate the gradient of the cost function $f$ with respect to $C_{k,j}$ and $\mu_{i,k}$ via

$$
\frac{\partial f}{\partial C_{k,j}} = -\sum_{i:(i,j)\in \Omega_1} e^{-\alpha(C_{k,j} - \mu_{i,k})} u + u^{1 - \frac{1}{\alpha}} + \sum_{i:(i,j)\in \Omega_0} e^{-\alpha(C_{k,j} - \mu_{i,k})} u + u^{1 + \frac{1}{\alpha}},
$$

$$
\frac{\partial f}{\partial \mu_{i,k}} = \sum_{j:(i,j)\in \Omega_1} e^{-\alpha(C_{k,j} - \mu_{i,k})} u + u^{1 - \frac{1}{\alpha}} - \sum_{j:(i,j)\in \Omega_0} e^{-\alpha(C_{k,j} - \mu_{i,k})} u + u^{1 + \frac{1}{\alpha}},
$$

where $u = \sum_{k'} e^{-\alpha(C_{k',j} - \mu_{i,k})}$. Then, perform the gradient step with respect to each $C_{k,j}$ and $\mu_{i,k}$, $\forall i, j, k$, via

$$
C_{k,j} \leftarrow C_{k,j} - s \frac{\partial f}{\partial C_{k,j}}, \quad \mu_{i,k} \leftarrow \mu_{i,k} - s \frac{\partial f}{\partial \mu_{i,k}},
$$

and perform a backtracking line-search [21] on $s$.

**Stepsize $s$ update:** Adaptively select the stepsize $s$ using the value of the variables from this iteration and the last iteration according to the Barzilai-Borwein rule [13]. This selection rule achieves faster empirical convergence than other methods, e.g., [15].

The steps above do not update the value of $\alpha$, but in practice, we update the value of $\alpha$ using a rule inspired by the continuation method [153] in convex optimization. The procedure we use works as follows. First, we hold the value of $\alpha$ fixed and perform the above iterations until convergence. Then, we increase the value of $\alpha$ by multiplying it by a constant factor (e.g., 5), and run the iterations again by initializing them with the converged estimates of $C_{k,j}$ and $\mu_{i,k}$ from the previous iterations. We terminate the iterations until they converge for a large value of $\alpha$ (e.g., $\alpha = 20$). At this point,
Table 3.1: Performance comparison in terms of the prediction accuracy (ACC) for the dealbreaker models (Hard DB and Soft DB) against the DINA, 3PL MIRT, and Rasch models, and also the 1-bit MC algorithm.

<table>
<thead>
<tr>
<th>Model</th>
<th>Hard DB</th>
<th>Soft DB</th>
<th>DINA</th>
<th>3PL MIRT</th>
<th>Rasch</th>
<th>1-bit MC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K 3 6 10</td>
<td>3 6 10</td>
<td>3 6 10</td>
<td>3 6 10</td>
<td>3 6 10</td>
<td></td>
</tr>
<tr>
<td>MT</td>
<td>0.798 0.796 0.797</td>
<td>0.801 0.799 0.796</td>
<td>0.770 0.775 0.700</td>
<td>0.673 0.723 0.700</td>
<td>0.795 0.802</td>
<td></td>
</tr>
<tr>
<td>UG</td>
<td>0.871 0.871 0.870</td>
<td>0.875 0.871 0.873</td>
<td>0.850 0.800 0.862</td>
<td>0.757 0.754 0.732</td>
<td>0.853 0.873</td>
<td></td>
</tr>
<tr>
<td>CE</td>
<td>0.689 0.685 0.686</td>
<td>0.685 0.682 0.683</td>
<td>0.684 0.641 0.590</td>
<td>0.533 0.558 0.573</td>
<td>0.686 0.688</td>
<td></td>
</tr>
<tr>
<td>edX</td>
<td>0.929 0.925 0.923</td>
<td>0.927 0.927 0.926</td>
<td>0.926 0.917 0.901</td>
<td>0.865 0.860 0.864</td>
<td>0.926 0.928</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: Performance comparison in terms of the area under the receiver operating characteristic curve (AUC) for the dealbreaker models (Hard DB and Soft DB) against the DINA, 3PL MIRT, and Rasch models, and also the 1-bit MC algorithm.

<table>
<thead>
<tr>
<th>Model</th>
<th>Hard DB</th>
<th>Soft DB</th>
<th>DINA</th>
<th>3PL MIRT</th>
<th>Rasch</th>
<th>1-bit MC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K 3 6 10</td>
<td>3 6 10</td>
<td>3 6 10</td>
<td>3 6 10</td>
<td>3 6 10</td>
<td></td>
</tr>
<tr>
<td>MT</td>
<td>0.841 0.839 0.839</td>
<td>0.840 0.839 0.837</td>
<td>0.784 0.730 0.787</td>
<td>0.646 0.690 0.647</td>
<td>0.839 0.838</td>
<td></td>
</tr>
<tr>
<td>UG</td>
<td>0.832 0.831 0.830</td>
<td>0.831 0.830 0.830</td>
<td>0.760 0.788 0.760</td>
<td>0.613 0.633 0.635</td>
<td>0.800 0.830</td>
<td></td>
</tr>
<tr>
<td>CE</td>
<td>0.744 0.744 0.745</td>
<td>0.748 0.746 0.742</td>
<td>0.750 0.679 0.596</td>
<td>0.524 0.560 0.573</td>
<td>0.747 0.747</td>
<td></td>
</tr>
<tr>
<td>edX</td>
<td>0.904 0.906 0.901</td>
<td>0.912 0.911 0.909</td>
<td>0.906 0.832 0.770</td>
<td>0.754 0.753 0.749</td>
<td>0.911 0.910</td>
<td></td>
</tr>
</tbody>
</table>
the large final value of $\alpha$ ensures that the soft min function closely approximates the true, non-smooth min function. We emphasize that this continuation approach also speeds up the numerical solver and reduces the chance that our method gets stuck in a bad local minimum, eventually improving the quality of our results.

3.5 Experiments

We now demonstrate the prediction performance of the dealbreaker model on unobserved student responses using four real-world educational datasets. We furthermore showcase the interpretability of the dealbreaker model by visualizing the “dealbreaker” concept for each question. In addition, we use a movie rating dataset to show that the dealbreaker model can be applied to other datasets outside of education.

3.5.1 Predicting unobserved student responses

We compare the hard and soft dealbreaker models against three state-of-the-art student-response models: the DINA model [42], the 3PL multi-dimensional item response theory (3PL MIRT) model [118], and the Rasch model [116]. We also include a comparison against the 1-bit matrix completion (1-bit MC) algorithm proposed in [40]. The following four datasets are used.

**MT:** $N = 99$ students answering $Q = 34$ questions in a high-school algebra test administered in Amazon’s Mechanical Turk [4]; 100% of the responses are observed.

**UG:** $N = 92$ students answering $Q = 203$ questions in an undergraduate course on introduction to computer engineering; 99.5% of the responses are observed.

**CE:** $N = 1567$ students answering $Q = 60$ questions in a college entrance exam; 70.7% of the responses are observed.

**edX:** $N = 6403$ students answering $Q = 197$ questions in a massive open online course (MOOC) on signals and systems; 15.0% of the responses are observed.
**Experimental setup:** To reduce the identifiability issue of the dealbreaker model, we add the regularization term $\frac{\lambda}{2}(\sum_{k,j} C_{k,j}^2 + \sum_{i,k} \mu_{i,k}^2)$ to the cost functions of both the hard and soft dealbreaker optimization problems and select the parameter $\lambda$ using cross-validation. In each cross-validation run, we randomly leave out 20% of the student responses in the dataset (the “unobserved” data) and train the algorithms on the rest of the responses before testing their prediction performance on the unobserved data. We repeat each experiment 20 times with different random partitions of the dataset.

For the Rasch model and the MIRT model, we perform inference using the R MIRT package [29]. The DINA model is implemented as detailed in [41, 42]. For the MIRT model, the DINA model, and both dealbreaker models, we sweep the number of concepts from $K \in \{3, 6, 10\}$.

We evaluate the prediction performance on the unobserved student responses of each model using two different metrics: (i) prediction accuracy (ACC), which is simply the portion of correct predictions, and (ii) area under the receiver operating characteristic curve (AUC) of the resulting binary classifier [69]. Both metrics take on values in $[0, 1]$, with large values indicating better prediction performance.

**Results and discussion:** Tables 1 and 2 show the average performance of each algorithm on each dataset using each metric over 20 random splits of the data. With only two exceptions, we see that both dealbreaker models slightly outperform the other educational models in terms of prediction accuracy (ACC) and achieve slightly better or comparable performance with the Rasch model in terms of AUC. Moreover, the performances of the dealbreaker models and the Rasch model are very close to each other and much better than the DINA model and the 3PL MIRT model. The performance of the dealbreaker models is comparable to the 1-bit MC algorithm, whose parameters admit no interpretability.

Note that the results shown in Tables 1 and 2 correspond to the prediction performance over the entire dataset. We now compare the prediction performance of
Table 3.3: Comparison of the dealbreaker (DB) model against the Rasch model in terms of ACC when both models are fitted separately on subsets of the MT dataset. The dealbreaker model performs well on both subsets while the Rasch model does not perform well on questions with diverse response patterns across students.

<table>
<thead>
<tr>
<th></th>
<th>MT-DB</th>
<th>MT-Rasch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft DB</td>
<td>0.799</td>
<td>0.810</td>
</tr>
<tr>
<td>Rasch</td>
<td>0.775</td>
<td>0.808</td>
</tr>
</tbody>
</table>

The dealbreaker models against other models (the Rasch model, in particular, as it is the best performing educational baseline algorithm) on different questions. Towards this end, we fit the soft dealbreaker model (with \( K = 3 \) concepts) and the Rasch model on the MT dataset and analyze their prediction performance on each question separately.

The top 5 questions that the dealbreaker model performs best on have average scores (portion of students with correct answers) of 67\%, 63\%, 67\%, 74\%, 70\%, while the top-5 questions that the Rasch model predicts best on have average scores of 83\%, 11\%, 95\%, 87\%, 99\%. Thus, we conclude that the Rasch model excels at very easy and very hard questions, while the dealbreaker model excels at questions with more diverse response patterns across students.

To further validate our observation, we divide the MT dataset with \( Q = 34 \) questions into two smaller, separate datasets, each with \( Q = 17 \) questions. One of them consists of questions on which the dealbreaker model outperforms the Rasch model (labeled MT-DB) in the prediction experiments above (using the entire dataset), and the other consists of questions on which the Rasch model outperforms the dealbreaker model (labeled MT-Rasch). We then repeat prediction experiments on these two small datasets separately.

Table 3.3 shows the performance of each algorithm on each small dataset using the ACC metric. We see that the dealbreaker model performs well on both small datasets,
while the Rasch model’s performance deteriorates significantly on the subset of the
questions in the MT-DB dataset. These results support our observation that the
simplicity of the Rasch model is best suited for questions with uniform response patterns
across students (i.e., very easy or very hard questions), whereas the dealbreaker model
is better suited for questions having more complex concept-understanding requirements
for students to achieve success.

We emphasize that the soft dealbreaker model enables more efficient parameter
inference compared to the hard dealbreaker model. For example, a single run of our
Python code for the soft dealbreaker model with the UG dataset with 92 students
and 203 questions takes only 10 s compared to 30 s for the hard dealbreaker model on
an Intel i7 laptop with a 2.8 GHz CPU and 8 GB memory.

3.5.2 Visualizing the dealbreaker model

We now demonstrate the parameter interpretability afforded by the dealbreaker model
using the MT dataset.

Experimental setup: The MT dataset comes with 13 domain-expert provided
tags (or labels) on every question, which summarize the tested concepts. We use
these tags as information on the underlying knowledge structure of the dataset and
set $K$ equal to the number of unique tags, letting each tag correspond to a unique
concept. For each question, we only estimate the difficulty parameters of the concepts
that it is associated with, and set the difficulty parameters of the other concepts to
$\mu_{i,k} = -\infty$ so that they cannot be chosen as the minimum element in the min function
on $C_{k,j} - \mu_{i,k}$ in the dealbreaker model.

Results and discussions: Figure 1 visualizes the estimated parameters $\mu_{i,k}$ for the
MT dataset. Each grid cell in the figure represents the difficulty of a question with
respect to a particular concept; “warm” colors (positive values) mean that the question
requires high knowledge of a concept, “cold” colors (negative values) mean that the
Figure 3.1: Visualization of the estimated question difficulty parameters $\mu_{i,k}$. “Warm” colors mean that the question requires the students to have high knowledge on those concepts. For questions testing multiple concepts, we can see that the estimated difficulty parameters clearly show which concept is the “dealbreaker.”

question requires only a moderate level of knowledge on a concept, and white means that a concept is not tested in the question.

Now we take a closer look at the questions that involve multiple concepts. For example, Question 3 corresponds to

$$\text{If } \frac{3x}{7} - \frac{9}{8} = -5, \text{ then } x = ?$$

The question tags are “Solving equations” and “Fractions,” and the estimated question concept difficulties show that “Fractions” is the dealbreaker in this question. This matches with the observation that the key to answering this question correctly is to understand fractions, while the part that involves equation solving is relatively straightforward. As another example, Question 20 in this dataset is:

Compute $\lim_{x \to 0} \frac{\sin x}{x}$.

The question tags are “Fractions,” “Trigonometry,” and “Limits,” and the estimated question concept difficulties show that “Limits” is the dealbreaker here, in agreement
with the fact that the key to solving this question is to have a good knowledge on limits (more precisely, l'Hôpital's rule), while the fraction and trigonometry concepts needed to answer this question are less critical.

These examples highlight the advantage of the nonlinear dealbreaker model over affine models, since it can identify the most critical concepts involved in a question (e.g., in the widely used Q-matrix model [11], every concept involved in the question is treated as though it contributes equally to the students' success probability). This information could enable a machine learning-based intelligent tutoring system to generate more targeted feedback for remediation or when a student asks for a hint on a question.

3.5.3 Interpreting movie ratings

To demonstrate the broader applicability of the dealbreaker model to domains outside of education, we perform inference on the “MovieLens 100k” dataset [65] consisting of the integer-valued (1-to-5) ratings of $N = 943$ users on $Q = 1682$ movies. To evaluate the performance of the dealbreaker model, we convert the entries into binary values using the approach proposed in [40], i.e., we compare each entry to the average rating across the entire dataset (1 and 0 implies above and below average, respectively). We perform a prediction experiment as in Sec. 3.5.1 and compare the performance of the soft dealbreaker model with $K = 19$ (using the provided 19 genres with the genre labels of each movie) to the Rasch model and to the 1-bit matrix completion algorithm (1-bit MC) as proposed in [40].

Table 3.4 shows the average prediction performance on the MovieLens dataset on both the ACC and AUC error metrics over 20 random splits of the dataset. Note that although the 1-bit MC algorithm slightly outperforms the soft dealbreaker model in terms of prediction performance, it offers virtually no interpretability of its model parameters.

We now report some interesting observations made by interpreting the estimated
Table 3.4: Prediction performance of the dealbreaker model, the Rasch model, and 1-bit MC on the MovieLens dataset.

<table>
<thead>
<tr>
<th></th>
<th>soft DB</th>
<th>Rasch</th>
<th>1-bit MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>0.713</td>
<td>0.689</td>
<td><strong>0.718</strong></td>
</tr>
<tr>
<td>AUC</td>
<td>0.777</td>
<td>0.730</td>
<td><strong>0.779</strong></td>
</tr>
</tbody>
</table>

dealbreaker model parameters. The movies “Pretty Woman,” “Sabrina,” and “While You Were Sleeping,” all have “Comedy” and “Romance” as their genres, with “Romantic” being the dealbreaker for these movies. “Romance” is the dealbreaker in all of these movies as they have large negative $\mu$ values on the “Comedy” genre (i.e., even users who do not particularly favor comedy would not dislike these movies) and large positive values on the “Romance” genre (i.e., users who do not favor romance would dislike these movies). On the contrary, “Bram Stoker’s Dracula” has both “Horror” and “Romance” genres but with “Horror” as its dealbreaker—users who dislike horror movies are much less likely to enjoy it than users who dislike romantic movies.

Another interesting observation is that most of the highly rated movies (e.g., “Fargo,” “Forrest Gump,” and “Star Wars”) cover many genres yet have no particular dealbreaker (i.e., they have large negative $\mu$ values for all involved genres). This implies that, even if a user does not like some of the genres, they may still like these movies. We feel that these preliminary results are encouraging, since they highlight the advantage of the nonlinear dealbreaker model for collaborative filtering applications as compared to affine models that excel in prediction but lack interpretability.

### 3.6 Conclusions

We have developed the dealbreaker model for analyzing students’ responses to questions. Our model is nonlinear and characterizes the probability of a student’s success on a question as a function of their weakest concept mastery, i.e., the “dealbreaker.”
model helps us to gain deep insights into the knowledge structure of questions and to identify the key factors behind student response patterns on different questions. We have developed two inference algorithms for estimating the parameters of the hard and soft versions of the dealbreaker model, and have shown that they achieve excellent prediction performance on unobserved student responses, while enabling human interpretability of the estimated parameters. In addition, an application of the dealbreaker model to a movie rating dataset has shown that it provides an advantage compared to affine models in terms of interpretability of the model parameters.

There are a number of avenues for future work. Clearly, the performance of the dealbreaker model on a variety of educational datasets as well as on a movie rating dataset (especially in terms of interpretability) provides a call-to-action for the exploration of other nonlinear models. Adding extra functionality to the dealbreaker model also appears promising. For example, it is often the case that each question only covers a small number of concepts out of many (i.e., concept usage is sparse) [81]. Enforcing such a sparsity property on a dealbreaker model is challenging when there are no a-priori question labels available for the dataset. Furthermore, it is possible to extend the dealbreaker model from modeling binary data to ordinal data (e.g., the actual ratings in collaborative filtering applications), which may further improve the performance of the dealbreaker model on other applications.
Chapter 4

Time-varying Learning and Content Analytics via Sparse Factor Analysis

In this chapter, we propose SPARFA-Trace, a framework for time-varying learning and content analytics that traces student knowledge evolution over time and also analyze the content and quality of learning resources.

4.1 Introduction

While powerful, the SPARFA framework has two important limitations. First, it assumes that the students’ concept knowledge states remain constant over time. This complicates its application in real learning scenarios, where students learn (and forget) concepts over time (weeks, months, years, decades) ([27, 36, 95]). Second, SPARFA models only the students’ interactions with questions, which measure concept knowledge states, and not other kinds of learning opportunities, such as reading a textbook, viewing a lecture, or conducting a laboratory or Gedankenexperiment. This complicates its application in automatically recommending new resources to individual students for remedial or enrichment studies.

4.1.1 SPARFA-Trace: Time-varying learning and content analytics

To address these limitations, we develop SPARFA-Trace, an on-line estimation algorithm that jointly performs time-varying LA and CA. The core machinery is based on blind approximate Kalman filtering, which makes SPARFA-Trace more computationally efficient than the dynamic factor analysis algorithm ([35]) and the dynamic latent trait model ([46]).
Figure 4.1: The SPARFA-Trace framework processes the binary-valued graded student response matrix $Y$ (binary-valued, with 1 denoting a correct response, 0 an incorrect one, and ? indicates an unobserved one) and the student activity matrices $\{R^{(t)}\}$ (binary-valued, with 1 denoting that a student studied a particular learning resource, and 0 otherwise). Upon analyzing this data, SPARFA-Trace jointly traces the student concept knowledge states $c_{j}^{(t)}$ (a happy face represents high concept knowledge, and a sad face represents low concept knowledge) over time, and estimates the learning resource content organization and quality parameters $D_{m}$, $d_{m}$ and $\Gamma_{m}$, together with question–concept association parameters $w_{i}$ and question difficulty parameters $\mu_{i}$.

The main working principles of SPARFA-Trace are illustrated in Fig. 4.1. Time-varying LA are performed by tracing (tracking) the evolution of each student’s concept knowledge state vector $c_{j}^{(t)}$ over time $t$, based on observed binary-valued (correct/incorrect) graded student responses to questions matrix $Y$ and on the student activity matrices $R^{(t)}$. CA are performed by estimating the student concept knowledge state transition parameters $D_{m}$, $d_{m}$ and $\Gamma_{m}$, the question–concept associations, and the question intrinsic difficulties $w_{i}$ and $\mu_{i}$ based on the estimated student concept knowledge states at all time instances.

Tracing the students’ concept knowledge states over time is complicated by the fact that the observations are noisy, binary-valued graded student responses to questions. Furthermore, the underlying state-transition and observation parameters are, in general, unknown in real educational scenarios. To perform this on-line estimation, we develop
a novel message passing-based algorithm that employs an elegant approximation (based on a novel convex optimization and expectation-maximization framework) that enables us to apply an approximate Kalman filter ([71]).

To test and validate the effectiveness of SPARFA-Trace, we conduct a series of validation experiments using synthetic educational datasets as well as real-world educational datasets collected with OpenStax Tutor ([25], [105]). We show that SPARFA-Trace can accurately trace student concept knowledge, estimate student concept knowledge state transition parameters, and estimate the question-dependent parameters. Furthermore, we show that it achieves comparable or better performance than existing approaches on predicting unobserved student responses.

4.1.2 Related work

The closest related work to SPARFA-Trace is knowledge tracing (KT), a popular technique for tracing student knowledge evolution over time and for predicting future student performance (see, e.g., [37, 106]). Powerful as it is, KT suffers from three key drawbacks. First, KT uses binary student knowledge state representations, characterizing students as to whether they have mastered a certain concept (or skill) or not. The limited explanatory power of binary concept knowledge state representations prohibits the design of more powerful and sophisticated LA and CA algorithms. Second, KT assumes that each question is associated with exactly one concept. This restriction limits KT to very narrow educational domains and prevents it from generalizing to typical courses/assessments involving multiple concepts. Third, KT uses a single “probability of learning” parameter to characterize the student knowledge state transitions over time and assumes that a concept cannot be forgotten once it is mastered. This limits KT’s ability to perform accurate CA, i.e., analyze the quality and organization of different learning resources that lead to different student knowledge state transitions.

Various other machine learning algorithms have been designed for personalized
learning. Specifically, matrix and tensor factorization approaches have been applied to analyze graded student responses in order to extract student ability parameters and/or question–concept relationships. Examples include item response theory (IRT) ([3, 66, 91, 116]), and other factor analysis models ([11, 34, 81, 88, 124]). While these methods have shown to provide good prediction performance on unobserved student responses, they do not take into account the temporal dynamics involved in the process of a course. Therefore, these approaches are only suitable to a static testing scenario, such as the graduate record examinations (GRE), standardized tests, placement exams, etc. (see [141] for details).

Moreover, most of the existing educational data analysis approaches rely on given question–concept mappings. A recent approach without requiring question–concept mappings, described in [57, 58], jointly estimates both question–concept (item–skill) mappings and student concept mastery evolution over time purely from response data. Their method, however, suffers from the following deficiencies: First, [57] models the students’ latent concept knowledge as a small number of discrete values and the entire dynamic process for learning is modeled as a hidden Markov model (HMM). Such discrete concept knowledge states do not provide desirable interpretability when the number of discrete student concept knowledge values is low (the authors used 3 distinct knowledge levels in their paper). In contrary, the proposed SPARFA-Trace framework models student latent concept knowledge states as continuous random variables, providing finer knowledge representations. Second, [57] does not handle questions that involve multiple concepts. In contrary, the proposed SPARFA-Trace framework directly takes into account questions involving multiple concepts in the probabilistic model. Third, [57, 58] introduced a Gibbs sampler approach to infer all of the parameters; such an approach is known to be computationally intensive and, hence, will not scale to large datasets, such as MOOC-sized data. In contrary, the proposed SPARFA-Trace framework uses a computationally efficient EM approach, which is capable of scaling to the MOOC scale.
4.2 Statistical Model for Time-Varying Learning and Content Analytics

We start by extending the SPARFA statistical model ([81]) to trace student concept knowledge over time in Sec. 4.2.1. In Sec. 4.2.2, we characterize the transition of a student’s concept knowledge states between consecutive time instances as an affine model, which is parameterized by (i) the learning resource(s) the student interacted with, and (ii) how these learning resource(s) affect students’ concept knowledge states.

4.2.1 Statistical model for time-varying graded student responses to questions

The SPARFA-Trace statistical model characterizes the probability that a student answers a question correctly at a particular time instance in terms of (i) the student’s knowledge on every concept at this particular time instance, (ii) how the question relates to each concept, and (iii) the intrinsic difficulty of the question. To this end, let $N$ denote the number of students, $K$ the number of latent concepts in the course/assessment, and $T$ the total number of time instances throughout the course/assessment. We define the $K$-dimensional vectors $c_j^{(t)} \in \mathbb{R}^K, t \in \{1, \ldots, T\}, j \in \{1, \ldots, N\}$, to represent the latent concept knowledge state of the $j^{th}$ student at time instance $t$. Let $Q$ be the total number of questions. We further define the mapping $i(t, j) : \{1, \ldots, T\} \times \{1, \ldots, N\} \mapsto \{1, \ldots, Q\}$, which maps student and time instance indices to question indices; this information can be extracted from the student activity log. We will use the shorthand notation $i_j^{(t)} = i(t, j)$ to denote the index of the question that the $j^{th}$ student answers $i_j^{(t)}$ at time instance $t$. Under this notation, we define the $K$-dimensional vector $w_i^{(t)} \in \mathbb{R}^K, i \in \{1, \ldots, Q\}$, as the question–concept association vector of the question that the $j^{th}$ student answered at time instance $t$. Finally, we define the scalar $\mu_{i_j^{(t)}} \in \mathbb{R}$ to be the intrinsic difficulty of question $i_j^{(t)}$, with large, positive values of $\mu_{i_j^{(t)}}$ representing difficult questions, while a small, negative
values of $\mu_{ij}^{(t)}$ representing easy ones.

Given these quantities, we characterize the binary-valued graded response, where 1 denotes a correct response and 0 an incorrect response, of student $j$ to question $i_j^{(t)}$ at time instance $t$ as a Bernoulli random variable:

\[
Y_j^{(t)} \sim \text{Ber}(\Phi(Z_j^{(t)})), \quad (t, j) \in \Omega_{ obs},
\]

\[
Z_j^{(t)} = w_{ij}^T c_j^{(t)} - \mu_{ij}^{(t)}, \quad \forall t, j.
\] (4.1)

Here, the set $\Omega_{ obs} \subseteq \{1, \ldots, T\} \times \{1, \ldots, N\}$ contains the indices associated with the observed graded student response data, since some student responses might not be observed in practice. $\Phi(z)$ denotes the inverse probit link function $\Phi_{\text{pro}}(z) = \int_{-\infty}^{z} \mathcal{N}(t) \, dt$, where $\mathcal{N}(t) = \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)$ is the probability density function (PDF) of the standard normal distribution. (Note that the inverse logit link function could also be used. However, the inverse probit link function simplifies the calculations in Sec. 4.3.3.) The likelihood of an observation $Y_j^{(t)}$ can, alternatively, be written as

\[
p(Y_j^{(t)} | c_j^{(t)}) = \Phi\left((2Y_j^{(t)} - 1)(w_{ij}^T c_j^{(t)} - \mu_{ij}^{(t)})\right),
\]

a shorthand expression that we will often use in the remainder of the chapter.

Following the original SPARFA framework ([81]), we impose the following model assumptions:

(A1) The number of concepts is much smaller than the number of questions and the number of students: This assumption imposes a low-dimensional model on the students’ responses to questions.

(A2) The vector $w_i$ is sparse: This assumption is based on the observation that each question should only be associated with a few concepts out of all concepts in the domain of a course/assessment.

(A3) The vector $w_i$ has non-negative entries: This assumption enables one to interpret the entries in $c_j$ to be the latent concept knowledge of each student, with positive
values representing high concept knowledge, and negative values representing low concept knowledge.

These assumptions are reasonable in the majority of real-world educational scenarios and alleviate the common identifiability issue inherent to factor analysis. To illustrate, if \( Z_{i,j} = w_i^T c_j \), then for any orthonormal matrix \( Q \) with \( Q^T Q = I \) we have \( Z_{i,j} = w_i^T Q^T Q c_j = \tilde{w}_i^T c_j \). Hence, the estimation of \( w_i \) and \( c_j \) is, in general, non-unique up to a unitary unitary transformation. See [62] and [81] for more details. The assumptions also improve the interpretability of the variables \( w_i \), \( c_j \), and \( \mu_i \).

4.2.2 Statistical model for student knowledge state transitions

The SPARFA model (4.1) assumes that each student’s concept knowledge remains constant throughout a course/assessment. Although this assumption is valid in the setting of a single test or exam, it provides limited explanatory power in analyzing the (possibly semester-long) process of a course, during which the students’ concept knowledge evolves through time. We assume here that the concept knowledge state evolves for two primary reasons: (i) A student may interact with learning resources (e.g., read a section of an assigned textbook, watch a lecture video, conduct a lab experiment, or run a computer simulation), all of which are likely to result in an increase of their concept knowledge. (ii) A student may simply forget a learned concept, resulting in a decrease of their concept knowledge. For the sake of simplicity of exposition, we will treat the forgetting effect ([151]) as a special learning resource that reduces students’ concept knowledge over time.

We propose a latent state transition model that models student concept knowledge evolution between two consecutive time instances. To this end, we assume that there are a total of \( M \) distinct learning resources. We define the mapping \( m(t, j) : \{1, \ldots, T\} \times \{1, \ldots, N\} \mapsto \{1, \ldots, M\} \) from time and student indices to learning resource indices; this information can be extracted from the student activity log. We will use the shorthand notation \( m_j^{(t-1)} = m(t-1, j) \) to denote the index of the learning
resource that student $j$ studies between time instance $t - 1$ and time instance $t$. Armed with this notation, the student activity summary matrices $\mathbf{R}^{(t)}$ illustrated in Fig. 4.1 are defined by $R_{j,m_j}^{(t)} = 1, \forall (t,j)$, meaning that student $j$ interacted with learning resource $m_j^{(t)}$ at time instance $t$, and 0 otherwise.

We are now ready to model the transition of student $j$’s latent concept knowledge state from time instance $t - 1$ to $t$ as

$$c_j^{(t)} = (\mathbf{I}_K + \mathbf{D}_{m_j^{(t-1)}})c_j^{(t-1)} + \mathbf{d}_{m_j^{(t-1)}} + \epsilon_j^{(t-1)}, \quad \epsilon_j^{(t-1)} \sim \mathcal{N}(0_K, \Gamma_{m_j^{(t-1)}}),$$

where $\mathbf{I}_K$ is the $K \times K$ identity matrix; $\mathbf{D}_{m_j^{(t-1)}}$, $\mathbf{d}_{m_j^{(t-1)}}$, and $\Gamma_{m_j^{(t-1)}}$ are latent student concept knowledge state transition parameters, which define an affine model on the transition of the $j$th student’s concept knowledge state by interacting with learning resource $m_j^{(t-1)}$ between time instances $t - 1$ and $t$. $\mathbf{D}_{m_j^{(t-1)}}$ is a $K \times K$ matrix, $\mathbf{d}_{m_j^{(t-1)}}$ is a $K \times 1$ vector, and $0_K$ is the $K$-dimensional zero vector. The covariance matrix $\Gamma_{m_j^{(t-1)}}$ characterizes the uncertainty induced in the student concept knowledge state transition by interacting with learning resource $m_j^{(t-1)}$. Note that (4.2) also has the following equivalent form

$$p(c_j^{(t)} | c_j^{(t-1)}) = \mathcal{N}(c_j^{(t)} | (\mathbf{I}_K + \mathbf{D}_{m_j^{(t-1)}})c_j^{(t-1)} + \mathbf{d}_{m_j^{(t-1)}}, \Gamma_{m_j^{(t-1)}}),$$

where $\mathcal{N}(x|\mu, \Sigma)$ represents a multivariate Gaussian distribution with mean vector $\mu$ and covariance matrix $\Sigma$.

In order to reduce the number of parameters and to improve identifiability of the parameters $\mathbf{D}_{m_j^{(t-1)}}$, $\mathbf{d}_{m_j^{(t-1)}}$, and $\Gamma_{m_j^{(t-1)}}$, we impose three additional assumptions on the student knowledge state transition matrix $\mathbf{D}_{m_j^{(t-1)}}$:

(A4) $\mathbf{D}_{m_j^{(t-1)}}$ is lower triangular: This assumption means that, the $k$th entry in the student concept knowledge vector $c_j^{(t)}$ is only influenced by the $1$st, $\ldots$, $(k - 1)$th entry in $c_j^{(t-1)}$. As a result, the upper entries in $c_j^{(t-1)}$ represent pre-requisite concepts that are covered early in the course, while lower entries represent advanced concepts that are covered towards the end of the course. Using this
assumption, it is possible to extract prerequisite relationships among concepts purely from student response data.

(A5) $D_{m_j(t-1)}$ has non-negative entries: This assumption ensures, for example, that having low concept knowledge at time instance $t - 1$ (negative entries in $c_j^{(t-1)}$) does not result in high concept knowledge at time instance $t$ (positive entries in $c_j^{(t)}$).

(A6) $D_{m_j(t-1)}$ is sparse: This assumption amounts for the observation that learning resources typically only cover a small subset of concepts among all concepts covered in a course.

In contrast to the student concept knowledge transition matrix $D_{m_j(t-1)}$, we do not impose sparsity or non-negativity properties on the intrinsic student concept knowledge state transition vector $d_{m_j(t-1)}$ in (4.2); large, positive values in $d_{m_j(t-1)}$ represent learning resources with good quality that boost students’ concept knowledge, while small, negative values in $d_{m_j(t-1)}$ represent learning resources that reduce students’ concept knowledge. This setup enables our framework to model cases of poorly designed, misleading, or off-topic learning resources that distract or confuse students. Note that the forgetting effect can also be modeled as a learning resource with negative entries in $d_{m_j(t-1)}$.

To further reduce the number of parameters, we assume that the covariance matrix $\Gamma_{m_j(t-1)}$ is diagonal. This assumption is mainly made for simplicity; the analysis of more evolved models is left for future work.

4.3 Time-Varying Learning Analytics

Recall that time-varying LA requires an on-line algorithm that traces the evolution of student concept knowledge over time, by analyzing binary-valued graded student responses. Designing such an algorithm it is complicated by the fact that the binary-valued graded student responses correspond to a non-linear and non-Gaussian
observation model (resulting from (4.1)). A number of approaches have been proposed to handle non-linear and non-Gaussian on-line estimation problems. Particle filter ([44, 126]) uses a set of Monte-Carlo particles to approximately estimate the latent states. However, its huge computational complexity prevent it from being applied to personalized learning at large scale, which requires immediate feedback. The Kalman filter ([71]) is an efficient approach for on-line state estimation problems in linear dynamical systems (LDSs) with Gaussian observations. However, the Kalman filter cannot be directly applied to time-varying LA since the observed binary-valued graded student responses are non-Gaussian. Various approximations have been proposed to fit the state estimation problem in a non-linear and non-Gaussian system into the Kalman filter framework ([48, 147, 157]), but they are still too computationally extensive for our application.

We now introduce a set of computationally efficient approximations that build upon ideas in expectation propagation ([98, 117]), which enable us to recast the time-varying LA problem as an approximate Kalman filter. We begin in Sec. 4.3.1 and Sec. 4.3.2 by reviewing the key elements of the Kalman filtering and smoothing approach, and then detail our approximate Kalman filter in Sec. 4.3.3.

For notational simplicity, we will omit the student index $j$ in this section, i.e., the quantities $D_{m_j(t-1)}$ and $d_{m_j(t-1)}$ are replaced by $D_{m(t-1)}$ and $d_{m(t-1)}$. Moreover, we use the shorthand notation $D_{m(t-1)}$ for the quantity $I_K + D_{m(t-1)}$.

### 4.3.1 Kalman filtering

The Kalman filter ([64, 71]) solves the problem of state estimation in LDSs, where the system consist of a series of continuous latent state variables that are separated by linear state transitions; the state observations are corrupted by Gaussian noise. Here we briefly summarize the main findings from [97]. Let the LDS consists of a series of $T$ latent state variables $c^{(t)}$, $t = 1, \ldots, T$, and observations $y^{(t)}$, $t = 1, \ldots, T$. The factor graph ([77, 90]) associated to this LDS is visualized in Fig. 4.2. The latent states
Figure 4.2: Factor graph message passing algorithm for the estimation of a set of $T$ latent state variables with Markovian transition properties from (possibly noisy) observations.

(denoted by dashed circles) form a Markov chain, meaning that the next state only depends on the current state but not on previous ones. The Kalman filter estimation procedure of the variables $c^{(t)}$, $\forall t$ based on the observations $y^{(t)}$, $\forall t$ (denoted by solid circles) can be formulated as a message-passing algorithm that consists of two phases. First, a forward message passing phase (i.e., the Kalman filtering phase) is performed. Then, using the estimates obtained during the Kalman filtering phase, a backward message passing phase (often referred to as Kalman smoothing or Rauch-Tung-Streibel (RTS) smoothing) is performed.

In the forward message passing phase (see Fig. 4.2), the goal is to estimate latent state variables $c^{(t)}$ based on the previous observations $y^{(1)}, \ldots, y^{(t)}$. In other words, the value of interest is $p(c^{(t)} | y^{(1)}, \ldots, y^{(t)})$, $\forall t$. This quantity can be obtained via a message passing algorithm outlined in Fig. 4.2. Specifically, by starting at $t = 1$, the incoming message to variable node $c^{(1)}$ is given by $\alpha'(c^{(1)}) = p(c^{(1)})$. The outgoing message from variable node $c^{(1)}$ to factor node $p(c^{(2)} | c^{(1)})$ is then given by

$$\alpha(c^{(1)}) = \alpha'(c^{(1)}) p(y^{(1)} | c^{(1)}) = p(c^{(1)}) p(y^{(1)} | c^{(1)}) = b_1 p(c^{(1)} | y^{(1)})$$

according to Bayes rule, where $b_1 = p(y^{(1)})$ is a scaling factor. Recursively following
these rules, the outgoing message $\alpha(c^{(t-1)})$ from variable node $c^{(t-1)}$ to the factor node $p(c^{(t)}|c^{(t-1)})$ at time $t$ is given by

$$\alpha(c^{(t-1)}) = \left(\prod_{r=1}^{t-1} b^{(r)}\right) p(c^{(t-1)}|y^{(1)}, \ldots, y^{(t-1)}).$$

The outgoing message $\alpha'(c^{(t)})$ from factor node $p(c^{(t)}|c^{(t-1)})$ to variable node $c^{(t)}$ is given by

$$\alpha'(c^{(t)}) = \int \alpha(c^{(t-1)}) p(c^{(t)}|c^{(t-1)}) dc^{(t-1)} = \left(\prod_{r=1}^{t-1} b^{(r)}\right) p(c^{(t)}|y^{(1)}, \ldots, y^{(t-1)}).$$

The outgoing message $\alpha(c^{(t)})$ from variable node $c^{(t)}$ is given by

$$\alpha(c^{(t)}) = \alpha'(c^{(t)}) p(y^{(t)}|c^{(t)}) = \left(\prod_{r=1}^{t} b^{(r)}\right) p(c^{(t)}|y^{(1)}, \ldots, y^{(t)}),$$

where $b^{(t)} = p(y^{(t)}|y^{(1)}, \ldots, y^{(t-1)})$. We can see that a scaled version of $\alpha(c^{(t)})$, $\hat{\alpha}(c^{(t)}) = \frac{\alpha(c^{(t)})}{\prod_{r=1}^{t} b^{(r)}} = p(c^{(t)}|y^{(1)}, \ldots, y^{(t)})$, is exactly the value of interest.

The derivations above show that $\hat{\alpha}(c^{(t)})$ can be obtained in recursive fashion via

$$b^{(t)} \hat{\alpha}(c^{(t)}) = p(y^{(t)}|c^{(t)}) \int p(c^{(t)}|c^{(t-1)}) \hat{\alpha}(c^{(t-1)}) dc^{(t-1)}. \quad (4.4)$$

The key to obtaining a tractable and efficient estimator for $p(c^{(t)}|y^{(1)}, \ldots, y^{(t)})$ is that the transition probability $p(c^{(t)}|c^{(t-1)})$ and the observation likelihood $p(y^{(t)}|c^{(t)})$ satisfy certain properties such that the messages $\hat{\alpha}(c^{(t)})$ and $\hat{\alpha}(c^{(t-1)})$ take on the same functional form, just with different parameters. A LDS is a special case in which the transition probability and the observation likelihood are (multivariate) Gaussians of are of the following form:

$$p(c^{(t)}|c^{(t-1)}) = N(c^{(t)}|D_{m(t-1)}c^{(t-1)} + d_{m(t-1)}, \Gamma_{m(t-1)}),$$

$$p(y^{(t)}|c^{(t)}) = N(y^{(t)}|W_{i(t)}c^{(t)}, \Sigma_{i(t)}).$$

Here, $\Gamma_{m(t-1)}$ is the covariance matrix for state transition, $W_{i(t)}$ is the measurement matrix, and $\Sigma_{i(t)}$ is the covariance matrix for the multivariate observation of the system. In order for the functional form of the messages to stay the same over
time, the messages are also Gaussian, i.e., $\tilde{\alpha}(c(t)) \sim \mathcal{N}(c(t) | m(t), V(t))$. Under these conditions, the forward message passing recursion (4.4) takes on a compact form

$$b^{(t)} \tilde{\alpha}(c(t)) = \mathcal{N}(c(t) | m(t), V(t)),$$

(4.5)

with the parameters $b^{(t)}$, $m^{(t)}$ and $V^{(t)}$ given by

$$m^{(t)} = D_{m(t-1)} m^{(t-1)} + d_{m(t-1)} + K^{(t)} \left( y^{(t)} - W_{i(t)} \left( D_{m(t-1)} m^{(t-1)} + d_{m(t-1)} \right) \right),$$

$$V^{(t)} = \left( I - K^{(t)} W_{i(t)} \right) P^{(t-1)},$$

$$b^{(t)} = \mathcal{N} \left( y^{(t)} | W_{i(t)} \left( D_{m(t-1)} m^{(t-1)} + d_{m(t-1)} \right), W_{i(t)} P^{(t-1)} W_{i(t)}^T + \Sigma_{i(t)} \right),$$

in which the matrices $K^{(t)}$ and $P^{(t-1)}$ are given by

$$K^{(t)} = P^{(t-1)} W_{i(t)}^T \left( W_{i(t)} P^{(t-1)} W_{i(t)}^T + \Sigma_{i(t)} \right)^{-1},$$

$$P^{(t-1)} = D_{m(t-1)} V^{(t-1)} D_{m(t-1)}^T + \Gamma_{m(t-1)}.$$  

The recursion starts with a prior $p(c^{(1)}) = \mathcal{N}(c^{(1)} | m^{(0)}, V^{(0)})$, and

$$m^{(1)} = m^{(0)} + K^{(1)} \left( y^{(1)} - W_{i(t)} m^{(0)} \right),$$

$$V^{(1)} = \left( I_K - K^{(1)} W_{i(t)} \right) V^{(0)},$$

$$K^{(1)} = V^{(0)} W_{i(t)}^T \left( W_{i(t)} V^{(0)} W_{i(t)}^T + \Sigma_{i(t)} \right)^{-1},$$

$$b^{(1)} = \mathcal{N} \left( y^{(1)} | W_{i(t)} m^{(0)}, W_{i(t)} V^{(0)} W_{i(t)}^T + \Sigma_{i(t)} \right).$$

We assume the initial prior mean and variance for $c^{(1)}$ to be $m^{(0)} = 0_K$ and $V^{(0)} = \sigma_0^2 I_K$.

### 4.3.2 Kalman smoothing

As detailed above, Kalman filtering can be utilized to obtain $p(c^{(t)} | y^{(1)}, \ldots, y^{(t)})$, an estimate on the latent state at time instance $t$, given all observations $y^{(\tau)}$ for $\tau < t$. This estimate is the value of interest for a variety of real-time tracking applications, since decisions have to be made based on all available observations up to a certain
time instance. However, in our application, one could also use observations at \( \tau \geq t \) to obtain a better estimate of the latent state at time instance \( t \). In other words, the value of interest is now \( p(c(t) | y^{(1)}, \ldots, y^{(T)}) \). In order to estimate this value, a set of backward recursions similar to the set of forward recursions (4.4) can be used.

The backwards message starts with a “one” message going into variable node \( c(T) \):

\[
\beta(c(T)) = 1 \quad \text{(as shown in Fig. 4.2)}.
\]

Then, the outgoing message from variable node \( c(T) \) into factor node \( p(c(T) | c(T-1)) \) is

\[
\beta'(c(T)) = p(y^{(T)} | c(T)),
\]

and the outgoing message from factor node \( p(c(T) | c(T-1)) \) into variable node \( c(T-1) \) is

\[
\beta(c(T-1)) = \int p(c(T) | c(T-1)) p(y^{(T)} | c(T)) dc(T) = p(y^{(T)} | c(T-1)).
\]

Following this convention, we obtain the following recursion:

\[
\beta(c^{(t-1)}) = \int p(c^{(t)}) p(y^{(t)} | c^{(t)}) \beta(c^{(t)}) dc^{(t)} = p(y^{(t)}, \ldots, y^{(T)} | c^{(t-1)}),
\]

where we have implicitly used the Markovian properties of the latent state variables. Now, the marginal distribution of latent state variables \( c^{(t)} \) can be written as a product of the incoming messages into variable node \( c^{(t)} \) from both forward and backward recursions, i.e.,

\[
p(c^{(t)} | y^{(1)}, \ldots, y^{(T)}) = \frac{p(c^{(t)} | y^{(1)}, \ldots, y^{(T)}) p(y^{(t+1)}, \ldots, y^{(T)} | y^{(1)}, \ldots, y^{(t)})}{p(y^{(t+1)}, \ldots, y^{(T)} | y^{(1)}, \ldots, y^{(t)})} = \tilde{\alpha}(c^{(t)}) \tilde{\beta}(c^{(t)}),
\]

where \( \tilde{\beta}(c^{(t)}) = \frac{\beta(c^{(t)})}{\prod_{\tau=t+1}^{T} b^{(\tau)}} \) is a scaled version of \( \beta(c^{(t)}) \). Now, the backward recursion is as follows:

\[
b^{(t)} \tilde{\beta}(c^{(t-1)}) = \int_{c^{(t)}} p(c^{(t)} | c^{(t-1)}) p(y^{(t)} | c^{(t)}) \tilde{\beta}(c^{(t)}) dc^{(t)}.
\] (4.6)

Although it is possible to obtain a backward recursion for \( \tilde{\beta}(c^{(t)}) \), the common approach uses a recursion directly on \( \tilde{\alpha}(c^{(t)}) \tilde{\beta}(c^{(t)}) \) to obtain the value of interest
\( p(c^{(t)}|y^{(1)},\ldots,y^{(T)}) \). By multiplying both sides of the equation (4.6) by \( \tilde{\alpha}(c^{(t-1)}) \), we obtain

\[
\tilde{\alpha}(c^{(t-1)})\tilde{\beta}(c^{(t-1)}) = \tilde{\alpha}(c^{(t-1)}) \int_{c^{(t)}} p(c^{(t)}|c^{(t-1)})p(y^{(t)}|c^{(t)})\frac{\tilde{\alpha}(c^{(t)})\tilde{\beta}(c^{(t)})}{\hat{b}(t)}\tilde{\alpha}(c^{(t)})\text{d}c^{(t)},
\]

which can be computed recursively as a backward message passing process, given the estimates (4.5) following the completion of the forward message passing process detailed in Sec. 4.3.1.

For an LDS, the recursions take the form:

\[
\tilde{\alpha}(c^{(t-1)})\tilde{\beta}(c^{(t-1)}) = \mathcal{N}(c^{(t-1)}|\hat{m}^{(t-1)},\hat{V}^{(t-1)}) \tag{4.7}
\]

with the parameters \( \hat{m}^{(t-1)} \) and \( \hat{V}^{(t-1)} \) given by

\[
\hat{m}^{(t-1)} = m^{(t-1)} + J^{(t-1)}(\hat{m}^{(t)} - \bar{D}_{m(t-1)}m^{(t-1)} - D_{m(t-1)}),
\]
\[
\hat{V}^{(t-1)} = V^{(t-1)} + J^{(t-1)}(\hat{V}^{(t)} - P^{(t-1)}) (J^{(t-1)})^T,
\]
\[
J^{(t-1)} = V^{(t-1)}(\bar{D}_{m(t-1)})^T (P^{(t-1)})^{-1}.
\]

We initialize the recursion with \( \hat{m}^{(T)} = m^{(T)} \) and \( \hat{V}^{(T)} = V^{(T)} \), since \( \beta(c^{(T)}) = 1 \).

In the above derivations, we have assumed that \( y^{(t)} \) is observed for all \( t \). If \( y^{(t)} \) is unobserved, then the message passing scheme will simply have \( \alpha(c^{(t)}) = \alpha'(c^{(t)}) \) and \( \beta'(c^{(t)}) = \beta(c^{(t)}) \) instead, while the rest of the recursions remain unaffected.

### 4.3.3 Approximate Kalman filtering for student concept knowledge tracing

The basic Kalman filtering and smoothing ((4.5) and (4.7)) are only suitable for applications with a Gaussian latent state transition model and a Gaussian observation model, while the forward and backward recursions (4.4) and (4.6) hold for arbitrary state transition and observation models. When attempting to trace latent student concept knowledge states under the SPARFA model, it is not possible to make Gaussian observations of these states. Concretely, we have only binary-valued graded student
responses as our observations. We will now detail approximations that enable the estimation of latent student concept knowledge states under our model.

As introduced in Sec. 5.4.1, the observation model at time $t$ is given by (4.1) and the state transition model is given by (4.3). Therefore, the recursion formula for the forward message passing process (4.4) becomes

$$b^{(t)} \hat{\alpha} (c^{(t)}) = p(Y^{(t)} | c^{(t)}) \int p(c^{(t)} | c^{(t-1)}) \hat{\alpha} (c^{(t-1)}) dc^{(t)}$$

$$= \Phi \left( (2Y^{(t)} - 1)(w^T_{i(t)} c^{(t)} - \mu_{i(t)}) \right) \int \mathcal{N} \left( c^{(t)} | \bar{D}_{m(t-1)} c^{(t-1)} + d_{m(t-1)}, \Gamma_{m(t-1)} \right)$$

$$\mathcal{N} (c^{(t-1)} | m^{(t-1)}, V^{(t-1)}) dc^{(t)}$$

$$= \Phi \left( (2Y^{(t)} - 1)(w^T_{i(t)} c^{(t)} - \mu_{i(t)}) \right) \mathcal{N} (c^{(t)} | \bar{D}_{m(t-1)} m^{(t-1)} + d_{m(t-1)},$$

$$\bar{D}_{m(t-1)} V^{(t-1)} \bar{D}^T_{m(t-1)} + \Gamma_{m(t-1)} \right)$$

$$= \Phi \left( (2Y^{(t)} - 1)(w^T_{i(t)} c^{(t)} - \mu_{i(t)}) \right) \mathcal{N} (c^{(t)} | \bar{m}^{(t)}, \bar{V}^{(t)}) ,$$

(4.8)

where we used a "tilde" to denote the mean and covariance of the messages $\alpha'(c^{(t-1)})$.

Equation (4.8) shows that, $\hat{\alpha} (c^{(t)})$ is no longer Gaussian even if $\hat{\alpha} (c^{(t-1)})$ is Gaussian, under the probit binary observation model. Thus, the closed-form updates in (4.5) and (4.7) can no longer be applied. Therefore, we have to perform an approximate message passing approach within the Kalman filtering framework to arrive at a tractable estimator of $c^{(t)}$. A number of approaches has been proposed to approximate $\hat{\alpha} (c^{(t)})$ by a Gaussian distribution $\mathcal{N} (c^{(t)} | \bar{m}^{(t)}, \bar{V}^{(t)})$; here, the "bar" on the variables denote the means and covariances of the approximated Gaussian messages. These approaches include the extended Kalman filter (EKF) ([48, 68, 93]), which uses a linear approximation of the likelihood term around the point $\bar{m}^{(t)}$, and thus reduce the non-Gaussian observation model to a Gaussian one; the unscented Kalman filter (UKF) ([70, 147]), which uses the unscented transform (UT) to create a set of sigma vectors from $p(c^{(t-1)})$ and uses them to approximate the mean and covariance of $\hat{\alpha} (c^{(t)})$ after the non-Gaussian observation; and Laplace approximations ([117, 157]), which use an iterative algorithm to find the mode of $\hat{\alpha} (c^{(t)})$ and the Hessian at the mode to approximate the mean and covariance of the approximated Gaussian messages. We
will employ an approximation approach introduced in the expectation propagation (EP) literature ([98]).

It is known that the specific values for $m(t)$ and $V(t)$ that minimize the Kullback-Leibler (KL) divergence between $N(c(t)|\tilde{m}(t), \tilde{V}(t))$ and a target distribution $q(c)$ are the first and second moments of $q(c)$ [117]. Fortunately, for the probit observation model $p(Y(t) | c(t)) = \Phi((2Y(t) - 1)(w_{i(t)}^Tc(t) - \mu_i))$, $\tilde{m}(t)$, $\tilde{V}(t)$ and $b(t)$ have closed-form expressions (see Sec. 4.7 for the details):

$$m(t) = \tilde{m}(t) + (2Y(t) - 1) \frac{\tilde{V}(t)w_{i(t)}}{\sqrt{1 + w_{i(t)}^T\tilde{V}(t)w_{i(t)}}} \frac{\mathcal{N}(z)}{\Phi(z)},$$

$$V(t) = \tilde{V}(t) - \frac{\tilde{V}(t)w_{i(t)}w_{i(t)}^T\tilde{V}(t)}{1 + w_{i(t)}^T\tilde{V}(t)w_{i(t)}} \left( z + \frac{\mathcal{N}(z)}{\Phi(z)} \right) \frac{\mathcal{N}(z)}{\Phi(z)},$$

$$b(t) = \Phi(z), \quad (4.9)$$

with

$$z = (2Y(t) - 1) \frac{w_{i(t)}^T\tilde{m}(t) - \mu_i}{\sqrt{1 + w_{i(t)}^T\tilde{V}(t)w_{i(t)}}},$$

and $\tilde{m}(t)$ and $\tilde{V}(t)$ as given by (4.8).

SPARFA naturally supports two different inverse link functions for analyzing binary-valued graded student responses: the inverse probit link function and the inverse logit link function. In this application, the inverse probit link function is preferred over the inverse logit link function, due to the existence of the closed-form first and second moments described above. The inverse logit link function is not preferred as such convenient closed-form expressions do not exist. Therefore, we will focus on the inverse probit link function in the sequel.

Armed with the efficient approximation (6.3), the forward Kalman filtering message passing scheme described in Sec. 4.3.1 can be applied to the problem at hand; the backward Kalman smoothing message passing scheme described in Sec. 4.3.2 remains unchanged. Using these recursions, estimates of the desired quantities
47

\( p(c^{(t)} | y^{(1)}, \ldots, y^{(T)}) \) can be computed efficiently, providing a way for student concept knowledge tracing under the model (4.1).

4.4 Content Analytics

So far, we have described an approximate Kalman filtering and smoothing approach for student concept knowledge tracing, i.e., to estimate 

\[ p(c^{(t)}_{j} | y^{(1)}_{j}, \ldots, y^{(T)}_{j}) \quad \forall t, j. \]

The method proposed in Sec. 4.3 is only able to provide these estimates if the observed binary graded student responses \( Y^{(t)}_{j} \), \( \forall t, j \), and all student initial knowledge parameters \( m^{(0)}_{j}, V^{(0)}_{j}, \forall j \), all student concept knowledge state transition parameters \( D_{m}, d_{m}, \) and \( \Gamma_{m}, \forall m \), and all question parameters, \( w_{i} \) and \( \mu_{i}, \forall i \), are given a priori.

However, in a typical PLS, these parameters are unknown, in general, and need to be estimated from the observed data. We now detail a set of convex optimization-based techniques to estimate the parameters \( m^{(0)}_{j}, V^{(0)}_{j}, \forall j, D_{m}, d_{m}, \) and \( \Gamma_{m}, \forall m, \) and \( w_{i}, \mu_{i}, \forall i, \) given the estimates of the latent student concept knowledge states \( c^{(t)}_{j} \) obtained from the approximate Kalman filtering approach described in Sec. 4.3. Since the estimates of \( c^{(t)}_{j} \) are distributions rather than point estimates, SPARFA-Trace jointly traces student concept knowledge and estimates student, learning resource, and question-dependent parameters, using an expectation-maximization (EM) approach.

4.4.1 SPARFA-Trace: An EM algorithm for parameter estimation

EM has been widely used in the Kalman filtering framework to estimate the parameters of interest in the system (see [17, Chap. 13] and [64] for more details) due to numerous practical advantages ([123]). SPARFA-Trace performs parameter estimation in an iterative fashion in the EM framework. All parameters are initialized to random initial values, and then each iteration of the algorithm consist of two phases: (i) the current parameter estimates are used to estimate the latent state distributions 

\[ p(c^{(t)}_{j} | y^{(1)}_{j}, \ldots, y^{(T)}_{j}) \quad \forall t, j; \] (ii), these latent state estimates are then used to maximize
the expected joint log-likelihood of all the observed and latent state variables, i.e.,

\[
\maximize \sum_{j=1}^{N} \mathbb{E}_{c_j^{(1)}} \left[ \log p(c_j^{(1)} | m_j(0), V_j(0)) \right] 
\]

\[
+ \sum_{t=2}^{T} \sum_{j=1}^{N} \mathbb{E}_{c_j^{(t-1)}, c_j^{(t)}} \left[ \log p(c_j^{(t)} | c_j^{(t-1)}, D_{m_j^{(t-1)}}, d_{m_j^{(t-1)}}, \Gamma_{m_j^{(t-1)}}) \right] 
\]

\[
+ \sum_{(t,j) \in \Omega_{\text{obs}}} \mathbb{E}_{c_j^{(t)}} \left[ \log p(Y_j^{(t)} | c_j^{(t)}, w_{ij}^{(t)}, \mu_{ij}^{(t)}) \right],
\]

in order to obtain new (and hopefully improved) parameter estimates. SPARFA-Trace alternates between these two phases until convergence, i.e., a maximum number of iterations is reached or the change in the estimated parameters between two consecutive iterations falls below a given threshold.

### 4.4.2 Estimating the initial student knowledge parameters

We start with the estimation method for the student initial knowledge parameters \(m_j(0), V_j(0), \forall j\). To this end, we minimize the expected negative log-likelihood for the \(j^{th}\) student

\[
\mathbb{E}_{c_j^{(1)}} [- \log p(c_j^{(1)} | m_j(0), V_j(0))] = \frac{1}{2} \log |V_j(0)| 
\]

\[
+ \mathbb{E}_{c_j^{(1)}} \left[ \frac{1}{2} (c_j^{(1)} - m_j(0))(V_j(0))^{-1}(c_j^{(1)} - m_j(0))^T \right],
\]

where \(|V_j(0)|\) denotes the determinant of the covariance matrix \(V_j(0)\). Since we do not impose constraints on \(m_j(0)\) and \(V_j(0)\), these estimates can be obtained as

\[
m_j(0) = \mathbb{E}_{c_j^{(1)}} [c_j^{(1)}] = \hat{m}_j^{(1)} \quad \text{and} \quad V_j(0) = \mathbb{E}_{c_j^{(1)}} [(c_j^{(1)} - \hat{m}_j^{(1)})(c_j^{(1)} - \hat{m}_j^{(1)})^T] = \hat{V}_j^{(1)},
\]

where the estimates \(\hat{m}_j^{(1)}\) and \(\hat{V}_j^{(1)}\) are obtained from the Kalman smoothing recursions (4.7) in Sec. 4.3.2.

### 4.4.3 Estimating the student concept knowledge state transition parameters

Next we estimate the latent student concept knowledge state transition (i.e., learning resource) parameters \(D_m, d_m, \Gamma_m, \forall m\). To this end, define \(M^m\) as the set
containing time and student indices \((t, j)\), indicating that student \(j\) studies the \(m^\text{th}\) learning resource between time instances \(t - 1\) and \(t\). With this definition, we aim to minimize the expected negative log-likelihood

\[
\sum_{t, j: (t, j) \in M^m} \mathbb{E}_{c_j^{(t-1)}, c_j^{(t)}}[- \log p(c_j^{(t)} | c_j^{(t-1)}, D_m, d_m, \Gamma_m)]
\]

\[
= \sum_{t, j: (t, j) \in M^m} \left( \frac{1}{2} \log |\Gamma_m| + \mathbb{E}_{c_j^{(t-1)}, c_j^{(t)}} \left[ \frac{1}{2} (c_j^{(t)} - c_j^{(t-1)} - D_m c_j^{(t-1)} - d_m)^T \Gamma_m^{-1} (c_j^{(t)} - c_j^{(t-1)} - D_m c_j^{(t-1)} - d_m) \right] \right)
\]

subject to the assumptions (A4)–(A6). We start by estimating \(D_m\) and \(d_m\) given \(\Gamma_m\), and then use these estimates to estimate \(\Gamma_m\). In order to induce sparsity on \(D_m\) to take (A6) into account, we impose an \(\ell_1\)-norm penalty on \(D_m\), which is defined as the sum of the absolute values of all entries of \(D_m\) ([63]). Taking only the terms containing \(D_m\) and \(d_m\), we can formulate the following augmented optimization problem:

\[
(P_d) \quad \text{minimize}_{D_m \in L^+, d_m} \sum_{t, j: (t, j) \in M^m} \mathbb{E}_{c_j^{(t-1)}, c_j^{(t)}} \left[ (\tilde{D}_m c_j^{(t-1)})^T \Gamma_m^{-1} (\tilde{D}_m c_j^{(t-1)}) - (c_j^{(t)} - c_j^{(t-1)})^T \Gamma_m^{-1} (c_j^{(t)} - c_j^{(t-1)}) + \gamma \|D_m\|_1 \right],
\]

where \(L^+\) denotes the set of lower-triangular matrices with non-negative entries. For notational simplicity, we have written \([D_m \ d_m]\) as \(\tilde{D}_m\). We also write the augmented latent state vectors \([(c_j^{(t-1)})^T \ 1]^T\) as \(\tilde{c}_j^{(t-1)}\), when multiplied by \(\tilde{D}_m\), correspondingly. Note that the \(\ell_1\)-norm penalty only applies to the matrix \(D_m\) in the used notation.

The problem \((P_d)\) is convex in \(\tilde{D}_m\), and hence, can be solved efficiently. In particular, we use the fast iterative shrinkage and thresholding algorithm (FISTA) framework ([15]). The FISTA algorithm starts with a random initialization of \(\tilde{D}_m\) and iteratively updates \(\tilde{D}_m\) until a maximum number of iterations \(\ell_{\max}\) is reached or the change in the estimate of \(\tilde{D}_m\) between two consecutive iterations falls below a certain threshold. In each iteration \(\ell = 1, 2, \ldots, L_{\max}\), the algorithm performs two
steps. First, a gradient step that aims to lower the cost function performs

\[
\tilde{D}_{m+1}^{\ell} \leftarrow \tilde{D}_m^{\ell} - \eta_{\ell} \nabla f(\tilde{D}_m), \tag{4.11}
\]

where \( f(\tilde{D}_m) \) corresponds to the differentiable part of the cost function (excluding the \( \ell_1 \)-norm penalty) in (P_d). The quantity \( \eta_{\ell} \) is a step size parameter for iteration \( \ell \). For simplicity, we will take \( \eta_{\ell} = 1/L \) in all iterations, where \( L \) is the Lipschitz constant given by

\[
L = \sigma_{\max} \left( \sum_{t,j; (t,j) \in M^m} \|E_{j,(t-1),j}^{(t)} [(c_j^{(t)} - c_j^{(t-1)})(c_j^{(t-1)})^T] \| \right) \sigma_{\max}(|\mathcal{M}^m| \Gamma_m^{-1}).
\]

Here \( \sigma_{\max}(\cdot) \) denotes the maximum singular value of a matrix, and \( |\mathcal{M}^m| \) denotes the cardinality of the set \( \mathcal{M}^m \). The gradient \( \nabla f(\tilde{D}_m) \) in (4.11) is given by

\[
\nabla f(\tilde{D}_m) = -\Gamma_m^{-1} \sum_{t,j; (t,j) \in M^m} \left( E_{j,(t-1),j}^{(t)} [(c_j^{(t)} - c_j^{(t-1)})(c_j^{(t-1)})^T] \right)
\]

\[
= -\Gamma_m^{-1} \sum_{t,j; (t,j) \in M^m} \left( J_j^{(t-1)} \tilde{V}_j^{(t)} + \tilde{m}_j^{(t)} (\tilde{m}_j^{(t-1)})^T - \tilde{V}_j^{(t-1)} - \tilde{m}_j^{(t-1)} (\tilde{m}_j^{(t-1)})^T 
\right)
\]

\[
\tilde{m}_j^{(t)} - \tilde{m}_j^{(t-1)} \right) \right)
\]

The parameters \( J_j^{(t-1)} \), \( \tilde{m}_j^{(t-1)} \), \( \tilde{m}_j^{(t)} \), \( \tilde{V}_j^{(t-1)} \), and \( \tilde{V}_j^{(t)} \) are obtained from the backward recursions in (4.7). Next, the FISTA algorithm performs a projection step, which takes into account the sparsifying regularizer \( \gamma \|D_m\|_1 \), and the assumptions (A4) and (A5):

\[
\tilde{D}_m^{\ell+1} \leftarrow P_{\mathcal{L}^+}(\max(\tilde{D}_m^{\ell+1} - \gamma \eta_{\ell}, 0)), \tag{4.12}
\]

where \( P_{\mathcal{L}^+}(\cdot) \) corresponds to the projection onto the set of lower-triangular matrices by setting all entries in the upper triangular part of \( D_m^{\ell+1} \) to zero. The maximum operator operates element-wise on \( D_m^{\ell+1} \). The updates (4.11) and (4.12) are repeated until convergence, eventually providing a new estimate \( \tilde{D}_m^{\text{new}} \) for \([D_m \ d_m]\).
Using these new estimates, the update for $\Gamma_m$ can be computed in closed form:

$$\Gamma_{m}^{\text{new}} = \frac{1}{|\mathcal{M}|} \sum_{t,j:(t,j) \in \mathcal{M}} \left( \mathbb{E}_{c_j^{(t)}}[c_j^{(t)}(c_j^{(t)})^T] - \tilde{D}_m^{\text{new}} \mathbb{E}_{c_j^{(t-1)}}[c_j^{(t-1)}(c_j^{(t)})^T] \right)$$

$$- \mathbb{E}_{c_j^{(t-1)}}[c_j^{(t)}(c_j^{(t-1)})^T](\tilde{D}_m^{\text{new}})^T + (\tilde{D}_m^{\text{new}}) \mathbb{E}_{c_j^{(t-1)}}[c_j^{(t-1)}(c_j^{(t-1)})^T](\tilde{D}_m^{\text{new}})^T$$

$$= \frac{1}{|\mathcal{M}|} \sum_{t,j:(t,j) \in \mathcal{M}} \left( \tilde{V}_j^{(t)} + \tilde{m}_j^{(t)}(\tilde{m}_j^{(t)})^T - \tilde{D}_m^{\text{new}} \right)$$

$$- \left[ J_j^{(t-1)} \tilde{V}_j^{(t)} + \tilde{m}_j^{(t-1)}(\tilde{m}_j^{(t-1)})^T \right](\tilde{D}_m^{\text{new}})^T$$

$$+ \tilde{D}_m^{\text{new}} \left( \tilde{V}_j^{(t-1)} + \tilde{m}_j^{(t-1)}(\tilde{m}_j^{(t-1)})^T - 1 \right)(\tilde{D}_m^{\text{new}})^T.$$  

### 4.4.4 Estimating the question-dependent parameters

We next show how to estimate the question-dependent parameters $w_i$, $\mu_i$, $\forall i$. To this end, we define $Q^i$ as the collection set of time and student indices $(t, j)$ that student $j$ answered the $i^{th}$ question at time instance $t$. We then minimize the expected negative log-likelihood of all the observed binary-valued graded student responses (4.1) for the $i^{th}$ question subject to assumptions (A2) and (A3) on the question–concept association vector $w_i$. In order to impose sparsity on $w_i$, we add an $\ell_1$-norm penalty to the cost function, which leads to the following optimization problem:

$$(P_w) \quad \text{minimize} \quad \sum_{w_i:|w_i,k| \geq 0, \forall k} \mathbb{E}_{c_j^{(t)}}[-\log \Phi((2Y_j^{(t)} - 1)(w_i^T c_j^{(t)} - \mu_i))] + \lambda \|w_i\|_1.$$ 

This problem corresponds to the (RR$^+_L$) problem of SPARFA detailed in [81], where the point estimates of $c_j$ are given and the problem is convex in $w_i$. In particular, given the distribution $c_j^{(t)} \sim \mathcal{N}(c_j^{(t)} | \tilde{m}_j^{(t)}, \tilde{V}_j^{(t)})$, $(P_w)$ is still convex in $w_i$, thanks to the linearity of the expectation operator. However, the inverse probit link function prohibits us from obtaining a simple form of this expectation. In order to develop a tractable algorithm to approximately solve this problem, we utilize the unscented transform (UT) ([147]) to approximate the cost function of $(P_w)$. 

The UT is commonly used in the Kalman filtering literature to approximate the statistics of a random variable undergoing a non-linear transformation. Specifically, given a $K$-dimensional random variable $x$ with known mean and covariance and a non-linear function $g(\cdot)$, the UT generates a set of $2K + 1$ so-called sigma vectors $\{X_n\}$ and a set of corresponding weights $\{u_n\}$ as detailed in [147, Eq.15], in order to approximate the mean and covariance of the vector $y = g(x)$. As shown in [147], this approximation is accurate up to the third order for Gaussian distributed random vectors $x$.

Following the paradigms of the UT, we generate a set of sigma vectors $\{(\tilde{c}_j(t))_n\}$ and a corresponding set of weights $\{u_n\}$, $n \in \{1, \ldots, 2K + 1\}$, for each latent state vector $c_j(t)$, given the mean $\hat{m}_j(t)$ and covariance $\hat{V}_j(t)$. For computational simplicity, we will use the same set of weights for all latent state vectors $c_j(t)$. The optimization problem ($P_w$) can now be approximated by

$$
\min_{\mathbf{w}_i; w_{i,k} \geq 0, \forall k} \sum_{n=1}^{2K+1} u_n \left( -\log \Phi((2Y_j(t) - 1)(\mathbf{w}_i^T(\tilde{c}_j(t))_n - \mu_i)) \right) + \lambda \|\mathbf{w}_i\|_1,
$$

which, once again, can be solved efficiently by using the FISTA framework. The resulting iterative procedure performs two steps in each iteration $\ell$: First, a gradient step that aims at lowering the cost function performs

$$
\mathbf{w}^{\ell+1}_i \leftarrow \mathbf{w}_i - \eta_\ell \nabla f(\mathbf{w}_i),
$$

(4.13)

where $f(\mathbf{w}_i)$ corresponds to the differentiable portion (excluding the $\ell_1$-norm penalty part) of the cost function in ($P_w$). The gradient $\nabla f(\mathbf{w}_i)$ is given by $\nabla f(\mathbf{w}_i) = -\tilde{C}_i \tilde{r}_i$, where $\tilde{r}_i$ is a $(2K + 1)|Q| \times 1$ vector $\mathbf{r}_i = [\mathbf{a}_1^q, \ldots, \mathbf{a}_{|Q|}^q]^T$. The vector $\mathbf{a}_i^q$ is defined by $\mathbf{a}_i^q = [(g_i^q)_1, \ldots, (g_i^q)_{2K+1}]$, where

$$
(g_i^q)_n = u_n 2(Y_{j_1}^{(t_q)} - 1) \frac{\mathcal{N}(2(Y_{j_1}^{(t_q)} - 1)\mathbf{w}_i^T(\tilde{c}_j(t))_n)}{\Phi(2(Y_{j_1}^{(t_q)} - 1)\mathbf{w}_i^T(\tilde{c}_j(t))_n)},
$$

in which $(t_q, j_q)$ represents the $q$th time–student index pair in $Q$. The $K \times (2K + 1)|Q|$ matrix $\tilde{C}_i$ is defined as $\tilde{C}_i = [(G_i)_1, \ldots, (G_i)|Q|]$, where the $K \times (2K + 1)$ matrix
$(G_i)_q$ is given by

$$(G_i)_q = \begin{bmatrix} (c_{j_q}^{(t_q)})_1, \ldots, (c_{j_q}^{(t_q)})_{2K+1} \end{bmatrix}. \tag{4.13}$$

The quantity $\eta_\ell$ is a step size parameter for iteration $\ell$. For simplicity, we will take $\eta_\ell = 1/L$ in all iterations, where $L$ is the Lipschitz constant given by $L = \sigma_{\text{max}}(\tilde{C}_i) \sigma_{\text{max}}(\tilde{C}_i')$, where $\tilde{C}_i'$ is a $K \times (2K + 1)|Q|$ matrix defined as $\tilde{C}_i' = [(G_i')_1, \ldots, (G_i')_{|Q'|}]$, where the $K \times (2K + 1)$ matrix $(G_i')_q$ is given by

$$(G_i')_q = \begin{bmatrix} u_1(c_{j_q}^{(t_q)})_1, \ldots, u_{2K+1}(c_{j_q}^{(t_q)})_{2K+1} \end{bmatrix}. \tag{4.14}$$

Next, the FISTA algorithm performs a projection step, which takes into account $\lambda \|w_i\|_1$ and the assumption (A3):

$w_i^{\ell+1} \leftarrow \max\{\tilde{w}_i^{\ell+1} - \lambda \eta_\ell, 0\}. \tag{4.14}$

The steps (4.13) and (4.14) are repeated until convergence, providing a new estimate $w_i^{\text{new}}$ of the question–concept association vector $w_i$. For simplicity of exposition, the question intrinsic difficulties $\mu_i$ are omitted in the derivations above, as they can be included as an additional entry in $w_i$ as $[w_i^T \mu_i]^T$; the corresponding latent student concept knowledge state vectors $c_j^{(t)}$ are augmented as $[(c_j^{(t)})^T 1]^T$.

### 4.5 Experimental Results

We now demonstrate the efficacy of SPARFA-Trace on real-world educational datasets. We first compare SPARFA-Trace against two established methods on predicting unobserved binary-valued student response data, namely knowledge tracing (KT) ([37, 106]) and SPARFA ([81]). Then, we show how SPARFA-Trace is able to visualize students’ concept knowledge state evolution over time, and the learning resource and question quality and their content organization. For all the synthetic and real data experiments shown next, the regularization parameters $\lambda$, $\gamma$, and $\sigma_0^2$ are chosen via cross-validation ([63]), and all experiments are repeated for 25 independent Monte–Carlo trials for each instance of the model parameter we control.
4.5.1 Predicting responses for new students

We now compare SPARFA-Trace against the KT method described in [106] for predicting responses for new students that do not have previous recorded response history.

Dataset: The dataset we use for this experiment is the computer engineering course dataset, referred to as “UG” in the previous experiments. This dataset consists of the binary-valued graded response from 92 students answering 203 questions, with 99.5% of the responses observed. Since the KT implementation of [106] is unable to handle missing data, we removed students that do not answer every question from the dataset, resulting in a pruned dataset of 73 students. The course is organized into three independent sections: The first section is on digital logic, the second on data structures, and the third on basic programming concepts. The full course consist of 11 assessments, including 8 homework assignments and an exam at the end of each section; we assume that the students’ concept knowledge state transitions can only happen between two consecutive assignments/exams, due to their interaction with all the lectures/readings/exercises.

Experimental setup: Since KT is only capable of handling educational datasets that involve a single concept, we partition the UG dataset into three parts, with each part corresponding to one of the three independent sections. We run KT independently on the three parts, and aggregate the prediction results. (We also ran KT on the entire the UG dataset without partitioning it into 3 independent sections. The results obtained were inferior to those obtained by running KT on 3 independent sections.) We initialize the four parameters of KT (student prior, learning probability, guessing probability, slipping probability) with the best initial value we find over 5 different initializations. For SPARFA-Trace, we use $K = 3$, with each concept corresponding to one section of the dataset. In order to alleviate the identifiability issue in our model,
we initialize the algorithm with $w_{i,k} = 1$ where question $i$ is in section $k$ and $w_{i,k} = 0$ otherwise. We also initialize the matrices $D_m$ with identity matrices $I_{3\times 3}$, the vectors $d_m$ with zero vectors, and covariance matrices $\Gamma_m$ with identity matrices.

For cross-validation, we randomly partition the UG dataset into 5 folds, with each fold consisting of $1/5$ of the students answering all questions. Four folds of the data are used as the training set and the other fold is used as the test set. We train both KT and SPARFA-Trace on the training set and obtain estimates on all student, learning resource and question-dependent parameters, and test their prediction performances on the test set. For previously unobserved new students in the test set, both algorithms make the first prediction of $Y_j^{(1)}$ at $t = 1$ using question-dependent parameters estimated from the training set. As time goes on, more and more observed responses $Y_j^{(t)}$ are available to both algorithms, and they use these responses to make future predictions.

We compare both algorithms on three metrics: prediction accuracy, prediction likelihood, and area under the receiver operation characteristic (ROC) curve. The prediction accuracy corresponds to the percentage of correctly predicted responses; the prediction likelihood corresponds to the average the predicted likelihood of the unobserved responses, i.e.,

$$\frac{1}{|\Omega_{\text{obs}}|} \sum_{t,j: (t,j) \in \Omega_{\text{obs}}} p(Y_j^{(t)} | w_{i,j}^{(t)}, c_j^{(t)})$$

where $\Omega_{\text{obs}}$ is the set of student responses in the test set; the area under the ROC curve is a commonly-used performance metric for binary classifiers (see [106] for details). The area under the ROC curve always is always between 0 and 1, with a larger value representing higher classification accuracy.

Since SPARFA-Trace does not provide point estimates of $c_j^{(t)}$ but rather their distributions, we compute the predicted likelihood of unobserved responses by:

$$\mathbb{E}_{c_j^{(t)}} \left[ p(Y_j^{(t)} | w_{i,j}^{(t)}, c_j^{(t)}) \right] = \Phi \left( \frac{2Y_j^{(t)} - 1}{\sqrt{1 + w_{i,j}^{(t)} V_j^{(t)}}} \right).$$
Table 4.1: Comparisons of SPARFA-Trace against knowledge tracing (KT) on predicting responses for new students using the UG dataset. SPARFA-Trace outperforms KT on all three metrics.

<table>
<thead>
<tr>
<th>Performance metric</th>
<th>KT</th>
<th>SPARFA-Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction accuracy</td>
<td>86.42 ± 0.16%</td>
<td>87.49 ± 0.12%</td>
</tr>
<tr>
<td>Prediction likelihood</td>
<td>0.7718 ± 0.0011</td>
<td>0.8128 ± 0.0044</td>
</tr>
<tr>
<td>Area under the ROC curve</td>
<td>0.5989 ± 0.0056</td>
<td>0.8157 ± 0.0028</td>
</tr>
</tbody>
</table>

**Results:** The means and standard deviations of all three metrics covering multiple cross-validation trials are shown in Tbl. 4.1. We can see that SPARFA-Trace outperforms KT on all performance metrics for the UG dataset. We also emphasize that SPARFA-Trace is capable of achieving superior prediction performance while simultaneously estimating the quality and content organization parameters of all learning resources and questions.

### 4.5.2 Predicting unobserved student responses

It has been shown ([56]) that collaborative filtering methods often outperform KT in predicting unobserved student responses, even though they ignore any temporal evolution aspects of the dataset. Hence, we compare SPARFA-Trace against the original SPARFA framework ([81]), which offers state-of-the-art collaborative filtering performance on predicting unobserved student responses.

**Datasets:** We will use two datasets in this experiment. The first dataset is the full UG dataset with 92 students answering 203 questions, explained in Sec. 4.5.1. The second dataset we use is from a signals and systems undergraduate course, consisting of 41 students answering 143 questions, with 97.1% of the responses observed. We will refer to this dataset as “SS” in the following experiments. All the questions were
manually labeled with a number of $K = 4$ concepts, with the concepts being listed in Fig. 4.5(b). The full course consist of 14 assessments, including 12 assignments and 2 exams; we will treat all the lectures/readings/exercises the students interact with between two consecutive assignments/exams as an learning resource.

**Experimental setup:** We randomly partition the $143 \times 43$ (or $203 \times 92$) matrix $Y$ of observed graded student responses into 5 folds for cross-validation. Four folds of the data are used as the training set and the other fold is used as the test set. We train both the probit variant of SPARFA-M and SPARFA-Trace on the training set to estimate the student concept knowledge states and the student, learning resource and question-dependent parameters, and then use these estimates to predict unobserved held-out responses in the test set.

**Results:** The means and standard deviations of the prediction accuracy and prediction likelihood metrics covering multiple cross-validation trials are shown in Tables 2 and 3. We see that SPARFA-Trace achieves comparable prediction performance to SPARFA-M on both datasets, although the datasets are treated as if they do not have time-varying effects. We emphasize that, in addition to providing competitive prediction performance, SPARFA-Trace is capable of (i) tracing student concept knowledge evolution over time and (ii) analyzing learning resource and question qualities and their content organization. This extracted information is very important as it allow a PLS to provide timely feedback to students about their strengths and weaknesses, and to automatically recommend learning resources to students for remedial studies based on their qualities and contents.

4.5.3 Visualizing time-varying learning and content analytics

In this section, we showcase another advantage of SPARFA-Trace over existing KT and collaborative filtering methods, i.e., the visualization of both student knowledge
Table 4.2: Comparisons of SPARFA-Trace against SPARFA-M on predicting unobserved student responses for the UG dataset.

<table>
<thead>
<tr>
<th></th>
<th>SPARFA-M</th>
<th>SPARFA-Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction accuracy</td>
<td>87.10 ± 0.04%</td>
<td>87.31 ± 0.05%</td>
</tr>
<tr>
<td>Prediction likelihood</td>
<td>0.7274 ± 0.0005</td>
<td>0.7295 ± 0.0007</td>
</tr>
</tbody>
</table>

Table 4.3: Comparisons of SPARFA-Trace against SPARFA-M on predicting unobserved student responses for the SS dataset.

<table>
<thead>
<tr>
<th></th>
<th>SPARFA-M</th>
<th>SPARFA-Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction accuracy</td>
<td>86.64 ± 0.14%</td>
<td>86.29 ± 0.25%</td>
</tr>
<tr>
<td>Prediction likelihood</td>
<td>0.7037 ± 0.0024</td>
<td>0.7066 ± 0.0028</td>
</tr>
</tbody>
</table>

State evolution over time and the estimated learning resource and question quality and content organization.

**Visualizing student concept knowledge state evolution:** Fig. 4.3(a) shows the estimated latent student concept knowledge states at all time instances for Student 1 in the UG dataset. We can see that their knowledge on Concepts 2 and 3 gradually improve over time, while their knowledge on Concept 1 does not. Therefore, recommending Student 1 remedial material on Concept 1 seems necessary, which is verified by the fact that Student 1 often responds incorrectly on questions covering Concept 1 towards the end of the course.

Fig. 4.3(b) shows the average student concept knowledge states over the entire class at all time instances for the UG dataset. Since Concept 1 is the basic concept that is covered in the early stages of the course, we can see that its mean knowledge among all students increases in early stages of the course and then remain constant.
Figure 4.3: Estimated latent student concept knowledge states for all time instances, for the UG dataset. (a) Student 1’s latent concept knowledge state evolution; (b) Average student latent concept knowledge states evolution.

...afterwards. In contrast, Concept 3 is the most advanced concept covered near the end of the course, and the improvement in which is not obvious until very late stages of the course. Hence, SPARFA-Trace can enable a PLS to provide timely feedback to individual students on their concept knowledge at all times, which reveals the learning progress of the students. SPARFA-Trace can also inform instructors on the trend of the concept knowledge state evolution of the entire class, in order to help them make timely adjustments to their course plans.

Visualizing learning resource quality and content: Fig. 4.4(a) and Fig. 4.4(b) show the quality and content organization of learning resources 3 and 9 for the SS dataset. These figures visualize the leaners’ concept knowledge state transitions induced by interacting with learning resources 3 and 9. Circular nodes represent concepts; the leftmost set of dashed nodes represent the concept knowledge state vector $c^{(t-1)}$, which are the students’ concept knowledge states before interacting with these learning resources, and the rightmost set of solid nodes represent the concept knowledge state vector $c^{(t)}$, which are the students’ concept knowledge states...
Figure 4.4: Visualized student knowledge state transition effect of two distinct learning resources for the SS dataset. (a) Student knowledge state transition effect for Learning resource 3; (b) Student knowledge state transition effect for Learning resource 9.

after interacting with these learning resources. Arrows represent the the student concept knowledge state transition matrix $D_m$, the intrinsic quality vector of the learning resource $d_m$, and their transformation effects on students’ concept knowledge states. Dotted arrows represent unchanged student concept knowledge states; these arrows correspond to zero entries in $D_m$ and $d_m$. Solid arrows represent the intrinsic knowledge gain of some concepts, characterized by large, positive entries in $d_m$. Dashed arrows represent the change in knowledge of advanced concepts due to their pre-requisite concepts, characterized by non-zero entries in $D_m$: High knowledge level on pre-requisite concepts can result in improved understanding and an increase on knowledge of advanced concepts, while low knowledge level on these pre-requisite concepts can result in confusion and a decrease on knowledge of advanced concepts.

As shown in Fig. 4.4(a), Learning resource 3 is used in early stage of the course, and we can see that this learning resource gives the students’ a positive knowledge gain of Concept 2, while also helping on the more advanced Concepts 3 and 4. As shown in Fig. 4.4(b), Learning resource 9 is used in later stage of the course, and we can see that it uses the students’ knowledge on all previous concepts to improve their knowledge on Concept 4, while also providing a positive knowledge gain on Concepts 3 and 4.
By analyzing the content organization of learning resources and their effects on student concept knowledge state transitions, SPARFA-Trace enables a PLS to automatically recommend corresponding learning resources to students based on their strengths and weaknesses. The estimated learning resource quality information also helps course instructors to distinguish between effective learning resources, and poorly-designed, off-topic, or misleading learning resources, thus helping them to manage these learning resources more easily.

**Visualizing question quality and content:** Fig. 4.5 shows the question–concept association graph obtained from the SS dataset. Circle nodes represent concept nodes, while square, box nodes represent question nodes. Each question box is labeled with the time instance at which it is assigned and its estimated intrinsic difficulty. From the graph we can see time-evolving effects, as questions assigned in the early stages of the course cover basic concepts (Concepts 1 and 2), while questions assigned in later stages cover more advanced concepts (Concepts 3 and 4). Some questions are associated with multiple concepts, and they mostly correspond to the final exam questions (boxes with dashed boundaries) where the entire course is covered.

Thus, by estimating the intrinsic difficulty and content organization of each question, SPARFA-Trace allows a PLS to generate feedback to instructors on the underlying knowledge structure of questions, which enables them to identify ill-posed or off-topic questions (such as questions that are not associated to any concepts in Fig. 4.5(a)).

### 4.6 Conclusions

We have proposed SPARFA-Trace, a novel, message passing-based approximate Kalman filtering approach for time-varying learning and content analytics. The proposed method jointly traces latent student concept knowledge and simultaneously estimates the quality and content organization of the corresponding learning resources (such as textbook sections or lecture videos), and the questions in assessment sets. In order to
(b)

Figure 4.5: (a) Question–concept association graph and concept labels for the SS dataset. (a) Question–concept association graph. Note that for the visualization to be compact, we show only 1/3 of all questions in the dataset; (b) Label of each concept.
estimate latent student concept knowledge states at each time instance from observed binary-valued graded student responses, we have introduced an approximate Kalman filtering framework, given all student concept knowledge state transition parameters of learning resources and the question-dependent parameters. In order to estimate these parameters, we have introduced novel block multi-convex optimization-based algorithms that estimate all the student concept knowledge state transition parameters of learning resources and question–concept associations and their intrinsic difficulties. The proposed approach applied to real-world educational datasets has shown its capability of accurately predicting unobserved student responses, while obtaining interpretable estimates of all student concept knowledge state transition parameters and question–concept associations.

A PLS can benefit from the information extracted by the SPARFA-Trace framework in a number of ways. Being able to trace students’ concept knowledge enables a PLS to make timely feedback to students on their strengths and weaknesses. Meanwhile, this information will also enable adaptivity in designing personalized learning pathways in real time, as instructors can recommend different actions for different students to take, based on their individual concept knowledge states. Furthermore, the estimated content-dependent parameters provide rich information on the knowledge structure and quality of learning resources. This capacity is crucial for a PLS to automatically suggest learning resources to students for remedial studies. Together with the question parameters estimated, a PLS would be able to operate in an autonomous manner, requiring only minimal human input and intervention; this paves the way of applying SPARFA-Trace to MOOC-scale education scenarios, where the massive amount of data precludes manual intervention.

We end with a number of avenues for future research. For example, more accurate message-passing schemes like expectation propagation ([112]) could be applied to improve the performance and accuracy of SPARFA-Trace. More sophisticated non-affine student concept knowledge state transition models can also be applied, in contrast
to the affine model proposed in Sec. 4.2.2. In order to provide better interpretation to the estimated student concept knowledge state transition and question parameters, tagging and question text information can be coupled with SPARFA-Trace (see [79, 80] for corresponding extensions to SPARFA that mine question tags and question text information). It is worth mentioning that SPARFA-Trace has potential to be applied to a wide range of other datasets, including (but not necessarily limited to) the analysis of temporal evolution in legislative voting data ([148]), and the study of temporal effects in general collaborative filtering settings ([128]). The extension of SPARFA-Trace to such applications is part of an on-going work.

4.7 Appendix

We derive the closed-form moment matching expressions for the approximate Kalman filtering approach detailed in Sec. 4.3.3. The following derivation can be seen as a multi-variate counterpart of the approach in [117, Sec. 3.9].

We start by associating the $K$-dimensional latent variable vector $c$ with a Gaussian prior $p(c) = \mathcal{N}(c | m, V)$, where $m$ and $V$ are the prior’s mean and covariance matrix, respectively. The observation likelihood takes the form $p(y | c) = \Phi((2y - 1)(w^T c - \mu))$. For simplicity of exposition, we will write $\tilde{w} = (2y - 1)w$ and $\tilde{\mu} = (2y - 1)\mu$ in the following derivations. According to Bayes rule, the posterior distribution of $c$ given the observation $y$ can be written as:

$$p(c | y) = \frac{p(c)p(y | c)}{p(y)} = \frac{p(c)p(y | c)}{\int p(y | c)p(c)dc}.$$

In order to approximate this posterior distribution of $c$, we start by evaluating its
denominator $p(y)$

$$p(y) = \int p(y|c)p(c)dc$$

$$= \int \Phi(\tilde{w}^T c - \tilde{\mu}) \mathcal{N}(c|m,V) dc$$

$$= \int \int_{-\infty}^{\infty} \mathcal{N}(t|0,1) dt \mathcal{N}(c|m,V) dc$$

$$= \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{(2\pi)^K|V|}} \int \int_{-\infty}^{\infty} \tilde{w}^T e^{-\mu} e^{-\frac{(t - \tilde{w}^T c)^2}{2}} dt e^{-\frac{c^T V^{-1} c}{2}} dc.$$

Now, substituting the variable $c$ with $c + \hat{m}$ and then, $t$ with $t - \tilde{w}^T c$, we have

$$p(y) = \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{(2\pi)^K|V|}} \int_{-\infty}^{\infty} \tilde{w}^T m - \tilde{\mu} \int_{-\infty}^{\infty} t e^{-\frac{(t - \tilde{w}^T c)^2}{2}} dt e^{-\frac{c^T V^{-1} c}{2}} dc$$

$$= \frac{1}{\sqrt{(2\pi)^{K+1}|V|}} \int_{-\infty}^{\infty} \tilde{w}^T m - \tilde{\mu} \int \mathcal{N} \left( \begin{bmatrix} t \\ c \end{bmatrix} \middle| \begin{bmatrix} 0 \\ \hat{m} \end{bmatrix} \right) \begin{bmatrix} 1 & -\tilde{w}^T \\ -\tilde{w} & \tilde{w}\tilde{w}^T + V^{-1} \end{bmatrix}^{-1} dc$$

$$= \int_{-\infty}^{\infty} \tilde{w}^T m - \tilde{\mu} \mathcal{N}(t|0,1 + \tilde{w}^T V \tilde{w}) dt = \Phi \left( \frac{\tilde{w}^T m - \tilde{\mu}}{\sqrt{1 + \tilde{w}^T V \tilde{w}}} \right). \quad (4.15)$$

In the last two steps of this derivation, we have used the Woodbury matrix identity ([67]) and marginal Gaussian properties ([117]). Since the posterior distribution is not Gaussian and prohibits the message passing procedure described in Sec. 4.3, our goal is to approximate it with a Gaussian distribution $q(c) = \mathcal{N}(c|\hat{m}, \hat{V})$ so that the message passing procedure is tractable. As shown in [117], the specific values for $\hat{m}$ and $\hat{V}$ that minimizes the Kullback-Leibler (KL) divergence between $q(c)$ and $p(c|y)$ are the first and second moments of the posterior $p(c|y)$.

Next, we evaluate the first and second moments of the posterior distribution

$$p(c|y) = p(y)^{-1} \Phi(\tilde{w}^T c - \tilde{\mu}) \mathcal{N}(c|m,V),$$

where $p(y)$ is given by (4.15). From (4.15) we can write

$$\Phi \left( \frac{\tilde{w}^T m - \tilde{\mu}}{\sqrt{1 + \tilde{w}^T V \tilde{w}}} \right) = \int \Phi(\tilde{w}^T c - \tilde{\mu}) \mathcal{N}(c|m,V) dc. \quad (4.16)$$
Taking the derivative with respect to \( m \) of both sides of (4.16) yields

\[
\mathcal{N}\left( \frac{\tilde{w}^T m - \tilde{\mu}}{\sqrt{1 + \tilde{w}^T V \tilde{w}}} \right) \frac{\tilde{w}}{\sqrt{1 + \tilde{w}^T V \tilde{w}}} = \int V^{-1} (c - m) \Phi(\tilde{w}^T c - \tilde{\mu}) \mathcal{N}(c | m, V) dc.
\]

Let \( z = \frac{\tilde{w}^T m - \tilde{\mu}}{\sqrt{1 + \tilde{w}^T V \tilde{w}}} \); then we have

\[
\mathcal{N}(z) \frac{\tilde{w}}{\sqrt{1 + \tilde{w}^T V \tilde{w}}} = V^{-1} \int c \Phi(\tilde{w}^T c - \tilde{\mu}) \mathcal{N}(c | m, V) dc - V^{-1} m \Phi(z).
\]

Thus, the mean of the posterior distribution of \( c \) is given by:

\[
\mathbb{E}_{p(c|y)}[c] = \int c p(c | y) dc
= \int c \frac{\Phi(\tilde{w}^T c - \tilde{\mu}) \mathcal{N}(c | m, V)}{p(y)} dc
= m + \frac{V \tilde{w}}{\sqrt{1 + \tilde{w}^T V \tilde{w}}} \mathcal{N}(z) \Phi(z).
\]

(4.17)

Similarly, taking the derivative with respect to \( m \) twice of both sides of (4.16) yields

\[
-z \mathcal{N}(z) \frac{\tilde{w} \tilde{w}^T}{1 + \tilde{w}^T V \tilde{w}} = -V^{-1} \int \Phi(\tilde{w}^T c - \tilde{\mu}) \mathcal{N}(c | m, V) dc
+ V^{-1} \left( \int (c - m)(c - m)^T \Phi(\tilde{w}^T c - \tilde{\mu}) \mathcal{N}(c | m, V) dc \right) V^{-1}
= -V^{-1} \Phi(z) + V^{-1} \mathbb{E}_{p(c|y)}[cc^T] V^{-1} \Phi(z)
- V^{-1} \left( \mathbb{E}_{p(c|y)}[c] m^T + m \mathbb{E}_{p(c|y)}[c]^T \right) V^{-1} \Phi(z)
+ V^{-1} mm^T V^{-1} \Phi(z),
\]

where we implicitly used the fact that the covariance matrix \( V \) is symmetric. Therefore, we have

\[
\mathbb{E}_{p(c|y)}[cc^T] = V + mm^T + \left( \mathbb{E}_{p(c|y)}[c] m^T + m \mathbb{E}_{p(c|y)}[c]^T \right) - z \frac{\mathcal{N}(z) V \tilde{w} \tilde{w}^T V}{\Phi(z) \sqrt{1 + \tilde{w}^T V \tilde{w}}}.
\]
Thus, the covariance of the posterior distribution is given by

\[
E_{p(c|y)}[(c - E_{p(c|y)}[c])(c - E_{p(c|y)}[c])^T] = E_{p(c|y)}[cc^T] - E_{p(c|y)}[c]E_{p(c|y)}[c]^T = V + mm^T + (E_{p(c|y)}[c]m^T + mE_{p(c|y)}[c]^T)
\]

\[
- z \frac{\mathcal{N}(z)}{\Phi(z)} \frac{\mathbf{V}\mathbf{w}\mathbf{w}^T\mathbf{V}}{1 + \mathbf{w}^T\mathbf{V}\mathbf{w}} - E_{p(c|y)}[c]E_{p(c|y)}[c]^T
\]

\[
= V - \frac{\mathcal{N}(z)}{\Phi(z)} \left( z + \frac{\mathcal{N}(z)}{\Phi(z)} \right) \frac{\mathbf{V}\mathbf{w}\mathbf{w}^T\mathbf{V}}{1 + \mathbf{w}^T\mathbf{V}\mathbf{w}}, \tag{4.18}
\]

where in the last step we have used (4.17) to simplify the expression.

Thus, given the prior distribution \( p(c) = \mathcal{N}(c \mid \mathbf{m}, \mathbf{V}) \) and the observation likelihood

\[
p(y \mid c) = \Phi \left( (2y - 1) \left( \mathbf{w}^T \mathbf{c} - \mu \right) \right),
\]

we can approximate the posterior distribution \( p(c \mid y) \approx q(c) = \mathcal{N}(c \mid \hat{\mathbf{m}}, \hat{\mathbf{V}}) \), with \( \hat{\mathbf{m}} \) and \( \hat{\mathbf{V}} \) as in (4.17) and (4.18), respectively.
Chapter 5

Mathematical Language Processing: Automatic Grading and Feedback for Open Response Mathematical Questions

In this chapter, we develop the mathematical language processing (MLP) framework for automatic grading and feedback generation for open response mathematical questions.

5.1 Introduction

There has been a recent wave of emerging large-scale educational platforms include massive open online courses (MOOCs) [30, 38, 43, 47, 61, 156], intelligent tutoring systems [158], computer-based homework and testing systems [7, 127, 144, 150], and personalized learning systems [105]. While computer and communication technologies have provided effective means to scale up the number of students viewing lectures (via streaming video), reading the textbook (via the web), interacting with simulations (via a graphical user interface), and engaging in discussions (via online forums), the submission and grading of assessments such as homework assignments and tests remains a weak link.

There is a pressing need to find new ways and means to automate two critical tasks that are typically handled by the instructor or course assistants in a small-scale course: (i) grading of assessments, including allotting partial credit for partially correct solutions, and (ii) providing individualized feedback to students on the locations and types of their errors.

Substantial progress has been made on automated grading and feedback systems in several restricted domains, including essay evaluation using natural language processing
(NLP) [7, 130], computer program evaluation [52, 60, 120, 129, 131], and mathematical proof verification [39, 94, 101].

We study the problem of automatically grading the kinds of open response mathematical questions that figure prominently in STEM (science, technology, engineering, and mathematics) education. To the best of our knowledge, there exist no tools to automatically evaluate and allot partial-credit scores to the solutions of such questions. As a result, large-scale education platforms have resorted either to oversimplified multiple choice input and binary grading schemes (correct/incorrect), which are known to convey less information about the students’ knowledge than open response questions [72], or peer-grading schemes [109, 115], which shift the burden of grading from the course instructor to the students.*

5.1.1 Contributions

We develop a data-driven framework for mathematical language processing (MLP) that leverages solution data from a large number of students to evaluate the correctness of solutions to open response mathematical questions, assign partial-credit scores, and provide feedback to each student on the likely locations of any errors. The scope of our framework is broad and covers questions whose solution involves one or more mathematical expressions. This includes not just formal proofs but also the kinds of mathematical calculations that figure prominently in science and engineering courses. Examples of solutions to two algebra questions of various levels of correctness are given in Figures 5.1 and 5.2. In this regard, our work differs significantly from that of [39], which focuses exclusively on evaluating logical proofs.

Our MLP framework, which is inspired by the success of NLP methods for the analysis of textual solutions (e.g., essays and short answer), comprises three main

---

*While peer grading appears to have some pedagogical value for students [125], each student typically needs to grade several solutions from other students for each question they solve, in order to obtain an accurate grade estimate.
Figure 5.1: Example solutions to the question “Find the derivative of \((x^3 + \sin x)/e^x\)” that were assigned scores of 3, 2, 1 and 0 out of 3, respectively, by our MLP-B algorithm.
Figure 5.2: Examples of two different yet correct paths to solve the question “Simplify the expression 
\((x^2 + x + \sin^2 x + \cos^2 x)(2x - 3)\).”

steps.

First, we convert each solution to an open response mathematical question into a series of numerical features. In deriving these features, we make use of symbolic mathematics to transform mathematical expressions into a canonical form.

Second, we cluster the features from several solutions to uncover the structures of correct, partially correct, and incorrect solutions. We develop two different clustering approaches. MLP-S uses the numerical features to define a similarity score between pairs of solutions and then applies a generic clustering algorithm, such as spectral clustering (SC) [102] or affinity propagation (AP) [51]. We show that MLP-S is also useful for visualizing mathematical solutions. This can help instructors identify groups of students that make similar errors so that instructors can deliver personalized remediation. MLP-B defines a nonparametric Bayesian model for the solutions and applies a Gibbs sampling algorithm to cluster the solutions.

Third, once a human assigns a grade to at least one solution in each cluster, we automatically grade the remaining (potentially large number of) solutions based on their assigned cluster. As a bonus, in MLP-B, we can track the cluster assignment
of each step in a multistep solution and determine when it departs from a cluster of correct solutions, which enables us to indicate the likely locations of errors to students.

In developing MLP, we tackle three main challenges of analyzing open response mathematical solutions. First, solutions might contain different notations that refer to the same mathematical quantity. For instance, in Fig. 5.1, the students use both $e^{-x}$ and $\frac{1}{e^x}$ to refer to the same quantity. Second, some questions admit more than one path to the correct/incorrect solution. For instance, in Fig. 5.2 we see two different yet correct solutions to the same question. It is typically infeasible for an instructor to enumerate all of these possibilities to automate the grading and feedback process. Third, numerically verifying the correctness of the solutions does not always apply to mathematical questions, especially when simplifications are required. For example, a question that asks to simplify the expression $\sin^2 x + \cos^2 x + x$ can have both $1 + x$ and $\sin^2 x + \cos^2 x + x$ as numerically correct answers, since both these expressions output the same value for all values of $x$. However, the correct answer is $1 + x$, since the question expects the students to recognize that $\sin^2 x + \cos^2 x = 1$.

Thus, methods developed to check the correctness of computer programs and formulae by specifying a range of different inputs and checking for the correct outputs, e.g., [129], cannot always be applied to accurately grade open response mathematical questions.

5.1.2 Related work

Prior work has led to a number of methods for grading and providing feedback to the solutions of certain kinds of open response questions. A linear regression-based approach has been developed to grade essays using features extracted from a training corpus using Natural Language Processing (NLP) [7, 130]. Unfortunately, such a simple regression-based model does not perform well when applied to the features extracted from mathematical solutions. Several methods have been developed for automated analysis of computer programs [60, 129]. However, these methods do not
apply to the solutions to open response mathematical questions, since they lack the structure and compilability of computer programs. Several methods have also been developed to check the correctness of the logic in mathematical proofs [39, 94, 101]. However, these methods apply only to mathematical proofs involving logical operations and not the kinds of open-ended mathematical calculations that are often involved in science and engineering courses.

The idea of clustering solutions to open response questions into groups of similar solutions has been used in a number of previous endeavors: [14, 23] uses clustering to grade short, textual answers to simple questions; [103] uses clustering to visualize a large collection of computer programs; and [119] uses clustering to grade and provide feedback on computer programs. Although the high-level concept underlying these works is resonant with the MLP framework, the feature building techniques used in MLP are very different, since the structure of mathematical solutions differs significantly from short textual answers and computer programs.

This chapter is organized as follows. In the next section, we develop our approach to convert open response mathematical solutions to numerical features that can be processed by machine learning algorithms. We then develop MLP-S and MLP-B and use real-world MOOC data to showcase their ability to accurately grade a large number of solutions based on the instructor’s grades for only a small number of solutions, thus substantially reducing the human effort required in large-scale educational platforms. We close with a discussion and perspectives on future research directions.

5.2 MLP Feature Extraction

The first step in our MLP framework is to transform a collection of solutions to an open response mathematical question into a set of numerical features. In later sections, we show how the numerical features can be used to cluster and grade solutions as well as generate informative student feedback.

A solution to an open response mathematical question will in general contain a
mixture of explanatory text and core mathematical expressions. Since the correctness of a solution depends primarily on the mathematical expressions, we will ignore the text when deriving features. However, we recognize that the text is potentially very useful for automatically generating explanations for various mathematical expressions. We leave this avenue for future work.

A workhorse of NLP is the bag-of-words model; it has found tremendous success in text semantic analysis. This model treats a text document as a collection of words and uses the frequencies of the words as numerical features to perform tasks like topic classification and document clustering [19, 23].

A solution to an open response mathematical question consists of a series of mathematical expressions that are chained together by text, punctuation, or mathematical delimiters including $=, \leq, >, \propto, \approx$, etc. For example, the solution in Figure 5.1(b) contains the expressions $((x^3 + \sin x)/e^x)'$, $((3x^2 + \cos x)e^x - (x^3 + \sin x)e^x))/e^{2x}$, and $(2x^2 - x^3 + \cos x - \sin x)/e^x$ that are all separated by the delimiter "$=$".

MLP identifies the unique mathematical expressions contained in the students’ solutions and uses them as features, effectively extending the bag-of-words model to use mathematical expressions as features rather than words. To coin a phrase, MLP uses a novel bag-of-expressions model.

Once the mathematical expressions have been extracted from a solution, we parse them using SymPy, the open source Python library for symbolic mathematics [136].\footnote{In particular, we use the \texttt{parse_expr} function.} SymPy has powerful capability for simplifying expressions. For example, $x^2 + x^2$ can be simplified to $2x^2$, and $e^x x^2/e^{2x}$ can be simplified to $e^{-x}x^2$. In this way, we can identify the equivalent terms in expressions that refer to the same mathematical quantity, resulting in more accurate features. In practice for some questions, however, it might be necessary to tone down the level of SymPy’s simplification. For instance, the key to solving the question in Figure 5.2 is to simplify the expression using the Pythagorean identity $\sin^2 x + \cos^2 x = 1$. If SymPy is called on to perform such a
simplification automatically, then it will not be possible to verify whether a student has correctly navigated the simplification in their solution. For such problems, it is advisable to perform only arithmetic simplifications.

After extracting the expressions from the solutions, we transform the expressions into numerical features. We assume that $N$ students submit solutions to a particular mathematical question. Extracting the expressions from each solution using SymPy yields a total of $V$ unique expressions across the $N$ solutions.

We encode the solutions in a integer-valued solution feature matrix $Y \in \mathbb{N}^{V \times N}$ whose rows correspond to different expressions and whose columns correspond to different solutions; that is, the $(i,j)^{th}$ entry of $Y$ is given by

$$Y_{i,j} = \text{times expression } i \text{ appears in solution } j.$$

Each column of $Y$ corresponds to a numerical representation of a mathematical solution. Note that we do not consider the ordering of the expressions in this model; such an extension is an interesting avenue for future work. In this paper, we indicate in $Y$ only the presence and not the frequency of an expression, i.e., $Y \in \{0, 1\}^{V \times N}$ and

$$Y_{i,j} = \begin{cases} 1 & \text{if expression } i \text{ appears in solution } j \\ 0 & \text{otherwise.} \end{cases} \quad (5.1)$$

The extension to encoding frequencies is straightforward.

To illustrate how the matrix $Y$ is constructed, consider the solutions in Fig. 5.2(a) and (b). Across both solutions, there are 7 unique expressions. Thus, $Y$ is a $7 \times 2$ matrix, with each row corresponding to a unique expression. Letting the first four rows of $Y$ correspond to the four expressions in Fig. 5.2(a) and the remaining three rows to expressions 2–4 in Fig. 5.2(b), we have

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}^T.$$
We end this section with the crucial observation that, for a wide range of mathematical questions, many expressions will be shared across students’ solutions. This is true, for instance, in Figure 5.2. This suggests that there are a limited number of types of solutions to a question (both correct and incorrect) and that solutions of the same type tend to be similar to each other. This leads us to the conclusion that the $N$ solutions to a particular question can be effectively clustered into $K \ll N$ clusters. In the next two sections, we will develop MLP-S and MLP-B, two algorithms to cluster solutions according to their numerical features.

5.3 MLP-S: Similarity-Based Clustering

In this section, we outline MLP-S, which clusters and then grades solutions using a solution similarity-based approach.

5.3.1 The MLP-S model

We start by using the solution features in $Y$ to define a notion of similarity between pairs of solutions. Define the $N \times N$ similarity matrix $S$ containing the pairwise similarities between all solutions, with its $(i, j)^{th}$ entry the similarity between solutions $i$ and $j$

$$S_{i,j} = \frac{y_i^T y_j}{\min\{y_i^T y_i, y_j^T y_j\}}.$$  \hspace{1cm} (5.2)

The column vector $y_i$ denotes the $i^{th}$ column of $Y$ and corresponds to student $i$’s solution. Informally, $S_{i,j}$ is the number of common expressions between solution $i$ and solution $j$ divided by the minimum of the number of expressions in solutions $i$ and $j$. A large/small value of $S_{i,j}$ corresponds to the two solutions being similar/dissimilar. For example, the similarity between the solutions in Figure 5.1(a) and Figure 5.1(b) is $1/3$ and the similarity between the solutions in Figure 5.2(a) and Figure 5.2(b) is $1/2$. $S$ is symmetric, and $0 \leq S_{i,j} \leq 1$. Equation (5.2) is just one of any possible solution similarity metrics. We defer the development of other metrics to future work.
Figure 5.3: Illustration of the clusters obtained by MLP-S by applying affinity propagation (AP) on the similarity matrix $S$ corresponding to students’ solutions to four different mathematical questions (see Table 1 for more details about the datasets and the Appendix for the question statements). Each node corresponds to a solution. Nodes with the same color correspond to solutions that are estimated to be in the same cluster. The thickness of the edge between two solutions is proportional to their similarity score. Boxed solutions are correct; all others are in varying degrees of correctness.

5.3.2 Clustering solutions in MLP-S

Having defined the similarity $S_{i,j}$ between two solutions $i$ and $j$, we now cluster the $N$ solutions into $K \ll N$ clusters such that the solutions within each cluster have high similarity score between them and solutions in different clusters have low similarity score between them.
Given the similarity matrix $S$, we can use any of the multitude of standard clustering algorithms to cluster solutions. Two examples of clustering algorithms are *spectral clustering* (SC) [102] and *affinity propagation* (AP) [51]. The SC algorithm requires specifying the number of clusters $K$ as an input parameter, while the AP algorithm does not.

Figure 5.3 illustrates how AP is able to identify clusters of similar solutions from solutions to four different mathematical questions. The figures on the top correspond to solutions to the questions in Figures 5.1 and 5.2, respectively. The bottom two figures correspond to solutions to two signal processing questions. Each node in the figure corresponds to a solution, and nodes with the same color correspond to solutions that belong to the same cluster. For each figure, we show a sample solution from some of these clusters, with the boxed solutions corresponding to correct solutions. We can make three interesting observations from Fig. 5.3:

- In the top left figure, we cluster a solution with the final answer $3x^2 + \cos x - (x^3 + \sin x))/e^x$ with a solution with the final answer $3x^2 + \cos x - (x^3 + \sin x))/e^x$. Although the later solution is incorrect, it contained a typographical error where $3 \ast x \wedge 2$ was typed as $3 \wedge x \wedge 2$. MLP-S is able to identify this typographical error, since the expression before the final solution is contained in several other correct solutions.

- In the top right figure, the correct solution requires identifying the trigonometric identity $\sin^2 x + \cos^2 x = 1$. The clustering algorithm is able to identify a subset of the students who were not able to identify this relationship and hence could not simplify their final expression.

- MLP-S is able to identify solutions that are strongly connected to each other. Such a visualization can be extremely useful for course instructors. For example, an instructor can easily identify a group of students who lack mastery of a certain skill that results in a common error and adjust their course plan accordingly to
help these students.

5.3.3 Auto-grading via MLP-S

Having clustered all solutions into a small number $K$ of clusters, we assign the same grade to all solutions in the same cluster. If a course instructor assigns a grade to one solution from each cluster, then MLP-S can automatically grade the remaining $N - K$ solutions. We construct the index set $I_S$ of solutions that the course instructor needs to grade as

$$I_S = \left\{ \arg \max_{i \in C_k} \sum_{j=1}^{N} S_{i,j}, \ k = 1, 2, \ldots, K \right\},$$

where $C_k$ represents the index set of the solutions in cluster $k$. In words, in each cluster, we select the solution having the highest similarity to the other solutions (ties are broken randomly) to include in $I_S$. We demonstrate the performance of auto-grading via MLP-S in the experimental results section below.

5.4 MLP-B: Bayesian Nonparametric Clustering

In this section, we outline MLP-B, which clusters and then grades solutions using a Bayesian nonparameterics-based approach. The MLP-B model and algorithm can be interpreted as an extension of the model in [159], where a similar approach is proposed to cluster short text documents.

5.4.1 The MLP-B model

Following the key observation that the $N$ solutions can be effectively clustered into $K \ll N$ clusters, let $z$ be the $N \times 1$ cluster assignment vector, with $z_j \in \{1, \ldots, K\}$ denoting the cluster assignment of the $j^{th}$ solution with $j \in \{1, \ldots, N\}$. Using this latent variable, we model the probability of the solution of all students’ solutions to
the question as

\[ p(Y) = \prod_{j=1}^{N} \left( \sum_{k=1}^{K} p(y_j | z_j = k) p(z_j = k) \right), \]

where \( y_j \), the \( j \)th column of the data matrix \( Y \), corresponds to student \( j \)'s solution to the question. Here we have implicitly assumed that the students' solutions are independent of each other. By analogy to topic models [19, 134], we assume that student \( j \)'s solution to the question, \( y_j \), is generated according to a multinomial distribution given the cluster assignments \( z \) as

\[ p(y_j | z_j = k) = \text{Mult}(y_j | \Phi_k) = \frac{(\sum_i Y_{i,j})!}{Y_{1,j}!Y_{2,j}! \ldots Y_{V,j}!} \Phi_{1,k}^{Y_{1,j}} \Phi_{2,k}^{Y_{2,j}} \ldots \Phi_{V,k}^{Y_{V,j}}, \]

(5.3)

where \( \Phi \in [0,1]^{V \times K} \) is a parameter matrix with \( \Phi_{v,k} \) denoting its \((v,k)\)th entry. \( \Phi_k \in [0,1]^{V \times 1} \) denotes the \( k \)th column of \( \Phi \) and characterizes the multinomial distribution over all the \( V \) features for cluster \( k \).

In practice, one often has no information regarding the number of clusters \( K \). Therefore, we consider \( K \) as an unknown parameter and infer it from the solution data. In order to do so, we impose a Chinese restaurant process (CRP) prior on the cluster assignments \( z \), parameterized by a parameter \( \alpha \). The CRP characterizes the random partition of data into clusters, in analogy to the seating process of customers in a Chinese restaurant. It is widely used in Bayesian mixture modeling literature [18, 59]. Under the CRP prior, the cluster (table) assignment of the \( j \)th solution (customer), conditioned on the cluster assignments of all the other solutions, follows the distribution

\[ p(z_j = k | z_{\neg j}, \alpha) = \begin{cases} \frac{n_{k,\neg j}}{N-1+\alpha} & \text{if cluster } k \text{ is occupied,} \\ \frac{\alpha}{N-1+\alpha} & \text{if cluster } k \text{ is empty,} \end{cases} \]

(5.4)

where \( n_{k,\neg j} \) represents the number of solutions that belong to cluster \( k \) excluding the current solution \( j \), with \( \sum_{k=1}^{K} n_{k,\neg j} = N - 1 \). The vector \( z_{\neg j} \) represents the cluster assignments of the other solutions. The flexibility of allowing any solution
Figure 5.4: Graphical model of the generation process of solutions to mathematical questions. \( \alpha_\alpha, \alpha_\beta \) and \( \beta \) are hyperparameters, \( z \) and \( \Phi \) are latent variables to be inferred, and \( Y \) is the observed data defined in (5.1).

to start a new cluster of its own enables us to automatically infer \( K \) from data. It is known [137] that the expected number of clusters under the CRP prior satisfies \( K \sim O(\alpha \log N) \ll N \), so our method scales well as the number of students \( N \) grows large. We also impose a Gamma prior \( \alpha \sim Gam(\alpha_\alpha, \alpha_\beta) \) on \( \alpha \) to help us infer its value.

Since the solution feature data \( Y \) is assumed to follow a multinomial distribution parameterized by \( \Phi \), we impose a symmetric Dirichlet prior over \( \Phi \) as \( \phi_k \sim Dir(\phi_k|\beta) \) because of its conjugacy with the multinomial distribution [54].

The graphical model representation of our model is visualized in Fig. 5.4. Our goal next is to estimate the cluster assignments \( z \) for the solution of each student, the parameters \( \phi_k \) of each cluster, and the number of clusters \( K \), from the binary-valued solution feature data matrix \( Y \).

5.4.2 Clustering solutions in MLP-B

We use a Gibbs sampling algorithm for posterior inference under the MLP-B model, which automatically groups solutions into clusters. We start by applying a generic clustering algorithm (e.g., \( K \)-means, with \( K = N/10 \)) to initialize \( z \), and then initialize \( \Phi \) accordingly. Then, in each iteration of MLP-B, we perform the following steps:

1. **Sample \( z \):** For each solution \( j \), we remove it from its current cluster and sample its cluster assignment \( z_j \) from the posterior \( p(z_j = k|z_{-j}, \alpha, Y) \). Using Bayes
rule, we have

\[ p(z_j = k | \mathbf{z}_{-j}, \Phi, \alpha, \mathbf{Y}) = p(z_j = k | \mathbf{z}_{-j}, \phi_k, \alpha, y_j) \]

\[ \propto p(z_j = k | \mathbf{z}_{-j}, \alpha) p(y_j | z_j = k, \phi_k). \]

The prior probability \( p(z_j = k | \mathbf{z}_{-j}, \alpha) \) is given by (5.4). For non-empty clusters, the observed data likelihood \( p(y_j | z_j = k, \phi_k) \) is given by (5.3). However, this does not apply to new clusters that are previously empty. For a new cluster, we marginalize out \( \phi_k \), resulting in

\[ p(y_j | z_j = k, \beta) = \int_{\phi_k} p(y_j | z_j = k, \phi_k)p(\phi_k | \beta) \]

\[ = \int_{\phi_k} \text{Mult}(y_j | z_j = k, \phi_k) \text{Dir}(\phi_k | \beta) \]

\[ = \frac{\Gamma(V\beta)}{\Gamma(\sum_{i=1}^{V} Y_{i,j} + V\beta)} \prod_{i=1}^{V} \frac{\Gamma(Y_{i,j} + \beta)}{\Gamma(\beta)}, \]

where \( \Gamma(\cdot) \) is the Gamma function.

If a cluster becomes empty after we remove a solution from its current cluster, then we remove it from our sampling process and erase its corresponding multinomial parameter vector \( \phi_k \). If a new cluster is sampled for \( z_j \), then we sample its multinomial parameter vector \( \phi_k \) immediately according to Step 2 below. Otherwise, we do not change \( \phi_k \) until we have finished sampling \( z \) for all solutions.

2. **Sample \( \Phi \):** For each cluster \( k \), sample \( \phi_k \) from its posterior \( \text{Dir}(\phi_k | n_{1,k} + \beta, \ldots, n_{V,k} + \beta) \), where \( n_{i,k} \) is the number of times feature \( i \) occurs in the solutions that belong to cluster \( k \).

3. **Sample \( \alpha \):** Sample \( \alpha \) using the approach described in [154].

4. **Update \( \beta \):** Update \( \beta \) using the fixed-point procedure described in [96].

The output of the Gibbs sampler is a series of samples that correspond to the approximate posterior distribution of the various parameters of interest. To make
meaningful inference for these parameters (such as the posterior mean of a parameter), it is important to appropriately post-process these samples. For our estimate of the true number of clusters, \( \hat{K} \), we simply take the mode of the posterior distribution on the number of clusters \( K \). We use only iterations with \( K = \hat{K} \) to estimate the posterior statistics [149].

In mixture models, the issue of "label-switching" can cause a model to be unidentifiable, because the cluster labels can be arbitrarily permuted without affecting the data likelihood. In order to overcome this issue, we use an approach reported in [149]. First, we compute the likelihood of the observed data in each iteration as \( p(Y|\Phi^{\ell}, z^{\ell}) \), where \( \Phi^{\ell} \) and \( z^{\ell} \) represent the samples of these variables at the \( \ell \)th iteration. After the algorithm terminates, we search for the iteration \( \ell_{\text{max}} \) with the largest data likelihood and then permute the labels \( z^{\ell} \) in the other iterations to best match \( \Phi^{\ell} \) with \( \Phi^{\ell_{\text{max}}} \). We use \( \hat{\Phi} \) (with columns \( \hat{\phi}_k \)) to denote the estimate of \( \Phi \), which is simply the posterior mean of \( \Phi \). Each solution \( j \) is assigned to the cluster indexed by the mode of the samples from the posterior of \( z_j \), denoted by \( \hat{z}_j \).

5.4.3 Auto-grading via MLP-B

We now detail how to use MLP-B to automatically grade a large number \( N \) of students’ solutions to a mathematical question, using a small number \( \hat{K} \) of instructor graded solutions. First, as in MLP-S, we select the set \( I_B \) of “typical solutions” for the instructor to grade. We construct \( I_B \) by selecting one solution from each of the \( \hat{K} \) clusters that is most representative of the solutions in that cluster:

\[
I_B = \{ \arg\max_j p(y_j|\hat{\phi}_k), k = 1, 2, \ldots, \hat{K} \}.
\]

In words, for each cluster, we select the solution with the largest likelihood of being in that cluster.

The instructor grades the \( \hat{K} \) solutions in \( I_B \) to form the set of instructor grades \( \{g_k\} \) for \( k \in I_B \). Using these grades, we assign grades to the other solutions \( j \notin I_B \).
according to
\[
\hat{g}_j = \frac{\sum_{k=1}^{K} p(y_j|\hat{\phi}_k) g_k}{\sum_{k=1}^{K} p(y_j|\hat{\phi}_k)}.
\] (5.5)

That is, we grade each solution not in \( I_B \) as the average of the instructor grades weighted by the likelihood that the solution belongs to cluster. We demonstrate the performance of auto-grading via MLP-B in the experimental results section below.

5.5 Experiments

In this section, we demonstrate how MLP-S and MLP-B can be used to accurately estimate the grades of roughly 100 open response solutions to mathematical questions by only asking the course instructor to grade approximately 10 solutions. We also demonstrate how MLP-B can be used to automatically provide feedback to students on the locations of errors in their solutions.

5.5.1 Auto-grading via MLP-S and MLP-B

Datasets Our dataset that consists of 116 students solving 4 open response mathematical questions in an edX course. The set of questions includes 2 high-school level mathematical questions and 2 college-level signal processing questions (details about the questions can be found in Tbl. 5.1, and the question statements are given in the Appendix). For each question, we pre-process the solutions to filter out the blank solutions and extract features. Using the features, we represent the solutions by the matrix \( Y \) in (5.1). Every solution was graded by the course instructor with one of the scores in the set \( \{0, 1, 2, 3\} \), with a full credit of 3.

Baseline: Random sub-sampling We compare the auto-grading performance of MLP-S and MLP-B against a baseline method that does not group the solutions into clusters. In this method, we randomly sub-sample all solutions to form a small set of solutions for the instructor to grade. Then, each ungraded solution is simply assigned
<table>
<thead>
<tr>
<th>Question</th>
<th>No.of solutions $N$</th>
<th>No.of features (unique expressions) $V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>108</td>
<td>78</td>
</tr>
<tr>
<td>Question 2</td>
<td>113</td>
<td>53</td>
</tr>
<tr>
<td>Question 3</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>Question 4</td>
<td>110</td>
<td>45</td>
</tr>
</tbody>
</table>

Table 5.1: Datasets consisting of the solutions of 116 students to 4 mathematical questions on algebra and signal processing. See the Appendix for the question statements.

Experimental setup For each question, we apply four different methods for auto-grading:

- Random sub-sampling (RS) with the number of clusters $K \in \{5, 6, \ldots, 40\}$.
- MLP-S with spectral clustering (SC) with $K \in \{5, 6, \ldots, 40\}$.
- MLP-S with affinity propagation (AP) clustering. This algorithm does not require $K$ as an input.
- MLP-B with hyperparameters set to the non-informative values $\alpha_a = \alpha_\beta = 1$ and running the Gibbs sampling algorithm for 10,000 iterations with 2,000 burn-in iterations.

\[\text{‡}\] Other baseline methods, such as the linear regression-based method used in the edX essay grading system [130], are not listed, because they did not perform as well as random sub-sampling in our experiments.
MLP-S with AP and MLP-B both automatically estimate the number of clusters $K$. Once the clusters are selected, we assign one solution from each cluster to be graded by the instructor using the methods described in earlier sections.

**Performance metric** We use mean absolute error (MAE), which measures the “average absolute error per auto-graded solution”

$$\text{MAE} = \frac{\sum_{j=1}^{N-K} |\hat{g}_j - g_j|}{N - K},$$

as our performance metric. Here, $N - K$ equals the number of solutions that are auto-graded, and $\hat{g}_j$ and $g_j$ represent the estimated grade (for MLP-B, the estimated grades are rounded to integers) and the actual instructor grades for the auto-graded solutions, respectively.

**Results and discussion** In Fig. 5.5, we plot the MAE versus the number of clusters $K$ for Questions 1–4. MLP-S with SC consistently outperforms the random sampling baseline algorithm for almost all values of $K$. This performance gain is likely due to the fact that the baseline method does not cluster the solutions and thus does not select a good subset of solutions for the instructor to grade. MLP-B is more accurate than MLP-S with both SC and AP and can automatically estimate the value of $K$, although at the price of significantly higher computational complexity (e.g., clustering and auto-grading one question takes 2 minutes for MLP-B compared to only 5 seconds for MLP-S with AP on a standard laptop computer with a 2.8GHz CPU and 8GB memory).

Both MLP-S and MLP-B grade the students’ solutions accurately (e.g., an MAE of 0.04 out of the full grade 3 using only $K = 13$ instructor grades to auto-grade all $N = 113$ solutions to Question 2). Moreover, as we see in Fig. 5.5, the MAE for MLP-S decreases as $K$ increases, and eventually reaches 0 when $K$ is large enough that only solutions that are exactly the same as each other belong to the same cluster. In practice, one can tune the value of $K$ to achieve a balance between maximizing grading
accuracy and minimizing human effort. Such a tuning process is not necessary for MLP-B, since it automatically estimates the value of $K$ and achieves such a balance.

### 5.5.2 Feedback generation via MLP-B

**Experimental setup** Since Questions 3–4 require some familiarity with signal processing, we demonstrate the efficacy of MLP-B in providing feedback on mathematical solutions on Questions 1–2. Among the solutions to each question, there are a few types of common errors that more than one student makes. We take one incorrect solution out of each type and run MLP-B on the other solutions to estimate the parameter $\hat{\phi}_k$ for each cluster. Using this information and the instructor grades $\{g_k\}$, after each expression $v$ in a solution, we compute the probability that it belongs to a cluster $p(y_j^{(v)}|\hat{\phi}_k)$ that does not have full credit ($g_k < 3$), together with the expected credit using (5.5). Once the expected grade is calculated to be less than full credit, we consider that an error has occurred.

**Results and discussion** Two sample feedback generation process are shown in Fig. 5.6. In Fig. 5.6(a), we can provide feedback to the student on their error as early as Line 2, before it carries over to later lines. Thus, MLP-B can potentially become a powerful tool to generate timely feedback to students as they are solving mathematical questions, by analyzing the solutions it gathers from other students.

### 5.6 Conclusions

We have developed a framework for mathematical language processing (MLP) that consists of three main steps: (i) converting each solution to an open response mathematical question into a series of numerical features; (ii) clustering the features from several solutions to uncover the structures of correct, partially correct, and incorrect solutions; and (iii) automatically grading the remaining (potentially large number of) solutions based on their assigned cluster and one instructor-provided grade per
cluster. As our experiments have indicated, our framework can substantially reduce
the human effort required for grading in large-scale courses. As a bonus, MLP-S enables instructors to visualize the clusters of solutions to help them identify common
errors and thus groups of students having the same misconceptions. As a further
bonus, MLP-B can track the cluster assignment of each step of a multistep solution
and determine when it departs from a cluster of correct solutions, which enables us to
indicate the locations of errors to students in real time. Improved learning outcomes
should result from these innovations.

There are several avenues for continued research. We are currently planning more
extensive experiments on the edX platform involving tens of thousands of students.
We are also planning to extend the feature extraction step to take into account both
the ordering of expressions and ancillary text in a solution. Clustering algorithms that
allow a solution to belong to more than one cluster could make MLP more robust
to outlier solutions and further reduce the number of solutions that the instructors
need to grade. Finally, it would be interesting to explore how the features of solutions
could be used to build predictive models, as in the Rasch model [116] or item response
theory [91].

5.7 Appendix: Question Statements

Question 1: Multiply

\[(x^2 + x + \sin^2 x + \cos^2 x)(2x - 3)\]

and simplify your answer as much as possible.

Question 2: Find the derivative of \(\frac{x^3 + \sin(x)}{e^x}\) and simplify your answer as much as
possible.

Question 3: A discrete-time linear time-invariant system has the impulse response
shown in the figure (omitted). Calculate \(H(e^{j\omega})\), the discrete-time Fourier transform
of \(h[n]\). Simplify your answer as much as possible until it has no summations.
Question 4: Evaluate the following summation

$$\sum_{k=-\infty}^{\infty} \delta[n - k] x[k - n].$$
Figure 5.5: Mean absolute error (MAE) versus the number of instructor graded solutions (clusters) $K$, for Questions 1–4, respectively. For example, on Question 1, MLP-S and MLP-B estimate the true grade of each solution with an average error of around 0.1 out of a full credit of 3. “RS” represents the random sub-sampling baseline. Both MLP-S methods and MLP-B outperforms the baseline method.
(a) A sample feedback generation process where the student makes an error in the expression in Line 2 while attempting to solve Question 1.

\[
\frac{(x^3 + \sin x)}{e^x} \]
\[
= \frac{\left( e^x \cdot \frac{d}{dx}(x^3 + \sin x) \right)}{e^x} = \frac{e^x \cdot (3x^2 + \cos x) - (x^3 + \sin x) \cdot e^x}{e^{2x}} \\
\text{prob.incorrect} = 0.11, \quad \text{exp.grade} = 3 \\
= (2x^2 + \cos x - x^3 - \sin x)/e^x \\
\text{prob.incorrect} = 0.66, \quad \text{exp.grade} = 2 \\
= (x^2(2 - x) + \cos x - \sin x)/e^x \\
\text{prob.incorrect} = 0.99, \quad \text{exp.grade} = 2
\]

(b) A sample feedback generation process where the student makes an error in the expression in Line 3 while attempting to solve Question 2.

\[
(x^2 + x + \sin^2 x + \cos^2 x)(2x - 3) \\
= (x^2 + x + 1)(2x - 3) \\
\text{prob.incorrect} = 0.09, \quad \text{exp.grade} = 3 \\
= 4x^3 + 2x^2 + 2x - 3x^2 - 3x - 3 \\
\text{prob.incorrect} = 0.82, \quad \text{exp.grade} = 2 \\
= 4x^3 - x^2 - x - 3 \\
\text{prob.incorrect} = 0.99, \quad \text{exp.grade} = 2
\]

Figure 5.6: Demonstration of real-time feedback generation by MLP-B while students enter their solutions. After each expression, we compute both the probability that the student’s solution belongs to a cluster that does not have full credit and the student's expected grade. An alert is generated when the expected credit is less than full credit.
Chapter 6

A Contextual Multi-Armed Bandits Framework for Personalized Learning Action Selection

In this chapter, we propose a contextual multi-armed bandits framework to automatically select personalized learning actions for each student given their learning history to maximize learning.

6.1 Introduction

Machine learning-based personalized learning systems [105] have shown great promise in reaching beyond ITS to scale to large numbers of subjects and students. These systems automatically create personalized learning schedules, i.e., a series of personalized learning actions (PLAs), for each individual student to take that maximize their learning. Examples of PLAs include reading a textbook section, watching a lecture video, interacting with a simulation or lab, solving a practice question, etc. Instead of domain-specific rules, machine learning algorithms are used to select PLAs automatically by analyzing the data students generate as they interact with learning resources.

A related area of interest is computerized adaptive testing (CAT) [141, 142, 145], which aims to optimize the assessment (rather than the learning) process. CAT systems iteratively update their estimate of a student’s ability based on newly observed student responses and then adaptively select the next question that is most informative for estimating their ability. As a result, they tend to select questions that the student has, for example, a 50% chance of answering correctly. CAT systems are powerful in that they can accurately assess each student’s ability using a small number of questions.
However, assessment is not learning and, moreover, a number of works have suggested that selecting questions that are more challenging to a student within their capability is more beneficial to learning [87, 100].

The problem of creating a fully personalized learning schedule for each student can be formulated in general using the partially observed Markov decision process (POMDP) framework [110]. POMDPs utilize models on the students’ latent knowledge states [81, 91] and their transitions [28, 37, 73, 78] to learn a PLA selection policy (a mapping from the knowledge state space to the set of learning actions) that maximizes a reward received in the possibly distant future (long-term learning outcome). Previous work applying POMDPs to personalized learning have achieved some degree of success [12, 33, 113, 114]. However, learning a personalized learning schedule using a POMDP is greatly complicated by the curse of dimensionality; the situation quickly becomes intractable as the dimensions of the state and action spaces grow [110]. Therefore, POMDPs have made only a limited impact in large-scale personalized learning applications involving large numbers of students and learning actions.

A more scalable approach to personalized learning is to learn a PLA selection policy using the multi-armed bandits (MAB) framework [53, 89], which is more suitable to optimizing students’ success on immediate follow-up assessments (short-term learning outcome). The simplicity of the MAB framework makes it more practical than the POMDP framework in real-world educational applications, since it requires less training data.

6.1.1 Contributions

In this paper, we study the problem of selecting PLAs for each student given their learning history using MABs. We first estimate each student’s latent concept knowledge profile from their learning history (specifically, their binary-valued graded responses to questions in previous assessments) using the sparse factor analysis (SPARFA) framework [81]. Then, we use these concept knowledge profiles as contexts in the
contextual (multi-armed) bandits framework to learn a policy to select PLAs for each student that maximize their performance on the follow-up assessment.

We develop three algorithms for PLA selection. The first algorithm, CLUB, has theoretical guarantees on its ability to identify the optimal PLA for each student. The second and third algorithms, A-CLUB and CPT, are more intuitive and practical; we experimentally validate their performance using two real-world educational datasets. Our experimental results demonstrate that A-CLUB and CPT achieve superior or comparable performance against existing algorithms in terms of maximizing students’ immediate success.

Our choice of SPARFA to estimate the students’ knowledge profiles is not exclusive. Indeed, our approach can leverage a range of other approaches, including item response theory (IRT)-based models [91, 118] and factor analysis-based models [16, 28, 56].

6.1.2 Related work

The work in [89] applies a MAB algorithm to educational games in order to trade off scientific discovery (learning about the effect of each learning resource) and student learning. Their approach is context-free and thus not ideally suited for applications with significant variation among the knowledge states of individual students. Indeed, it can be seen as a special case of our work in this paper when there is no context information available.

The work in [138] applies a contextual bandits algorithm to the problem of selecting the optimal PLA for each student given the student’s previous exposure to learning resources. In their approach, each dimension of the context vector corresponds to the students’ exposure to one learning resource. Thus, the context space quickly explodes as the number of learning resources increases. Our approach, in contrast, performs dimensionality reduction on student learning histories using the SPARFA framework, and uses the resulting student concept knowledge profiles as contexts. This feature enables our work to be applied to datasets where student learning histories contain a
The work in [92] collects high-dimensional student–computer interaction features as they play an educational game and uses them to search for a good teaching policy. We emphasize that our approach can be applied to almost all educational applications, not just computerized educational games, since it only requires graded response data, which is very common in all levels of education.

The works in [53] and [76] both use some form of expert knowledge to learn a teaching policy. The approach of [53], in particular, uses expert knowledge to narrow down the set of possible PLAs a student can take. Our approach, in contrast, requires no expert knowledge, and is therefore fully data-driven and domain-agnostic.

The work in [86] fuses MAB algorithms with Gaussian process regression in order to reduce the amount of training data required to search for a good teaching policy. Their work requires the policy to be parameterized by a few parameters, while our framework does not and can thus learn more complicated policies using only reward observations.

The work in [121] found that various student response models, including knowledge tracing (KT) [37, 107], IRT models [91], additive factor models (AFM) [28], and performance factor models (PFM) [56], can have similar predictive performance yet lead to very different teaching policies. Although these results are indeed interesting, we emphasize that the focus of the current work is to develop policy learning algorithms rather than comparing student models.

6.2 Problem Formulation

We study the problem of creating a personalized learning schedule of each student by selecting the PLA they should take based on their prior learning experience. We assume that a student’s learning schedule consists of a series of assessments with PLAs embedded in between, a setting that is typical in traditional classrooms, blended learning environments, and online courses like MOOCs [38, 47]. Each PLA can
correspond to studying a learning resource, e.g., reading a textbook section, watching a lecture video, conducting an interactive simulation, solving a practice question, etc., or a combination of several learning resources. Each assessment could be a pop-quiz with a single question, a homework set with multiple questions, or a longer exam. Each student’s personalized learning schedule can be visualized as in Figure 1, where a PLA is taken between consecutive assessments (starting after Assessment 1).

The goal of this work is to select the optimal PLA for each student given their learning history (their graded responses to previous assessments) that maximizes their immediate success, i.e., the credit they receive on the following assessment. For simplicity of exposition, we will place PLA 1 between Assessment 1 and Assessment 2 (as encased in the box in Figure 1) as a running example throughout the paper.

Let $A$ denote the number of total PLAs available, let $K$ denote the number of latent concepts covered up to Assessment 1, and let $Q$ denote the number of questions in Assessment 2, with $s_i, i = 1, \ldots, Q$ the maximum credit of each question. Let $Y_{i,j}$ denote the binary-valued graded response of student $j$ to question $i$, with $Y_{i,j} = 1$ denoting a correct response and $Y_{i,j} = 0$ an incorrect response. In order to pin down a feasible PLA selection algorithm, we make the following simplifying assumptions:

1. We assume that the process of learning is Markov, i.e., a student’s future performance depends only on their current concept knowledge state and not on their complete previous learning history.

*Our notion of PLA is very general, and we do not restrict ourselves to studying a single learning resource [53].
2. We assume that a reliable estimate of each student’s latent concept knowledge vector (estimated from their graded responses to Assessment 1), denoted by \( c_j \in \mathbb{R}^K \), is available to the PLA selection algorithm. Such an estimate can be obtained using any IRT-like method, e.g., SPARFA [81].

3. We assume that the PLA selected for each student will directly affect their performance on Assessment 2.

In summary, the goal of our algorithm is to select a PLA for student \( j \), given their current concept knowledge \( c_j \dagger \), in order to maximize their performance (i.e., their expected credit \( \sum_{i=1}^{Q} s_i \mathbb{E}[Y_{i,j}] \)) on Assessment 2.

### 6.2.1 Background on bandits

The MAB framework [9] studies the problem of a player trying to learn a policy that maximizes the total expected reward by playing (pulling the arms of) a collection of slot machines with a fixed number of trials and no prior information about each machine. Each machine has a fixed reward distribution that is unknown to the player. The key to maximizing the total expected reward is to find the right balance between exploration (playing machines that might yield high rewards) and exploitation (repeatedly playing the machine with the highest observed reward). Analogously, a personalized learning system must strike a balance between testing the efficacy of every learning action (exploration) and maximizing the students’ learning outcomes using observations on the actions (exploitation) [89].

An extension of the MAB framework, contextual (multi-armed) bandits [2, 8, 50, 83, 139] account for the existence of additional information on the player and/or the machines, referred to as “contexts”, in order to improve the policy. Our PLA selection problem fits squarely the contextual bandits framework, where the current estimates of students’ concept knowledge correspond to the contexts and each PLA corresponds

\[1] In practice, we augment \( c_j \) as \( [c_j^T 1]^T \) to add an “offset” parameter to each arm.
to an arm. Pulling an arm corresponds simply to selecting a PLA. In this paper, the context will include only information on the students. See Sec. 6.5 for a discussion on extending our framework to incorporate information on the learning resources into the contexts.

6.3 Algorithms

The three algorithms we develop in this section can be divided into two groups. The first two algorithms are upper confidence bound (UCB)-based algorithms [9]. These algorithms maintain estimates of the expected reward of each arm, together with confidence intervals around these estimates, and iteratively update them as each new pull and its corresponding reward is observed. They then pull the arm with the highest UCB on the reward, which is equal to the expected reward plus the width of the confidence interval. The third algorithm builds on Thompson sampling [140], which uses Bayes’ rule to iteratively update the posterior distributions of the parameters of each arm, and then pulls the arm with the highest estimated reward using randomly drawn samples from these distributions.

6.3.1 CLUB: An algorithm in theory

Our first algorithm, contextual logistic upper confidence bound (CLUB), exhibits theoretical guarantees. We assume that the binary-valued student responses to the questions in Assessment 2 are Bernoulli random variables with success probabilities following a logistic model, i.e.,

\[ p(Y_{i,j_{as}} = 1) = \Phi_{\text{log}}(c_{j_{as}}^T \mathbf{w}_i^a) = \frac{1}{1 + e^{-c_{j_{as}}^T \mathbf{w}_i^a}}, \quad s = 1, \ldots, n_a, \]

where \( \mathbf{w}_i^a \in \mathbb{R}^K \) is the parameter vector that characterizes the students’ responses to question \( i \) after taking PLA \( a \). Also, \( j_{as} \) denotes the index of the \( s^{\text{th}} \) student to take PLA \( a \), and \( n_a \) denotes the total number of students to take PLA \( a \). \( \Phi_{\text{log}}(\cdot) \) denotes the inverse logit link function.
The maximum-likelihood estimate (MLE) of $w^a_i$ is

$$
\hat{w}^a_i = \arg \min_w \sum_{s=1}^{n_a} \log p(Y_{i,j,s} | c^a_{j,s}, w),
$$

which can be computed using standard logistic regression algorithms [63] whenever the MLE exists (see [145, Sec. 5.1] for a detailed discussion on the conditions under which the MLE exists).

As detailed in Algorithm 1, CLUB maintains MLEs of the parameter vector $w^a_i$ of each PLA together with a confidence interval around it. Then, after receiving a student’s concept knowledge vector $c_j$, CLUB selects the PLA with the highest UCB on the expected credit on the student’s following assessment.

The constants in Algorithm 1 are given by $c_i(n_a) = \sqrt{2K(3 + 2\log(1 + 2a_m^2/\lambda_0)) \log n_a K/\delta/b_{i,a}}$, where $a_m = \sqrt{K + 2\sqrt{K \log(1/\eta)} + 2\log(1/\eta)}$ and $b_{i,a} = 1/(2 + e^{\|w^a_i\|_{2a_m}} + e^{-\|w^a_i\|_{2a_m}})$, and $\delta, \eta \ll 1$ are positive constants. The following theorem characterizes the probability that $a_j$, the PLA selected for student $j$ to take, is optimal. The proof is omitted due to space constraints.

**Theorem 3** Let $\epsilon \ll 1$ be a positive constant and $\delta, \eta, a_m$ and $b_{i,a}$ as defined above. Let $a^*_j$ denote the optimal PLA for student $j$, i.e., $a^*_j = \arg \max_a \sum_{i=1}^Q s_i \Phi \log(c^T_j w^a_i)$. Define constants $\Delta_a$ as $\Delta_{a,j} := \sum_{i=1}^Q s_i (\Phi \log(c^T_j w^a_i) - \Phi \log(c^T_j w^a_{i,j}))$ as the minimum gap between the expected credit student $j$ receives on Assessment 2 by taking the optimal PLA $a^*_j$ and a suboptimal PLA $a$. Assume that the students’ concept knowledge vectors are distributed as $c_j \sim \mathcal{N}(0, I_K)$, we have that if the number of times PLA $a$ has been taken $n_a$ satisfies

$$
n_a \geq \max \left\{4(K + 2\sqrt{K \log(1/\epsilon)} + 2\log(1/\epsilon)),
128K^2a_m^4 \log(K/\delta)(3 + 2\log(1 + 2a_m^2/\lambda_0))^2
\times \left(\sum_{i=1}^P s_i/b_{i,a}\right)^4 / \Delta_{a,j}^4 \right\}, \forall a.
$$
Algorithm 1: CLUB

**Input:** A set of student concept knowledge state estimates, \( c_j, j = 1, 2, \ldots \),
parameters \( \lambda_0, \delta, \eta, \epsilon \)

**Output:** PLA \( a_j \) for each student

\( \text{MLE}_{\text{all exist}} \leftarrow \text{False}, \ n_a \leftarrow 0, \forall a \)

for \( j \leftarrow 1 \) to \( \infty \) do

if \( \text{MLE}_{\text{all exist}} \) then

Estimate \( \hat{w}_{i}^{a}, \forall i, a \) according to (6.1)

\[ \Sigma_{a} \leftarrow \lambda_0 I_K + \sum_{s=1}^{n_a} c_{ja} c_{ja}^T, \forall a \]

\[ a_j \leftarrow \arg \max_a \sum_{i=1}^{Q} s_i (\Phi \log (c_j^T \hat{w}_{i}^{a}) + c_i(n_a) \sqrt{c_j^T \Sigma_{a}^{-1} c_j}) \]

else

Randomly select \( a_j \) among PLAs where \( \exists \ i \) s.t. \( \hat{w}_{i}^{a} \) does not exist

\( n_{a_j} \leftarrow n_{a_j} + 1 \)

\( \text{MLE}_{\text{all exist}} \leftarrow \text{True} \)

for \( a \leftarrow 1 \) to \( A \) do

for \( i \leftarrow 1 \) to \( Q \) do

if \( \hat{w}_{i}^{a} \) does not exist (verified via [145, Thm. 2]) then

\( \text{MLE}_{\text{all exist}} \leftarrow \text{False} \)
Then with probability at least $1 - 2(A - 1)Q(\epsilon + \delta + \eta)$, we have that $a_j = a_j^*$, i.e., CLUB selects the optimal PLA for student $j$.

6.3.2 A-CLUB: An algorithm in practice

Since in practice we do not know the values of the constants $\Delta_{a,j}$ and also need to set the parameters $\epsilon$, $\delta$, and $\eta$, Algorithm 1 and its theoretical guarantees are not directly applicable. Furthermore, as the number of students grows, the confidence bounds around the estimates of each PLA’s parameters might become overly pessimistic, causing the algorithm to over-explore [50]. Therefore, we now develop a second CLUB-like algorithm based on the following theorem on asymptotic normality of the MLE of each PLA’s parameters [49].

*Theorem 4 (Asymptotic normality property) Assume the Fisher information matrix $F_a := \sum_{s=1}^{n_a} \frac{c_{ja_s}c_{ja_s}^T}{2 + c_{ja_s}w_i^a + e^{-c_{ja_s}w_i^a}}$ is invertible for PLA $a$ and $n_a \geq K$. Then, the scaled error in the MLE of $w_i^a$ tends to be normally distributed, i.e., $n_a^{1/2} (\hat{w}_i^a - w_i^a) \xrightarrow{D} \mathcal{N}(0, F_a^{-1})$ as $n_a \to \infty$, where $\xrightarrow{D}$ denotes convergence in distribution.*

Intuitively, Theorem 4 states that, as the number of students grows large, the estimation error of the parameter $w_i^a$ for each PLA converges to a normally distributed random vector with zero mean and a covariance matrix that is a scaled inverse of the Fisher information matrix. Thus, we can build a confidence ellipsoid around the point estimate generated by (6.1) using Theorem 4, albeit asymptotically. In practice, since the true values of the parameters $w_i^a \forall i, a$ are unknown, we will use their estimates $\hat{w}_i^a$ to approximate the Fisher information matrix.

Armed with the confidence ellipsoid, we now have to compute the upper bound of the expected response of student $j$ on each question in Assessment 2 after taking
PLA \( a \). This problem corresponds to the following constrained optimization problem\(^\dagger\)

\[
\begin{align*}
\text{minimize} & \quad - \frac{1}{1 + e^{-c_j^T w}} \\
\text{subject to} & \quad (w - \hat{w}_i^a)^T F_a (w - \hat{w}_i^a) \leq \alpha / n_a,
\end{align*}
\]

where \( \alpha \) is a parameter controlling the size of the confidence ellipsoid and thus the amount of exploration. The solution to this problem is equal to the solution of the following problem

\[
\begin{align*}
\text{minimize} & \quad - c_j^T w \\
\text{subject to} & \quad (w - \hat{w}_i^a)^T F_a (w - \hat{w}_i^a) \leq \alpha / n_a, \quad (6.2)
\end{align*}
\]

since \( \Phi_{\text{log}}(x) = 1/(1 + e^{-x}) \) is a non-decreasing function. The following theorem characterizes the solution to (6.2).

**Theorem 5** The solution to (6.2) is given by

\[
\begin{align*}
w = \hat{w}_i^a + \sqrt{\frac{\mu}{n_a c_j^T F_a c_j}} F_a^{-1} c_j.
\end{align*}
\]

**Proof 3** First, note that the change of variable \( w \to w + \hat{w}_i^a \) does not change the optimization problem, since \( -c_j^T (w + \hat{w}_i^a) = -c_j^T w - c_j^T \hat{w}_i^a \) and \( c_j^T \hat{w}_i^a \) does not depend on \( w \). Therefore, we need to solve the following problem:

\[
\begin{align*}
\text{minimize} & \quad - c_j^T w \\
\text{subject to} & \quad w^T F_a w \leq \alpha / n_a.
\end{align*}
\]

The Lagrangian form of this problem is

\[
\begin{align*}
\text{minimize} & \quad - c_j^T w + \gamma (w^T F_a w - \alpha / n_a).
\end{align*}
\]

Taking the derivative w.r.t. \( w \) and setting it to zero gives

\[
w = F_a^{-1} c_j / (2\gamma). \]

Therefore, \( \gamma > 0 \), and so the complimentary slackness condition requires that

\[
w^T F_a w = \alpha / n_a.
\]

Substituting \( w \) back to this equation, we obtain

\[
c_j^T F_a^{-1} F_a F_a^{-1} c_j / (4\gamma^2) = \alpha / n_a \quad \Rightarrow \quad \gamma = \sqrt{\frac{n_a c_j^T F_a^{-1} c_j}{\alpha \gamma^2}} / 2.
\]

\(^\dagger\)We assume \( c_j \) is non-zero; otherwise we would simply select a PLA at random.
Thus, the solution to (6.2) is given by $w_i = \hat{w}_i^a + \sqrt{\frac{\alpha}{n_a c_j^T F^{-1}_a c_j}} F^{-1}_a c_j$.

Therefore, we obtain an upper bound for the expected grade for student $j$ on question $i$ after taking PLA $a$ as $\Phi\log(c_j^T \hat{w}_i^a + \sqrt{\alpha c_j^T F^{-1}_a c_j/n_a})$. We thus arrive at Algorithm 2, which we dub asymptotic CLUB (A-CLUB).

**Algorithm 2: A-CLUB**

**Input:** A set of student concept knowledge state estimates, $c_j$, $j = 1, 2, \ldots$, parameter $\alpha$

**Output:** PLA $a_j$ for each student

MLE$_{\text{all exist}}$ ← False, $n_a$ ← 0, $\forall a$

for $j$ ← 1 to $\infty$ do

if MLE$_{\text{all exist}}$ then

Estimate $\hat{w}_i^a$, $\forall i, a$ according to (6.1)

$F_a \leftarrow \lambda a I_K + \sum_{s=1}^{n_a} e^{c_i^T w_i^a} c_j^T e^{c_i^T w_i^a} w_i^a$, $\forall a$

$a_j \leftarrow \arg \max_a \sum_{i=1}^Q s_i \Phi \log(c_j^T \hat{w}_i^a + \sqrt{\alpha (c_j^T F^{-1}_a c_j)/n_a})$

else

Randomly select $a_j$ among PLAs where $\exists i$ s.t. MLE of $w_i^a$ does not exist

$n_{a_j} \leftarrow n_{a_j} + 1$

MLE$_{\text{all exist}}$ ← True

for $a$ ← 1 to $A$ do

for $i$ ← 1 to $Q$ do

if MLE does not exist for $w_i^a$ (verified via [145, Thm. 2]) then

MLE$_{\text{all exist}}$ ← False

end

end

end

end

end

6.3.3 CPT: A Bayesian algorithm

We now develop a Bayesian PLA selection algorithm using Thompson Sampling [31, 140]. This algorithm, contextual probit bandits with Thompson sampling (CPT),
uses the inverse probit link function instead of the inverse logit link function to model student responses. The reason is that the inverse probit link function enables a more computationally efficient rule to update the posterior distribution on $w_a$ than the Laplace approximation technique used for the inverse logit link function [117] associated with approximating the posterior with a Gaussian distribution.

Specifically, the graded response of the $s^{th}$ student to question $i$ in Assessment 2 after taking PLA $a$ is characterized as

$$p(Y_{i,j_a} = 1) = \Phi_{\text{pro}}(c_{j_a}^T w_a^i) = \int_{-\infty}^{c_{j_a}^T w_a^i} \mathcal{N}(t; 0, 1) dt,$$

where $s = 1, \ldots, n_a$,

where $\mathcal{N}(t; 0, 1)$ denotes a standard normal distribution and $\Phi_{\text{pro}}(\cdot)$ denotes the corresponding inverse probit link function. We put a prior distribution on $w_a$ as $\mathcal{N}(m_0, V_0)$. Consequently, the posterior distribution on $w_a$ can be approximated by $\mathcal{N}(m, V)$ where

$$m = m_0 + (2Y_{i,j_a} - 1) \frac{V_0 w_a}{\sqrt{1 + w_a^T V_0 w_a}} \Phi_{\text{pro}}(z),$$

$$V = V_0 - \frac{V_0 w_a^T V_0}{1 + w_a^T V_0 w_a} \left( z + \frac{\mathcal{N}(z)}{\Phi_{\text{pro}}(z)} \right) \frac{\mathcal{N}(z)}{\Phi_{\text{pro}}(z)},$$

with

$$z = (2Y_{i,j_a} - 1) \frac{m_0^T w_a}{\sqrt{1 + w_a^T V_0 w_a}},$$

where $m_0$ is initialized as an all-zero vector $0$ and $V_0$ is initialized as $\sigma^2 I$ where $I$ denotes the identity matrix. Details of this approximation can be found in [78].

Algorithm 3 summarizes the CPT algorithm.

### 6.4 Experiments

In this section, we validate our algorithms experimentally using two real-world educational datasets. We will compare the performance of Algorithms 2 and 3 against other
Algorithm 3: CPT

**Input:** A set of student concept knowledge state estimates, $c_j, j = 1, 2, \ldots,$

parameter $\sigma^2$

**Output:** PLA $a_j$ for each student

$m_i^a \leftarrow 0, V_i^a \leftarrow \sigma^2 I, \forall i, a$

for $j \leftarrow 1$ to $\infty$ do

| for $a \leftarrow 1$ to $A$ do |
| Sample $\hat{w}_i^a \sim N(m_i^a, V_i^a), \forall i.$ |
| $a_j \leftarrow \text{arg max}_a \sum_{i=1}^Q s_i \Phi_{\text{pro}}(c_j^T \hat{w}_i^a)$ |
| Update $m_i^{a_j}$ and $V_i^{a_j}, \forall i,$ according to (6.3) |

baseline (contextual) MAB algorithms. We do not compare Algorithm 1, since its theoretical bounds are usually too pessimistic in practice [50]. We will see that both A-CLUB and CPT improve students’ performance on follow-up assessments through a judicious choice of PLA.

### 6.4.1 Personalized cohort selection

We begin with an experiment on personalized cohort selection in a college physics course.

**Dataset** The dataset we study consists of the binary-valued graded responses in a semester-long college physics course administered on OpenStax Tutor [105] with $N = 39$ students answering 286 questions. Cognitive science experiments were conducted in this course to test the effect of spacing versus massed practice on the students’ long-term retrieval performance of knowledge [25]. For this purpose, the students were randomly divided into two cohorts with 19 and 20 students, respectively. There are a total of 11 weekly assessments and 3 review assessments throughout the course. In the first three assessments, both cohorts received the same set of assessment
questions. Starting from Assessment 4, apart from the same set of assessment questions both cohorts received on the concepts covered in the current week, each cohort also received some additional, different questions. One cohort received spaced practice questions related to the concepts they learned several weeks earlier, while the other cohort received massed practice questions related to the concepts they learned in the current week. Each cohort received some spaced practices and some massed practices throughout the semester so that the sets of questions assigned to each cohort were identical at the end.

**Experimental setup** Since students in Cohort 1 and Cohort 2 receive different sets of questions on Assessment 4, we investigate how this difference affects their learning on the concepts they learn next, i.e., their performance on Assessment 5. Treating each cohort as a PLA, our goal is to maximize the students’ performance on Assessment 5 by assigning them to the cohort (selecting the PLA) that benefits them the most. Therefore, in our setting the number of PLAs is $A = 2$.

We take the students’ graded responses to questions in Assessments 1–3 and apply SPARFA to estimate each student’s $K$-dimensional concept knowledge vector $c_j$, which we use as the context vectors. We set the number of concepts to $K = 3$.§

Since Cohorts 1 and 2 also receive different questions for Assessment 5 as part of the spacing vs. mass retrieval practice experiment on new concepts covered in Week 5, we take the set of $Q = 5$ questions shared between the two cohorts to evaluate their performance. Since MAB algorithms analyze students sequentially, we randomly permute the order of the students and average our results over 2000 random permutations.

**Evaluation method** We use the unbiased offline evaluation approach in [83, 84] to evaluate our algorithms. We use only the students that were actually assigned to

§In our experiments, we have found that the performance of our algorithms is robust to the number of concepts $K$ as long as $K \ll Q$. 
Figure 6.2: Average student credit on Assessment 5 vs. number of students used by the algorithms on the college physics dataset. The students’ performance on the follow-up assessment increases as the algorithms have access to more training data. Concretely, using data from 38 students, A-CLUB finds a PLA selection policy for students that is about 10% better than selecting randomly (as computed from the first and last data points on its curve). The same cohort as chosen by our algorithms and ignore the other students. This approach evaluates the decision making algorithms under the scenario where the data is collected in a specific “off-line, off-policy” manner, i.e., the data is collected by selecting PLAs for each student uniformly at random across every PLA, as opposed to a more typical MAB setting where PLAs are chosen for students sequentially given the observed follow-up assessment performance of previous students. Such a scenario fits our experimental setup well and yields an unbiased estimate of the expected reward for each student [84]. We use the students’ total credit on Assessment 5, i.e., $\sum_{i=1}^{Q} s_i Y_{i,j}$, as the metric to evaluate the performance of the algorithms.
Results and discussion  Figure 2 shows the students’ average credit (out of a full credit of 5) on Assessment 5 vs. the number of students the algorithms use for the algorithms A-CLUB, CPT, LINUCB [83], and UCB [9]. The parameters in every algorithm were tuned for best performance. We see that the average student credit increases as the number of students the algorithms observe increases, i.e., our proposed algorithms improve their PLA selection policy as they see more and more training data. As a concrete example, by comparing the average student credit at the first and last points on the curves, we see that our proposed CPT algorithm has found a policy that is about 5% better than selecting randomly, using data from 38 students. A-CLUB has found a policy that is about 10% better than selecting randomly, albeit with higher computational complexity.

Following the approach in [83], we also conduct an experiment by separating the dataset into a training set with 80% of the students and a test set with 20% of the students, to validate both the efficiency (performance on the training set) and efficacy (performance on the test set) of our algorithms. We train all algorithms on the training set and apply the learned PLA selection policy to the test set, and report the average student credits obtained on both sets. A-CLUB outperforms the other algorithms on both the training set and the test set, as shown in Table 6.1. Better performance on the test set means that A-CLUB learns a better policy than the other algorithms, while better performance on the training set means that it learns this policy very quickly as the amount of training data increases.

6.4.2 Personalized practice question selection

We now detail an experiment on personalized practice question selection in a high school physics course.

Dataset  The dataset we study consists of the binary-valued graded responses in the first half of a year-long high school physics course administered on OpenStax
A-CLUB CPT LINUCB UCB

<table>
<thead>
<tr>
<th></th>
<th>A-CLUB</th>
<th>CPT</th>
<th>LINUCB</th>
<th>UCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>3.69</td>
<td>3.66</td>
<td>3.68</td>
<td>3.65</td>
</tr>
<tr>
<td>Test set</td>
<td>3.89</td>
<td>3.73</td>
<td>3.77</td>
<td>3.70</td>
</tr>
</tbody>
</table>

Table 6.1: Performance comparison of A-CLUB and CPT against other baseline algorithms on personalized cohort selection on the college physics course dataset. A-CLUB outperforms the other algorithms in terms of average student credit on the follow-up assessment (out of a full credit of 5) on both the training and test sets.

Tutor [105] with $N = 16$ students answering 437 questions. There are a total of 24 assessments in the dataset. The first assessment consists of only reading tasks. After that, each assessment consists of a set of reading tasks and a set of practice questions. In addition to the practice questions that the instructor assigns to every student in the class, OpenStax Tutor randomly selects one personalized practice question for each student.

**Experimental setup**  We treat each personalized practice question selected by OpenStax Tutor as a PLA and study the impact of selecting one personalized practice question for each student in Assessment 4 on their subsequent performance on Assessment 5. The number of PLAs, i.e., the number of different personalized questions assigned to every student in Assessment 4, is $A = 5$. The context vectors, i.e., students’ concept knowledge vectors, are estimated from their responses to the questions in Assessments 2–3 using SPARFA. The other settings of this experiment are identical to those detailed in the first experiment.

**Results and discussion**  Table 6.1 compares A-CLUB and CPT against other baseline algorithms on both the training and testing sets in terms of average student credit on Assessment 5 with $Q = 3$ questions. The performance of A-CLUB and CPT are very close to UCB and better than LINUCB.
Table 6.2: Performance comparison of A-CLUB and CPT against other baseline algorithms on personalized practice question selection on the high school physics course dataset. A-CLUB, CPT, and UCB achieve comparable performance in terms of average student credit on the follow-up assessment (out of a full credit of 3).

6.5 Conclusions and Future Work

In this paper, we have proposed a contextual (multi-armed) bandits framework for PLA selection that aims to maximize the students’ immediate success on the follow-up assessment, given their latent concept knowledge estimated from their binary-valued graded responses to questions in previous assessments. We have proposed three algorithms to learn such a policy and have demonstrated that they achieve better or comparable performance against other baseline algorithms in terms of maximizing the students’ immediate success.

There are a number of avenues for future work. First, our context vectors are indexed by student features only, while in the general contextual bandits setting the contexts can be indexed by both student features and features of the learning resources. SPARFA-Trace [78], a recently developed framework for time-varying learning and content analytics, features an mechanism to analyze the content, quality, and difficulty of all kinds of learning resources (i.e., textbook sections, lecture videos, practice questions, etc). We can apply this approach to extract features from the learning resources that we can integrate into the contexts in our algorithms. Second, we can incorporate an additional PLA that corresponds to “no action”, due to the cost of taking actions, as considered in [138]. This extension would enable students with high knowledge on the concepts covered to avoid repeated practice and advance more
quickly to new concepts. Third, we are interested in integrating our approach into more sophisticated contextual bandit algorithms, e.g., [139] to reap further performance improvements.

6.6 Appendix: Proofs

In this section we prove Theorem 3. We need a couple of lemmas to do that.

The following lemma bounds the magnitude of the student concept knowledge vectors $c_j$.

**Lemma 6** Let $c_j \sim \mathcal{N}(0, I_K)$. Then, $P(||c_j||_2 > a_m\sqrt{K} + 2\sqrt{K}\log(1/\eta) + 2\log(1/\eta)) \leq \eta$.

**Proof 4** The proof follows trivially from the fact that $||c_j||_2^2$ is a $\chi^2$-distributed random variable with $K$ degrees of freedom and [82, Lem. 5].

The following lemma bounds the maximum eigenvalue of the matrix $\Sigma_a^{-1}$.

**Lemma 7** Let $\Sigma_a$ be given by Line 5 of LCMAB-UCB. Then,

$$\lambda_{\max}(\Sigma_a^{-1}) \leq (\sqrt{n_a} - \sqrt{K} - \sqrt{2\log(2/\epsilon)})^{-2}$$

with probability at least $1 - \epsilon$, where $\lambda_{\max}(\cdot)$ denote the maximum eigenvalue of a matrix.

**Proof 5** Denote $C_k = [c_{j_{k1}}, \ldots, c_{j_{kna}}] \in \mathbb{R}^{K \times na}$. Then, $C_k$ is a matrix with its entries being independent standard normal random variables. Then, using [146, Cor. 5.35], we have

$$\sigma_{\min}(C_k) \geq \sqrt{n_a} - \sqrt{K} - \sqrt{2\log(2/\epsilon)}$$
with probability at least $1 - \epsilon$, where $\sigma_{\min}(\cdot)$ denotes the minimum singular value of a matrix. Therefore, we have

$$\lambda_{\max}(\Sigma_a^{-1}) = \lambda_{\min}(\Sigma_a) = \lambda_{\min}(\lambda_0 I_K + C_k C_k^T) = (\lambda_0 + \sigma_{\min}^2(C_k))^{-1}$$

$$\leq \sigma_{\min}^2(C_k) \leq (\sqrt{n_a} - \sqrt{K} - \sqrt{2\log(2/\epsilon)})^{-2},$$

with probability at least $1 - \epsilon$.

The following lemma characterizes a property of the inverse logit link function $\Phi_{\log}(x) = 1/(1 + e^{-x})$.

**Lemma 8** If $\|c_j\|_2 \leq a_m$, then $\Phi'_{\log}(c_j^T w_{i,a}) \geq b_{i,a} = 1/(2 + e^{\|w_{i,a}\|_2 a_m} + e^{-\|w_{i,a}\|_2 a_m})$.

**Proof 6** Since $\Phi'_{\log}(x) = 1/(2 + e^x + e^{-x})$ is symmetric w.r.t. $x = 0$, we only need to study the case of $x > 0$. We have

$$\Phi''_{\log}(x) = -\frac{e^x - e^{-x}}{(2 + e^x + e^{-x})^2} = -e^{-x} \frac{e^x - 1}{(2 + e^x + e^{-x})^2} < 0$$

when $x > 0$. Therefore, $\Phi'_{\log}(x)$ is a strictly decreasing function on $x > 0$ and therefore

$$\Phi'_{\log}(c_j^T w_{i,a}) \geq \Phi'_{\log}(\|c_j\|_2 \|w_{i,a}\|_2) \geq \Phi'_{\log}(\|w_{i,a}\|_2 a_m) \geq \frac{1}{2 + e^{\|w_{i,a}\|_2 a_m} + e^{-\|w_{i,a}\|_2 a_m}}.$$

The following lemma (adapted from [50, Prop. 1]) bounds the estimation error of student $j$’s expected credit on question $i$ in Assessment 2.

**Lemma 9** Let $y_{i,j}^a = \Phi_{\log}(c_j^T w_{i,a})$ denote the true unknown expected response of student $j$ to question $i$ on Assessment 2 after performing PTA $a$ (probability the student answers the question correctly) and $\tilde{y}_{i,j}^a = \Phi_{\log}(\hat{c}_j^T \hat{w}_{i,a})$ denote the estimate of this expected response via (6.1). Then, with probability at least $1 - \delta - \eta$,

$$|\tilde{y}_{i,j}^a - y_{i,j}^a| \leq c_i(n_a) = \sqrt{2K(3 + 2\log(1 + 2a_m^2/\lambda_0))} \log n_a \log(K/\delta)/(2b_{i,a}) \sqrt{c_j^T \Sigma_a^{-1} c_j}.$$
c_i(n_a) with probability at least 1 − δ. Therefore,

\[ P(|\tilde{y}_{i,j}^a - y_{i,j}^a| \leq c_i(n_a)) \geq P(|\tilde{y}_{i,j}^a - y_{i,j}^a| \leq c(n_a) \mid \|c_j\|_2 \leq a_m \cap \|c_j\|_2 \leq a_m) \]

\[ = 1 - P(|\tilde{y}_{i,j}^a - y_{i,j}^a| > c_i(n_a) \mid \|c_j\|_2 \leq a_m \cup \|c_j\|_2 > a_m) \]

\[ \geq 1 - P(|\tilde{y}_{i,j}^a - y_{i,j}^a| > c_i(n_a) \mid \|c_j\|_2 \leq a_m) - P(\|c_j\|_2 > a_m) \]

\[ = 1 - \delta - \eta, \]

using the union bound. This finishes the proof.

Now we are ready to prove Thm. 3.

**Proof 8** First, we continue Lem. 9 to further bound the deviation of \( y_{i,j}^a \) from its estimated value. Observe that Lem. 7 gives

\[ \sqrt{c_j^T \Sigma^{-1}_a c_j} \leq \|c_j\|_2 \sqrt{\lambda_{\text{max}}(\Sigma^{-1}_b)} \leq \frac{a_m}{\sqrt{n_a} - \sqrt{K} - \sqrt{2 \log(2/\epsilon)}}, \]

we have that

\[ |\tilde{y}_{i,j}^a - y_{i,j}^a| \leq c_i(n_a) \sqrt{c_j^T \Sigma^{-1}_a c_j} \leq \frac{a_m \sqrt{2K(3 + 2 \log(1 + 2a_m^2/\lambda_0)) \log n_a \log(K/\delta)}}{2b_{i,a} (\sqrt{n_a} - \sqrt{K} - \sqrt{2 \log(2/\epsilon)})} \]

(6.4)

with probability at least 1 − \( \epsilon - \delta - \eta \), via a similar analysis as the proof of Thm. 3.

Then, we bound the predicted expected credit of the student on Assessment 2, aggregated over every question, from its true value. Observe that if \( s_i |\tilde{y}_{i,j}^a - y_{i,j}^a| \leq s_i c_i(n_a), \forall i \), then \( |\sum_{i=1}^Q s_i (\tilde{y}_{i,j}^a - y_{i,j}^a)| \leq \sum_{i=1}^Q s_i |\tilde{y}_{i,j}^a - y_{i,j}^a| \leq \sum_{i=1}^Q s_i c_i(n_a) \). Therefore,

\[ P(|\sum_{i=1}^Q s_i (\tilde{y}_{i,j}^a - y_{i,j}^a)| \leq \sum_{i=1}^Q s_i c_i(n_a)) \]

\[ \geq P(s_i |\tilde{y}_{i,j}^a - y_{i,j}^a| \leq s_i c_i(n_a), \forall i) \]

\[ = P(|\tilde{y}_{i,j}^a - y_{i,j}^a| \leq s_1 c_1(n_a) \cap \cdots \cap |\tilde{y}_{Q,j}^a - y_{Q,j}^a| \leq s_Q c_Q(n_a)) \]

\[ = 1 - P(|\tilde{y}_{i,j}^a - y_{i,j}^a| > s_1 c_1(n_a) \cup \cdots \cup |\tilde{y}_{Q,j}^a - y_{Q,j}^a| > s_Q c_Q(n_a)) \]

\[ \geq 1 - \sum_{i=1}^Q P(s_i |\tilde{y}_{i,j}^a - y_{i,j}^a| > s_i c_i(n_a)) \]

\[ = 1 - Q(\epsilon + \delta + \eta). \]
Now, we look at when does the algorithm LCMAB-UCB selects the optimal PTA \( a_j^* \) for student \( j \) to perform. We will compare a sub-optimal decision with the optimal one. The LCMAB-UCB algorithm will pick a suboptimal PTA \( a \) instead of the optimal action \( a_j^* \) if \( \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^a + \sum_{i=1}^{Q} s_i c_i(n_a) \sqrt{c_j^a \Sigma_{a}^{-1} c_j} > \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} + \sum_{i=1}^{Q} s_i c_i(n_{a_j^*}) \sqrt{c_j^{a_j^*} \Sigma_{a_j}^{-1} c_j} \). Call this event as \( e_{a,j} \) and let \( v_{a,j} := \sum_{i=1}^{Q} s_i c_i(n_{a_j^*}) \sqrt{c_j^{a_j^*} \Sigma_{a_j}^{-1} c_j} \). We have that

\[
P(e_{a,j}) = P\left( \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^a + v_{a,j} > \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} + v_{a_j^*,j} \right)
= 1 - P\left( \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^a + v_{a,j} \leq \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} + v_{a_j^*,j} \right)
= 1 - P(e_{a,j}^c).
\]

And observe at a sufficient condition for the event \( e_{a,j}^c \) to happen is that the following three conditions all hold:

\[
\sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^a < \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} + v_{a,j} \tag{6.5}
\]

\[
\sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} > \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} - v_{a_j^*,j} \tag{6.6}
\]

\[
2v_{a,j} < \Delta_a, \tag{6.7}
\]

since then we will have

\[
\sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^a + v_{a,j} < \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} + 2v_{a,j} = \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} - \Delta_a + 2v_{a,j}
< \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} < \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} + v_{a_j^*,j}.
\]

Define the events in (6.5), (6.6), and (6.7) as \( e_1 \), \( e_2 \), and \( e_3 \), respectively. We have

\[
P(e_{a,j}) = 1 - P(e_{a,j}^c) \geq 1 - P(e_1 \cap e_2 \cap e_3) = P(e_1^c \cup e_2^c \cup e_3^c) \leq P(e_1^c) + P(e_2^c) + P(e_3^c)
= P\left( \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^a > \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} + v_{a,j} \right) + P\left( \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} < \sum_{i=1}^{Q} s_i \tilde{y}_{i,j}^{a_j^*} - v_{a_j^*,j} \right) + P(2v_{a,j} > \Delta_a)
\leq \frac{Q(\epsilon + \delta + \eta)}{2} + \frac{Q(\epsilon + \delta + \eta)}{2} + P(2v_{a,j} > \Delta_a)
= Q(\epsilon + \delta + \eta) + P(2v_{a,j} > \Delta_a),
\]
by the union bound.

Now we investigate $P(2v_{a,j} > \Delta_{a,j})$. By our assumption on $n_a$ in Thm. 3 and some algebra, we have

\[
2v_{a,j} = 2 \sum_{i=1}^{Q} s_i c_i(n_a) \sqrt{c_j^T \Sigma_a^{-1} c_j}
\leq 2 \sum_{i=1}^{Q} s_i \frac{a_m \sqrt{2K(3 + 2 \log(1 + 2a_m^2/\lambda_0)) \log n_a \log(K/\delta)}}{2b_{i,a}(\sqrt{n_a} - \sqrt{K} - \sqrt{2 \log(2/\epsilon)})}
\leq 4 \sum_{i=1}^{Q} \frac{s_i a_m \sqrt{2K(3 + 2 \log(1 + 2a_m^2/\lambda_0)) \log(K/\delta)}}{2b_{i,a} \sqrt{n_a} \log n_a}
\leq \Delta_{a,j},
\]

where we have used (6.4), and the identity $\log x \leq x^s/s$ for $s > 0$. Therefore, $P(2v_{a,j} > \Delta_{a,j}) = 0$ and we have

\[
P(e_{a,j}) \leq Q(\epsilon + \delta + \eta).
\]

Therefore, we have the following bound on the probability that LCMAB-UCB selects the optimal PTA $a^*_j$ for student $j$ to perform as

\[
P(a_j = a^*_j) = P(\cap_k e\bar{e}_a^c) = 1 - P(\cup_a e_a) \geq 1 - \sum_a P(e_a) \geq 1 - (A - 1)Q(\epsilon + \delta + \eta),
\]

using the union bound, finishing up the proof.
Chapter 7

Conclusions and Future Work

In this thesis, we have proposed a set of machine learning techniques for personalized learning, covering the four most important building blocks of a PLS. The dealbreaker model and the SPARFA-Trace framework perform learning analytics by estimating the strengths and weaknesses in students’ knowledge. The SPARFA-Trace framework performs content analytics by estimating the content, quality, and difficulty of learning resources, i.e., questions, textbook sections, lecture videos, etc. The MLP framework performs automatic grading and feedback generation for open-response mathematical questions, enabling a PLS to process more complex types of student response data. The contextual (multi-armed) bandits framework performs scheduling by selecting a personalized learning action for each student given their learning history to maximize their immediate learning outcome. Together, these algorithms enable a PLS to close the feedback loop in learning as they take student data as input and produce analytics, feedback and personalized recommendation as output. Our work lays out a blueprint for the development of a fully automatic, scalable PLS.

There are a number of avenues for future work; most of them have been discussed at the end of every chapter. Additional avenues of future work can be broadly classified into three categories: theory, models & algorithms, and applications.

First, more rigorous theory is needed for some of the problems we study and some of the algorithms we develop. For example, an interesting research question is that whether there is an information-theoretic limit on the performance of collaborative filtering algorithms in terms of predicting unobserved entries in a matrix. In the 1-bit matrix completion work [40], such guarantees were proposed under some data
assumptions and a certain type of estimation model. It remains to be seen what assumptions are realistic for student response data and what family of algorithms can perform better estimation under such assumptions. Another interesting direction is to tie the theoretical guarantees in machine learning to cognitive science principles, e.g., [5], to investigate the limits of personalized learning, i.e., find an upper bound on the amount of improvement in students’ learning outcome through personalization.

Second, new algorithms are required for deeper understanding of the types of student responses and content. For example, the MLP framework is only able to generate feedback on the students’ likely locations of errors, but cannot obtain any insights on the types and causes of errors. Deeper understanding of, say, math, using neural network and deep learning techniques [160]. Another interesting direction is to account for multi-modal data in student learning, i.e., facial video recordings, lecture audio recordings, discussion forum posts, demographics, etc [155]. These data formats would enable us to go beyond graded responses and analyze other factors that affects learning, namely other cognitive factors such as engagement, stress, and social interaction dynamics [22], and non-cognitive factors like social background.

Third, new ways of experimenting with the proposed methods are also required. So far the data that are used are all collected either offline or via OpenStax Tutor [105]. Possible directions on this front include expanding the scope of personalization experiments on OpenStax Tutor and use other creative ways to collect data in psychology labs and crowdsourcing platforms, e.g., Amazon’s Mechanical Turk [4].
Bibliography


