BLAh: Boolean Logic Analysis for Graded Student Response Data

Andrew S. Lan, Andrew E. Waters, Christoph Studer, and Richard G. Baraniuk

Abstract—Machine learning (ML) models and algorithms can enable a personalized learning experience for students in an inexpensive and scalable manner. At the heart of ML-driven personalized learning is the automated analysis of student responses to assessment items. Existing statistical models for this task enable the estimation of student knowledge and question difficulty solely from graded response data with only minimal effort from instructors. However, most existing student–response models are generalized linear models, meaning that they characterize the probability that a student answers a question correctly through a linear combination of their knowledge and the question's difficulty with respect to each concept that is being assessed. Such models cannot characterize complicated, non-linear student–response associations and hence, lack human interpretability in practice. In this paper, we propose a non-linear student–response model called Boolean Logic Analysis (BLAh) that models a student's binary-valued graded response to a question as the output of a Boolean logic function. We develop a Markov chain Monte Carlo inference algorithm that learns the Boolean logic functions for each question solely from graded response data. A refined BLAh model improves the identifiability, tractability, and interpretability by considering a restricted set of ordered Boolean logic functions. Experimental results on a variety of real-world educational datasets demonstrate that BLAh not only achieves best-in-class prediction performance on unobserved student responses on some datasets but also provides easily interpretable parameters when questions are tagged with metadata by domain experts, which can provide useful feedback to instructors and content designers to improve the quality of assessment items.

Index Terms—Boolean logic, machine learning, Markov chain Monte Carlo, personalized learning, student–response data.

I. INTRODUCTION

Machine learning (ML)-based personalized education leverages sophisticated statistical models and algorithms to analyze and extract knowledge from student–response data. The computed model parameters and algorithm outputs can be used to automatically deliver personalized feedback to students and analytics to the instructor, in the process transforming the outdated "one-size-fits-all" educational practice into one that improves learning outcomes with less human intervention. As a consequence, research in personalized learning has been identified as a national priority within the United States [?].

Andrew S. Lan is with the Department of Electrical Engineering, Princeton University, Princeton, NJ; e-mail: andrew.lan@princeton.edu

Andrew E. Waters is with OpenStax, Houston, TX; e-mail: aew2@rice.edu

Christoph Studer is with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY; e-mail: studer@cornell.edu

Richard G. Baraniuk is with the Electrical and Computer Engineering Department, Rice University, Houston, TX; e-mail: richb@rice.edu

A. Student response modeling

Since the most prevalent form of student data consists of their responses to assessment items like practice, homework, and exam questions, the problem of student–response modeling lies at the heart of ML-based personalized learning research. Statistical models of student responses that model student performance are crucial to personalized learning, especially for learning analytics that automatically estimate the strengths and weaknesses of each of student's current knowledge. An automated learning system can then deliver personalized feedback and suggest a personalized learning action to each student [?], [?] that helps them learn more efficiently.

The problem of student–response modeling is complicated by the facts that (i) the model has to be as *data-driven* as possible, since manually labeling the content and difficulty of items is a labor-intensive process, (ii) the model has to be *robust* in how it predicts unobserved student responses (missing answers), and (iii) the model has to be *interpretable* in order to provide meaningful feedback to the students and instructors.

In this paper, we focus on *static* student-response models that can be used to estimate the students' current knowledge; such models do not capture the evolution of students' knowledge over time (such as learning and forgetting), which is the main subject of study in the line of work referred to as knowledge tracing (KT) [?], [?]. Therefore, the models we focus on in this paper are best suited for applications in which the students' knowledge are assumed to be static, i.e., in exams or quizzes. Through the years, a wide range of such static student–response models have been proposed in the literature, including linear item response theory (IRT) [?] and factor analysis [?], [?], [?], [?], [?], [?] models, and a few non-linear models [?], [?].

IRT methods model student responses to questions in terms of a small number of parameters. The simplest method is the 1PL IRT (or Rasch [?]) model that parameterizes each student and question by an ability and difficulty parameter, respectively; the probability of a correct response depends only on the student's ability minus the question's difficulty. The 2PL IRT model adds an additional parameter to each question that characterizes its ability to discriminate between students with high ability and low ability. The 3PL IRT model adds an additional parameter to each question that characterizes the probability of guessing the correct answer to the question without mastering the required knowledge; this is of particular importance in multiple-choice testing.

While easy to implement and often effective, IRT models are limited in their ability to analyze student responses to questions that require the mastery of diverse knowledge components

(which we term "concepts"). The multidimensional IRT (MIRT) model [?] attempts to address this shortcoming using ability and difficulty parameters that are *multi-dimensional*. This setting makes the MIRT model more suitable for questions in science, technology, engineering and mathematics (STEM) domains that often involve multiple skills.

Factor analysis models can be viewed as extensions to the basic MIRT model. Some factor analysis models employ additional hand-crafted features [?], [?], [?], [?], [?], while others are fully data-driven [?], [?]. These models typically provide superior prediction performance on unobserved student responses than simple IRT models [?], [?]. The recently proposed SPARFA (short for SPARse Factor Analysis) model [?] aims to provide interpretable model parameters while remaining purely data-driven.

B. Limits of linear student–response models

A common feature of current multi-dimensional student–response models is that they are *linear* (or affine). IRT-type models and factor analysis models belong to the class of *generalized linear models (GLMs)* (see, e.g., [?, Chap. 4]). That is, apart from a non-linear inverse logit or probit link function, they characterize the probability that a student answers a question correctly as a linear (or affine) function of the student's knowledge of the concepts underlying a question and the difficulty of that question. Thanks to their simple structure, linear models are easily applicable and tend to predict unobserved student responses accurately. However, linear models lack interpretability, since they allow a student to make up for weak knowledge on certain concepts with strong knowledge on other concepts when answering a question, an issue referred to as "explaining away" [?].

Indeed, many questions involve more complicated, non-linear interactions between the underlying concepts. To illustrate, consider the following two questions:

Question 1: Simplify the following expression:
$$(5x^2 \sin^2 x + 5x^2 \cos^2 x + 10x)/(x+2).$$

Question 2: Consider the two discrete-time signals:
$$x[n] = 2\delta[n] + 4\delta[n-1] + 6\delta[n-2] + 8\delta[n-3]$$
$$g[n] = \delta[n] + \delta[n-2].$$
Compute the circular convolution $y[n] = x[n] \otimes g[n]$.

Question 1 consists of two concepts: polynomial division and the trigonometric identity $\sin^2 x + \cos^2 x = 1$. In order to answer this question correctly, a student must have mastered both concepts. If a student does not know the trigonometric identity, then they will not be able to fully simplify the expression, regardless of their knowledge level on polynomial division, and vice versa. Therefore, we see that Question 1 tests students' knowledge on polynomial division *AND* trigonometry.

Question 2 also consists of two concepts: convolution and the discrete time Fourier transform (DTFT). On one hand, if a

student understands convolution, then they will be able to arrive at the correct answer by directly performing the convolution in the time domain. On the other hand, if a student understands the DTFT, then they would also be able to arrive at the correct answer by transforming both signals into the Fourier domain using the DTFT, performing an element-wise multiplication, and converting the result back into discrete time using the inverse DTFT. Note that this second solution approach does not require that the student understand how to compute a convolution. Therefore, we see that Question 2 tests students' knowledge on convolution *OR* the DTFT.

Linear response models, such as IRT and factor analysis models, are fundamentally ill-equipped to model such questions.

C. Recent developments in non-linear student–response models

Limited progress has been made recently on developing non-linear student–response models. We recently proposed the *dealbreaker* (DB) model [?] that characterizes the probability of a student answering a question correctly as a function of their weakest knowledge among all the tested concepts. In addition to enabling computationally efficient parameter inference, the DB model exhibits excellent prediction performance and enables interpretability of the model parameters. However, the DB model is limited to AND-like concept mechanisms and hence, is unable to model questions with other types of non-linear response–concept associations, e.g., the OR Question 2 above.

There exist related results on AND-like models, including the Q-matrix model [?] and the deterministic input, noisy AND (DINA) model [?]. The generalized DINA (G-DINA) model put forward in [?] posits that a student's probability of answering a question correctly depends on the probability of all possible combinations of their binary-valued concept knowledge states that characterize whether or not a student has mastered each concept. Although the G-DINA model is capable of learning more general, non-linear response–concept associations, its prediction performance is, in many cases, inferior to that of the DB model, despite DB's limitation to AND-type questions. Similar to the G-DINA model, the work in [?] tries to learn a conditional probability table relating a student's probability of answering each question correctly to their binary-valued concept knowledge states; however, their focus is to learn the prerequisite relationships among concepts.

Another important recent line of results employ neural networks for student-response data analysis, especially in the context of KT [?]. These methods are also non-linear but have limited interpretability and are therefore mainly useful for prediction tasks. Furthermore, it remains unclear whether they outperform even the most basic methods in terms of predicting unobserved student responses; see [?] for a detailed discussion of this issue.

D. Contributions

In this paper, we introduce *Boolean Logic Analysis* (BLAh), a modeling and inference framework capable of learning arbitrary non-linear response–concept associations that can be represented by *Boolean logic functions*. The BLAh model

TABLE I
EXAMPLE OF A BOOLEAN LOGIC FUNCTION WITH $K = 3$.

	$S_{n,1}$	$S_{n,2}$	$S_{n,3}$	$M_{i,n} = M_i(\mathbf{s}_n)$
$\mathbf{s}_1$	0	0	0	0
$\mathbf{s}_2$	0	0	1	0
$\mathbf{s}_3$	0	1	0	1
$\mathbf{s}_4$	0	1	1	1
$\mathbf{s}_5$	1	0	0	0
$\mathbf{s}_6$	1	0	1	0
$\mathbf{s}_7$	1	1	0	0
$\mathbf{s}_8$	1	1	1	1

characterizes the correctness of a student's response to a question as the output of a Boolean logic function, the input to which is a set of binary-valued student latent concept knowledge exhibition states, which is governed by the student's knowledge and the difficulty of the question with respect to each concept. See Table I for an example of such a Boolean logic function. The set of Boolean logic functions includes as special cases the examples introduced in Questions 1 and 2 above, namely AND functions (students have to master all of the underlying concepts in order to answer a question correctly) and OR functions (students need only master one of the underlying concepts to answer a question correctly). This latter case is a case that DB model does not cover.

The BLAh model provides substantial flexibility in modeling complex response–concept associations in student–response data but, unfortunately, suffers from the curse of dimensionality. Indeed, the total number of possible Boolean logic functions is a super-exponential function of the number of concepts. In order to mitigate this issue, we propose a restricted set of *ordered* Boolean logic functions, based on the observation that mastering more concepts will not negatively impact a student's chance to answer a question correctly. Such a restricted set of logic functions not only greatly reduces the number of Boolean logic functions and makes parameter inference more tractable, but it also provides improved model interpretability.

We develop a novel Markov chain Monte Carlo (MCMC) inference algorithm to learn arbitrary Boolean logic functions solely from binary-valued graded student–response data. We also develop an algorithm variant for the restricted set of ordered Boolean logic functions.

Using three real-world educational datasets, we demonstrate that the BLAh framework achieves comparable or superior performance to state-of-the-art student–response models (including the DB model) in terms of predicting unobserved responses, and significantly outperforms other models (including the G-DINA model, in particular). Moreover, we demonstrate that the restricted set of ordered Boolean logic function achieves some interpretability of the model parameters and reveals new insights on the response–concept associations in questions involving multiple concepts, given concept tags provided by domain experts. To demonstrate the interpretability of BLAh, we provide examples of the learned Boolean logic functions (including AND and OR functions) for several questions in a real-world student dataset.

The rest of the paper is organized as follows. Section II

details the BLAh model. Section III proposes the two parameter inference algorithms. Section IV provides our experimental results on real-world datasets. We conclude in Section V.

II. BLAh: BOOLEAN LOGIC ANALYSIS

We now describe BLAh. In Section II-A, we introduce the BLAh statistical model for predicting unobserved student responses. In Section II-B we extend it to focus on a restricted set of ordered Boolean logic functions that greatly enhances the learned logic function's interpretability for educational datasets.

A. The BLAh statistical model

Let N denote the number of students, Q denote the number of questions, and K denote the number of latent concepts involved in the questions, following the convention in [**?**], with $K \ll Q, N$. Let $C_{k,j} \in \mathbb{R}$ denote student j's knowledge level on concept k, with large, positive values representing high knowledge, and let $\mu_{i,k} \in \mathbb{R}$ denote question i's intrinsic difficulty level on concept k, with large, positive values representing high difficulty. When a student responds to a question, these parameters dictate whether the student is able to exhibit enough knowledge to successfully solve different parts of the question involving different concepts. Correspondingly, let $S_{i,j,k}$ denote the binary-valued latent concept knowledge exhibition state of student j on concept k when responding to question i, given by

$$p(S_{i,j,k} = 1) = \Phi(C_{k,j} - \mu_{i,k}), \tag{1}$$

where $\Phi(\cdot) = \int_{-\infty}^{x} \mathcal{N}(t; 0, 1) \mathrm{d}t$ denotes the inverse probit link function and $\mathcal{N}(t; 0, 1)$ denotes the standard normal distribution.

$S_{i,j,k} = 1$ means that student j shows mastery of concept k when responding to question i, and $S_{i,j,k} = 0$ means that student j does not show mastery of concept k when responding to question i. Therefore, the knowledge state of student j when responding to question i can be fully characterized by the vector $\mathbf{s}_{i,j} = [S_{i,j,1}, \ldots, S_{i,j,k}, \ldots, S_{i,j,K}]^T \in \{0, 1\}^K$. Such a model enables a student's mastery on a concept to exhibit different outcomes in different questions (depending on the difficulty of the question). This feature makes BLAh more flexible than modeling each student as a single set of binary-valued latent knowledge mastery states when responding to every question.

In the BLAh statistical model, the key that governs students' responses to each question is a *Boolean logic function* that maps the binary-valued latent knowledge exhibition state vector $\mathbf{s}_{i,j}$ to the binary-valued graded response of student j to question i, $Y_{i,j}$. $Y_{i,j} = 1$ denotes a correct response, and $Y_{i,j} = 0$ denotes an incorrect response. The Boolean logic function corresponding to question i, $M_i(\cdot)$, is parameterized by a vector of truth-table entries $\mathbf{m}_i = [M_{i,1}, \ldots, M_{i,n}, \ldots, M_{i,2^K}]^T \in \{0, 1\}^{2^K}$. See Table I for an example truth-table for $K = 3$[1].

---

[1] The most significant bit is the first bit, the least significant bit is the last bit, and the latent states $\mathbf{s}_n$ are ordered by increasing value.

Given student j's latent knowledge exhibition state vector $\mathbf{s}_{i,j}$ when responding to question i, their binary-valued graded response $Y_{i,j}$ is simply equal to $M_i(\mathbf{s}_{i,j})$, i.e.,

$$p(Y_{i,j}|\mathbf{s}_{i,j}) = I(M_i(\mathbf{s}_{i,j}) = Y_{i,j}), \qquad (i,j) \in \Omega_{\text{obs}},$$

where $I(\cdot)$ denotes the indicator function, and Ω_{obs} denotes the index set of student responses that are observed in the case that not every student answers every question.

Equivalently, we can also marginalize out the latent knowledge exhibition state vector $\mathbf{s}_{i,j}$ and write the probability of a graded response $Y_{i,j}$ as

$$\begin{aligned}
&p(Y_{i,j}|C_{k,j}, \mu_{i,k}, \forall k, M_i)\\
&= \sum_{\mathbf{s}_n \in \{0,1\}^K} p(Y_{i,j}|\mathbf{s}_n)p(\mathbf{s}_n|C_{k,j}, \mu_{i,k}, \forall k)\\
&= \sum_{\mathbf{s}_n \in \{0,1\}^K} I(M_i(\mathbf{s}_n) = Y_{i,j})p(\mathbf{s}_n|C_{k,j}, \mu_{i,k}, \forall k)\\
&= \sum_{n:M_i(\mathbf{s}_n)=Y_{i,j}} \prod_{k=1}^K \Phi(C_{k,j} - \mu_{i,k})^{S_{n,k}}\\
&\qquad\qquad \times (1 - \Phi(C_{k,j} - \mu_{i,k}))^{1-S_{n,k}}. \quad (2)
\end{aligned}$$

In words, the probability of a particular graded response is the sum of the probabilities of the student being in the latent concept knowledge exhibition states that, when used as inputs to the Boolean logic function of the question, produce the corresponding graded response as output.

The goal of the inference algorithm we develop in Section III is to estimate the parameters $C_{k,j}, \forall k,j, \mu_{i,k}, \forall i,k$, and $\mathbf{m}_i$, $\forall i$ given only the observed graded responses $Y_{i,j}, (i,j) \in \Omega_{\text{obs}}$.

B. Restricted set of ordered Boolean logic functions

The BLAh statistical model with arbitrary Boolean logic functions proposed in the previous section suffers from the curse of dimensionality. That is, the total number of possible binary-valued student latent concept knowledge exhibition states is 2^K, and the total number of possible logic functions is $M_i(\cdot)$ is 2^{2^K}, which grows at a super-exponential rate with the number of concepts K. Therefore, parameter inference with this model is computationally intractable except for extremely small values of K. Moreover, identifiability issues arise, since one can flip the signs of the variables $C_{k,j}$ and $\mu_{i,k}, \forall i,j,k$ and also simultaneously flip the truth-table entries ($\mathbf{m}_i, \forall i$) in the Boolean logic functions and still arrive at the same likelihood for the observed responses. These identifiability issues limits the interpretability of the BLAh model parameters with arbitrary Boolean logic functions.

In order to alleviate these issues, we introduce a restricted set of Boolean logic functions for student–response data, based on the following realistic assumption:

Assumption 1. *Mastery of more concepts does not negatively impact a student's chance of answering a question correctly.*

	$S_{n,1}$	$S_{n,2}$	$S_{n,3}$	$M_{i,n} = M_i(\mathbf{s}_n)$
$\mathbf{s}_1$	0	0	0	0
$\mathbf{s}_2$	0	0	1	1
$\mathbf{s}_3$	0	1	0	0
$\mathbf{s}_4$	0	1	1	1
$\mathbf{s}_5$	1	0	0	0
$\mathbf{s}_6$	1	0	1	1
$\mathbf{s}_7$	1	1	0	0
$\mathbf{s}_8$	1	1	1	1

This corresponds to [**?**, Assumption 3]. In the context of the BLAh model, we define the following restricted set of *ordered* Boolean logic functions as

$$\mathcal{M}_o = \{M(\cdot): M(\mathbf{s}_p) \leq M(\mathbf{s}_q) \ \forall p,q \ \text{such that} \ \mathbf{s}_p \preceq \mathbf{s}_q\}, \quad (3)$$

where $\mathbf{s}_p \preceq \mathbf{s}_q$ is defined by

$$\mathbf{s}_p \preceq \mathbf{s}_q \quad \text{if} \quad S_{p,k} \leq S_{q,k} \ \forall k.$$

Intuitively, $\mathcal{M}_o$ defines a restricted set of Boolean logic functions that satisfy Assumption 1. Concretely, for every pair of binary-valued latent knowledge exhibition state vectors satisfying $\mathbf{s}_p \preceq \mathbf{s}_q$, where $\mathbf{s}_q$ corresponds to a higher knowledge state than (or as high as) $\mathbf{s}_p$, their corresponding truth-table values specify that higher knowledge ($\mathbf{s}_q$) leads to an equally good or better outcome on a question than $\mathbf{s}_p$. For example, with $\mathbf{s}_p = [0,1,0]$, corresponding to mastery of concept 2 but not concepts 1 and 3, and $\mathbf{s}_q = [1,1,0]$, corresponding to mastery of concepts 1 and 2 but not concept 3), we have $M(\mathbf{s}_p) \leq M(\mathbf{s}_q)$. Based on this definition, it is easy to see that the logic function in Table I does not belong to the set of ordered Boolean logic functions $\mathcal{M}_o$ ($\mathbf{s}_3 \preceq \mathbf{s}_7$ but $M(\mathbf{s}_3) > M(\mathbf{s}_7)$), while the one in Table II does.

Such a restricted set of ordered Boolean logic functions is significantly smaller than the set of arbitrary Boolean logic functions on length-K binary-valued latent knowledge exhibition state vectors. For example, for $K = 4$, a simple calculation[2] shows that the restricted set has $|\mathcal{M}_o| = 168$ functions, much smaller than the total of $2^{2^4} = 65536$ possible functions. Therefore, by restricting ourselves to Boolean logic functions in the restricted set, we can significantly reduce the size of the parameter space that must be explored by an MCMC inference algorithm, thereby reducing the resulting computational complexity. Furthermore, such a set of ordered Boolean logic functions eliminates the identifiability issue mentioned above, since a logic function generated by flipping every value in the truth-table of an ordered function in the restricted set will not be ordered, in general. In Section III-B, we will detail a corresponding MCMC inference algorithm that adapts the MCMC inference algorithm for arbitrary Boolean logic functions to functions in the restricted set $\mathcal{M}_o$.

[2]We simply enumerate every possible Boolean logic function and check whether it is ordered to calculate the size of the restricted set; however, such an approach is not presently computationally tractable for $K \geq 5$.

III. INFERENCE ALGORITHMS FOR BLAH

We now develop two MCMC inference algorithms to learn the BLAh model parameters for arbitrary Boolean logic functions and the restricted set of ordered functions given only binary-valued graded student responses.

A. Inference algorithm for arbitrary Boolean logic functions

We develop a Metropolis-Hastings (MH)-within-Gibbs sampling algorithm for parameter inference for the BLAh model with arbitrary Boolean logic functions. Since the latent knowledge exhibition states $S_{i,j,k}$ are binary-valued, we introduce the slack variables $Z_{i,j,k} \in \mathbb{R}, \forall i, j, k$ for the probit likelihood model [?], which enables us to sample efficiently from the posterior distributions of $C_{k,j}$ and $\mu_{i,k}$. Specifically, we define the slack variables as

$$Z_{i,j,k} = C_{k,j} - \mu_{i,k} + \epsilon_{i,j,k}, \quad \epsilon_{i,j,k} \sim \mathcal{N}(0,1),$$

and

$$S_{i,j,k} = \begin{cases} 1 & \text{if } Z_{i,j,k} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The prior distributions for each latent variable are given by

$$C_{k,j} \sim \mathcal{N}(\mu_c, \sigma_c^2),$$
$$\mu_{i,k} \sim \mathcal{N}(\mu_\mu, \sigma_\mu^2),$$
$$\mathbf{m}_i \sim \mathcal{U}(2^{2^K}),$$

where $\mathcal{U}(2^{2^K})$ denotes the uniform distribution over a set of 2^{2^K} discrete values, i.e., we put an uninformative prior over the set of arbitrary Boolean logic functions so that each Boolean logic function is equally likely.

After randomly initializing the latent variables $C_{k,j}$, $\mu_{i,k}$, and $\mathbf{m}_i$ according to their prior distributions, our MH-within-Gibbs sampling algorithm performs the following steps in each iteration:

1) *Impute* $Y_{i,j}$, $(i,j) \in \Omega_{obs}^c$: If there are some unobserved graded responses, i.e., $\Omega_{\text{obs}} \neq \{1, \ldots, Q\} \times \{1, \ldots, N\}$, then we impute their values following the approach detailed in [?]. We first calculate $p_{i,j} = p(Y_{i,j} = 1 | C_{k,j}, \mu_{i,k}, \forall k, M_i), \forall (i,j) \in \Omega_{\text{obs}}^c$ from (2), and then sample $Y_{i,j}, \forall (i,j) \in \Omega_{\text{obs}}^c$ from the Bernoulli distribution $\text{Ber}(p_{i,j})$.

2) *Sample* $\mathbf{m}_i$: Since the total number of arbitrary logic functions is 2^{2^K}, the cardinality of the set of such Boolean logic functions explodes even for very small values of the numbers of concepts K. Therefore, it is computationally intractable to sample directly from the posterior distribution of $\mathbf{m}_i$ (2^{2^K} likelihood calculations in each iteration). Instead, we use a more tractable Metropolis-Hastings step to sample from $\mathbf{m}_i$ (one likelihood calculation in each iteration). Specifically, for question i, we propose the parameter of a new logic function, $\mathbf{m}_i'$, by randomly flipping each entry of the parameter vector of the old logic function $\mathbf{m}_i$ with probability p_F. We then evaluate the data likelihood for both the new logic function and the old logic function via $p' = \prod_{j=1}^N p(Y_{i,j} | C_{k,j}, \mu_{i,k}, \mathbf{m}_i')$

and $p = \prod_{j=1}^N p(Y_{i,j} | C_{k,j}, \mu_{i,k}, \mathbf{m}_i)$. We accept this new proposal with the following acceptance probability

$$\min\left\{1, \frac{p'}{p}\right\},$$

since the proposal distribution is symmetric and the prior distribution on $\mathbf{m}_i$ is uniform.

3) *Sample* $\mathbf{s}_{i,j}$: In order to sample from the posterior distributions of $C_{k,j}$ and $\mu_{i,k}$, we need to first sample the slack variables $S_{i,j,k}$ and $Z_{i,j,k}$. We begin by sampling the latent knowledge exhibition state vectors $\mathbf{s}_{i,j}$, whose posterior distribution is given by

$$p(\mathbf{s}_{i,j} | Y_{i,j}, C_{k,j}, \mu_{i,k}, \forall k)$$
$$\propto p(Y_{i,j} | \mathbf{s}_{i,j}, M_i) p(\mathbf{s}_{i,j} | C_{k,j}, \mu_{i,k}, \forall k)$$
$$= I(Y_{i,j} = M_i(\mathbf{s}_{i,j}))$$
$$\times \prod_{k=1}^K \Phi(C_{k,j} - \mu_{i,k})^{S_{i,j,k}} (1 - \Phi(C_{k,j} - \mu_{i,k}))^{1-S_{i,j,k}}.$$

Therefore, we can sample the values of $\mathbf{s}_{i,j}$ from the multinomial distribution defined above.

4) *Sample* $Z_{i,j,k}$: We sample the slack variable $Z_{i,j,k}$; its posterior distribution is given by

$$p(Z_{i,j,k} | S_{i,j,k}, C_{k,j}, \mu_{i,k})$$
$$\propto p(S_{i,j,k} | Z_{i,j,k}) p(Z_{i,j,k} | C_{k,j}, \mu_{i,k})$$
$$= I(Z_{i,j,k} \geq 0 \text{ if } S_{i,j,k} = 1, Z_{i,j,k} \leq 0 \text{ if } S_{i,j,k} = 0)$$
$$\times \mathcal{N}(C_{k,j} - \mu_{i,k}, 1)$$
$$= \mathcal{N}^{\pm}(C_{k,j} - \mu_{i,k}, 1),$$

where $\mathcal{N}^{\pm}(\cdot)$ denotes the truncated normal distribution with $+$ corresponds to truncation on the interval $[0, \infty)$ and $-$ corresponds to truncation on $(-\infty, 0]$.

5) *Sample* $C_{k,j}$ and $\mu_{i,k}$: We sample the variable $C_{k,j}$ from its posterior distribution

$$p(C_{k,j} | Z_{i,j,k}, \mu_{i,k}, \forall i)$$
$$\propto \prod_{i=1}^Q p(Z_{i,j,k} | C_{k,j}, \mu_{i,k}) p(C_{k,j})$$
$$= \prod_{i=1}^Q \mathcal{N}(Z_{i,j,k} | C_{k,j}, \mu_{i,k}) \mathcal{N}(C_{k,j} | \mu_c, \sigma_c^2)$$
$$\propto \mathcal{N}(\mu_c', \sigma_c^{2'}),$$

where $\sigma_c^{2'} = (Q + 1/\sigma_c^2)^{-1}$ and $\mu_c' = \sigma_c^{2'}(\sum_{i=1}^Q Z_{i,j,k} + \mu_{i,k} + \mu_c/\sigma_c^2)$. Using a similar procedure, we can sample the variables $\mu_{i,k}$ from their posterior distribution

$$p(\mu_{i,k} | Z_{i,j,k}, C_{k,j}, \forall j) \propto \mathcal{N}(\mu_\mu', \sigma_\mu^{2'}),$$

where $\sigma_\mu^{2'} = (N + 1/\sigma_\mu^2)^{-1}$ and $\mu_\mu' = \sigma_\mu^{2'}(\sum_{j=1}^N C_{k,j} - Z_{i,j,k} + \mu_\mu/\sigma_\mu^2)$.

We repeat the above sampling steps for T iterations (including a burn-in period) to generate posterior statistics for the latent variables of interest $C_{k,j}$, $\mu_{i,k}$, and $\mathbf{m}_i$. The parameters μ_c, σ_c^2, μ_μ, σ_μ^2 are hyperparameters, and p_F is an algorithm parameter. Details on the selection of these parameters are discussed in Section IV.

B. Inference algorithm on the restricted set of ordered Boolean logic functions

Despite the fact that the number of ordered Boolean logic functions is much smaller than the number of arbitrary Boolean logic functions, it is still not efficient to sample directly from its posterior ($|\mathcal{M}_o|$ likelihood calculations in every iteration of the MCMC algorithm). On the other hand, the MH-within-Gibbs inference algorithm developed in the previous section for arbitrary Boolean logic functions cannot be directly applied to inference with the restricted set $\mathcal{M}_o$ introduced in Section II-B. The reason is that randomly flipping the entries in the parameter vector $\mathbf{m}_i$ of a function $M_i \in \mathcal{M}_o$ can result in a logic function $M_i' \notin \mathcal{M}_o$. An obvious approach is to repeat the MH proposal step multiple times until a newly proposed function also belongs to $\mathcal{M}_o$. Such an approach, however, is very computationally inefficient, since the size of the restricted set is much smaller than that of the set of arbitrary Boolean logic functions, which means that most proposals will be rejected, leading to a very inefficient exploration of the parameter space. We therefore need another, more efficient approach to adapt the MH step (Step 2) to the restricted set.

We start by noting that, since for a small value of K the size of the restricted set $\mathcal{M}_o$ is small enough (e.g., $|\mathcal{M}_o| \leq 168$ for $K \leq 4$), we can enumerate all of the ordered Boolean logic functions. Then, if we can calculate the exact transition probability from each ordered logic function to every other one, then we can employ a MH proposal step using these transition probabilities that guarantees a newly proposed logic function to be within the set $\mathcal{M}_o$, leading to a better sampling algorithm that explores the parameter space more efficiently. We emphasize that most questions in real-world assessments involve no more than a few (e.g., four) concepts, thus enabling the use of a more efficient sampling algorithm. For questions involving more than four concepts, we can still apply the regular MH step and simply reject every proposed logic function that is unordered, with lower computational efficiency.

The key to this efficient sampling algorithm is to calculate the transition probabilities between pairs of ordered Boolean logic functions induced by randomly flipping the entries of $\mathbf{m}_i$. However, this calculation is complicated by the fact that if we simply randomly flip the entries of $\mathbf{m}_i$ independently, then there can be multiple different combinations of flips that result in the same logic function M_i'. As a concrete example, in Table III, a direct comparison of M_i and M_i' shows that both $M_{i,5}$ and $M_{i,6}$ have to be flipped; however, since the new logic function also has to be ordered, only flipping $M_{i,5}$ will also lead to M_i'. The reason is that, since $\mathbf{s}_5 \preceq \mathbf{s}_6$, $M_{i,6}$ is also mandated to flip in order for M_i' to remain ordered upon the flipping of $M_{i,5}$, according to the definition of the ordered Boolean logic function (3).

We now detail an efficient method to calculate the transition probabilities between pairs of ordered Boolean logic functions. We will first need a method to generate the entire restricted set $\mathcal{M}_o$. We first generate the *index set* on the latent knowledge exhibition states

$$\mathcal{I} = \{(p,q) : \mathbf{s}_p \preceq \mathbf{s}_q\}.$$

TABLE III
EXAMPLE OF A TRANSITION FROM ONE LOGIC FUNCTION IN $\mathcal{M}_o$ TO ANOTHER. FLIPS IN PARENTHESES DENOTE NON-ESSENTIAL FLIPS, WHILE THE OTHER FLIPS ARE ESSENTIAL FLIPS.

$\mathbf{s}_n$				$M_{i,n}$	flip$_{0\to1}$	flip$_{1\to0}$	$M_{i,n}'$
$\mathbf{s}_1$	0	0	0	0			0
$\mathbf{s}_2$	0	0	1	0			0
$\mathbf{s}_3$	0	1	0	0			0
$\mathbf{s}_4$	0	1	1	1		$\to$	0
$\mathbf{s}_5$	1	0	0	0	$\to$		1
$\mathbf{s}_6$	1	0	1	0	$(\to)$		1
$\mathbf{s}_7$	1	1	0	1			1
$\mathbf{s}_8$	1	1	1	1			1

Using the index set, we can simply enumerate all Boolean logic functions and construct $\mathcal{M}_o$ as defined in (3).

Now we can start to calculate the transition probabilities between pairs of ordered Boolean logic functions. As an illustrative example, consider the transition from a logic function in $\mathcal{M}_o$, parameterized by $\mathbf{m}_i$, to another, parameterized by $\mathbf{m}_i'$, as illustrated in Table III. First, the entries in M_i that need to be flipped in order to result in another logic function $M_i' \in \mathcal{M}_o$ can be divided into two classes: flips from 0 to 1 and flips from 1 to 0, as indicated by the corresponding columns in Table III. These classes can be treated separately. Second, taking a closer look at flips from 0 to 1, we see that there is a subtle difference between the flips to $M_{i,5}$ and $M_{i,6}$, as described above; since $\mathbf{s}_5 \preceq \mathbf{s}_6$, flipping $M_{i,5}$ mandates the flipping of $M_{i,6}$ in order for M_i' to remain in $\mathcal{M}_o$; the flipping of $M_{i,6}$ does not mandate the flipping of $M_{i,5}$. Therefore, as described above, flipping $M_{i,5}$ alone results in the same new ordered Boolean logic function M_i' as flipping both $M_{i,5}$ and $M_{i,6}$. This subtle difference is key to calculating the transition probability from M_i to M_i'. To emphasize this difference, we call the flip to $M_{i,5}$ an *essential* flip, since it is necessary for the transition from M_i to M_i'; we call the flip to $M_{i,6}$ a *non-essential* flip. Denoting the set of all 0 to 1 flips as $\mathcal{F}_{0\to1}(M_i \to M_i')$, we can formally define the set of essential 0 to 1 flips as

$$\mathcal{F}_{0\to1}(M_i \to M_i')^{\mathrm{E}} = \{n : M_{i,n} = 0,\ M_{i,n}' = 1,\ \text{and}$$
$$M_{i,m}' = 0\ \forall\ (m,n) \in \mathcal{I}\} \subseteq \mathcal{F}_{0\to1}(M_i \to M_i'),$$

which is a set consisting of all the flips that are essential to go from M_i to M_i'; the superscript E stands for "essential". The set of non-essential 0 to 1 flips that contains flips that are not essential is then given by

$$\mathcal{F}_{0\to1}(M_i \to M_i')^{\mathrm{NE}} = \mathcal{F}_{0\to1}(M_i \to M_i') \backslash \mathcal{F}_{0\to1}(M_i \to M_i')^{\mathrm{E}},$$

where the superscript NE stands for "non-essential". A similar definition for sets of essential and non-essential 1 to 0 flips is defined analogously. Using this notation, we detail the procedure to find all essential and non-essential 0 to 1 and 1 to 0 flips in order to go from an ordered logic function M_i to another ordered logic function M_i' in Algorithm 1.

In Algorithm 1, "fq01" is short for "flip queue 0 to 1", and "ef01" is short for "essential flips 0 to 1". "fq10" and "ef10" are defined analogously. Algorithm 1 works as follows. We

---

Algorithm 1: FIND ALL (NON-)ESSENTIAL FLIPS
BETWEEN TWO ORDERED BOOLEAN LOGIC FUNCTIONS

Input: Logic functions M_i, M_i' in the restricted set $\mathcal{M}_o$

Output: Set of essential flips $\mathcal{F}(M_i \to M_i')^{\mathrm{E}}$, and set of
non-essential flips $\mathcal{F}(M_i \to M_i')^{\mathrm{NE}}$

(fq01) $\leftarrow \{n : M_i(\mathbf{s}_n) = 0 \text{ and } M_i'(\mathbf{s}_n) = 1\}$

(ef01) $\leftarrow \{n : M_i(\mathbf{s}_n) = 0 \text{ and } M_i'(\mathbf{s}_n) = 1\}$

while *fq01 is not empty* **do**

    for $j : (fq01[1], j) \in \mathcal{I}$ **do**

        ef01 $\leftarrow$ ef01 $\setminus j$

        fq01 $\leftarrow$ fq01 $\cup \{j\}$

    fq01 $\leftarrow$ fq01 $\setminus$ fq01[1]

(fq10) $\leftarrow \{n : M_i(\mathbf{s}_n) = 1 \text{ and } M_i'(\mathbf{s}_n) = 0\}$

(ef10) $\leftarrow \{n : M_i(\mathbf{s}_n) = 1 \text{ and } M_i'(\mathbf{s}_n) = 0\}$

while *fq10 is not empty* **do**

    for $j : (j, fq10[1]) \in \mathcal{I}$ **do**

        ef10 $\leftarrow$ ef10 $\setminus j$

        fq10 $\leftarrow$ fq10 $\cup \{j\}$

    fq10 $\leftarrow$ fq10 $\setminus$ fq10[1]

$\mathcal{F}(M_i \to M_i')^{\mathrm{E}} \leftarrow ef01 \cup ef10$

$\mathcal{F}(M_i \to M_i')^{\mathrm{NE}} \leftarrow (\{n : M_i(\mathbf{s}_n) = 0 \text{ and } M_i'(\mathbf{s}_n) = 1\} \setminus ef01) \cup (\{n : M_i(\mathbf{s}_n) = 1 \text{ and } M_i'(\mathbf{s}_n) = 0\} \setminus ef10)$

---

start by adding every flip from 0 in the original logic function M_i to 1 in the new logic function M_i' into fq01 and ef01. Then, we iteratively take the first flip out of fq01 and eliminate all the flips that are mandated by this flip from ef01, according to $\mathcal{I}$, while also adding every such flip to the end of fq01. After fq01 becomes empty, the set containing the remaining elements in ef01 is exactly the set of essential flips $\mathcal{F}_{0 \to 1}(M_i \to M_i')^{\mathrm{E}}$, while $\mathcal{F}_{0 \to 1}(M_i \to M_i')^{\mathrm{NE}}$ is then simply the set containing flips that are eliminated from ef01. The above procedure is then repeated to obtain the sets of essential and non-essential flips from 1 to 0.

Once we identify the set of essential flips $\mathcal{F}(M_i \to M_i')^{\mathrm{E}}$, set of non-essential flips $\mathcal{F}(M_i \to M_i')^{\mathrm{NE}}$, and the set of non-flips $\mathcal{NF}(M_i \to M_i') = \{n : M_i(\mathbf{s}_n) = M_i'(\mathbf{s}_n)\}$), we can calculate the *unnormalized* probability to go from logic function M_i to M_i', *induced by randomly flipping the truth-table values with flipping probability* p_F defined by

$$q(M_i \to M_i') = p_F^{|\mathcal{F}(M_i \to M_i')^{\mathrm{E}}|}(1 - p_F)^{|\mathcal{NF}(M_i \to M_i')|} \times$$
$$\sum_{k=0}^{|\mathcal{F}(M_i \to M_i')^{\mathrm{NE}}|} C_k^{|\mathcal{F}(M_i \to M_i')^{\mathrm{NE}}|} p_F^k (1 - p_F)^{|\mathcal{F}(M_i \to M_i')^{\mathrm{NE}}| - k},$$

since the essential flips must occur and each non-essential flip can either occur or not occur. Finally, the exact transition probability from M_i to M_i' is given by

$$p(M_i \to M_i') = \frac{q(M_i \to M_i')}{\sum_{i'' \neq i} q(M_i \to M_i'')}. \qquad (4)$$

Using these transition probabilities between any pair of Boolean logic functions in the restricted set $\mathcal{M}_o$, the only modification to the inference algorithm for arbitrary Boolean logic functions in Section III-A is Step 2, which becomes

$2_o)$ *Sample* $\mathbf{m}_i$: We use an MH step to sample from $\mathbf{m}_i$. Specifically, for question i, we propose the parameter of a new logic function $\mathbf{m}_i'$ according to (4). Then, we evaluate the likelihood of both the new logic function and the old logic function as $p' = \prod_{j=1}^N p(Y_{i,j} | C_{k,j}, \mu_{i,k}, \mathbf{m}_i')$ and $p = \prod_{j=1}^N p(Y_{i,j} | C_{k,j}, \mu_{i,k}, \mathbf{m}_i)$. Then, we accept this new proposal with probability

$$\min\{1, r\}, \quad \text{where} \quad r = \frac{p'}{p} \frac{p(M_i' \to M_i)}{p(M_i \to M_i')},$$

since the proposal distribution is not symmetric.

This inference algorithm is only practical for a small number of concepts, i.e., $K \leq 4$; in this case, a (non-trivial) calculation shows that the new MH step is approximately three times more efficient than the regular MH step in terms of proposing to explore new ordered logic functions. For larger values of K, since we cannot enumerate every ordered Boolean logic function, we must resort to the regular MH approach that keeps proposing and rejecting new functions until an ordered one is proposed.

IV. EXPERIMENTS

We now experimentally validate our proposed BLAh model and inference algorithm using three real-world educational datasets. We first compare its performance on predicting unobserved student responses to state-of-the-art baseline algorithms, and then showcase the power of BLAh in generating interpretable model parameters by investigating the learned Boolean logic functions for a number of test questions.

A. Predicting unobserved student responses

We first compare the prediction performance of BLAh to several existing models and algorithms: the DB model [**?**], the G-DINA model [**?**], the 3PL MIRT model [**?**], and the SPARFA-Lite model [**?**], a simplification of the SPARFA model that exhibits better computational efficiency. The first two models are non-linear; the second two are linear (see Section I-C for details).

a) Datasets: We consider the following three real-world educational datasets in our experiments:

- **MT**: $N = 99$ students answering $Q = 34$ questions in a high-school algebra test administered on Amazon's Mechanical Turk [**?**]; 100% of the responses are observed. Additional tagging information is available for this dataset, i.e., a domain expert defined $K = 12$ concepts for the entire dataset and associated each question with between one and four concepts. This expert tagging information will be used to interpret the learned Boolean logic functions in Section IV-B.

- **CE**: $N = 92$ students answering $Q = 203$ questions in a semester-long undergraduate engineering course on introduction to computer engineering; 99.5% of the responses are observed. The majority of the questions in this dataset belong to the midterm and final exams.

- **SS**: $N = 41$ students answering $Q = 143$ questions in a semester-long undergraduate engineering course on signals and systems; 97.1% of the responses are observed. The

TABLE IV
PERFORMANCE COMPARISON OF PREDICTING UNOBSERVED STUDENT RESPONSES IN TERMS OF THE PREDICTION ACCURACY (ACC) FOR THE BLAH MODEL AGAINST THE DB, G-DINA, 3PL MIRT, AND SPARFA-LITE MODELS.

Model	BLAh		DB		G-DINA		3PL MIRT		SPARFA-Lite
K	3	6	3	6	3	6	3	6	
MT	**0.806±0.013**	0.803±0.015	0.801±0.013	0.799±0.012	0.770±0.012	0.775±0.017	0.673±0.024	0.723±0.020	0.802±0.016
CE	0.870±0.004	0.873±0.004	**0.875±0.004**	0.871±0.004	0.850±0.005	0.800±0.006	0.757±0.017	0.754±0.015	0.874±0.004
SS	0.869±0.009	**0.874±0.008**	0.869±0.009	0.868±0.008	0.834±0.013	0.800±0.020	0.757±0.016	0.732±0.021	0.873±0.009

TABLE V
PERFORMANCE COMPARISON OF PREDICTING UNOBSERVED STUDENT RESPONSES IN TERMS OF THE AREA UNDER THE RECEIVER OPERATING CHARACTERISTIC CURVE (AUC) FOR THE BLAH MODEL AGAINST THE DB, G-DINA, 3PL MIRT, AND SPARFA-LITE MODELS.

Model	BLAh		DB		G-DINA		3PL MIRT		SPARFA-Lite
K	3	6	3	6	3	6	3	6	
MT	**0.852±0.015**	0.849±0.015	0.840±0.018	0.839±0.018	0.784±0.023	0.730±0.021	0.646±0.027	0.690±0.024	0.838±0.019
CE	0.829±0.009	0.830±0.009	**0.831±0.005**	0.830±0.004	0.760±0.014	0.788±0.011	0.613±0.019	0.633±0.015	0.830±0.007
SS	0.815±0.021	0.822±0.020	0.810±0.019	0.812±0.018	0.678±0.026	0.648±0.030	0.692±0.025	0.672±0.028	**0.825±0.020**

majority of the questions in this dataset belong to the midterm and final exams.

b) Experimental setup: We test the prediction performance of the algorithms using 5-fold cross validation: in each run, we randomly partition the observed graded student responses $Y_{i,j}$ into 5 equal-sized folds. Four folds of the data are used for training (and validation for the algorithms that have tuning parameters), while the other held-out fold is treated as unobserved student responses and used to test the performance of the algorithms. We repeat our experiments on 20 different random partitions of the dataset, and report the average performance and standard deviation of every algorithm.

For the BLAh model, we resort to the arbitrary set of Boolean logic functions and use the algorithm detailed in Section III-A, since (i) we observed that the restricted set does not provide any improvement in prediction performance over the arbitrary set, and (ii) enumerating every Boolean logic function in the restricted set $\mathcal{M}_o$ is computationally prohibitive for $K > 4$. Furthermore, there is no need to consider the label switching problem that is common in Bayesian factor models [?], since interpretability is not essential to prediction performance. The final predicted success probability of each unobserved student response is obtained by averaging the predictive probability $p_{i,j}$, $(i, j) \in \Omega_{\text{obs}}^c$ as in (2) calculated in Step 1 of the MCMC algorithm from Section III over a number of iterations after a certain burn-in period, while the predicted response is the corresponding maximum likelihood estimate (1 if the average predicted success probability > 0.5). The hyperparameters are set as $\mu_c = \mu_\mu = 0$, $\sigma_c^2 = \sigma_\mu^2 = 5$, $p_F = \frac{1}{2^K}$ since that we have found that the performance of BLAh is robust to the values of these hyperparameters in our experiments. We run the MCMC inference algorithm for a total of $T = 20000$ iterations; we use the first 10000 iterations as burn-in to ensure that the Markov chain is well-mixed, although we observed good mixing after as few as 1000 iterations and no more than 5000 iterations in our experiments.

For the DB model, we use only the soft DB model due to its computational efficiency. For the DINA model, we implemented its EM inference algorithm as detailed in [?].[3] For the 3PL MIRT model, we estimate the model parameters using the MIRT package in R [?]. For every model except SPARFA-Lite (since SPARFA-Lite does not require the number of concepts K as an input), we use multiple values $K \in \{3, 6\}$. For simplicity of exposition, we do not show results for other values of K, because we found that the performance of BLAh is robust for a large range of values (for any $K \leq 10$).

c) Evaluation metrics: We compare the prediction performance of the algorithms on the unobserved student responses using two standard metrics: (i) the prediction accuracy (ACC) metric, i.e., the portion of correct predictions, and (ii) the area under curve (AUC) metric, i.e., the area under the receiver operating characteristic (ROC) curve of the resulting binary classifier [?]. The range of both metrics is $[0, 1]$, with larger values corresponding to better prediction performance.

d) Results and discussion: Tables IV and V show the performance of all the algorithms on predicting unobserved student responses on all datasets using the ACC and AUC metrics. The results show that the BLAh model achieves superior performance on one dataset (MT) and comparable performance on the other datasets (CE, SS) compared with the other models. We note that the performance of the other algorithms are all in the same range except for the basic linear model 3PL MIRT, which is subpar.

Table VI compares the prediction performance of multiple versions of BLAh. O-BLAh denotes the BLAh model with ordered Boolean logic functions using $K = 3$ latent concepts (the same as BLAh in Table VI), and O-BLAh-Tag denotes the BLAh model with ordered Boolean logic functions with expert tagging (for each question, we only learn an ordered Boolean logic function on the tagged concepts). The performance of O-BLAh is similar to that of BLAh; the performance of O-BLAh-Tag is significantly worse than that of BLAh and O-BLAh. This

---

[3]We also used the R package CDM [?], which does not perform as well as our own implementation.

TABLE VI
PERFORMANCE COMPARISON OF PREDICTING UNOBSERVED STUDENT
RESPONSES FOR MULTIPLE VERSIONS OF THE BLAh MODEL USING THE
MT DATASET.

	BLAh	O-BLAh	O-BLAh-Tag
ACC	0.806±0.013	0.800±0.014	0.777±0.016
AUC	0.852±0.015	0.850±0.015	0.813±0.017

fact is not surprising, as it has been shown that expert tagging often performs much worse than blind collaborative filtering [?]. Therefore, we suggest BLAh with arbitrary Boolean logic functions when prediction performance is paramount and BLAh with ordered Boolean logic functions when expert tags are available and interpretability is paramount. Next we dig deeper into interpretability.

B. Interpreting logic functions using the restricted set

Although we have shown that the BLAh model achieves comparable or in some cases, slightly better prediction performance than the existing algorithms, the learned logic functions are difficult to interpret due to the identifiability issue discussed in Section II-B. Therefore, in this experiment, we focus on the restricted set of ordered Boolean logic functions to see what insights they can provide.

TABLE VII
AN EXAMPLE QUESTION AND ITS OPTIONS IN THE MT DATASET.

Option	If $\frac{3x}{7} - \frac{9}{8} = -5$, then $x =$?
A	$14\frac{7}{24}$
B	$9\frac{7}{24}$
C	$9\frac{1}{24}$
D	$14\frac{1}{24}$

TABLE VIII
LEARNED BOOLEAN LOGIC FUNCTION FOR THE QUESTION IN TABLE VII.

Solve equations	Fractions	$\widehat{M}_i(\cdot)$
0	0	0.00
0	1	0.99
1	0	0.00
1	1	1.00

a) Experimental setup: We use only the MT dataset in this experiment, since its questions have been manually labeled by a domain expert. Our goal is to examine the non-compensatory response–concept associations that BLAh with ordered Boolean logic functions can learn given question tags provided by domain experts. For each question, we learn a Boolean logic function only on the tagged concepts, i.e., concepts that the expert identifies as tested by the question. We set the proposal truth-table bit flipping probability parameter to $p_F = 1/K_i$ for each question, where K_i denotes the number of tags on question i. The rest of the experimental setup are the same as those detailed above in Section IV-A. We investigate the learned logic function $\widehat{M}_i(\cdot)$ given by the posterior mean of $\mathbf{m}_i$.

TABLE IX
AN EXAMPLE QUESTION AND ITS OPTIONS IN THE MT DATASET.

Option	Which step is the first *incorrect* step in the following procedure to solve $-5(2x - 1) = -3x + 6$?
A	$-10x + 5 = -3x + 6$
B	$-7x + 5 = 6$
C	$-7x = 11$
D	$x = -\frac{11}{7}$

TABLE X
LEARNED BOOLEAN LOGIC FUNCTION FOR THE QUESTION IN TABLE IX.

Simplify expressions	Solve equations	$\widehat{M}_i(\cdot)$
0	0	0.00
0	1	0.21
1	0	0.97
1	1	1.00

b) Results and discussion: We select a few questions with interesting insights offered by the learned logic function and detail them below.

First, consider the question in Table VII. The posterior mean of the learned logic function is shown in Table VIII. The names of the columns correspond to the tags provided by the domain experts for this question. The learned Boolean logic function states that, in order to answer this question correctly, a student only has to master the concept "Fractions," but not necessarily the concept "Solving equations". Upon further investigation, it is clear that the focus of this question is indeed on the concept of "Fractions," since solving the equation is a trivial task if the fractions are handled properly. This observation is validated by the learned question intrinsic difficulties for these concepts: $\widehat{\mu}_{i,k} = 1.23$ for "Fractions" and $\widehat{\mu}_{i,k} = -0.97$ for "Solving equations."

Second, consider the question shown in Table IX. The posterior mean of the learned logic function is shown in Table X. We see that the learned logic function states that, in order to answer this question correctly, a student only has to master the concept "Simplifying expressions." This is indeed the case for this question, since the steps up to the first incorrect step (which is Step 3) are all about simplifying expressions; the other tested concept of the question, "Solve equations," is only required in the last step. Therefore, students' knowledge of "Solving equations" would not be tested if they master the concept "Simplifying expressions" and identify that the incorrect step is Step 3. This example shows that BLAh can be used to provide valuable feedback to instructors and content authors towards improving the quality of educational content—in this case, suggesting the design of better distractor options.

Third, consider the question shown in Table XI. The posterior mean of the learned logic function is shown in Table XII. We see that the learned logic function states that, in order to answer this question correctly, a student must have mastered both the concept of "Quadratic form" and the concept of "Inequalities." Indeed, the path to the correct answer to this question requires students to first simplify a polynomial in

TABLE XI
AN EXAMPLE QUESTION AND ITS OPTIONS IN THE MT DATASET.

Option	The inequality $6x^2 - 12x > 18$ is equivalent to
A	$x < -1$ or $x > 3$
B	$-1 < x < 3$
C	$-3 < x < 1$
D	$x > 1$ or $x < -3$

TABLE XII
LEARNED BOOLEAN LOGIC FUNCTION FOR THE QUESTION IN TABLE XI.

Solve equations	Quadratic form	$\widehat{M}_i(\cdot)$
0	0	0.00
0	1	0.02
1	0	0.05
1	1	1.00

quadratic form and then identify the correct interval using the concept of inequalities.

These three examples from the algebra test dataset demonstrate that BLAh is able to learn more complicated response–concept associations than linear models, thus making it more suitable for questions involving the interaction among multiple tested concepts. The insights BLAh can extract from graded student responses can provide useful feedback to instructors and content designers to improve the quality of questions.

V. CONCLUSIONS

We have proposed a new, non-linear student–response model, the BLAh model, that characterizes a student's response to a question as the output of a Boolean logic function. BLAh improves over traditional linear models, since it can learn complicated associations between questions and the underlying concepts they assess. Experimental results on real-world educational datasets have shown that BLAh with arbitrary Boolean logic functions achieves good prediction performance. Furthermore, resorting to a restricted set of ordered Boolean logic functions, BLAh shows the potential to achieve great interpretability by identifying the role each tested concept plays in the success on a question in situations where prediction is not the main objective.

Possible avenues of future work include (i) incorporating the Indian buffet process [?] to estimate the number K of concepts from data as a "BLAh Buffet," (ii) studying other restricted sets of Boolean logic functions, e.g., the set of logic functions that are invariant under permutations, which would eliminate the label-switching problem [?] common in factor analysis, (iii) employing sparsity constraints, since each question should only assess a small number of concepts out of all concepts in a domain (as noted in [?, Assumption 2] and [?], [?]), and (iv) exploring advanced MCMC techniques to speed up the convergence of the Markov chain or more computationally efficient convex optimization-based techniques (e.g., that build upon [?]), which may reduce the computational complexity of the BLAh inference algorithm.