Better? Automatic Discovery of Prerequisites of Skills from Student Performance Data

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

ABSTRACT

Knowing the prerequisite structure of skills is crucial for designing curriculum, assessing mastery and for student modeling. These prerequisite structures are often hand-engineered by subject matter expert in a costly and time-consuming process. In this paper, we introduce a novel data-driven pipeline for inferring the prerequisite structure of skill from student performance on test items. By modeling the prerequisite relations as a Bayesian network, the pipeline estimates the causal structure and the probabilistic dependence among the skills via a two-stage learning process. In the first stage, the Structural Expectation Maximization (Structural EM) algorithm is used to select a class of Bayesian networks based on distribution fitting of student data; in the second stage, a single Bayesian network structure is determined by enforcing a constraint on the estimated conditional probability tables. We validate the proposed pipeline using simulations and by post-hoc analysis of student data. We show the discovered prerequisite structures can improve the student model in predicting student performance.

Keywords

ACM proceedings, LaTeX, text tagging

1. INTRODUCTION

Students learn much better when the skills are not randomly introduced but organized in a meaningful order which starts from relatively simple concepts and gradually introduces more complex ones. Further, among these skills, some are preliminary of others such that they must be mastered before the subsequent concepts can be learned. For instance, students have to know how to do addition before they learn to do multiplication. In this work, we use prerequisite structure to refer to the relationships among skills that place strict constraints on the order in which these skills can be acquired. Determining the prerequisite relations among skills is crucial for designing curriculum and for assessing mastery.

Most prerequisite structures of skills are hand-engineered by subject matter experts in a costly and time-consuming process. Further, the prerequisite structures specified by the experts are seldom tested and might be unreliable in the sense that experts may hold "blind spots". Nowadays, large volume of educational data has been cumulated through the online tutoring systems. Thus, learning prerequisite structures from educational data has drawn substantial interest from both education and data mining communities [5, 20, 2, 16, 3, 15]. However, inferring the prerequisite relationships between skills is still challenging since a student's knowledge of a skill is a latent variable in the data. In this paper, we introduce BNPD, a novel pipeline for discovering prerequisite structure of skills from data. BNPD models the prerequisite structure of the skills with a statistical model called Bayesian network. It then learns the Bayesian network through a two-stage process.

2. PREVIOUS WORK

Many works have investigated the discovery of prerequisite structures within domain models from data. Some of these works aim to learn the prerequisite structure of observed variables, e.g., the relationships among exercise problems. These include the Partial Order Knowledge Structures (POKS) algorithm [5, 13] to learn the item-item relationships and the approach to determine the dependency structure of units in a curriculum with the student performance data observed at the unit level [21].

To learn the relationships among skills from data where a student's knowledge of a skill is latent, Brunskill proposed a method of computing and comparing the log likelihoods between the prerequisite model and the flat model (skills are independent) on each skill pair to determine which model better fits the data [2]. Chen et al. [3] then used probabilistic association rules mining to determine the prerequisite relationships between each pair of skills. Both methods focus on estimating the pairwise prerequisite relationships, instead of optimizing the full structure of the skills.

Bayesian networks have been extensively used to model skill topologies [10]. By modeling prerequisite relations as a Bayesian network, Scheines et al. used a Bayesian network structural learning algorithm to find the Bayesian network structure that best represents the distribution of the data [16]. Bayesian network allows encoding conditional independence relationships between skills, which enables more accurate modeling of the probabilistic dependence between skills. However, output from their algorithm is a class of Bayesian networks. These Bayesian networks represent different prerequisite structures.

In this paper, we propose a novel pipeline to infer the prerequisite structures of skills. Our approach also uses Bayesian network to

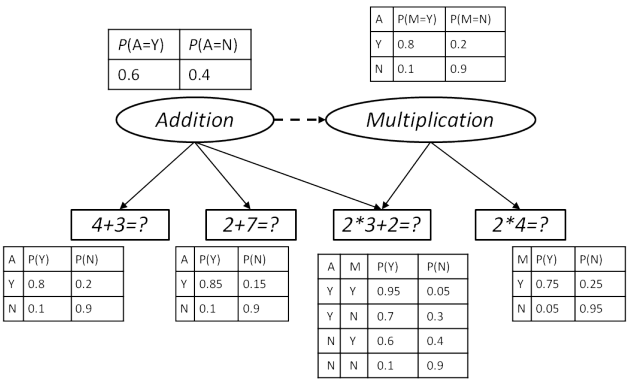


Figure 1: A hypothetical Bayesian network learned with Algorithm **??**. Solid edges are given by the item to skill mapping, dashed edges between skill variables are to be discovered from data. The conditional probability tables are to be learned.

model the prerequisite relationships. The pipeline first use a popular Bayesian network structure learning algorithm, called Structural EM to select a class of Bayesian network models based on the distribution fitting to the data. It then uses constraint on the conditional probability tables to select one Bayesian network, which gives a unique prerequisite structure of skills. Finally, we show the discovered prerequisite structure can be used to improve student modeling.

3. THE PREREQUISITE STRUCTURE DISCOVERY PIPELINE

BNPD learns the prerequisite structure of the skills using data with a statistical model called Bayesian network [14, 17]. Bayesian networks are also called probabilistic graphical models because they can be represented visually and algebraically as a collection of nodes and edges. A tutorial description of Bayesian networks in education can be found elsewhere [1], but for now we say that they are often described with two components: the nodes represent the random variables, which we describe using *conditional probability tables* (CPTs), and the set of edges that form a *directed acyclic graph* (DAG) represent the conditional dependencies between the variables. Bayesian networks are a flexible tool that can be used to model an entire curriculum.

Figure 1 illustrates an example of a prerequisite structure modeled with a Bayesian network. Here, we relate four test items with the skills of addition and multiplication. Addition is a prerequisite of multiplication thus there is an arrow from addition to multiplication. Modeling prerequisites as edges in a Bayesian network allows us to frame the discovery of the prerequisite relationships as the well-studied machine learning problem of learning a DAG (with the presence of latent variables).

Algorithm 1 describes the BNPD pipeline. Suppose we collect data from n students, answering p items. Then, the input of BNPD is a matrix $\mathbf{D}$ with $n \times p$ dimensions, an item to skill mapping.Each entry in $\mathbf{D}$ encodes the performance of a student (see Table 1 for an example).

BNPD relies on a popular machine learning algorithm called Structural Expectation Maximization (EM), which has not been used in educational applications. A secondary contribution of our work is introducing Structural EM for learning Bayesian network structures from educational data. We now describe the steps of BNPD in detail.

Table 1: Example data matrix to use with BNPD. The performance of a student is encoded with 1 if the student answered correctly the item, and 0 otherwise.

User	Item 1	Item 2	Item 3	Item p
Alice	0	1		0
Bob	1	1	...	1
Carol	0	0		1
		...		

Algorithm 1 The BNPD algorithm

Require: A matrix $\mathbf{D}$ of student performance on a set of test items, skill-to-item mapping Q (containing a set of skills $\mathbf{S}$).

1: $G_0 \leftarrow \text{Initialize}(\mathbf{S}, Q)$ } Initialization
2: $i \leftarrow 0$ } Initialization
3: **do** } Structural EM
4: E-step: } Structural EM
5: $\theta_i^* \leftarrow \text{ParametricEM}(G_i, \mathbf{D})$ } Structural EM
6: $\mathbf{D}_i^* \leftarrow \text{Inference}(G_i, \theta_i^*, \mathbf{D})$ } Structural EM
7: M-step: } Structural EM
8: $\langle G_{i+1}, \theta_{i+1} \rangle \leftarrow \text{BNLearning}(G_i, \mathbf{D}_i^*)$ } Structural EM
9: $i \leftarrow i+1$ } Structural EM
10: **while** Stop criteria is not met } Structural EM
11: $RE \leftarrow FindReversibleEdges(G_i)$ } Orient edges
12: $EC \leftarrow EnumEquivalentDAGs(G_i)$ } Orient edges
13: $DE \leftarrow \{\}$ } Orient edges
14: **for** every reversible edge $S_i - S_j$ in RE **do** } Orient edges
15: $ratio \leftarrow \frac{P(S_i=1|S_j=1)\cdot P(S_j=0|S_i=0)}{P(S_j=1|S_i=1)\cdot P(S_i=0|S_j=0)}$ [1] } Orient edges
16: **if** $ratio > 1$ **then** } Orient edges
17: $ratio^* = ratio$ } Orient edges
18: $DE \leftarrow DE \cup S_i \rightarrow S_j$ } Orient edges
19: **else** } Orient edges
20: $ratio^* = \frac{1}{ratio}$ } Orient edges
21: $DE \leftarrow DE \cup S_i \leftarrow S_j$ } Orient edges
22: **end if** } Orient edges
23: **end for** } Orient edges
24: $sort(DE)$ by $ratio^*$ in descending order } Orient edges
25: **while** DE is not empty **do** } Orient edges
26: $e \leftarrow dequeue(DE)$ } Orient edges
27: $\forall G \in EC$, remove G from EC if $e \notin G$ } Orient edges
28: **end while** } Orient edges
29: return EC } Orient edges

3.1 Initial Bayesian Network

BNPD represents the prerequisite structure using Bayesian networks that use latent variables to represent the student knowledge of a skill, and observed variables that represent the student performance answering items (e.g, correct or incorrect). We first create an initial Bayesian network that complies to the skill-to-item Q-matrix on the prerequisite structure That is, we create an arc to each item from each of its required skills and leave all skill variables disconnected. With the created Bayesian network as an initial network, we learn the arcs between the skill variables using Structural EM.

[1] $P(S_i = a|S_j = b)$ can be computed using any Bayesian network inference algorithm such as Junction tree algorithm[12]

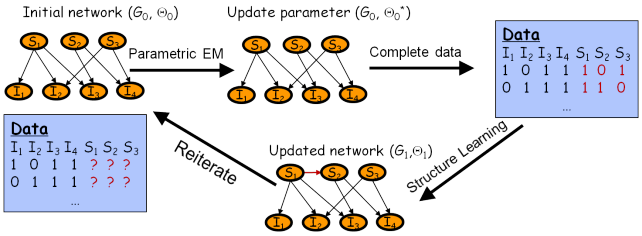


Figure 2: An illustration of the Structure EM algorithm to discover the structure of the latent variables.

3.2 Structural EM

A common solution to learning a Bayesian network from data is the score-and-search approach [4, 9]. This approach uses a scoring function to measure the fitness of a Bayesian network structure to the observed data, and manages to find the optimal model in the space of all possible Bayesian network structures. However, the conventional score-and-search approaches rely on efficient computation of the scoring function, which is only feasible for problems where data contains observations for all variables in the Bayesian network. Unfortunately, our domain has skill variables that are not directly observed. An intuitive work-around is to use the Expectation Maximization (EM) to estimate the scoring function. However, EM in this case takes a large number (hundreds) of iterations to converge and each iteration requires Bayesian network inference, which is computationally prohibitive. Further, we need run EM for each candidate structure. The number of possible Bayesian network structures is super-exponential with respect to the number of nodes. The Structural Expectation Maximization algorithm [6, 7] is an efficient alternative.

Structural EM is an iterative algorithm that inputs a matrix **D** of student performance (see example Table 1). Figure 2 illustrates one iteration of the Structural EM algorithm. The relevant steps are also sketched in Algorithm 1. Each iteration consists of an Expectation step (*E-step*) and a Maximization step (*M-step*). In *E-step*, we first find the maximum likelihood estimate θ^* of the parameters for the current structure G calculated from previous iteration using parametric EM.[2] We then do Bayesian inference to compute the expected values for the hidden variables using the current model (G, θ^*) and use the values to complete the data. In the *M-step*, we use the conventional score-and-search approach to optimize the structure according to the completed data. Since the space of possible Bayesian network structures is super-exponential, exhaustive search is intractable and local search algorithms, such as greedy hill-climbing search, are often used. The *E-step* and *M-step* interleave and iterate until some stop criteria is met, e.g., the scoring function does not change significantly. Contrast to the conventional score-and-search algorithm, Structural EM runs EM only on one structure in each iteration, thus is computationally more efficient.

BNPD's initialization step fixes the arcs from skills to items according to the Q-matrix. In the *M-step* of our Structural EM implementation we only consider the candidate structures that comply with the Q-matrix.

An advantage of using Structural EM to discover the prerequisite relationship of skills, is that it is easily extensible to incorporate domain knowledge. For example, we can place constraints on the output structure to force or to disallow a skill to be a prerequisite from another other skill. Consider, an intelligent tutor that teaches the content of a book. The book content structure provides a natural ordering of the chapters. We may use the book structure to engineer domain knowledge that an introductory chapter cannot be prerequisite of content that appears later in the book.

Another advantage of Structural EM is that it can be applied when there are missing data in the student performance matrix **D**. That is, some students do not answer all the items thus some responses are missing. This "missing values" problem is very common in real-world situations. Structural EM can be applied to this type of data since it was originally developed to solve the "missing values" problem [6]. The general idea is, in the *E-step*, the algorithm also computes the expected values for missing data points, in addition for hidden variables, .

3.3 Discriminate Between Equivalent BNs

Structural EM selects Bayesian network model based on how well it explains the distribution of the data. Bayesian network theory states that some Bayesian networks are statistically equivalent in representing the data. Thus, the output from Structure EM is an equivalence class that may contain many Bayesian network structures. These equivalent Bayesian networks have the same skeleton and the same v-structures[3]. For instance, Figure 3 gives an example of two simple Bayesian networks that are not distinguishable by Structural EM algorithm. They share the skeleton but differ in the orientation of the edge. They apparently represent two different prerequisite structures. To determine which structure is the case, we shall determine the orientation of each reversible edge. For this purpose, we can use the following domain knowledge.

Knowledge 1. If S_1 is a prerequisite of S_2, i.e., $S_1 \rightarrow S_2$, then $P(S_1 = 1|S_2 = 1) = 1$ and $P(S_2 = 0|S_1 = 0) = 1$.

Knowledge 1 says if a skill S_1 is the prerequisite of a skill S_2, a student must master skill S_1 before he masters S_2 and it is impossible for him to master S_2 if he has not mastered S_1. In other words, S_1 is not a prerequisite of S_2 if at least one of the conditions does not hold. This puts a constraint on the joint distribution encoded by the Bayesian network to be learned. We can check for each reversible edge $S_i - S_j$ in the Bayesian network, either $P(S_i = 1|S_j = 1) = 1$ and $P(S_j = 0|S_i = 0) = 1$, or $P(S_j = 1|S_i = 1) = 1$ and $P(S_j = 0|S_i = 0) = 1$. If the former holds, $S_i \rightarrow S_j$; otherwise, $S_i \leftarrow S_j$. However, in real situations, it is possible for a student to master the post-requisite even he does not know the prerequisite. Further, there is always noise in the data. Thus, the two conditional probabilities $P(S_1 = 1|S_2 = 1)$ and $P(S_2 = 0|S_1 = 0)$ are not exactly but close to 1. Thus, we use the following empirical rule: if $P(S_1 = 1|S_2 = 1) \cdot P(S_2 = 0|S_1 = 0) > P(S_2 = 1|S_1 = 1) \cdot P(S_1 = 0|S_2 = 0)$, we determine $S_1 \rightarrow S_2$; otherwise, we determine $S_1 \leftarrow S_2$. The intuition behind this is that the two probabilities $P(S_1 = 1|S_2 = 1)$ and $P(S_2 = 0|S_1 = 0)$ are certificates of the prerequisite structure $S_1 \rightarrow S_2$. The larger of these two probabilities, the more likely the relationship $S_1 \rightarrow S_2$ holds. Since here we are concerned with which direction the edge goes, we simply compare the products of two probabilities and select the direction that is more probable.

Using the empirical rule, we can orient every reversible edge in the

[2] In the first iteration, the current network is created from the initialization step.

[3] A v-structure in a Bayesian network G is an ordered triple of nodes (u, v, w) such that G contains the directed edges $u \rightarrow v$ and $w \rightarrow v$ and u and w are not adjacent in G. [19].

(a) (b)

Figure 3: Two equivalent Bayesian networks representing different prerequisite structures.

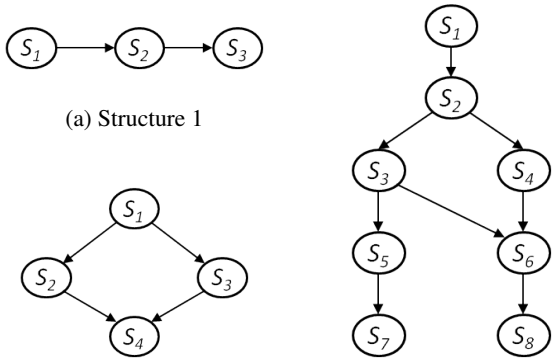
network structure. However, the determinations of all edges' orientations are not independent and may conflict each other. Having oriented one edge would constrain the orientation of other reversible edges because we have to ensure the graph is a DAG and in the same equivalence class. Thus, for each reversible edge $S_i - S_j$, we compute $ratio = \frac{P(S_i=1|S_j=1)\cdot P(S_j=0|S_i=0)}{P(S_j=1|S_i=1)\cdot P(S_i=0|S_j=0)}$, then $ratio^* = ratio$ if $ratio > 1$ and $ratio^* = \frac{1}{ratio}$ otherwise. The larger the $ratio^*$ is, the more confidently when we decide the orientation. We sort the list of reversible edges by $ratio^*$ in descending order. We then orient the edges in this order. When one edge is oriented, the constraint is propagated to other reversible edges. In practical implementation, we take a different strategy: we first enumerate all equivalent Bayesian networks and make them a list of candidates; when an edge is oriented to $S_i \rightarrow S_j$, we remove all contradicting Bayesian networks from the list. Eventually only one Bayesian network structure stands. This procedure is also detailed in the *Orient edges* section of Algorithm 1

4. EVALUATION

In § 4.1, we evaluate BNPD with simulated data to assess the quality of the discovered prerequisite structures. Then, in § 4.2 we use data collected from real students.

4.1 Simulated Data

Synthetic data allows us to study how BNPD compares to ground truth. For this, we engineered three prerequisite structures (DAGs), shown in Figure 4. Here, each figure represents different causal relations between the simulated latent skill variables.



(a) Structure 1

(b) Structure 2 (c) Structure 3

Figure 4: Three different DAGs between latent skill variables. Item nodes are omitted.

For clarity, Figure 4 omits the item nodes; but each skill node is parent of six item variables and each item variable has 1-3 skill nodes as parents. All of these nodes are modeled using binary random variables. More precisely, the latent nodes represent whether the student achieves mastery of the skill, and the observed nodes indicate if the student answers the item correctly. Notice that these Bayesian networks include the prerequisite structures as well as the skill-item mapping.

We consider simulated data with different number of observations ($n = 150, 500, 1000, 2000$). For each sample size and each DAG, we generate ten different sets of conditional probability tables randomly, with two constraints. First, we enforce that achieving mastery of the prerequisites of a skill will increase the likelihood of mastering the skill. Second, mastery of a skill increases the probability of student correctly answering the test item. Thus, in total we generated 120 synthetic datasets (3 DAGs x 4 sample sizes x 10 CPTs), and report the average results.

We evaluate how BNPD can discover the true prerequisite structure using metrics designed to evaluate Bayesian networks structure discovery. In particular, we use the F_1 *adjacency score* and the F_1 *orientation score*. The adjacency score measure how well we can recover connections between nodes. It is a weighted average of the true positive adjacency rate and the true discovery adjacency rate. On the other hand, the orientation score measures how well we can recover the direction of the edges. It is calculated as a weighted average of the true positive orientation rate and true discovery orientation rate. In both cases, the F_1 score reaches its best value at 1 and worst at 0. Moreover, for comparison, we compute the F_1 *adjacency score* for Bayesian network structures whose skill nodes are fully connected with each other. These fully connected DAGs will serve as baselines for evaluating the adjacency discovery.[4] For completeness, we list these formulas in tables 2 and 3, respectively.

Table 2: Formulas for measuring adjacency rate (AR)

Metric	Formula
True positive (*TPAR*)	$\frac{\text{\# of correct adjacencies in learned model}}{\text{\# of adjacencies in true model}}$
True discovery (*TDAR*)	$\frac{\text{\# of correct adjacencies in learned model}}{\text{\# of adjacencies in learned model}}$
F_1-*AR*	$\frac{2 \cdot TPAR \cdot TDAR}{TPAR + TDAR}$

Table 3: Formulas for measuring orientation rate (OR)

Metric	Formula
True positive (*TPOR*)	$\frac{\text{\# of correctly directed edges in learned model}}{\text{\# of directed edges in true model}}$
True discovery (*TDOR*)	$\frac{\text{\# of correctly directed edges in learned model}}{\text{\# of directed edges in learned model}}$
F_1-*OR*	$\frac{2 \cdot TPOR \cdot TDOR}{TPOR + TDOR}$

We use these metrics to evaluate the effect of varying the number of observations of the training set (sample size) on the quality of learning the prerequisite structure. We designed experiments to specifically answer the following four questions:

1. How does the type of items affect BNPD's ability to recover the prerequisite structure? We consider the case where in the model each item requires only one skill and the case where item require multiple skills.
2. How well does BNPD perform when there is noise in the data? We focus on studying noise due to the presence of unaccounted hidden variables.
3. How well does BNPD perform when the student performance data has missing values?

[4]We do not compute F_1 orientation score for fully connected DAGs because all edges in a fully connected DAG are reversible.

4. How is BNPD compared with other prerequisite discovery methods? Here we compare BNPD to the Probabilistic Association Rules Mining (PARM) method [3].

We now investigate these questions.

4.1.1 *Single-skill vs Multi-skill Items*

We consider two situations where different types of Q-matrix are used. In the first situation, each item node maps to only one skill node. In the second one, each item loads 1-3 skills. Figure 5 compares the F_1 of adjacency discovery and edge orientation result under two types of Q-matrix. We observe that the accuracy for both the adjacency and the edge orientation improves with the amount of data. With just 2000 observations, the algorithm can recover the true structures almost perfectly. Additionally, when the model contains multiple-skill items, the accuracy is slightly lower than that where all items in the model are single-skilled items, probably because there are more parameters to be estimated in the case of multi-skill items.

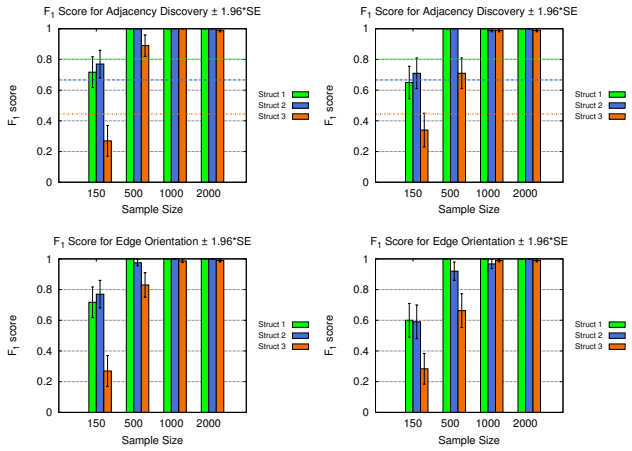


Figure 5: Comparison of F_1 scores for adjacency discovery (top row) and for edge orientation (bottom row). Horizontal lines are baseline F_1 scores computed for fully connected (complete) Bayesian networks.

4.1.2 *Sensitivity to Noise*

Real-world data sets often contain various types of noise. For example, noise may occur due to hidden variables that are not explicitly modeled. To evaluate the sensitivity of REMIND to noise, we synthesize Bayesian networks including a *StudentAbility* node that takes three possible states (low/med/high). In these Bayesian networks, students' performance depends not only on whether they have mastered the skill, but also on their individual ability. For simplicity, all items in the setting are single-skilled items. We first simulated data from Bayesian networks that have a *StudentAbility* variable to generate "noisy" data samples, and then use this data to recover the prerequisite structure. Figure 6 illustrates the procedure of this sensitivity analysis experiment.

Figure 7 compare the results were noise was introduced or not. Interestingly, the noise does not harm the learning accuracy at all, and actually improves the accuracy. We hypothesize that the existence of additional hidden variable increases the variance of the data which can help the structure learning.

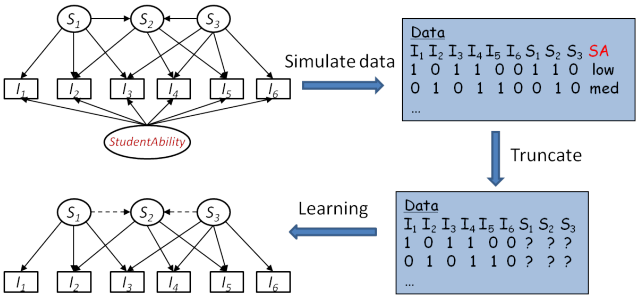


Figure 6: Evaluation of BNPD with noisy data

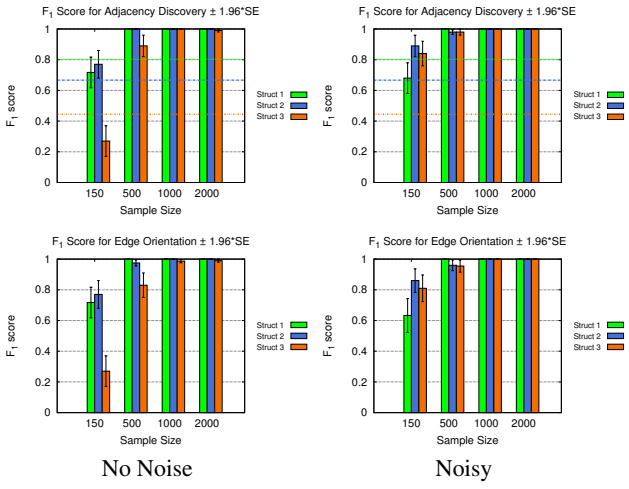


Figure 7: Results of adding systematic noise. Top: Comparison of F_1 scores for adjacency discovery. Horizontal lines are baseline F_1 scores computed for fully connected Bayesian networks. Bottom: Comparison of F_1 scores for edge orientation.

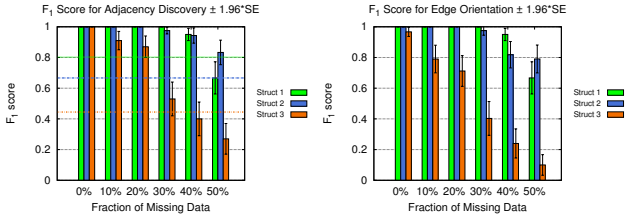


Figure 8: Results of learning with missing data. Left: Comparison of F_1 scores for adjacency discovery. Horizontal lines are baseline F_1 scores computed for fully connected Bayesian networks. Right: Comparison of F_1 scores for edge orientation.

4.1.3 *In Presence of Missing Values*

To evaluate how BNPD performs on data with missing values, we generated data sets of size 1000 with varying fraction of missing values (10%, 20%, 30%, 40%, 50%) and ran BNPD to recover the structures from these data sets. Again, the models only contains single-skilled items. The results are presented in Figure 8. It is

clearly seen that the accuracy decreases when the fraction of missing values increases. The algorithm is still able to recover the true structures of Structure 1 and 2 even the data contains up to 30% missing values.

4.1.4 Comparison With PARM

Chen and his colleagues proposed to use Probabilistic Association Rules Mining (PARM) for discovering the prerequisite relationships between skills [3]. Since PARM discovers pair-wise prerequisite relationship, instead of constructing the full structure, we only compare BNPD with PARM for discovering the pair-wise relationships. That is, we derived pair-wise prerequisite relationships from the Bayesian network structure and see how the two approaches discover these relationships. Here we simulated data from Structure 3 (Figure 4(c)) (with single-skilled items), which has 21 pair-wise prerequisite relationships. When experimenting iwth PARM, the following cutoff values were used: $minsup = 0.125$, $minconf = 0.76$, $minprob = 0.9$. These values have been used in the simulation experiments in [3]. The evaluation metric used is $F_1 = \frac{2*TPR*TDR}{TPR+TDR}$, where $TPR = \frac{\text{\# of correct relationships learned}}{\text{\# of relationships in true model}}$ and $TDR = \frac{\text{\# of correct relationships learned}}{\text{\# of relationships in learned model}}$. The result is presented in Figure 9. It shows that BNPD significantly outperforms PARM F_1 score for sample size $n \geq 500$. The lower F_1 score by PARM is caused by lower TPR, i.e., PARM failed to discover many prerequisite relationships (data not shown).

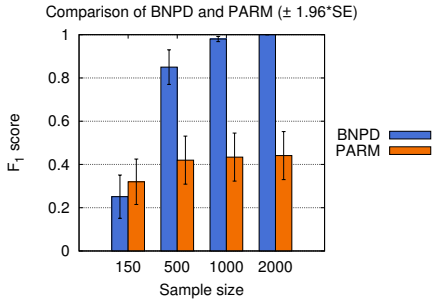


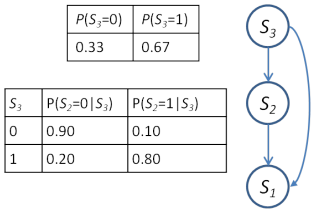
Figure 9: Comparison of BNPD and PARM for discovering prerequisite relationships in Structure 3. The cutoff values used for PARM experiments are: $minsup = 0.125$, $minconf = 0.76$, $minprob = 0.9$.

4.2 Real Student Performance Data

We now evaluate BNPD using two real-world student data sets.

4.2.1 ECPE Data Set

The ECPE (Examination for the Certification of Proficiency in English) data set was collected from a test by the English language Institute of the University of Michigan to examine individual cognitive skills required for understanding English language grammar [18]. It contains a sample of 2922 examinees who is tested by 28 items on 3 skills, i.e., *morphosyntactic rules* (S_1), *cohesive rules* (S_2) and *lexical rules* (S_3). Each item requires either one or two of the three skills. The prerequisite structure output from BNPD are depicted in Figure 10. The estimated conditional probability tables are showed correspondingly. The discovered structure says *lexical rules* is a prerequisite of *cohesive rules* and *morphosyntactic rules*, and *cohesive rules* is a prerequisite of *morphosyntactic rules*. This totally agrees with the findings in [18] and by the PARM method in [3]. Further, BNPD also outputs the conditional probabilities associated with each skill and its direct prerequisite, which provides insights into how prerequisites impact the post-requisite probabilistically.



$P(S_3=0)$	$P(S_3=1)$
0.33	0.67

S_3	$P(S_2=0 \mid S_3)$	$P(S_2=1 \mid S_3)$
0	0.90	0.10
1	0.20	0.80

S_2	S_3	$P(S_1=0 \mid S_2,S_3)$	$P(S_1=1 \mid S_2,S_3)$
0	0	0.97	0.03
1	0	0.57	0.43
0	1	0.87	0.13
1	1	0.33	0.67

Figure 10: Result on ECPE data set. S_1: Morphosyntactic rules; S_2: Cohesive rules; S_3: Lexical rules. The estimated conditional probability tables are shown correspondingly.

4.2.2 HED Data Set

We now evaluate BNPD using data collected from a commercial non-adaptive tutoring system. The data set, named as `HED`, is from anonymized students interacting with an implementation of a seventh grade math curriculum. The textbook items are classified in chapters, sections, and objectives; but for this paper, we only use the data from Chapter 2 and Chapter 3 of the book. We use performance data while students solve test items.

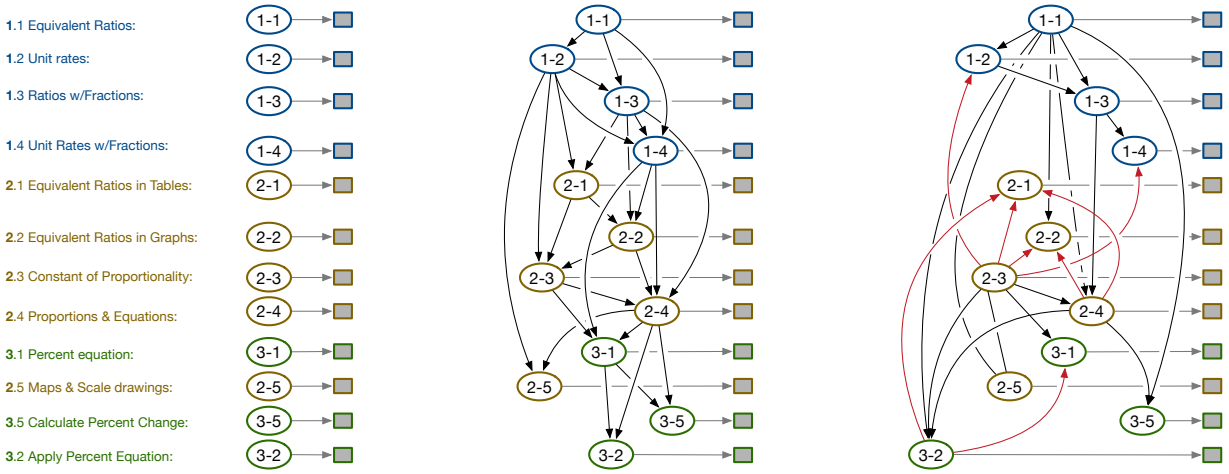
Q-matrix and preprocessing. We use an item-to-skill mapping (Q-matrix) that assigns each exercise to a skill solely as the book section in which the item appears. We process the data set to find a subset of items and students that does not have missing data. This is, the dataset we use in BNPD has students responding to *all* of the items.

After filtering, each skill (book section) has two to four items, for a total of 33 items. The resulting dataset from the `traditional` implementation includes student test results for 5202 students; while the `virtual` implementation has the test results for 467 students. Our synthetic data experiments suggest that a large number of training data improves the learning quality of the prerequisite structure. For this reason, we use the larger `traditional` dataset to build the prerequisite model.

Prerequisite Structure Discovery. We now describe how we use the REMIND algorithm to learn a remedial model. For simplicity we use binary variables to encode performance data (i.e, correct or incorrect) and skill variables (i.e., mastery or not mastery). This simplification is not necessary, as REMIND is able to use discrete variables with arbitrary number of states. We use an implementation of Structural EM available online called LibB[5]. We experiment with two conditions:

- We use domain knowledge to constrain the prerequisite structure. Since our skills correspond to the book sections, we use the table of contents as expert knowledge. We constraints skills so that a skill a can be prerequisite of a skill b, iff a appears before b in the table of contents of the textbook.
- Alternatively, we also experiment using a fully data-driven approach in which REMIND learns the prerequisite structure without any constraints.

[5] `http://compbio.cs.huji.ac.il/LibB/programs.html`



(a) Q-matrix represented as a Bayes network (b) Network constrained by domain knowledge (c) Network with no constraints

Figure 11: Prerequisite structures constructed by Structural EM. Nodes have been augmented with color information to emphasize the chapter association. Reversible edges are depicted as undirected. Squares represent the exercise items.

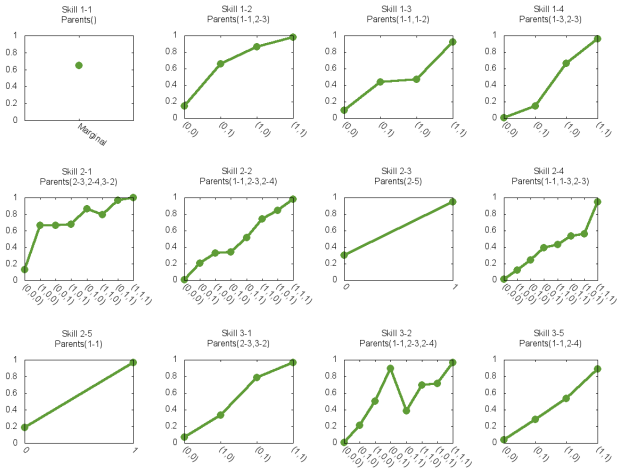


Figure 12: Visualization of the estimated conditional probability distributions for prerequisite structure model in Figure 11 (a). Each plot depicts the conditional probability of mastering a skill given the mastery status of its direct prerequisites. Parents(·) specifies the direct prerequisites of the skill. In X-axis, 0 means the corresponding prerequisite is not mastered and 1 otherwise.

Figure 11 illustrates Bayesian networks generated with the REMIND algorithm. Figure 11a represents a disconnected model generated by simply converting the input Q-matrix into a Bayesian network; this corresponds to the output of the initialization step of Algorithm **??**. Figures 11b and 11c represent the prerequisite structures discovered by using domain knowledge or not. In the fully data-driven model, we draw with red the edges that contradict our text book ordering heuristic. Our observation is that the structures learned are not random, sections (skills) in the same chapter tend to form clusters, even in the structure learned without the ordering constraint. We should mention that the BIC score[6] of the structure learned without the ordering constraint is slightly better than the BIC score of that learned with the ordering constraint, indicating that the first model fits the data better than the second one does.

Our approach also outputs the conditional probabilities associated with each skill and its direct prerequisites. Figure 12 plots the conditional distribution table for each skill. Each sub-figure plots the conditional probability of student achieving the mastery of a skill against the mastery status of the skill's direct prerequisites. More specifically, X-axis specifies all possible mastery statuses of a skill's direct prerequisite, Y-axis is the probability of student achieving the mastery of the skill given the corresponding mastery status of its direct prerequisites. We can see a trend in all plots that the probability of student mastering a skill increases monotonically when the student has acquired more prerequisites of the skill. This is consistent with our intuition that mastering a skill's prerequisites will help student master the skill. We also observed a small contradiction in the case of skill 3-2, which exemplifies the limitation of fully data-driven methods.

Quantitative evaluation. We now evaluate REMIND using data that has already been collected (*post-hoc* analysis). We evaluate how well the discovered Bayesian networks can predict student performance on a on a test item given performance on other items. In particular, we compute the posterior probability of a student's response to an item I_i given his performance on all other items $\mathbf{I}_{-i} = \mathbf{I} \setminus \{I_i\}$, by marginalizing:

$$P(I_i = 1 | \mathbf{I}_{-i} = \mathbf{i}_{-i}) = \sum_{\mathbf{S}} P(I_i, \mathbf{S} | \mathbf{I}_{-i} = \mathbf{i}_{-i}), \qquad (1)$$

This can be computed efficiently using the Junction tree algorithm [12]. We then do binary classification based on the posterior probability.

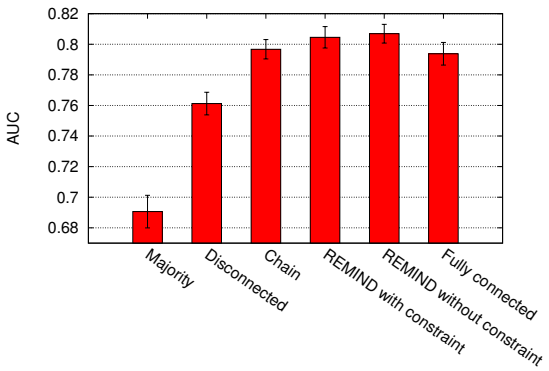
[6] In our experiment, we use the Bayesian information criterion (BIC) score to measure the fitness of a Bayesian network to the data. The BIC score for a Bayesian network model is composed of a likelihood term and a term to penalize the model complexity.

We compare REMIND with four baseline predictors:

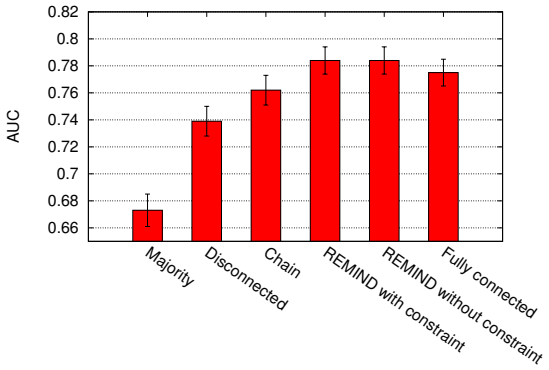
- A *majority* classifier which always classifies an instance to the majority class. For example, if majority of the students get an item wrong, other students would likely get it wrong.
- A Bayesian network model in which the skill variables are *disconnected*. This corresponds to using the Q-matrix Bayesian network of Figure 11a. This model assumes that the skill variables are marginally independent of each other.
- A Bayesian network model in which the skill variables are connected in a *chain* structure, i.e., 1-1→1-2→1-3→ ... This assumes that a section (skill) only depends on the previous section. In other words, a first-order Markov chain dependency structure.
- A *fully connected* Bayesian network where skill variables are fully connected with each other. This model assumes no conditional independence between skill variables and can encode any joint distribution over the skill variables. However, it has exponential number of free parameters and thus can easily overfit the data.

The parameters of these baseline Bayesian network predictors are estimated from the `traditional` data set. We first did cross-validation experiments to evaluate these classifiers using the `traditional` data. Figure 13a evaluates the model predictions using the *Area Under the Curve* (AUC) of the Receiver Operating Characteristic (ROC) curve metric. The error bars show the 95% confidence intervals calculated from the 10-fold cross-validation. The best performing models are REMIND with ordering constraint with an AUC of 0.804 ± 0.007 and REMIND with no ordering constraint with an AUC of 0.807 ± 0.006. These two REMIND models outperform the other four models. The *chain* model performs the best among the four baseline predictors with an AUC of 0.796 ± 0.006. A paired t-test reveals that both of the REMIND models significantly outperform the *chain* model (the p-values are 0.0064 and 0.003). However, the two REMIND prerequisite models are not statistically different from each other. We note that the *fully connected* model was outperformed by the two prerequisite models and the *chain* model, suggesting overfitting.

We now investigate the extent of which a REMIND model can be generalized to a different implementation of the curriculum. For this, we use the models constructed from students using the `traditional` curriculum to make prediction on the `virtual` curriculum. The comparison of AUCs is illustrated in Figure 13b. The error bars show the 95% confidence intervals calculated with an implementation of the Logit method[7], which corrects for the non-independence of the points of the ROC curve. Again, the two REMIND prerequisite models clearly outperform other four models. The AUCs of both REMIND models are 0.784 ± 0.010. Curiously, in this experiment the difference between REMIND and the chain baseline becomes more prominent.



(a) `traditional` AUC results. The results are from 10 fold cross-validation.



(b) `virtual` AUC results. Models are trained using `traditional` data set.

Figure 13: Comparison of AUC of six models to predict student performance on exercise items

5. RELATION TO PRIOR WORK

We believe we are the first ones to propose a pipeline of learning the prerequisite dependencies from data and use it for student modeling. Our work builds on prior work that discovers prerequisite from data [5, 20, 2, 16, 3, 15]. However, these approaches do not attempt to validate the prerequisite discovered using real student performance data. Our approach differs from prior work in several ways. The approaches discover the relationship of items without using latent variables. This is, the prerequisites do not use skill mappings, and only find dependencies between items [5, 20, 15]. For the approaches that can account for latent variables [2, 3], they focus on estimating the pairwise prerequisite relationships. By contrast, we try to optimize the full structure of the model.

A contribution of our work is introducing the Structural EM algorithm to the educational community. This approach has many advantages over prior work in that it does not require tuning of many parameters. For example, prior work [2] assumes domain parameters called *guess probability* and *slip probability* are provided for each pair of item and skill. Similarly, more recent approaches [3] require manually specified thresholds to determine the existence of a prerequisite relationship. The determination of these thresholds requires experts' intervention. By contrast, the only required input of our algorithm is the observed student performance.

A limitation of our work is that we do not address in this paper is comparing Structural EM with these approaches. Future work may address a comprehensive comparison of different prerequisite structure methods.

[7] `http://www.subcortex.net/research/code/area_under_roc_curve`

6. CONCLUSION

Although in some educational paradigms [11] students are not supposed to move to subsequent lessons until they have mastered all of the prerequisites, these is not always attainable in practice. We propose and evaluate the REMIND pipeline, a simple but effective novel algorithm that detects when a student needs remediation.

A limitation of our study is that we only evaluated REMIND using simulations and posthoc analyses. Future work may evaluate the REMIND algorithm in a randomized control trial. Further, we evaluated our models using traditional statistical metrics, but recent work suggest that tailored evaluation metrics for tutoring system may be superior [8]. Thus, another piece of future work is to evaluate REMIND using these evaluation metrics.

The main contributions of our work are: a novel data-driven algorithm that allows domain knowledge for designing remediation triggers; a novel methodology to evaluate prerequisite graphs using student data; and suggesting the Structural EM algorithm for educational applications.

The advantage of REMIND are both qualitative and quantitative. Qualitatively, REMIND allows us to understand the organization of the skills in the curriculum as it builds a prerequisite network of the skills in a Q-matrix. When used in a learning model, REMIND builds a hypothesis of how practice affects the knowledge of the student. Sometimes the practice may affect the skill directly, but sometimes it may affect a prerequisite. Future work may validate these hypothesis in a controlled experiment. Additionally, our quantitative results suggest that REMIND can be used to improved student modeling. Overall, we believe that REMIND is promising technology to detect when a student needs help.

7. REFERENCES

[1] Russell G Almond, Robert J Mislevy, Linda Steinberg, Duanli Yan, and David Williamson. 2015. *Bayesian networks in educational assessment*. Springer.

[2] Emma Brunskill. 2010. Estimating prerequisite structure from noisy data. In *Educational Data Mining 2011*.

[3] Yang Chen, Pierre-Henri Wuillemin, and Jean-Marc Labat. 2015. Discovering Prerequisite Structure of Skills through Probabilistic Association Rules Mining. In *Proceedings of the 8th International Conference on Educational Data Mining*. 117–124.

[4] Gregory F Cooper and Edward Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine learning* 9, 4 (1992), 309–347.

[5] Michel C Desmarais, Peyman Meshkinfam, and Michel Gagnon. 2006. Learned student models with item to item knowledge structures. *User Modeling and User-Adapted Interaction* 16, 5 (2006), 403–434.

[6] Nir Friedman. 1997. Learning belief networks in the presence of missing values and hidden variables. In *ICML*, Vol. 97. 125–133.

[7] Nir Friedman. 1998. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. 129–138.

[8] José P. González-Brenes and Yun Huang. 2015. Your model is predictive— but is it useful? Theoretical and Empirical Considerations of a New Paradigm for Adaptive Tutoring Evaluation. In *Proceedings of the 8th International Conference on Educational Data Mining*.

[9] David Heckerman, Christopher Meek, and Gregory Cooper. 1997. *A Bayesian approach to causal discovery*. Technical Report. MSR-TR-97-05, Microsoft Research.

[10] Tanja Käser, Severin Klingler, Alexander Gerhard Schwing, and Markus Gross. 2014. Beyond knowledge tracing: Modeling skill topologies with bayesian networks. In *Intelligent Tutoring Systems*. Springer, 188–198.

[11] Kenneth R Koedinger, Albert T Corbett, and Charles Perfetti. 2010. The knowledge-learning-instruction (KLI) framework: Toward bridging the science-practice chasm to enhance robust student learning. *Cognitive Science* (2010).

[12] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[13] Philip I Pavlik Jr, Hao Cen, Lili Wu, and Kenneth R Koedinger. Using Item-type Performance Covariance to Improve the Skill Model of an Existing Tutor. *Educational Data Mining 2008* (????), 77.

[14] Judea Pearl. 2000. *Causality: models, reasoning and inference*. Vol. 29. Cambridge Univ Press.

[15] Chris Piech, Jonathan Spencer, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. 2015. Deep Knowledge Tracing. *arXiv preprint arXiv:1506.05908* (2015).

[16] Richard Scheines, Elizabeth Silver, and Ilya Goldin. 2014. Discovering prerequisite relationships among knowledge components. In *Educational Data Mining 2014*.

[17] Peter Spirtes, Clark Glymour, and Richard Scheines. 2001. *Causation, prediction, and search*. MIT Press.

[18] Jonathan Templin and Laine Bradshaw. 2014. Hierarchical diagnostic classification models: A family of models for estimating and testing attribute hierarchies. *Psychometrika* 79, 2 (2014), 317–339.

[19] Thomas Verma and Judea Pearl. 1990. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*. 255–270.

[20] Annalies Vuong, Tristan Nixon, and Brendon Towle. 2010. A method for finding prerequisites within a curriculum. In *Educational Data Mining 2011*.

[21] Annalies Vuong, Tristan Nixon, and Brendon Towle. 2011. A Method for Finding Prerequisites Within a Curriculum.. In *EDM*. 211–216.