Joint Discovery of Skill Prerequisite Graphs and Student Model From Student 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

Skill prerequisite information is useful for tutoring systems that assess student knowledge or that provide remediation. These systems often encode prerequisites as graphs designed by subject matter experts in a costly and time-consuming process. In this paper, we introduce *Combined student Modeling and prerequisite Discovery* (COMMAND), a novel algorithm for jointly inferring a prerequisite graph and a student model from data. Learning a COMMAND model requires student performance data and a mapping of items to skills (Q-matrix). COMMAND learns the skill prerequisite relations as a Bayesian network (an encoding of the probabilistic dependence among the skills) via a two-stage learning process. In the first stage, it uses an algorithm called Structural Expectation Maximization to select a class of equivalent Bayesian networks; in the second stage, it uses the domain knowledge that it is unlikely for a student to master a skill before mastering its prerequisites to select a single Bayesian network. Our experiments on simulations and real student data suggest that COMMAND is better than prior methods in the literature.

Keywords

Prerequisite discovery, Bayesian network, student modeling

1. INTRODUCTION

Course *curricula* are usually organized in a meaningful sequence that evolves from relatively simple lessons to more complex ones. Among these lessons, some are required to be mastered by the student before the subsequent ones can be learned. For instance, students have to know how to do addition before they learn to do multiplication. We refer to *prerequisite structure* as the relationships among skills that place strict constraints on the order in which skills can be acquired.

Prerequisite structures are crucial for designing intelligent tutoring systems that assess student knowledge or that offer remediation interventions to students. Building such systems require prerequisite information that is often hand-engineered by subject matter experts in a costly and time-consuming process. Moreover, the prerequisite structures specified by the experts are seldom tested and might be unreliable in the sense that experts may hold “blind spots".

Recent interest in computer assisted education promises large amounts of data from students solving *items*— questions, problems, parts of questions. In this paper, we introduce *Combined student Modeling and prerequisite Discovery* (COMMAND), a novel algorithm for simultaneously discovering prerequisite structure of skills and a student model from student performance data (what items a learner answers correctly). Here, the prerequisite structure suggests an ordering of skills, while the student model allows to make predictions of future student performance.

2. RELATION TO PRIOR WORK

Prior work has investigated the prerequisite relationship of items without considering their mapping into skills [7, 18]. Item-to-skill mappings (also called Q-matrices) are desirable because they allow more interpretable diagnostic information. Because of this, follow-up work [3, 5] has addressed how to discover the prerequisite relationship of skills. They do so by comparing data of a pair of skills using two different models. One model assumes that there is a prerequisite between the skill pair, and the other assumes independence. The two models can be compared using data likelihood [3] or association rule mining [5]. These works –although very promising– have important limitations that we address in this paper. First, they are only capable of estimating the pairwise relationships between skills and do not optimize the entire prerequisite structure. Second, it is unclear how to use the output of these prerequisite structures for student modeling, (i.e., to make predictions of future student performance). Third, they only provide a quantitative evaluation of simulated data. Overall, prior work has used real student data as an example, but has not developed a quantitative evaluation methodology.

A statistical technique called Bayesian network is useful to model skill structures [12]. Bayesian networks allow modeling the full structure of skills and can encode conditional independence between the skills. Unfortunately, prior work with Bayesian networks requires a domain expert to design the prerequisite structures (e.g.,[10]), because automatic techniques have not been demonstrated to scale to real student data [14]. We now describe the COMMAND algorithm that discovers a Bayesian network that encodes the prerequisite structure of skills.

3. THE COMMAND ALGORITHM

COMMAND learns the prerequisite structure of the skills from data with a statistical model called Bayesian network [13, 15]. Bayesian

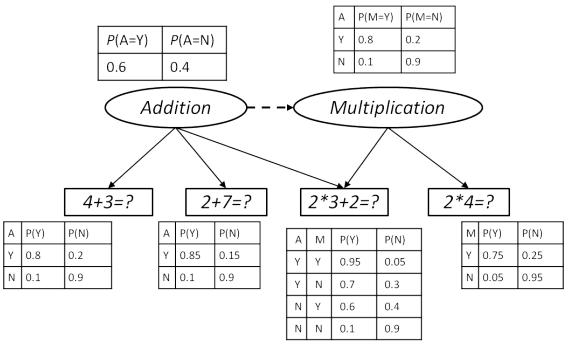


Figure 1: A hypothetical Bayesian network. Solid edges are given by item to skill mapping, dashed edges between skill variables are to be discovered from data. The conditional probability tables are to be learned.

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 Bayesian network from data (with the presence of unobserved latent variables).

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

Table 1: Example data matrix to use with COMMAND. 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
		...		

COMMAND 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

[1] $P(S_i = a|S_j = b)$ can be computed using any Bayesian network inference algorithm such as Junction tree algorithm[11].

Algorithm 1 The COMMAND 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 $\mathbf{S}$).

$G_0 \leftarrow \text{Initialize}(\mathbf{S}, Q)$ } Initialization
$i \leftarrow 0$ } Initialization
do } Structural EM
  E-step: } Structural EM
    $\theta_i^* \leftarrow \text{ParametricEM}(G_i, \mathbf{D})$ } Structural EM
    $\mathbf{D}_i^* \leftarrow \text{Inference}(G_i, \theta_i^*, \mathbf{D})$ } Structural EM
  M-step: } Structural EM
    $\langle G_{i+1}, \theta_{i+1} \rangle \leftarrow \text{BNLearning}(G_i, \mathbf{D}_i^*)$ } Structural EM
    $i \leftarrow i + 1$ } Structural EM
while stop criterion is not met } Structural EM
$RE \leftarrow FindReversibleEdges(G_i)$ } Discriminate between equivalent BNs
$EC \leftarrow EnumEquivalentDAGs(G_i)$ } Discriminate between equivalent BNs
$DE \leftarrow \{\}$ } Discriminate between equivalent BNs
for every reversible edge $S_i - S_j$ in RE **do** } Discriminate between equivalent BNs
  $ratio \leftarrow \frac{P(S_j=0|S_i=0)}{P(S_i=0|S_j=0)}$ [1] } Discriminate between equivalent BNs
  if $ratio \geq 1$ **then** } Discriminate between equivalent BNs
    $ratio^* = ratio$ } Discriminate between equivalent BNs
    $DE \leftarrow DE \cup S_i \rightarrow S_j$ } Discriminate between equivalent BNs
  else } Discriminate between equivalent BNs
    $ratio^* = \frac{1}{ratio}$ } Discriminate between equivalent BNs
    $DE \leftarrow DE \cup S_i \leftarrow S_j$ } Discriminate between equivalent BNs
  end if } Discriminate between equivalent BNs
end for } Discriminate between equivalent BNs
$sort(DE)$ by $ratio^*$ in descending order } Discriminate between equivalent BNs
while DE is not empty **do** } Discriminate between equivalent BNs
  $e \leftarrow dequeue(DE)$ } Discriminate between equivalent BNs
  if $\exists G \in EC\ e \in G$ **then** } Discriminate between equivalent BNs
    $\forall G \in EC$, remove G from EC if $e \notin G$ } Discriminate between equivalent BNs
  end if } Discriminate between equivalent BNs
end while } Discriminate between equivalent BNs
return EC } Discriminate between equivalent BNs

from educational data. We now describe the steps of COMMAND in detail.

3.1 Initial Bayesian Network

COMMAND represents the prerequisite structure using Bayesian networks that use latent binary variables to represent the student knowledge of a skill, and observed binary 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. 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.

3.2 Structural EM

A common solution to learning a Bayesian network from data is the score-and-search approach [6, 9]. This approach uses a scoring function[2] to measure the fitness of a Bayesian network structure to the observed data, and attempts 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

[2] A commonly used scoring function is Bayesian information criterion (BIC), which is composed of a likelihood term and a term to penalize the model complexity.

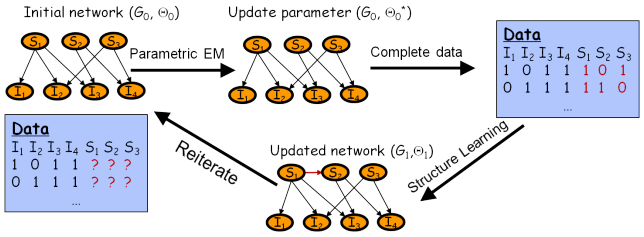


Figure 2: An illustration of the Structure EM algorithm to discover the structure of the latent variables.

where data contain 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. However, the number of possible Bayesian network structures is super-exponential with respect to the number of nodes. The Structural Expectation Maximization algorithm [8] 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.[3] We then do Bayesian inference to compute the expected values for the latent 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 criterion 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.

COMMAND'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 of another skill.

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 [8]. The general idea is, in the *E-step*, the algorithm also computes the expected values for missing data points, in addition for latent variables.

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

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[4]. For instance, Figure 3 gives an example of two simple Bayesian networks that are not distinguishable by Structural EM algorithm and the method in [14]. They share the skeleton but differ in the orientation of the edge (we will call such an edge a reversible edge). They apparently represent two different prerequisite structures.

3.3.1 Use Domain Knowledge

To determine a unique structure, we shall determine the orientation of each reversible edge. For this purpose, we propose to use the following domain knowledge.

Knowledge 1. If S_1 is a prerequisite of S_2, i.e., $S_1 \rightarrow S_2$, then $S_1 = 0 \Rightarrow S_2 = 0$.

Knowledge 1 says if a skill S_1 is the prerequisite of a skill S_2, a student can not master skill S_2 before he masters S_1, i.e., $P(S_2 = 0|S_1 = 0) = 1$. In other words, S_1 is not a prerequisite of S_2 if $P(S_2 = 0|S_1 = 0) = 1$ does not hold. This puts a constraint on the joint distribution encoded by the Bayesian network to be learned. In the case of determining the orientation of a reversible edge $S_1 - S_2$, we can check whether $P(S_2 = 0|S_1 = 0) = 1$ or $P(S_1 = 0|S_2 = 0) = 1$.[5] 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 conditional probability $P(S_2 = 0|S_1 = 0)$ is not exactly but close to 1. Thus, we use the following empirical rule:

Rule 1. if $P(S_2 = 0|S_1 = 0) \geq P(S_1 = 0|S_2 = 0)$, we determine $S_1 \rightarrow S_2$; otherwise, we determine $S_1 \leftarrow S_2$.

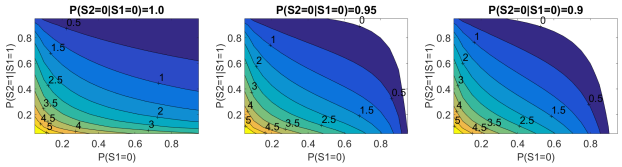
Note that these two conditional probabilities can be computed easily from the Bayesian network model output from Structural EM. The intuition behind this rule is that the conditional probability $P(S_2 = 0|S_1 = 0)$ can be interpreted as the strength of the prerequisite relationship $S_1 \rightarrow S_2$. The larger of this probability, 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 two probabilities and select the direction that is more probable.

[4] 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. [17].

[5] $P(S_2 = 0|S_1 = 0) = 1$ and $P(S_1 = 0|S_2 = 0) = 1$ can hold simultaneously. If $S_1 \rightarrow S_2$ is true, $P(S_1 = 0|S_2 = 0) = 1$ only if $P(S_1 = 1) = 0$ or $P(S_2 = 0|S_1 = 1) = 0$. $P(S_1 = 1) = 0$ means all students in the data do not know S_1. $P(S_2 = 0|S_1 = 1) = 0$ means learning S_2 becomes trivial once students know S_1. Here, we assume these two extreme cases never happen.

(a) (b)

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



(a) (b) (c)

Figure 4: Contour plots of $log(ratio)$ against $P(S_1 = 0)$ and $P(S_2 = 1|S_1 = 1)$ for various values of $P(S_2 = 0|S_1 = 0)$.

3.3.2 *Theoretical Justification*

Next we provide theoretical justification for the empirical rule proposed in § 3.3.1.. We consider the simple Bayesian network in Figure 3a, which has only one reversible edge $S_1 - S_2$. This model has three free conditional probability parameters: $P(S_1 = 0) = p$, $P(S_2 = 0|S_1 = 0) = q$, $P(S_2 = 1|S_1 = 1) = r$. If we observe data generated from this model, can we use the rule to discriminate between $S_1 \rightarrow S_2$ and $S_1 \leftarrow S_2$? We define

$$ratio = \frac{P(S_2 = 0|S_1 = 0)}{P(S_1 = 0|S_2 = 0)}. \quad (1)$$

Using Bayes rule and rules of probability, the rule $ratio \geq 1$ becomes $(1-p)(1-r) - p(1-q) \geq 0$. Since $ratio$ depends on p, q and r, we study how $ratio$ changes with these parameters. Figure 4 shows the contour plots of $log(ratio)$ against $P(S_1 = 0)$ and $P(S_2 = 1|S_1 = 1)$ for three different values of $P(S_2 = 0|S_1 = 0)$. The white region in each contour plot is the region where the rule is not applicable. That is, in this region, $ratio < 1$ even if $S_1 \rightarrow S_2$ is the true model. As it is showed, if $P(S_2 = 0|S_1 = 0) = q = 1$, the rule is always applicable no matter what values $P(S_1 = 0)$ and $P(S_2 = 1|S_1 = 1)$ take. With $P(S_2 = 0|S_1 = 0)$ decreasing, the white region becomes larger and the rule becomes more inapplicable. As mentioned, $P(S_2 = 0|S_1 = 0)$ can be interpreted as the strength of the prerequisite relationship. If we fix the value of $P(S_2 = 0|S_1 = 0)$ and assume that the two free parameters p and r are independent and uniformly distributed, then the area of the white region represents the probability that the rule makes a wrong decision. As the strength of the prerequisite relationship gets weaker, using the rule to determine the prerequisite relationship is less accurate.

3.3.3 *Orient All Reversible Edges*

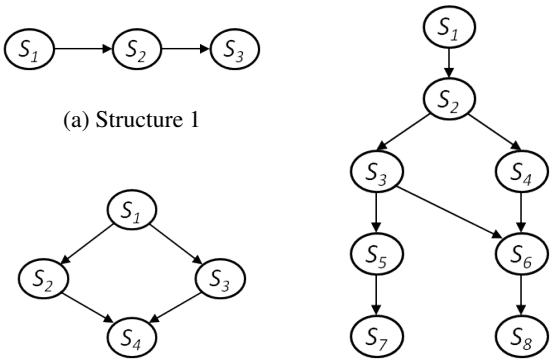
Using the empirical rule, we can orient every reversible edge in the network structure. However, orienting each reversible edge is not independent and may conflict with each other. Having oriented one edge would constrain the orientation of other reversible edges because we have to ensure the graph is a DAG and the equivalence property is not violated. For example, in Figure 5a, if we have determined $S_1 \rightarrow S_2$, the edge $S_2 \rightarrow S_3$ is enforced. In this paper, we take an ad-hoc strategy to determine the orientation for all reversible edges. For each reversible edge $S_i - S_j$, we let $ratio^* = ratio$ if $ratio \geq 1$ 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 by this ordering. When one edge is oriented, the constraint is propagated to other reversible edges. In practical implementation, we use the following 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 detailed in the *Discriminate between equivalent BNs* section of Algorithm 1. The $EnumEquivalentDAGs(G_i)$ implements the algorithm of enumerating equivalent DAGs in [4].

4. EVALUATION

In § 4.1, we evaluate COMMAND with simulated data to assess the quality of the discovered prerequisite structures. Then, in § 4.2 we use data collected from real students. In all our experiments, we use the Bayesian information criterion (BIC) as the scoring function in Structural EM .

4.1 Simulated Data

Synthetic data allow us to study how COMMAND compares to ground truth. For this, we engineered three prerequisite structures (DAGs), shown in Figure 5. Here, each figure represents different causal relations between the simulated latent skill variables.



(a) Structure 1

(b) Structure 2 (c) Structure 3

Figure 5: Three different DAGs between latent skill variables. Item nodes are omitted.

For clarity, Figure 5 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, and $P(S_j = 0|S_i = 0)$ for each edge $S_i \rightarrow S_j$ are restricted to be randomly selected in $[0.9, 1.0]$. 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 well COMMAND can discover the true prerequi-

site 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 measures 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.[6] 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 COMMAND's ability to recover the prerequisite structure? We consider the situation where in the model each item requires only one skill and the situation where each item requires multiple skills.
2. How well does COMMAND perform when there is noise in the data? We focus on studying noise due to the presence of unaccounted latent variables.
3. How well does COMMAND perform when the student performance data have missing values?
4. How is COMMAND compared with other prerequisite discovery methods? In particular, we compare COMMAND to the Probabilistic Association Rules Mining (PARM) method [5].

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 maps to 1-3 skills. Figure 6 compares the F_1 of adjacency discovery and edge orientation results 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.

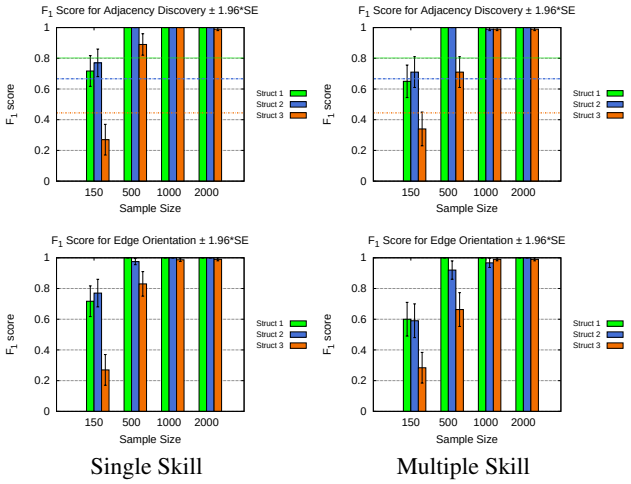


Figure 6: Comparison of F_1 scores for adjacency discovery (top row) and for edge orientation (bottom row). Horizontal lines are baseline F_1 scores computed for fully connected (complete) Bayesian networks. The error bars show the 95% confidence intervals, i.e., $\pm 1.96 * \text{SE}$.

4.1.2 Sensitivity to Noise

Real-world data sets often contain various types of noise. For example, noise may occur due to latent variables that are not explicitly modeled. To evaluate the sensitivity of COMMAND to noise, we synthesize the three Bayesian networks in Figure 5 to include 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 skills, 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 7 illustrates the procedure of this sensitivity analysis experiment for Structure 1.

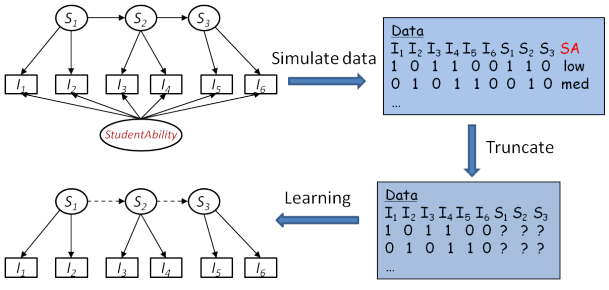


Figure 7: Evaluation of COMMAND with noisy data

Figure 8 compares the results where noise was introduced or not. Interestingly, the noise does not harm the learning accuracy at all, and actually improves the accuracy. This improvement is more evident when the sample size is small (see $n = 150$). We hypothesize that the

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

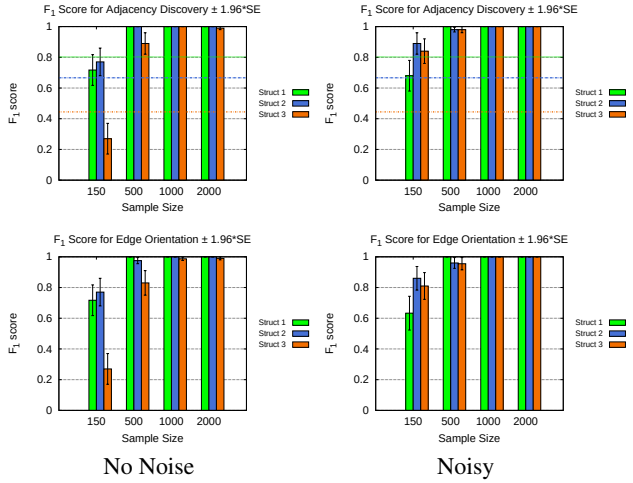


Figure 8: Results of adding systematic noise. Top: Comparison of F_1 scores for adjacency discovery. Horizontal lines are baseline F_1 scores computed for fully connected Bayesian networks. Bottom: Comparison of F_1 scores for edge orientation.

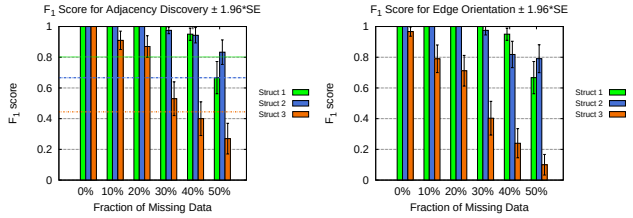


Figure 9: Results of learning with missing data. Left: Comparison of F_1 scores for adjacency discovery. Horizontal lines are baseline F_1 scores computed for fully connected Bayesian networks. Right: Comparison of F_1 scores for edge orientation.

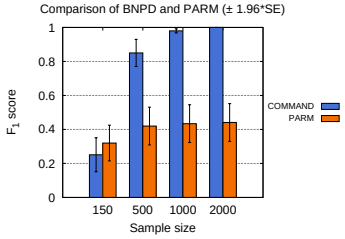


Figure 10: Comparison of COMMAND and PARM for discovering prerequisite relationships in Structure 3. The cutoff values used for PARM experiments are: $minsup = 0.125$, $minconf = 0.76$, $minprob = 0.9$.

correlations caused by *StudentAbility* node would be compensated by adding more edges between skill nodes. For small sample size, this might help because BIC score tends to give sparse structures.

4.1.3 In Presence of Missing Values

Real-world student data often have missing values, i.e., students do not respond to all items. To evaluate how COMMAND performs on data with missing values, we generated data sets of size 1000 with varying fraction of randomly missing values (10%, 20%, 30%, 40%, 50%) and used COMMAND to recover the structures from these data sets. Again, the models only contain single-skilled items. The results are presented in Figure 9. It is seen that the accuracy decreases when the fraction of missing values increases. However, COMMAND is still able to recover the true structures for Structure 1 and 2 even when the data contain 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 [5]. Since PARM discovers pair-wise prerequisite relationship, instead of constructing the full structure, we only compare COMMAND 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 5(c)) (with single-skilled items), which has 21 pair-wise prerequisite relationships. When experimenting with 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 [5]. 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 10. It shows that COMMAND significantly outperforms PARM for sample size $n \geq 500$. The low F_1 score by PARM is caused by low TPR, i.e., PARM failed to discover many prerequisite relationships (data not shown). It is speculated that selecting a different set of cutoff values for PARM might improve the results. However, determining these thresholds is not trivial and requires experts' intervention. By contrast, COMMAND does not require tuning of many parameters. Another problem of PARM is that it does not respect transitivity. That is, PARM may not identify $S_1 \rightarrow S_3$ even when it has discovered $S_1 \rightarrow S_2$ and $S_2 \rightarrow S_3$. This partially explains the low TPR.

4.2 Real Student Performance Data

We now evaluate COMMAND using two real-world data sets.

4.2.1 ECPE Data Set

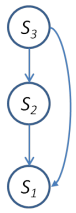
The ECPE (Examination for the Certification of Proficiency in English) data set was collected from a test by the English language Institute of the University of Michigan to examine individual cognitive skills required for understanding English language grammar [16]. It contains a sample of 2922 examinees who were 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 COMMAND is depicted in Figure 11. The estimated conditional probability tables (CPT) 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 [16] and that by the PARM method in [5]. Further, COMMAND also outputs the conditional probabilities associated with each skill and its direct prerequisite. We clearly see that the probability of student mastering a skill increases when the student has acquired more prerequisites of the skill.

4.2.2 Math Data Set

We now evaluate COMMAND using data collected from a commercial non-adaptive tutoring system. The textbook items are classified in chapters, sections, and objectives; but for this paper, we only

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

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



S_2	S_3	$P(S_1=0\|S_2,S_3)$	$P(S_1=1\|S_2,S_3)$
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 11: Result on ECPE data set. S_1: Morphosyntactic rules; S_2: Cohesive rules; S_3: Lexical rules. The estimated conditional probability tables (CPTs) are shown correspondingly.

use the data from Chapter 2 and Chapter 3 of the book. We use performance data while students solve test items. That is, students are tested on the items after they have been taught all relevant skills.

Q-matrix and preprocessing. We define skills as book sections. 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.[7] For each chapter, We process the data to find a subset of items and students that do not have missing values. That is, the datasets we use in COMMAND have students responding to *all* of the items.

After filtering, two data sets, `Math-chap2` and `Math-chap3`, were obtained for Chapter 2 and 3 respectively. In `Math-chap2`, six skills are included and each skill is tested on three to eight items, for a total of 30 items. In `Math-chap3`, seven skills are included and each skill has three to seven items, for a total of 33 items. `Math-chap2` includes student test results for 1720 students, while the `Math-chap3` has test results for 1245 students.

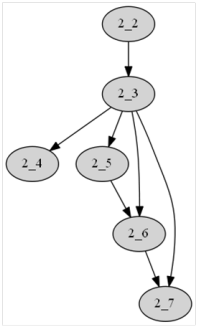
For simplicity we use binary variables to encode performance data (i.e, correct or incorrect) and skill variables (i.e., mastery or not mastery).

Prerequisite Structure Discovery. The Bayesian networks generated with the COMMAND algorithm are illustrated in Figure 12. Our observation is that the topological order of the sections in both structures are fully consistent with the book ordering heuristic. This shows an agreement between our fully data-driven method and human experts. We also ran PARM approach to learn pair-wise prerequisite relationships from these data sets. Given $minsup = 0.125$, $minconf = 0.76$ and $minprob = 0.9$, 2_5 $\rightarrow$ 2_6, 2_5 $\rightarrow$ 2_7 and 2_6 $\rightarrow$ 2_7 are discovered for `Math-chap2`, 3_1 $\rightarrow$ 3_3 and 3_2 $\rightarrow$ 3_3 are discovered for `Math-chap3`. These relationships are small subset of the set of relationships discovered by COMMAND.

Predictive Performance of Prerequisite Models. COMMAND not only discovers the topological relationships among skills, but also outputs a statistical model in the form of Bayesian network. This statistical model can be used for inference and predictive modeling. For example, given a student's response to a set of items, we can infer the student's knowledge status of a skill. This is very useful for remedial intervention, i.e., discovering when a

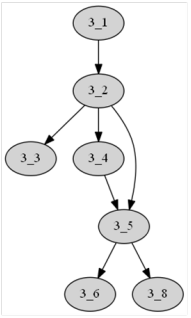
[7]Here we assume the items are single-skilled despite that they might be multi-skilled. In future a more accurate Q-matrix should be used.

Skill ID	Skill Name
2_2	Symbols and Sets of Numbers
2_3	Fractions and Mixed Numbers
2_4	Exponents, Order of Operations, Variable Expressions, and Equations
2_5	Adding Real Numbers
2_6	Subtracting Real Numbers
2_7	Multiplying and Dividing Real Numbers



(a) Prerequisite structure learned for `Math-chap2`.

Skill ID	Skill Name
3_1	Simplifying Algebraic Expressions
3_2	The Addition and Multiplication Properties of Equality
3_3	Solving Linear Equations
3_4	An Introduction to Problem Solving
3_5	Formulas and Problem Solving
3_6	Percent and Mixture Problem Solving
3_8	Solving Linear Inequalities



(b) Prerequisite structure learned for `Math-chap3`.

Figure 12: Prerequisite structures constructed by COMMAND for `Math` data sets.

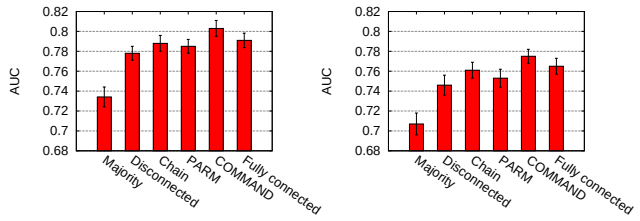
student is lacking background knowledge. In this paper, we evaluate these models by evaluating how well the generated Bayesian networks predict student performance on a test item given response on other items. In particular, we compute the posterior probability of a student's response to an item I_i given his performance on all other items $\mathbf{I}_{-i} = \mathbf{I} \setminus \{I_i\}$, by marginalizing over the set of latent variables $\mathbf{S}$:

$$P(I_i|\mathbf{I}_{-i} = \mathbf{i}_{-i}) = \sum_{\mathbf{S}} P(I_i, \mathbf{S}|\mathbf{I}_{-i} = \mathbf{i}_{-i}). \quad (2)$$

This can be computed efficiently using the Junction tree algorithm [11]. We then do binary classification based on the posterior probability. We compare the Bayesian network models generated from COMMAND with five 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 model assumes that the skill variables are marginally independent of each other. Most existing knowledge tracing approaches make this assumption.
- A Bayesian network model in which the skill variables are connected in a *chain* structure, i.e., 2-2$\rightarrow$2-3$\rightarrow$2-4$\rightarrow\ldots$ This assumes that a section (skill) only depends on the previous section. In other words, a first-order Markov chain dependency structure.
- A Bayesian network model constructed using the pairwise relationships output from *PARM*. That is, we create an edge $S_i \rightarrow S_j$ if PARM says S_i is the prerequisite of S_j.
- 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.



(a) `Math-chap2` AUC results. (b) `Math-chap3` AUC results.

Figure 13: Ten fold cross-validation results of evaluating the predictions of student performance.

The parameters of these baseline Bayesian network predictors are estimated from the data. The model predictions were evaluated using the *Area Under the Curve* (AUC) of the Receiver Operating Characteristic (ROC) curve metric calculated from 10-fold cross-validation. Results are presented in Figure 13. The error bars show the 95% confidence intervals calculated from the cross-validation. On both `Math-chap2` and `Math-chap3` data sets, the COMMAND models outperform the other five models. The *fully connected* models are the second best performing models. On `Math-chap2`, COMMAND model has an AUC of 0.803 ± 0.008 and the *fully-connected* model has an AUC of 0.791 ± 0.007 (Figure 13a). A paired t-test reveals that the AUC difference of two models are statistically significant with a p-value of 0.0022. On `Math-chap3`, COMMAND model has an AUC of 0.775 ± 0.007 and the *fully-connected* model has an AUC of 0.765 ± 0.008 (Figure 13b). The AUC difference of two models are also statistically significant with a p-value of 0.01. The *fully connected* models are outperformed by the much simpler prerequisite models, suggesting overfitting.

Using prerequisite structures to improve student models has been demonstrated in previous works [10, 2]. In their works, the prerequisite structures are hand-engineered by human experts. But here, we demonstrate with fully data-driven prerequisite structures.

5. CONCLUSION AND DISCUSSION

Discovering prerequisite structures of skills from data is challenging since skills are latent variables in the data. In this paper, we proposed COMMAND, a novel algorithm for jointly inferring a prerequisite graph and a student model from data. We demonstrated our approach using both synthetic and real-world data under different conditions such as data contain missing values or noises. We show the discovered prerequisite models are useful for student modeling and remedial intervention by demonstrating its predictive performance on unseen data.

COMMAND has many advantages over prior work on prerequisite discovery. First, it tries to optimize the full structure of skills instead of only estimating the pairwise relationship. Second, it does not require tuning of many parameters. For example, prior work [3] assumes domain parameters called *guess probability* and *slip probability* are provided for each pair of item and skill while COMMAND learns these parameters automatically. Similarly, more recent approaches [5] require manually specified thresholds to determine the existence of a prerequisite relationship. Determining these thresholds requires experts' intervention. By contrast, the only required input to our algorithm is the observed student performance and the Q-matrix.

Learning prerequisite graph as a Bayesian network from data is not merely a Bayesian network structure learning problem because we have to discriminate between equivalent Bayesian network structures which represent different prerequisite structures. We addressed this problem by using domain knowledge to refine the prerequisite models output from Structural EM. In future, it would be wiser to refine the model within the Structural EM learning. For example, in Structural EM search, we can use a scoring function with an additional term that penalizes the structures that violate the domain knowledge. In such way, learning may be more accurate.

6. REFERENCES

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

[2] Anthony Botelho, Hao Wan, and Neil Heffernan. 2015. The prediction of student first response using prerequisite skills. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. ACM, 39–45.

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

[4] Yetian Chen and Jin Tian. 2014. Finding the k-best Equivalence Classes of Bayesian Network Structures for Model Averaging.. In *AAAI*. 2431–2438.

[5] 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.

[6] 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.

[7] 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.

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

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

[12] RJ Mislevy, RG Almond, D Yan, and LS Steinberg. 2000. *Bayes Nets in Educational Assessment: Where do the Numbers Come From, CRESST/Educational Testing Service*. Technical Report. CSE Technical Reports 518.

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

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

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

[16] 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.

[17] 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.

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