Joint Discovery of Skill Prerequisite Graphs and Student Models

Yetian Chen[⋆†], José P. González-Brenes[†], Jin Tian[⋆]

[⋆]Computer Science Department
Iowa State University
Ames, IA, USA
{yetianc, jtian}@iastate.edu

[†]Advance Computing and Data Science Lab
Pearson
San Diego, CA, USA
jose.gonzalez-brenes@pearson.com

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 curriculum information 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 have "blind spots".

Recent interest in computer assisted education promises large amounts of data from students solving *items*— questions, problems, parts of questions. Performance data –what items a learner answers correctly– can be used to create *student models*. These models represent an estimate of skill proficiency at a given point in time [17]. For example, a student model can represent that Alice has already mastered integer addition, but Bob has not. Student models are often used to personalize instruction in tutoring systems or to predict future student performance. 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.

2. RELATION TO PRIOR WORK

Prior work has investigated how to discover prerequisites among items without considering their mapping into skills [6, 19]. Item-to-skill mappings (also called Q-matrices) are desirable because they allow more interpretable diagnostic information. Because of this, follow-up work [2, 4] has studied whether a pair of skills have a prerequisite relationship or not. For this, we can measure if a model that assumes a dependency between the two skills explain the data better than a model that assumes independence. This comparison can be done with data likelihood [2] or association rule mining [4]. Although very promising, these methods 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. explain this more, give a concrete example of where this fails? 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 do not provide a quantitative evaluation using real student data. Overall, prior work has used real student data for examples, but without any quantitative evaluation methodology that can help compare different techniques.

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

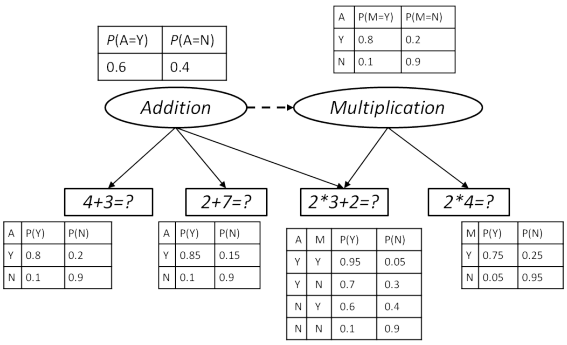


Figure 1: A hypothetical Bayesian network. Solid edges are given by item to skill mapping, dashed edges between skill variables are to be discovered from data. The conditional probability tables are to be learned.

3. THE COMMAND ALGORITHM

COMMAND learns the prerequisite structure of the skills from data with a statistical model called Bayesian network [13, 15]. Bayesian networks are one type of 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 [12], 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. We represent the prerequisite structure using Bayesian networks that use latent binary variables to represent the student knowledge of a skill (i.e., mastery or not mastery), and observed binary variables that represent the student performance answering items (i.e., correct or incorrect).

Algorithm 1 describes the COMMAND pipeline. The input to COMMAND is a matrix $\mathbf{D}$ with $n \times p$ dimensions, representing n students, answering p items. Each entry in $\mathbf{D}$ encodes the performance of a student (see Table 1 for an example). Additionally, we require a Q-matrix to represent the item-to-skill mapping. Q-matrices are often designed by subject matter experts but automatic methods to discover them exist [8].

COMMAND relies on a popular machine learning algorithm called *Structural Expectation Maximization* (Structural EM), which to the extent of our knowledge has not been used in educational applications before. Structural EM extends the Expectation Maximization (EM) algorithm to allow efficient discovery of edges between nodes, and not just inferring the CPT. is this ok? a reviewer didn't get that Structural EM wasn't EM-R.
A secondary contribution of our work is introducing Structural EM for learning Bayesian network structures from educational data. We now describe the steps of COMMAND in detail.

Table 1: Example student performance 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
		...		

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

3.1 Initial Bayesian Network

COMMAND first creates an initial Bayesian network using the Q-matrix by creating an arc to each item from each of its required skills. Because there are no edges between the skills, this initial network does not encode any prerequisite information. COMMAND uses Structural EM to learn arcs (prerequisites) between the skill variables.

3.2 Structural EM

A common solution to learning a Bayesian network from data is the score-and-search approach [5, 9]. This approach uses a scoring function (like the Bayesian Information Criterion (BIC)) to measure the fitness of a Bayesian network structure to the observed

data, and it 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 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 EM to estimate the scoring function. However, in this case EM takes a large number (hundreds) of iterations that require Bayesian network inference, which is computationally prohibitive. Further, we need run EM for each candidate structure, and the number of possible Bayesian network structures is super-exponential with respect to the number of nodes. The Structural EM algorithm [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 the *E-step*, it first finds the maximum likelihood estimate Θ^* of the CPTs for the current structure G calculated from previous iteration using parametric EM. It then does Bayesian inference to compute the expected values for the latent variables using the current model (G, Θ^*), and uses the values to complete the data. In the *M-step*, it uses the conventional score-and-search approach to optimize the structure according to the completed data (as if the latent variables were observed). 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.

We use an efficient implementation of Structural EM available online called LibB[1]. Because COMMAND's initialization step fixes the arcs from skills to items according to the Q-matrix, the *M-step* only needs to 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 can be easily extended 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** [7]. That is, some students do not answer all the items. The general idea is, in the *E-step*, the algorithm also computes the expected values for missing data points, in addition for latent variables.

3.3 Discriminate Between Equivalent BNs

Structural EM selects a 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 Structural EM is actually an equivalence class (EC) that may contain many Bayesian network structures[2]. These equivalent Bayesian networks have the same skeleton and the same v-structures[3]. For instance, Figure 3 gives an example of a simple equivalence class containing three 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 at least one of the edges (we will call such an edge a reversible edge). They apparently represent three different prerequisite structures.

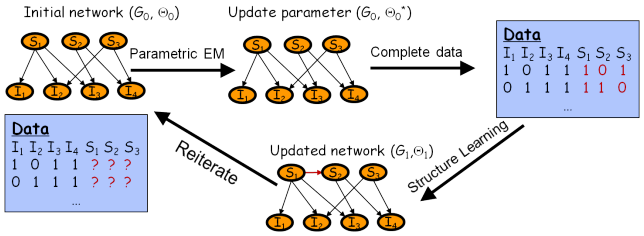


Figure 2: An illustration of the Structure EM algorithm to discover the structure of the latent variables. G represents the DAG structure. Θ is the set of conditional probability tables (CPTs).

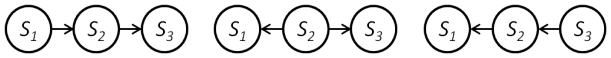


Figure 3: Three equivalent Bayesian networks representing different prerequisite structures.

3.3.1 Domain Knowledge

To determine a unique structure, we use a heuristic based in domain knowledge to determine the orientation of each reversible edge. For convenience in notation, let's assume that the random variables that represent skill proficiency can take two values: 0 if the skills is not mastered, and 1 if the skill is mastered. Our assumption is that if a skill S_1 is the prerequisite of a skill S_2, a student can not master skill S_2 before she masters S_1. More formally:

Assumption. If S_1 is a prerequisite of S_2 (i.e., $S_1 \rightarrow S_2$), then $S_1 = 0 \Rightarrow S_2 = 0$. In other words $P(S_2 = 0|S_1 = 0) = 1$.

Our assumption implies that S_1 cannot be 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.

For example, consider the case of choosing the orientation of a reversible edge $S_1 - S_2$ from $S_1 \leftarrow S_2$ or $S_1 \rightarrow S_2$. We can check whether $P(S_2 = 0|S_1 = 0) = 1$ or $P(S_1 = 0|S_2 = 0) = 1$ However, it is possible that our assumption does not hold, and a student got to master a skill even if he does not know the prerequisite. Moreover, because of statistical noise, the conditional probability $P(S_2 = 0|S_1 = 0)$ may not be exactly 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

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

[2] Structural EM outputs a DAG. However, the scoring function does not discriminate between the many DAGs of the equivalence class.

[3] A v-structure with nodes u, v, w are the directed edges $u \rightarrow v$ and $w \rightarrow v$ [18].

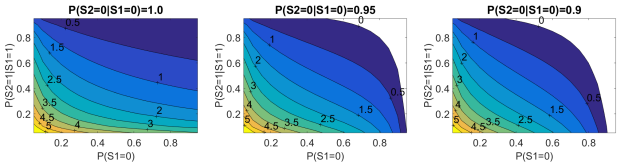
$P(S_2 = 0|S_1 = 0)$ can be interpreted as the strength of the prerequisite relationship $S_1 \rightarrow S_2$. The larger of this probability, the more likely the relationship $S_1 \rightarrow S_2$ holds. Since here we are concerned with which direction the edge goes, we simply compare the two probabilities and select the direction that is more probable. Note that $P(S_2 = 0|S_1 = 0) = 1$ and $P(S_1 = 0|S_2 = 0) = 1$ may hold simultaneously. If $S_1 \rightarrow S_2$ is true, then $P(S_1 = 0|S_2 = 0)$ is 1 only if $P(S_1 = 1) = 0$ or if $P(S_2 = 0|S_1 = 1)$ is 0 why??. If $P(S_1 = 1)$ is 0, this implies that no student knows S_1. If $P(S_2 = 0|S_1 = 1)$ is 0 means that learning S_2 becomes trivial once students know S_1. For simplicity, we ignore this extreme case.

3.3.2 *Theoretical Justification of Heuristic*

We now provide theoretical justification for the rule we propose. Consider a simple equivalence class, which contains two equivalent DAGs $S_1 \rightarrow S_2$ and $S_1 \leftarrow S_2$, where the true model is $S_1 \rightarrow S_2$. We have three free conditional probability parameters: $P(S_1 = 0) = p$, $P(S_2 = 0|S_1 = 0) = q$, $P(S_2 = 1|S_1 = 1) = r$. Let's define a *ratio* that quantifies choosing the true model:

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

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 our heuristic fails because $ratio < 1$. Figure 4(a) shows that when $P(S_2 = 0|S_1 = 0) = q = 1$, our heuristic rule is always correct, no matter what, because there is no white space. With $P(S_2 = 0|S_1 = 0)$ decreasing, the white region becomes larger and the rule becomes less accurate. 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, our rule to determine the prerequisite relationship becomes less accurate.



(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 $P(S_2 = 0|S_1 = 0)$.

3.3.3 *Orient All Reversible Edges*

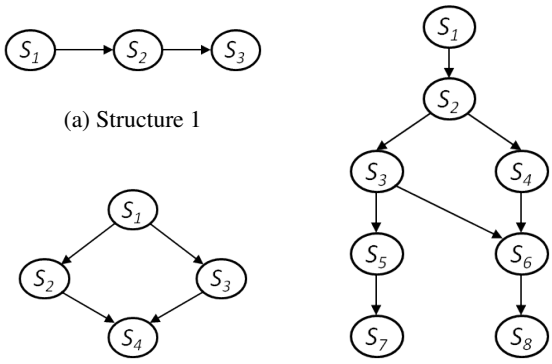
Using our proposed rule, we can orient every reversible edge in the network structure. However, orienting each reversible edge is not independent and may conflict with each other. Having oriented one edge would constrain the orientation of other reversible edges because we have to ensure the graph is 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. <- how do this different procedures differ? -> In our 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 [3].

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 BIC as the scoring function in Structural EM .

4.1 Simulated Data

Synthetic data allow us to study how COMMAND compares to the 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 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 three constraints. First, we enforce that achieving mastery of the prerequisites of a skill will increase the likelihood of mastering the skill. Second, for each prerequisite pair $S_i \rightarrow S_j$, $P(S_j = 0|S_i = 0)$ is randomly selected to be in $[0.9, 1.0]$. Finally, mastery of a skill increases the probability of student correctly answering the test item. 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 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 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[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 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 [4].

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 exactly 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 the two types of Q-matrices. With only 500 observations, COMMAND improves on a fully connected Bayesian Network baseline trained with ?? observations. COMMAND's accuracy improves with the amount of data, but its accuracy is slightly lower when the Q-matrix contains items that require more than one skill. A possible explanation for this is that multi-skill items may introduce more spurious correlations in the data. With just 2000 observations, COMMAND recovers the true structures almost perfectly.

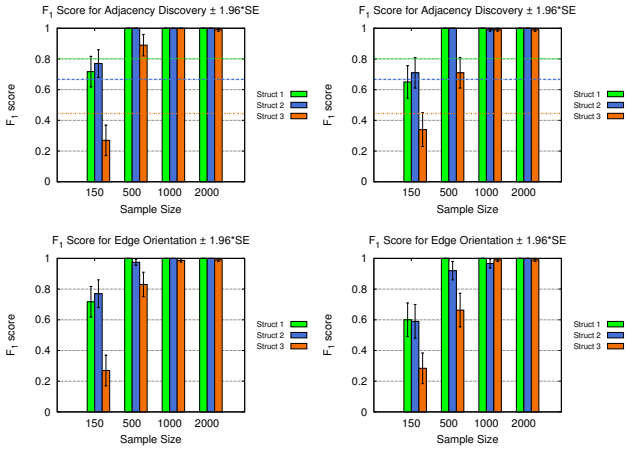


Figure 6: Comparison of F_1 scores for adjacency discovery (top row) and for edge orientation (bottom row). Horizontal lines are baseline scores for fully-connected (complete) networks. The error bars show the 95% confidence intervals, i.e., $\pm 1.96 * \text{SE}$.

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.

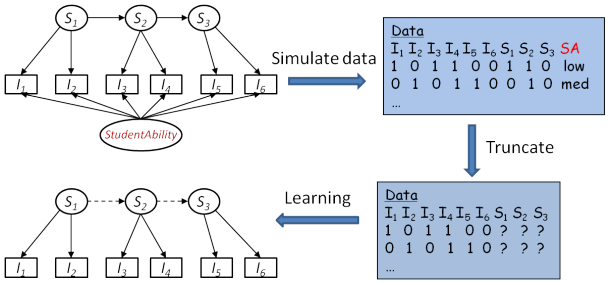


Figure 7: Evaluation of COMMAND with noisy data

Figure 8 compares the results where noise was introduced or not. Interestingly, the noise actually improves COMMAND's accuracy. This improvement is more evident when the sample size is small (see $n = 150$). We hypothesize that the correlations caused by

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

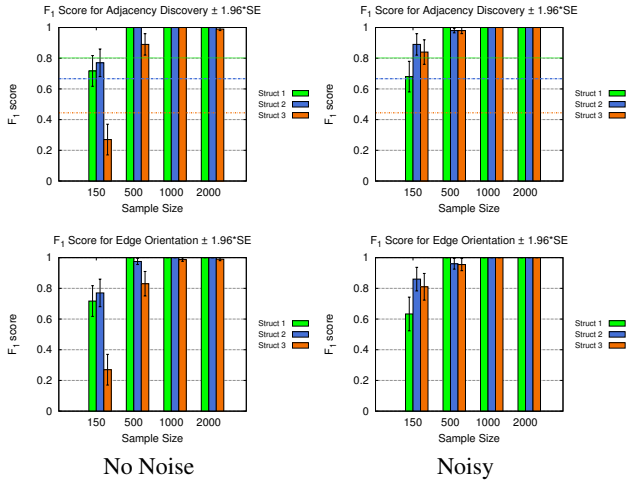


Figure 8: Results of adding systematic noise. Top: Comparison of F_1 scores for adjacency discovery. Horizontal lines are baseline F_1 scores computed for fully connected Bayesian networks. Bottom: Comparison of F_1 scores for edge orientation.

StudentAbility node would be compensated by adding more edges between skill nodes rewrite, what is a compensated correlation?? .

4.1.3 Sensitivity to Missing Values

Real-world datasets collected from students often have missing values, for example, when learners do not answer all items. To evaluate how COMMAND performs on data with missing values, we generated data sets of with 1000 observations with varying fraction of randomly missing values (10%, 20%, 30%, 40%, 50%). We used COMMAND to recover the structures from these data sets. Again, the models only contain single-skilled items. Figure 9 shows the results of this experiment. Although accuracy decreases when the fraction of missing values increases, COMMAND is able to recover the true structures for Structure 1 and 2 even when the data contain up to 30% missing values.

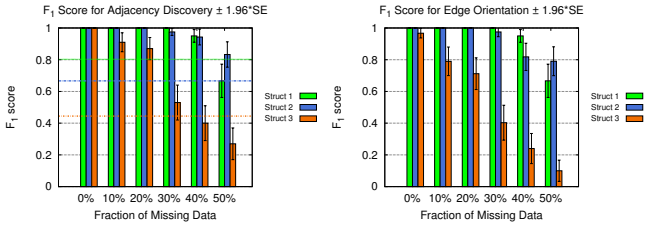


Figure 9: Results of learning with missing data. Left: Comparison of F_1 scores for adjacency discovery. Horizontal lines are baseline F_1 scores computed for fully connected Bayesian networks. Right: Comparison of F_1 scores for edge orientation.

4.1.4 Comparison With Prior Work

The Probabilistic Association Rules Mining (PARM) is a recent algorithm for discovering the prerequisite relationships between skills [4]. In this approach, a prerequisite relationship $S_1 \rightarrow S_2$ is considered if $P(S_1 = 1, S_2 = 1) \geq minsup \wedge P(S_1 = 1 | S_2 = 1) \geq minconf) \geq minprob$ and $P(P(S_1 = 0, S_2 = 0) \geq minsup \wedge P(S_2 = 0 | S_1 = 0) \geq minconf) \geq minprob$, where *minsup*, *minconf* and

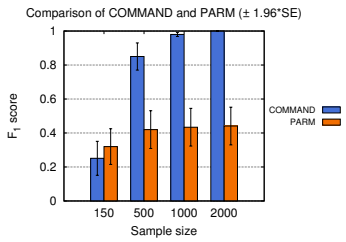


Figure 10: Comparison of COMMAND and PARM for discovering prerequisite relationships in Structure 3.

minprob are pre-specified constants between 0 and 1. PARM is limited to discovering pair-wise prerequisite relationships (instead of constructing the full structure). Because of this, we compare COMMAND with PARM for discovering the pair-wise relationships. We simulate data from Structure 3 from Figure 5(c) (with single-skilled items), which has 21 pair-wise prerequisite relationships. We derive pair-wise prerequisite relationships from this network and see how the two approaches discover these relationships. When experimenting with PARM, we use $minsup = 0.125$, $minconf = 0.76$, $minprob = 0.9$, because they were suggested by the authors [4].

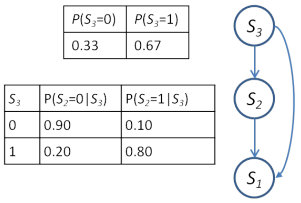
We compare our approaches using $F_1 = \frac{2*TPR*TDR}{TPR+TDR}$, where $TPR = \frac{\text{\# of correct relationships learned}}{\text{\# of relationships in true model}}$ and $TDR = \frac{\text{\# of correct relationships learned}}{\text{\# of relationships in learned model}}$. is this different from the Table? how and why? Figure 10 shows that COMMAND outperforms PARM, and the difference becomes significant for sample size $n \geq 500$. The low F_1 score of by PARM is because it fails to discover many prerequisite relationships (data not shown), and because PARM does not respect transitivity. For example, PARM may reject $S_1 \rightarrow S_3$ even it has discovered $S_1 \rightarrow S_2$ and $S_2 \rightarrow S_3$. We speculate that selecting a different set of cutoff values for PARM may improve the results. However, determining these thresholds is not trivial and may require experts' intervention. By contrast, COMMAND does not require tuning.

4.2 Real Student Performance Data

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

4.2.1 English Data Set

The Examination for the Certification of Proficiency in English (ECPE) datase describes 2922 examines in their understanding of English language grammar [16]. The dataset includes student performance in 28 items on 3 skills (S_1: morphosyntactic rules, S_2: cohesive rules, and S_3:lexical rules). Each item requires either one or two of the three skills.



$P(S_3=0)$	$P(S_3=1)$
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 11: The estimated DAG and CPTs of the ECPE data set.

Figure 11 shows the prerequisite structure discovered with COMMAND. It hypothesizes that lexical rules is a prerequisite of cohesive rules and morphosyntactic rules; cohesive rules is a necessary skill for learning morphosyntactic rules. The pair-wise prerequisite

relationships totally agrees with the findings in [16] and that by the PARM method in [4]. Our model infers a complete DAG, suggesting that there are no conditional independencies among the three skills. This is an interesting insight that previous approaches can not provide. 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. We only use student performance data from tests in Chapter 2 and 3. 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 a Q-matrix that assigns each exercise to a skill solely as the book section in which the item appears.[5] 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 and skill variables.

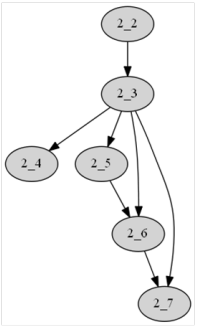
Prerequisite Structure Discovery. The Bayesian networks generated with the COMMAND algorithm are illustrated in Figure 12. Our observation is that the topological order of the sections in both structures are fully consistent with the book ordering heuristic. This shows an agreement between our fully 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. COMMAND outputs a Bayesian network model that 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. We could use COMMAND to identify students that may need remediation because they lack some background. We evaluate the accuracy of the predicted student performance on an item, when we observe the student response on the other items. More precisely, 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 $\mathbf{S}$:

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

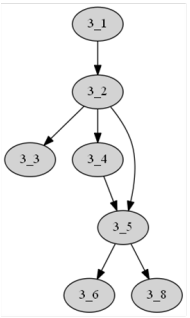
[5]Here we assume the items are single-skilled despite that they might be multi-skilled.

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



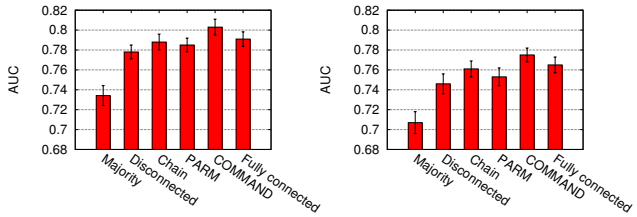
(b) Prerequisite structure learned for `Math-chap3`.

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

This probability can be computed efficiently using the Junction tree algorithm [11]. We then do binary classification based on the posterior probability to determine if the student is likely to answer correct. 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.

The parameters of these baseline Bayesian network predictors are estimated from the data using parametric EM. The model predictions were evaluated using the *Area Under the Curve* (AUC) of the



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

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

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.

5. CONCLUSION AND DISCUSSION

Prerequisite graphs have been shown [1, 10] to improve student models. However, discovering the prerequisites between skills requires significant effort from subject matter experts. The main contribution of our work is a novel algorithm that simultaneously infers a prerequisite graph and a student model from data with less human intervention.

We extend on prior work in significant ways. We optimize the full structure of skills instead of only estimating the pairwise relationships. Our experiments suggests that this results in better accuracy. Moreover, we argue that our strategy is easier to use because it does not require manual tuning of parameters. Other methods [2] require the *guess* and *slip* probabilities to be provided as input, or alternatively [4], thresholds to determine the existence of a prerequisite relationship. Determining these values requires experts' intervention. COMMAND does not require such tuning.

We analyze how missing values, noise and dataset size can affect the performance of COMMAND. Further research could explore additional datasets and baselines. A secondary contribution of our work is that we develop a methodology to evaluate prerequisite structures on real student data. We believe that we are the first to compare prerequisite discovery strategies by how well they can be used to predict student performance. Therefore, we validate COMMAND not only with synthetic data, but with two real-world datasets. Our results suggest that COMMAND improves on the state of the art because it significantly improves on a recently published technique.

Learning a prerequisite graph is not merely discovering a Bayesian network— equivalent Bayesian network structures in fact represent different prerequisite structures. We believe we are the first to address this problem. We use domain knowledge to refine the prerequisite models output using a theoretically motivated method.

6. REFERENCES

[1] Anthony Botelho, Hao Wan, and Neil Heffernan. 2015. The prediction of student first response using prerequisite skills. In *Learning At Scale*. ACM, 39–45.

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

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

[4] Yang Chen, Pierre-Henri Wuillemin, and Jean-Marc Labat. 2015. Discovering Prerequisite Structure of Skills through Probabilistic Association Rules Mining. In *Educational Data Mining*. 117–124.

[5] 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.

[6] 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.

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

[8] José P. González-Brenes. 2015. Modeling Skill Acquisition Over Time with Sequence and Topic Modeling. In *International Conference on Artificial Intelligence and Statistics*. 296–305.

[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] Robert J Mislevy, Russell G Almond, Duanli Yan, and Linda S Steinberg. 1999. Bayes nets in educational assessment: Where the numbers come from. In *Uncertainty in artificial intelligence*. 437–446.

[13] Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

[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] Kurt VanLehn. 1988. Student modeling. *Foundations of intelligent tutoring systems* 55 (1988), 78.

[18] Thomas Verma and Judea Pearl. 1990. Equivalence and synthesis of causal models. In *Uncertainty in Artificial Intelligence*. 255–270.

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