Constructing a Dynamic Bayes Net Model of Academic Advising

Joshua T. Guerin and Judy Goldsmith*
Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046
jtguer2@uky.edu, goldsmit@cs.uky.edu

Abstract

In this paper we apply ideas from collaborative filtering to the problem of building dynamic Bayesian network (DBN) models for planning. We demonstrate that item-based collaborative filtering can be used to construct dynamic Bayesian networks for use in large, factored domains with sparse data. Such Bayesian networks can model the transition function for decision-theoretic planning. We demonstrate the feasibility and effectiveness of this technique on an academic advising domain, based on student grades in computer science and related courses at the University of Kentucky.

1 Introduction

In this paper we examine the use of memory-based CF algorithms for constructing static models of data. This work is grounded in the real-world domain of academic advising. We use an item-based collaborative filtering algorithm to generate dynamic Bayesian network models of an advising domain from sparse grade data.

Collaborative filtering (CF) algorithms are designed to aggregate the opinions or preferences of a large number of users to extrapolate information about unnamed preferences for new and existing users. Recommendation systems are constructed using CF techniques to locate items in a database which a target user is likely to prefer. Preferences are typically defined by grades that the user provides either explicitly (by the user providing grades for items that have already seen) or implicitly (often indicated by patterns of behavior such as browsing habits). These grades can be represented in a number of ways, but are often numerical in nature; most recommender systems ask for a numerical grade (1–5) or a grade based on letters or "stars" which is easily mapped to numerical grade (for instance 1–5 stars, or a letter grade of A–E).

CF algorithms can be roughly divided into model-based and memory-based algorithms. Model-based algorithms involve generating a predictive model based on the data and using it to make preference-related predictions. One formalism that has seen success in model-based CF is the Bayesian network [1].

Memory-based CF operate over the database of items to make predictions, leveraging a measure of similarity between users or (more commonly) between items to determine grades for unseen items. This class of algorithms provides us with several notable features which are useful for making predictions. Namely, these algorithms are designed to operate over very large datasets (common examples include the Netflix dataset, the MovieLens datasets, or the Amazon.com recommendation system). Such datasets typically contain tens of thousands of items and grades from hundreds of thousands of users, however since most users only provide grades for a small percentage of items these datasets are very sparse. Because of this, modern recommendation systems must scale well and must work well with very sparse data.

2 A Predictive Model for Academic Advising

Reasoning in the domain of undergraduate academic advising is often approached as a deterministic process. Short and long-term decision making is based on the assumption that a student's actions (i.e., taking one or more courses) will succeed. This doesn't capture the nuances and complexity of the real world. The outcome of taking a course can not always be predicted with certainty; even a student who makes consistent A's may perform poorly at some point.

Given the stochastic nature of grade prediction, it may

*This material is based upon work supported by the National Science Foundation under Grant No. 1049360.

be desirable to construct statistics-based models of student performance from real world data. Students leave behind tangible evidence of progress in the form of transcript data. Universities amass a wealth of data with which to make predictions about grades. From this we can construct probabilistic predictive models. The Dynamic Bayesian Network (DBN) formalism has a number of features which make it ideal for this sort of modeling.

A DBN model consists of a directed acyclic graph with links representing temporal, probabilistic relationships between variables and conditional probability tables (CPTs) that specify those relationships quantitatively [2] (a discussion of DBNs will follow in Section 3.2). We are interested in a class of DBNs which model only a single time-step known as 2-slice DBNs. This imposes restrictions on the underlying graphical structure. Specifically, variable values at one time-step are conditioned only on the values of parent variables at the previous time-step.

The structural restrictions imposed on 2-slice DBNs make them a potentially compact representation for decision theoretic planning. For this reason we limit our attention to 2-slice DBNs.

In the case of discrete-valued variables, each child node in the DBN has an associated conditional probability table (CPT) which gives a probability distribution over possible values for every possible assignment to parent variables (incoming edges in the graph) at a previous time-step. Because all possible assignments to parent variables may need to be enumerated explicitly, CPT size is exponential in the number of parent variables. For example, a CPT for a single course with 5 parents, each of which has 6 possible values (A–D, Failure, and NOT_TAKEN) will have $6^5 = 7,776$ rows, each containing a probability distribution over the 6 possible outcomes.

For modern computers, tables of this size are unlikely to cause representational issues. However the need for enough data to populate a table's $6^6 = 46,656$ probabilities makes seemingly abundant data seem rather sparse. Popular or required courses may be taken by hundreds or even thousands of students within the span of several years, but even this is insufficient to derive realistic probability distributions from straight statistical analysis. This problem is worse for most courses (and for smaller colleges and universities) where enrollment over several years may reach only hundreds of students or fewer.

In order to deal with the problem of prediction when data is sparse, we turn to techniques from collaborative filtering to aggregate the data that is available. Collaborative filtering algorithms are commonly used to narrow down choices based on a user's preferences and the preferences of current and past users. A common example application is predicting preferences over unseen items (movies, music, groceries) based on grades given for other items [1].

The problem of grade prediction very closely resembles the problem of grade prediction in collaborative filtering: make predictions about a student's grades in untaken courses, given their past grades and the transcript data from many past students. Letter "grades" can map directly to integers where A=1 and Failure=5.

In this paper we present a simple collaborative filtering algorithm, and demonstrate how it is used to generate a valid DBN model of state transitions in the advising domain. We use real-world data from the Computer Science Department at our university as a testbed for our model generation techniques.

3 Background

3.1 Bayesian Networks

A *Bayesian network* is a directed acyclic graph $G = \langle V, E \rangle$, where each vertex $v \in V$ is a variable with domain $dom(v)$. Each $v \in V$ has an associated probability distribution over values in $dom(v)$, conditioned on the values of $Pa_v \subset V$, the parents of v. These conditional probability distributions are usually enumerated in tabular form as conditional probability tables (CPTs) for each variable.

Learning of Bayesian networks is often divided into structure learning and parameter learning. Structure learning is the problem of learning the graphical structure E by discovering predictive or causal dependencies between variables. Parameter learning is the problem of learning the conditional probability distributions for a given network structure.

Because the space of all possible networks is very large, structure learning is usually approached as a heuristic search problem or an exact search of a constrained version of the search space (see [4,6,11] for examples). Search for an an optimal (or near optimal) network structure is guided by some scoring function (one example is the log-likelihood scoring function).

Once structure is known, CPT parameters (probability distributions over outcomes) are generally learned from the data. Examples of parameter learning for DBNs include maximum likelihood estimation (one example being the expectation maximization [3] algorithm), or Bayesian estimation.

Unlike most Bayesian network learning algorithms, our

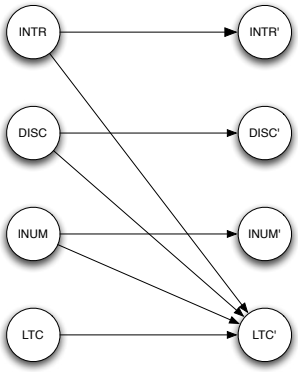

Figure 1: An example DBN structure.

validation is based on the quality of *predictions* rather than of *inference*. In other words, our work looks forward in time rather than backward. We conjecture that good learned probabilistic planning models may actually differ from probabilistic inference models learned from the same data.

3.2 2-Slice Dynamic Bayesian Networks

Bayesian networks have been demonstrated to be useful for inference in a number of domains, however the standard framework does not have an explicit notion of time. A *dynamic Bayesian network* builds upon the Bayesian network idea, incorporating temporal or sequential aspects of data into its structure. Variables at one time-step may influence the value of variables at future time-steps (or at the same time-step).

We are interested in a special case of dynamic Bayesian networks, the 2-slice dynamic Bayesian network. A 2-slice dynamic Bayesian network is a Bayesian network with $V = V, V'$, representing variables at time t and $t+1$, and edges from V to V' (and sometimes between vertices in V'). In DBNs of this form, V and V' may be visualized as two separate columns representing, respectively, the variables at time t and $t+1$.

This structural formulation implies two theoretical assumptions under which we operate. These are a *stationary assumption* where models are not time-dependant and a *Markov assumption* where there is no memory of past states; future values are conditioned *only* over the current system state.

Figure 1 gives the structure of an example of a 2-slice DBN which could be used for planning in an academic domain. This DBN structure shows that the expected grade in Logic and Theory of Computing (LTC) is conditioned over the grades obtained in Introduction to Programming (INTR), Discrete Mathematics (DISC), and Introduction to Numerical Methods (INUM).

Rather than selecting a single ideal structural size we choose to make structure size a parameter of our algorithm. Since we are considering models for the purpose of planning, we must consider the tradeoff between accuracy of the representation and tractability of planning. Our goal is to be able to generate DBNs of different sizes for different purposes. We examine how our algorithm fares as a function of structure size in Section 5. At this point we are left with the question of how to select n parent nodes for each node.

Goldenberg et al. approached a similar problem of learning Bayesian network structures from sparse data using frequent set mining [5]. Frequent sets are widely used in data mining for learning common co-occurrence between sets of items. The idea of applying frequent set mining to academic advising may be useful in other capacities (learning combinations of courses which should or should not be taken together), however co-occurrence of actions is less applicable to building predictive models of advising; courses which are frequently taken together are unlikely to make good predictors for each other. Parent courses should be taken before child courses, otherwise they provide little information.

Rather than using co-occurance we make the assumption that similar variables make better predictors than dissimilar variables. We examine the use of pairwise *item similarity* in selecting parent nodes. Item similarity is commonly used in collaborative filtering and other data mining applications to determine which items hold the most predictive power for a target item, allowing for better predictions to be made.

One of the most common approaches for collaborative filtering is to use the database of user grades to determine item-item similarity. For each pair of items in the database a vector of grades is created (retaining only grades where users voted for both items) [7]. To these vectors a number of distance metrics can be applied. In our implementation we tested two common vector similarity metrics: Pearson's correlation coefficient and cosine similarity.

3.3 Collaborative Filtering

Collaborative filtering recommendation algorithms typically fall into one of two general categories: model-based algorithms and memory-based algorithms [1]. Model-based algorithms involve generating a model based on data, and using the model to make predictions. We are interested in memory-based algorithms which use the entire data set to make predictions. This class of algorithms is described in Section 3.4.

Collaborative filtering algorithms also rely heavily on the notion of similarity. That is, similar users are likely

to assign similar grades to items. Likewise, similar items may also be given similar grades. Collaborative filtering systems often employ one of these assumptions. These are known as *user-based* and *item-based* collaborative filtering. In this paper we focus on the use of item-based collaborative filtering because of the performance demonstrated by these algorithms and because of their user-independent nature.

3.4 Item-Based Collaborative Filtering

The collaborative filtering algorithm that we used in this paper is an item-based algorithm presented by Sarwar et al. [7]. First, item-item similarity is calculated over all items in the database. For item-item similarity we are using Pearson's correlation coefficient and cosine similarity. For a user u and an item i, predictions are made using the weighted sum of u's grades for all items which are similar to i. This can be expressed as:

$$p_{u,i} = \frac{\sum_{all\ similar\ items,N}(s_{i,N} * R_{u,N})}{\sum_{all\ similar\ items,N(|s_{i,N}|)}}. \qquad (1)$$

Here, $p_{u,i}$ is the predicted grade that user u might give item i, $s_{i,N}$ is the similarity between items i and N, and $R_{u,N}$ is the grade that u provided for item N.

Equation 1 produces a single, most likely grade for the given user and item. Because a DBN requires a probability distribution over *all* possible grades, we are not yet ready to encode our DBNs.

4 Algorithm Details

The CF algorithm based on the function $p_{u,i}$ described in Section 3.4 defines a deterministic version of the DBN CPTs that we want to generate. We use these predictions and the data from past students' transcripts to generate probability distributions over possible grades to produce full CPTs. Algorithm 1 describes the process of turning deterministic predictions from $p_{u,i}$ into CPTs.

In this algorithm we make the assumption that deviation from predictions in past data will produce a distribution which is a reasonable approximation of the probability distribution.

Given the predictions from the CF function described in 4, we build a distribution table, *grade_distribution*, for the set of items with rows and columns indexed by predicted and actual grades. If G_1 and G_2 are possible grades, then the *grade_distribution*$[G_1][G_2]$ entry in the table is the number of transcripts for which the CF algorithm predicted G_1 and the student received G_2 for the class in question.

After we construct *grade_distribution* we normalize each row of the table to form probability distributions. For a grade g, row *grade_distribution*$[g]$ is now a probability distribution over actual grades when R predicts g.

```
Input: Past_Users - a database of past user grades.
Output: CPT - A set of CPTs for each course
foreach user in Past_Users do
    foreach item in user's graded items do
        p = p_{user,item};
        actual = actual grade for item;
        grade_distribution[p][actual]++;
    end
end
normalize rows of grade_distribution;
foreach item do
    T = create prediction table for item;
    foreach row in T do
        u* = temporary user using grade assignments
        in row;
        p = p_{user,item};
        add distribution from grade_distribution[p] to
        current row of T;
    end
    CPT(course) = T;
end
```

Algorithm 1: Generate DBNs from CF predictions

The second half of our algorithm constructs a set of prediction tables for each course. A prediction table T for a course c reflects the overall structure of a final CPT for c; each row of T contains a set of values for parent variables (defined by δ and our distance metric). For each row of T, we fill in the probability distribution over grades using the appropriate row of *grade_distribution*.

Each row of the prediction table T implies a hypothetical user transcript as an assignment over past grades. Using R we can make a prediction p for each row. We select a probability distribution from *grade_distribution*$[p]$, adding probability distributions over grades to each row of T.

After completion, CPT is a set of CPTs for each course, where $CPT(c)$ is the CPT for course c.

5 Results

In this section we describe the tests we run on the academic advising data. We evaluate the two variants of the item-based collaborative filtering algorithm on this dataset. We also generate two baseline DBN models and two collaborative filtering based models, and

analyze their performance on this dataset.

5.1 Data and Experimental Setup

Models are generated from the transcript data for approximately 4760 undergraduate students who enrolled during the 2000–2003 academic years. These anonymized data are a time-stamped (semester and year) series of transactions labeled with course and instructor information and grade outcomes. Because we have meta-data from computer science courses, we restricted our attention to students who took computer science courses during their academic careers.

Our analysis is broken down into two steps: collaborative filtering evaluation and DBN evaluation. We chose to evaluate the item-based collaborative filtering algorithm first to give a measurement of the algorithm's performance on an academic dataset. Testing of both collaborative filtering and DBNs is performed using 10-fold cross validation (partitioned randomly).

We are looking at two methods for evaluating the item-based collaborative filtering algorithm on this dataset: mean absolute error and the percent of misclassified predictions. Together, these statistics give us an indication of how far predictions are from actual grades and how often predictions are misclassified, respectively. We selected these statistics because they are fairly straightforward to interpret, and because mean absolute error has been used in the past for collaborative filtering evaluation, allowing comparison to performance on other datasets.

As a baseline for comparison of our 2-slice DBNs we generated baseline DBNs using more standard techniques. Baseline DBN structures were found through exhaustive search of the network structure space, using Bayesian information criterion (BIC) [8] as a scoring function. The highest scoring network was selected for a specified neighborhood size, and parameters were estimated using both maximum likelihood (ML) and Bayesian parameter estimation. Baseline DBNs were generated using the bnlearn software package [9].

As a means of evaluating the performance of the DBNs we calculated the log-likelihood loss of the models, and the percent of misclassified predictions. Log-likelihood loss is the negation of the log-likelihood, which we wish to minimize. "Predictions" in this case are similar to the deterministic predictions made by a collaborative filtering algorithm. We select the most likely outcome as a deterministic prediction and count the number that were classified correctly/incorrectly. This also gives us a basis for comparing our DBNs to the item-based collaborative filtering algorithm.

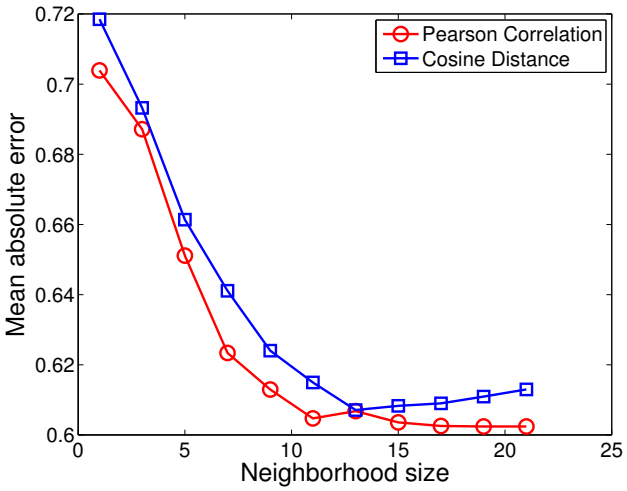

Figure 2: CF prediction mean absolute error

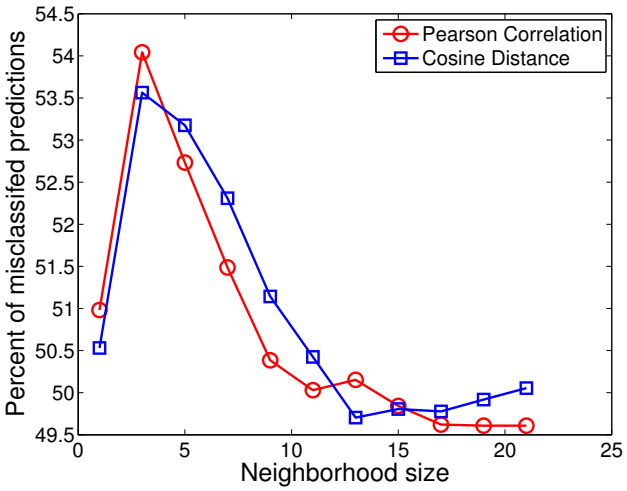

Figure 3: Percent of misclassified predictions.

5.2 Collaborative filtering evaluation

Figures 2 and 3 show that the two different distance metrics, Pearson's correlation and cosine similarity, yield DBNs that perform very similarly. These figures display the mean absolute error and percent of misclassified predictions for both Pearson correlation and cosine distance similarity metrics.

Figure 2 shows that mean absolute error decreases swiftly as the neighborhood size increases. After about 11 neighbors this decrease slows and little change is observed as the number of predictors continues to increase. This curve is similar to tests conducted on the MovieLens dataset [7].

Figure 3 shows how the percent of misclassified predictions changes as neighborhood size increases. At first there is an abrupt jump in this percent, however

afterward this curve resembles the curve for mean absolute error, with an apparent ideal neighborhood size of about 15 neighbors.

5.3 DBN evaluation

Figures 4 and 5 show that the DBNs learned using collaborative filtering (Pearson and cosine) outperform the baseline DBNs. Figure 4 shows the log-likelihood loss averaged over all models for a given neighborhood size. Figure 5 shows the percent of misclassified predictions for each model. Baseline DBNs are labeled as "Bayes" and "ML" for their parameter estimation methods. Collaborative filtering inspired DBNs are labeled "Pearson" and "Cosine" for the distance metric used in the collaborative filtering algorithm.

In terms of minimizing loss (Figure 4), the maximum-likelihood, Pearson, and cosine models show similar performance. At a neighborhood size of 2, these models have a log-likelihood loss tightly clustered around 1.14–1.16. Loss shows a steady decrease as the neighborhood size increases. However, the Bayesian model appears unable to cope with increasing neighborhood size, showing an increasing loss. This is likely due to the sparsity of data, and the increase in the possible number of configurations that corresponds with an increased neighborhood size.

Classification accuracy (Figure 4) shows steady improvement as neighborhood size increases across all models, with collaborative filtering models showing much better accuracy than Bayes and maximum-likelihood models at all neighborhood sizes. At a neighborhood size of only one the Pearson and cosine models show comparable accuracy (48.74-49.45% misclassified respectively) to the ML model Bayesian model at a neighborhood size of 5 (49.52% misclassified) and 7 (49.47% misclassified), respectively. At a neighborhood size of 10, the Pearson model shows the lowest misclassification rate at approximately 42.18%.

Comparing Figures 3 and 5, we find that at 6-7 parent variables, our baseline DBNs outperformed the item-based collaborative filtering algorithm. This is consistent with other experiments that demonstrated that Bayesian methods of classification showed better results than the standard item-based algorithm [10].

However, in terms of the percent of misclassified observations the CF-based DBNs outperformed both our benchmarks *and* the item-based collaborative filtering algorithm that they were based on at all neighborhood sizes. At 17 neighbors the CF algorithm hit a misclassification rate of approximately 49.6%, however at a neighborhood size of only 10 the CF-based DBNs had a misclassification rate of approximately 42.18%. This indicates that by observing the way that predicted

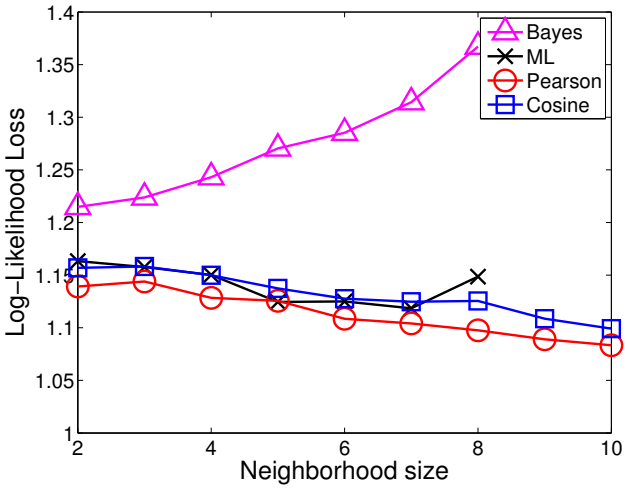

Figure 4: Average Log-Likelihood Loss.

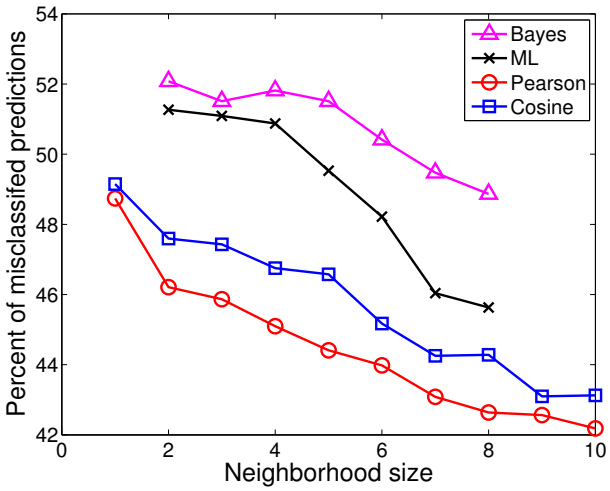

Figure 5: Observations misclassified by the DBN.

grades deviated from actual grades on a per-item basis (as we did with CPT generation in algorithm 1) one may be able to construct a better collaborative filtering algorithm.

Across all tests the Pearson model showed a slight advantage over the cosine model. This indicates that improvements in the item-based collaborative filtering used to generate DBNs may lead to improvements in resulting DBN models.

6 Conclusions and future directions

Our goal is to develop DBN transition models for the purpose of decision-theoretic planning. In this paper we have presented a novel approach for generating DBN planning models from sparse data. We used academic advising data to show the validity of

our method. One of the benefits of this method is the flexibility regarding the use of collaborative filtering recommendation algorithms. Our models were constructed using a generic item-based collaborative filtering algorithm. Any similar item-based collaborative filtering algorithm can be used in its place, giving us a wide variety of algorithms which can be employed using off-the-shelf software packages.

We are also investigating methods for modeling utility in this and similar domains, as well as decision-theoretic planning algorithms that can run on domains of the size and complexity presented here, and larger.

Acknowledgments

We thank the University's SSTARS lab consultants for assistance with statistical analysis. We also thank Nick Mattei, Robert Crawford, Daniel Michler and other members of our lab for their contributions to this and related projects.

References

[1] John Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52. Morgan Kaufmann, 1998.

[2] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1990.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[4] Nir Friedman, Iftach Nachman, and Dana Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *The 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 206–215, 1999.

[5] Anna Goldenberg and Andrew Moore. Tractable learning of large Bayes net structures from sparse data. In *ICML '04: Proceedings of the 21st International Conference on Machine learning*, New York, NY, USA, 2004. ACM.

[6] Kaname Kojima, Eric Perrier, Seiya Imoto, and Satoru Miyano. Optimal search on clustered structural constraint for learning Bayesian network structure. *Journal of Machine Learning Research*, 11:285–310, 2010.

[7] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[8] Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[9] Marco Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

[10] Xiaoyuan Su and Taghi M. Khoshgoftaar. Collaborative filtering for multi-class data using belief nets algorithms. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI '06, pages 497–504, 2006.

[11] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.