Selecting Bayesian Network Parameterizations for Generating Simulated Data

John Burge
Computer Science Department
University of New Mexico
Albuquerque, NM 87120

Terran Lane
Computer Science Department
University of New Mexico
Albuquerque, NM 87120

Abstract

The correct solution for an unsupervised learning task is never known. Thus, experimental results for such tasks must be validated. This can be done by turning the task into a supervised learning problem by training on simulated data where the correct solution is known and controllable. While this is a common practice, little research is done on generating interesting classes of random data that can be applied to many domains. We present a method to select the parameters of a Bayesian network (BN) which can be used to generate simulated data. Our methods allow the magnitude of correlations among random variables in the data to be precisely controlled. We apply our methods to the validation of BN structure search techniques and demonstrate that by precisely controlling correlations in the simulated data, qualitative differences among structure learning algorithms can be observed.

1 INTRODUCTION

There are a wide variety of models used to elicit correlations among random variables in a dataset. If the true correlations are never known, the elicitation is an unsupervised learning problem and the results must be validated. There are (at least) three methods for validation. First, the resulting learned models can be statistically analyzed to determine, for example, if the model was found by chance, if the model is not robust to deviations in the training data or if classifiers based on trained models perform well relative to other classification methods. Second, domain experts can be consulted to determine if the features are sensible. Third, the unsupervised learning problem can be transformed into a supervised learning problem by training on simulated data, where ground truth is known and controllable.

While validating with simulated data is a widely employed technique, little attention is given to the task of generating classes of simulated data that can be used for validating of a wide variety of experiments. In this paper, we propose a method for generating a class of simulated data such that the correlational strength among random variables can be precisely controlled. Specifically, we introduce methods for selecting parameters for Bayesian networks (BN) that can be used to generate the simulated data.

A BN is a compact representation of a joint probably distribution (JPD). Given a parameterized BN with a fixed topology, generating data consistent with the JPD represented by the BN is a straightforward and well understood problem. However, selecting a model that generates simulated data with desired features is more challenging.

To illustrate this, assume we are testing the hypothesis that a new BN structure search algorithm identifies weak correlations more effectively than another algorithm. To test this hypothesis, each algorithm could be used to learn a BN structure given a series of simulated datasets with finely controlled RV correlations. But how can such datasets be constructed? In other words, how would the appropriate parameters for the models used to generate the data be selected?

To facilitate the construction of such data, we annotate each of the links in a BN's topology with a score that quantifies how strongly correlated the link's parent node is with the child node. We use normalized mutual information scores (NMIS) for these quantifications. We then use an iterative hill-climbing algorithm to find a parameterizations for the BN that causes the BN's underlying JPD to be compliant with the specified NMISs.

For example, using our technique, it is possible to identify the CPT between a child, X_c, and its two parents, X_{p1} and X_{p2}, such that X_c is "strongly" correlated with X_{p1} but only "weakly" correlated with X_{p2}. The precise levels of correlation indicated by "strong" or "weak" are quantified

```
Assume X_1, …, X_n are topologically ordered
For m = 1 to desired number of data points
    For i = 1 to n
        j ← configuration Pa(X_i) is in
        rand ← Rand number between 0 and 1
        k ← 0
        while rand > 0
            rand ← rand - θ^B_{i,j,k}
            k ← k + 1
        end
        output k . "\t"
    end
    output "\n"
end
```

Figure 1. Simulated data generation algorithm. Generates a table of data from BN *B*.

with a NMIS and very subtle differences between correlational strengths are possible.

In the process of generating simulated data, it is often desirable to add noise into the data. Noise is commonly added by simply generating multiple datasets from the same distribution and then randomly changing some of the values in the datasets. In many cases, this process will *whiten* the data. I.e., the overall strength of correlations in the data will be diminished.

Our technique for generating simulated data allows for noise to be added by changing the values of the NMISs in the generative models. Thus, each generated dataset is different from other datasets (i.e., is noisy) but the data is not inadvertently whitened.

As an example of using the simulated data as a tool for validating unsupervised learning tasks, we demonstrate that different BN structure search methods result in qualitatively different behaviors given our simulated data. BN structure searches are a widely employed method for eliciting correlations in underlying data. One example is to use BN structure search to find correlations in neuroimaging datasets (Burge, 2007). Unfortunately, getting at the ground truth in this domain can be nearly impossible. Classification performance tends to also make a poor criteria to select between search methods, as the performance of competing methods is quite similar.

Instead, we construct simulated data that behaves similarly to the neuroimaging data and gauge the performance of different BN structure search methods on simulated data.

We also demonstrate how our methods can be used to generate a dataset that can easily be classified with BNs learned with *class-discriminative scores* such as CCL (Grossman & Domingos, 2004) or ACL (Burge & Lane, 2005) but not via *generative* scores such as BDe (Heckerman, Geiger, & Chickering, 1995).

Finally, we discuss issues that arise when generating data for dynamic BNs (networks that explicitly represent time) and conclude with a discussion on the limitations of our approach and avenues for future work.

2 BACKGROUND

2.1 BAYESIAN NETWORKS

In this section, we will briefly review BNs (Pearl, 1986) and the notation we use to describe them.

A Bayesian network is a compact representation of a joint probability distribution (JPD) over a set of n random variables (RVs), $\boldsymbol{X} = X_1, X_2, \ldots, X_n$. We will typically denote RVs with capital letters and sets of RVs as bolded capital letters. The arity of X_i is denoted r_i. The JPD is the function, $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(\boldsymbol{X} = \boldsymbol{x})$, which gives a probability that $\boldsymbol{X}$ takes on the values $\boldsymbol{x}$ in some dataset D where $1 \leq x_i \leq r_i$. We will usually denote the JPD with the shorthand $P(\boldsymbol{X})$ or $P(X_1, X_2, \ldots, X_n)$. The dataset D is composed of multiple data points, $D = \{d^1, d^2, \ldots, d^{|D|}\}$, where $|D|$ indicates the number of data points in the dataset. The value the i^{th} RV takes on in data point ℓ is denoted d_i^{ℓ}.

We will assume that the RVs have a finite discrete domain. Thus, the JPD can be represented with an n-dimensional table. As the number of RVs in a system increases, the size of the JPD grows exponentially, quickly becoming too large to represent explicitly. Using the chain rule of probability, a JPD can be rewritten as a product of conditional probabilities, $P(\boldsymbol{X}) = P(X_n) \prod_{i=1}^{n-1} P(X_i | X_{i+1}, \ldots, X_n)$. If two variables, X_a and X_b, are statistically independent, then $P(X_a \mid X_b) = P(X_a)$. Using the chain rule and assumptions of independence, the number of terms required to represent the JPD can be dramatically reduced.

BNs are a modeling framework that explicitly represents the conditional independence assumptions as links in a directed acyclic graph (DAG). RVs are represented as nodes in a directed acyclic graph (DAG) and dependencies between RVs are represented as directed links. Associated with each node is a set of conditional probabilities. These probabilities are frequently represented by a conditional probability table (CPT) (which we assume will be the case in this paper). CPTs are themselves collections of multinomial distributions, with a single multinomial describing the child's distribution given a unique setting of the parents.

We denote a BN, B, with the pair (B_S, B_Θ), where B_S specifies B's corresponding DAG and B_Θ contains the parameters specifying B's CPTs. Occasionally, we will represent a BN that was trained on just one class of data as B_1 or B_2 where the subscript corresponds to which class of data the BN represents.

B_S contains a node for each variable in the system. we will denote the nodes with the same notation used for the

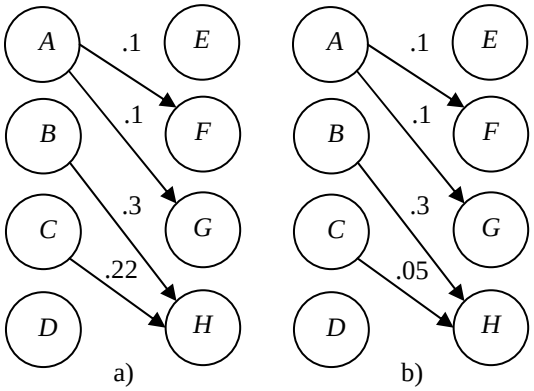

Figure 2. Two BNs that are annotated with the NMISs. The two annotations only differ in the $C \rightarrow H$ link.

underlying RVs. Thus "X_i" can either denote the i^{th} RV or the node that represents i^{th} RV in a BN. B_S also contains the links (arcs) in the DAG. The parents of X_i are denoted $Pa(X_i)$, which have $q_i = \prod_{\ell=1}^{K} r_{i,\ell}$ unique configurations, where $r_{i,\ell}$ is the arity of X_i's ℓ^{th} parent and $K = |Pa(X_i)|$, the number of parents X_i has. The set containing the child node and the child's parents is referred to as a *family*.

As a legal topology is acyclic, there always exists a topological ordering of the RVs such that a parent node occurs before its corresponding child in the ordering. Without loss of generality, we will assume that the RVs $X_1, \ldots, X_n$, are topologically ordered.

B_Θ is the set of CPTs for all of the nodes in B_S. It encodes the degree and magnitude to which the BN's families' parents are correlated with the children. If a node has K parents, the CPT for that node will have $K+1$ dimensions specifying $P\big(X_i|Pa(X_i)\big) = P\big(X_i|Pa_1(X_i), \ldots, Pa_K(X_i)\big)$ where $Pa_\ell(X_i)$ is the ℓ^{th} parent of X_i. B_Θ is composed of the individual CPTs, $\{\theta_i^B : 1 \le i \le n\}$ such that $\theta_i^B = P\big(X_i|Pa(X_i)\big)$. Each θ_i^B is composed of the multinomials $\big\{\theta_{i,j}^B : 1 \le j \le q_i\big\}$, such that $\theta_{i,j}^B = P(X_i|Pa(X_i) = j)$. Each $\theta_{i,j}^B$ is in turn composed of the individual probabilities, $\big\{\theta_{i,j,k}^B : 1 \le k \le r_i\big\}$ such that $\theta_{i,j,k}^B = P(X_i = k|Pa(X_i) = j)$.

2.2 SAMPLING DATA FROM BAYESIAN NETWORKS

Given a topology and a set of parameters for a BN, generating data is a straightforward process, Figure 1. Starting with the network's source nodes (at least one node with no parents is guaranteed to exist given the BN's topological ordering), randomly pick a value for the source nodes based on their CPTs (since the source nodes have no parents, their CPTs are simply single multinomial distributions). Then, continuing down RVs in the ordering, randomly pick a value for each node given the multinomial in the CPT corresponding to the values of the node's parents (which are guaranteed to already have values chosen for them due to the topological ordering).

Assuming all data points in a dataset are identically and independently distributed, this process can be repeated for each data point desired.

3 METHODS

In this section, we introduce our method for generating simulated data. We begin by describing the normalized mutually information score we use to quantify correlational strength. We then show how these scores can be calculated given a parameterized BN and conversely, how a parameterized BN can be found given a set of scores.

3.1 NORMALIZED MUTUAL INFORMATION SCORE

A BN's parameters determine how strongly correlated parents will be to their children. To quantify the strength of this correlation, we use mutual information (Cover & Thomas, 1991). As a convenience, we normalize the mutual information score so that it ranges between 0 and 1. We refer to this score as the *normalized mutual information score* (NMIS),

$$NMIS(X_c, X_p) = \frac{\sum_{\alpha=1}^{r_c}\sum_{\beta=1}^{r_p} P\big(X_c = \alpha, X_p = \beta\big)\log_2\left(\frac{P\big(X_c = \alpha, X_p = \beta\big)}{P(X_c = \alpha)P\big(X_p = \beta\big)}\right)}{\log_2\Big(min\big(r_c, r_p\big)\Big)} \quad (1)$$

The numerator is the mutual information of two discrete probability distributions and the denominator is the maximum value the numerator can be. When the two RVs share no mutual information, the NMIS equals zero. This occurs when the JPD is completely uniform. When the two RVs share total mutual information, the NMIS equals one. This occurs when the JPD contains only min(r_c, r_p) non-zero values such that no two values are placed in the same row or column.

We annotate a BN's topological structure with a NMIS at each link, as seen in Figure 2. This annotation provides a highly comprehensible summary of the strength of correlations in the BN given a parameterization. Given a parameterization for the BN, it is possible to calculate the corresponding NMISs. For each link, this requires the JPD of the parent and child, $P(X_c\ ,\ X_p)$, where X_p and X_c are the parent and child nodes. This distribution is not directly given by the parameters of a BN, but it can be calculated as the product of the marginal distribution of the parents $P\big(Pa(X_c)\big)$ and the conditional distribution of the child given the parents $P\big(X_c|Pa(X_c)\big)$.

$$P\big(X_c, X_p\big) = \sum_{\downarrow\{c,p\}} P\big(X_c|Pa(X_c)\big)P\big(Pa(X_c)\big) \quad (2)$$

$P(X_c, X_p) \leftarrow \sum_{\downarrow\{X_c,X_p\}} P(X_c|Pa(X_c))P(Pa(X_c))$
score ← NMIS(X_c , X_p)
γ ← some small value (e.g., 0.01)
while |*score* – *Target*| > *Precision*
    //select the indices to perturb mass in
    ParentIndex ← random # between 1 and r_p
    OtherParentIndex ← random # between 1 and r_p
                (not equal to *ParentIndex*)
    ChildSourceIndex ← random # between 1 and r_c
    ChildTargetIndex ← random # between 1 and r_c

    //Perturb the mass
    P(*ChildSourceIndex*, *ParentIndex*) −= γ
    P(*ChildTargetIndex*, *ParentIndex*) += γ

    //Make counterbalancing perturbations
    P(*ChildSourceIndex*, *OtherParentIndex*) += γ
    P(*ChildTargetIndex*, *OtherParentIndex*) −= γ

    //Determine if perturbations should be accepted
    *new*Score ← NMIS(X_c, X_p)
    if |*newScore* – *Target*| > |*score* – *Target*|
        Undo all mass transfers
    else
        score ← *newScore*
    End if
end while

Figure 3. Simplified version of an algorithm to sample JPDs for RVs X_c and X_p such that the NMIS for the resulting JPD is at least within *Precision* of a *Target* NMIS. The actual algorithm improves the running time of the algorithm by controlling the size of increment variable γ in a fashion roughly analogous to cooling schedules in simulated annealing algorithms.

where the ↓ { } notation indicates the *marginalize down to* operation. The JPD of the patents, $P(Pa(X_c))$, is not directly provided by the parameters of the BN, but can be computed from the parent's CPTs and the JPDs of the parents' parents (i.e., X_c's grandparents),

$$P(Pa(X_c)) = \sum_{\downarrow\{Pa(X_c)\}} \bigcup_{p=1}^{|Pa(X_c)|} \left[P\left(Pa_p(X_c)|Ga_p(X_c)\right)P\left(Ga_p(X_c)\right)\right], \quad (3)$$

where $Ga_p(X_c)$ is the set of grandparents of X_c that are also the parents of $Pa_p(X_c)$ and the union operator merges JPDs together such that the set of RVs in the merged JPD is the union of all the JPDs being merged. This calculation requires the JPD for the grandparents, which are in turn recursively calculated from the JPDs of the grandparents' parents. Eventually, this recursive procedure halts when the JPDs for nodes without parents are reached. These nodes are guaranteed to exist given the BN's topological ordering and indeed, efficient calculation of a BN's NMIS scores proceeds based on that ordering.

We refer to the function that returns the NMIS for a parent-child link as $NMIS(X_c, X_p)$.

3.2 DETERMINING PARAMETERS FROM NMIS SCORES

Figure 3 gives the algorithm to fill in a JPD for RVs X_c and X_p such that their NMIS score is (almost) equal to a target NMIS. The basic idea is to start with the JPD distribution $P(X_c, X_p)$ as initially calculated via the marginalization down to $P(X_c, X_p)$ of $P(X_c|Pa(X_c))P(Pa(X_c))$. Then, probability mass is perturbed across random locations in the JPD such that the conditional probability $P(X_c \mid X_p)$ is altered but the marginal distributions are not. By only moving probability mass between JPD elements with the same parent index, the parent marginal is unchanged. However, such a move will still modify the child marginal. This can be fixed by making counterbalancing perturbations of an equal amount of probability mass among JPD elements with the same child indices, but differing parent indices. Figure 4 provides a simple example of a probability mass transfer and how the marginals remain unaffected.

After each perturbation, the NMIS is recalculated and if the new NMIS is closer to the target NMIS, the perturbation is accepted. Otherwise, it is rejected. This simple and straightforward sampling strategy is guaranteed to converge on a solution for a single parent-child JPD (if it exists) since mutual information is a convex function of $P(X_c \mid X_p)$ for a fixed $P(X_p)$ (Cover et al., 1991) (Theorem 2.7.4).

However, this algorithm is not guaranteed to find all parameterizations for the BN as a whole. This is because the marginal distribution of the parents provided to the algorithm for a single family may not have a distribution of mass that will allow arbitrary NMIS scores to be satisfied. In practice, we have found that the only time solutions are not found is when strong correlations are requested among many RVs. For instance, if all the links in Figure 2 were set to .99, a legal parameterization would not likely be found, however, reductions in some of the NMIS would eventually allow for the identification of valid parameterizations.

After the algorithm is run for each parent-child pair in a family, the JPDs are merged into a single JPD over all parents and the child. The family's CPT is then calculated from the JPD, which is then saved for future applications of the algorithm on descendant families. We refer to the overall method of finding the CPT parameters as the *NMISGen* algorithm.

NMISGen can also be used to modify a previously selected CPT. For instance, this occurs if a second class of data is desired that only differs from the first class by

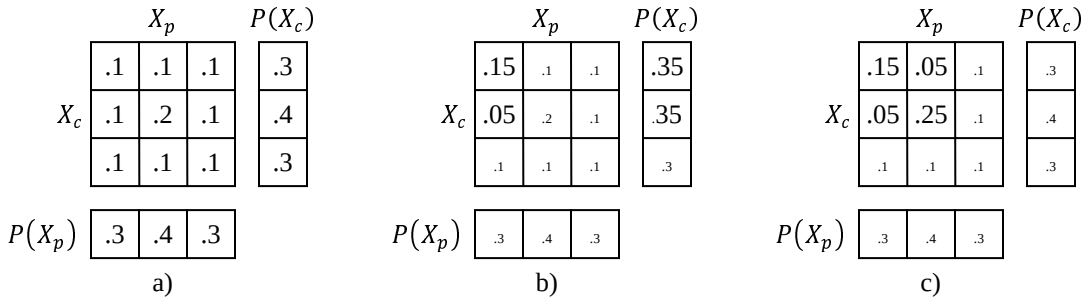

Figure 4. Examples of perturbing probability mass. The 3x3 matrix is the $P(X_c, X_p)$ JPD. The 1x3 and 3x1 vectors are the parent and child marginals. a) Hypothetical initial JPD and corresponding marginals. b) The results of moving 0.05 probability mass from the 2,1 position to the 1,1 position. This affects the child marginal, but the parent marginal is unchanged. Small-font values indicate elements that remain unchanged. c) A counterbalancing transfer of 0.05 mass from the 1,2 position to the 2,2 position to repair the change made to the child marginal in b).

one NMIS value. The CPTs for the first class would be generated and copied into the second class. The CPT for the link with a different NMIS link would then be calculated by running NMISGen with the final JPD selected by NMISGen for the first class as the starting distribution. Since NMISGen does not affect the child marginal distributions, the new JPD created for the second class does not require recalculating JPDs for other nodes in the network in order to maintain the marginal distributions of other RVs that are the descendants of the node with a new CPT.

We refer to the distribution of CPTs sampled by NMISGen as the $P(\Theta_i^B|S_i^B)$ distribution, where Θ_i^B is the set of CPT parameters for node X_i in BN B and $S_i^B = \{s_{i,1}^B, \ldots, s_{i,p}^B\}$ is a list of NMIS's, one NMIS for each of the p parents of X_i. When NMISGen is used to come up with a new parameterization given a previous parameterization and a new list of NMISs, we will refer it as the $P(\Theta_i'^B|\Theta_i^B, S_i'^B)$ distribution where $\Theta_i'^B$ is the new resulting CPT distribution and $S_i'^B$ is the new set of NMIS scores. The closer $S_i'^B$ is to S_i^B, the smaller the Kullback-Leibler (KL) divergence between multinomials in $\Theta_i'^B$ and Θ_i^B will be. If $S_i'^B = S_i^B$, then $\Theta_i'^B = \Theta_i^B$.

3.3 EXTENSIONS TO DYNAMIC BAYESIAN NETWORKS

In some domains, it is important to explicitly model time. BNs that do this are often referred to as dynamic Bayesian networks (DBNs) (Murphy, 2002). For the purposes of this paper, the distinction between a BN and a DBN is only significant in one regard. The data for a DBN are time series and each time series can be represented by more than one RVs in the DBN topology.

For instance, see the DBNs in Figure 5. Each node is denoted with two values, the subscript indicates which time series the node represents (each data point contains multiple time series), and the superscript indicates which point in time the node corresponds to. In this example, the subscripts range from 1 to 4, indicating that there are 4

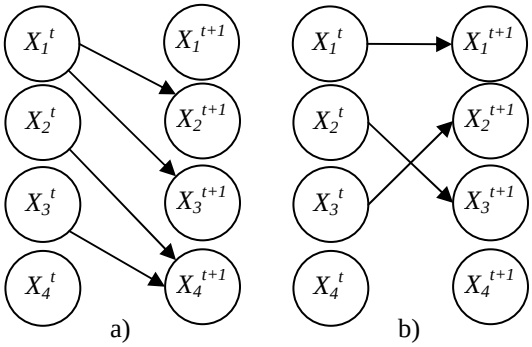

Figure 5. DBNs and temporal cycles. a) DBN with no temporal cycles. b) DBN with two temporal cycles. The $X_1^t \rightarrow X_1^{t+1}$ link represents a temporal cycle, as well as the pair of links $X_1^t \rightarrow X_1^{t+1}$ and $X_1^t \rightarrow X_1^{t+1}$.

individual time series in each data point. The superscripts are *t* and *t+1*, indicating that correlations between the nodes represent correlations among temporally consecutive time points in the 4 time series.

NMISGen selects probability mass perturbations that move mass between elements of the JPD $P(X_c, X_p)$. This JPD was computed from the product of the conditional distribution $P(X_c \mid X_p)$ and the parent marginal distribution $P(X_p)$. It is assumed that the parent marginal distribution is known and that it remain fixed. However, if X_c and X_p correspond to the same time series, then modifications to the conditional distribution will result in modifications to the marginal distribution. Thus, the marginal cannot be known and certainly cannot remain fixed. When generating data for dynamic Bayesian networks, NMISGen places constraints on the DBN's topologies. "Temporal cycles" are not allowed. The BN in Figure 5a does not contain any temporal cycles, but the BN in Figure 5b contains two.

To adhere to this constraint, NMISGen places an ordering restriction on the nodes in a DBNs. There must be an ordering of RVs in the DBN such that if X_i^t is a parent of

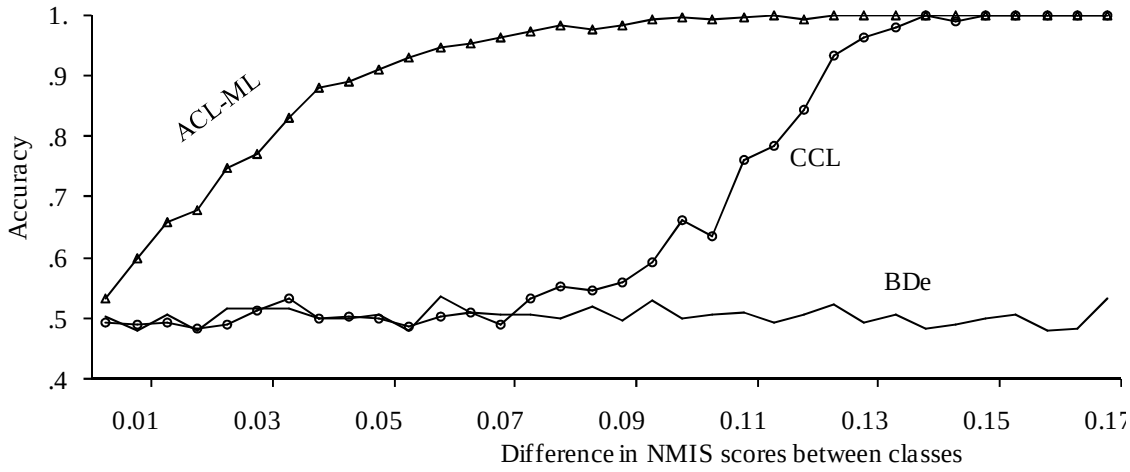

Figure 6. Results of simulated data that is easier for class-discriminative trained BNs to classify than for generatively trained BNs to classify.

X_j^{t+1}, then $i < j$ in the ordering. In other words, all of the links in the DBN must "point downwards" when the nodes are placed via the ordering as the descend downwards.

This is not to say that no algorithm exists which can generate simulated data for temporally-cyclic DBNs. Determining such an algorithm is left for future work.

4 EXPERIMENTS

4.1 GENERATIVE VERSUS CLASS-DISCRIMINATIVE CLASSIFICATION

In BN structure search, a score is employed to determine how well a proposed topological structure corresponds to an underlying dataset. Many commonly used scores are *generative* scores, as they select structures that increase the posterior likelihood of the BN given the data. However, when BNs are being trained to create a classifier, structures should be selected for their ability to discriminate between classes.

We demonstrate how BN parameterizations can be selected such that data generated from the BNs will be difficult for new BNs trained with a generative score to classify, but will be easier for BNs trained with a class-discriminative score to classify[1]. Refer back to the two BNs in Figure 2. The only difference between the BNs is the $C \rightarrow H$ link in one class has an NMIS of .22 and only 0.05 in the other class. Further, the $B \rightarrow H$ link has the same NMIS score of 0.3 in both classes—which is higher than the $C \rightarrow H$ link in either class. In other words, the link that actually differs between classes has a relatively weaker correlation than the link that does not change between classes. NMISGen can be used to obtain a set of CPTs for each of these BNs. Data can then be generated from these BNs.

Given the generated simulated data, we learn the structure for two new BNs—one to represent each class of data—using the class-discriminative CCL score (Grossman et al., 2004), the class-discriminative ACL score (Burge et al., 2005) or the generative BDe score (Heckerman et al., 1995). Each search was only allowed to learn a single parent per node[2]. Thus, in order for classification to be successful, the structure search would need to find the $C \rightarrow H$ link, which has a weaker correlation than the $B \rightarrow H$ link in both classes.

The learned BNs were then used to classify additionally generated testing data points that were not used to learn the networks (see (Burge et al., 2005) for more details). Figure 6 demonstrates the results of classification experiments performed on the generated data. The classification experiments were performed across a series of simulated datasets with varying degrees of difference between the NMIS score associated with the $C \rightarrow H$ link in each class.

In the BNs given in Figure 2, the difference between the NMIS scores of the $C \rightarrow H$ link is 0.17. At this point, represented at the far right hand side of the graph in Figure 6, the differences between the $C \rightarrow H$ correlation in both classes of data is easily observed by all structure learning scores. However, the BDe score recognizes that the $B \rightarrow H$ link has a stronger correlation and selects that link over the $C \rightarrow H$ link. Thus, BDe fails to identify a structure that can discriminate between the classes of

[1] Note that BNs are now being used in two contexts. They are being used to generate simulated data via our proposed methods and in these experiments, BN structure search is being used to learn BNs that represent the correlations in the simulated data.

[2] Otherwise, every search would find the class-discriminating link. In real-world BNs, limiting structure search to a single parent is overly restrictive, but the searches will always be limited to some degree. The point of this experiment is to demonstrate that generative scores do not favor class-discriminative links over links with strong correlations.

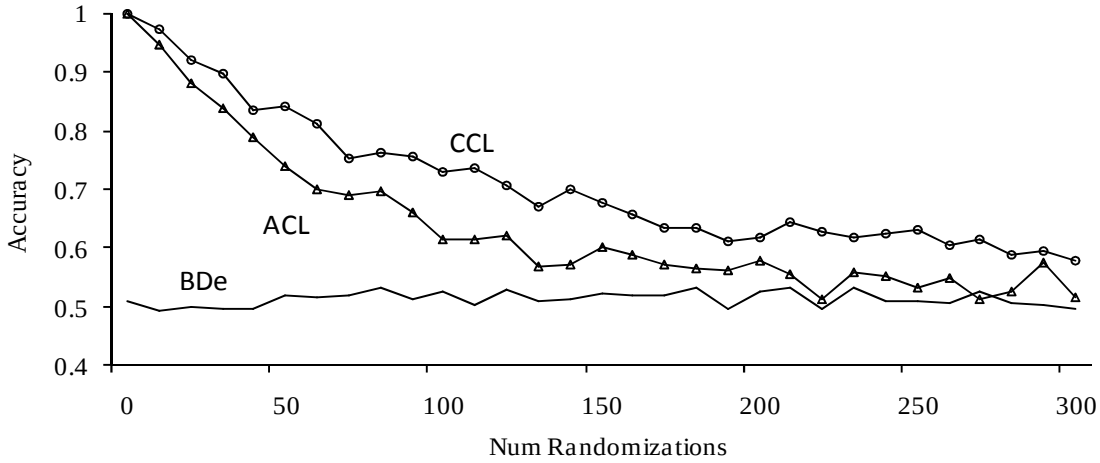

Figure 7. Results of datasets with data points drawn from BNs with randomly modified datasets.

data. This can be seen as BDe's classification accuracy of around 50% throughout the results in Figure 6.

However, at this difference in NMIS scores, both the ACL and CCL scores instead choose the $C \rightarrow H$ link, which can been seen in on the far right of the graph in Figure 6 where both scores have 100% classification accuracy. Thus, NMISGen was capable of selecting a parameterization for data-generating BNs that resulted in data easily classifiable via class-discriminatively trained BNs, but not generatively-trained BNs.

However, the ability for NMISGen to select for parameters with precise correlational strengths also allowed data to be generated with varying differences between the correlational strength of the $C \rightarrow H$ link. As seen in the results, the classification performance of the ACL classifier was found to be higher than the classification performance of the CCL score as the NMIS difference decreased. Such an observation would only be possible if the method for generating the simulated data had the ability to precisely control the correlations in the generated data. NMISGen can do this.

4.2 ADDING NOISE TO GENERATED DATA

It is often desirable to add noise to a dataset. A straight forward method of adding noise is to generate a dataset and randomly change some of the values in the dataset. This can have unforeseen effects on the correlations in the underlying data. For instance, it may globally decrease the correlation among all RVs (i.e., it may whiten the data). Further, the relationships among RVs in the data no longer match the parameters in the generative model used to construct the data.

Noise can also be added by making random changes to the NMIS for a generative BN. This allows each sampled data point to be generated from a different, but known and controllable, distribution. Further, all correlations in the underlying data are accounted for and inadvertent whitening of the data is avoided.

Each class of data is represented by a base-line data-generating BN. However, data points for each classes' datasets are not drawn from this base-line model. Instead, for each data point to be generated, some of the base-line model's NMIS scores are randomly changed. The number and magnitude of the changes determines how much noise is added to each data point. This allows for a precise control over how much each of the data points in the same class differ from each other.

BNs are then learned for each class of data as they were previously, and a classifier based on the learned BNs is constructed. Figure 7 shows the results. As the number of NMISs are changed (towards the right hand side of the graph), increasing amounts of noise are present in the data points of a same class. This makes classification more difficult for both the ACL and CCL classifier (the BDe classifier still performs badly in all cases). However, given NMISGen's ability to subtly control for the amount of variance in the data points in the same class without adversely effecting the overall level of correlation in the data (something that CCL was less capable of dealing with given as demonstrated by the results in Figure 6), the difference between ACL's and CCL's decline in accuracy can be observed. In this case, CCL was shown to be more robust to the noise introduced in the data.

5 CONCLUSIONS

Simulated data is often used to validate unsupervised learning tasks such as Bayesian network (BN) structure search. However, little attention has been given to methods that can generate interesting classes of data. In fact, we are not familiar with any other published work that describes methods for generating simulated data with precisely controlled RV correlations.

We have proposed quantifying the pairwise correlation among parent and child RVs in a BN with a normalized mutual information score (NMIS). Given a BN topology annotated with NMISs, we have demonstrated how the parameters for the BN can be determined such that strength of correlation in data generated from the BNs will be constrained by the NMISs.

We have demonstrated how our methods can be used to generate data that is easily classified given BNs learned with class-discriminative scores, but not with generative scores. Further, as our methods allow for the precise control over the strength of correlations, we have shown how one class-discriminative score learns structures that perform better (in terms of classification accuracy) than an another. We have also proposed that adding noise into simulated data should be done by modifying the NMIS constraints for the data-generating BN as opposed to simply randomizing values in already-generated datasets—a process that will likely whiten the data.

There are several limitations to our approach. Most prominently, we individually label each link the BN topology with a single NMIS. However, the relationship between a child node and its parents in a BN can be much more complex than the sum of pairwise correlations. One of the difficulties of learning BN structures is that the pairwise correlations between the child and its parents may be very small, but the correlation between the child and the joint-distribution of the parents may be very large, e.g., the XOR relationship. An improvement to our technique allowing for NMIS to be associated with more than just pairwise relationships could be quite beneficial.

We also discussed application of our method to dynamic Bayesian networks (DBNs) (which explicitly represent time). Our approach requires that generative DBNs do not have any "temporal cycles" in the topology. This is not a constraint normally placed on DBNs and future work extending our methods to such DBNs seems promising.

For any child/parent pair, our algorithm is guaranteed to converge on a solution if it exists. However, it is not guaranteed to find a solution for an entire topology of NMIS constraints. We have found in practice that if a graph contains a large number of strong correlations—particularly many strong correlations in the same family—our methods will not always find a solution. Reducing the strength of the correlations always resolved the issue, however, future work getting around this limitation is certainly warranted.

Even given these limitations, we have found our technique is capable of creating a large class of interesting simulated data that could be used in many types of experiments.

Acknowledgements

This project was partially funded through grant DA012852 from the NIDA, NIH, from NIMH grant number 1R01MH076282-01 as part of the NSF/NIH Collaborative Research in Computational Neuroscience Program and The MIND Institute, DOE Grant DE-FG02-99ER62764.

References

Burge, J. (2007). *Learning Bayesian Networks from Hierarchically Related Data with a Neuroimaging Application.* University of New Mexico, Albuquerque, NM.

Burge, J., & Lane, T. (2005). *Class-Discriminative Dynamic Bayesian Networks.* Paper presented at the ICML, Bonn, Germany.

Cover, T. M., & Thomas, J. A. (1991). *Elements of Information Theory*: Wiley-Interscience.

Grossman, D., & Domingos, P. (2004). *Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood.* Paper presented at the International Conference on Machine Learning.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*(3), 197-243.

Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning.* University Of California, Berkeley.

Pearl, J. (1986). *Fusion, Propagation, and Structuring in Belief Networks*. Paper presented at the Artificial Intelligence.