A COMPARISON OF STATE-OF-THE-ART ALGORITHMS FOR LEARNING BAYESIAN NETWORK STRUCTURE FROM CONTINUOUS DATA

By

Lawrence D. Fu

Masters Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Biomedical Informatics

December, 2005

Nashville, Tennessee

Approved:	Date:
Dr. Ioannis Tsamardinos	12/2/2005
Dr. Constantin Aliferis	11/30/2005
Dr. Douglas Hardin	12/1/2005

ACKNOWLEDGEMENTS

I would like to thank the members of my committee for their guidance, patience, and enthusiasm throughout the course of this work. I also thank Alex Statnikov and Laura Brown for their assistance with general questions. I am grateful to Francis Bach, Alexander Hartemink, Stefano Monti, and Harald Steck for providing helpful feedback concerning their algorithms and methods. Finally, this work was supported by a training grant from the National Library of Medicine (T15 LM 007450-03).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Bayesian networks have become a popular tool in biomedical research (Friedman, 2004; Lucas et al., 2004). Researchers utilize them for tasks such as reasoning with uncertainty, classification, and causal discovery in areas ranging from clinical applications to basic science research. Examples of clinical uses include decision support systems and medical diagnosis. Specific tasks include aiding in tumor classification (Antal et al., 2003) or radiology (Burnside et al., 2004). In basic science research, Bayesian networks are widely used in the study of gene regulatory networks (Friedman et al., 2000; Pe'er et al., 2001; Helman et al., 2004) and other applications such as protein secondary structure prediction (Robles et al., 2004). Furthermore, Bayesian networks are capable of suggesting manipulation experiments in the study of gene networks (Yoo and Cooper, 2004; Pournara and Wernisch, 2004) as well as detecting disease outbreak (Cooper et al., 2004). Learning the Bayesian network structure of a system or environment gives researchers useful information about the causal relationships among variables.

Despite the diverse applicability of Bayesian networks, the fact that most Bayesian network structure learning algorithms require discrete data is a limitation since biomedical and biological data routinely are continuous. There are three general approaches to learning network structure with continuous data.

- Prediscretization methods: the data is discretized prior to application of the learning algorithm. Due to the pertinence of these methods to most machine learning algorithms, a great deal of research has focused on this area (Liu et al., 2002). Unfortunately, many of these methods focus on supervised classification and are not applicable to Bayesian network structure learning, but we can still use those that are not tailored to classification.
- Integrated methods: the learning of the variable discretization and structure can be integrated (Friedman and Goldszmidt, 1996; Monti, 1999; Steck and Jaakkola, 2003). These methods follow greedy, iterative procedures by starting with an initial discretization, learning a model

based on the discretized data, and re-discretizing given the learned model. These steps repeat until a termination condition is met. The approaches output a discretization of the input variables.

- Direct methods: learning can be done directly with continuous data without committing to a specific discretization of the variables (Bach and Jordan, 2003; Margaritis, 2005; Imoto et al., 2003).

Typically, studies use prediscretization techniques, such as frequency-based partitions. The main advantage of this approach is efficiency. Discretization is performed initially before applying a discrete learning algorithm. Another advantage is the easy interpretation of data (Dougherty et al., 1995). For example, if a researcher deems that the most sensible discretization of a gene expression measurement is three levels, this could be interpreted as three states: low, average, and elevated expression level. Also, if it is believed that the variables naturally are discrete but the data is continuous due to noisy observations, then discretization appears justified (Hartemink, 2001).

On the other hand, discretization unavoidably results in loss of information (Friedman et al., 2000). All variation within an interval is discarded. Poor decisions about the size or number of discretization levels can disadvantage the learning algorithm since dependencies between variables may become undetectable. Margaritis (Margaritis, 2005) provides an example shown in Figure I-1.

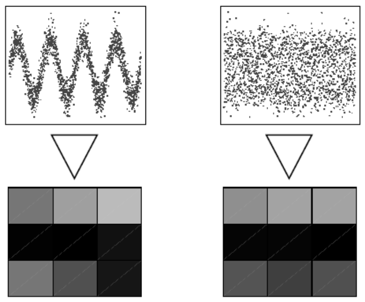

Figure I-1: An example of how discretization can obscure the dependencies and independencies between variables (Margaritis, 2005). The top row shows two dependent variables on the left and two independent variables on the right. When they are discretized into three values, the histograms look similar.

In the first row, the left box depicts two dependent variables with each axis representing a variable. The right box depicts two independent variables. The second row displays the results of discretizing each variable into three equally sized intervals. Darkly shaded boxes signify highly populated cells. The histograms are similar despite the inital fact that one case was strongly dependent while the other was independent. A learning algorithm with access only to the discretized data will have difficulties as a result of discretization. Thus, by neglecting to adequately address the ramifications of discretization, researchers can lose vital information such as interactions and dependencies between variables and hinder structure learning.

The alternative methods pose other strengths and weaknesses. Integrated methods do not necessarily commit to a poor initial discretization. If the initial discretization is poor, it will be modified in a subsequent iteration. Also, integrated methods are tailored to Bayesian network learning. They consider the interaction among variables during discretization while the structure is fixed. Their main disadvantage is that they are computationally intensive compared to the prediscretization methods which only perform discretization and structure learning once. The integrated methods perform them multiple times and consequently require more time. As for the direct methods, they have the advantage of not discarding any information. On the other hand, they can be computationally intensive. Some methods must make distributional assumptions to ensure computational feasibility.

This research compared the three approaches to ascertain the relative strengths and weaknesses of each and to quantify the impact on network learning. The focus was Bayesian network structure learning from continuous data. For the remainder of this thesis, learning will refer specifically to this task. While there has been a comparison of prediscretization techniques (Liu et al., 2002), there has not yet been an evaluation of all approaches in one unified study. For example, studies of the integrated methods usually make comparisons to prediscretization techniques. However, there has not been an evaluation of all integrated methods. The same holds true for the approaches learning directly with continuous data.

The evaluation involved a representative sample of methods from the three classes. Their performances were compared on two types of data. The first type was continuous data simulated from discrete Bayesian networks. These data sets were noisy observations of originally discrete variables. The second type was real data without known structures. For large simulated data sets, the discretization-based methods yielded the highest quality structures. With small simulated data sets or real data, a direct method was best. In terms of efficiency, methods from the prediscretization and direct categories were best depending on the metric used. The integrated methods required the most computational time.

Another goal of the evaluation was to determine if the integrated methods provided improvements in quality over the prediscretization approaches to justify the decreased efficiency. When considering edge orientation errors, the integrated methods did not induce more accurate structures. However, when only undirected edges were considered, the integrated methods yielded more accurate structures.

The rest of the thesis is organized as follows. Chapter II provides background information necessary for the remaining discussion. Chapter III explains the specific algorithms evaluated. Chapter IV reviews previous relevant work that is not included in our evaluation. Chapter V outlines the experimental results, and Chapter VI presents conclusions.

CHAPTER II

BACKGROUND

This chapter provides background information necessary for subsequent discussion. A brief review of Bayesian networks precedes a formal definition of discretization.

II.1 Bayesian Networks

A simple directed graph is a pair (V, E). V is a set of nodes or vertices, and E is the set of all directed edges or arcs connecting the elements of V. Self edges or multiple edges are not allowed. If there is a directed edge from node X to node Y, X is the parent of Y. Conversely, Y is the child or descendant of X. A directed acyclic graph (DAG) is a directed graph where there does not exist a path from any node to itself. A partially directed acyclic graph (PDAG) is a simple graph containing directed and undirected edges without cycles.

Let P be a joint probability distribution of the random variables in some set V, and $\mathbb{G} = (V, E)$ be a DAG. ($\mathbb{G}$, P) is a Bayesian network if it satisfies the Markov condition (Spirtes et al., 2001; Neapolitan, 2003). The Markov condition holds if for each variable $X \in V$, X is conditionally independent of its nondescendants given the set of its parents. Two variables X and Y are conditionally independent if $P(X|Y) = P(X)$ when $P(Y) > 0$. Moreover, a Bayesian network is a graphical structure modelling the probabilistic relationships between variables.

The two major approaches to Bayesian network structure learning from data are search-and-score methods and constraint-based learning. Search-and-score methods search the space of possible networks with a scoring metric that measures the fit of the structure to the data. The final structure is the one with the highest score. Typically, heuristic search techniques make the search feasible. Constraint-based methods determine conditional independencies between variables from the probability distributions of the data. The final structure is consistent with the independencies.

II.2 Discretization Policy

The discretization sequence of a variable is a vector of thresholds dividing the range of values into a set of mutually exclusive and exhaustive intervals. A discretization policy is the set of discretization sequences for all variables. Suppose we have a data set D with N instances and n variables denoted as $X = (X_1, \dots, X_n)$. Let us represent the value of variable X_i in the l^{th} instance as $x_i^{(l)}$ and the set of instances $(x_i^{(1)}, \dots, x_i^{(l)})$ as $x_i^{(1,l)}$. Then, the discretization sequence of a variable X_i is $\Lambda_i = \{t_{i,1}, \dots, t_{i,r_i-1}\}$ or an increasing set of real-valued thresholds (i.e. $t_{i,1} < t_{i,2} < ... < t_{i,r_i-1}$) where r_i is the number of intervals in variable X_i. As a result, function f_{Λ_i} performs the mapping $f_{\Lambda_i} : X_i \mapsto Y_i$ where Y_i is the vector of discretized variables defined as follows:

$$f_{\Lambda_i}(x_i) = \begin{cases} 0 \text{ if } x_i < t_{i,1} \\ k \text{ if } t_{i,k} \leq x_i < t_{i,k+1} \quad \text{for} \quad 1 \leq k < r_i \\ r_i \text{ if } t_{i,r_i-1} \leq x_i \end{cases}$$

$Y = (Y_1, \dots, Y_n)$ is the resulting vector of discretized variables, and the discretization policy Λ is $(\Lambda_1, ..., \Lambda_n)$. If variable X_i is initially discrete, no discretization is needed and $Y_i = X_i$.

CHAPTER III

DESCRIPTION OF ALGORITHMS INCLUDED IN THE STUDY

This chapter provides detailed explanations of the methods included in the experimental evaluation, which contains a representative sample of approaches from the three general categories. The discussion is organized as follows: prediscretization methods, integrated, and direct methods.

III.1 Prediscretization methods

These methods discretize data prior to structure learning. They are considered unsupervised techniques since they do not consider class labels of instances during discretization. Thus, any unsupervised technique is suitable.

III.1.1 Equal-frequency and equal-width binning

Equal-width binning divides the range of values for each variable into k equally sized intervals, where k is pre-defined. Arbitrary values of k are usually chosen, but Margaritis (Margaritis, 2005) discusses other methods for determining values of k. *Equal-frequency binning* assigns an equal number of data instances to each of the k intervals. For example, assume that we are discretizing height within a data set of 20 patients where the maximum height is 6 feet and the minimum is 5 feet. If $k = 2$, equal-width binning assigns patients shorter than 5.5 feet to one interval while assigning the rest to another interval. On the other hand, equal-frequency binning assigns the 10 shortest patients (i.e. the shortest 50%) to one level while assigning the rest to another level.

III.1.2 Method by Hartemink

Hartemink's approach (Hartemink, 2001) initially discretizes all variables such that each value is in a separate level. The method iterates two loops. First, the outer loop counts down from the initial number of intervals to one as levels merge. For each variable, the inner loop merges the neighboring intervals resulting in the smallest decrease in total mutual information. The total mutual information score, TMI(n), is defined for n discretization levels as the sum of the pairwise mutual information between all pairs of variables when each has been discretized into n discretization levels. The algorithm terminates when a single level remains for all variables. As n decreases, TMI(n) remains at approximately the same level until decreasing rapidly as n approaches one. The final number of levels is manually selected by trading off a relatively high value for mutual information with a relatively low number of levels. Values such as 3 or 4 are typically chosen.

III.2 Integrated Methods

The integrated approaches (Friedman and Goldszmidt, 1996; Monti, 1999; Steck and Jaakkola, 2003) employ a greedy iterative search by alternating between structure learning and discretization. Monti (Monti, 1999) contains pseudo-code for the framework:

Algorithm 1 Learn Hybrid BN

```
1: procedure LearnHybridBN($D$, $\Lambda^0$)
   Input:  data $D$; initial discretization $\Lambda^0$
   Output: structure $G^{new}$; discretization policy $\Lambda$
2:     $\Lambda \leftarrow \Lambda^0$
3:     $G^{new} \leftarrow LearnDiscreteBN(D, \Lambda)$
4:     repeat
5:         $G^{old} \leftarrow G^{new}$
6:         $\Lambda \leftarrow LearnDP(D, G^{old}, \Lambda)$
7:         $G^{new} \leftarrow LearnDiscreteBN(D, \Lambda)$
8:     until termination criteria met
9:     return $G^{new}$ and$\Lambda$
10: end procedure
```

The methods require an initial discretization, and equal-frequency binning is typically used. They hold the discretization fixed while learning the structure and then hold the structure fixed while re-discretizing. The two learning procedures repeat until a termination condition is met, which is different for each method. A final discretization policy is output along with the structure. The function *LearnDiscreteBN* can be any learning algorithm intended for discrete data (Heckerman et al., 1995). The function *LearnDP* discretizes with a fixed structure as follows:

Algorithm 2 Learn Discretization Policy

```
1: procedure LearnDP($G$, $\Lambda^0$,$D$)
   Input:  structure $G$; initial discretization $\Lambda^0$; data $D$;
   Output: new discretization policy $\Lambda$
2:     $\Lambda \leftarrow \Lambda^0$
3:     Push all continuous variables onto queue $Q$
4:     while $Q$ is not empty do
5:         $X_i \leftarrow$ Pop($Queue$)
6:         $\Lambda_i^{new} \leftarrow LearnVariableDP(X_i, G, \Lambda, D)$
7:         if $S(\Lambda_i^{new}; \Lambda, D, G) > S(\Lambda_i; \Lambda, D, G)$ then
8:             $\Lambda[i] \leftarrow \Lambda_i^{new}$
9:             $Q \leftarrow$ Push($MarkovBlanket(X_i)$)
10:        end if
11:    end while
12:    return $\Lambda$
13: end procedure
```

Changing the discretization of a variable affects the discretization of all variables within its markov blanket, i.e. its parents, children, and all parents of its children. A queue, which initially contains all continuous variables, is maintained to ensure that variables affected by the new discretization are re-discretized. When a variable is popped from the queue, it is discretized in the function $LearnVariableDP$ according to a scoring function that ranks thresholds, and the resulting discretizations may be scored by another scoring function. The methods differ in their implementations of the scoring functions and termination criteria, and the specifics will be discussed in the following sections III.2.1, III.2.2, and III.2.3.

III.2.1 Method by Friedman

Friedman's algorithm (Friedman and Goldszmidt, 1996) is motivated by the Minimal Description Length (MDL) principle (Lam and Bacchus, 1994), which balances the complexity of a network with how well it models the data. Friedman adapts the score to include the description length of the information required to recover the original data from the discretized data. Within the *LearnVariableDP* function, discretization of a variable uses a greedy search starting with an empty threshold list and considers all midpoints. Since different discretizations only affect certain terms of the description length, computing a local score, DL_{local}, rather than the full score reduces calculations. The choice of thresholds affects the local score terms related to mutual information. Consequently, thresholds are scored by maximizing the information gain of a candidate threshold t by considering variable i, current set of thresholds Λ_i, structure G, and data D:

$$\begin{aligned} Gain(t,i,\Lambda,G,D) &= I(f_{\Lambda_i \cdot t}(X_i);Y_{\Pi_i}) \quad + \textstyle\sum_{j\in Ch_i} I(Y_j; f_{\Lambda\cdot t}(Y_{\Pi_j})) \\ &\quad -I(f_{\Lambda_i}(X_i);Y_{\Pi_i}) \quad - \textstyle\sum_{j\in Ch_i} I(Y_j; f_{\Lambda}(Y_{\Pi_j})) \end{aligned}$$

where $f_{\Lambda_i \cdot t}(X_i)$ is variable i discretized with the thresholds of Λ_i and candidate threshold t. The parent set of variable i is denoted by Π_i, and Y_i and Y_{Π_i} are respectively the previously discretized variable i and parents of variable i. The term $I(f_{\Lambda_i \cdot t}(X_i);Y_{\Pi_i})$ is the mutual information between $f_{\Lambda_i \cdot t}(X_i)$ and Y_{Π_i}. Also, Ch_i is the set of children for variable i, and the term $f_{\Lambda\cdot t}(Y_{\Pi_j})$ represents the set of discretized parents for variable j, which includes variable i discretized according to the thresholds of Λ_i and threshold t. After adding a threshold, the gain terms from the same interval as the chosen threshold need to be recomputed. Thresholds are added until the local score, DL_{local}, increases.

After a variable discretization is output from *LearnVariableDP*, the *LearnDP* function accepts it if the description length score, DL, is less than the previous score with the old discretization. The score of a policy Λ with structure G and data D is:

$$DL(\Lambda,G,D) = DL_{net}(Y,G) + DL_{\Lambda}(\Lambda) - N\sum_i I(Y_i;Y_{\Pi_i})$$

where Y is the set of variables X discretized by Λ, and Y_{Π_i} is the discretized set of variable i's parents. The description length of a network, DL_{net}, for discretized variables Y and structure G:

$$DL_{net}(Y,G) = \sum_i (\log \|Y_i\| + (1+|Y_{\Pi_i}|)\log n) + \frac{\log N}{2}\sum_i \|Y_{\Pi_i}\|(\|Y_i\|-1)$$

where $\|Y_i\|$ and $\|Y_{\Pi_i}\|$ are the cardinalities of Y_i and Y_{Π_i}, and $|Y_{\Pi_i}|$ is the number of parents of Y_i.

The description length of a discretization policy, $DL_{\Lambda}(\Lambda)$ is

$$DL_{\Lambda}(\Lambda) = \sum_i (\|X_i\| - 1) H(\frac{k_i - 1}{\|X_i\| - 1})$$

where $\|X_i\|$ is the cardinality of X_i, k_i is the number of intervals in Y_i, and $H(p) = -p \log p - (1 - p) \log(1 - p)$.

After the *LearnDP* function stops when the queue of variables is empty, the algorithm proceeds from the discretization phase of learning to structure learning. The algorithm iterates between the two phases until terminating when the score increases.

III.2.2 Method by Monti

Instead of using an MDL-motivated scoring function as Friedman does, Monti's method (Monti, 1999) utilizes a Bayesian-based metric. Monti augments the traditional Bayesian scoring metric to score a discretization policy with respect to a given data set and structure. The same scoring function is used for thresholds and the discretization policy.

Discretization of a variable starts with an empty threshold list. Thresholds are added in a greedy fashion until the scoring function does not increase with a new threshold. To derive the function, Monti assumes an underlying discrete mechanism governing the behavior of the observed continuous variables. In this case, each continuous variable X_i is independent of all other variables given the value of its corresponding discrete variable Y_i. This problem formulation results in searching for the discretization policy that maximizes the posterior probability $p\,(\Lambda|D)$. Assuming a uniform prior $p(\Lambda)$ means that maximizing the likelihood of the data given the discretization, $p\,(D|\Lambda)$, will find the discretization we want. For a given variable, $p\,(x_i|\Lambda)$ factors into:

$$p\,(x_i|\Lambda) = p\,(x_i|y_i, \Lambda) p\,(y_i|y_{\Pi_i}, \Lambda)$$

where a lower case variable represents the vector of values for the variable, and y_i is the discretized version of x_i. By computing the log-likelihood, the score for the discretization becomes a sum of the discrete component, $\log p\,(y_i|y_{\Pi_i}, \Lambda)$, and continuous component, $\log p\,(x_i|y_i, \Lambda)$. The traditional

Bayesian scoring metric for discrete data can be used to calculate the discrete component for variable i with discretization policy Λ and data D:

$$\begin{aligned} S_d(i,\Lambda,D) &\equiv \log p\,(y_i|y_{\Pi_i},\Lambda) \\ &= \sum_{j=1}^{q_i} \log \tfrac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} + \sum_{k=1}^{r_i} \log \tfrac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})} \end{aligned}$$

where q_i is the number of values that the discretized parents of variable i can take, r_i is the number of values that the discretized variable i can take, N_{ijk} is the number of cases in the data where variable i takes the value k while its parents take the joint value j, $N_{ij} = \sum_k N_{ijk}$, $\Gamma(\cdot)$ is the Gamma function, α_{ijk} is a hyperparameter of the Dirichlet distribution, and $\alpha_{ij} = \sum_k \alpha_{ijk}$.

The continuous component is the conditional density of the continuous data given the discrete data. The computation is simplified since the distribution of variable i in the l^{th} instance, $x_i^{(l)}$, only depends on $y_i^{(l)}$. By assuming it is a uniform distribution, we get:

$$\begin{aligned} S_c(i,\Lambda,D) &\equiv \log p\,(x_i|y_i,\Lambda) \\ &= \sum_{l=1}^{N} \log p\,(x_i^{(l)}|\,y_i^{(l)}, x_i^{(1,\,l-1)}, y_i^{(1,\,l-1)}\Lambda) \\ &= \sum_{l=1}^{N} \log p\,(x_i^{(l)}|\,y_i^{(l)},\Lambda) \\ &= \sum_{k=1}^{r_i} N_k \log \tfrac{1}{\Delta_k} \end{aligned}$$

where $x_i^{(1,\,l-1)}$ is the first $l-1$ cases of x_i, r_i is the number of intervals, Δ_k is the width of the k-th interval, and N_k is the number of instances where the variable takes value k in the data.

Since the discrete component is affected by the discretization of the parent variables, the algorithm maximizes the score:

$$\begin{aligned} S_\Lambda(i,\Lambda,D,G) &= S_c(i,\Lambda_i,D) + S_d(i,\Lambda,D) \\ &+ \sum_{j\in \mathrm{Ch}_i} S_d(j,\Lambda,D). \end{aligned}$$

where Ch_i is the set of variable i's children. The last term derives from the fact that changing the discretization of a variable requires re-computing the discretization of all of the variable's children.

This scoring function is used within the *LearnVariableDP* and *LearnDP* functions. The algorithm terminates when the learned graph does not change between iterations.

III.2.3 Method by Steck

Similarly to Monti's algorithm, Steck's method (Steck and Jaakkola, 2003) derives a scoring function for the marginal likelihood $p\,(D|\Lambda, G)$ of the data given the discretization policy Λ and structure G. Steck's approach considers the data in a sequential manner with steps and transforms the likelihood into

$$p\,(D|\Lambda, G) = \prod_{l=1}^{N} p\,(x^{(l)}|D^{(1,\,l-1)}, \Lambda, G)$$

where $x^{(l)}$ is the instantiation of X at step l and $D^{(1,\,l-1)} = (x^{(1)}, \ldots, x^{(l-2)}, x^{(l-1)})$ represents the data points encountered prior to step l along the sequence[1]. Since each continuous value only discretizes to a single value, each term factors into

$$p\,(x^{(l)}|D^{(l-1)}, \Lambda, G) = p\,(x^{(l)}|y^{(l)}, \Lambda)\, p\,(y^{(l)}|D^{(l-1)}, G, \Lambda).$$

Assuming that any two continuous variables are independent conditioned on their corresponding discrete variables, the term $p\,(x^{(l)}|y^{(l)}, \Lambda)$ can be additionally factored into

$$p\,(x^{(l)}|y^{(l)}, \Lambda) = \prod_{i=1}^{n} p\,(x_i^{(l)}|y^{(l)}, \Lambda_i).$$

The conditional densities $p\ (x_i^{(l)}|y^{(l)}, \Lambda_i)$ are implicitly estimated by using the finest grid implied by the data, $\Omega = (\Omega_1, \ldots, \Omega_n)$. The grid discretizes each variable i by a set of thresholds $(\omega_{i,1}, \ldots, \omega_{i,N-1})$ such that each value is in a separate interval. The policy Ω defines a mapping from the original continuous variables to a new vector of discrete variables called Z (i.e. $f_\Omega : X \mapsto Z$). The algorithm requires the final discretization policy Λ to be a subset of the thresholds from the finest grid, which is not really a strict restriction since the thresholds of Ω can be any value between points. Also, the finest grid defines hypercubes for the n-dimensional vector X with volumes determined by the

[1]Steck originally used variable Y to designate the continuous variables. I have continued to use X to represent them for consistency with the previous sections.

widths of the intervals for each Ω_i.

Now, the densities $p\,(x_i^{(l)}|y^{(l)},\Lambda_i)$ can be factored efficiently via the finest grid:

$$p\,(x_i^{(l)}|y^{(l)},\Lambda_i,\Omega_i) = p\,(x_i^{(l)}|z_i^{(l)},\Omega_i)p\,(z_i^{(l)}|y^{(l)},\Lambda_i,\Omega_i)$$

Combining the previous equations yields:

$$p\,(D|\Lambda,G,\Omega) = p\,(D_\Lambda|G)\cdot\left(\prod_{l=1}^{N}\prod_{i=1}^{n}p\,(x_i^{(l)}|z_i^{(l)},\Omega_i)\right)\cdot\left(\prod_{l=1}^{N}\prod_{i=1}^{n}p\,(z_i^{(l)}|y^{(l)},\Lambda_i,\Omega_i)\right)$$

The first term is the likelihood of structure G with data D_Λ discretized by Λ and is easily computed for discrete Bayesian networks with the traditional Bayesian scoring metric(Heckerman et al., 1995). The second term can be ignored since it is independent of Λ and G. The third term simplifies by assuming that the probability mass predicted for $y^{(l)}$ is divided evenly among all hypercubes and that at least one hypercube is mapped to each y. Thus, the final version of the predictive scoring function is

$$\mathcal{L}_P(\Lambda,G) = \log p\,(D_\Lambda|G) - \sum_{i=1}^{n}\sum_{k=1}^{r_i}\log\Gamma(N_{k_i})$$

where r_i is the number of discretization intervals for X_i, and N_{k_i} is the number of instances where X_i takes the value k in the discretized data. This scoring function is used within the *LearnVariableDP* and *LearnDP* functions, and the algorithm terminates when the score decreases.

III.3 Direct Methods

The following approaches learn from continuous data without committing to a final discretization policy. They adapt the structure learning approaches, search-and-score and constraint-based methods, to handle continuous data. For the search-and-score strategy, the scoring metric is modified to score continuous data. For constraint-based methods, a conditional independence test suitable for continuous data is required.

III.3.1 Method by Bach

Bach's method (Bach and Jordan, 2003) uses a local greedy search with a new scoring function based on the MDL/BIC score (Lam and Bacchus, 1994), which penalizes the likelihood of a structure by $\frac{1}{2}\log N$ times the number of parameters. The maximum likelihood J_{ML} can be decomposed as $J_{ML} = \sum_i J_{ML}(i, \pi_i)$ where $J_{ML}(i, \pi_i) = -NI(x_i, x_{\pi_i})$. The term π_i is the set of parents of node i in the structure, and $I(x_i, x_{\pi_i})$ is the mutual information of node i and its parents. Bach's scoring function maps the data into high-dimensional feature spaces, calculates the mutual information of the feature variables, and uses this approximation to rank models during structure learning.

The mapping of the data uses Mercer kernels and treats all data as Gaussian in feature space. A Mercer kernel on a space $\mathcal{X}$ is a function $k(x, y)$ from $\mathcal{X}^2$ to $\mathbb{R}$ such that for any set of points $\{x^{(1)}, \ldots, x^{(N)}\}$ in $\mathcal{X}$, the $N \times N$ matrix K, defined by $K_{ij} = k(x_i, x_j)$, is positive semidefinite (Bach and Jordan, 2003). A Mercer kernel determines a space $\mathcal{F}$ and a map Φ from $\mathcal{X}$ to $\mathcal{F}$ such that $k(x, y)$ is the dot product in $\mathcal{F}$ of $\Phi(x)$ and $\Phi(y)$.

If there are n random variables $X_1, \ldots, X_n$ with spaces $\mathcal{X}_1, \ldots, \mathcal{X}_n$, then assigning a Mercer kernel k_i to each $\mathcal{X}_i$ leads to a feature space $\mathcal{F}_i$ and feature map Φ_i. The vector of feature images $\phi = (\phi_1, \ldots, \phi_n) \triangleq (\Phi_1(x_1), \ldots, \Phi_n(x_n))$ has a covariance matrix C where block C_{ij} is the covariance matrix between ϕ_i and ϕ_j. Let $\phi^G = (\phi_1^G, \ldots, \phi_n^G)$ be a jointly Gaussian vector with the same mean and covariance as ϕ. The sample covariance matrix of ϕ can be calculated using the kernel trick, and the correlation matrix R of ϕ_i^G is approximated with an incomplete Cholesky decomposition.

The vector ϕ^G is used to calculate the KGV-mutual information[2], which is the mutual information between $\phi_1^G, \ldots, \phi_n^G$:

$$I^K(x_1, \ldots, x_n) = -\frac{1}{2}\log\frac{|R_{1n}|}{|R_{11}|\cdots|R_{nn}|}$$

where $|R|$ is the determinant of the specified correlation matrix , and the subscripts of R denote the submatrix indexed by the values shown. Bach's scoring function incorporates KGV into the

[2]KGV stands for kernel generalized variance.

MDL/BIC score by defining the objective function J for a structure G as $J(G) = \sum_i J(i, \pi_i)$ where

$$J(i, \pi_i) = -NI(x_i, x_{\pi_i}) = \frac{N}{2} \log \frac{|R_{\{i\}\cup\pi_i,\{i\}\cup\pi_i}|}{|R_{\pi_i,\pi_i}||R_{i,i}|} + \frac{d_{\pi_i} d_i}{2} \log N.$$

In the equation, d_i is the dimension of the Gaussian variable, and $d_{\pi_i} = \sum_{j\in\pi_i(G)} d_j$. The KGV of the feature variables indirectly approximates the mutual information of the original variables. Since the likelihood of a structure can be decomposed into terms based on the mutual information of the original variables, KGV is used to rank models during structure learning.

III.3.2 Constraint-based method

This approach involves a conditional independence test for continuous data to be used with a constraint-based learning algorithm. In this case, the PC algorithm (Spirtes et al., 2001), which is the most popular constraint-based learning algorithm, will be used with Fisher's Z test. The PC algorithm starts with a complete undirected graph over the set of all variables. All zero order independence tests are performed, and edges are removed when independence is concluded. Independence tests of increasing order are performed to further remove edges. Once no more edges can be removed, the algorithm orients the edges. Collider nodes are oriented first, and then several other orientation steps are followed (Spirtes et al., 2001).

Fisher's Z concludes the independence of two variables X_1 and X_2 given a set of variables S if the partial correlation coefficient ρ is zero. Fisher's Z (Neapolitan, 2003) is calculated by

$$Z = \frac{1}{2}\sqrt{N - |S| - 3}\left(\ln \frac{1+\rho}{1-\rho}\right)$$

where N is the sample size, $|S|$ is the number of variables in the conditioning set S, and ρ is the partial correlation coefficient of X_1 and X_2 given S. To determine if ρ is zero, we substitute zero for ρ in the equation and calculate Z. Using a table for the standard normal distribution, we can determine the probability that the standard normal is greater than our calculated Z value. If the probability is less than our significance level (i.e. .05), then we reject the hypothesis of conditional independence. In other words, we would say that X_1 and X_2 are conditionally dependent given S.

CHAPTER IV

RELATED WORK

This chapter reviews previous work in learning Bayesian network structure with continuous data that is not included in the evaluation. It is organized by the two categories: prediscretization methods and direct methods. All known integrated methods were included in the evaluation.

IV.1 Prediscretization

Research in discretization has spanned many fields including statistics (Scott, 1992), machine learning (Liu et al., 2002), and information theory (Gray and Neuhoff, 1998). Kozlov (Kozlov and Koller, 1997) discussed discretization tailored to Bayesian networks but focused on inference rather than structure learning. Since discretization techniques are independent of the learning algorithm, they have been used widely in tasks such as decision tree learning (Catlett, 1991; de Merckt, 1993; Kohavi and Sahami, 1996) and feature selection (Liu and Setiono, 1997). They are used as preprocessing steps prior to the induction algorithm and usually perform either merging or splitting of thresholds. Merging approaches (Kerber, 1992; Tay and Shen, 2002) initially set the discretization of each variable to all the midpoints of the data. In other words, each data point is in a separate interval. During each step, thresholds are removed to merge neighboring intervals and decrease the number of intervals. Splitting approaches start with an empty set of thresholds and all data points in a single interval. During each step, thresholds are added to split existing intervals and increase the number of intervals.

The methods can be also categorized as supervised or unsupervised. Supervised methods utilize class labels associated with data instances (Kerber, 1992; Tay and Shen, 2002; Fayyad and Irani, 1993; Ho and Scott, 1997; Kohavi and Sahami, 1996). They can be characterized further by the type of measure used to rank candidate thresholds. One possibility is using an information theoretic measure such as entropy. Entropy methods choose cut-points minimizing entropy in some manner. For example, Fayyad and Irani's method (Fayyad and Irani, 1993) starts with one interval and recursively creates binary partitions at each step. For each of the midpoints within the data, it calculates a quantity called class information entropy of the partition. This weights the entropy of each potential partition by its relative size. The boundary minimizing the sum of weighted entropies of the two new intervals is selected. Another possibility is measuring the association between the variable and class. Examples include zeta (Ho and Scott, 1997) and chi-square (Kerber, 1992; Tay and Shen, 2002).

Unfortunately, supervised techniques cannot be applied to the structure learning task since they require a discrete target. Even if we somehow used each variable as a target and discretized accordingly, each variable would be discretized differently with the various targets. It is not clear how to resolve or merge the different discretizations. Thus, methods are needed that specialize in Bayesian network structure learning.

The discretization techniques discussed so far have been univariate approaches since they consider a variable in isolation or in relationship to the class variable. Multivariate discretization approaches consider the interactions among variables (Kwedlo and Kretowski, 1999; Bay, 2001; Muhlenbach and Rakotomalala, 2002; Monti and Cooper, 1999). Another possible improvement, in terms of efficiency, has been work on discarding unpromising thresholds (Elomaa and Rousu, 2004).

IV.2 Direct Methods

Gaussian Bayesian networks (Geiger and Heckerman, 1994) are a search-and-score approach that assumes continuous data are sampled from a multivariate normal distribution. A limitation is that only linear dependencies can be learned from data. Imoto's method (Imoto et al., 2002) overcomes this restriction by using nonparametric regression models to detect nonlinear relationships among variables. This approach derives a scoring criterion called BNRC, which stands for Bayesian network and nonparametric regression criterion. Structure learning performs a greedy search that minimizes BNRC. The BNRC score uses B-splines as basis functions to model the conditional densities of a variable given its parents. The BNRC method was excluded from the study because it requires significant running time, which was mentioned by its authors as a limitation (Imoto et al., 2003).

Another type of Bayesian network for continuous data is a Conditional Gaussian network (Lauritzen and Wermuth, 1989), which has the restriction that continuous variables cannot have discrete descendants. Other approaches use complex conditional distributions (Friedman and Nachman, 2000; Hofmann and Tresp, 1996) or avoid the same distributional assumptions (Nachman et al., 2004).

Davies' method (Davies, 2002) is a greedy structure learning algorithm using tree-based density estimators to score potential network structures. The algorithm inputs an initial network structure and ranks all single arc additions and deletions with a relatively fast scoring function. Then it performs the changes in decreasing order of score and estimates the effects of the changes with a more accurate scoring function. If the new score is greater than the old score, it changes the parent set of the variable. The scoring functions grow tree structures that represent the conditional distribution of a variable given its parents and measure how well the tree models the data.

Margaritis (Margaritis, 2005) developed another conditional independence test for constraint-based methods. To determine if two variables are conditionally independent given a set of variables, it temporarily discretizes the variables by maximizing the posterior probability of dependence given the data. If the probability is greater than a certain value, the test concludes dependence. The test determines the probability of dependence by calculating the likelihoods of modelling the data as dependent with a joint multinomial distribution or as independent with two marginal multinomial distributions. Margaritis' method was not included in they study because of computational restrictions. Preliminary experiments with the method revealed that it would require considerable time. The time needed to compute a single conditional independence test was too long considering the number of tests needed to learn a full network with multiple variables and a large sample size.

CHAPTER V

EXPERIMENTAL EVALUATION

This work performed a comprehensive evaluation of algorithms for Bayesian network structure learning with continuous data. Comparisons metrics were based on the quality of the learned structures and the efficiency of the learning process. This study is unique in the range of algorithms examined. Most studies typically compare fewer methods without including all classes of approaches. The evaluation aimed to determine if a single method or class of methods would consistently outperform other methods over multiple data sets. If a superior method did not emerge, the study hoped to identify the data characteristics that would favor certain methods. For example, some methods may be better suited to learn with small sample sizes. Another consideration was whether the integrated methods provided sufficient performance benefits over binning methods to merit the additional computational costs.

The following eight algorithms were included in the study:

- Prediscretization methods
  - equal-width binning (k = 2, 3)
  - equal-frequency binning (k = 2, 3)
  - Hartemink's method
- Integrated methods
  - Monti's method (k = 2, 3)
  - Friedman's method (k = 2, 3)
  - Steck's method (k = 2, 3)
- Direct methods
  - Bach's method
  - Constraint-based method with independence test for continuous data

The parameter k specifies the initial number of discretization intervals. The methods requiring this parameter were run once for each value, and results are presented separately.

All methods required a Bayesian network structure learning algorithm. For the search-and-score approaches, a standard greedy search was used. After starting with an empty graph, greedy search performs one of the following three operations: add an edge, delete an edge, or re-orient an edge. It chooses the action that results in the greatest improvement according to a scoring function. The prediscretization and integrated methods maximized the BDeu score to guide the search during structure learning phases. Bach's method utilized a greedy search minimizing its scoring function. For the constraint-based approach, PC (Spirtes et al., 2001) was used since it is the most popular constraint-based learning algorithm.

The specifics of the algorithms were discussed in depth in Chapter III. When possible, the author's original Matlab code was used, which was the case with Hartemink and Bach's methods. For the constraint-based method, we used two different implementations of the PC algorithm: Tetrad 4.3.1 [1] and our lab's Matlab implementation. Tetrad implements a version of PC in Java along with significant modifications from the original PC algorithm. The details of the changes are unpublished. The rest of the algorithms were re-implemented in Matlab. In order to ensure that our implementations were equivalent to the original implementations, we replicated experimental results from the original papers. All experiments were run in Matlab on Pentium Xeons with 2.4 GHz processors and 2 GB RAM.

Since we wanted to compare the algorithms on equal footing, some algorithm modifications were made. Monti's method used a modified version of K2 (Cooper and Herskovits, 1992), which required a node ordering, to learn structure. In our implementation, we substituted greedy search since a node ordering is not always available and to maintain consistency with the other methods. Furthermore, in Monti's method, a new variable discretization is accepted if its score is greater than the score of the previous discretization. However, in our experiments, the discretization learning phase would not terminate unless we required the new discretization to increase the sum of discretization scores for all variables. We replicated Monti's experimental results to verify that the method was not severely handicapped by the change. These two changes resulted in our implementation varying slightly from Monti's original version, but it maintained the essence of the algorithm by interleaving discretization and structure learning.

Ideally, we would have preferred to use known Bayesian networks with continuous variables. Since these do no exist, we used simulated data from known discrete networks and real data. The experimental results are separated by the type of data used for learning.

[1] Tetrad 4.3.1 is available at http://www.phil.cmu.edu/projects/tetrad/.

V.1 Simulated Data

V.1.1 Data Sets

The simulated data used the following networks: Alarm (Beinlich et al., 1989), Child (Cowell et al., 1999), Hailfinder (Abramson et al., 1996), and Insurance (Binder et al., 1997). They were derived from real world decision support systems [2]:

Table V-1: Bayesian networks used to simulate data.

Network	**Num. of Variables**	**Num. of Edges**	**Max In-Degree**	**Max Out-Degree**	**Domain Range**
Alarm	37	46	4	5	2 - 4
Child	20	25	2	7	2 - 6
Hailfinder	56	66	4	16	2 - 11
Insurance	27	52	3	7	2 - 5

Networks with more variables were considered, but preliminary experiments demonstrated that some methods would require a prohibitive amount of running time.

Continuous data was simulated according to Monti's technique (Monti, 1999). First, discrete data was sampled from the known distributions of the networks. Then, each discrete value was converted into a continuous value by treating it as the mean to a Gaussian distribution with a pre-specified standard deviation. The value used for the standard deviation was .35, as selected by Monti. Section V.1.4 includes studies using alternate values for standard deviation. For each of the networks, five data sets of sizes 500, 1000, and 5000 were generated, and results were averaged over the five runs.

[2] Networks available at http://www.cs.huji.ac.il/labs/compbio/Repository

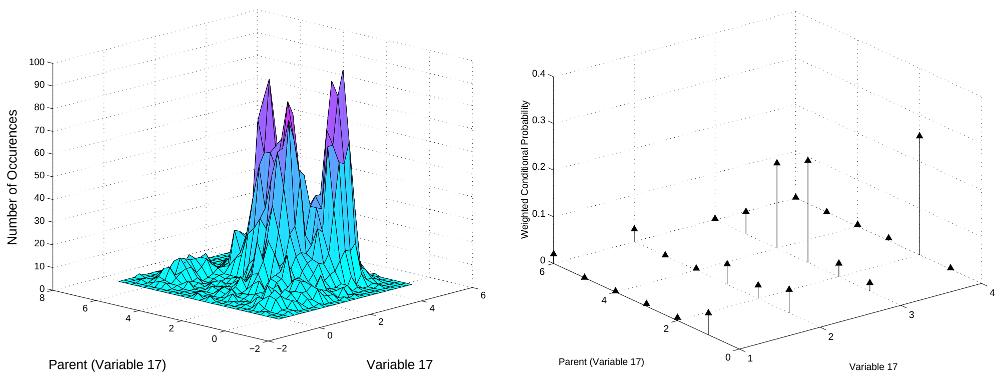

Figure V-1: Left: In the Child network, the data distribution of variable 17 and its parent is a mixture of Gaussians with heights proportional to the probability of a value given its parent times the probability that its parent takes that value. Right: For the original discrete values, the heights of the lines depict the probability of a value given its parent times the probability that its parent takes that value.

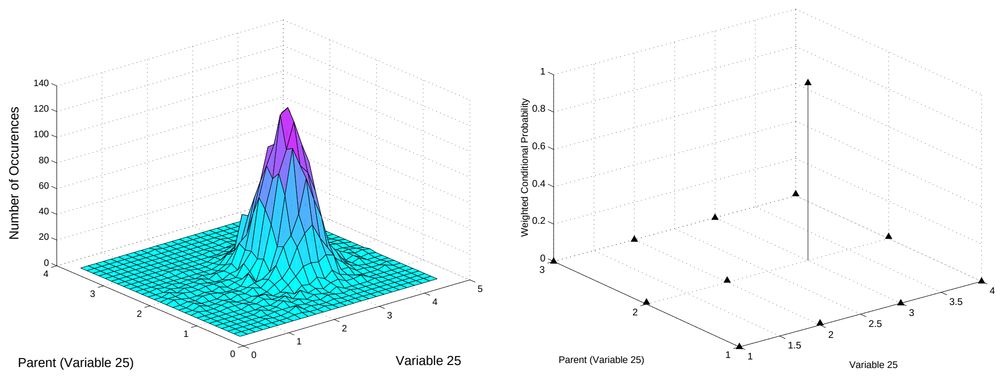

Figure V-2: Left: In the Alarm network, the data distribution of variable 25 and its parent is a mixture of Gaussians with heights proportional to the probability of a value given its parent times the probability that its parent takes that value. Right: For the original discrete values, the heights of the lines depict the probability of a value given its parent times the probability that its parent takes that value.

The left portions of figures V-1 and V-2 show distributions of variables given their parent from simulated data sets used in the evaluation. Darkly shaded regions represent frequent occurrences of data. The distributions are a mixture of Gaussians with heights proportional to the probability of a value given its parent times the probability that its parent takes that value. The right portions of the figures display the corresponding probabilities for the original discrete values. The conditional probability tables are included in Tables V-2 and V-3.

Table V-2: Conditional probability table of Child network for variable 17 given its parent.

	Parent of X_{17}					
X_{17}	1	2	3	4	5	6
1	0.40	0.02	0.02	0.01	0.01	0.40
2	0.43	0.09	0.16	0.02	0.03	0.53
3	0.15	0.09	0.80	0.95	0.95	0.05
4	0.02	0.80	0.02	0.02	0.01	0.02

Table V-3: Conditional probability table of Alarm network for variable 25 given its parent.

	Parent of X_{25}		
X_{25}	1	2	3
1	0.01	0.01	0.01
2	0.97	0.01	0.01
3	0.01	0.97	0.01
4	0.01	0.01	0.97

This simulation technique favors the binning methods, which univariately discretize variables in isolation, since the Gaussian behavior is still evident in the marginal distributions of variables, as shown by Figure V-3. As a result, the binning methods can detect the original discrete values without considering other variables. The simulation technique was used in order to make comparisons with Monti's work.

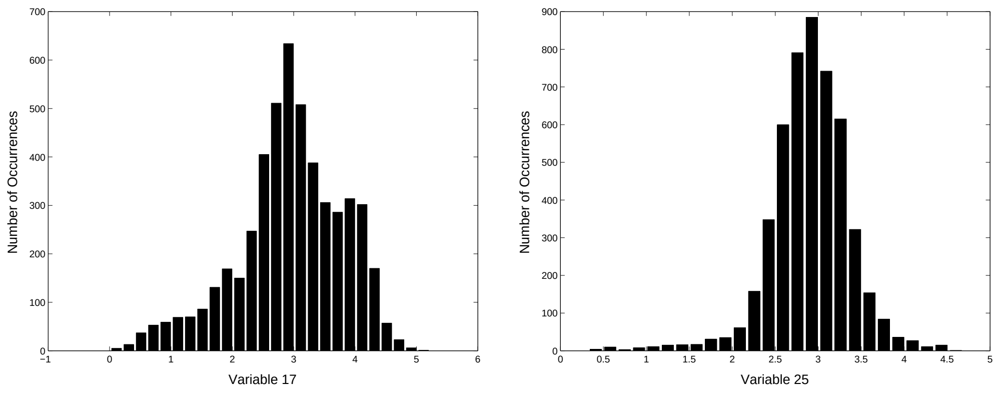

Figure V-3: Left: The marginal distribution of variable 17 from the Child network is Gaussian, which favors the binning . Right: The marginal distribution of variable 17 from the Alarm network.

V.1.2 Metrics for Comparison

Performance was measured as the quality of the learned structure and the efficiency of the learning procedure. For simulated data, the accuracy metric was Structural Hamming Distance (SHD) (Tsamardinos et al., 2004). Prior to scoring, the learned graph and true graph are converted into their equivalent PDAGs (Chickering, 1995) to avoid penalizing statistically indistinguishable structures. The SHD score penalizes a learned structure for every added, missing, and incorrectly oriented edge relative to the true structure. For example, if the learned graph has an undirected edge where the true graph has no edge, the SHD score is increased by 1 point. If the learned graph has an undirected edge where the true graph has a directed edge, the score is increased by 1 point. If the learned graph has an extra or missing directed edge, the score is increased by 2 points. The extra or missing unoriented edge counts for 1 point while the orientation error counts for another point. The best structure will have the lowest SHD score. For clarity, pseudo-code is included below.

Algorithm 3 SHD Algorithm

```
1: procedure SHD(Learned DAG pattern $H$, True DAG pattern $G$)
2:     shd = 0
3:     for every edge $E$ different in $H$ from $G$ do
4:         if $E$ is missing in $H$ then
5:             shd = shd + 1
6:         end if
7:         if $E$ is extra in $H$ then
8:             shd = shd + 1
9:         end if
10:        if $E$ is incorrectly oriented in $H$ then
           % This includes reversed edges and edges that are undirected in one graph and directed
           in the other.
11:            shd = shd + 1
12:        end if
13:    end for
14: end procedure
```

Algorithm efficiency was measured in two ways. The first metric was computational running time. The second metric was the number of statistical calls made during structure learning. These calls corresponded to calculations of a scoring function or a conditional independence test. For the discretization methods, the calls were calculations of the BDeu score. The number of calls was summed over all iterations of structure learning for the integrated methods. For Bach's method, the calls were calculations of Bach's scoring metric. For PC, the calls were conditional independence tests.

The number of calls provided an efficiency metric since each type of call required roughly the same amount of time. Furthermore, during structure learning, the algorithms spent most of their time performing these calls. Unlike running time, this metric is independent of the implementation or the hardware performing the experiments.

V.1.3 Results

The results were normalized to make a direct comparison of the relative performances of the algorithms. First, results were averaged over the number of runs for a given network and sample size. Then, the averaged results were divided by the averaged results of Bach's method for the same network and sample size. As a last step, the normalized results were averaged over all networks. Without normalization, the average would be skewed by large values from the difficult networks. Bach's method served as the basis of comparison since it had the best overall performance. Of the twelve combinations of networks and sample sizes, Bach's method yielded the lowest SHD four times, more than any other method. Furthermore, there was only one case (Child network with sample size of 500) where another method (equal-width binning with k = 3) outperformed it simultaneously on both metrics.

Table V-4: Average Normalized SHD Results: SHD results were normalized by results of Bach's method on the same network and sample size. Results were averaged over all networks. Values less than one indicate an algorithm learned more accurate networks than Bach's method for a given sample size. The numbers at the end of an algorithm denotes k, the number of initial discretization intervals.

General Category	**Algorithm**	**Average Normalized SHD**			**Average over SS**
		SS=500	SS=1000	SS=5000	
Preprocessing Discretization	EQFREQ2	1.40	1.61	1.62	1.55
	EQFREQ3	1.12	1.21	1.07	1.13
	EQWIDTH2	1.29	1.43	1.46	1.39
	EQWIDTH3	1.04	1.09	0.87	1.00
	HARTEMINK	1.77	2.08	1.60	1.82
Integrated Discretization and Structure Learning	FRIEDMAN2	1.08	1.15	1.01	1.08
	FRIEDMAN3	1.03	1.11	0.91	1.02
	MONTI2	1.26	1.41	1.18	1.28
	MONTI3	1.06	0.98	0.83	.95
	STECK2	1.00	1.04	0.82	.96
	STECK3	1.04	1.10	0.78	.97
Direct Methods	BACH	1	1	1	1
	PC-MATLAB	1.38	1.59	1.70	1.56
	TETRAD4	1.27	1.44	1.23	1.31

Quality Results The SHD results are shown in Table V-4 and are separated by sample size. Averages over sample sizes are also included. An algorithm with a value greater than one produced more structural errors than Bach's method on average, and an algorithm with a value less than one is considered more accurate than Bach's method.

At the smallest sample size, Bach's method and Steck's method (k=2) performed best while equal-width binning (k=3) was the best pre-discretization method. With sample size 1000, Monti's method (k=3) was slightly more accurate than Bach's method. Once again, equal-width binning (k=3) was the best pre-discretization method. At the largest sample size, a number of methods outperformed Bach's method, specifically equal-width binning (k=3), Friedman's method (k=3), Monti's method (k=3), and Steck's method (k=2, 3). Overall, equal-width binning (k=3), Monti's method (k=3), Steck's method, and Bach's method were the most accurate methods from their respective categories.

For the discretization approaches, the user must specify a parameter for the initial number of intervals. It is interesting to note the effect this choice had on results. If the initial number of levels affects accuracy so drastically, it is important for researchers to use the best value. However, there is no theoretically justified manner for selecting it, and this can be viewed as a weakness for the class of approaches since researchers can only guess or try a number of values.

Another interesting result is that the binning techniques performed comparably to the integrated methods. The integrated approaches potentially avoid poor initial discretizations. However, the theoretical benefits do not appear to warrant the additional computation. Further investigation is presented in section V.1.4. Specific running time results are detailed in the next section.

Table V-5: Average Normalized Time Results: Time results were normalized by results of Bach's method on the same network and sample size. Results were averaged over all networks. Values less than one indicate an algorithm requires less time than Bach's method for a given sample size. The numbers at the end of an algorithm denotes k, the number of initial discretization intervals.

General Category	**Algorithm**	**Average Normalized Time**			**Average over SS**
		SS=500	SS=1000	SS=5000	
Preprocessing Discretization	EQFREQ2	1.58	1.90	3.24	2.24
	EQFREQ3	1.15	1.84	2.69	1.89
	EQWIDTH2	1.36	2.01	3.39	2.25
	EQWIDTH3	1.16	1.85	2.50	1.84
	HARTEMINK	179.46	245.24	576.91	333.87
Integrated Discretization and Structure Learning	FRIEDMAN2	62.92	165.18	1375.24	534.45
	FRIEDMAN3	50.53	138.10	1001.61	396.75
	MONTI2	139.83	274.30	1327.84	580.65
	MONTI3	153.94	388.61	1183.26	575.27
	STECK2	244.48	553.39	2464.61	1087.49
	STECK3	286.09	536.81	2557.84	1126.91
Direct Methods	BACH	1	1	1	1
	PC-MATLAB	3.77	7.61	59.94	23.77
	TETRAD4	0.23	0.22	0.22	0.23

Efficiency Results The timing results are shown in Table V-5, and they were normalized in the same fashion as the SHD results. A value less than one signifies that a method required less time than Bach's method on average. Comparisons between Tetrad4 and the other methods should not be made since the Tetrad implementation was in Java while the others were in Matlab. Tetrad4 has been excluded from subsequent timing discussion.

As shown in the table, the integrated methods required considerably more time than the other categories. These methods considered all midpoints in the data, and as sample size increased, the number of candidate thresholds increased quadratically. Also, the relatively slow performance of Hartemink's method demonstrated that discretization methods are not necessarily the fastest since the discretization process may still require much time.

The fastest methods were the binning approaches and Bach's method. There were cases where binning methods required less time than Bach's method, but when results were averaged over all networks, Bach's method was the fastest method. Even though the binning methods are the simplest techniques, they were not the most efficient, and this finding was explained by using another efficiency metric.

In addition to running time, we analyzed the number of statistical calls made during structure learning. For the discretization methods, the calls were calculations of the BDeu score. For Bach's method, the calls were calculations of Bach's scoring metric, and for PC, the calls were conditional independence tests. Table V-6 shows the average normalized number of calls for all algorithms. Results for Tetrad4 were not reported since Tetrad did not output this information. The binning methods made fewer calls than Bach's method for all sample sizes. However, Bach's method was faster despite requiring more calls since it pre-computes the correlation matrix, or all the sufficient statistics, before structure learning. During structure learning, it only needs to compute determinants of the correlation matrix without the data. The binning methods consider the data each time they calculate a BDeu score to compute the number of instances, which takes more time and is linear to sample size.

Table V-6: Average Normalized Number of Statistical Calls During Structure Learning: The number of statistical calls (i.e. calls to a scoring function or conditional independence test) was normalized by the number of calls performed by Bach's method on the same network and sample size. Values less than one indicate an algorithm make fewer calls than Bach's method for a given sample size. Results were averaged over all networks. The numbers at the end of an algorithm denotes k, the number of initial discretization intervals.

General Category	**Algorithm**	**Avg. Norm. Num. of Calls**			**Average over SS**
		SS=500	SS=1000	SS=5000	
Preprocessing Discretization	EQFREQ2	0.88	0.73	0.76	0.79
	EQFREQ3	0.74	0.73	0.73	0.73
	EQWIDTH2	0.89	0.79	0.78	0.82
	EQWIDTH3	0.73	0.69	0.69	0.70
	HARTEMINK	0.69	0.60	0.58	0.62
Integrated Discretization and Structure Learning	FRIEDMAN2	20.12	20.14	18.63	19.63
	FRIEDMAN3	17.08	18.30	15.81	17.06
	MONTI2	14.83	14.49	17.66	15.66
	MONTI3	12.93	13.59	10.50	12.34
	STECK2	19.05	18.88	17.07	18.33
	STECK3	19.18	16.70	16.02	17.30
Direct Methods	BACH	1	1	1	1
	PC-MATLAB	7.40	15.45	76.04	32.96
	TETRAD4	n/a	n/a	n/a	n/a

For the integrated methods, we analyzed the number of scoring function calls during the discretization phases. Results were summed over all iterations and normalized by the results for Friedman's method (k=3) since it performed the fewest number of calls. Table V-7 displays the results. For comparison purposes, table V-8 shows the timing results for the integrated methods. Friedman's method made the fewest number of calls, while Steck's method made the most number of calls. These findings were consistent with the timing results where Friedman's method was the fastest, while Steck's method was the slowest.

Table V-7: Average Normalized Number of Scoring Function Calls During Discretization for the Integrated Methods: The number of calls to discretization scoring functions was normalized by the number of calls performed by Friedman's method (k=3) on the same network and sample size. Results were averaged over all networks. Values less than one indicate an algorithm made fewer calls than Friedman's method (k=3) for a given sample size. The numbers at the end of an algorithm denotes k, the number of initial discretization intervals.

Algorithm	**Avg. Norm. Num. of Calls**			**Average over SS**
	SS=500	SS=1000	SS=5000	
Friedman2	1.13	1.09	1.15	1.12
Friedman3	1	1	1	1
Monti2	1.29	1.06	1.04	1.13
Monti3	1.41	1.41	0.91	1.25
Steck2	1.87	1.69	1.58	1.71
Steck3	1.90	1.47	1.44	1.60

Table V-8: Average Normalized Time for the Integrated Methods: Time results were normalized by results of Bach's method on same network and sample size. Results were averaged over all networks. Values less than one indicate an algorithm requires less time than Bach's method for a given sample size. The numbers at the end of an algorithm denotes k, the number of initial discretization intervals.

Algorithm	**Avg. Norm. Time**			**Average over SS**
	SS=500	SS=1000	SS=5000	
Friedman2	62.92	165.18	1375.24	534.45
Friedman3	50.53	138.10	1001.61	396.75
Monti2	139.83	274.30	1327.84	580.65
Monti3	153.94	388.61	1183.26	575.27
Steck2	244.48	553.39	2464.61	1087.49
Steck3	286.09	536.81	2557.84	1126.91

Comparison of SHD and Time Results By graphing SHD and time on the same graph, we can analyze the tradeoffs between quality and efficiency. For example, if a method provided slight quality gains but required much more time, then the improvement in accuracy might not be worth the additional computation. Figures V-4, V-5, and V-6 display the data previously reported in Tables V-4 and V-5. The x-axis depicts average normalized time, while the y-axis depicts average normalized SHD. Each point represents the performance of an algorithm for both metrics. If a point is left of another point, then it is faster than the other algorithm. If a point is below another point, then it is more accurate than the other algorithm. Point (1,1) denotes Bach's method since it is used for normalization.

With sample size 500, Bach's method was the faster of the two most accurate methods. With sample size 1000, Monti's method (k=3) was slightly more accurate than Bach's method but required much more time. With a sample size of 5000, a number of algorithms yielded better accuracy results than Bach's method, but all of them required more time. In all cases, there was no method that simultaneously outperformed Bach's method in accuracy and efficiency.

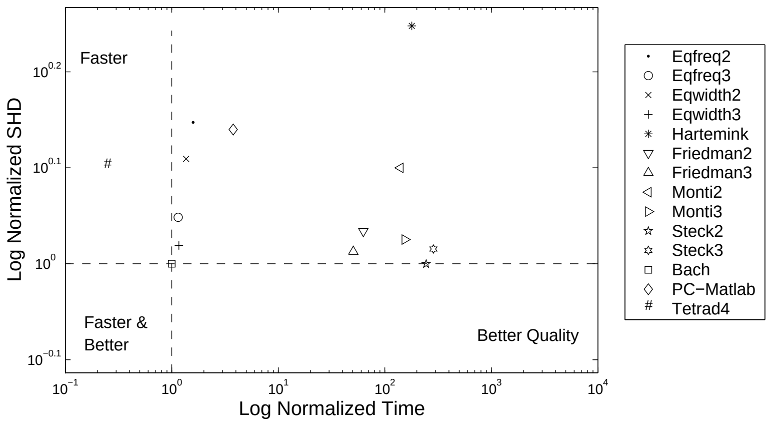

Figure V-4: Normalized Time vs. Normalized SHD for Sample Size 500.

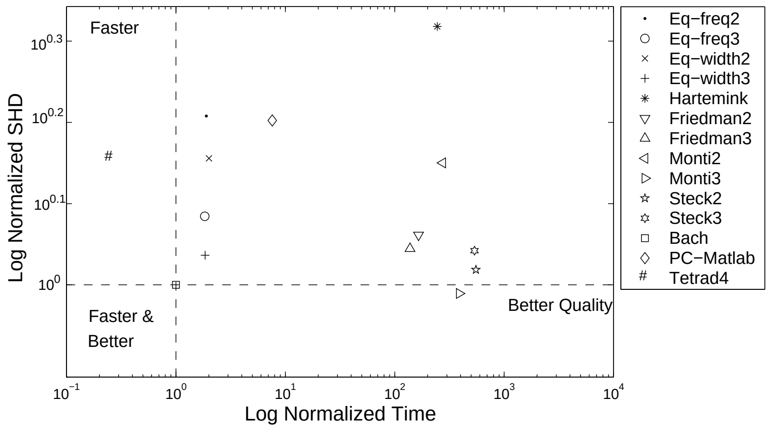

Figure V-5: Normalized Time vs. Normalized SHD for Sample Size 1000.

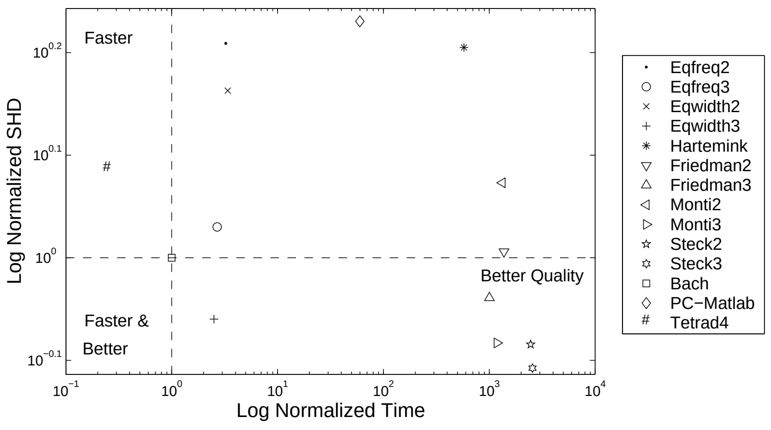

Figure V-6: Normalized Time vs. Normalized SHD for Sample Size 5000.

V.1.4 Supplemental Results

Comparison of Binning Methods to Integrated Methods

Using SHD as the quality metric, the integrated methods did not learn better structures than the binning methods. These results do not necessarily contradict Friedman's results (Friedman and Goldszmidt, 1996) since that evaluation focused on classification. Furthermore, Steck's evaluation (Steck and Jaakkola, 2003) did not perform a quantitative analysis. However, our results contradicted Monti's conclusion that his method yielded more accurate structures than equal-frequency binning (Monti, 1999). His findings employed a different quality metric, the percentage of extraneous and missing edges relative to the true structure. These values were calculated as the number of extraneous or missing edges divided by the total number of edges in the true graph. Results were averaged over all sample sizes and all values for k. Performing a similar analysis of our data results in Table V-9.

Monti's evaluation used the same values for k but consisted of sample sizes 1000, 2000, and 3000. The only common network between both evaluations is Alarm. The remaining networks were included for completeness along with the results for the other methods.

Table V-9: Average Percentage of Extraneous and Missing Edges: E is the average percentage of extraneous edges for the structures learned by an algorithm for a given network. M is the average percentage of missing edges for the structures learned by an algorithm for a given network. These values were calculated as the number of extraneous or missing edges divided by the total number of edges in the true graph. Results were averaged over all sample sizes and all values for k, the number of initial intervals.

	Alarm		**Child**		**Hailfinder**		**Insurance**	
Algorithm	**E**	**M**	**E**	**M**	**E**	**M**	**E**	**M**
Eqfreq	0.31	0.60	0.29	0.36	0.68	0.41	0.21	0.51
Eqwidth	0.37	0.42	0.24	0.33	0.56	0.43	0.18	0.46
Hartemink	0.76	0.73	0.63	0.73	0.74	0.88	0.43	0.78
Friedman	0.41	0.51	0.13	0.27	0.46	0.41	0.16	0.39
Monti	0.18	0.45	0.18	0.35	0.48	0.49	0.14	0.50
Steck	0.20	0.53	0.08	0.29	0.43	0.45	0.10	0.42
Bach	0.24	0.41	0.10	0.32	0.46	0.44	0.11	0.50
PC-Matlab	0.69	0.30	0.46	0.20	0.86	0.47	0.30	0.39
Tetrad4	0.21	0.69	0.23	0.49	0.17	0.71	0.06	0.75

For Alarm, Monti and Steck's methods produced a lower proportion of extraneous and missing edges than equal-frequency binning. Monti did not include results for equal-width binning. These findings are consistent with Monti's evaluation. For the remaining networks, all integrated methods output less extraneous edges while showing mixed results with respect to missing edges.

The edge identification results imply that the integrated methods are better than the binning methods for detecting dependencies and independencies among variables. However, the discrepancy with the SHD results suggests that the integrated methods may make relatively more errors when orienting edges or identifying the directionality of the relationship between dependent variables. This observation would explain why performance gains in edge identification did not translate to results based on SHD.

Simulated Data with Different Values of Sigma

An additional experiment explored the effect of using different standard deviation values during simulation. The purpose was to determine if algorithm performance varied depending on noise. From each category, the method with lowest average SHD over all networks and sample sizes was selected: equal width binning (k=3), Monti's method (k=3), and Bach's method. Data was simulated for all networks with sigma values of .30 and .40 to contrast the original value of .35. The results were not normalized to observe absolute changes in SHD. The graphs of SHD for each sigma value are in figures V-7, V-8, and V-9. For sample sizes 500 and 1000, SHD increased with larger sigma values. For sample size 5000, SHD did not fluctuate as much as it did with smaller sizes. The slight dip at 0.35 was probably an artifact of the SHD metric. With large sample size, more orientation errors likely occurred since an algorithm is more confident with additional data instances. This claim was supported by considering the extraneous and missing edge errors. For sample sizes 500 and 1000, SHD values increased with larger sigma values due to more missing edges. For sample size 5000, the number of missing edges similarly increased. The reason that SHD did not exhibit the same increasing behavior with large sample size was probably due to more orientation errors for sigma values 0.30 and 0.40.

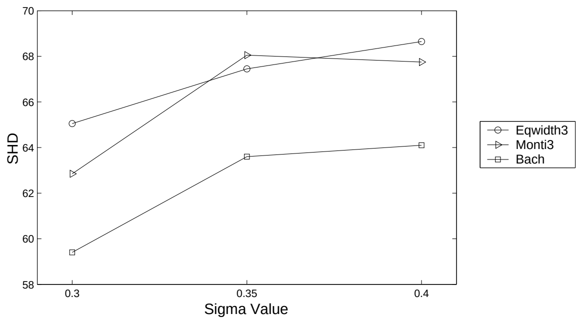

Figure V-7: SHD with Multiple Sigma Values for Simulated Data of Sample Size 500.

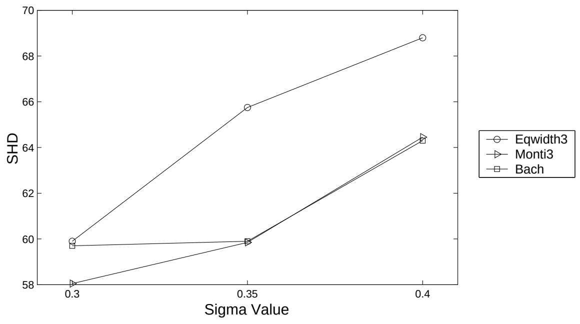

Figure V-8: SHD with Multiple Sigma Values for Simulated Data of Sample Size 1000.

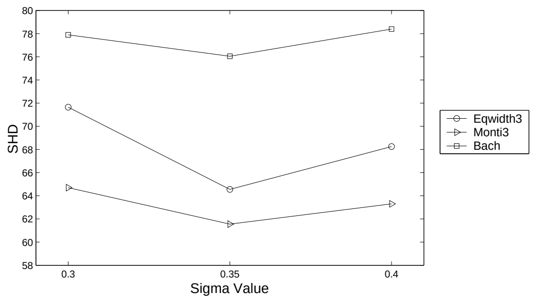

Figure V-9: SHD with Multiple Sigma Values for Simulated Data of Sample Size 5000.

V.2 Real Data

V.2.1 Data Sets

Real data sets were used from the UCI Machine Learning Repository[3]. Also, Hartemink's yeast data set (Hartemink et al., 2002) was used. The data sets ranged in size from 270 to 4000 cases as well as from 8 to 19 variables. Table V-10 lists the data sets used in the evaluation.

Table V-10: Real data sets: D is the number of discrete variables, and C is the number of continuous variables. Max Card. is the maximum cardinality for all continuous variables. In other words, it is the maximum number of unique values. Min Card. and Avg. Card are the minimum and average cardinalities for all continuous variables.

Network	**Sample Size**	**D**	**C**	**Max Card.**	**Min Card.**	**Avg Card.**
ABALONE	4175	1	8	2429	28	759.25
AUSTRALIAN	690	9	6	350	23	188.50
BREAST	683	1	10	10	9	9.89
CARS1	392	1	7	346	13	125.83
CLEVE	296	8	6	152	40	74.80
HARTEMINK	320	0	32	317	301	314.03
HEART	270	9	5	144	39	72.20
HOUSING	506	1	13	504	9	235.62
PIMA	768	1	8	517	17	156.75
VEHICLE	846	1	18	424	13	79.4

Real data do not necessarily exhibit the same Gaussian distributions as the simulated data sets. Figure V-10 shows an example from the Cars data set of the joint distribution of the target variable and the most associated variable to it as measured by Fisher's Z-test.

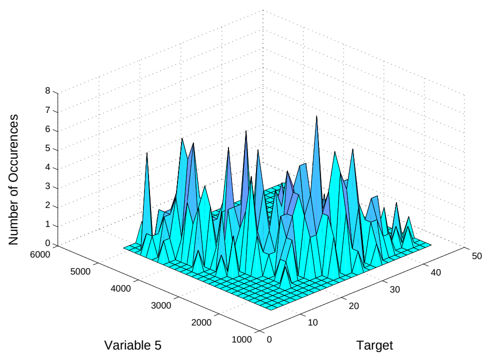

Figure V-10: For the Cars data set, the distribution of the target and its most associated variable does not show the same Gaussian behavior as the simulated data.

[3]Data sets available at http://www.ics.uci.edu/ mlearn/MLRepository.html

V.2.2 Metrics of Comparison

Unlike the simulated data sets, we do not know the true generating structures for the real data sets. Instead of using SHD to measure quality, Bach's scoring function was used to rank structures, and it was explained in detail in section III.3.1. A disadvantage of this metric is that it favors Bach's method. Using a Bayesian score such as the BDe score was considered, but this score requires all methods to output a final discretization of the data. Unfortunately, the direct methods do not discretize the data, which makes this approach unsuitable. Once again, running time and the number of statistical calls were used to evaluate the efficiency of the algorithms.

V.2.3 Results

All results were normalized in the same fashion as those from the simulated data.

Table V-11: Results for Average Normalized Bach Score, Time, and Number of Statistical Calls: Results were normalized by results of Bach's method on the same data set. Results were averaged over all data sets. For the score results, values less than one denote that a method learned lower quality networks than Bach's method. For the time results, values less than one denote that a method required less time than Bach's method. For the number of calls results, values less than one denote that a method made fewer calls than Bach's method. The numbers at the end of an algorithm denotes k, the number of initial discretization intervals.

General Category	**Algorithm**	**Avg. Norm. Score**	**Avg. Norm. Time**	**Avg. Norm. Num. of Calls**
Preprocessing Discretization	EQFREQ2	0.91	1.14	0.68
	EQFREQ3	0.95	1.04	0.63
	EQWIDTH2	0.79	0.97	0.60
	EQWIDTH3	0.89	0.92	0.59
	HARTEMINK	0.88	47.14	0.67
Integrated Discretization and Structure Learning	FRIEDMAN2	0.91	288.25	2.86
	FRIEDMAN3	0.91	169.32	2.48
	MONTI2	0.71	278.47	2.23
	MONTI3	0.84	194.05	2.08
	STECK2	0.93	691.65	3.06
	STECK3	0.92	502.56	2.62
Direct Methods	BACH	1	1	1
	PC-MATLAB	1.13	4.69	2.55
	TETRAD4	0.68	0.53	n/a

Quality Results The score results are shown in Table V-11. An algorithm with a value greater than one produced a greater quality structure than Bach's method. Bach's method and PC-Matlab induced the highest quality structures. For the prediscretization category, equal-frequency binning (k=3) was better than equal-width binning. The simulated data likely favored equal-width binning since the Gaussian distributions that centered around the original discrete values had equal widths. Also, Friedman and Steck's methods were the best integrated methods.

Two direct methods were distinctly better than the discretization methods. This differentiation did not appear in the simulated data, which suggests an important difference between the approaches and types of data. Discretization methods assume continuous data are noisy observations of an underlying discrete mechanism. Direct methods do not make this assumption. The simulated data sets were generated in a manner consistent with the assumptions made by the discretization methods, while the real data sets do not. The lack of an underlying discrete mechanism may explain the difference between performances on the two types of data.

Efficiency Results The timing results are shown in Table V-11. A value less than one signifies that a method required less time on average than Bach's method. The timing results for the real data were consistent with those from the simulated data. Bach's method required approximately the same amount of time on average as the binning methods. As with the simulated data, the integrated methods were much slower than the other methods.

Results for the number of statistical calls made by each algorithm during structure learning are also shown in Table V-11. A value less than one denotes that a method made fewer calls than Bach's method. As with the simulated data, the binning methods made fewer calls than Bach's method.

For the integrated methods, the results for the number of calls during the discretization phase are presented along with the timing results in Table V-12. The findings confirm the results from the simulated data that Friedman's method was the most efficient while Steck's method was the least efficient.

Table V-12: Average Normalized Number of Scoring Function Calls During Discretization for the Integrated Methods: The number of calls to discretization scoring functions was normalized by the number of calls performed by Friedman's method (k=3) on the same data set. Values greater than one indicate an algorithm made more calls than Friedman's method (k=3). Results were averaged over all data sets. The numbers at the end of an algorithm denotes k, the number of initial discretization intervals. The timing results are included for comparison.

Algorithm	**Avg. Norm. Num. of Calls**	**Avg. Norm. Time**
FRIEDMAN2	1.19	288.25
FRIEDMAN3	1	169.32
MONTI2	1.37	278.47
MONTI3	1.39	194.05
STECK2	2.25	691.65
STECK3	1.73	502.56

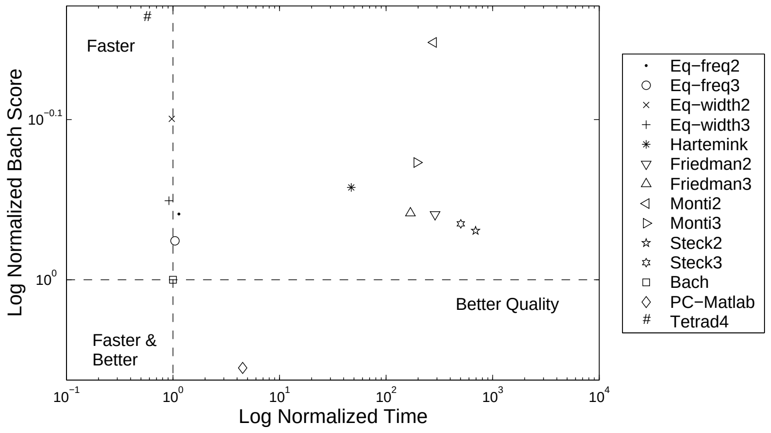

Figure V-11: Normalized Time vs. Normalized Score for Real Data: The y-axis was flipped to maintain consistency with previous figures.

Comparison of Score with Time Figure V-11 displays the data previously reported in Table V-11 with the y-axis flipped. The transformation was necessary to maintain consistency with previous figures. The x-axis depicts average normalized time, while the y-axis depicts average normalized Bach score. Each point represents algorithm performance for both metrics. If a point is left of another point, then it is faster than the other algorithm. If a point is below another point, then it output higher quality structures. Point (1,1) denotes Bach's method since it is used for normalization.

Bach's method resulted in higher quality structures compared to the binning methods while requiring about the same amount of time. PC-Matlab produced better graphs compared to Bach's method but required more time. Finally, the integrated methods produced structures comparable to the binning methods in terms of quality but required considerably more time.

V.3 Summary of Results

The empirical evaluation compared the performances of eight algorithms on two types of continuous data. Performance was measured by the quality of learned structures and efficiency. The discretization-based approaches yielded the most accurate structures when learning from simulated data with large sample size. These methods assume that data is generated from an underlying discrete mechanism, which was true for the simulated data. With small simulated data sets, Steck's method (k=2) was as accurate as Bach's method but required more time. With real data, two direct methods, Bach's method and PC-Matlab, learned the highest quality structures. In terms of efficiency, Bach's method was fastest with respect to running time. When the number of statistical calls was measured, the binning methods were most efficient.

Another finding was that the integrated methods did not necessarily learn more accurate structures than the binning methods. The SHD results demonstrated comparable performances between the integrated and binning methods. However, the integrated methods learned more accurate structures when we ignored edge orientation errors.

Overall, Steck's method and Bach's method were the best methods from their respective categories. Equal-width binning (k=3) was the best pre-discretization approach for simulated data, while equal-frequency binning (k=3) was better with real data.

CHAPTER VI

DISCUSSION

Biomedical researchers today often study continuous data sets, and Bayesian network structure learning algorithms are an increasingly popular tool for learning the relationships among variables. This work focused on the task of learning Bayesian network structure from continuous data. It compared the three general approaches of discretization as a preprocessing step, integrating discretization and structure learning, and learning directly with continuous data.

The conclusions from the study may benefit researchers learning Bayesian network structure from continuous data. First, of the methods included, Bach's method demonstrated the best overall performance across all comparison metrics and data types. This is an important finding since it showed that a direct method can be as efficient as the binning approaches. Furthermore, the assumptions made by this direct method did not prohibit it from learning accurate structures.

The work also emphasized the importance of considering the nature of the data. If the data is fundamentally discrete, a discretization-based approach is appropriate. Discretization techniques assume that data are noisy observations originating from an underlying discrete mechanism, while the direct methods do not. Also, the sample size of the data should be factored into the decision of which approach to utilize since the discretization methods performed best with large sample size.

Another goal of the work was to assess whether the integrated methods provided quality improvements over the binning methods. The purpose of the integrated methods is to improve the accuracy of pre-discretization approaches by avoiding poor initial discretizations. When considering edge orientation of the induced structures, integrated methods did not provide benefits that would warrant the additional computation. However, when only factoring in edge identification, the integrated methods increased accuracy.

VI.1 Limitations

While this work performed a comparison of the three approaches in a unified study, there were some limitations that could be addressed in future work. First, the evaluation would have benefited from known gold standards for continuous Bayesian networks. In the absence of such networks, simulated and real data sets without known structures were used. Second, the simulation technique was relatively simple. Real world data does not necessarily start as discrete data with added Gaussian noise. However, it still provided useful insights, such as the improved performance of discretization with large sample sizes, which may generalize. Third, for the real data sets, Bach's scoring metric was a biased metric. Although there is no guarantee that the learned structures resembled the true networks, it still provided some intuition about the quality of the learned structure.

VI.2 Future Work

The results displayed the effect that the number of initial intervals can have on the output of discretization-based methods. This experimental evaluation only tried two values, but it would be useful for researchers if studies were conducted to identify scenarios where more or fewer intervals would yield better results. Also, researchers would benefit if there were some recommendations on how to select the number of intervals instead of simply guessing or trying multiple values.

Another extension to this work could be comparing algorithm performance on data sets of different characteristics from those presented. One option is studying networks with more variables. A class of methods may be better suited than the others to handle the additional requirements. Another option is studying smaller sample sizes. Real world data sets do not always have thousands of instances, and small data sets would more closely mimic the situations encountered by researchers.

Bach's method demonstrated that directly learning with continuous data does not necessarily involve crippling assumptions or excessive computational requirements. This work was not exhaustive, and there were some methods that were not included due to computational restrictions. These methods should be investigated further. Also, the development of a novel method for learning directly with continuous data is another possible direction to continue this work.

BIBLIOGRAPHY

Abramson, B., Brown, J., Edwards, W., Murphy, A., and Winkler, R. (1996). Hailfinder: A bayesian system for forecasting sever weather. *International Journal of Forecasting*, 12(1):57–71.

Antal, P., Fannes, G., Timmerman, D., Moreau, Y., and De Moor, B. (2003). Bayesian applications of belief networks and multilayer perceptrons for ovarian tumor classification with rejection. *Artificial Intelligence in Medicine*, 29(1-2):39–60.

Bach, F. R. and Jordan, M. I. (2003). Learning graphical models with mercer kernels. In S. Becker, S. T. and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1009–1016. MIT Press, Cambridge, MA.

Bay, S. D. (2001). Multivariate discretization for set mining. *Knowledge and Information Systems*, 3(4):491–512.

Beinlich, I., Suermondt, G., R., C., and Cooper, G. (1989). The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *2nd European Conference in Artificial Intelligence in Medicine*, pages 247–256.

Binder, J., Koller, D., Russell, S., and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2-3):213–244.

Burnside, E. S., Rubin, D. L., and Shachter, R. D. (2004). Using a bayesian network to predict the probability and type of breast cancer represented by microcalcifications on mammography. In *Proceedings of the 11th World Congress on Medical Informatics (Medinfo 2004)*, pages 13–18.

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. In *Proceedings of the Fifth European Working Session on Learning.*

Chickering, D. M. (1995). A transformational characterization of equivalent bayesian network structures. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 87–98. Morgan Kaufmann Publishers.

Cooper, G., Dash, D., Levander, J., Wong, W.-K., Hogan, W., and Wagner, M. (2004). Bayesian biosurveillance of disease outbreaks. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 94–103, Arlington, Virginia. AUAI Press.

Cooper, G. F. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347.

Cowell, R., Dawid, A., Lauritzen, S., and Spiegelhalter, D. (1999). *Probabilistic Networks and Expert Systems.* Springer.

Davies, S. (2002). *Fast Factored Density Estimation and Compression with Bayesian Networks.* PhD thesis, Carnegie Mellon University.

de Merckt, T. V. (1993). Decision trees in numerical attribute spaces. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1016–1021. Morgan Kaufmann.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202.

Elomaa, T. and Rousu, J. (2004). Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. *Data Mining and Knowledge Discovery*, 8(2):97–126.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1022–1027. Morgan Kaufmann.

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799–805.

Friedman, N. and Goldszmidt, M. (1996). Discretizing continuous attributes while learning bayesian networks. In *Proceedings of the 13th International Conference on Machine Learning (ICML).*

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620.

Friedman, N. and Nachman, I. (2000). Gaussian process networks. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 211–219, San Francisco, CA. Morgan Kaufmann Publishers.

Geiger, D. and Heckerman, D. (1994). Learning gaussian networks. In *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 235–243, San Francisco, CA. Morgan Kaufmann Publishers.

Gray, R. M. and Neuhoff, D. L. (1998). Quantization. *IEEE Transactions on Information Theory*, 44(6):1–63.

Hartemink, A. J. (2001). *Principled Computational Methods for the Validation and Discovery of Genetic Regulatory Networks.* PhD thesis, Massachusetts Institute Of Technology.

Hartemink, A. J., Gifford, D. K., Jaakkola, T. S., and Young, R. A. (2002). Combining location and expression data for principled discovery of genetic regulatory network models. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 437–449.

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

Helman, P., Veroff, R., Atlas, S. R., and Willman, C. (2004). A bayesian network classification methodology for gene expression data. *Journal of Computational Biology*, 11(4):581–615.

Ho, K. and Scott, P. D. (1997). Zeta: A global method for discretization of continuous variables. In *Third International Conference of Knowledge Discovery and Data Mining.*

Hofmann, R. and Tresp, V. (1996). Discovering structure in continuous variables using bayesian networks. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 500–506. The MIT Press.

Imoto, S., Goto, T., and Miyano, S. (2002). Estimation of genetic networks and functional structures between genes by using bayesian network and nonparametric regression. In *Pacific Symposium On Biocomputing.*

Imoto, S., Kim, S., Goto, T., Aburatani, S., Tashiro, K., Kuhara, S., and Miyano, S. (2003). Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network. *Journal of Bioinformatics and Computational Biology*, 1(2):231–252.

Kerber, R. (1992). Chimerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 123–128.

Kohavi, R. and Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 114–119.

Kozlov, A. and Koller, D. (1997). Nonuniform dynamic discretization in hybrid networks. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 314–325, San Francisco, CA. Morgan Kaufmann Publishers.

Kwedlo, W. and Kretowski, M. (1999). An evolutionary algorithm using multivariate discretization for decision rule induction. In *Principles of Data Mining and Knowledge Discovery*, pages 392–397.

Lam, W. and Bacchus, F. (1994). Learning bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence*, 10(4):269–293.

Lauritzen, S. L. and Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1):31–57.

Liu, H., Hussain, F., Tan, C. L., and Dash, M. (2002). Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423.

Liu, H. and Setiono, R. (1997). Feature selection via discretization. *Knowledge and Data Engineering, IEEE Transactions on*, 9(4):642–645.

Lucas, P. J. F., van der Gaag, L. C., and Abu-Hanna, A. (2004). Bayesian networks in biomedicine and health-care. *Artificial Intelligence in Medicine*, 30(3):201–214.

Margaritis, D. (2005). Distribution-free learning of bayesian network structure in continuous domains. In *Proceedings of The Twentieth National Conference on Artificial Intelligence (AAAI).*

Monti, S. (1999). *Learning Hybrid Bayesian Networks from Data.* PhD thesis, University of Pittsburgh.

Monti, S. and Cooper, G. (1999). A latent variable model for multivariate discretization. In *Proceedings of the Seventh International Workshop on AI & Statistics (Uncertainty 99).*

Muhlenbach, F. and Rakotomalala, R. (2002). Multivariate supervised discretization, a neighborhood graph approach. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 314–321.

Nachman, I., Elidan, G., and Friedman, N. (2004). ideal parent structure learning for continuous variable networks. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 400–409, Arlington, Virginia. AUAI Press.

Neapolitan, R. E. (2003). *Learning Bayesian Networks*. Prentice Hall.

Pe'er, D., Regev, A., Elidan, G., and Friedman, N. (2001). Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(90001):215S–224.

Pournara, I. and Wernisch, L. (2004). Reconstruction of gene networks using bayesian learning and manipulation experiments. *Bioinformatics*, 20(17):2934–2942.

Robles, V., Larranaga, P., Pena, J. M., Menasalvas, E., Perez, M. S., Herves, V., and Wasilewska, A. (2004). Bayesian network multi-classifiers for protein secondary structure prediction. *Artificial Intelligence in Medicine*, 31(2):117–136.

Scott, D. W. (1992). *Multivariate Density Estimation*. John Wiley & Sons, Inc.

Spirtes, P., Glymour, C., and Scheines, R. (2001). *Causation, Prediction, and Search*. MIT Press.

Steck, H. and Jaakkola, T. (2003). (semi)-predictive discretization during model selection. Technical report, Massachusetts Institute of Technolody.

Tay, F. and Shen, L. (2002). A modified chi2 algorithm for discretization. *Knowledge and Data Engineering, IEEE Transactions on*, 14(3):666–670.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2004). The max-min hill-climbing bayesian network structure learning algorithm. Technical Report DSL TR-05-01, Vanderbilt University.

Yoo, C. and Cooper, G. F. (2004). An evaluation of a system that recommends microarray experiments to perform to discover gene-regulation pathways. *Artificial Intelligence in Medicine*, 31(2):169–182.