Using Minimum Description Length for Discretization Classification of Data Modeled by Bayesian Networks

by

Nicholas D. Levine

B.S. in Mathematics, University of Denver, 2003

Master of Engineering Management, Old Dominion University, 2007

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Applied Mathematics

2011

This thesis entitled:
Using Minimum Description Length for Discretization Classification of Data Modeled by Bayesian Networks
written by Nicholas D. Levine
has been approved for the Department of Applied Mathematics

Dr. Jem N. Corcoran

Dr. Debra S. Goldberg

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Levine, Nicholas D. (M.S., Applied Mathematics)

Using Minimum Description Length for Discretization Classification of Data Modeled by Bayesian Networks

Thesis directed by Prof. Dr. Jem N. Corcoran

A Bayesian network is a graphical model that encodes conditional probability relationships among multiple variables. Their applications extend through computational biology and bioinformatics, medicine, image processing, decision support systems, and engineering. When applying statical analysis to Bayesian networks several advantages are gained. First, Bayesian networks are able to model many variables at once. They are able to predict casual relationships between variables that allow for predictions to be made if changes are made to specific area in the network. Second, Bayesian networks represent both casual and probabilistic relationships combining prior knowledge to currently viewed data. This proves as a very valuable tool in problem solving. The fundamental aspect of these powerful networks is the data that is used to construct them. Existing recovery algorithms require either discrete or Gaussian data. Non-Gaussian continuous data is normally discretized in an ad-hoc and careless manner which is highly likely to destroy the precise conditional dependencies we are out to recover.

We explore the effectiveness of a method due to Friedman and Goldszmidt (1996) that is based on a metric from information theory known as "description length". While theoretical interesting, their proposed search strategy for an optimal discretization is infeasible for even moderately sized data sets on the smallest of networks. We introduce a new search strategy based on a "top-down" approach utilizing a local description length metric that can be implemented with significant time savings.

Dedication

To my wife, Jill, and our four little girls, Dylan, Riley, Keegan and Sawyer.

Acknowledgements

I would like to thank my advisor, Jem Corcoran, for her continued support. I think that it is only fitting that my coursework both started and ended with her teachings. You have made a lasting impression on my desire to continue to learn both in the classroom and in research. Thank you for sacrificing so much of your personal time to teach!

To Anne Dougherty, thanks for being such a great supporter and advocate for me in this program. You were my first impression of the department (from Cuban soil) and I appreciate your open communication and effort in securing my place in the department.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

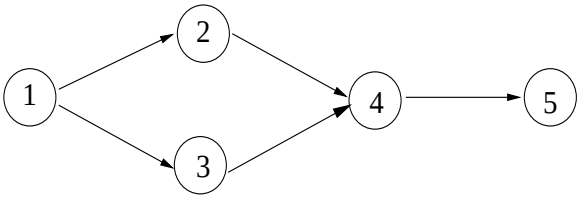
A Bayesian network is a convenient graphical way to visualize and convey potentially complex probabilistic dependencies between a large number of random variables. It consists of a set of conditional probability distributions and a directed acyclic graph in which nodes represent the random variables and edges between nodes represent dependencies. There has been a great deal of work done ([6],[8],[11], [12], [14],[15],[17],[18],[20]) on the problem of recovering (learning) the structure of a generating network given data. Typically, the network random variables are either discrete or, if not, they are either assumed to be Gaussian or are otherwise discretized before network recovery. Often ([9], [16], [18], [22]) this discretization is performed in an ad hoc manner. Unfortunately, such a non-rigorous approach is highly likely to destroy the precise conditional dependencies one is out to recover (see sections 1.2.1 and 1.2.2). In this thesis, we explore the effectiveness of a method due to Friedman and Goldszmidt [10] and introduce search strategies that take this method from interesting to feasible. Throughout this thesis, we make heavy use of simulated data so that we can verify that network recovery and discretization algorithms are working correctly. We will focus on the ideal circumstances such that we are given small networks with lots of data. If the true network or discretization cannot be recovered in these instances, they will not be recovered in the less than ideal cases.

1.1 Bayesian Networks

A directed acyclic graph (DAG) is a set of nodes and directed edges which, as the name suggests, do not form a closed loop or cycle. In a DAG for a Bayesian network, the nodes represent random variables and the edges represent dependencies between them.

Consider the directed acyclic graph on five nodes shown in Figure 1.1.

Figure 1.1: A Directed Acyclic Graph



In this graph, we say that node 1 is a **parent** of nodes 2 and 3 and nodes 2 and 3 are **children** of node 1. Figure 1.1 represents a joint density $p(x_1, x_2, \ldots, x_5)$ for random variables $X_1, X_2, \ldots, X_5$ that can be written as

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3)p(x_5|x_4). \tag{1.1}$$

We can see from (1.1) that the random variables X_2 and X_3, for example, are independent once the value of X_1 is fixed or given. For a general Bayesian network, we say that "children of common parents are conditionally independent given their parents". Equation (1.1) also tells us that the random variables X_1 and X_4 are independent once the value of X_2 is given. We say in general that "each random variable in a Bayesian network is independent of its non-descendants given its parents".

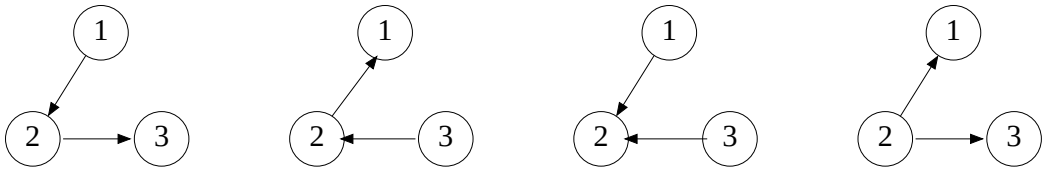
To further illustrate this point, see Figure 1.2 below. Given are four examples of DAGs that possess the same edges, ignoring directions. They correspond to the following four joint probability

densities.

$$\begin{aligned} p(x_1, x_2, x_3) &= p(x_1)p(x_2|x_1)p(x_3|x_2) && \text{(a)} \\ p(x_1, x_2, x_3) &= p(x_3)p(x_2|x_3)p(x_1|x_2) && \text{(b)} \\ p(x_1, x_2, x_3) &= p(x_1)p(x_3)p(x_2|x_1, x_3) && \text{(c)} \\ p(x_1, x_2, x_3) &= p(x_2)p(x_1|x_2)p(x_3|x_2) && \text{(d)} \end{aligned} \tag{1.2}$$

From (1.2), it is clear that for graphs (a), (b), and (d) X_1 and X_2 are dependent, X_2 and X_3 are dependent, and X_1 and X_3 are dependent. Additionally, if we are given X_2, then X_1 and X_3 are independent. In contrast, graph (c) is oriented such that X_2 is a child of both X_1 and X_3. Thus, X_1 and X_2 are dependent and X_2 and X_3 are dependent, but X_1 and X_3 are independent (without being giving X_2). As they encode the same dependencies, graphs (a), (b), and (d) are equivalent in some sense.

Figure 1.2: Four DAGs With the Same (Undirected) Edges



(a) (b) (c) (d)

A set of DAGs with the same set of edges are said to be **Markov equivalent** if, in general, they DAGs share the same set of conditional independence relations among variables. Within a Markov equivalence class DAGs differ when it comes to causality. Network recovery algorithms run on a fixed data set can not distinguish between Markov equivalent graphs. The distinction can be made if one has the ability to generate or collect data where certain nodes are being held to fixed values. In this thesis, we will focus only on graph recovery up to the equivalence class.

In general, a Bayesian network on n nodes (random variables) consists of a directed acyclic graph on n nodes and a set of conditional probability density functions that make up the terms in

an assumed joint density

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i|\Pi_i).$$

We are using Π_i to denote the set of random variables corresponding to parent nodes of node i.

1.2 Data and Recovery

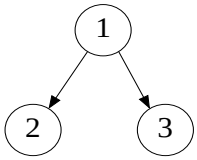
To date, much work has been done on the recovery of these Bayesian networks to identify their specific structure. Examples can be found in [6],[8],[11], [12], [14],[15],[17],[18], and [20]. Although recovery is important, most Bayesian network learning procedures make one of the following two assumptions: (1) that the data are discrete **or** (2) that the data are continuous and either follow a Gaussian distribution or are otherwise discretized before recovery. In many cases the data is discretized in an ad hoc manner which often destroys the conditional relationships among variables.

As discretization of continuous random variables involves reassigning all values in a particular interval to a single value, it is, in a sense, a classification problem which may also be encountered in problems with originally discrete data. We will spend a significant portion of this thesis discussing the "discretization of discrete variables".

1.2.1 Discretizing Discrete Data

To illustrate the concern in discretizing already discrete data consider the DAG in Figure 1.3.

Figure 1.3: A Three Node DAG



Now consider the discretization defined for $i = 1, 2, 3$, let

$$Y_i = \begin{cases} 1 \quad , \quad \text{if } X_i \in \{1, 2\} \\ \\ 2 \quad , \quad \text{if } X_i = 3. \end{cases} \tag{1.3}$$

Our goal is to now show that the conditional independence of nodes 2 and 3 given node 1 is not preserved. From above we have:

$$\begin{aligned} P(Y_2 = 1, Y_3 = 1 | Y_1 = 1) &= P(X_2 \in \{1,2\}, X_3 \in \{1,2\} | X_1 \in \{1,2\}) \\ &= \frac{P(X_1 \in \{1,2\}, X_2 \in \{1,2\}, X_3 \in \{1,2\})}{P(X_1 \in \{1,2\})} \\ &= \frac{P(X_2 \in \{1,2\}, X_3 \in \{1,2\} | X_1 = 1) P(X_1 = 1) + P(X_2 \in \{1,2\}, X_3 \in \{1,2\} | X_1 = 2) P(X_1 = 2)}{P(X_1 \in \{1,2\})}. \end{aligned}$$

Since X_2 and X_3 are conditionally independent given X_1, we can factor the numerator to get

$$\begin{aligned} P(Y_2 = 1, Y_3 = 1 | Y_1 = 1) &= \frac{P(X_2 \in \{1,2\} | X_1 = 1) P(X_3 \in \{1,2\} | X_1 = 1) P(X_1 = 1)}{P(X_1 \in \{1,2\})} \\ &\quad + \frac{P(X_2 \in \{1,2\} | X_1 = 2) P(X_3 \in \{1,2\} | X_1 = 2) P(X_1 = 2)}{P(X_1 \in \{1,2\})} \end{aligned}$$

Similarly one can show that:

$$P(Y_2 = 1 | Y_1 = 1) = \frac{P(X_2 \in \{1,2\} | X_1 = 1) P(X_1 = 1) + P(X_2 \in \{1,2\} | X_1 = 2) P(X_1 = 2)}{P(X_1 = 1) + P(X_1 = 2)}$$

and

$$P(Y_3 = 1 | Y_1 = 1) = \frac{P(X_3 \in \{1,2\} | X_1 = 1) P(X_1 = 1) + P(X_3 \in \{1,2\} | X_1 = 2) P(X_1 = 2)}{P(X_1 = 1) + P(X_1 = 2)}$$

Combining the equalities above we see that:

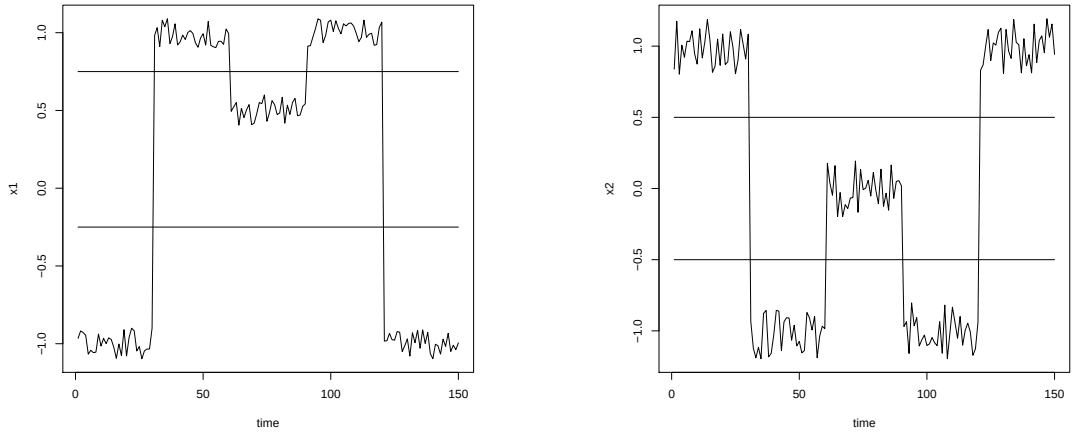
$$P(Y_2 = 1, Y_3 = 1 | Y_1 = 1) \neq P(Y_2 = 1 | Y_1 = 1) \cdot P(Y_3 = 1 | Y_1 = 1) \tag{1.4}$$

Thus, nodes 2 and 3 are not conditionally independent given node 1.

1.2.2 Discretizing Continuous Data

An example of ad hoc discretization of continuous data can be found in [18]. In this paper, the authors had simulated time series data for several nodes similar to that depicted in Figure 1.4. There was a fairly obvious relationship between the random variables X_1 and X_2 in a "high/low sense", but once high or low, each random variable was then, independently, augmented by Gaussian noise. The discretizations used divided up the data (the y-axis) into the dominant regimes of low, middle, and high range values (assigned as 1's, 2's, and 3's) as depicted by the breaks determined by the horizontal lines in Figure 1.4. However, the independence of the noise components were completely lost and the resulting data appeared to have completely deterministic relationships. For example, every instance of X_1 as 1 was matched by an instance of X_2 as -1. This sort of overly coarse discretization, resulting in a loss of information about dependence or conditional independence, appears typical among practitioners in the literature.

Figure 1.4: An Example of Loss of Conditional Independence



1.3 The Minimum Description Length Approach

In this thesis, we explore an approach for discretization by Friedman and Goldszmidt that is based on a measure of information know as "minimum description length (MDL)." This thesis is organized as follows. In Chapter 2 we formally define a Bayesian network to include notation used throughout this paper and discuss discrete recovery. In Chapter 3 we describe minimum description length and how it is currently applied. Chapter 4 focuses on MDL as it is utilized for discretization. We introduce a local description length as a score for discretization and show how it can significantly reduce the number of possible discretizations. Finally, in Chapter 5 we look at the continuous data and evaluate techniques to maintain the integrity of the data while producing accurate results in a timely manner.

Chapter 2

Formalities and Previous Work

2.1 Bayesian Network Notation

Bayesian networks are directed acyclic graphs that encode joint probability distributions for the random variables used to construct the network. Formally, for a finite set of discrete random variables U=(X_1,...,X_n), a Bayesian network consists of the pair $B = (G, \Theta)$ where G is a directed acyclic graph on n nodes and Θ is a parameter set containing discrete values $\theta_{ijk} = p(X_i = k|\Pi_i = j)$ for all possible k in the support of X_i, and all possible j, where j is an enumeration of a configuration of Π_i, the set of random variables corresponding to parent nodes of node i. G defines a unique joint probability distribution over U given by:

$$p(u) = p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i|\Pi_i) = \prod_{i=1}^{n} \theta_{i,\Pi_i,x_i}. \tag{2.1}$$

As stated before, each node is independent of it non-descendants given its parents and second, children of common parents are conditionally independent given their parents. Π_i represents the set of random variables corresponding to parent nodes X_i in G.

2.2 The Multinomial Connection

The multinomial distribution is an extension of the binomial distribution where the outcomes of a given experiment can result in more than one outcome. We now describe the relationship between a Bayesian network and the multinomial distribution.

2.2.1 The Multinomial Distribution

Consider an experiment with m independent trials and r possible outcomes on each trial. Let θ_i, for $i = 1, 2, \ldots, r$ be the probability that any one trial results in outcome i. Define the random variables $Y_1, Y_2, \ldots, Y_r$ where Y_i is the number of trials that result in outcome i. Then the vector $Y := (Y_1, Y_2, \ldots, Y_r)$ has a **multinomial distribution** with parameters m, r, and $\theta = (\theta_1, \theta_2, \ldots, \theta_r)$.

The probability distribution for Y is given by

$$p(Y_1 = y_1, Y_2 = y_2, \ldots, Y_r = y_r) = \frac{m!}{y_1! y_2! \cdots y_r!} \theta_1^{y_1} \theta_2^{y_2} \cdots \theta_r^{y_r} \tag{2.2}$$

where $\sum_{i=1}^{r} \theta_i = 1$ and $y_1, y_2, \ldots, y_r$ are non-negative integers summing to m.

2.2.2 The Multinomial Network

Consider now a Bayesian network on n nodes. Let X_i, for $i = 1, 2, \ldots, n$ denote the random variable associated with node i. We assume that X_i can take on r_i values and, for simplicity, we will assume that the r_i values are the integers $1, 2, \ldots, r_i$. We express the dependency of each variable X_i on its parents as:

$$p(X_i = k | \Pi_i) = \theta_{i,\Pi_i,k}$$

where Π_i is a particular configuration of the parent variables of X_i. We will also use the reduced subscript notation

$$\theta_{i,k} = p(X_i = k)$$

if $\Pi_i = \emptyset$.

If we enumerate the number of possible configurations of values taken on by the parent nodes of X_i as $1, 2, \ldots, q_i$ where $q_i = |\Pi_i|$, then we may write

$$p(X_i = k | \Pi_i = j) = \theta_{i,j,k} \tag{2.3}$$

for

$$i = 1, 2, \ldots, n, \qquad j = 1, 2, \ldots, q_i, \qquad \text{and} \quad k = 1, 2, \ldots, r_i.$$

Now consider data consisting of m observations of the n nodes of a network, and restrict attention for a moment to the m values of the ith node. Consider any one configuration j of the parent nodes to node i that exists in the data. Let m_j be the number of times that the parents of node i take on configuration j in the data set. Then within the m_j values of $X_i|\Pi_i = j$, we can describe the number of observed 1's, 2's, and so on, up to the number of r_i's with a multinomial distribution with parameters m_j, r_i, and $(\theta_{i,j,1}, \theta_{i,j,2}, \ldots, \theta_{i,j,r_i})$.

In this way, using (2.2), we may write the likelihood for the entire $m \times n$ data set D as

$$L_D(\theta) = \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} \tag{2.4}$$

where n_{ijk} is the total number of times in the sample that X_i is observed to have value k when it's parents take on configuration j. (The likelihood is any function proportional to the joint pdf for m independent copies of $(X_1, X_2, \ldots, X_n)$ considered as a function of the θ's.) This is equivalent to the likelihood function associated with m copies of the random vector with density given by (2.1).

2.2.3 Priors for the Multinomial Network

The terminology *Bayesian network* derives from the application of Bayes rule in order to determine certain conditional probabilities. A study using Bayesian networks does not necessarily imply a Bayesian modeling approach. However, in the case that one wishes to use Bayesian inferential methods, it becomes necessary to assign prior distributions to the network parameters given by θ_{ijk}. Typically, for multinomial networks, one uses the conjugate prior given by the **Dirichlet distribution**. That is, we will assume that the joint density for the θ_{ijk} for a particular Bayesian network BN, is given by

$$p(\theta|BN) = \frac{\Gamma(\sum \alpha_{ijk})}{\prod \alpha_{ijk}} \prod \theta_{ijk}^{\alpha_{ijk}-1}$$

for some fixed hyperparameters $\alpha_{ijk} > 0$. Note that this is a high dimensional generalization of the more familiar Beta distribution. It is a convenient way to assign values between 0 and 1 to each θ_{ijk} in a way such that $\sum_k \theta_{ijk} = 1$. It is called a **conjugate** prior for the multinomial distribution because if the data given the θ_{ijk} follow a multinomial distribution and our "prior"

belief about the θ_{ijk} before observing the data is that they follow a Dirichlet distribution, then the the **posterior** joint distribution of the θ_{ijk} given the data (i.e. after we have observed the data) is another (different parameter) Dirichlet distribution. This is a mathematical convenience for Bayesian analysis.

With this Dirichlet prior, the probability, for any particular Bayesian network BN, of us seeing the data set D is

$$\begin{aligned} p(D|BN) &= \int\int p(D|BN,\theta)\cdot p(\theta|BN)\,d\theta \\ &= \int \prod_{i=1}^{n}\prod_{j=1}^{q_i^*}\theta_{ijk}^{n_{ijk}}\frac{\Gamma\left(\sum_{k=1}^{r_i}\alpha_{ijk}\right)}{\prod_{k=1}^{r_i}\Gamma\left(\alpha_{ijk}\right)}\prod_{k=1}^{r_i}\theta_{ijk}^{\alpha_{ijk}-1}\,d\theta \qquad (2.5) \\ &= \prod_{i=1}^{n}\prod_{j=1}^{q_i^*}\frac{\Gamma\left(\sum_{k=1}^{r_i}\alpha_{ijk}\right)}{\Gamma\left(\sum_{k=1}^{r_i}(\alpha_{ijk}+n_{ijk})\right)}\prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}+n_{ijk})}{\Gamma(\alpha_{ijk})} \end{aligned}$$

where q_i^* is the number of distinct configurations of parents of node i observed in the data. This is as opposed to q_i which is the total number of possible configurations of parents of node i, though q_i^* may be replaced by q_i since the lack of parent configuration j in the data will be reflected by n_{ijk} taking on the value 0.

2.3 Network Scores

There are many ways to recover networks from data. Indeed, we may not even want to think in terms of "one best network" and instead use a model averaging approach or one that constructs a best network by combining best "features" (for example, high scoring edges) from several networks. In the case that we are searching for the "best network" a penalized likelihood approach is commonly used.

2.3.1 Maximizing the Likelihood

Given m n-tuples of data points, $u_1, u_2, \ldots, u_m$, the likelihood function for a Bayesian network is given by

$$L_D(\theta) = \prod_{i=1}^{m} p(u_i) = \prod_{i,j,k} \theta_{ijk}^{n_{ijk}},$$

where n_{ijk} is the total number of times in the sample that X_i is observed to have value k when it's parents take on configuration j.

Given a particular DAG, we can estimate each θ with its maximum likelihood estimator

$$\hat{\theta}_{ijk} = \frac{\#\text{ observations with } X_i = k \text{ and } \Pi_i = j}{\#\text{ observations with } \Pi_i = j},$$

and then compute and compare the maximized likelihoods

$$L_D(\hat{\theta}) = \prod_{i,j,k} \hat{\theta}_{ijk}^{n_{ijk}}.$$

In the event that there are no observations where $\Pi_i = j$, we set $\hat{\theta}_{ijk} = 1$. However, it is important to note that we can always increase the likelihood by including additional θ parameters. Therefore, we will observe the greatest likelihoods ("most likely models") to coincide with DAGs with a maximal number of edges. Thus, the log-likelihood alone is not useful for recovering networks. However, it is the building block for other scoring criteria which generally include penalties for overparameterized models. The two most common penalized likelihood statistics are given by the following information criteria.

2.3.2 Akaike's Information Criterion (AIC)

Akaike's Information Criterion (AIC) is essentially a simple transformation of the above defined likelihood with a term included that penalizes for overparameterization. For theoretical reasons (In that it is an approximation to a Kullback-Leibler divergence distance between two probability distributions) it is defined by:

$$AIC = -2 \ln L_D(\hat{\theta}) + 2 \cdot (\#\text{ parameters}).$$

The goal then is to minimize the AIC to ensure a good fitting model in the sense of maximizing the log-likelihood while penalizing for having too many parameters.

2.3.3 Bayesian Information Criterion (BIC)

The Bayesian Information Criterion (BIC) is defined by

$$BIC = -2 \ln L_D(\hat{\theta}) + (\#\text{ parameters}) \cdot \ln(m),$$

where m is, as before, the sample size. As with the AIC, the goal is to minimize the BIC.

Both the AIC and BIC have rigorous justifications, from both Bayesian and frequentist points of view. AIC was derived from information theory, though it can be thought of as Bayesian if one uses a clever choice of prior. On the other hand, BIC, originally derived through Bayesian statistics as a measure of the Bayes factor, can also be derived as a non-Bayesian result. For more information on these widely used scoring criteria, please see [1],[2], [3], and [5] (AIC), and [19] and [21] (BIC). To make a broad generalization, the AIC tends to overfit the model in terms of number of parameters and the BIC tends to overpenalize, or underfit the model. This often causes BIC to choose a more simplistic model than AIC.

2.4 Network Recovery

In this section we illustrate recovery of a Bayesian network on a simple three node example. In this case we are able to easily evaluate the AIC and BIC scores for all possible DAGs. Since the number of possible DAGs increases super-exponentially as the number of nodes increases (see Table 2.1), evaluating the scoring criteria for every DAG can quickly become overwhelming. In these cases, it may become necessary to implement network space search methods. Examples of search methods are the greedy hill-climbing, stochastic hill climbing, simulated annealing, and Markov Chain Monte Carlo (MCMC).

Table 2.1: Number of Nodes Versus Number of DAGs

3 nodes	25 dags
4 nodes	543 dags
5 nodes	29,281 dags
6 nodes	3,781,503 dags
7 nodes	1,138,779,265 dags
8 nodes	783,702,329,343 dags

2.4.1 Simulating Data

Given a three node network with three random variables our goal is to recover the arrows (edges) that describe their joint probability distribution. The list of all 25 DAGs corresponding to 3 node networks can be found in Table 2.2 and we will refer to these DAGs as they are numbered here throughout this thesis.

We will assume that the data associated with each node is multinomial and they take on the values 1,2,3,4,5,6. In our previously used notation, this means r_1=r_2=r_3=6. In order to simulate data from network 8, Table 2.2 we need to specify the following probabilities.

$$\begin{array}{rclcrcl}
\theta_{1,1} & = & P(X_1 = 1), & \cdots & \theta_{1,5} & = & P(X_1 = 5) \\
\theta_{2,1,1} & = & P(X_2 = 1|X_1 = 1), & \cdots & \theta_{2,1,5} & = & P(X_2 = 5|X_1 = 1) \\
\theta_{2,2,1} & = & P(X_2 = 1|X_1 = 2), & \cdots & \theta_{2,2,5} & = & P(X_2 = 5|X_1 = 2) \\
\theta_{2,3,1} & = & P(X_2 = 1|X_1 = 3), & \cdots & \theta_{2,3,5} & = & P(X_2 = 5|X_1 = 3) \\
\theta_{3,1,1} & = & P(X_3 = 1|X_1 = 1), & \cdots & \theta_{3,1,5} & = & P(X_3 = 5|X_1 = 1) \\
\theta_{3,2,1} & = & P(X_3 = 1|X_1 = 2), & \cdots & \theta_{3,2,5} & = & P(X_3 = 5|X_1 = 2) \\
\theta_{3,3,1} & = & P(X_3 = 1|X_1 = 3), & \cdots & \theta_{3,3,5} & = & P(X_3 = 5|X_1 = 3) \\
\vdots & & & \cdots & \vdots & &
\end{array}$$

(Note that $\theta_{1,6} = 1-\sum_{i=1}^{5} \theta_{1,i}$ and $\theta_{i,j,6} = 1-\sum_{k=1}^{5} \theta_{i,j,k}$ for $i \in \{2,3,4,5,6\}$ and $j \in \{1,2,3,4,5,6\}$.)

For convenience, values for these probabilities were simulated from Dirichlet distributions with uniform hyperparameters on the interval $[0, 5]$.

We simulated (X_1, X_2, X_3) by assigning values to X_1 and then assigning values to X_2 and X_3 given X_1 according to the probabilities above. For this specific example we generated 100,000 values. Given this list of data, we used the AIC and BIC scoring mechanisms to select network 8 out of the 25 possible DAGs in Table 2.2. As mentioned in Chapter 1, AIC and BIC will not be able to distinguish between networks that are in the same Markov equivalent class, so if a data set scored by AIC and BIC recovers a network in the same Markov class it is considered successful. The Markov classes for the 25 DAGs listed in Table 2.2 are: {1}, {2,3}, {4,5}, {6,7}, {8,9,10}, {11,12,13}, {14}, {15}), {16,17,18}, {19}, {20,21,22,23,24,25}. Table 2.3 shows the AIC and BIC scores for all 25 DAGs and a successful recovery of DAG 8.

Table 2.2: Directed Acyclic Graphs on Three Nodes

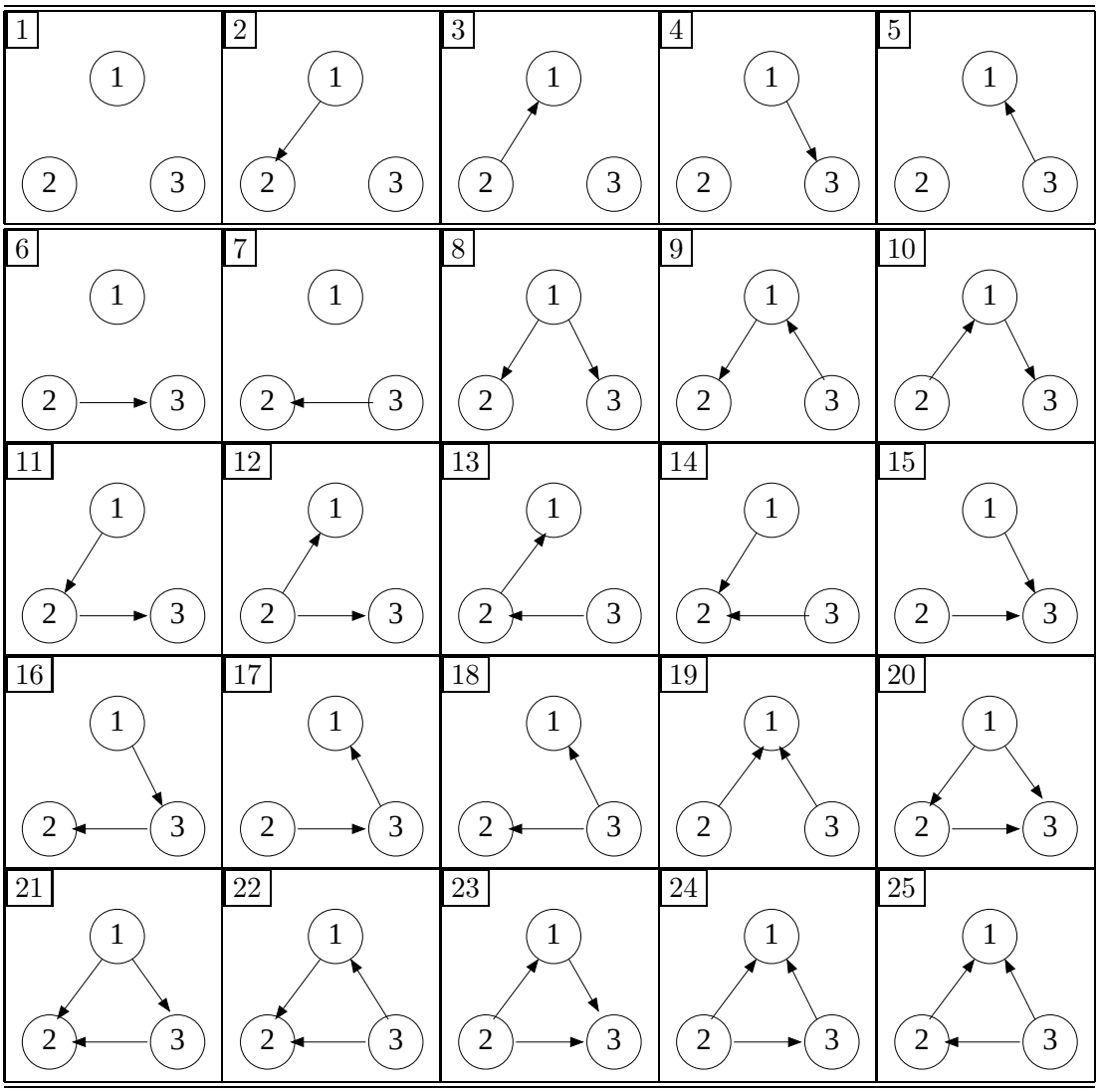


Table 2.3: AIC and BIC Recovery

Graph Number (n)	AIC	BIC
1	1040886.085	1041057.317
2	1001501.922	1001958.543
3	1001501.961	1001958.543
4	991255.256	991711.879
5	991255.259	991711.879
6	1036182.434	1036639.054
7	1036182.434	1036639.054
8	951871.096	952613.104
9	951871.096	952613.104
10	951871.096	952613.104
11	996798.271	997540.279
12	996798.271	997540.279
13	996798.271	997540.279
14	991435.745	936604.692
15	986551.608	987293.616
16	986551.608	987293.616
17	986551.608	987293.616
18	956755.233	958924.180
19	952051.582	954505.917
20	952051.582	954505.917
21	952051.582	954505.917
22	952051.582	954505.917
23	952051.582	954505.917
24	952051.582	954505.917
25	952051.582	954505.917

Chapter 3

Minimum Description Length

3.1 Introduction

As an alternative to AIC and BIC, the *minimum description length principle* (MDL principle) states that the best model is the one which allows for the shortest description, in the sense of encoding, of the data and model itself. With its origins in computer science and information theory, "description length" is the number of bits required to store such an encoding. Unlike AIC and BIC, the concept of minimum description length does not seem to be a familiar one to statisticians and mathematicians. Thus, we devote this chapter to a more in-depth explanation.

3.2 A Bit About Bits

The binary representation of an integer is a string of 0's and 1's which represent coefficients for terms that are powers of 2, starting with power 0 and read from the right. For example, the binary number

$$1101$$

is equal to the decimal number

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 13.$$

Table 3.1 shows the binary representations for the integers 0 through 16. Additional columns in this table show the number of digits used in the binary representations and the base 2 logarithm of the original integers in the cases where $\log_2$ is integer valued. The missing logs are in between

the consecutive integer log values. For example, $2 < \log_2 6 < 3$. The emerging pattern is that the binary representation of the integer n takes $O(\lceil \log_2 n \rceil)$ digits where $\lceil \cdot \rceil$ is the ceiling function. For the remainder of this thesis, we shall use log to denote $\log_2$. Also, we will omit the ceiling function as storing n using approximately $\log n$ bits will be sufficient for our purposes.

Table 3.1: Lengths of Binary Descriptions of Integers

Decimal Number (n)	Binary Representation	Number of Digits	$\log_2 n$
0	0	1	
1	1	1	0
2	10	2	1
3	11	2	
4	100	3	2
5	101	3	
6	110	3	
7	111	3	
8	1000	4	3
9	1001	4	
10	1010	4	
11	1011	4	
12	1100	4	
13	1101	4	
14	1110	4	
15	1111	4	
16	10000	5	4

3.3 Description Length for Bayesian Networks

Description length, or the "minimum description length (MDL) score", for a Bayesian network is the number of bits required to store the network, including the DAG, the variables, and the parameters, and to store the data. In this Section, we follow the construction of Friedman and Goldszmidt [10] to derive the MDL score.

3.3.1 Storing the Network

Storing The Variables

We need to store the number of variables and the number of possible values taken on by each variable. The number of possible values taken on by the random variable X_i will be denoted by $||X_i||$.

The number of variables is n. As described in Section 3.2, this number can be stored using $\log n$ bits. The integer $||X_i||$ can be stored using $\log ||X_i||$ bits for each of $i = 1, 2, \ldots, n$. So, the total variable storage contribution to the MDL score is

$$\log n + \sum_{i=1}^{n} \log ||X_i||.$$

Storing The DAG

We can describe the DAG by storing, for each of X_1 through X_n, the number of parents and a list of parents.

Recall that Π_i denotes the set of parents for X_i. Using $|\cdot|$ to denote the number of elements in a set, we have that X_i has $|\Pi_i|$ parents. This number takes $\log |\Pi_i|$ bits to store. As a simplification though, we will be conservative and say that it takes less than $\log n$ bits to store the number of parents for each random variable since $|\Pi_i| < n$.

Since the maximum node number is n, we will once again use the conservative value $\log n$ to store any node number. To store the list of parents for the node corresponding to X_i we use $\log n$ bits for each parent. That is, we use $\log n$ bits $|\Pi_i|$ times. Thus, we store the list of parents for X_i using at most $|\Pi_i| \cdot \log n$ bits.

The total conservative contribution of the DAG to the MDL score is

$$\sum_{i=1}^{n} (1 + |\Pi_i|) \log n.$$

Storing The Parameters

We now address storage of the θ_{ijk} parameters. While $|\Pi_i|$ represents the number of parents for X_i, we will use $||\Pi_i||$ to denote the number of configurations for the parent set for X_i. That is, if X_i has two parents, one taking on 3 possible values and the other taking on 5 possible values, $|\Pi_i| = 2$ but $||\Pi_i|| = 15$.

Since X_i takes on $||X_i||$ possible values for each parent configuration, we need to store $||X_i||-1$ parameters (since they will add up to 1) for each parent configuration. Therefore, we need to store $||\Pi_i||(||X_i|| - 1)$ parameters.

These parameters are not integers. According to Friedman and Goldszmidt, who cite [4] and [14], the "usual choice in the literature" is $\frac{1}{2}\log m$ bits per parameter where m is the number of n-dimensional data points.

The total parameter contribution to the MDL score is

$$\frac{1}{2}\log m \sum_{i=1}^{n} ||\Pi_i||(||X_i|| - 1).$$

In summary, the total description length (and contribution to the MDL score) **for the network** is

$$\begin{aligned}\widetilde{DL}_{net} &= \log n + \textstyle\sum_{i=1}^{n} \log ||X_i|| + \sum_{i=1}^{n}(1 + |\Pi i|)\ \log n + \frac{1}{2}\log m \sum_{i=1}^{n} ||\Pi_i||(||X_i|| - 1) \\ &= \log n + \textstyle\sum_{i=1}^{n} (\log ||X_i|| + (1 + |\Pi_i|)\ \log n) + \frac{1}{2}\log m \sum_{i=1}^{n} ||\Pi_i||(||X_i|| - 1).\end{aligned}$$

For the purpose of scoring and discretizing graphs that will be over a fixed number of variables, we will drop the constant $\log n$ term and define

$$DL_{net} = \sum_{i=1}^{n} (\log ||X_i|| + (1 + |\Pi_i|)\ \log n) + \frac{1}{2}\log m \sum_{i=1}^{n} ||\Pi_i||(||X_i|| - 1). \tag{3.1}$$

Note that this is not explicitly dependent on the θ parameters.

3.3.2 Storing the Data

Our data consists of m realizations of $U = (X_1, X_2, \ldots, X_n)$. We will denote the realizations as $u_1, u_2, \ldots, u_m$. Using Shannon coding (see Appendix A), the data point u_i is encoded in $-\log p(u_i)$ bits where $p(u) = p(x_1, x_2, \ldots, x_n)$ is given by (2.1). So, the description length of the data is

$$DL_{data} = -\sum_{i=1}^{m} \log p(u_i). \tag{3.2}$$

This is simply the negative log-likelihood for the model and is DAG dependent since the joint density is determined by the DAG.

3.3.3 Minimum Description Length as a Scoring Mechanism

The minimum description length score is for a network and data is defined as

$$MDL = DL_{net} + DL_{data}.$$

This score is similar to the AIC and BIC scores in that it is also a negative log-likelihood plus a term (DL_{net}) that penalizes for the number of parameters.

In this Section, we will use the MDL score to recover a network structure using simulated data from DAG 8 in Table 2.2. Each of X_1, X_2, and X_3 were assumed to take on values in $\{1, 2, \ldots, 6\}$, and parameters for DAG 8 were drawn from the Dirichlet distribution using hyperparameters that were uniformly distributed on the interval $[0, 5]$. (This is the same data used in Section 2.4.1.)

Results are shown in Table 3.2 along with the previously computed values of AIC and BIC. We see that the AIC, BIC, and MDL scores all recovered the correct network up to the Markov equivalence class.

Table 3.2: AIC, BIC and MDL Recovery

Graph Number (n)	AIC	BIC	MDL
1	1040886.085	1041057.317	520443.714
2	1001501.922	1001958.543	500885.425
3	1001501.961	1001958.543	500885.425
4	991255.256	991711.879	495762.093
5	991255.259	991711.879	495762.093
6	1036182.434	1036639.054	518225.681
7	1036182.434	1036639.054	518225.681
8	951871.096	952613.104	476213.804
9	951871.096	952613.104	476213.804
10	951871.096	952613.104	476213.804
11	996798.271	997540.279	498677.392
12	996798.271	997540.279	498677.392
13	996798.271	997540.279	498677.392
14	991435.745	936604.692	496536.904
15	986551.608	987293.616	493554.060
16	986551.608	987293.616	493554.060
17	986551.608	987293.616	493554.060
18	956755.233	958924.180	479196.648
19	952051.582	954505.917	476988.615
20	952051.582	954505.917	476988.615
21	952051.582	954505.917	476988.615
22	952051.582	954505.917	476988.615
23	952051.582	954505.917	476988.615
24	952051.582	954505.917	476988.615
25	952051.582	954505.917	476988.615

Chapter 4

Minimum Description Length For Discretization

Friedman and Goldszmidt [10] have developed a method for discretization of continuous data for Bayesian networks which is based on the minimum description length principle. Essentially, the MDL score is augmented with the description length necessary to be able to recover the original data set from discretized data. The Friedman and Goldszmidt paper stands out among others in this area as one of the only to suggest a rigorous approach to the problem. We begin this Chapter by describing this MDL discretization and then give some improvements to take the method from interesting to feasible. Examples in this Chapter, we will focus the "discretization" of random variables that are already discrete. That is, we will categorize or group together values into a smaller number of values. In Chapter 5, we will consider truly continuous data.

4.1 Description Length for Discretization

A discretization is a mapping from the range of values in the original data set to the set $\{1, 2, \ldots, k\}$ for some $k \geq 1$. It will be described by $k - 1$ "thresholds" .

For simplicity, we will refer to node i and the random variable X_i interchangeably. Also, for simplicity, we will assume at this point that node i is the only continuous one in the network and that the others are discrete or have already gone through a discretization process. Furthermore, we assume that X_i takes on m_i distinct values in the data set with m n-dimensional points. (Clearly $m_i \leq m$ with equality in the case of truly continuous data.) The discretized version of X_i will be denoted by X_i^* and we will use k_i to denote the number of values taken on by X_i^*.

Friedman and Goldszmidt augment the MDL score with description lengths for

- the discretization rule which consists of thresholds for mapping data to $\{1, 2, \ldots, k\}$,
- the description of the discretized data, and
- the description of the original data set based on the discretized data set.

For a fixed DAG and a fixed threshold assignment, we proceed as follows.

Description Length for the Discretization Rule

The number of thresholds used in the discretization can be determined by looking at the discretized data, which will be encoded in the description of the discretized data. Here, we will include only the values of the thresholds. A useful observation is that, as we only need to distinguish between values in the data set, we may choose thresholds from among the set of $m_i - 1$ midpoints in the set of distinct values for X_i. In order to get a discretization with k_i values, we need to choose $k_i - 1$ thresholds from the set of $m_i - 1$ thresholds. There are $\left(\begin{smallmatrix} m_i - 1 \\ k_i - 1 \end{smallmatrix} \right)$ such discretization policies which may be indexed as 1 through $\left(\begin{smallmatrix} m_i - 1 \\ k_i - 1 \end{smallmatrix} \right)$.

Storing this index will take at most $\log \left(\begin{smallmatrix} m_i - 1 \\ k_i - 1 \end{smallmatrix} \right)$ bits. In order to achieve a simplification towards the end of this Section, we note [7] the inequality

$$\log \begin{pmatrix} m_i - 1 \\ k_i - 1 \end{pmatrix} \leq (m_i - 1) H \left(\frac{k_i - 1}{m_i - 1} \right)$$

where $H(\cdot)$ is the **entropy function** for the Bernoulli distribution which is defined as

$$H(p) = -p \log p - (1 - p) \log(1 - p).$$

In summary, a conservative contribution of the discretization rule to the augmented MDL score is

$$DL_{DR} = (m_i - 1) H \left(\frac{k_i - 1}{m_i - 1} \right).$$

Description Length for the Discretized Data

Once the data has been discretized, its description can be stored in MDL bits where MDL is the original score defined in Chapter 3. We will denote this as

$$MDL^* = DL^*_{net} + DL^*_{data}$$

where DL^*_{net} is given by (3.1) and DL^*_{data} is given by (3.2).

Description Length for Information Needed to Recover the Original Data Set from the Discretized Data

We begin with a small illustrative example. Suppose we have a small data set of 10 n-dimensional values and that the values to be discretized for X_i are, in order,

$$0.17, 0.23, 0.46, 1.29, 1.78, 2.13, 2.44, 2.59, 2.99, 3.07.$$

Further suppose that these values have been discretized as follows.

$$\underbrace{0.17, 0.23, 0.46}_{1}, \underbrace{1.29, 1.78}_{2}, \underbrace{2.13, 2.44, 2.59, 2.99, 3.07}_{3}.$$

If we observe a discretized value of 3, for example, we know that the original value must have been one of the values in the set $\{2.13, 2.44, 2.59, 2.99, 3.07\}$. We can encode which one of these values appears in the specific instance of X_i we are observing. By using a Shannon coding of values based on the frequencies in the entire data set (as opposed to the frequencies in the 5 element example set), we can exclude storage space for the frequencies in the description length discretization scoring metric as it will be the same for all discretizations. Furthermore (see Appndix A), for a particular value of X^*_i, we can encode the value of X_i using approximately $-\log \widehat{P}(X_i|Xi^*)$ bits.

Thus, the contribution of the continuous data reconstruction to the augmented MDL score is

$$DL_{recover} = -\sum_{i=1}^{m} \log \hat{P}(X_i|Xi^*).$$

In summary, the discretization score is defined as

$$DL_{DR} + MDL^* + DL_{recover}. \tag{4.1}$$

As Friedman and Goldszmidt point out, there are many terms in the discretization score that remain constant as the discretization of X_i changes. Thus, to simplify computations, they consider only the terms in (4.1) that are affected by a change in discretization. After considerable simplification of these terms, they have defined a "DL Local" score to be

$$\begin{aligned} DL_{local} &= \tfrac{1}{2}\log m \cdot ||\Pi_i||(||X_i^*|| - 1) + \tfrac{1}{2}\log m \cdot \textstyle\sum_{j:X_i \in \Pi_j} ||\Pi_j||(||X_j|| - 1) \\ &+ \log k_i + (m_i - 1)H\left(\tfrac{k_i - 1}{m_i - 1}\right) - m\left[\widehat{I}(X_i^*, \Pi_i) + \textstyle\sum_{j:X_i \in \Pi_j} \widehat{I}(X_j, \Pi_j)\right] \end{aligned} \tag{4.2}$$

where $\widehat{I}(\vec{X}, \vec{Y})$ is the estimated **mutual information**

$$\widehat{I}(\vec{X}, \vec{Y}) = \sum_{\vec{x},\vec{y}} \widehat{P}(\vec{x}, \vec{y}) \log \frac{\widehat{P}(\vec{x}, \vec{y})}{\widehat{P}(\vec{x})\widehat{P}(\vec{y})}.$$

The chosen discretization of X_i should be the one that minimizes DL_{local}.

4.2 Searching for Discretizations

A discretization of X_i involves putting thresholds between values in the data set. In the case of $m_i \leq m$ distinct values, there are

$$\begin{pmatrix} m_i - 1 \\ 0 \end{pmatrix} + \begin{pmatrix} m_i - 1 \\ 1 \end{pmatrix} + \ldots \begin{pmatrix} m_i - 1 \\ m_i - 1 \end{pmatrix} = 2^{m_i - 1}$$

different discretizations to consider. Clearly, this number can get quite large for large data sets with truly continuous ($m_i = m$) data points. We now illustrate the enumeration of discretizations in a toy case of categorization ("discretizing discrete data") where they can be explicity listed. We choose a node with 6 distinct values ($m_i = 6$). Based on the section 4.1 we know that there are

$$\begin{pmatrix} m_i - 1 \\ k_i - 1 \end{pmatrix}$$

discretization policies that result in k_i categories. The total number of possible discretizations is $2^5 = 32$.

We will use bars between the 6 numbers to show thresholds for discretization. For example, writing

$$12|3|456$$

will imply a three category discretization where the values for X_i in the data will be reassigned as

$$\left.\begin{matrix}1\\2\end{matrix}\right\} \to 1, \qquad 3 \to 2, \text{ and} \qquad \left.\begin{matrix}4\\5\\6\end{matrix}\right\} \to 3.$$

For $k_i = 1$ categories,

$$\begin{pmatrix} 5 \\ 0 \end{pmatrix} = 1.$$

The only possible discretization is to have all values in the same category. This is written as

$$123456$$

and implies (as there are no bars) that all values will be mapped to the value 1. This will be labeled as "discretization 1".

For $k_i = 2$ categories,

$$\begin{pmatrix} 5 \\ 1 \end{pmatrix} = 5.$$

The 5 possible discretizations are:

$$\begin{aligned} &1|23456 && \text{(discretization 2)}\\ &12|3456 && \text{(discretization 3)}\\ &123|456 && \text{(discretization 4)}\\ &1234|56 && \text{(discretization 5)}\\ &12345|6 && \text{(discretization 6)} \end{aligned}$$

For $k_i = 3$ categories, the

$$\begin{pmatrix} 5 \\ 2 \end{pmatrix} = 10$$

possible discretizations are:

1|2|3456	(discretization 7)
1|23|456	(discretization 8)
1|234|56	(discretization 9)
1|2345|6	(discretization 10)
12|3|456	(discretization 11)
12|34|56	(discretization 12)
12|345|6	(discretization 13)
123|4|56	(discretization 14)
123|45|6	(discretization 15)
1234|5|6	(discretization 16)

For $k_i = 4$ categories, the

$$\left(\begin{array}{c} 5 \\ 3 \end{array} \right) = 10$$

possible discretizations are:

1|2|3|456	(discretization 17)
1|2|34|56	(discretization 18)
1|2|345|6	(discretization 19)
1|23|4|56	(discretization 20)
1|23|45|6	(discretization 21)
1|234|5|6	(discretization 22)
12|3|4|56	(discretization 23)
12|3|45|6	(discretization 24)
12|34|5|6	(discretization 25)
123|4|5|6	(discretization 26)

For $k_i = 5$ categories, the

$$\left(\begin{array}{c} 5 \\ 4 \end{array} \right) = 5$$

possible discretizations are:

$$1|2|3|4|56 \qquad \text{(discretization27)}$$

$$1|2|3|45|6 \qquad \text{(discretization 28)}$$

$$1|2|34|5|6 \qquad \text{(discretization 29)}$$

$$1|23|4|5|6 \qquad \text{(discretization 30)}$$

$$12|3|4|5|6 \qquad \text{(discretization 31)}$$

Finally, for $k_i = 6$, there is

$$\left(\begin{array}{c} 5 \\ 5 \end{array} \right) = 1$$

possible discretization which is

$$1|2|3|4|5|6 \qquad \text{(discretization 32)}$$

In Section 4.2.1, using an example of a 3 node network with simulated data, we will search these 32 discretizations for the minimum DL_{local} score. In general though, it is not feasible to search all possible discretizations. Friedman and Goldszmidt recommend starting a search with no breaks and trying to insert breaks that will reduce the local description length. In this thesis, we instead recommend a "top-down" search strategy, described in Section 4.2.3, as a **much** more efficient alternative and show that it will yield the "correct" discretization. In practice, the "correct" discretization of continuous data will be the one for which a graph recovery method, for example using AIC, BIC, or MDL, will yield the generating DAG. However, for testing the DL_{local} discretization approach of Friedman and Goldszmidt and our "top-down" search strategy, we will ensure that there is truly a correct discretization by simulating discrete data from a Bayesian network and then "exploding" values into several randomly selected values.

4.2.1 Local Description Length Score: One Network

We simulated $100,000$ triples from the network represented by DAG 8 (Table 2.2) with 3 possible values for each node. The θ_{ijk} parameters were simulated from the Dirichlet distribution with

hyperparameters simululated from the uniform distribution on the interval $[0, 5]$. We considered the discretization problem at node 1 by "exploding" the data for node one as follows.

- Whenever node 1 took on the value 1, we randomly reassigned the value to be 1 or 2.
- Whenever node 1 took on (originally) the value 2, we randomly assigned the value to be 3, 4, or 5.
- Whenever node 1 took on (originally) the value 3, we reassigned the value to be 6.

Node 1 now takes on values in $\{1, 2, 3, 4, 5, 6\}$ though 1 and 2 are indistinguishable and 3, 4, and 5 are indistinguishable. There are 32 possible discretizations ennumerated in Section 4.2. We calculated the local description length for all possible discretizations. The results are shown in Table 4.1. We observe that the local description length returned the smallest value for the correct discretization.

4.2.2 Local Description Length: Any Network

The DL_{local} score is DAG dependent. In Section 4.2.1, we recovered the correct discretization, but made all DL_{local} computations using the generating DAG 8. Since discretization is a precursor step to network recovery, we will not have the luxury of knowing the generating DAG. Friedman and Goldszmidt suggest searching over the DAG space in addition to the different discretizations. However, we will see that this Herculean task may not be necessary.

In Table 4.2 we calculated the local description length score for all possible DAGs and discretizations (using the same simulated data as above). What we find is that the lowest local description length score corresponds to the correct discretization for all DAGs with the exception of DAGs 1,6, and 7. Recall that we are working with a discretization of node 1 and note that DAGs 1,6, and 7 are the only DAGs where node 1 is not connected to any other nodes. In short, node 1 is isolated. As long as all the nodes are connected to the network by at least one edge, the lowest local description length calculated for all DAGs corresponds to discretization 13. Thus, no matter which discretization or network the data is generated from, it appears that one can calculate the

Table 4.1: Local Description Length Score

Discretization Number	Discretization	DL local score
1	123456	33.22
2	1|23456	-6191.66
3	12|3456	-18474.07
4	123|456	-15129.52
5	1234|56	-13830.91
6	12345|6	-15116.82
7	1|2|3456	-18439.89
8	1|23|456	-15795.48
9	1|234|56	-15747.67
10	1|2345|6	-18416.07
11	12|3|456	-21209.95
12	12|34|56	-24645.59
13	12|345|6	-29929.17
14	123|4|56	-18535.49
15	123|45|6	-23819.33
16	1234|5|6	-19081.35
17	1|2|3|456	-21177.18
18	1|2|34|56	-24612.82
19	1|2|345|6	-29896.40
20	1|23|4|56	-19202.87
21	1|23|45|6	-24486.72
22	1|234|5|6	-20999.53
23	12|3|4|56	-24617.33
24	12|3|45|6	-29901.18
25	12|34|5|6	-29897.45
26	123|4|5|6	-23787.35
27	1|2|3|4|56	-24585.90
28	1|2|3|45|6	-29896.74
29	1|2|34|5|6	-29866.02
30	1|23|4|5|6	-24456.07
31	12|3|4|5|6	-29870.53
32	1|2|3|4|5|6	-29841.52

local description length for any DAG that has all the nodes connected with at least one edge. The resulting lowest local description length corresponds to the correct discretization for the data. To see more examples of this refer to Appendix B.

Although local description length is a powerful calculation as shown above, it is too cum-

bersome to calculate for all possible discretizations, even if only for one network. In an effort to focus our search, we introduce a "top-down" approach to try and reduce the number of calculations required. (Our top-down method starts with the maximum thresholds as a discretization policy. This is the opposite of the top-down method defined by Goldszmidt and Friedman in which they start with no thresholds.)

4.2.3 The "Top-Down Approach"

The "top-down approach" is a search strategy that removes threshold for the discretization one at a time as long as they decrease the DL_{local} score. Formally, the search algorithm is as follows.

Top-Down Search Algorithm

Let $k = k_i$ be the number of distinct values taken on by X_i in the original data set.

Compute the DL_{local} score with all $k-1$ thresholds in place. This is known as the "full DL_{local}" score.

Assign the variable M to be the full DL_{local} score.

(1) Compute the set, $\{D_1, D_2, \ldots, D_{k-1}\}$ of DL_{local} scores where D_i corresponds to the score with the ith threshold removed.

(2) If $\min(D_1, D_2, \ldots, D_{k-1}) < M$, set $M = \min(D_1, D_2, \ldots, D_{k-1})$, set $k = k-1$, remove the minimal scoring threshold, and return to Step 1.

If $\min(D_1, D_2, \ldots, D_{k-1}) \geq M$ or if there are no more thresholds, stop.

For example, consider a node with 3 distinct values. The data is then exploded such that if $X_1 = 1$ the values in the new data set are $X_1 \in \{1, 2\}$. Similarly, if $X_1 = 2$ the values were exploded such that $X_1 \in \{3, 4, 5\}$ and if $X_1 = 3$ the value was changed to $X_1 = 6$. We then calculate the local description length with all thresholds. For notational purposes we number the thresholds or

breaks from 1 to k_i (the maximum number of thresholds) from left to right by removing them one at a time. The results are recorded in Table 4.3.

We see from the table that the local description length calculation corresponding to threshold 4 is the smallest of the 5 thresholds. We then compare this to the local description length calculated with all breaks inserted and we see that threshold 4 is smaller. Thus, we remove threshold 4 and the following discretization remains 1|2|3|45|6. Because a break was removed, we recalculate the local description length by removing one break at a time but we begin with the discretization above that has the break removed between the 4 and the 5. Table 4.4 shows the results. Of note here is that we renumber the breaks from left to right based on the single break that is being removed at very iteration.

We see that the lowest local description length corresponds to threshold 1 and it is lower than the local description length calculated with all breaks inserted. Therefore, we remove the break between the 1 and 2. What remains is the discretization corresponding to 12|3|45|6. We now repeat the process for the 3 remaining thresholds. The results are shown in Table 4.5.

Again, we note that threshold 2 is the lower local description length and that it is also lower than the local description length with all thresholds inserted. Thus, we remove break 2 and the remaining discretization is 12|345|6. Table 4.6 displays the next iteration.

In this case we note that the lowest local description length corresponds to threshold 2 but it is not lower than the local description length calculated with all thresholds inserted. This stops the iteration and the remaining discretization is 123|45|6. This is exactly discretization policy that the data was created from.

In general, the number of times that the local description length is calculated is

$$(m_i - 1) + (m_i - 2) + \ldots + 1 = \frac{m_i(m_i - 1)}{2}.$$

4.2.4 The Single Iteration Top-Down Approach

Interestingly, there is a way to even further reduce the number of calculations required to find the optimal discretization policy. In looking at the first iteration of the local description length calculation in the example above (See Table 4.3), we note that although the local description length value corresponding to threshold 4 is the smallest, the values for thresholds 1 and 3 are also smaller than the local description length values calculated with all thresholds inserted. So, for a discretization consisting of all breaks (1|2|3|4|5|6) we remove thresholds 1, 3, and 4 and what remains of the discretization policy is 12|345|6. Amazingly, for the example begining in scetion 4.2.3 this is also the correct discretization policy in which the data was generated. All we may need to calculate is the first iteration and compare all the calculated values by removing one break at a time (and then replacing it) with the local description length where all breaks are inserted. Any removed threshold corresponding to a local description length that is less than the local description length with all breaks inserted is subsequently removed. What remains is the correct discretization.

One iteration of the single iteration top-down method requires m_i calculations.

4.3 Conjecture

To generalize the findings in the simulation above we summarize with the following statement.

Conjecture:

Given a data set with m_i distinct values, calculate the local description length with all k_i thresholds inserted. Then calculate the local description length while removing one break at a time, resulting in $k_i - 1$ more calculations. Compare all the values returned. If the local description length with a break removed is smaller than the value returned for local description length with all thresholds inserted, remove this break from the discretization with all thresholds inserted. What remains is the optimal discretization.

To show this consider a node with 6 distinct values and consider the role of mutual information in the calculation of local description length. In this example the break between the 3 and the 4 is to be removed. Lets look specifically at the mutual information in the case where the discretization policy has all 6 distinct values and compare that to the mutual information for the policy that has the break between the 3 and the 4 removed, thus only 5 distinct values. The mutual informations is calculated as follows:

$$I_6 = \sum_{k=1}^{6}\sum_{y=1}^{3} P(x,y)\log\left(\frac{P(x,y)}{P(x)P(y)}\right)$$

$$I_5 = \sum_{k=1}^{5}\sum_{y=1}^{3} P(x,y)\log\left(\frac{P(x,y)}{P(x)P(y)}\right)$$

In writing out the terms, the difference is:

$$I_6 - I_5 = \sum_{y=1}^{3} P(3,y)\log\left(\frac{P(3,y)}{P(3)\cdot P(y)}\right) + P(4,y)\log\left(\frac{P(4,y)}{P(4)\cdot P(y)}\right) - \sum_{y=1}^{3} P^*(3,y)\log\left(\frac{P^*(3,y)}{P^*(3)P(y)}\right)$$

Since we assumed that the break should be removed, the chance of seeing a 3 is equally likely as seeing a 4. The following equalities are result.

$$P^*(3,y) = P(3,y) + P(4,y)$$

$$P(3,y) = P(4,y)$$

$$P^*(3,y) = 2\cdot P(3,y)$$

When substituting these equalities into the difference in mutual information above, it is noted that all terms cancel and the result is that there is *no change in mutual information*! Additionally, it is worth noting that the terms in front of the mutual information depend on the structure and size of the discretization policy and not the relationship between variables. In the case where we calculate the local description length with one break removed, this term is constant between all k_i calculations.

These findings are significant when identifying that the mutual information term dominates the local description length calculation because it is multiplied by N. The other terms is the calculation are significantly small or multiplied by the $\log N$, which is grows much more slowly than

N. It is clear from the simulated data that the mutual information term dominates the calculation as well because all the values for local description length are large negative numbers. Thus, we calculate the local description length with all breaks inserted. Then perform the calculation with a break that should be removed, and we will see a smaller (more negative number) because the terms in front of the mutual information are reduced as we are evaluating local description length at $k_i - 1$ thresholds compared to k_i thresholds.

In contrast, if a break is not to be removed there is a change in the mutual information. In this case, the mutual information would be reduced because the variables are exhibiting less mutual dependence. The contribution to the local description length is smaller. By subtracting a smaller number we produce a larger local description length as compared to the local description length with all thresholds inserted, which is not desired. Thus, the break will remain in the discretization policy.

Based on the logic above, when calculating the local description length for one break that should be removed we should see the same value for local description length when calculating any other single break that should be removed as there is no change in mutual information. This was not seen in the data given in Table 4.3. We attribute this difference to sample variability. To illustrate the variability in the sample, we calculated the mutual information for 100 different data sets produced by DAG 8. The results are shown in Figure 4.1.

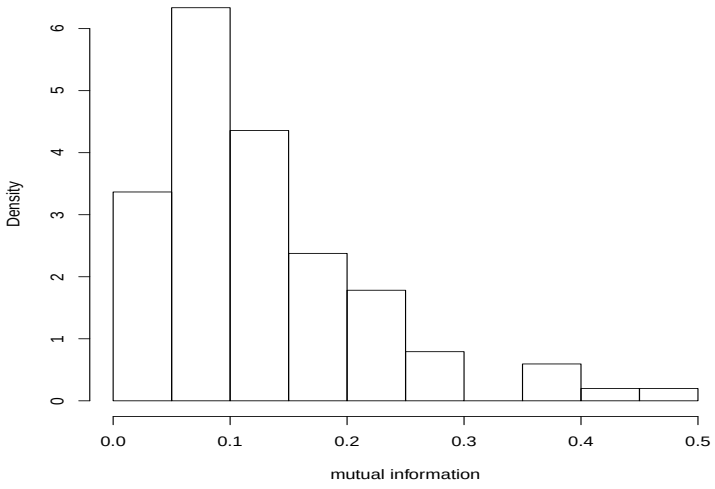
Mutual information is a measure of the mutual dependence between two variables. Because the data was generated from the same DAG, the same conditional dependency between variables was established. Thus, we expect mutual information to be the same throughout the samples. The histogram shows that this is not the case and that sampling variability exists.

4.4 Example

To conclude with another example, consider a graph with 3 distinct values. These values were then exploded into 6 distinct values such that the following is the correct discretization policy: 12|345|6. See Table 4.7 for the results of the local description length calculation.

In this case, the thresholds corresponding to 1,3, and 4 are removed. This returns the correct discretiation in 6 calculations of local description length. Further, we note that the local description length values for thresholds 1,3,and 4 are not equal as we might expect. This is due to sampling variance described above. Additional examples of the single iteration top-down method are provided in Appendix B.

Figure 4.1: 100 Independent Values of Estimated Mutual Information Between Nodes X_1 and X_2 for Graph 8 Based on Samples of Size 100,000



4.5 Discretization of Several Variables

All the work above has focused on one variable and it is not our intention to extend this work to several variables at this time. If required, we recommend that you perform Goldszmidt and Friedman's procedure. Mainly, select a variable X_i and find its optimal discretization while treating all other variables as discrete. Then move to a new variable X_{i+1} and perform a discretization in the same manner. Continue to discretize until you cannot improve the discretization of any variables. Because of the inherent dependencies between variables in Bayesian networks, it will require several iterations through all variables before the optimal discretization is discovered.

Table 4.2: Local Description Length Calculated for Three Node DAGs (NOTE: Graphs 16-25 were removed from the table as they are redundant and only graphs 1-15 are needed to illustrate the findings)

Disc Number	Graph Number														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.00	16.61	0.00	16.61	0.00	0.00	0.00	33.22	33.22	33.22	16.61	0.00	0.00	-628.81	-628.81
2	4.61	-948.15	-956.45	-5238.90	-5247.21	4.61	4.61	-6191.66	-6191.96	-6191.96	-948.15	-956.45	-956.45	-1445.92	-5736.68
3	4.61	-2955.45	-2963.76	-15514.01	-15522.31	4.61	4.61	-18474.07	-18474.37	-18474.37	-2955.45	-2963.76	-2963.76	-3319.21	-15877.77
4	4.61	-3889.27	-3897.58	-11235.63	-11243.94	4.61	4.61	-15129.52	-15137.82	-15137.82	-3889.28	-3897.58	-3897.58	-4278.37	-11624.73
5	4.61	-5777.49	-5785.79	-8048.82	-8057.13	4.61	4.61	-13830.91	-13839.22	-13839.22	-5777.48	-5785.79	-5785.79	-6266.01	-8537.34
6	4.61	-9371.57	-9379.89	-5740.62	-5748.93	4.61	4.61	-15116.82	-15125.12	-15125.12	-9371.59	-9379.89	-9379.89	-10005.70	-6374.74
7	6.44	-2937.45	-2937.45	-15495.99	-15495.99	6.44	6.44	-18439.89	-18439.89	-18439.89	-2937.45	-2937.45	-2937.45	-3271.20	-15829.74
8	6.44	-4050.53	-4050.53	-11738.51	-11738.52	6.44	6.44	-15795.48	-15795.48	-15795.48	-4050.53	-4050.53	-4050.53	-4470.99	-12158.97
9	6.44	-6235.95	-6235.95	-9505.29	-9505.29	6.44	6.44	-15747.67	-15747.67	-15747.67	-6235.94	-6235.95	-6235.95	-6736.71	-10006.05
10	6.44	-10161.89	-10161.89	-8247.73	-8247.73	6.44	6.44	-18416.07	-18416.07	-18416.07	-10161.89	-10161.89	-10161.89	-10723.36	-8809.20
11	6.44	-5240.87	-5240.87	-15962.63	-15962.63	6.44	6.44	-21209.95	-21209.95	-21209.95	-5240.87	-5240.87	-5240.87	-5428.91	-16150.68
12	6.44	-8102.47	-8102.47	-16536.68	-16536.69	6.44	6.44	-24645.59	-24645.59	-24645.59	-8102.47	-8102.47	-8102.47	-8128.92	-16563.14
13	6.44	-12561.32	-12561.32	-17361.40	-17361.40	6.44	6.44	-29929.17	-29929.17	-29929.17	-12561.32	-12561.32	-12561.32	-12467.46	-17267.54
14	6.44	-6737.04	-6737.04	-11792.01	-11792.01	6.44	6.44	-18535.49	-18535.49	-18535.49	-6737.04	-6737.04	-6737.04	-6933.35	-11988.33
15	6.44	-11195.81	-11195.81	-12617.09	-12617.09	6.44	6.44	-23819.33	-23819.33	-23819.33	-11195.81	-11195.81	-11195.81	-11271.44	-12692.72
16	6.44	-10219.31	-10219.31	-8855.60	-8855.60	6.44	6.44	-19081.35	-19081.35	-19081.35	-10219.31	-10219.31	-10219.31	-10556.54	-9192.83
17	6.85	-5224.29	-5224.29	-15946.03	-15937.73	6.85	6.85	-21177.18	-21168.87	-21168.87	-5224.29	-5215.98	-5215.98	-5382.31	-16104.10
18	6.85	-8085.88	-8085.88	-16520.09	-16511.78	6.85	6.85	-24612.82	-24604.52	-24604.52	-8085.88	-8077.58	-8077.58	-8082.32	-16516.53
19	6.85	-12544.74	-12536.43	-17344.80	-17336.50	6.85	6.85	-29896.39	-29888.09	-29888.09	-12544.74	-12536.43	-12536.43	-12420.86	-17220.92
20	6.85	-6899.70	-6891.40	-12296.31	-12288.00	6.85	6.85	-19202.87	-19194.57	-19194.57	-6899.70	-6891.40	-6891.40	-7127.38	-12523.99
21	6.85	-11358.47	-11350.17	-13121.39	-13113.08	6.85	6.85	-24486.72	-24478.41	-24478.41	-11358.47	-11350.17	-11350.17	-11465.47	-13228.38
22	6.85	-10679.19	-10670.88	-10313.48	-10305.18	6.85	6.85	-20999.53	-20991.22	-20991.22	-10679.19	-10670.88	-10670.88	-11028.66	-10662.96
23	6.85	-8090.05	-8081.74	-16520.43	-16512.12	6.85	6.85	-24617.33	-24609.03	-24609.03	-8090.05	-8081.74	-8081.74	-8085.31	-16515.69
24	6.85	-12548.82	-12540.51	-17345.50	-17337.20	6.85	6.85	-29901.18	-29892.87	-29892.87	-12548.82	-12540.51	-12540.51	-12423.40	-17220.09
25	6.85	-12545.71	-12537.40	-17344.89	-17336.58	6.85	6.85	-29897.45	-29889.14	-29889.14	-12545.71	-12537.40	-12537.40	-12420.87	-17220.05
26	6.85	-11180.28	-11171.98	-12600.21	-12591.91	6.85	6.85	-23787.35	-23779.04	-23779.04	-11180.28	-11171.98	-11171.98	-11225.30	-12645.24
27	5.93	-8074.80	-8058.19	-16505.17	-16488.56	5.93	5.93	-24585.90	-24569.29	-24569.29	-8074.80	-8058.19	-8058.19	-8040.05	-16470.41
28	5.93	-12533.57	-12516.96	-17330.24	-17313.63	5.93	5.93	-29869.74	-29853.13	-29853.13	-12533.57	-12516.96	-12516.96	-12378.14	-17174.81
29	5.93	-12530.46	-12513.85	-17329.62	-17313.01	5.93	5.93	-29866.02	-29849.41	-29849.41	-12530.46	-12513.85	-12513.85	-12375.61	-17174.77
30	5.93	-11344.29	-11327.68	-13105.85	-13089.24	5.93	5.93	-24456.07	-24439.46	-24439.46	-11344.29	-11327.68	-11327.68	-11420.67	-13812.23
31	5.93	-12534.63	-12518.02	-17329.97	-17313.36	5.93	5.93	-29870.53	-29853.92	-29853.92	-12534.63	-12518.02	-12518.02	-12378.60	-17173.94
32	2.58	-12521.81	-12496.89	-17317.13	-17292.21	2.58	2.58	-29841.52	-29816.60	-29816.60	-12521.81	-12496.89	-12496.89	-12335.76	-17131.08

Table 4.3: Local Description Length Score: Iteration 1

Threshold Number	Corresponding Discretization	DL local score
All k_i	1|2|3|4|5|6	-24528.36
1	12|3|4|5|6	-24555.96
2	1|23|4|5|6	-22135.06
3	1|2|34|5|6	-24557.09
4	1|2|3|45|6	-24559.81
5	1|2|3|4|56	-21229.63

Table 4.4: Local Description Length Score: Iteration 2

Threshold Number	Corresponding Discretization	DL local score
Min Threshold	1|2|3|45|6	-24559.81
1	12|3|45|6	-24589.32
2	1|23|45|6	-21772.70
3	1|2|345|6	-24586.56
4	1|2|3|456	-24262.39

Table 4.5: Local Description Length Score: Iteration 3

Threshold Number	Corresponding Discretization	DL local score
Min Threshold	12|3|45|6	-24589.32
1	123|45|6	-21645.78
2	12|345|6	-24619.02
3	12|3|456	-21294.84

Table 4.6: Local Description Length Score: Iteration 4

Threshold Number	Corresponding Discretization	DL local score
Min Threshold	12|345|6	-24645.78
1	12345|6	-19376.42
2	12|3456	-19306.80

Table 4.7: Local Description Length Score: Example 2

Threshold Number	Corresponding Discretization	DL local score
All k_i	1|2|3|4|5|6	-29841.52
1	12|3|4|5|6	-29870.53
2	1|23|4|5|6	-24456.07
3	1|2|34|5|6	-29866.02
4	1|2|3|45|6	-29896.74
5	1|2|3|4|56	-24585.90

Chapter 5

Continuous Data

Until this point, we have only considered examples of classification or "discretization of discrete" data. In this Chapter, we focus on truly continuous data. As before, we will use an example with 3 nodes and only one continuous variable, with the assumption (see Section 4.5) that the other nodes have already been discretized.

5.1 An Example

Again using DAG 8, we simulated 100,000 values of $X_1 \sim N(0,1)$ and then simulated X_2 and X_3 from independent discrete distributions on $\{1,2,3\}$ with probabilities for these values depending on X_1 being in the interval $(-\infty,-3]$, $(-3,-2]$, $(-2,-1]$, $(-1,0]$, $(0,1]$, $(1,2]$, $(2,3]$, or $(3,\infty)$. Both the top-down and modified top-down approaches to discretization removed all breaks when we were expecting them to leave 7 breaks. In fact, a full search of all possible discretizations would not have returned the correct result as we computed the DL local score for the correct discretization and found that it was larger than several other DL local scores encountered in our search.

The problem with the Friedman and Goldszmidt minimum description length approach to discretization is in the estimation of mutual information used in the DL local score in the case of non-repeated or rarely repeated values in the data. Note that, while estimating mutual information for continuous distributions is different than for discrete distributions (the probability estimates turn into kernel density estimation problems), this is not the issue here as all estimates of mutual information are only used in the calculation of the DL local score which is only considered on

various discretizations of the data. For discrete data with no (or few) repeated values, estimates of mutual information tend to be close to zero and the DL local score for X_1 becomes dominated by the following terms

$$\frac{1}{2}\log m \cdot ||\Pi_1||(||X_1^*|| - 1) + \frac{1}{2}\log m \cdot \sum_{j:X_1 \in \Pi_j} ||\Pi_j||(||X_j^*|| - 1)$$
$$+ \log k_1 + (m_1 - 1)H\left(\frac{k_1 - 1}{m_1 - 1}\right)$$

which is constant when considering different breaks being removed one at a time.

For the modified top-down approach, the full DL_{local} (all breaks removed) was approximately

$$\frac{1}{2}\log 100000 \cdot 0 \cdot (100000 - 1) + \frac{1}{2}\log 100000 \cdot 2 \cdot 100000(3 - 1)$$
$$+ \log 100000 + (100000 - 1)H\,(1)$$
$$= 2 \cdot 100000 \log 100000 + \log 100000 = 3453889$$

and the DL_{local} with any single break removed was approximately

$$\frac{1}{2}\log 100000 \cdot 0(99999 - 1) + \frac{1}{2}\log 100000 \cdot 2(99999)(3 - 1)$$
$$+ \log 99999 + (100000 - 1)H\left(\frac{99999 - 1}{100000 - 1}\right)$$
$$= 2 \cdot 99999 \log 100000 + log 99999 + 18.05231 = 3321930.$$

Since this is smaller than the full DL_{local}, all breaks were removed in the search process resulting in consolidation of all values for X_1 to a single value.

5.2 Bootstrapping

In order to get a better estimate of mutual information, we attempted a bootstrapping approach on node 1 and sampled, with replacement, 100,000 values of X_1 (along with their corresponding values for X_2 and X_3). This resulted in approximately $64,000$ unique values and produced some (incorrect) breaks in the discretization search but still did not give good enough estimates of mutual information to recover the correct discretization.

5.3 Estimating Mutual Information

In an effort to ensure that our sample size for calculating the local description length was adequate, we used a method derived by Gil, Fernandez and Martinez[13]) to calculate the sample size

required to accurately calculate mutual information. The method takes a subsample of significantly smaller size (magnitude of 30-100 in the paper) and uses a derived formula to estimate the necessary sample size to accurately determine the mutual information between two variables. Because the mutual information calculation dominates the local description length, this is the foundation for our sample size.

When using their derived formula on an example of 100,000 discrete data points derived with 6 distinct values at node 1 and 3 distinct values at node 2, the sample size required to accurately calculate mutual information ranged from 2000 to 9000 values required. This was determined using various subsample sizes of the same magnitude in the paper. The subsample was drawn with replacement and contained numerous repeated values.

In the case of continuous data at any one node, the subsample would not contain any repeated values. The calculation to determine sample size for the mutual information is virtually limitless in this case. There is no basis for comparison in the mutual information calculation therefore you can't ever have a large enough sample size. This renders the local description length calculation almost meaningless as mutual information such a dominant portion of the calculation.

5.4 Conclusion

We were unable to make the Friedman and Goldszmidt minimum description length principle work for a truly continuous data set with strictly observational data. If one is able to generate experimental data, holding certain nodes at fixed values while generating values for other nodes, it should be possible to improve estimates of mutual information and to have success with the minimum description length approach to discretization. The generation of experimental data is already necessary to determine causality in Bayesian networks and is in widespread use in the study of genetic regulatory networks. Thus, the discretization procedures outlined in this thesis could potentially be quite useful in this setting.

Bibliography

[1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In B.N. Petrov and F. Csaki, editors, Second international symposium on information theory., pages 267–281. Budapest: Academiai Kiado, 1973.

[2] H. Akaike. A new look at the statistical model identification. IEEE Transactions on Automatic Control, 19(6):716–723, 1974.

[3] H. Akaike. Likelihood of a model and information criteria. Journal of Econometrics, 16:3–14, 1981.

[4] R.R. Bouckaert. Properties of Bayesian learning algorithms. UAI '94, pages 102–109, 1994.

[5] H. Bozdogan. Akaike's information criterion and recent developments in information complexity. Journal of Mathematical Psychology, 44:62–91, 2000.

[6] T. Chen, H.L. He, and G.M. Church. Modeling gene expression with differential equations. In Pacific Symposium Biocomputing '99, pages 29–40. 1999.

[7] T. Cover and J. Thomas. Elements of Information Theory. Wiley & Sons, New York, 2006.

[8] A. Dobra, C. Hans, B. Jones, J.R. Nevins, G. Yao, and M. West. Sparse graphical models for exploring gene expression data. Journal of Multivariate Analysis, 90(1):196–212, 2004.

[9] N. Dojer, A. Gambin, A Mizera, B. Wilczynski, and J. Tiuryn. Applying dynamic Bayesian networks to perturbed gene expression data. BMC Bioinformatics, 7:249–260, 2006.

[10] N. Friedman and M. Goldszmidt. Discretizing continuous attributes while learning Bayesian networks. In Proceedings of ICML-1996, pages 157–165.

[11] N. Friedman and D. Koller. Being Bayesian about network structure. Machine Learning, 50:95–126, 2003.

[12] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. Journal of Computational Biology, 7:601–620, 2000.

[13] M.A. Gil, M.J. Fernandez, and I. Martinez. The choice of samle size in estimating mutual information. Applied Mathematics and Computation, 27:201–216, 1988.

[14] D. Heckerman. A tutorial on learning with Bayesian networks. Technical report, Microsoft Corporation, 1995.

[15] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. Bioinformatics, 19(17):2271–2282, 2003.

[16] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. Bioinformatics, 19(17):2271–2282, 2003.

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

[18] E.D. Jarvis, V.A. Smith, K. Wada, M.V. Rivas, M. McElroy, T.V. Smulders, P. Carninci, Y. Hayashizaki, F. Dietrich, X. Wu, P. McConnell, J. Yu, P.P. Wang, A.J. Hartemink, and S. Lin. A framework for integrating the Songbird brain. Journal of Comparative Physiology A, 188:961–980, 2002.

[19] R.L. Kashvap. A Bayesian comparison of different classes of dynamic models using empirical data. IEEE Transactions on Automatic Control, 22(5):715–727, 1977.

[20] I.M. Ong, J.D. Glasner, and D. Page. Modeling regulatory pathways in E. coli from time series expression profiles. Bioinformatics, 18:241–248, 2002.

[21] G. Schwartz. Estimating the dimension of a model. The Annals of Statistics, 5(2):461–464, 1978.

[22] D.E. Zak, F.J. Doyle, G.E. Goyne, and J.S. Schwaber. Simulation studies for the identification of genetic networks from cDNA array and regulatory activity data. Proc. 2nd Intl. Conf. Systems Biology, pages 231–238, 2001.

Appendix A

Huffman Coding and Shannon Coding

In this appendix we give a very brief description of two popular data compression algorithms. Suppose we wish to encode the string

22131234322132

using zeros and ones. Using the binary representation of these digits with varying lengths as needed(see Section 3.2) gives

10101111101110011101011110.

A compression algorithm will assign a different sequence of zeros and ones to each of the original characters (in this case digits) in such a way that more frequently occuring characters (like the 2 in this example) will have a shorter representation than less frequently occuring characters. A desirable feature of such an algorithm is that it produces a "prefix code" which means that there is no coded character that is a prefix of any other coded character. Because the prefixes are unique someone wanting to decode a sequence does not need special markers between words. A pure binary representation of a string of numbers is not a prefix code since, for example, 1111 may represent several things such as (but not limited to) four 1s, or two 3s, or a 3 followed by two 1s, or a 7 and a 1, or a 15.

A.1 Huffman Coding a Sequence of Characters

Huffman coding is an optimal compression algorithm ([7]) as it encodes data in the smallest string possible. In fact, it can be shown that no other prefix coding algorithm can do better than Huffman coding. To encode the string 22131234322132, we begin by listing the characters in descending frequency.

Character	Frequency
2	6
3	4
1	2
4	1

We then add the two lowest frequencies and list this frequency along with the remaining original frequencies in descending order.

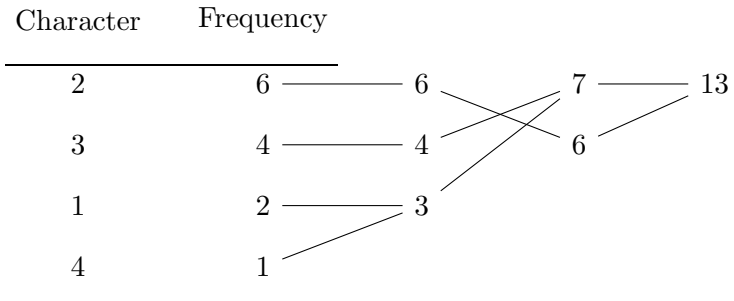
Character	Frequency	
2	6	6
3	4	4
1	2	3
4	1	

Repeating the process gives us

Character	Frequency		
2	6	6	7
3	4	4	6
1	2	3	
4	1		

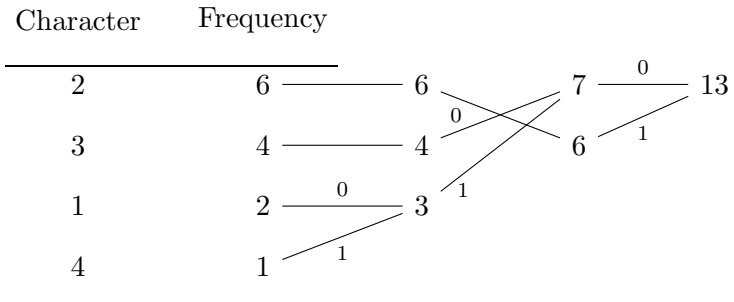
,

and finally



.

Now, in the cases where branches split, we put a 0 on the upper branch and a 1 on the lower branch.



.

Reading the branches from right to left (ignoring segments without 0's and 1's, we get the following code for each character.

Character	Code
1	010
2	1
3	00
4	011

The original message of 22131234322132 is now encoded as

$$1101000010100110001101000 1$$

which can easily be decoded using the above code table as each prefix is unique.

A.2 Huffman Coding a Random Variable

Suppose that X is a random variable taking on the values $1, 2, 3, 4$, or 5 with respective probabilities $0.20, 0.37, 0.12, 0.23$, and 0.08. The Huffman coding process is the same as in Section A.1

with "Frequency" replaced by "Probability". First, the possible values for X are listed in order of descending probability.

Value	Probability
2	0.37
4	0.23
1	0.20
3	0.12
5	0.08

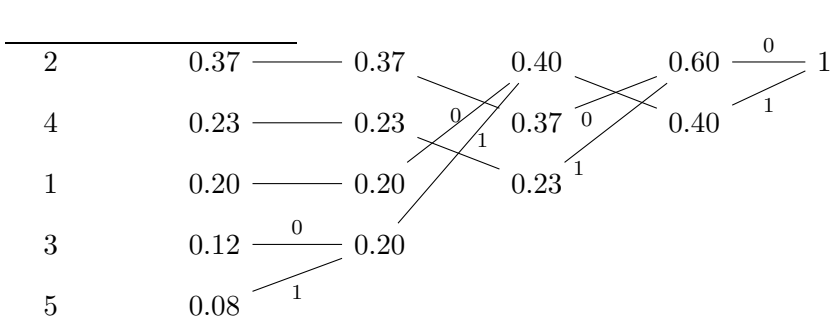
The two smallest probabilities are combined and listed with the remaining original probabilities in descending order.

Value	Probability	
2	0.37	0.37
4	0.23	0.23
1	0.20	0.20
3	0.12	0.20
5	0.08	

Continuing, we get

Value	Probability		
2	0.37	0.37	0.40
4	0.23	0.23	0.37
1	0.20	0.20	0.23
3	0.12	0.20	
5	0.08		

and, after a few more steps,



The coded values for the random variable X are

Value	Code
1	10
2	00
3	110
4	01
5	111

Notice that the higher probability values are assigned a shorter code. So, a data set consisting of realizations of X will be more compressed than, say, a binary encoding with a fixed number of bits.

A.3 Shannon Coding a Random Variable

Shannon coding of a random variable is designed to give a prefix code for the value i that has length $\ell_i = \lceil \log \frac{1}{p_i} \rceil$ where $p_i = p(X = i)$ and $\lceil \cdot \rceil$ is the ceiling function. Using the same random variable from Section A.2, we see that we want a code for $1, 2, 3, 4$, and 5 with lengths given in the following table.

i	p_i	ℓ_i
1	0.20	3
2	0.37	2
3	0.12	4
4	0.23	3
5	0.08	4

To construct the code, we begin by reordering the values from highest probability values to lowest. Let $val(j)$ denote the value of the random variable corresponding to the jth value in the probability ordered list. Define $F_1 = 0$ and, for $j \geq 2$, define $F_j = \sum_{k=1}^{j-1} p_{val(j)}$.

j	$val(j)$	$p_{val(j)}$	F_j
1	2	0.37	0
2	4	0.23	0.37
3	1	0.20	0.60
4	3	0.12	0.80
5	5	0.08	0.92

The codeword (or unique prefix code) for the jth value is the truncated (to the desired length) number after the decimal point of the binary representation of F_j.

For example, $F_1 = 0$ becomes, in binary $0.0000\ldots$ and so $val(1) = 2$, for which we want a code of length 2, is encoded as 00.

$F_2 = 0.37$ becomes, in binary, $0.010011010\ldots$ and so $val(2) = 4$, for which we want a code length of 3, is encoded as 010.

$F_3 = 0.60$ becomes, in binary, $0.10011001\ldots$ and so, $val(3) = 1$, for which we want a code length of 3, is encoded as 100.

$F_4 = 0.80$ becomes, in binary, $0.11001100\ldots$ and so, $val(4) = 3$, for which we want a code length of 4, is encoded as 1100.

Finally, $F_5 = 0.92$ becomes, in binary, $0.11101011\ldots$ and so, $val(5) = 5$, for which we want a code length of 4, is encoded as 1110.

The Shannon coded values for the random variable X are

Value	Code
1	100
2	00
3	1100
4	010
5	1110

Note that the Shannon code in this case produced a prefix code. To prove that this will always be the case, we note that, based on our choice of ℓ_i, p_i is bounded as follows: $2^{-\ell_i} < p_i < 2^{-(\ell_i - 1)}$. From this we know that each p_i must differ by at least $2^{-\ell_i}$. This implies that each F_j will differ by at least $2^{-\ell_i}$ and there will be at least once place different in the first $L - i$ bits of the binary expansion for F_j and F_{j+1}. In summary, the codeword for F_j will differ from the codeword for F_{j+1} at least one in the first l_i places of the binary expressions.

Shannon coding is utilized for encoding data because it equates to the negative log likelihood that more traditional statistical models are based upon. Although Huffman coding is the best, this provides a common basis for the scoring metrics used in statistical analysis.

Appendix B

More Examples

In this appendix we provide further computational evidence to support claims that were made in Chapter 4. Although significantly more computations were performed to verify our findings, the examples below illustrate the significance and power on other three node networks.

B.1 Searching All Discretizations

In this section we provide further evidence to support that the searching all discretizations is not necessary. We can significantly reduce the calculation time required by choosing a DAG in which all nodes are connected by at least one edge. In this example, we begin with 3 unique values for each node generated by DAG 14. The data at node 1 is then exploded such that if $X_1 = 1$, the values in the new data set are $X_1 \in \{1, 2\}$. Similarly, if $X_1 = 2$, the values were exploded such that $X_1 \in \{3, 4\}$ and if $X_1 = 3$ the value was changed to $X_1 = \{5, 6\}$. This corresponds to discretization 12 and results of calculating all DL_{local} scores are contained in Table B.1.

Like the example in Chapter 4, we see that the lowest local description length score corresponds to discretization 12 for all networks except DAGs 1,6,and 7. So given a data set, we can choose to search any DAG in which all nodes are connected by one edge and still return the correct discretization.

To provide an additional example, the data in Table B.2 was generated in the same manner as the previous example except it was derived from DAG 12. The data was then exploded using discretization 9 $\{1|234|56\}$. Table B.2 contains DL_{local} scores calculated for all DAGs and all

discretizations. We see that the lowest local description length score for all DAGs with the exception of DAGs 1,6, and 7 corresponds to discretization 9. Thus, choosing a DAG with all nodes connected by at least one edge can save significant computational time when attempting to recover the correct discretization.

B.2 Single Iteration Top-Down

In this section we provide two more examples of the single iteration top-down method. The first example begins with 3 distinct values assigned to each node based on DAG 17. With a sample size of 100,000 data points, the values at node 1 were then exploded into 6 distinct values such that the following is the correct discretization policy: {123|4|56}. See Table B.3 for the results of the local description length calculations.

Using the conjecture presented in Chapter 4, we will remove any threshold the is less than the local description length score calculated with all thresholds inserted. From Table B.3 we see that thresholds 1,2,and 5 are to be removed. This results in the following discretization {123|4|56} which is the correct result. The correct answer was found in only six calculations.

To further illustrate the power of single iteration top-down, consider an additional example. The three distinct values at node 1 were generated using DAG 20 with a sample size of 100,000 data points. The data was exploded into the following discretization policy {1|23|456} The results of the DL_{local} calculations are depicted in Table B.4.

Applying our conjecture to Table B.4, we see that thresholds 2,4, and 5 are to be removed. This returns the correct discretization of {1|23|456}.

Table B.1: Example 2: Local Description Length Calculated for Three Node DAGs (NOTE: Graphs 11-25 were removed from the table as they are redundant and only graphs 1-10 are needed to illustrate the findings)

Disc Number	Graph Number									
	1	2	3	4	5	6	7	8	9	10
1	0	16.61	0	16.61	0	0	0	33.219	16.61	16.61
2	4.61	-3387.45	-3395.755	-1470.765	-1479.07	4.61	4.61	-1439.096	-1447.401	-1447.401
3	4.61	-8783.985	-8792.29	-3490.963	-3499.268	4.61	4.61	-3458.508	-3466.813	-3466.813
4	4.61	-1403.988	-1412.293	-1800.639	-1808.944	4.61	4.61	-1767.693	-1775.998	-1775.998
5	4.61	-632.196	-640.5017	-1036.309	-1044.614	4.61	4.61	-1003.149	-1011.454	-1011.454
6	4.61	-283.992	-292.296	-258.211	-266.515	4.61	4.61	-226.922	-235.226	-235.226
7	6.44	-8765.991	-8765.991	-3743.743	-3473.381	6.44	6.44	-3425.394	-3425.394	-3425.394
8	6.44	-3653.614	-3653.614	-2210.743	-2210.743	6.44	6.44	-2162.497	-2162.497	-2162.497
9	6.44	-4955.712	-4955.712	-1961.12	-1961.12	6.44	6.44	-1913.226	-1913.226	-1913.226
10	6.44	-4085.369	-4085.369	-1599.039	-1559.039	6.44	6.44	-1512.005	-1512.005	-1512.005
11	6.44	-9998.884	-9998.884	-3801.052	-3801.52	6.44	6.44	-3752.322	-3752.322	-3752.322
12	6.44	-13455.98	-13455.98	-4245.882	-4245.882	6.44	6.44	-4197.811	-4197.811	-4197.811
13	6.44	-10651.65	-10651.65	-3610.538	-3610.538	6.44	6.44	-3562.978	-3562.978	-3562.978
14	6.44	-4842.736	-4842.736	-2227.622	-2227.622	6.44	6.44	-2178.821	-2178.821	-2178.821
15	6.44	-2546.543	-2546.543	-1830.861	-1830.861	6.44	6.44	-1783.001	-1783.001	-1783.001
16	6.855	-614.911	-614.911	-1019.786	-1019.786	6.855	6.855	-972.66	-972.6	-972.6
17	6.855	-9982.305	-9974	-3784.885	-3776.58	6.855	6.855	-3720.623	-3712.318	-3712.318
18	6.855	-13439.4	-13431.09	-4229.716	-4221.411	6.855	6.855	-4166.111	-4157.806	-4157.806
19	6.855	-10635.07	-10626.77	-3594.372	-3586.067	6.855	6.855	-3531.278	-3522.973	-3522.973
20	6.855	-7093.777	-7085.473	-2639.14	-2630.835	6.855	6.855	-2575.04	-2566.735	-2566.735
21	6.855	-4797.584	-4789.279	-2242.379	-2234.074	6.855	6.855	-2179.22	-2170.916	-2710.916
22	6.855	-4939.841	-4931.536	-1946.012	-1937.707	6.855	6.855	-1884.092	-1875.787	-1875.787
23	6.855	-13439.05	-13430.74	-4229.449	-4221.144	6.855	6.855	-4164.865	-4156.56	-4156.56
24	6.855	-11142.85	-11134.55	-3832.688	-3824.383	6.855	6.855	-3769.045	-3760.741	-3760.741
25	6.855	-13440.11	-13431.8	-4230.774	-4222.469	6.855	6.855	-4168.677	-4160.372	-4160.372
26	5.932	-4826.866	-4818.561	-2212.514	-2204.209	5.932	5.932	-2149.687	-2141.382	-2141.382
27	5.932	-13423.81	-13407.2	-4214.621	-4198.011	5.932	5.932	-4134.504	-4117.894	-4117.894
28	5.932	-11127.61	-11111	-3817.859	-3801.25	5.932	5.932	-3738.684	-3722.075	-3722.075
29	5.932	-13424.87	-13408.26	-4215.946	-4199.336	5.932	5.932	-4138.316	-4121.706	-4121.706
30	5.932	-7079.245	-7062.635	-2625.37	-2608.76	5.932	5.932	-2547.244	-2530.635	-2530.635
31	2.585	-13424.51	-13407.91	-4215.679	-4199.069	2.585	2.585	-4137.069	-4120.46	-4120.46
32	2.585	-13411.7	-13386.78	-4203.274	4178.36	2.585	2.585	-4109.132	-4084.217	-4084.217

Table B.2: Example 3: Local Description Length Calculated for Three Node DAGs (NOTE: Graphs 11-25 were removed from the table as they are redundant and only graphs 1-10 are needed to illustrate the findings)

Disc Number	Graph Number									
	1	2	3	4	5	6	7	8	9	10
1	0	16.61	0	16.61	0	0	0	33.219	16.61	16.61
2	4.61	-758.46	-766.764	-7.435	-15.739	4.61	4.61	-770.504	-778.809	-778.809
3	4.61	-1538.386	-1546.691	-52.47	-60.774	4.61	4.61	-1595.465	-1603.77	-1603.77
4	4.61	-2470.34	-2748.675	-121.92	-130.224	4.61	4.61	-2866.899	-2875.204	-2875.204
5	4.61	-4292.939	-4301.244	-218.234	-226.539	4.61	4.61	-4515.783	-4524.088	-4524.088
6	4.61	-1504.321	-1512.626	-55.902	-64.207	4.61	4.61	-1564.832	-1573.137	-1573.137
7	6.44	-1928.062	-1928.062	-56.225	-56.225	6.44	6.44	-1990.727	-1990.727	-1990.727
8	6.44	-3529.589	-3529.589	-148.351	-148.351	6.44	6.44	-3684.38	-3684.38	-3684.38
9	6.44	-5371.403	-5371.403	-262.751	-262.751	6.44	6.44	-5640.593	-5640.593	-5640.593
10	6.44	-1658.141	-1658.141	-46.071	-46.071	6.44	6.44	-1710.652	-1710.652	-1710.652
11	6.44	-3122.992	-3122.992	-126.316	-126.316	6.44	6.44	-3255.748	-3255.748	-3255.748
12	6.44	-4964.749	-4964.749	-241.01	-241.009	6.44	6.44	-5212.197	-5212.197	-5212.197
13	6.44	-2051.306	-2051.306	-67.792	-67.792	6.44	6.44	-2125.538	-2125.538	-2125.538
14	6.44	-4564.828	-4564.828	-220.162	-220.162	6.44	6.44	-4791.43	-4791.43	-4791.43
15	6.44	-2889.861	-2889.861	-114.875	-114.875	6.44	6.44	-3011.176	-3011.176	-3011.176
16	6.44	-4275.023	-4275.023	-200.476	-200.498	6.855	6.855	-4481.942	-4481.942	-4481.942
17	6.855	-3514.083	-3505.778	-131.487	-123.182	6.855	6.855	-3652.425	-3644.12	-3644.12
18	6.855	-5355.84	-5347.535	-246.18	-237.876	6.855	6.855	-5608.874	-5600.57	-5600.57
19	6.855	-2442.397	-2434.092	-72.963	-64.658	6.855	6.855	-2522.215	-2513.91	-2513.91
20	6.855	-5355.463	-5347.158	-248.09	-239.704	6.855	6.855	-5610.327	-5602.022	-5602.022
21	6.855	-3680.496	-3672.191	-142.721	-134.416	6.855	6.855	-3830.072	-3821.767	-3821.767
22	6.855	-5354.902	-5346.598	-246.411	-238.106	6.855	6.855	-5608.168	-5599.863	-5599.863
23	6.855	-4948.866	-4940.561	-225.974	-217.669	6.855	6.855	-5181.695	-5173.39	-5173.39
24	6.855	-3273.899	-3265.594	-120.686	-112.382	6.855	6.855	-3401.44	-3393.135	-3393.135
25	6.855	-4948.248	-4939.943	-224.669	-216.364	6.855	6.855	-5179.772	-5171.467	-5171.467
26	6.855	-4548.328	-4540.023	-203.823	-195.517	5.932	5.932	-4759.005	-4750.7	-4750.7
27	5.932	-5341.295	-5324.685	-232.483	-215.873	5.932	5.932	-5579.71	-5563.1	-5563.1
28	5.932	-3666.328	-3649.718	-127.195	-110.586	5.932	5.932	-3799.455	-3782.845	-3782.845
29	5.932	-5340.677	-5324.068	-231.178	-214.569	5.932	5.932	-5577.787	-5561.178	-5561.178
30	5.932	-5340.301	-5323.691	-233.007	-216.397	5.932	5.932	-5579.24	-5562.63	-5562.63
31	5.932	-4933.703	-4917.094	-210.973	-194.363	2.585	2.585	-5150.608	-5133.998	-5133.998
32	2.585	-5328.556	-5303.642	-219.91	-194.991	2.585	2.585	-5551.046	-5526.132	-5526.132

Table B.3: Local Description Length Score: Example 3

Threshold Number	Corresponding Discretization	DL local score
All k_i	1|2|3|4|5|6	-28764.00
1	12|3|4|5|6	-28784.49
2	1|23|4|5|6	-28781.20
3	1|2|34|5|6	-21685.14
4	1|2|3|45|6	-21708.83.03
5	1|2|3|4|56	-28785.03

Table B.4: Local Description Length Score: Example 4

Threshold Number	Corresponding Discretization	DL local score
All k_i	1|2|3|4|5|6	-339.41
1	12|3|4|5|6	-100.16
2	1|23|4|5|6	-359.31
3	1|2|34|5|6	-300.21
4	1|2|3|45|6	-356.26
5	1|2|3|4|56	-357.34