A Relational Database Approach to Bayesian Network Knowledge Discovery

Tzu-Tsung Wong / tzu-tsun@iis.sinica.edu.tw

Chun-Nan Hsu / chunnan@iis.sinica.edu.tw

Chia-Che Ma / chiache@iis.sinica.edu.tw

Institute of Information Science, Academia Sinica

128 Academy Road, Section 2, Nankang, Taipei, Taiwan

Abstract

A Bayesian network is a powerful formalism for decision-making and knowledge discovery. An approach to Bayesian network training for large scaled real-world applications is important. Bayesian network training includes the following two steps. Experts first select appropriate parameters consistent with their confidence to transform the conditional probabilities into Dirichlet priors. Then the conditional posteriors for the variables in the network can be obtained by Bayesian updating. In this paper, we present a database scheme to store and manipulate a large Bayesian network as well as training data sets in a relational database. This scheme facilitates Bayesian network training and allows the system to take advantage of the benefits of relational data models. Other features of this scheme are that it tolerates incomplete training data and is generally applicable for Bayesian networks with arbitrary graph structures. Since RDBMS are widely available, this scheme greatly simplifies the construction of a Bayesian network based KDD system.

Keywords: Bayesian networks, Dirichlet distribution, normalization, relational databases

1 Introduction

A Bayesian network is a graphical representation of conditional probability distributions for a set of variables, and can be used to build models for the problems with uncertainty. Hence, Bayesian networks gradually become a popular tool for constructing knowledge discovery and decision support systems (Heckerman et al., 1995). Especially, the Bayesian network provides a systematic method for incorporating expert knowledge and training data. This advantage increases the applicability of the Bayesian network.

The computation for the optimal solution of a problem in a large Bayesian network is generally intractable. Thus, researchers have developed heuristic approaches to obtain an acceptable, though not optimal, solution. In order to demonstrate the efficiency of those methods, experimental results are often necessary (Heckerman *et al*., 1995; Friedman *et al*., 1997; Kearns *et al*., 1997; Friedman and Goldszmidt, 1997). An easy and applicable method for building a Bayesian network with the capability of learning can facilitate the implementation of experiments and increase the applicability of Bayesian networks in realistic cases. So, we will attempt to propose a relational database scheme that can be used to store and manipulate the data for various Bayesian networks with discrete variables.

The data in a relational database are perceived by users as tables. Unlike either a hierarchic or a network database, there does not exist any pre-specified data access path for a relational database. This implies that the data manipulation in a relational database is more flexible. In addition, it is easier to maintain the data integrity in a relational

database (Date, 1995). Since relational database management systems are popular and provide some mechanism for query optimization, a relational database should be a proper tool for handling the data of Bayesian networks. By using relational databases, researchers can focus on the operations in the Bayesian network. Due to these advantages, we adopt the relational database as a tool for building Bayesian networks.

As you will see, our relational database scheme facilitates Bayesian network training and allows the system to take advantage of the benefits of relational data models. Other features of this scheme are that it tolerates incomplete training data and is generally applicable for Bayesian networks with arbitrary graph structures. Since RDBMS are widely available, this scheme greatly simplifies the construction of a Bayesian network based KDD system.

The remainder of this paper is organized as follows. In section 2, we will briefly introduce Bayesian networks and relational databases. Under the assumption that all variables in the Bayesian network are discrete, a relational database scheme for storing the model data (the data for the Bayesian network and the training data) is proposed in section 3. An illustrative example is presented in section 4. Finally, section 5 describes the conclusions and discusses the directions for future research.

2 Bayesian networks and relational databases

2.1 Bayesian networks

A Bayesian network has two components: a directed acyclic graph where the nodes represent the variables in a domain of interest and the edges represent the conditional independencies among the variables; and the conditional probability distribution of each variable. Node (Variable) j is said to be a parent of node (variable) m if there is a directed edge from j to m. Let π_j be the set of the parents of node j, and let Ω be the set of the variables X_j in the network. Then by the chain rule, the joint distribution of $\mathbf{X} = (X_1, X_2, \ldots, X_{|\Omega|})$ can be represented as:

$$p(\mathbf{x}) = \prod_{j=1}^{|\Omega|} p(x_j \mid \pi_j),$$

where $|\Omega|$ is the number of variables in Ω. Thus, when the conditional distributions $p(x_j|\pi_j)$ for j = 1, 2, …, $|\Omega|$ are known, Bayes' formula can be used to calculate any probability of interest in the Bayesian network.

Random vector $\theta = (\theta_1, \theta_2, \ldots, \theta_k)$ has a k-variate Dirichlet distribution with parameters $\alpha_j > 0$ for j = 1, 2, ..., k+1 if it has density

$$f(\boldsymbol{\theta}) = \frac{\Gamma(\alpha)}{\prod_{j=1}^{k+1} \Gamma(\alpha_j)} \prod_{j=1}^{k} \theta_j^{\alpha_j - 1} (1 - \theta_1 - \ldots - \theta_k)^{\alpha_{k+1} - 1}$$

for $\theta_1+\theta_2+\ldots+\theta_k \le 1$ and $\theta_j \ge 0$ for j = 1, 2, ..., k, where $\alpha = \alpha_1+\alpha_2+\ldots+\alpha_{k+1}$. This distribution will be denoted $D_k(\alpha_1, \alpha_2, \ldots, \alpha_k; \alpha_{k+1})$. Suppose that the number of possible outcomes for a trial is k+1, and let $D = \{y_1, y_2, \ldots, y_{k+1}\}$ be a data set for the outcomes in

n trials, where y_j is the number of trials turning to be outcome j. When the prior distribution $p(\theta)$ is a Dirichlet distribution $D_k(\alpha_1, \alpha_2, ..., \alpha_k; \alpha_{k+1})$, and the likelihood function $L(D|\theta)$ follows a multinomial distribution, then the posterior distribution of θ given D is also a Dirichlet distribution $D_k(\alpha_1+y_1, \alpha_2+y_2, \ldots, \alpha_k+y_k; \alpha_{k+1}+y_{k+1})$. Deriving a posterior distribution from a prior distribution and a data set is called *Bayesian updating*.

The expected value of θ_j given D is $E(\theta_j|D) = (\alpha_j+y_j)/(\alpha+n)$ for j = 1, 2, …, k+1. This expression can be rewritten as follows:

$$E(\theta_j \mid D) = \frac{\alpha_j + y_j}{\alpha + n} = \frac{\alpha}{\alpha + n}\frac{\alpha_j}{\alpha} + \frac{n}{\alpha + n}\frac{y_j}{n} = wE(\theta_j) + (1 - w)\frac{y_j}{n},$$

where $w = \alpha/(\alpha+n)$. Note that $E(\theta_j)$ and y_j/n are the prior and the sample means of θ_j, respectively. Hence, w and 1-w can be thought of as the weights of prior and sample means, respectively, and the weights of the θ_j are all identical. Thus, the posterior mean of θ_j is just the sum of the prior and sample means multiplied by their weights. This result reveals an advantage of the Bayesian updating.

When the value of α is large, this implies that the estimates of the $E(\theta_j)$ are *reliable* (Wong, 1998). If the training data are sparse (i.e., the value of n is small), the value of w is close to one, and hence we can still have reliable conditional posteriors to evaluate the probability of interest. On the contrary, if we have plenty of training data (i.e., the value of n is much larger than α), then the value of w is close to zero, and the conditional posteriors are primarily determined by the training data. So, Bayesian updating can incorporate expert knowledge and training data to obtain a reliable result. Note that

when θ is not Dirichlet distributed, the advantage for Bayesian updating is still true, but we may not have a simple expression for $E(\theta_j|D)$ as when θ is Dirichlet distributed. Thus, the Bayesian updating for the Dirichlet distribution is an example instead of a formal proof for the advantage.

Suppose that all variables in the Bayesian network are discrete. Let the number of possible outcomes of node j be k+1, and let θ_i for i = 1, 2, …, k+1 be the conditional probability for variable j to have outcome i given parent π_j. Then the joint distribution of $\theta = (\theta_1, \theta_2, \ldots, \theta_k)$ is usually assumed to have a k-variate Dirichlet distribution $D_k(\alpha_1, \alpha_2, \ldots, \alpha_k; \alpha_{k+1})$, since the Dirichlet distribution is conjugate to the multinomial sampling (as discussed above). Suppose that a Bayesian network possesses the property of parameter independence. Then the Bayesian updating of each conditional probability distribution can be performed independently. In this case, the computation of Bayesian updating of the Dirichlet prior is simple and fast.

Generally, the expected value $E(\theta_m|D)$ is assumed to be the probability that variable X_j is turning to be outcome m given parent π_j in a Bayesian network after learning. The complete training process in a Bayesian network includes two steps. The conditional probability distribution of each variable is first transformed into a Dirichlet prior by selecting appropriate parameters consistent with analysts' confidence. The training data are then used to update the Dirichlet priors to obtain Dirichlet posteriors. Let Δ be a subset of Ω that does not include variable X_j, and let $p(x_j|\Delta)$ be the probability of interest. Then by Bayes' formula, we have

$$p(x_j \mid \Delta) = \frac{p(x_j, \Delta)}{p(\Delta)} = \frac{\sum_{x_i \in \Omega \setminus \{x_j, \Delta\}} p(\mathbf{x})}{\sum_{x_i \in \Omega \setminus \Delta} p(\mathbf{x})} = \frac{\sum_{x_i \in \Omega \setminus \{x_j, \Delta\}} \prod_{m \in \Omega} p(x_m \mid \pi_m)}{\sum_{x_i \in \Omega \setminus \Delta} \prod_{m \in \Omega} p(x_m \mid \pi_m)}.$$

This implies that any probability of interest can be represented as a combination of the expected values $E(\theta_m|D)$ obtained from the Dirichlet posteriors. Thus, there are generally no restrictions for setting queries in a Bayesian network.

2.2 Relation databases

A relational database is a collection of tables. In general, there are some indices in a table to speed up the performance of queries. One of those indices is called a primary index that generally does not allow duplicated values. In building a relational database, system analysts first use a data flow diagram to represent users' requirements, and refine those requirements to obtain an entity-relationship model for the system. Then the entity-relationship model is used to construct the database by means of normalization.

Basically, a database with normalized tables has a simpler structure, and its data manipulation is also simpler. There exist five types of normal form for tables. Practically, a necessary condition for designing a relational database is that every table in the database is in third normal form. Since both fourth and fifth normal forms can increase the complexity of data manipulation, a table will be in either fourth or fifth normal form only when necessary.

A table is in first normal form if there are no repeating groups in the table. For example, there is a table for the information of the children of each employee in a

company. An employee may have several children. If each employee has only one record in the table, then this table is not in first normal form. A table is in second normal form if it is in first normal form and the primary index can determine the value of any column that is not in the primary index. A table can violate second normal form when its primary index includes more than one column. For instance, suppose that the primary index of a table includes employee ID and department code. Then if this table also has employee name, this table will not be in second normal form, since the employee name can be determined by part of the primary index.

A table is in third normal form if it is in second normal form and the value of each column that is not in the primary index cannot be determined by the value of some other column that is not in the primary index neither. For instance, let the primary index of a table be employee ID. Then if this table includes both department code and department name, this table will not be in third normal form, since the department code which is not in the primary index uniquely determine the department name. When a table is in first normal form but not in second normal form, or in second normal form but not in third normal form, please refer to Date (1995) for the methods of normalization.

3 Relational database scheme

In this section, we will present a relational database scheme that can be used to store the model data of the Bayesian networks with arbitrary structures. In particular, this scheme includes a data structure that can facilitate the Bayesian updating. This implies that the

scheme cannot only meet the requirement for the queries in the Bayesian networks, but also has a mechanism for learning when the training data are available. As described in Section 2.1, we need two elements to delineate a Bayesian network: a directed acyclic graph and the conditional probability distributions among variables. The only entity in the graph is the nodes (or variables), and the directed edges in the graph are self-relationships among the variables. The conditional probabilities are attributes of the variables. Hence, the entity-relationship diagram of a Bayesian network is simple.

Figure 1 shows the relational database scheme for Bayesian networks with discrete variables. The content of each table is given in Figure 2. Each directed edge in Figure 1 indicates a one-to-many relationship. Tables **Node**, **NodeContent**, and **Parent** represent the directed acyclic graph of a Bayesian network, while tables **ProbMaster**, **ProbDetail**, **Probability**, and **Parameter** represent the conditional probabilities and keep the information for Bayesian updating. The training data are stored in table **Case**.

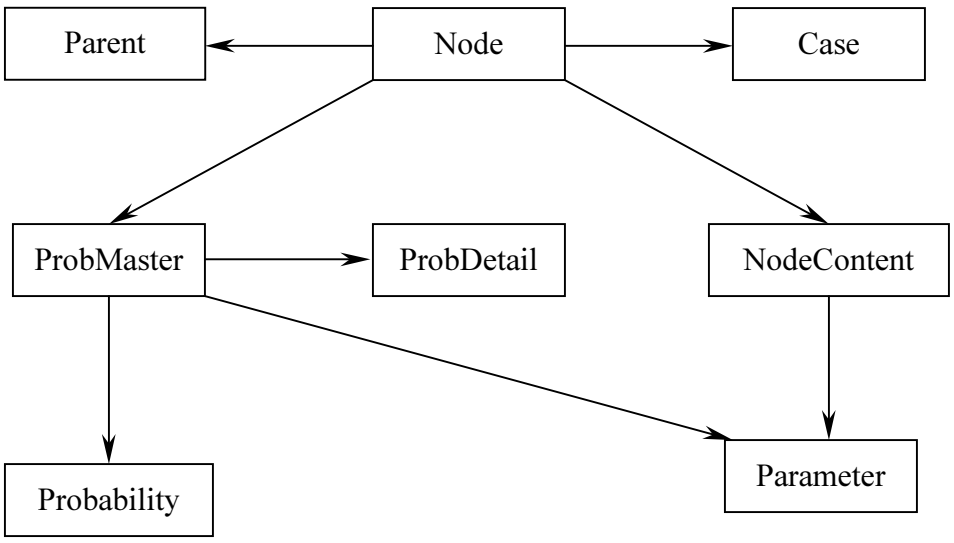

Figure 1. The relationship diagram for tables

TABLE **Node** (Node ID, Node name, Number of outcomes, Number of parents)

PRIMARY INDEX (Node ID);

TABLE **NodeContent** (Node ID, Outcome ID, Outcome content)

PRIMARY INDEX (Node ID, Outcome ID);

TABLE **Parent** (Node ID, Parent node ID)

PRIMARY INDEX (Node ID, Parent node ID);

TABLE **Case** (Case ID, Node ID, Outcome ID)

PRIMARY INDEX (Case ID, Node ID);

TABLE **ProbMaster** (Node ID, Combination ID, Alpha)

PRIMARY INDEX (Node ID, Combination ID);

TABLE **ProbDetail** (Node ID, Combination ID, Parent node ID, Parent outcome ID)

PRIMARY INDEX (Node ID, Combination ID, Parent node ID);

TABLE **Probability** (Node ID, Combination ID, Outcome ID, Probability)

PRIMARY INDEX (Node ID, Combination ID, Outcome ID);

TABLE **Parameter** (Node ID, Combination ID, Outcome ID, Prior, Posterior)

PRIMARY INDEX (Node ID, Combination ID, Outcome ID);

Figure 2. The content of each table in the database

Since all variables are discrete, we can assign each possible outcome of a variable a unique outcome ID located in table **NodeContent** for data manipulation. The numbers of possible outcomes of the variables in a Bayesian network can be different. Thus, the data structure for storing possible outcomes must be normalized to satisfy the requirement of third normal form. Table **Parent**, which represents the self-relationship among variables in the entity-relationship diagram, has the data structure for edges.

Since the number of variables in a Bayesian network can vary, each case in the training data is split into several records according to the number of variables. This way, table **Case** can store the training data regardless of the number of variables in the Bayesian network. Note that we can also use table **Case** to store the training data even if the training data include missing variable values.

Similar to tables **Node** and **NodeContent** for the nodes, we need tables **ProbMaster** and **Probability** for the conditional probabilities. In practice, we will generally have the conditional probabilities $p(x_j|\pi_j)$ instead of the parameters in the Dirichlet priors. Hence, we need to select the value for α to transform the conditional probabilities into a Dirichlet prior. Table **ProbMaster** includes column 'Alpha' for this purpose. For example, when $p(x_j=3|\pi_j) = 0.05$ and $\alpha = 30$, we will have $\alpha_3 = \alpha \cdot p(x_j=3|\pi_j) = 1.5$. Thus, when the values of α and the conditional probabilities are known, we can set the values for column 'Prior' in table **Parameter**.

As mentioned above, we hope that this relational database scheme can be used for arbitrary Bayesian networks with discrete variables. However, the number of parents of a node can vary. Thus, we divide each possible parent outcome combination of a node into several records in table **ProbDetail** according to the number of parents of the node. Like the possible outcomes of each node, each possible parent outcome combination of a node has a unique combination ID for data manipulation.

Suppose that we choose the combination IDs arbitrarily. Then when a whole case is retrieved from the database for Bayesian updating, we will need to search the

combination ID in table **ProbDetail** for current case. Otherwise, we will not be able to know which value of column 'Posterior' in table **Parameter** should be increased by one. Since the number of possible parent outcome combinations of a node can be large, this approach for Bayesian updating is likely to be time-consuming. Therefore, we develop the following hashing function-like technique to determine the combination ID.

Let π_m be the set of the parents of node m. Assume without loss of generality that the nodes in π_m are in a pre-determined order. Furthermore, let n_j be the number of possible outcomes of the j^{th} variable in π_m, and for any possible parent outcome combination of node m, let a_j be the outcome ID of the j^{th} variable in π_m. Then each possible parent outcome combination of node m can be represented as $\{a_j, j = 1, \ldots, |\pi_m|\}$, where $|\pi_m|$ is the number of variables in π_m. Note that the value of a_j can vary from 1 to n_j. Define the combination ID s_c for each parent outcome combination $c = \{a_j, j = 1, \ldots, |\pi_m|\}$ as:

$$s_c = \sum_{i=1}^{|\pi_m|-1} \left[(a_i - 1) \prod_{q=i+1}^{|\pi_m|} n_q \right] + a_{|\pi_m|}.$$

The total number of possible parent outcome combinations of node m is $\prod_{i=1}^{|\pi_m|} n_i$. When the parent outcome combination of node m is $c = \{n_j, j = 1, \ldots, |\pi_m|\}$, we will have $s_c = \sum_{i=1}^{|\pi_m|} n_i$, which is the largest combination ID for node m. It can be shown that each parent outcome combination will have a unique combination ID in this transformation.

The above method for determining the combination ID can greatly increase the speed of Bayesian updating, especially when the number of parent outcome combinations of some node is very large. Recall that the variables in π_m are in a pre-determined order.

When we read a whole case from table **Case** for Bayesian updating, column 'Outcome ID' gives us the values of the a_j. Table **Node** will provide the values of the n_j. Hence, we are able to know the combination ID for each node in current case without expensive string comparisons.

Example. Let nodes A, B, and C be parent nodes of node D, and let the numbers of possible outcomes of these three nodes are 8, 10, and 14, respectively. Hence, we have π_D = {A, B, C}, and n_1 = 8, n_2 = 10, and n_3 = 14. The number of possible parent outcome combinations of node D is 8×10×14 = 1120, and the number of records in table **ProbDetail** for these parent outcome combinations is 1120×3 = 3360 (3 parents). If the content of a case is {6, 5, 10, 3}, then we will need to access more than 3000 records in table **ProbDetail** for comparisons in the worst case. By using the above method, the combination ID of c = $\{a_1 = 6, a_2 = 5, a_3 = 10\}$ for node D is s_c = (6-1) ×10×14 + (5-1) ×14 + 10 = 766. Now, we can use the values of the node ID of node D, s_c, and the outcome ID of node D in current case as a compound primary index to access the record in table **Parameter** for Bayesian updating.

Although there are $|\pi_m|(|\pi_m|-1)/2$ multiplications in the expression for s_c, we can calculate the value of the combination ID for each node in a linear time. Note that the term $(a_{i-1}-1)\prod_{q=i}^{|\pi_m|} n_q$ in the expression of s_c can be rewritten as $(a_{i-1}-1)\times n_i \times \prod_{q=i+1}^{|\pi_m|} n_q$. If we perform the summation of s_c in a reverse order, then the value $\prod_{q=i+1}^{|\pi_m|} n_q$ obtained in calculating $(a_i-1)\prod_{q=i+1}^{|\pi_m|} n_q$ can be used to calculate the value of $(a_{i\ 1}-1)\prod_{q=i}^{|\pi_m|} n_q$ by only two multiplications. In this approach, there will be only $2|\pi_m|$-3 multiplications for

calculating the value of s_c.

4 An illustration

Consider the problem of detecting credit-card fraud discussed in Heckerman (1995). The Bayesian network in this problem has five variables: *Fraud* (F); *Gas* (G); *Jewelry* (J); *Age* (A); and *Sex* (S), as shown in Figure 3. From Figure 3, we can see that the conditional independencies of the variables are:

$p(a|f) = p(a)$

$p(s|a,f) = p(s)$

$p(g|f,a,s) = p(g|f)$

$p(j|f,a,s,g) = p(g|f,a,s)$.

Hence, the joint distribution of $\mathbf{X} = (F, A, S, G, J)$ is

$$p(\mathbf{x}) = p(f,a,s,g,j) = p(j|f,a,s,g)p(g|f,a,s)p(s|f,a)p(a|f)p(f)$$

$$= p(j|f,a,s)p(g|f)p(s)p(a)p(f).$$

The data for the directed acyclic graph of this Bayesian network is shown in Figure 4.

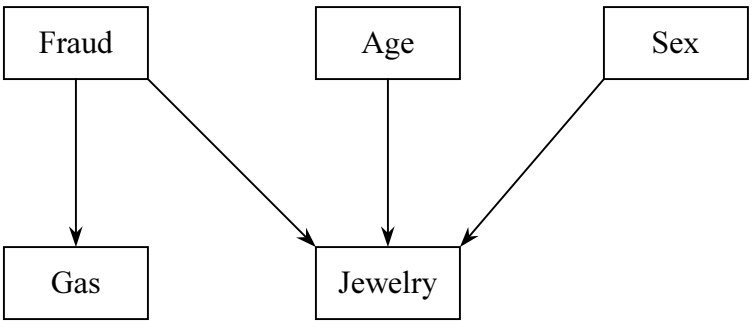

Figure 3. The Bayesian network for detecting credit-card fraud

Node

1	Fraud	2	0
2	Age	3	0
3	Sex	2	0
4	Gas	2	1
5	Jewelry	2	3

Parent

4	1
5	1
5	2
5	3

NodeContent

1	1	Yes
1	2	No
2	1	< 30
2	2	30 – 50
2	3	> 50
3	1	Male
3	2	Female
4	1	Yes
4	2	No
5	1	Yes
5	2	No

Figure 4. The data for the directed acyclic graph

Suppose that the conditional probabilities are given as follows:

$p(F = yes) = 0.00001$

$p(A < 30) = 0.25, \quad p(30 \le A \le 50) = 0.40$

$p(S = male) = 0.5$

$p(G = yes|F = yes) = 0.2, \qquad p(G = yes|F = no) = 0.01$

$p(J = yes|F = yes, A = *, S = *) = 0.05$

$p(J = yes|F = no, A < 30, S = male) = 0.0001$

$p(J = yes|F = no, 30 \le A \le 50, S = male) = 0.0004$

$p(J = yes|F = no, A > 50, S = male) = 0.0002$

$p(J = yes|F = no, A < 30, S = female) = 0.0005$

$p(J = yes|F = no, 30 \le A \le 50, S = female) = 0.0002$

$p(J = yes|F = no, A > 50, S = female) = 0.0001.$

If the probability of interest is $p(F = yes|A < 30, S = male, G = no, J = no)$, then we have

$$p(F = \text{yes} \mid A < 30, S = \text{male}, G = \text{no}, J = \text{no})$$

$$= \frac{p(F = \text{yes}, A < 30, S = \text{male}, G = \text{no}, J = \text{no})}{p(A < 30, S = \text{male}, G = \text{no}, J = \text{no})}.$$

The numerator can be evaluated as

$$p(F = \text{yes}, A < 30, S = \text{male}, G = \text{no}, J = \text{no})$$

$$= p(J = \text{no} \mid F = \text{yes}, A < 30, S = \text{male}) p(G = \text{no} \mid F = \text{yes}) p(S = \text{male})$$

$$p(A < 30) p(F = \text{yes})$$

$$= 0.95 \times 0.8 \times 0.5 \times 0.25 \times 0.00001 = 9.5 \times 10^{-7},$$

and similarly for the denominator. Thus, any query in this Bayesian network can be expressed as a combination of the conditional probabilities of the variables.

Suppose that the parameter vector corresponding to each variable in this Bayesian network has a Dirichlet prior. If we choose $\alpha = 30$ for the Dirichlet prior of *Age*, then the parameter vector $\theta(A)$ will have a bivariate Dirichlet distribution $D_2(7.5, 12.0; 10.5)$. Since node *Age* has no parent nodes, we will use node *Jewelry* to show the data structure for the conditional probabilities. If we choose $\alpha = 10$ for the Dirichlet prior of J given F = "No", A = "30-50", and S = "Male", then $\theta(J|F = \text{no}, 30 \le A \le 50, S = \text{male})$ will have a beta distribution with parameters 0.004 and 9.996. The data in tables **ProbMaster**, **ProbDetail**, **Probability**, and **Parameter** for the conditional probabilities of J given F = "No", A = "30-50", and S = "Male" are:

ProbMaster: (5, 9, 10);

ProbDetail: (5, 9, 1, 2), (5, 9, 2, 2), (5, 9, 3, 1);

Probability: (5, 9, 1, 0.0004), (5, 9, 2, 0.9996);

Parameter: (5, 9, 1, 0.004, 0.004), (5, 9, 2, 9.996, 9.996).

The data in each pair of parentheses represent a record in the table. Note that the value of column 'Posterior' in table **Parameter** is set to be the value of column 'Prior' before the Bayesian network training starts.

Suppose that the pre-determined order for the parents of node *Jewelry* is π_J = {*Fraud*, *Age*, *Sex*} consistent with the ascending order of their node IDs. Hence, we have $n_1 = 2$, $n_2 = 3$, and $n_3 = 2$, and there are 12 possible parent outcome combinations of node *Jewelry*. When the values of these three parent nodes are "No", "30-50", and "Male", respectively, the representation of this parent outcome combination is {2, 2, 1}; i.e., $a_1 = 2$, $a_2 = 2$, and $a_3 = 1$. The combination ID of c = {2, 2, 1} is $s_c = (2-1) \times 3 \times 2 + (2-1) \times 2 + 1 = 9$. If the original format of the first case in the training data is (F = "No", A = "30-50", S = "Male", G = "No", J = "Yes"), then the five records in table **Case** corresponding to this case are

Case: (1, 1, 2), (1, 2, 2), (1, 3, 1), (1, 4, 2), (1, 5, 1).

By using the combination ID s_c, we will be able to access the record (5, 9, 1, 0.004, 0.004) in table **Parameter** for node *Jewelry* instantly, and the new values of this record after the Bayesian updating are (5, 9, 1, 0.004, 1.004).

This example has been implemented in a personal computer. We select HUGIN as a graphical user interface for drawing the Bayesian network and compile the text file from the output of HUGIN to obtain the model data of the Bayesian network. All of the model data are stored in a database created by Microsoft Access. After the Bayesian updating, we use the posterior data to construct a new text file for HUGIN. Then users can query any probability of interest from HUGIN. This process reveals that using our

approach to build a Bayesian network with the capability of learning is not difficult. Figure 5 shows that the elapsed time of the Bayesian updating is a linear function of the number of cases in the training data set. This experimental result is consistent with the analytical discussion given in Section 3.

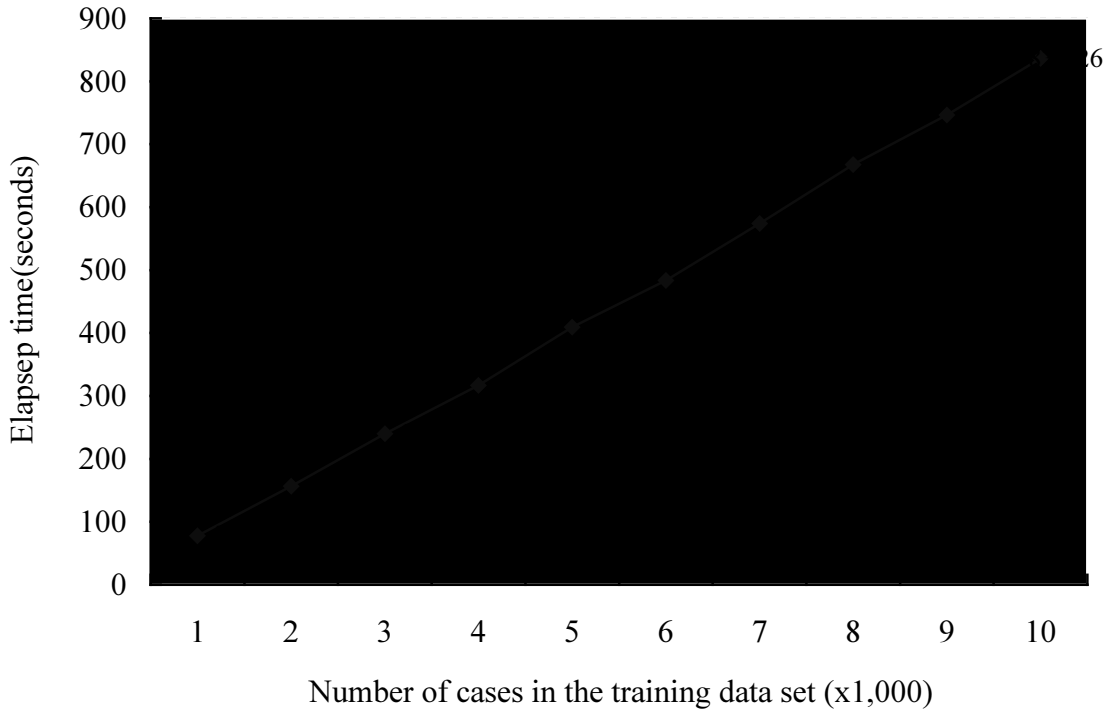

Figure 5. Elapsed time of the Bayesian updating

5 Discussion

The data structure for handling the model data of a Bayesian network is important for the performance of Bayesian updating. In this paper, we present a relational database scheme for arbitrary Bayesian networks with discrete variables. This scheme not only

provides a mechanism for Bayesian updating, but also allows incomplete training data. Since searching for a correct record in a table for Bayesian updating can be time-consuming, we proposed a direct access method to resolve this problem. The tools that we choose to build Bayesian networks are widely available, and the implementation of our relational database scheme is not difficult.

Model selection is one of the most interesting problems in Bayesian networks. Our future research will try to enhance the database scheme given in Section 3 for model selection. In implementing a method for model selection, we generally need to evaluate the model score for each possible Bayesian network. However, evaluating the model score of each possible Bayesian network can be very complex, and the number of possible Bayesian networks can be extremely large when the number of variables in a system is large. Most of the methods for model selection start with an arbitrarily chosen network and improve the network according to some rules (Spirtes and Meek, 1995; Friedman *et al*., 1997; Kearns *et al*., 1997). Thus, developing a database scheme and identifying necessary operations that can increase the performance for those methods is critical.

References

Date, C. J. (1995), *An Introduction to Database Systems*, 6th Edition, Addison-Wesley.

Friedman, N., Geiger, D., and M. Goldszmidt (1997), Bayesian Network Classifiers, *Machine Learning*, 29, 131-163, 1997.

Friedman, N. and Goldszmidt, M. (1997), Sequential Update of Bayesian Network Structure, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*.

Heckerman, D. (1995), A Tutorial on Learning with Bayesian Networks, *Microsoft Research Technical Report* MSR-TR-94-09, Microsoft Research, Redmond, Washington.

Heckerman, D., Geiger, D., and Chickering, D. (1995), Learning Bayesian Networks: The Combination of Knowledge and Statistical Data, *Machine Learning*, 20, 197-243.

Kearns, M., Mansour, Y., Ng, A. Y., and Ron, D. (1997), An Experimental and Theoretical Comparison of Model Selection Methods, *Machine Learning*, 27, 7-50.

Spirtes P. and Meek C. (1995), Learning Bayesian Networks with Discrete Variables from Data, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 294-299.

Wong , T. T. (1998), Generalized Dirichlet Distribution in Bayesian Analysis, *Applied Mathematics and Computation*, 97, 165-181.