Study of Online Bayesian Networks Learning in a Multi-Agent System

Yonghui CAO[1, 2]
1, School of Economics & Management, Henan Institute of Science and Technology, Xin Xiang, 453003 ,China
2, School of Management, Zhejiang University, Hang Zhou,310058 ,China

Abstract
This paper introduces online Bayesian network learning in detail. The structural and parametric learning abilities of the online Bayesian network learning are explored. The paper starts with revisiting the multi-agent self-organization problem and the proposed solution. Then, we explain the proposed Bayesian network learning, three scoring functions, namely Log-Likelihood, Minimum description length, and Bayesian scores.

Keywords:*Bayesian Network, Search Algorithms, Heuristic Search, Exhaustive Search*

1. Introduction

We attempt to find how a common task can be performed by a multi-agent self-organizing system. The agents are independent in terms of their model of environment and their actions. Each agent explores the environment and decides its actions by itself. Agents will have no information about the environment at the beginning of their exploration of the environment. They will explore the environment, model the environment and take actions to change the environment according to the common task. We attempt to solve these problems by utilizing Bayesian networks and influence diagrams.

Bayesian networks are employed to model the environment. Because the agents have no or limited information about the environment at the beginning of their exploration, an online Bayesian network learning method will be used. Influence diagrams will be employed to obtain the agents' actions. Bayesian networks and influence diagrams are combined to produce a decision-theoretic agent in a multi-agent system.

Bayesian network learning is examined broadly. There are four cases of Bayesian network learning depending on the availability of the network and the data. The unknown structure and incomplete data case is the nearest case to our problem. Our network structure is not defined in advance and the sensor data may not be complete. On the other hand, for simplicity we will assume the data is complete during the simulations. The agents do not have significant amounts of prior knowledge about the environment. Therefore, the BN will be formed during the agents' exploration of the environment. Each new data case will affect the structure of the network.

Online Bayesian network learning consists of two parts, namely parameter learning and structural learning. Parameter learning is the calculation of the conditional probability table elements of each node in a given Bayesian network. In this research, we use a modified version of Maximum Likelihood Expectation method to calculate the network parameters. Maximum likelihood estimation method is modified so that it has a closed form when the probabilities need to be updated.

Structural learning is the problem of finding the network that represents the data the best. This involves two parameters, complexity of the network and fitness of the network to the data. The structural learning process tries to find the optimal network that provides optimal complexity and fitness. The main building block in structural learning is the search algorithm that generates the network with the highest score.

2 .The Parameter Learning

There are two types of parameter learning techniques used in the literature, MLE and Bayesian estimation. It is stated that with a database having a large number of data cases, these two methods converge to each other. The latter can take prior knowledge if it is available. Also, it is shown that the latter has a closed form. In this section, we have redefined the Maximum Likelihood calculation to have a closed form calculation. Because MLE is computationally simpler than Bayesian estimation, it is employed in our parameter learning. The following paragraphs explain how the parameter learning is performed by modified MLE method.

Let $X = \{X_1, X_2, \cdots X_m\}$ be the discrete variables (nodes) in a Bayesian network, B. Assume that we know that the node X_j is the child of the node X_i, which means $X_i \rightarrow X_j$ .In this case, the parameter learning has to calculate the values in the conditional probability table in the node X_j. The conditional probability can be calculated by utilizing using the fundamental formula for probability calculus as in Equation (1)

$$P(X_i \mid X_j) = \frac{P(X_i, X_j)}{P(X_i)} \quad (1)$$

Since MLE is employed in parameter learning, the probabilities can be calculated by utilizing the natural frequencies of the data cases. A natural frequency of a data case is calculated by counting the number of occurrences of the data case in the database. For individual probabilities, we count the number of occurrences of a state of a variable in the database. Let n_{ij} be the number of occurrences of the state j of the ith variable in the database and n is the total number of data cases in the database. Using these frequency values, we can calculate the probabilities in the following way:

$$P(X_i = x_j) = \frac{n(X_i = x_j)}{n} = \frac{n_{ij}}{n} \quad (2)$$

Thus, the conditional probabilities can be calculated by using the individual probabilities in Equation (1). The conditional probability $P(X_i \rightarrow X_j)$ can be obtained as in the following equations.

$$P(X_i \mid X_j) = \frac{P(X_i, X_j)}{P(X_j)} \quad (3)$$

$$P(X_i, X_j) = \frac{n(X_i, X_j)}{n} \quad (4)$$

$$P(X_i) = \frac{n(X_j)}{n} \quad (4.5)$$

As can be seen in Equations (4) and (5), the denominators are the same in the both terms. When we put these two terms into Equation (3), the denominators cancel each other as shown in the following equation.

$$P(X_i \mid X_j) = \frac{n(X_i, X_j) \,/\, n}{n(X_j) \,/\, n} = \frac{n(X_i, X_j)}{n(X_j)} \quad (6)$$

In the resulting equation, there are only two natural frequencies. There is no need to involve the number of elements in the database for conditional probability calculations. This technique simplifies the computations in the parameter learning. Equation (6) has a closed form because if a new data case is encountered, we can easily update the corresponding natural frequencies accordingly to update the conditional probabilities. The following example provides practical results to the conditional probability calculation technique. For the cases that have not seen yet, the uniform probability distribution is used to fill the conditional probability tables in the nodes. For online Bayesian network learning, the parameter learning is not enough because the agents do not know the system dynamics in advance. Thus, the structural learning part is also necessary to discover the system dynamics.

3. The structural learning

Structural learning is finding the best network that fits the available data and is optimally complex. This can be accomplished by utilizing a search algorithm over the possible network structures. In this research, a greater importance is given to the search algorithm because we have assumed that the data will be complete. That is, each element of the database is a valid state of a variable. If there are non-applicable entries in the database, then the database is said to be incomplete.

The greedy search algorithm is employed to accomplish the structural learning in the online Bayesian network learning. The search algorithm is a score based searching algorithm. The search algorithm is evaluated in terms of the score function used and the technique used to create the candidate networks, such as adding an edge and removing an edge. The greedy search algorithm is also upgraded to have some online properties such as updating the network parameters and its structure adaptively.

The algorithm is a generic greedy search algorithm. How the arc addition is done and which scoring method is used are not specified in the above algorithm. We explore the search algorithms used in this research. In the algorithms, the arcs are added heuristically and exhaustively.

3.1 Search Algorithms

A Bayesian network is not allowed to have a cycle because of the computational difficulties. A cycle in a Bayesian network leads to a "circular reasoning" between the variables. For example, if the dependencies in above

network are: $X_1 \rightarrow X_2$, $X_2 \rightarrow X_3$, and $X_3 \rightarrow X_1$, a cycle will be formed. If evidence is entered into the variable X_1, the Bayesian network will run the evidence to X_2, then to X_3. Then, the evidence will travel to X_1 because X_1 depends on X_3. The evidence may run in the network forever because all the variables depend on each other in a circular way.

A heuristic arc addition is employed not to have a cycle in the Bayesian network while generating the Bayesian structure. An exhaustive arc addition is also employed to explore more network possibilities without limitation. In the exhaustive arc addition algorithm, a cycle check is employed before and arc is added. The following section presents the details of heuristic and exhaustive search algorithms.

(1) Heuristic Search

In the heuristic search algorithm, the variables of the system have to be ordered in a certain way to prevent cycles from being created. The decision variables should be in the last columns in the database; and, the first columns of the database should be filled with the variables without parents, independent variables. After placing the independent variables in the first columns, the children of the independent variables should be placed in the following columns. The rest of the columns are filled with the children of the previously placed variables. Ordering of the variables is necessary because the heuristic arc addition adds the arcs from the first variables to the last variables. Because of the ordering, we need to have some knowledge about the variables. This does not mean that we need to know the dependencies between the variables. For example, let B be a Bayesian network with three variables, $\{X_1, X_2, X_3\}$.If we know the variable X_1 is the first variable and the variable X_2 is the decision node. Then the column order will be $\{X_1, X_2, X_3\}$.

The heuristic search starts with adding and removing arcs from the each variable to the last variable. Let the network have n variables. After adding an arc, the algorithm calculates the network score, records the score in a list, and removes the arc. The algorithm finds the arc that gives the highest increase in the network score. Let us assume that the arc from the kth variable to the last variable, n, gives the highest increase in network score. Then, the algorithm adds the arc from the kth variable to the last variable. After the arc is added, the algorithm adds and removes arcs from the remaining variables to the last variable. Then, the algorithm chooses the arc with the highest score increase and adds the arc to the network. This continues until no increase in the network score can be obtained by adding an arc to the last variable. Then, the algorithm starts adding arcs from the variables $\{1, 2, \cdots n-2\}$ to the $(n-1)th$ node. The algorithm adds arcs to $(n-1)th$ node until there is no increase in the network score. The algorithm stops when it adds an arc from the first variable to the second variable. The following is the heuristic search algorithm used in this research.

(1)Collect data
(2)Define the variables from the available data
(3)Start with a network with no arc.
(4)Estimate the parameters (only independent probabilities) of the BN using the MLE method using initial data
(5)Add a new arc from the ith variable to the jth variable to generate a network candidate and remove the arc. Repeat the process with $i = \{1, 2, \cdots j-1\}$ and generate networks $(B_1, B_2, \cdots, B_{j-1})$. Start j from n and decrease j by 1.
(6)Calculate the scores of the candidate networks and record them in a list.
(7)Find the network (B) with the maximum score and keep it for the next step.
(8)Repeat the steps 5, 6, and 7 until there is no increase in the network score.
(9)If $j > 1$, then go to step 5.
(10)Update the network parameters along with new data
(11)Update the network structure:

If enough new data obtained, go to step 1 and generate a new network structure.

If no structural update is necessary go to step 10.

Consequently, the heuristic search algorithm adds arcs only in the forward direction because this protects the network from having cycles and complex network structure. On the other hand, there is a price of arranging the variables at the creation of the database in the heuristic algorithm. Since the agents will not have much knowledge about the environmental variables, it is hard to arrange the variables at the beginning. There is a need for a better search algorithm that explores more possibilities in the network. The following paragraph introduces

another searching algorithm that eliminates the arranging the variables, namely exhaustive search.

(2) Exhaustive Search

The exhaustive search algorithm explores all the possible arcs in the network during its execution. The algorithm starts adding arcs from the ith variable to the jth variable where $i=\{1,2,\cdots,n\}$, $j=\{1,2,\cdots,n\}$, $i\neq j$. This covers $n\cdot(n-1)$ arcs throughout the network. The algorithm calculates the network score for each arc addition. Then, it chooses the arc with the highest increase in the network score. The algorithm repeats the above steps until there is no increase in the network score.

There are two major drawbacks in the exhaustive search algorithm. First, the number of arcs to be tried might become intractable when the number of variables is large. Second, during the search, the algorithm might introduce cycles to the network because it can add an arc in any direction. An additional algorithm is incorporated to the search algorithm to keep track of cycles. Using the additional algorithm, the search algorithm checks whether the new arc introduces a cycle or not. If the arc introduces a cycle, the algorithm does not add the arc to the network. The following is the exhaustive search algorithm used in this research.

(1)Collect data
(2)Define the variables from the available data
(3)Start with an empty network
(4)Estimate the parameters (only independent probabilities) of the BN using the MLE method using initial data
(5)Add a new arc from the ith variable to the jth variable to create a candidate network and remove the arc. Repeat the process for every value of i and j where $i=\{1,2,\cdots,n\}$, $j=\{1,2,\cdots,n\}$, $i\neq j$.This step creates m possible networks $(B_1,B_2,\cdots,B_m)$ .Algorithm creates $m=n\cdot(n-1)$ networks in first visit to step 5.
(6)Remove the network with cycles from the candidate fist.
(7)Calculate the scores of the candidate networks and record it in a list.
(8)Find the network (B) with the maximum score and keep it for the next step.
(9)Do step 5 through 8 until there is no increase in the network score.
(10)Update the network parameters along with new data
(11)Update the network structure:
If enough new data obtained, go to step 1 and generate a new network structure.
If no structural update is necessary go to step 10.

The search algorithms are explained in detail. There is a need to analyze the complexity of the search algorithm before there are implemented. The following section gives the complexity analysis of both search algorithms.

(3) Complexity Analysis for Search Algorithms

As stated earlier, the heuristic search algorithm needs prior knowledge about the variables in terms of their order in the database. On the other hand, the number of iterations in the heuristic search algorithm may be tractable. In the heuristic search, the algorithm tries $(n-1)$ arcs in the first trip from step 5 to step 7. The algorithm repeats steps 5 through 7 until there is no increase in the network score. Assuming the algorithm adds an arc in every trip, the number of arcs tried will be one less then the previous trip. Algorithm can repeat step 5 through 7 at most $(n-1)$ times. In $(n-1)$ trips, the algorithm generates $(n-1)+(n-2)+\cdots+1$ networks candidates. When the algorithm reaches step 8, the algorithm loops back to step 5 and repeats the same process for the variables $\{X_{n-1},X_{n-2},\cdots,X_2\}$ .Therefore, after the first loop, the algorithm generates $(n-1)+(n-2)+\cdots+1$ network candidates. The complexity of the heuristic search algorithm is denoted as C_h.

In the following complexity analysis, each loop shows the number of network candidates tried until the algorithm reaches to the step 8. Since the algorithm will repeat itself for $(n-1)$ variables, the analysis has $(n-1)$ loops as the following.

Loop 1

$$(n-1)+(n-2)+\cdots+1=n(n-1)-(1+2+\cdots+(n-1))$$
$$=n(n-1)-\frac{n(n-1)}{2}=\frac{n(n-1)}{2}$$

Loop 2

$$(n-2)+(n-3)+\cdots+1=\frac{(n-1)(n-2)}{2}$$

$\vdots$

$\vdots$

Loop (n-1)

$$\frac{(n-(n-1))(n-(n-2))}{2}=1$$

If we add the number of candidate networks from each loop, the following can be obtained:

$$C_n = \frac{n(n-1)+(n-1)(n-2)+\cdots+(n-(n-1))(n-(n-2))}{2}$$

$$C_n = \frac{2(n-1)^2+2(n-3)^2+\cdots+2(n-(n-2))^2}{2}$$

Then, we can further modify the equation as follows:

$$C_n = (n-1)^2+(n-3)^2+\cdots+2(n-(n-2))^2 \quad (7)$$

Since each element in C_n is less than n^2. We can state that

$$C_n < n^2(n-3) < n^3 \quad (8)$$

Equation (8) illustrates the complexity of the heuristic search. The following paragraphs will explore the complexity of the exhaustive search algorithm.

The exhaustive search algorithm tries every possible arc in the network during its first visit to step 5. In a graph with n nodes, there can be $n(n-1)$ possible directed edges in the graph. Therefore, the algorithm generates $n(n-1)$ network candidates and the complexity of the first visit is $n(n-1)$.Then the algorithm continues until it reaches to step 9 and loops back to step 5 until there is no increase in the network score.

After the first loop, the complexity decreases by 1 in each step because the algorithm will not try the arc added in the previous step. The following presents the complexity analysis of the exhaustive search algorithm. First, the complexity is calculated for each loop. Then, they are added to obtain the complexity of the algorithm.

Loop 1 $n(n-1)$

Loop 2 $n(n-1)-1$

⋮

⋮

Loop N $n(n-1)-N+1$

The exhaustive search algorithm does not perform a certain number of loops. The algorithm will continue until there is no increase in the network score. Therefore, we will assume that the algorithm end after N loops for the complexity calculations. If we add the complexities of all the loops together, the complexity of the exhaustive search, C_e, becomes the following.

$$C_e = n(n-1)N-(1+2+\cdots+(N-1)) \quad (9)$$

$$C_e = n(n-1)N-\frac{N(N-1)}{2} \quad (10)$$

If the network has great number of arcs, then the complexity of the algorithm becomes large. For example, if the algorithm ends in step $N=n$, the complexity becomes

$$C_e = n^2(n-1)-\frac{n(n-1)}{2} = \frac{2n^2(n-1)-n(n-1)}{2} \quad (11)$$

$$C_e = \frac{(n-1)n(2n-1)}{2} \quad \text{for } n=N \quad (12)$$

In general, number of nodes in a Bayesian network, n, is much larger than 1. Therefore, we can reevaluate the complexity by assuming $n>>1$ .The following equation represents the computational complexity of the exhaustive search algorithm when the number of steps is equal to the number of variables.

$$C_e \cong \frac{n\cdot n\cdot 2n}{2} = \frac{2n^3}{2} \Rightarrow C_e \cong n^3 \quad (13)$$

As can be seen above, the complexity of the exhaustive algorithm is larger than the complexity of the heuristic algorithm when $N=n$.

For the networks with large number of variables (nodes), the algorithm does not stop when $N=n$.Let us calculate the worst case scenario for the exhaustive algorithm. The algorithm might explore all possible arcs in the network, which is equal to $n(n-1)$.This is true because a complete graph with n nodes has $n(n-1)$ possible directed edges. Therefore, we will replace N with $n(n-1)$ in the complexity analysis. Then, the complexity of the exhaustive search algorithm becomes the following.

$$C_e = n(n-1)N-\frac{N(N-1)}{2} = n(n-1)n(n-1)-\frac{n(n-1)(n(n-1)-1)}{2} \quad (14)$$

$$C_e = \frac{2n^2(n-1)^2-n^2(n-1)^2-n(n-1)}{2} = \frac{n^2(n-1)^2-n(n-1)}{2} \quad (15)$$

We can simplify the equation above by assuming $n>>1$. In this case, the complexity of the algorithm becomes the following.

$$C_e \cong \frac{n^2\cdot n^2-n^2}{2} = \frac{n^2(n-1)^2}{2} \Rightarrow C_e \cong \frac{n^4}{2} \quad (16)$$

Two search algorithms are introduced to learn the structure of a Bayesian network in the previous sections. The heuristic search algorithm is simple and explores a limited number of network structures. On the other hand, the exhaustive search algorithm is complex and explores many possible network structures. The complexity of the exhaustive algorithm is approximately n-fold larger than the complexity of the heuristic search algorithm.

3.2 Network scoring functions

Three scoring functions are employed in this research, namely Log-Likelihood, Minimum description length (MDL), and Bayesian (BDE) scores. The Log-Likelihood method measures the likelihood of the network given the available data. The MDL also uses likelihood of the network but it includes the measure of the network's complexity. The Bayesian score involves the calculation of the probability of a network given the data. Bayesian scoring method also penalizes complex networks as the MDL scoring. If the length of the database is large enough these two methods converge to each other. The following sections provide the details of the scoring methods used in the research.

(1) Log-Likelihood Scoring

The Log-Likelihood score of a network, B, is obtained by calculating the likelihood of the data, D, given the network, B, and the network parameters, q_B. After calculating the likelihood of the data, a natural logarithm is applied to get the Log-Likelihood of the data. The following formulas explain the details of the Log-Likelihood calculation.

$$Score_L(B:D) = L(D|B,q_B) \quad (17)$$

$$L(D|B,q_B) = \prod_m P(d[m]|B,q_B) \quad (18)$$

In the above formula, $d[m]$ represents the mth data case in the database. Let us take the logarithm of the likelihood. The logarithm converts the multiplication in to a summation.

$$l(D|B,q_B) = \log L(D|B,q_B) \quad (19)$$

$$l(D|B,q_B) = \sum_m \log P(d[m]|B,q_B) \quad (20)$$

This is basically equal to calculating the probability of each data case in the database, taking their logarithms and adding them together. For example, assume that the network given in the previous section has the relations $X_1 \to X_3$ and $X_3 \to X_2$. Then, we can calculate the log-likelihood of the data with the following equation.

$$\begin{aligned} l(D|B,q_B) &= \sum_m \log P(X_1[m]|q_{X_1}) \\ &+ \sum_m \log P(X_3[m]|x_{10},q_{X_2}) + \sum_m \log P(X_3[m]|x_{11},q_{X_3}) \\ &+ \sum_m \log P(X_2[m]|x_{30},q_{X_2|x_{30}}) + \sum_m \log P(X_2[m]|x_{31},q_{X_2|x_{31}}) \end{aligned} \quad (21)$$

In the log-likelihood approach, the score of the network increases as long as the length of the database and the number of arc in the network increase. Therefore, the search algorithm tries to add as many arcs as possible to the network to get the highest scoring network. At the end of the search, the algorithm ends up with almost a complete network. For the networks with a large number of nodes, this might cause a great increase in complexity of the network. To overcome the complexity problem, we need to find out a way to include the complexity of the network to the scoring function. If the network gets complex, the scoring function should decrease accordingly. The following scoring method handles the complexity problem by introducing the complexity parameter in the scoring function.

(2) Minimum Description Length Scoring

The MDL method combines the likelihood of the data and the complexity of the network to find optimally complex and accurate networks. The MDL method penalizes networks with complex structures. The MDL has two parts, the complexity of the network, $L_{NETWORK}$, and the likelihood of the data, L_{DATA}. Then, the MDL score can be calculated by the following.

$$Score_{MDL} = L_{DATA} - L_{NETWORK} \quad (22)$$

The complexity part involves the dimension of the network, $Dim(B)$, and structural complexity of the network, $DL(B)$. The dimension of the network can be calculated using the number of states in each node, S_i. The following equation illustrates the dimension of the network.

$$Dim(B) = \sum_{i=1}^{N} (S_i - 1) \prod_{j \in pa(x_i)} S_j \quad (23)$$

Where N is the number of nodes in the network. Let M be the number of data cases in the database. Using the central limit theorem, each parameter has a variance of $\sqrt{M}$. Thus, for each parameter in the network, the number of bits required is given by the following.

$$d = \log \sqrt{M} \Rightarrow d = \frac{\log M}{2} \quad (24)$$

The structural complexity of the network depends on the number of parents of the nodes. The following formula calculates the structural complexity.

$$DL(B) = \sum_{i=1}^{N} k_i \log_2(N) \quad (25)$$

Where k_i is the number of parents the node X_i has. Finally, the following formula presents the complexity part of the MDL score by combining the dimension of the network and the structural complexity.

$$L_{NETWORK} = \frac{\log M}{2} Dim(B) + DL(B) \quad (26)$$

$$L_{NETWORK} = \frac{\log M}{2}\left[\sum_{i=1}^{N}(S_i - 1)\prod_{j \in pa(x_i)} S_j\right] + \sum_{i=1}^{N} k_i \log_2(N) \quad (27)$$

The likelihood of the data needs to be defined after presenting the network complexity part of the MDL score. The likelihood of the data given a network can be calculated by using cross-entropy. The difference between the distribution of the data (P) and the estimated distribution (Q) is from the network. Kullback-Leiber and Euclidean distance are the commonly used cross-entropy methods. Therefore, the likelihood of a data can be calculated by measuring the distance between two distributions. If we use the Kullback-Leiber cross-entropy, the likelihood of the data can be calculated by the following.

$$l(D|B,q_B) = \sum_{i=1}^{M} p_i \log \frac{p_i}{q_i} \quad (28)$$

$$L_{DATA} = \sum_{i=1}^{M} p_i \log \frac{p_i}{q_i} \quad (29)$$

Where p_i is the probability of data case i using the database and q_i is the estimate of the probability of data case i from the network parameters. If Euclidean distance measure is employed to calculate the distance between the distributions, the likelihood of the data is calculated by the following.

$$l(D|B,\hat{q}_B) = \sum_{i=1}^{M} (p_i - q_i)^2 \quad (30)$$

$$L_{DATA} = \sum_{i=1}^{M} (p_i - q_i)^2 \quad (31)$$

After defining the likelihood and complexity parts, the MDL score can be given as

$$Score_{MDL}(B:D) = l(D|B,q_B) - \frac{\log M}{2} Dim(B) - DL(B) \quad (32)$$

(3) Bayesian Scoring

Another commonly used scoring method is Bayesian score. Now, we will provide the details of the Bayesian scoring technique. Bayesian scoring is calculated by utilizing the Dirichlet parameters of the network.

Bayesian statistics tells us that we should rank a prior probability over anything we are uncertain about. In this case, we put a prior probability both over our parameters and over our structure. The Bayesian score can be evaluated as the probability of the structure given the data:

$$Score_{BDE}(B:D) = P(B|D) = \frac{P(D|B)P(B)}{P(D)} \quad (33)$$

The probability $P(D)$ is constant. Therefore, it can be ignored when comparing different structures. Thus, we can choose the model that maximizes $P(D|B)P(B)$ .Let us assume that we do not have prior over the network structures. Assume that we have uniform prior over the structures. One might ask whether we get back to the maximum likelihood score. The answer is 'no' because the maximum likelihood score for B was $P(D|B,q_B)$, i.e. the probability of the data in the most likely parameter instantiation. In Bayesian scoring, we have not given the parameters. Therefore, we have to integrate over all possible parameter vectors:

$$P(D|B) = \int P(D|q_B, B)P(q_B|B)dq_B \quad (34)$$

This is, of course, different from the maximum likelihood score. To understand the Bayesian scoring better, consider two possible structures for a two-node network, where $B_1 = [A\ B]$ and $B_2 = [A \rightarrow B]$ .Then, the probability of the data given the network structures can be calculated by the following equations.

$$P(D|B_1) = \int_0^1 P(q_A, q_B)P(D|[q_A, q_B])d[q_A, q_B] \quad (35)$$

$$P(D|B_2) = \int_0^1 P(q_A, q_{B|q_0}, q_{B|q_1})P(D|[q_A, q_{B|q_0}, q_{B|q_1}])d[q_A, q_{B|q_0}, q_{B|q_1}] \quad (36)$$

The latter is a higher dimensional integral, and its value is therefore likely to be somewhat lower. This is because there are more numbers less than 1 in the multiplication. Multiplying the numbers less than 1 results in a number smaller than any of the number in the multiplication. For example, multiplying three small numbers (less than 1)is likely to be smaller than the number obtained by multiplying two small numbers (less than 1). Since the probabilities in the integrals are less than 1, the above argument applies to the integrals. Therefore, it can be said that the higher dimensional integral is likely to have lower value that the lower dimensional integral. This idea presents preference to the networks with fewer parameters. This is an automatic control in the complexity of the network.

Let us analyze $P(D|B)$ a little more closely to understand the Bayesian score calculations. It is helpful to first consider the single parameter case even though there is no structure learning to learn there. In that case, there is a

simple closed form solution for the probability of the data given by the following.

$$P(D) = \frac{\Gamma(a)}{\Gamma(a_0 + a_1)} \cdot \frac{\Gamma(a_0 + n_0) \cdot \Gamma(a_1 + n_1)}{\Gamma(a + n)} \quad (37)$$

Where $\Gamma(m)$ is equal to $(m-1)!$ for an integer m, n is the number of data cases in the database, n_0 and n_1 are the number of zeros and ones, respectively, and $a = a_0 + a_1$. Let us assume we have 40 zeros and 60 ones in the database. Assuming that we have uniform priors, $a_0 = a_1 = 3$, the probability of data is

$$P(D) = \frac{\Gamma(6)}{\Gamma(3)\Gamma(3)} \cdot \frac{\Gamma(3+40) \cdot \Gamma(3+60)}{\Gamma(6+100)} \quad (38)$$

The probability for a structure with several parameters is simply the product of the probabilities for the individual parameters. For example, in our two-node network, if the same priors are used for all three parameters, and we have 45 zeros and 55 ones for the variable B, then, the probability of the data for the network B_1 can be calculated as

$$P(D|B_1) = \frac{\Gamma(6)}{\Gamma(3)\Gamma(3)} \cdot \frac{\Gamma(43) \cdot \Gamma(43)}{\Gamma(106)} \cdot \frac{\Gamma(6)}{\Gamma(3)\Gamma(3)} \frac{\Gamma(48) \cdot \Gamma(58)}{\Gamma(106)} \quad (39)$$

For the second network, let us assume that $a_{00} = 23$, $a_{01} = 22$, $a_{10} = 29$ and $a_{11} = 26$, where $a_{ij} = n(a_i, b_j)$ is the number of cases with $A = a_i$ and $B = b_j$. Then, we can compute the probability of the data for the network B_2 using following equation.

$$P(D|B_2) = \frac{\Gamma(6)}{\Gamma(3)\Gamma(3)} \cdot \frac{\Gamma(43) \cdot \Gamma(43)}{\Gamma(106)} \cdot \frac{\Gamma(6)}{\Gamma(3)\Gamma(3)} \frac{\Gamma(23+3) \cdot \Gamma(22+3)}{\Gamma(45+3)} \cdot \frac{\Gamma(6)}{\Gamma(3)\Gamma(3)} \frac{\Gamma(29+3) \cdot \Gamma(26+3)}{\Gamma(55+3)} \quad (40)$$

The intuition is clearer. The analysis shows that we get a higher score by multiplying a smaller number of bigger factorials rather than a larger number of small ones.

It turns out that if we approximate the log posterior probability, and ignore all terms that do not grow with M, we can obtain

$$\log P(D|B) = l(D|q_B, B) - \frac{\log M}{2} Dim(B) \quad (41)$$

i.e, as M grows large, the Bayesian score and the MDL score converge to each other using Dirichlet priors. In fact, if we use a good approximation to the Bayesian score, and eliminate all terms that do not grow with M, then we are left exactly with MDL score. Therefore, it can be concluded that the Bayesian score gives us, automatically, a tradeoff between network complexity and fit to the data. The Bayesian score is also decomposable like the MDL score since it can be expressed as a summation of terms that corresponds to individual nodes. In this research, we have decomposed the Bayesian score to make efficient calculations and a uniform distribution is employed for Dirichlet priors. The simulation results will show that the Bayesian score provides optimally complex and accurate network structures.

4 .Conclusions

Structural learning is finding the best network that fits the available data and is optimally complex. This can be accomplished by utilizing a search algorithm over the possible network structures. A greater importance is given to the search algorithm because we have assumed that the data will be complete. That is, each element of the database is a valid state of a variable. If there are non-applicable entries in the database then the database is said to be incomplete. We explore the search algorithms used in this research. In the algorithms, the arcs are added heuristically and exhaustively.We calculate the quality (score) of the networks to find the best network. In this paper, three scoring functions are employed, namely Log-Likelihood, Minimum description length (MDL), and Bayesian (BDE) scores. The Log-Likelihood method measures the likelihood of the network given the available data. The MDL also uses likelihood of the network but it includes the measure of the network's complexity. The Bayesian score involves the calculation of the probability of a network given the data.

Acknowledgments

This work is financially supported by the National Natural Science Foundation of China (Project No. 90718038). Thanks for the help.

References

[1] S. Russell and P. Norvig, Artificial Intelligence: A modern Approach, New Jersey: Prentice Hall, 1995.
[2] F. V. Jensen, an Introduction to Bayesian Networks. London, UK: University College London Press, 1996.
[3] D. Heckerman, "A tutorial on learning Bayesian networks," Technical Report MSR-TR-95-06, Microsoft Research, 1995.
[4] Y. Shoham, "Agent-oriented programming," Artificial intelligence, vol. 60(1), pp. 51-92, 1993.

[5] J. Pearl, "Bayesian networks", in M. Arbib (Ed.), Handbook of Brain Theory and Neural Networks, MTT Press, pp. 149-153, 1995
[6] J. Pearl, "Bayesian networks," Technical Report R-246, MTT Encyclopedia of the Cognitive Science, October 1997.
[7] F.V. Jensen, "Bayesian network basics," AISB Quarterly, vol. 94, pp. 9-22, 1996.
[8]W. Lam and F. Bacchus, "Learning Bayesian belief networks: an approach based on the MDL principle," Computational Intelligence, vol. 10, pp. 269-293, 1994.
[9] N. Friedman, M. Goldszmidt, D. Heckerman, and S. Russell, "Challenge: Where is the impact of the Bayesian networks in learning?" In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI), pp.10-15, 1997.
[10] N. Friedman, K. Murphy, and S. Russell, "Learning the structure of dynamic probabilistic networks," in G.F. Cooper and S. Moral (Eds.), Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98), San Francisco, CA: Morgan Kaufmann, 1998.
[11]G. F. Cooper and E. Herskovits, "A Bayesian method for constructing Bayesian belief networks from databases," in Proceedings the Conference on Uncertainty in AI, pp.88-94, 1990.
[12] B. Theisson, C. Meek, and D. M. Chickering, and D. Heckerman, "Learning mixtures of Bayesian networks," in G.F. Cooper and S. Moral (Eds.), Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98), San Francisco, CA: Morgan Kaufmann, 1998.
[13] N. Friedman, "The Bayesian structural EM algorithm," in G.F. Cooper and S. Moral (Eds.), Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98), San Francisco, CA: Morgan Kaufmann, 1998.
[14] D. Spiegelhalter, P. Dawid, S. L. Lauritzen, and R. Cowell, "Bayesian analysis in expert systems," Statistical Science, vol. 8, pp. 219-282, 1993.
[15] D. Heckerman, D. Gieger, and M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," Technical Report MSR-TR-94-09, Microsoft Research, Redmond, WA, 1994.
[16] C. Claus, "Dynamics of multi-agent reinforcement learning in Cooperative multi-agent systems," Ph.D. Dissertation, Univ. of British Colombia, Canada, 1997.
[17] S. Sen and M. Sekaran, "Multi-agent coordination with learning classifier systems," in Proceedings of the IJCAI Workshop on Adaptation and Learning in Multi-agent Systems, Montreal, pp. 84-89, 1995.
[18] C. Boutilier, "Planning, learning and coordination in multi-agent decision processes," in Sixth conference on Theoretical Aspects of Rationality and Knowledge (TARK'96), The Netherlands, 1996

Author Yonghui Cao received the MS degree in business management from Zhejiang University in 2006. He is currently a doctorate candidate in Zhejiang University. His research interest is in the areas of management information systems.