An Incremental Approach for Sparse Bayesian Network Structure Learning

Shuanzhu Sun[1], Zhong Han[2], Xiaolong Qi[2,3], Chunlei Zhou[1], Tiancheng Zhang[2], Bei Song[2], and Yang Gao[2](✉)

[1] Jiangsu Frontier Electric Technology Co. Ltd., Nanjing 211102, China
15905166613@139.com, 13851845492@163.com

[2] Department of Computer Science and Technology, Nanjing University, Nanjing 210046, China
763719732@qq.com, qxl_0712@sina.com, tiancheng_zhang@163.com, songbei07@gmail.com, gaoy@nju.edu.com

[3] Department of Electronics and Information Engineering, Yili Normal University, Yining 835000, Xinjiang, China

Abstract. A Bayesian network is a graphical model which analyzes probabilistic relationships among variables of interest. It has become a more and more popular and effective model for representing and inferring some process with uncertain information. Especially when it comes to the failure of uncertainty and correlation of complex equipment, and when the data is big. In this paper, we present an incremental approach for sparse Bayesian network structure learning. In order to analysis the correlation of heating load multidimensional feature factor, we use Bayesian network to establish the relationship between operating parameters of the heating units. Our approach builds upon previous research in sparse structure Gaussian Bayesian network, and because our project requires us to deal with a large amount of data with continuous parameters, we apply an incremental method on this model. Experimental results show that our approach is the efficient, effective, and accurate. The approach we propose can both deal with discrete parameters and continuous parameters, and has great application prospect in the big data field.

Keywords: Incremental approach · Sparse Bayesian network structure learning Big data · Correlation analysis

1 Introduction

Bayesian network is a complete model for the variables and their relationship, it can be used to answer probabilistic queries about them. Furthermore, it has a strong ability to deal with uncertain problems logically and understandably, and can make inferences from uncertain large amount of information [1]. Especially with the increasing computing power and the emergence of big data makes Bayesian network increasingly powerful. It shows great promise in the big data field.

Accordingly, our team's project is to design and develop a correlation model of heating load multidimensional feature factors and an online diagnose model for the monitoring data of large heating units. For the former model, for each specific unit parameter (such as heat supply), we need to describe the interaction between one and the remaining parameters. For the latter model, it should be able to identify outliers in data online. They are correlation analysis problem and detection problem of time sequence outliers, which means we need a model that can answer probabilistic queries about the variables and their relationships, and can make inference from uncertain information to help us retrospect the outliers [2]. Thus, we choose to build a Bayesian network model to solve the analysis problem and we use the autoregressive model and the posterior inspection to carry out the outlier detection. In this paper, we concentrate on the first problem and the solution to it.

A Bayesian network consists of two components: the structure, which is a Directed Acyclic Graph (DAG), for representing the conditional dependencies among variables, and a set of parameters for representing the quantitative information of the dependency [3]. Accordingly, learning a BN from data includes structure learning and parameter learning.

This paper focuses on the structure learning of Bayesian network. We will discuss our method of structure learning of Bayesian network in the part 3, our method of dealing with the data of our project (incremental learning) in part 4 in this paper, and experiments and testing results in part 5 in this paper.

2 Background

In the big data time, the problem of data sparsity still exist. It will be quite difficult to solve such problems with classical statistical methods. Under this circumstance, the Bayesian network provides us a wonderful answer to these complex problems.

A Bayesian network (BN) is composed of a directed acyclic graph (DAG) and a set of parameters. In a DAG G = (X, E), X is a collection of nodes or vertices, and E is a collection of edges. Those nodes of the DAG represent variables in the Bayesian sense: they might be discrete, continuous, known, or unknown [3]. In our project, we deal with the continuous parameters. Edges represent conditional dependencies. Each node is associated with a probability function that takes, as input, a particular set of values for the node's parent variables, and gives (as output) the probability (or probability distribution, if applicable) of the variable represented by the node.

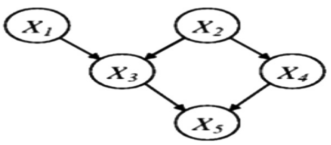

Fig. 1. Example of Bayesian network

As shown in Fig. 1 this is a DAG diagram with P nodes $\left[X_1, X_2, \ldots, X_p\right]$ and a directed edge to each other nodes. In DAG, each node represents a free variable, while the opposite side represents the influence of variables. The parent node is “cause” and the child node is effect. By establishing such a relationship, the independence assumption between variables is constructed, that is, a set of parent nodes of a given node, which is independent of all its non-descendant nodes. Therefore, the joint distribution probability of all nodes represented by Bayesian network can be expressed as the product of the conditional probability of each node, namely:

$$\mathrm{P}(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|X_1, X_2, \ldots, X_{i-1}) = \prod_{i=1}^{n} P(X_i|\pi(X_i))$$

In the graphical structure of Bayesian network, there are three possible types of adjacent triplets allowed in a directed acyclic graph (DAG):

$$\begin{gathered} \mathrm{X} \rightarrow \mathrm{Y} \rightarrow \mathrm{Z} \\ \mathrm{X} \leftarrow \mathrm{Y} \rightarrow \mathrm{Z} \\ \mathrm{X} \rightarrow \mathrm{Y} \leftarrow \mathrm{Z} \end{gathered}$$

Structural learning is the key point of the whole Bayesian network, which directly influences the subsequent parameter learning and inference. The scientific problem of structure learning of Bayesian network is: how to find a directed acyclic graph structure (DAG) that is corresponding to data distribution in a reasonable time. The challenge is that the structural space is an exponential function of variables. In the face of such a surpass-exponential structure space, it is not feasible to find the global optimal solution in a reasonable time and limited storage space.

The current structure learning method can be divided into two categories: discrete Bayesian network structure learning and continuous Bayesian network structure learning; by the adopted technical methods, it can be divided into: based on the constraints of the Bayesian network structure learning [4–6], based on the scoring-search of the Bayesian network structure learning [7–10], and the mixed method of structure learning; according to the data processing method can be divided into: batch learning and incremental learning [11–13]; by the accuracy of the solution can be divided into: accurate learning and approximate learning.

The comparisons between these approaches can be found in the Tables 1, 2 and 3 below.

Table 1. Advantages and disadvantages of three types of Bayesian network structure learning methods

Methods	Strength	Weakness
Constraint-based	High efficiency	Sensitive to individual detection
Scoring-search-based	Insensitive to individual errors	Large searching-space
Mixed	High efficiency	Inconsistency

Table 2. Advantages and disadvantages of batch learning and incremental learning

Methods	Strength	Weakness
Batch learning	Be able to find global optimal solution	High time cost
Incremental learning	Low time cost	Might end up in local solution

Table 3. Advantages and disadvantages of accurate learning and approximation learning

Methods	Strength	Weakness
Accurate learning	Be able to find global optimal solution	High time cost, high store demand
Approximate learning	Low time cost	Might end up in local solution

The application background is as follows:

(1) We deal with continuous, high-dimensional and large sample data;
(2) The degree of dependence between quantitative variables;
(3) The complexity of reasoning is the exponential of the largest group;

Considering the three reasons, we decided to learn by incremental Sparse Bayesian networks (SBN) to deal with these problems at the same time.

Firstly we introduce SBN [14], it is a kind of continuity for high-dimensional variable method for generating a Bayesian network, different from traditional discrete Bayesian networks, it finally be able to generate a directed acyclic graph, and the corresponding mixture Gaussian distribution. On the specific implementation, the core idea of the algorithm is to maintain a coefficient matrix B. The matrix saves the relation coefficients of each node (attribute characteristics) relative to the parent node, which is continuously updated in the iteration, and finally a relational coefficient matrix is generated, which is further transformed into a directed acyclic graph.

The advantage of using this method to study Bayesian network structure is:

(1) The current mainstream continuous Bayesian network structure learning method mainly includes the conditional Gaussian-Bayesian network and the linear Gaussian-Bayesian network. The conditional covariance matrix is invertible, which limits its application. Linear Gaussian-Bayesian network not only is not subject to this restriction but also combines the sparse technology represented by the current, such as lasso, to reduce the complexity of the model.
(2) It can theoretically guarantee the ring detection in the structure.
(3) Due to the particularity of SBN algorithm, the prior knowledge of domain experts can be easily added, and it is easy to extend to incremental learning.

3 Structure Learning

BN structure learning aims at a NP-hard problem that selecting a probabilistic model that explains a given set of data. The main task of Bayesian network structure learning is to construct a directed graph which conforms to the dependence of the characteristics through the data of the original sample. It is an important basis for subsequent parameter learning and Bayesian inference.

In our project, our team needs to learn large BN structures with high accuracy and efficiency from limited samples. Therefore, a useful strategy is to impose a sparse-constraint of some kind. Many real-world networks are indeed sparse, such as the gene association networks, and our relationship network of heating load factors. When learning the structure of these networks, a sparse constraint helps prevent overfitting and improves computational efficiency.

Meanwhile, considering the common Bayesian Network structure learning is dependent on the characteristics of discretization and continuous raw data, in order to avoid the unreliability caused by human for data discretization as well as the complexity of the reasoning, we adopt a Bayesian network for continuous data structure learning algorithm (Sparse Bayesian Network, SBN) [14]. The method (SBN) itself is aimed at constructing model which has small sample size of continuous data. However, given our sample size is larger, the implementation of SBN based on the further provision of batch training and incremental training, can effectively shorten the training time.

In the SBN algorithm, the following structures are mainly defined. First, the number of variables is p. We use a two dimensional array to represent the DAG we get (a p*p matrix G). $G_{ij} = 1$ represents that there is a directed edge from node i to j. In addition, a p-dimensional matrix is needed to indicate whether there is a path between nodes, namely the relationship closure.$P_{ij} = 1$ represents that there is a path from node i to j, and vice there is none. At last, we need a $(\mathrm{p} - 1) * \mathrm{p}$ matrix B to record all the coefficients of every node and its parent, since any one node is unlikely to become their own parent node (assuming that there is no the loop), matrix B has one less row compared to the above matrixes. SBN is essentially a model which learn a group of relation coefficients iteratively on each dimension to build a connection between nodes and their parents through data. There is no directed edge between two nodes when the relationship coefficient is zero, the core of algorithm is to optimize a set of formula below:

$$\hat{B} = \underset{i=1}{\overset{p}{min}} \left\{ \left(x_i - \beta_i^T x_{/i}\right)\left(x_i - \beta_i^T x_{/i}\right)^T / 2 + \lambda_1 ||\beta_i|| 1 \right\}$$
$$\text{s.t } \beta_{ji} \times P_{ij} = 0, i,j = 1,\ldots,p, i \neq j$$

Where β_i is the ith column of matrix B, and $x_{/i}$ represents the sample matrix after remove the variable i in matrix B. The optimization goal of the formula is to minimize the fitting error between real value and the coefficient, then plus the regular penalty term L_1. Moreover, λ_1 is used to control the number of non-zero in matrix B, namely controls the sparsity of network structure. The larger the λ_1, the smaller the number of non-zero in matrix B and the more sparse network structure. In the iterative process, it is necessary to maintain the $\beta_{ji} \times P_{ij}$ constant to zero, so as to guarantee the network structure is not circular.

Unfortunately, the formula above is hard to solve. Therefore, in practical development, we change the optimization to matrix B to aiming at each column. That is, the cumulative process of each variable optimization. The transform formula is:

$$\begin{aligned}\widehat{B_{ap}} &= \min_B \sum_{i=1}^{p} f_i(\beta_i) \\ &= \min_B \sum_{i=1}^{p} \left\{ \left(x_i - \beta_i^T x_{/i}\right)\left(x_i - \beta_i^T x_{/i}\right)^T/2 + \lambda_1 ||\beta_i||1 + \lambda_2 \sum_{j \subset X_i} \left|\beta_{ji} \times P_{ij}\right| \right\}\end{aligned}$$

In the formula, $j \subset X_i$ represents the sample without variable i. We add two penalty terms into this formula, Where λ_1 is still the L_1 regularization penalty term, and it is used to control the sparsity of network structure. Then, λ_2 make $\left|\beta_{ji} \times P_{ij}\right|$ close to zero, so as to avoid having a loop in the graph, and by proving that, when λ_2 meets the following condition:

$$\lambda_2 > \frac{(n-1)^2 p}{\lambda_1} - \lambda_1$$

It ensures that no ring is formed during training.

After given the value of λ_1,λ_2, we can calculate it by the BCD algorithm. BCD algorithm is a method of block optimization. For matrix B, BCD algorithm holds all the remaining columns, updating β_i in turn, which is equivalent to optimizing $f_i(\beta_i)$ until the preset convergence conditions are satisfied. Specifically, in our scheme, we adopted the L_2 paradigm change less than 0.001 as the convergence condition. For each optimization of $f_i(\beta_i)$, it's feasible to use a form similar to LASSO optimization:

$$f_i(\beta_i) = \frac{\left(x_i - \beta_i^T x/i\right)\left(x_i - \beta_i^T x/i\right)^T}{2} + \sum_{j \subset X_i} \left(\lambda_1 + \lambda_2 \left|\beta_{ji} \times P_{ij}\right|\right)\left|\beta_{ji}\right|$$

In the above formula, x_i is the sample vector with the characteristic of i. β_i^T is the column that corresponds to feature i in the relational matrix,P_{ij} is the connectivity of feature i and j at this stage. In particular, the judgment of connectivity can be resolved using BFS (depth-first search). For the optimization of $f_i(\beta_i)$, the shooting algorithm can be used to iterate. Finally, we calculate the following formula through constant calculation:

$$\beta_{ji}^{t+1} = \left(\left| \frac{\left(x_i - \beta_{i/j}^{t}{}^{T} x_{/(i,j)}\right) x_j^T}{x_i x_i^T} \right| - \frac{\lambda_1 + \lambda_2 \left|P_{ij}\right|}{x_i x_i^T} \right) + \mathrm{sign}\left(\frac{x_i - \beta_{i/j}^{t}{}^{T} x_j^T}{x_i x_i^T} \right)$$

Then the convergence conditions are going to be approximated. Finally, considering the implementation details of the algorithm, we need to normalize the original samples (keep the mean value 0 and the variance 1).

4 Bayesian Incremental Learning

The aim of incremental learning is for the learning model to adapt to new data without forgetting its existing knowledge, it does not retrain the model. Incremental algorithms are less restrictive than online algorithms, and incremental algorithms process input examples one by one (or batch by batch) and update the decision model after receiving each example. Incremental algorithms may have random access to previous examples or representative/selected examples. In such a case, these algorithms are called incremental algorithms with partial memory.

Typically, in incremental algorithms, for any new presentation of data, the update operation of the model is based on the previous one. Streaming algorithms are online algorithms for processing high-speed continuous flows of data. In streaming, instances are processed sequentially as well and can be examined in only a few passes (typically just one). These algorithms use limited memory and limited processing time per item.

Considering the large sample size (training, using data a year sample size of 500000 or so), although the overall training is able to achieve a more accurate solution, the training process is time-consuming. Based on two characteristics of overall training, we made a few improvements and changes.

First of all, we have to deal with a large number of data samples. The cost of all training to achieve a global solution is to sacrifice the training time, so we take the sliding window as the core. Because is streaming data at the same time, in practical applications, the sample data may not all have, many times the relationship between the variables can cause some changes with time and, this is called online learning based on streaming data. Considering that boiler data satisfies the definition of streaming data, an incremental updating interface is provided based on SBN. Learning through some offline data first, obtain a fairly network structure, and also using the ideas of sliding window, with the passage of time constant sliding window, the data change, once every sliding window, try to calculate the coefficient of a matrix. On the basis of the changes to set a threshold value, and the relationship between the threshold, when the relationship matrix in the numerical change more than threshold, modified, this is to prevent the noise to interfere with the accuracy of the model, and use of the threshold value to determine the relationship between directed edge, when the relation matrix of value across the threshold, the relationship between the network structure of the directed graph updates, because every incoming sample occupies smaller share in the window, it can ensure no dramatic changes, the relationship between matrix when a concept drift, window after drift value will be more and more, makes the relationship matrix gradually closer to the new concept, Thus, an incremental learning method is realized.

In the actual implementation, we take the original sample and store it in a matrix, each row representing a set of boiler data. The number of rows in this matrix is advance given. First, we accept a certain amount of data in the matrix, and then we learn it, and we get the raw matrix B. When the data that we accept exceeds the number of rows in the matrix, we delete the first row of the matrix and add a new set of data to the last row of the matrix, which is the idea of a sliding window. In the process of checking, because our Bayesian network structure is done with linear Gaussian fitting every time, the structure and solution we get corresponding to the next new window are close to the convergence. Furthermore, because the sample size of a single batch is much smaller than the total sample, the relational matrix can converge to a local solution in a short time. With the arrival of the subsequent batch, the relationship matrix is constantly updated. After update, we check the new relationship matrix, and we update the edges according to the value of threshold, to control the sparsity of the structure (Tables 4 and 5).

Table 4. Shows a more detailed description of proposed algorithm.

```
Input: sample matrix ,SM; number of variable, p; regularization
parameters , {$\lambda_i$}i=1,2;initial ,B0 ;stopping criterion , $\varepsilon$ .
-----------------------------------------------------------------
Initialize:
      Let converge=false;
      Let  t=0;
Repeat
   For i=1,2,…,p
      A Breadth-first search on G with Xi being the
       Root node to calculate $P_{ij}$ for
      j=1,…p.
       Using the shooting algorithm[1] to Optimize $f_i(\beta_i)$ and get $\beta^{t+1}_i$
   End for
   If $||B^{t+1}-B^t|| \le \varepsilon$ and no new sample update SM
         Converge=true;
   Else
      Converge=false;
       Update SM;
  End if
  Let t=t+1;
Until converge =true
Output:$B^{t+1}$
```

Table 5. Shows sliding window update algorithm.

```
SM update algorithm:
-----------------------------------------------------------------
Input: new data, SM
   Delete first row sample in  sample matrix SM;
   Remainder m-1 row move forward first row;
   New a set of data fill last row of SM;
```

For the SBN algorithm, the time complexity is np^2. n is the iteration number of BCD and p is the number of variables. In most cases, the number of variables is small, no more than 100, and the iteration number can be really big when the data size is huge. For example, for our boiler data from our client, it is usually a hundred thousand orders of magnitude. In this case, the n is too big and the time complexity is high. However, our incremental approach can converge within 10 iterations. In our implementation of the window slide in the project, we set a window in 3000 sets of data, and the time complexity of this method is cp^2. In this case, c3000*10 which is much less than a hundred thousand.

In addition, the experiment found that if the overall distribution of a one-dimensional data is basically unchanged, the first batch of samples can converge the matrix which corresponds to the dimension to an approximate global solution, and subsequent iterations will be greatly reduced.

Whether the data is large or small, the approach of incremental sparse Bayesian network structure learning theoretically takes much less time than other Bayesian network learning method.

5 Experiments and Testing Results

Based on the methods above, we conducted test based on the provided data, because an important idea of BCD + shooting method is to use a specific column, fixed other data to do block optimization. So the order of the original data characteristics will produce certain effect to the learned structure. According to the proof of theory and experiment, the higher the variable sequence, the more the tendency to become the ancestor node. Therefore, it is an important part to find a more reasonable feature order for this method. According to the prior knowledge, choosing a good order of characteristic variables can effectively improve the validity of the network. We conducted experiments based on two variables and different sample sizes.

5.1 Time Efficiency Experiments

We firstly conducted two time efficiency experiments on two data sets which belong to two kinds of variable order. One data set is large and another data set is small. Since we need to deal with data with continuous parameters, we choose batch learning approach to compare with our incremental approach, they both can deal with continuous parameters by using SBN algorithm.

oder1 = [0, 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, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]

The size of this group of data is 520,000, with 57dimensions (57 variables) (Figs. 2 and 3).

Time efficiency:
Incremental approach : 14h Batch approach: 29h
Analysis:

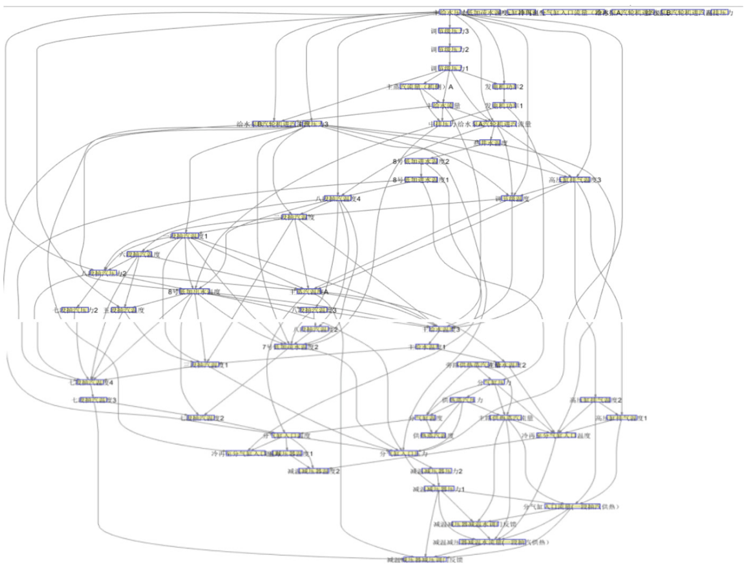
Fig. 2. Structure constructed by incremental learning for order 1 (sample size is 520,000).

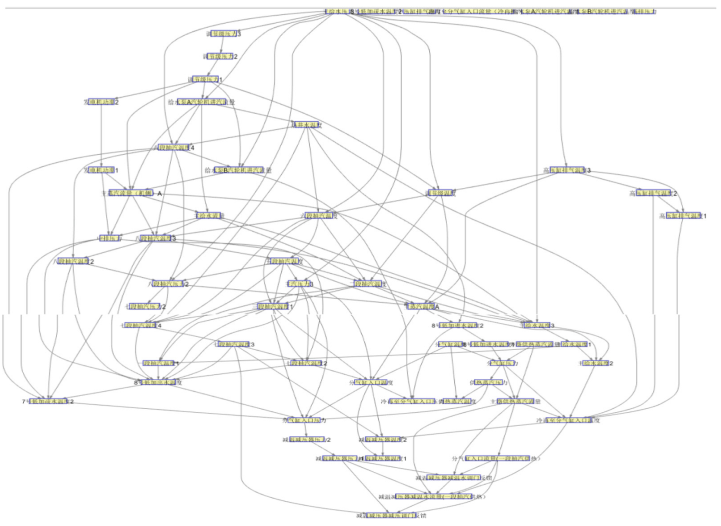
Fig. 3. Structure constructed by batch approach for order 1 (sample size is 520,000)

This group of data has 57 sets of variables, of which 37 groups converge before the 5th iteration, and 14 groups converge in the sixth iteration, and the remaining six groups converge in the eighth iteration. The time complexity is much lower than the batch.

It can be seen from the above time that the incremental learning method adopted by us is obviously better than the batch mode in time performance, and the convergence to the final solution is much faster. At the same time, the structure generated by the two methods is different, which is less accurate than the batch type, but it is still accurate. From the comparison experiment on order1, we can know that the incremental learning method we have improved is very effective and advanced for the situation size of 500,000+.

oder2 = [0, 1, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]

The size of this group of data is 3, 000, with 30 dimensions (30 variables).

Therefore, the running time is very fast. In the case of incremental situation, 25 dimensions converge in the first five rounds of iteration, and four dimensions converge in the sixth round, and the last one will converge in the seventh round. Because the sample size is small, there is no great advantage for incremental approach in time complexity (Figs. 4 and 5).

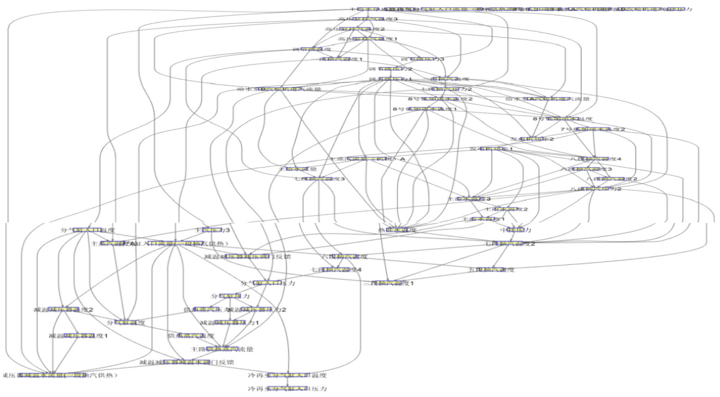

Fig. 4. Structure constructed by incremental learning for order 2 (sample size is 3,000)

Time efficiency:
Incremental approach : 0.4h Batch approach: 0.4h
Analysis:

Through the above time comparison, we can see that the incremental learning method adopted by us is not significantly different from the batch type in terms of time performance, and the obtained structure is basically the same. This shows that for smaller sample sizes, incremental learning and batch can be equally accurate and fast.

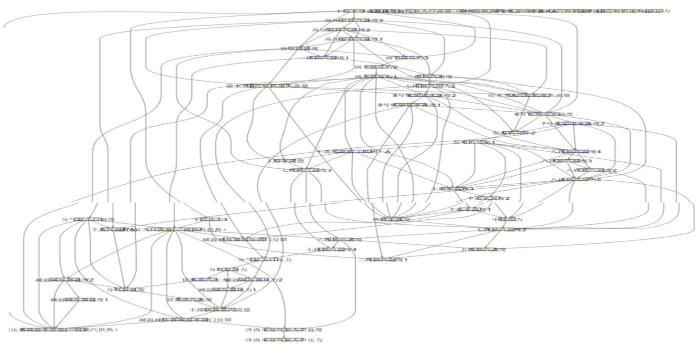

Fig. 5. Structure constructed by batch approach for order 2 (sample size is 3,000)

5.2 Accuracy Experiment of Structure Learning Results

In the first experiment, we managed to prove that the incremental approach for sparse Bayesian network structure learning is more efficient in time than batch learning. In this experiment, we try to conduct a comparison experiment to find the most suitable penalty value and the relation threshold value that can make our method most accurate. At the same time, prove that our method can generate an accurate structure through structure learning.

Before the test, we should explain the two parameters in our system: penalty and threshold. The penalty controls the sparsity and avoid becoming a circle through training (It is explained in detail in part IV. structure learning of this paper), and the relation threshold removed some edge with weak correlation, namely after-pruning (It is explained in detail in part V. Incremental learning of this paper).

It's worth mentioning that the effect of redundant edges on structural correctness is less than absent edges, which means we should pay more attention on decrease the number on absent edges. Moreover, the bigger the threshold is, the more sparse the structure is, and normally the bigger the penalty is, the more sparse the structure is.

Dataset: CHILD

Number of nodes: 20; Number of arcs: 25; Number of parameters: 230; Average Markov blanket size: 3.00; Average degree: 1.25; Maximum in-degree: 2.

The influence of the value of these two parameters can be found in Table 6, we can find a best setting for them when the redundant edges (RE) and absent edges (AE) are least.

As we can observe in chart 4, when the penalty value is 0.8 and the relation threshold is 0.10, the absent edges and redundant edges are the least. It is mathematical that there must be a most suitable setting that the structure is closest to the ground truth, since the two parameters can both regulate the sparsity of the structure.

Apparently, when the threshold or the penalty is too small, the structure can be so tense, in that case, there will be too much redundant edges. On the other hand, when the threshold or the penalty is too big, the structure can be so sparse, which leads to a result that it is more likely that more edges will be missed. Therefore, the balance spot is when the penalty value is 0.8 and the relation threshold is 0.10.

Table 6. The redundant edges and absent edges with different parameters for CHILD

Threshold value	Penalty value				
	0.6	0.7	0.8	0.9	1.0
0.02	19 (RE) 10 (AE)	19 9	18 9	17 9	17 8
0.04	18 9	17 9	17 8	16 8	15 8
0.07	16 9	15 7	14 6	14 8	13 8
0.09	13 7	13 6	12 5	11 7	11 8
0.10	10 5	8 5	8 (right 4setting)	7 6	6 7
0.11	10 6	8 6	8 5	9 7	8 8

Dataset: ALARM

Number of nodes: 37; Number of arcs: 46; Number of parameters: 509; Average Markov blanket size: 3.51; Average degree: 2.49; Maximum in-degree: 4.

Table 7. The redundant edges and absent edges with different parameters for ALARM

Threshold value	Penalty value				
	0.6	0.7	0.8	0.9	1.0
0.02	23 (RE) 13 (AE)	23 13	22 13	22 13	20 13
0.04	23 11	22 11	22 12	21 12	19 13
0.07	22 10	22 11	21 11	20 12	18 13
0.09	20 10	19 10	17 11	16 13	16 12
0.10	19 8	17 8	15 (right 8setting)	14 9	13 11
0.11	17 8	15 (right 8setting)	14 10	14 9	13 10

As we can observe in the Table 7, there are two situations which have the least redundant edges and absent edges. First one is when the penalty value is 0.7 and the relation threshold is 0.10, and the second one is when the penalty value is 0.8 and the relation threshold is 0.09. For a dataset which has 44 edges, a structure with 8 absent edges is considerably accurate in machine learning field.

Compare this setting of the dataset ALARM and the last setting of the dataset CHILD, we conclude that when the penalty value closes to 0.8 and the relation threshold value closes to 0.10, the structure our system learned is the most suitable one.

On the other hand, we need to evaluate the performance between the offline learning and online learning based on sliding windows which we set as 10,000. Due to the size of our data is large, online learning is more efficient than offline method. In order to prevent the influence of individual data on parameter updating, we slide the windows once when every 100 new data coming. What's more, we set penalty based on the size of sliding windows. Due to the lack of correct network based on our data, we assume the offline result as the groundtruth. In order to reduce the influence of data size, we set the size of data as 50,0000. The result is showing in Tables 8, 9 and 10.

Table 8. The comparison of offline learning and online learning Bayesian networks for 30 nodes

	Nodes	Edges	RE	AE
Offline	30	56		
Online_1	30	63	10	3
Online_2	30	60	6	2
Online_3	30	58	3	1

Table 9. The comparison of offline learning and online learning Bayesian networks for 50 nodes

	Nodes	Edges	RE	AE
Offline	50	74		
Online_1	50	83	16	7
Online_2	50	78	8	4
Online_3	50	77	7	4

Table 10. The comparison of offline learning and online learning Bayesian networks for 62 nodes

	Nodes	Edges	RE	AE
Offline	62	89		
Online_1	62	99	19	9
Online_2	62	93	10	6
Online_3	62	93	9	5

Tables 8, 9 and 10 show the comparison of offline learning and online learning Bayesian Networks. First line is the offline groudtruth, second line is the network based on first batch, The third line is the networks based on the half of data, The last line is the networks based on the total data.

From the result of table above, we can find the result of online learning is close to offline learning, but the edges of online learning is more than offline learning, it may because the penalty we set is not reasonable enough. The first batch result has a gap with groundtruth, but with the online updating of the network, the result is slowly approaching the groundtruth. Base on the experiment, we can find the online learning method we proposed is more efficient than offline learning which could maintain accuracy.

6 Conclusion of Experiments

In conclusion, the time efficiency experiments prove that our incremental approach for sparse Bayesian network structure learning more efficient than the existing approach of Bayesian structure learning. The accuracy experiments prove that our method is considerably accurate among machine learning methods.

The results demonstrate the accuracy, efficiency of our approach, and our approach can solve the current intractability that learns structure with continuous parameters. All the advantages above reflect the advancement of our approach.

Last but not the least, those experiments we conduct prove that our incremental approach of Bayesian network structure learning can be wildly applied in big data filed. For example, many situations in big data like the analysis of the data of complex equipment, the origin of galaxies, the pathogenic genes, the operation mechanism of the large heating units, etc. To uncover the laws underlying these problems, we must understand their genetic networks and tease out the intricacies of the events. Due to the invalidation of classical statistics in those problems, our approach appears to be more valuable.

References

1. Borsuk, M.E., Stow, C.A., Reckhow, K.H.: A Bayesian network of eutrophication models for synthesis, prediction, and uncertainty analysis. Ecol. Model. **173**, 219–239 (2004)
2. Agosta, J.M., Gardos, T.R., Druzdzel, M.J.: Query-based diagnostics. In: Jaeger, M., Nielsen, T.D. (eds.) Proceedings of the Fourth European Workshop on Probabilistic Graphical Models, PGM-08, Aalborg, Denmark, pp. 1–8 (2008)
3. Cooper, G., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Mach. Learn. **9**(4), 309–347 (1992)
4. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction and Search. Springer, New York (1993). https://doi.org/10.1007/978-1-4612-2748-9
5. Cheng, J., Greiner, R., Kelly, J., et al.: Learning Bayesian networks from data: an information-theory based approach. Artif. Intell. **137**(1–2), 325–331 (1997)
6. Zhang, H., Zhou, S., Zhang, K., et al.: Causal discovery using regression-based conditional independence tests. In: AAAI, pp. 1250–1256 (2017)
7. Alcobé, J.R.: Incremental hill-climbing search applied to Bayesian network structure learning, pp. 1320–1324 (2008)
8. Ko, S., Kim, D.W.: An efficient node ordering method using the conditional frequency for the K2 algorithm. Pattern Recognit. Lett. **40**, 80–87 (2014)

9. Alonso-Barba, J.I., de la Ossa, L., Regnier-Coudert, O., et al.: Ant colony and surrogate tree-structured models for orderings-based Bayesian network learning. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 543–550. ACM (2015)
10. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: the combination of knowledge and statistical data. Mach. Learn. **20**(3), 197–243 (1995)
11. Dimkovski, M., An, A.: A Bayesian model for canonical circuits in the neocortex for parallelized and incremental learning of symbol representations. Neurocomputing **149**(4), 1270–1279 (2015)
12. Yue, K., Fang, Q., Wang, X., et al.: A parallel and incremental approach for data-intensive learning of Bayesian networks. IEEE Trans. Cybern. **45**(12), 2890–2904 (2017)
13. Yasin, A., Leray, P.: Incremental Bayesian network structure learning in high dimensional domains. In: International Conference on Modeling, Simulation and Applied Optimization, pp. 1–6. IEEE (2013)
14. Huang, S., et al.: A sparse structure learning algorithm for Gaussian Bayesian network identification from high-dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **35**(6), 1328–1341 (2013)