Consistent Learning Bayesian Networks with Thousands of Variables

Kazuki Natori NATORI@AI.LAB.UEC.AC.JP

Masaki Uto UTO@AI.LAB.UEC.AC.JP

Maomi Ueno UENO@AI.LAB.UEC.AC.JP
The University of Electro-Communications
Tokyo (Japan)

Abstract

We have already proposed a constraint-based learning Bayesian network method using Bayes factor. Since a conditional independence test using Bayes factor has consistency, the learning method improves the learning accuracy of the traditional constraint-based learning methods. Additionally, the method is expected to learn larger network structures than the traditional methods do because it greatly improves computational efficiency. However, its expected benefits have not been demonstrated empirically. This report describes some experiments related to the learning of large network structures. Results show that the proposed method can learn surprisingly huge networks with thousands of variables.

Keywords: Bayesian networks, Conditional independence test, Learning Bayesian networks, Bayes factor

1. Introduction

Bayesian networks are graphical models that encode probabilistic relations among variables using a directed acyclic graph (DAG). Given the DAG structure, a joint probability distribution can be decomposed exactly into a product of the conditional probability of variables given their parent variables. Therefore, the Bayesian networks are expected to provide a good approximation of the joint probability distribution.

Because the Bayesian network structure is generally unknown, it is necessary to estimate the structure of Bayesian network from observed data in a process called "learning Bayesian networks". The most common learning approach is a "score-based approach", in which "exact learning" is often used. It is intended to seek the best structure with a score function that has asymptotic consistency. The most popular Bayesian network learning score is the marginal likelihood, which finds the maximum a posteriori (MAP) structure, as described by Buntine (1991). However, this approach has an associated NP-hard problem (Chickering, 1996): as the number of variables increases, the number of structure searches increases exponentially. In this approach, various algorithms have been developed, such as dynamic programming (Silander and Myllymaki, 2006), A^* search (Yuan et al., 2011), breadth-first branch and bound search (Malone et al., 2011), and integer programming (Cussens, 2011). Nevertheless, it cannot be applied to network structures with more than 60 variables. In

contrast, the state-of-the-art algorithm of the exact inference with Bayesian networks can make inferences using the structure with more than 200 variables (Li and Ueno, 2017). Therefore, a new learning Bayesian algorithm is required for huge networks.

Constraint-based approaches relax computational costs and learn huge networks. Such approaches learn by conditional independence (CI) tests and the direction using the orientation rules. Among these approaches, Peter and Clark (PC) algorithm (Spirtes et al., 2000), max-min hill climb (MMHC) algorithm (Tsamardinos et al., 2006), and recursive autonomy identification (RAI) algorithm (Yehezkel and Lerner, 2009) are well known. The RAI algorithm is the state-of-the-art algorithm adopting this approach. However, this approach depends on CI checks between each pair of variables using statistical tests or information theory tests. The statistical test necessarily has type I error (detecting the dependency when the true structure is independent) even for large data. The information theory test also depends on the user-determined threshold. Therefore, earlier methods using this approach have no asymptotic consistency.

Abellán et al. (2006) proposed the learning method by application of the CI test with the BDeu score to the PC algorithm. However, the BDeu score was proposed for the score-based approach. For that reason, results obtained using this method were somehow unstable.

For this reason, Natori et al. (2015) proposed a consistent CI test using Bayes factor for constraint-based learning. Concretely, the study derived the marginal likelihood for the constraint-based approach. In the proposed CI test, they used a Jeffreys' prior as the hyperparameter. It is known that the Jeffreys' prior is the minimum risk of the Dirichlet prior. It maximizes the entropy of the prior (Clarke and Barron, 1994). The study demonstrated the effectiveness of the CI test using the Jeffreys' prior. In Natori et al. (2015), the proposed CI test was applied to the RAI algorithm.

However, Natori et al. (2015) did not empirically demonstrate the expected advantage of the proposed method. Furthermore, they did not compare the learning performance with those of other constraint-based methods. Moreover, the study examined no experiment using large network structures.

This paper presents some experiments designed to learn large network structures. Results show that the proposed method provides faster calculation and more stable results than those obtained using the PC algorithm or the MMHC algorithm using the same CI tests. In addition, results show that the proposed method can learn surprisingly huge networks with 1,041 variables. Although Scanagatta et al. (2015) proposed the method that can learn 10,000 variables, the method is the approximated algorithm.

This paper is organized as follows. Section 2 presents fundamental knowledge of learning Bayesian networks. Section 3 explains CI tests for constraint-based approaches and some attendant difficulties. Section 4 introduces the Bayes factor for the constraint-based approach. Section 5 reports the results of numerical experiments. Section 6 concludes this paper and presents some avenues of future work.

2. Learning Bayesian networks

Let $\{x_1, x_2, \cdots, x_N\}$ be a set of N discrete variables. Each can take values in the set of states $\{1, \cdots, r_i\}$. We write $x_i = k$ when we observe that an x_i is state k. The joint

probability distribution of a Bayesian network is given as

$$p(x_1, x_2, \cdots, x_N \mid g) = \prod_{i=1}^{N} p(x_i \mid \Pi_i, g), \tag{1}$$

where $g \in G$ is the possible set of Bayesian network structures, and where Π_i is the parent variable set of x_i.

Let θ_{ijk} be a conditional probability parameter of $x_i = k$ when the j-th instance of the parents of x_i is observed (we write $\Pi_i = j$). In an earlier study Buntine (1991), the Dirichlet prior was assumed and an expected a posteriori (EAP) estimator was used as the parameter estimator $\hat{\theta}_{ijk}$ $(i = 1, \cdots, N, j = 1, \cdots, q_i, k = 1, \cdots, r_i - 1)$:

$$\hat{\theta}_{ijk} = \frac{\alpha_{ijk} + n_{ijk}}{\alpha_{ij} + n_{ij}}, \quad (k = 1, \cdots, r_i - 1). \tag{2}$$

Therein, n_{ijk} represents the number of samples of $x_i = k$ when $\Pi_i = j$, $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$. Also, α_{ijk} denotes the hyperparameters of the Dirichlet prior distributions (α_{ijk} is a pseudo-sample corresponding to n_{ijk}), $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$, and $\hat{\theta}_{ijr_i} = 1 - \sum_{k=1}^{r_i - 1} \hat{\theta}_{ijk}$.

The marginal likelihood is obtained as

$$p(\mathbf{D} \mid g, \alpha) = \prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})}. \tag{3}$$

Here, q_i signifies the number of instances of Π_i, where $q_i = \prod_{x_l \in \Pi_i} r_l$ and $\mathbf{D}$ is a dataset. The learning Bayesian networks is to find the maximum marginal likelihood structure that maximizes score (3).

Particularly, Heckerman et al. (1995) presented a sufficient condition for satisfying the likelihood equivalence assumption in the form of the following constraint related to hyperparameters of (3):

$$\alpha_{ijk} = \alpha p(x_i = k, \Pi_i = j \mid g^h). \tag{4}$$

Here, α is the user-determined equivalent sample size (ESS). Additionally, g^h is the hypothetical Bayesian network structure that reflects a user's prior knowledge. This metric was designated as the BDe score metric. As Buntine (1991) described, $\alpha_{ijk} = \alpha/(r_i q_i)$ is regarded as a special case of the BDe metric. Heckerman et al. (1995) called this special case "BDeu." Actually, $\alpha_{ijk} = \alpha/(r_i q_i)$ does not mean a uniform prior. It is the same value of all hyperparameters for a variable. Silander et al. (2007) have shown the behavior of BDeu is very sensitive to the ESS value. Ueno (2008) showed that the learning accuracy is highly sensitive to the ESS value. Suzuki (2016) studied the asymptotic properties of BDeu using BD with Jeffrey's prior.

A popular structure learning approach is the score-based approach, which seeks the best structure with these score functions with consistency. However, the score-based approach entails heavy computational costs. Generally, the constraint-based approach is known to have lower computational costs. Originally, the constraint-based approach sequentially checks CI relations among all variables. Then it directs edges of the structure from observed data. The methods of the constraint-based approach are generally asymptotically correct when the CI tests are exact (Cheng et al., 1997; Spirtes et al., 2000).

3. CI tests

Constraint-based learning depends on CI tests. Typically, CI tests use χ^2 test, G^2 test or conditional mutual information. In this section, we explain those CI tests.

3.1 CI test using χ^2 and G^2 test

Statistical tests compare the null hypothesis that two variables are independent with the alternative hypothesis. If the null is rejected (cannot be rejected), then the edge is added (removed). A statistic that is asymptotically χ^2 distributed is calculated and compared to a critical value. If it is greater (smaller) than the critical value, then the null is rejected (cannot be rejected) (Agresti and Kateri, 2011; Spirtes et al., 2000). In χ^2 test between X and Y given a set of conditional variables $\mathbf{Z}$ for Bayesian networks, the statistic χ^2_{xyz} is defined as

$$\chi^2_{xyz} = \sum_{x \in X, y \in Y, z \in \mathbf{Z}} \frac{(N_{xyz} - N_{xz}N_{yz}/N)^2}{N_{xz}N_{yz}/N}, \tag{5}$$

where N is the total number of samples. N_{xyz} denotes the number of samples when $X = x$, $Y = y$, and $\mathbf{Z} = z$, where x and y signifies the states of X and Y respectively, and z is z-th combination of states for all variables in $\mathbf{Z}$. N_{xy} and N_{xz} are defined similarly.

In addition, based on maximum likelihood, the statistic G^2 is defined as

$$G^2 = 2 \sum_{x \in X, y \in Y, z \in \mathbf{Z}} N_{xyz} \log\left(\frac{N_{xyz}}{N_{xz}N_{yz}/N}\right), \tag{6}$$

then χ^2 and G^2 tests follow the χ^2 distribution asymptotically with the degree of freedom df. The degree of freedom df is obtained as

$$df = (\|X\| - 1)(\|Y\| - 1) \prod_{Z \in \|\mathbf{Z}\|} \|Z\|, \tag{7}$$

where $\|X\|$ and $\|Y\|$ are the numbers of states of each variable. $\|Z\|$ represents the number of states of variable Z in $\mathbf{Z}$.

The CI tests depends on a p-value and a significance-level α. The p-value is derived from each statistic and df. In those CI tests, the null hypothesis is rejected when p-value is greater than α. Furthermore, α is defined as the occurrence ratio of type I error. Generally, $\alpha = 0.05$ is often used for the CI tests in constraint-based learning Bayesian networks.

3.2 CI test using conditional mutual information

Conditional mutual information between variable X and Y given $\mathbf{Z}$ provides a measure of the conditional mutual independence between X and Y given $\mathbf{Z}$, defined as

$$\mathrm{CMI}(X; Y \mid \mathbf{Z}) = \sum_{x \in X, y \in Y, z \in \mathbf{Z}} P(x, y, z) \log \frac{P(x, y \mid z)}{P(x \mid z)P(y \mid z)}. \tag{8}$$

Here, $P(x, y, z)$ stands for the joint probability distribution between X, Y, and $\mathbf{Z}$ when $X = x$, $Y = y$, and $\mathbf{Z} = z$. $p(x \mid z)$ signifies the conditional probability when $X = x$ given

$\mathbf{Z} = z$. $p(y \mid z)$ denotes the conditional probability when $Y = y$ given $\mathbf{Z} = z$. The CI test judges conditional independence between X and Y given $\mathbf{Z}$ when $\mathrm{CMI}(X; Y \mid \mathbf{Z})$ is smaller than the threshold value.

However, these CI tests have serious problems. First, χ^2 and G^2 tests have the problem that the type I error converges to the user-determined significance level, as the sample size increases. Second, the CI test using the conditional mutual information depends on the value of threshold. Finally, these CI tests do not perform again, if the CI tests mistake as independence in low order (the small number of conditional variables) when true CI tests are dependence in high order. Therefore, these CI tests have no asymptotic consistency. For that reason, there is no guarantee of obtaining the true Bayesian network structure.

4. Bayes factor for constraint-based learning Bayesian networks

In this section, we introduce the CI test using a Bayes factor. Natori et al. (2015) proposed a CI test using a Bayes factor. The Bayes factor is the ratio of the marginal likelihood (Kass and Raftery, 1995), which has an asymptotic consistency. For example, the Bayes factor is given as $p(\mathbf{D} \mid g_1)/p(\mathbf{D} \mid g_2)$, where g_1 and g_2 are hypothetical structures from observed data $\mathbf{D}$. If the value is greater than 1.0, then g_1 is favored more than g_2; otherwise, g_2 is favored more than g_1.

Steck and Jaakkola (2002) proposed a CI test using the Bayes factor. In this method, $\mathbf{D}$ presents observed data for only two variables X and Y given conditional variables as $\log(p(\mathbf{D} \mid g_1)/p(\mathbf{D} \mid g_2))$.

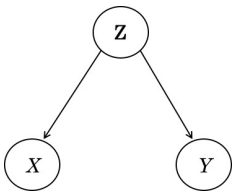

Figure 1: g_1: independent model

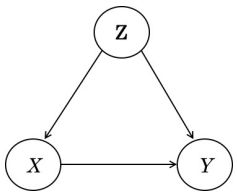

Figure 2: g_2: dependent model

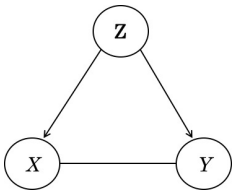

Figure 3: g_3: Natori et al. (2015) dependent model

In the CI test, g_1 shows an independent model in Figure 1. g_2 shows a dependent model in Figure 2, where $\mathbf{Z}$ is a set of conditional variables. When the log-Bayes factor takes a positive value, then the edge between X and Y is deleted. When the true structure satisfies the following assumption, the constraint-based learning using the CI test can obtain the true structure.

Assumption *Let $g = (\mathbf{V}, \mathbf{E})$ be the true Bayesian network structure, where $\mathbf{V}$ is a set of nodes, $\mathbf{E}$ is a set of edges.*

$$\exists X \in \mathbf{V},\ \exists Y \in \mathbf{V},\ \exists \mathbf{Z} \subset \mathbf{V} \setminus \{X, Y\},$$
$$s.t.\ Ind(X, Y \mid \mathbf{Z}) \rightarrow \forall \mathbf{Q} \subset \mathbf{V} \setminus \{X, Y, \mathbf{Z}\},\ Ind(X, Y \mid \mathbf{Z}, \mathbf{Q}),$$

where $Ind(X, Y \mid \mathbf{Z})$ signifies that X and Y are conditionally independent given a set of conditional variables $\mathbf{Z}$. In the worse case like every $N-1$ subset has no correlation among each other while all N variables are strongly correlated, the constraint-based learning is not

more effective than the score-based learning. This method applies BDeu as the marginal likelihood of the Bayes factor. In a report by Abellán et al. (2006), a method of applying the CI test in the PC algorithm is proposed. However, BDeu is not guaranteed to optimize results of CI tests because it was developed for the score-based approach. The CI test does not address the orientation of the edge between two variables.

To resolve this problem, Natori et al. (2015) defined the dependent model g_3 in Figure 3. In Natori et al. (2015), the marginal likelihood of g_3 is obtained as

$$p(\mathbf{D} \mid g_3, \alpha) = \prod_{j=1}^{q} \frac{\Gamma(r_1 r_2 \alpha_{g_3})}{\Gamma(r_1 r_2 \alpha_{g_3} + n_j)} \prod_{k_1=1}^{r_1} \prod_{k_2=1}^{r_2} \frac{\Gamma(\alpha_{g_3} + n_{jk_1k_2})}{\Gamma(\alpha_{g_3})}. \tag{9}$$

In those equations, $n_{jk_1k_2}$ denotes the number of samples of $x_1 = k_1$ and $x_2 = k_2$ when the j-th instance of the common parents of x_1 and x_2 is observed (we write $\Pi_{(x_1,x_2)} = j$). Also, q represents the number of instances of $\Pi_{(x_1,x_2)}$, and $n_j = \sum_{k_1=1}^{r_1} \sum_{k_2=1}^{r_2} n_{jk_1k_2}$. In addition, α_{g_3} is a hyperparameter.

Furthermore, Natori et al. (2015) defined the marginal likelihood of g_1 as shown below.

$$p(\mathbf{D} \mid g_1, \alpha) = \prod_{i=1}^{2} \prod_{j=1}^{q_i} \frac{\Gamma(r_i \alpha_{g_i})}{\Gamma(r_i \alpha_{g_i} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{g_i} + n_{ijk})}{\Gamma(\alpha_{g_i})}. \tag{10}$$

Here, α_{g_i} is a hyperparameter.

The remaining problem is determination of the values of hyperparameters α_{g_3} and α_{g_i}. Some options exist for selecting α_{g_3} and α_{g_i}. For example, 1 (uniform prior) or 1/2 (Jeffreys' prior) are well known.

It is known that the Jeffreys' prior is the minimum risk of the Dirichlet prior. It maximizes the entropy of the prior (Clarke and Barron, 1994). Natori et al. (2015) demonstrated the effectiveness of the CI test using the Jeffreys' prior.

Natori et al. (2015) applied this CI test to the recursive autonomy identification (RAI) algorithm (Yehezkel and Lerner, 2009). The salient benefit of the RAI algorithm is that it minimizes the number of parent nodes using CI tests in the constraint-based approach because it decomposes into autonomous sub-structures the orientation operations based on observed *V-structures*. In this method, starting from a complete undirected structure, the RAI algorithm finds the correct structure by performing the following procedure recursively for each sub-structure. The RAI algorithm detail is presented in Algorithm 1.

From Algorithm 1, the RAI algorithm comprises the following four steps. In step 1 (lines 11–15 and lines 17–21), all relations between nodes in the structure are checked using CI tests. In step 2 (lines 16 and 22), the remaining edges are directed by orientation rules. In step 3 (lines 24–25), the structure decomposes sub-structures. In step 4 (lines 26–30), for each sub-structure, the RAI algorithm is applied recursively while increasing the order (the number of conditional variables in a CI test).

Using this procedure, the RAI algorithm can achieve lower computational costs than those of the traditional the constraint-based algorithms.

Algorithm 1 The RAI algorithm

$g_s = (\mathbf{V}_s, \mathbf{E}_s)$: Input graph structure (Default: complete undirected graph structure)
$g_o = (\mathbf{V}_o, \mathbf{E}_o)$: Output graph structure
$g_{ex} = (\mathbf{V}_{ex}, \mathbf{E}_{ex})$: Set of sub-structure (Default: empty set)
$g_{all} = (\mathbf{V}_{all}, \mathbf{E}_{all})$: Temporary graph structure
N_z: The number of conditional variables (Default: 0)
function RAI(N_z, g_s, g_{ex}, g_{all}, $\mathbf{D}$)
    $g_{all} \leftarrow g_s$
    if all nodes in g_s have fewer than $N_z + 1$ potential parents **then**
        return $g_o \leftarrow g_s$
    else
        for node $Y \in \mathbf{V}_s$, node $X \in \mathbf{V}_{ex}$ **do**
            if conditional independence between X and Y given conditional variables $\mathbf{Z}$ **then**
                Remove edge between X and Y from g_{all}.
            end if
        end for
        Direct the edges in g_s using orientation rules.
        for node $Y \in \mathbf{V}_s$, node $X \in \mathbf{V}_s$ **do**
            if conditional independence between $X \not\perp\!\!\!\perp Y$ given conditional variables $\mathbf{Z}$ **then**
                Remove edge between X and Y from g_{all} and g_s.
            end if
        end for
        Direct the edges in g_s using orientation rules.
    end if
    Group the nodes in g_s having the lowest topological order into a substructure g_c.
    Remove g_c from g_s temporarily and define the resulting unconnected structures as sub-structures $g_{p_1}, g_{p_2}, \cdots, g_{p_k}$.
    for $i = 1$ *to* k **do**
        $g_o \leftarrow$ RAI($N_z + 1$, g_{p_i}, g_{ex}, g_{all})
    end for
    $g_{ex_c} \leftarrow \{g_{p_1}, g_{p_2}, \cdots, g_{p_k}, g_{ex}\}$ as set of sub-structures from g_s except g_c
    $g_o \leftarrow$ RAI($N_z + 1$, g_c, g_{ex_c}, g_{all})
end function

5. Numerical experiments

5.1 Experimental design

This section presents some numerical experiments conducted to evaluate the efficiency of the method presented by Natori et al. (2015). Particularly, we compare the performances of the method with $\alpha_{g_3}, \alpha_{g_i} = 1/2$ (Natori et al., 2015), those with BDeu (ESS=1.0) (Steck and Jaakkola, 2002), those with G^2 test and those with the original RAI (Yehezkel and Lerner, 2009). In the original RAI, we use threshold 0.003 for the CI test, using the conditional mutual information in the same manner as Yehezkel and Lerner (2009). Table (1) presents CI tests used for the experiments.

In the experiments, we applied all CI tests in Table 1 to the RAI algorithm, the MMHC algorithm, and the PC algorithm. We compare run-times and learning accuracies of those methods using some benchmark networks. The method by Natori et al. (2015) is expected to decrease the run-time compared to other algorithms because higher-order CI tests between two variables are not required when a lower-order test detects CI between the variables. Thereby, the proposed method is expected to learn larger network structures than other

Table 1: Comparative CI tests

	method
Natori	Bayes factor ($\alpha_{g3}, \alpha_{g_i} = 1/2$)(Natori et al., 2015; Clarke and Barron, 1994)
Steck	Bayes factor ($\alpha_{ijk} = 1/r_i q_i$)(Steck and Jaakkola, 2002)
G^2	G^2 test (significance-level: 0.05)
CMI	Conditional mutual information (threshold: 0.003) (Yehezkel and Lerner, 2009)

algorithms do because it greatly improves the computational efficiency. Additionally, we calculate the Structural Hamming Distance (SHD) (Tsamardinos et al., 2006), which is the most useful metric for evaluating the learning accuracy of extremely large networks.

Experiments were performed on a 2.7 GHz 12-Cores Intel Xeon with 128 GB of RAM, 256 GB of SSD space and running Mac OS 10.11.4. Benchmark datasets were obtained from the *bnlearn* repository (Scutari, 2011). The RAI algorithm and the PC algorithm were implemented with MATLAB using the Bayes Net Toolbox (BNT) (Murphy, 2001). The MMHC algorithm was implemented using Java. For this experiment, we set 6 hr as the maximum running time.

5.2 Experiment results

As described in this subsection, we conducted the following experiments using the six benchmark networks (Scutari, 2011) presented in Table 2. Table 2 presents the number of variables, number of edges, maximum number of states in a variable, the maximum number of parents of a variable ('maximum in-degree') for each variable, and the range of sample sizes.

Table 2: Benchmark networks

network	variables	edges	maximum number of states	maximum in-degree	n
cancer	5	4	2	2	10,000–2,000,000
sachs	11	17	2	3	10,000–200,000
alarm	37	46	4	4	10,000–10,000,000
win95pts	76	112	2	7	10,000–2,000,000
andes	223	338	2	6	10,000–2,000,000
munin	1041	1397	2	3	10,000–200,000

The procedures of the experiments are described below.

1. We generated samples randomly from each benchmark network.
2. Given the generated samples, Bayesian network structures were estimated using each method.
3. Procedures 1 and 2 were repeated 10 times. The average run-time and the average number of CI tests in each order were calculated.

The run-times and SHD averages are presented, respectively, in Tables 3 and 4. In Tables 3 and 4, "n" stands for the sample size, the other values in Tables 3 and 4 respectively denote

Table 3: Average run-time (s) of each method using benchmark networks

cancer

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	0.08	0.09	0.07	0.06	MMHC	0.17	0.16	0.20	0.16	PC	0.09	0.08	0.06	0.03
20,000		0.12	0.15	0.10	0.04		0.54	0.48	0.63	0.47		0.13	0.13	0.10	0.03
50,000		0.20	0.29	0.17	0.06		1.23	1.17	1.47	0.98		0.23	0.27	0.16	0.05
100,000		0.34	0.50	0.30	0.11		1.72	2.22	1.76	1.27		0.42	0.53	0.33	0.08
200,000		0.61	0.91	0.50	0.16		3.07	3.50	3.38	2.52		0.79	1.01	0.59	0.13
500,000		1.81	2.46	1.48	0.47		7.48	7.47	6.94	5.24		2.10	2.66	1.65	0.38
1,000,000		4.34	6.16	3.51	0.78		15.17	14.16	13.73	9.83		5.40	7.02	4.09	0.73
2,000,000		8.82	13.24	6.99	1.89		30.59	31.80	26.81	18.80		11.22	15.36	8.25	1.76

sachs

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	2.47	1.75	1.03	1.68	MMHC	0.91	0.88	0.96	19.18	PC	4.37	3.02	1.80	1.67
20,000		4.36	3.34	2.00	2.28		1.61	1.31	1.64	37.66		7.21	5.57	2.89	2.32
50,000		7.91	6.65	2.41	3.98		3.05	2.91	3.31	84.73		12.73	10.75	5.46	4.02
100,000		14.34	13.36	5.50	7.06		5.54	5.25	7.14	174.27		23.61	20.90	10.25	6.79
200,000		22.18	27.23	13.24	14.07		11.34	10.05	13.36	336.88		21.34	28.24	13.23	13.41

alarm

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	3.62	5.37	3.87	4.44	MMHC	4.86	4.94	5.37	OT	PC	7.15	6.31	5.48	1.78
200,000		56.80	74.13	78.21	32.71		135.50	137.98	167.91	OT		69.49	90.75	72.20	15.61
500,000		144.54	174.83	240.24	84.83		350.68	359.12	456.77	OT		174.19	241.49	182.60	35.69
1,000,000		339.07	371.61	512.21	166.1		728.09	758.06	812.28	OT		387.92	463.17	402.54	71.19
2,000,000		707.29	962.18	1080.10	333.56		1287.36	1347.12	1768.79	OT		810.09	1127.20	863.22	143.11
10,000,000		5949.20	8313.90	9626.70	2406.30		6782.79	7708.54	9278.05	OT		7463.50	9584.30	7530.40	1165.90

win95pts

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	13.08	15.39	6.27	6.10	MMHC	OT	OT	OT	OT	PC	16.08	13.25	3748.30	3.31
20,000		21.19	25.20	7.97	8.39		OT	OT	OT	OT		26.76	24.11	4359.70	4.71
50,000		36.66	43.66	27.53	14.65		OT	OT	OT	OT		53.54	56.27	29.46	9.89
100,000		60.67	80.77	49.93	27.02		OT	OT	OT	OT		107.55	138.42	63.69	18.74
200,000		117.4	156.42	93.15	48.05		OT	OT	OT	OT		262.28	326.64	118.34	39.89
500,000		325.75	414.55	310.76	116.96		OT	OT	OT	OT		467.13	1051.3	545.86	111.62
1,000,000		691.77	862.38	766.20	270.57		OT	OT	OT	OT		1171.90	2201.20	1258.30	332.05
2,000,000		1635.90	2120.40	1588.10	593.36		OT	OT	OT	OT		2530.60	5429.30	2701.80	647.61

andes

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	63.59	58.85	66.43	36.83	MMHC	167.27	133.79	OT	OT	PC	205.25	122.34	150.03	139.12
20,000		106.62	95.99	95.68	46.58		358.29	306.24	OT	OT		379.13	228.36	239.62	191.09
50,000		193.28	166.70	164.75	94.12		1101.53	931.14	OT	OT		644.96	537.28	509.34	341.10
100,000		355.54	320.45	321.78	148.76		2732.94	2463.15	OT	OT		1314.50	1122.80	1319.30	576.970
200,000		777.27	616.17	622.31	252.54		6844.92	6017.95	OT	OT		2491.10	2375.00	2256.20	1288.70
500,000		2080.1	1668.70	1733.70	568.41		OT	OT	OT	OT		6466.30	9511.90	6097.50	3495.60
1,000,000		4421.80	3934.20	4853.20	1055.30		OT	OT	OT	OT		15555.00	20151.00	15372.00	7358.30
2,000,000		10345.00	11641.00	14005.00	2286.60		OT	OT	OT	OT		OT	OT	OT	17018.00

munin

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	2490.80	943.79	2398.60	1522.80	MMHC	OT	OT	OT	OT	PC	OT	OT	OT	OT
20,000		3756.40	1851.50	5514.70	1903.10		OT	OT	OT	OT		OT	OT	OT	OT
50,000		5897.60	11336.00	OT	2924.90		OT	OT	OT	OT		OT	OT	OT	OT
100,000		10261.00	OT	OT	5260.60		OT	OT	OT	OT		OT	OT	OT	OT
200,000		14476.00	OT	OT	8516.80		OT	OT	OT	OT		OT	OT	OT	OT

means of comparative CI tests from Table 1, and " OT (over time)" signifies failure to learn the structure because of running for more than 6 hr.

Table 3 shows that the CI tests conducted using the RAI algorithm exhibit the fastest learning. The run-times of the RAI algorithm using the CI tests increase linearly as the sample size increases. The run-times of the MMHC algorithm and the PC algorithm using the CI tests increase rapidly as the sample size increases. Results show that the MMHC and the PC algorithm were unable to learn within the time limit as the number of variables increased. In the RAI algorithm, Bayes factor (Natori) finished far faster than other methods, except for CMI. Moreover, Bayes factor (Steck) and G^2 could not finish learning within the time limit using the munin network.

In Table 4, Bayes factor (Natori, Steck) with the RAI algorithm showed that the learning accuracy was almost equal to that of the PC algorithm for cancer, sachs, and win95pts networks. Especially, SHDs of these methods converged to zero in the cancer and sachs networks. According to results obtained using the alarm network, Bayes factor (Natori,

Table 4: Average SHD error of each method using benchmark networks

cancer

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	3.0	2.7	3.4	3.0	MMHC	1.2	1.3	1.1	3.5	PC	2.5	2.5	2.4	3.0
20,000		2.9	2.8	3.2	3.0		1.1	1.5	1.9	2.9		2.3	2.4	1.8	3.0
50,000		3.4	3.3	1.8	3.0		0.4	0.3	0.2	2.4		1.8	1.9	1.0	3.0
100,000		2.8	2.9	1.2	3.0		0.1	0.3	0.1	3.1		1.5	1.6	0.7	3.0
200,000		2.5	2.8	0.7	3.0		0.2	0.3	0.5	2.5		1.2	1.3	0.5	3.0
500,000		1.2	1.2	1.1	3.0		0.1	0.2	0.7	2.8		0.6	0.6	1.0	3.0
1,000,000		0.4	0.6	0.3	3.0		0.6	0.2	0.2	3.6		0.2	0.3	0.3	3.0
2,000,000		0.0	0.0	0.1	3.0		0.0	0.1	0.1	2.8		0.0	0.0	0.1	3.0

sachs

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	16.6	16.2	12.5	18.3	MMHC	17.0	17.0	17.0	17.0	PC	13.2	14.1	15.8	14.0
20,000		15.5	17.3	1.6	18.3		17.0	17.0	17.0	17.0		11.7	12.9	16.3	13.8
50,000		14.0	14.0	7.7	17.9		17.0	17.0	17.0	17.0		10.0	10.2	14.2	14.0
100,000		5.6	12.6	7.7	18.4		17.0	17.0	17.0	17.0		14.2	10.0	17.0	14.0
200,000		0.0	0.0	12.5	17.9		17.0	17.0	17.0	17.0		0.0	0.0	0.0	14.0

alarm

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	25.3	24.3	49.1	36.4	MMHC	22.4	23.0	22.5	OT	PC	17.1	18.7	14.3	37.2
200,000		20.2	25.6	56.9	31.3		23.6	24.3	25.1	OT		10.0	8.8	10.5	37.0
500,000		15.5	17.0	54.2	31.7		22.9	22.1	26.7	OT		10.0	3.0	2.9	37.0
1,000,000		17.5	16.2	54.2	31.8		20.5	22.7	25.7	OT		2.8	3.0	2.2	37.0
2,000,000		16.9	15.9	55.6	32.0		20.9	21.5	23.8	OT		2.0	2.0	1.7	37.0
10,000,000		15.1	18.2	56.5	32.0		25.0	22.7	25.2	OT		1.0	0.6	0.9	37.0

win95pts

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	48.7	59.9	139.2	59.3	MMHC	OT	OT	OT	OT	PC	51.8	52.8	67.0	74.2
20,000		44.3	53.9	137.6	57.1		OT	OT	OT	OT		46.4	46.6	72.6	74.8
50,000		39.1	44.2	61.6	55.8		OT	OT	OT	OT		40.8	41.2	40.0	73.5
100,000		33.3	36.5	58.3	56.9		OT	OT	OT	OT		39.1	37.2	37.8	76.1
200,000		34.2	36.8	56.8	57.1		OT	OT	OT	OT		36.5	35.7	37.1	74.4
500,000		34.1	35.5	59.7	56.2		OT	OT	OT	OT		34.0	31.3	32.9	75.8
1,000,000		32.0	33.3	61.6	56.2		OT	OT	OT	OT		30.9	30.2	30.4	75.8
2,000,000		31.3	33.3	56.8	55.9		OT	OT	OT	OT		29.8	28.6	30.1	74.5

andes

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	72.7	70.1	174.9	84.8	MMHC	227.5	223.9	OT	OT	PC	113.0	137.5	97.6	113.4
20,000		48.8	51.8	159.5	84.3		219.6	216.3	OT	OT		97.4	112.4	79.6	112.6
50,000		30.2	29.8	152.2	85.6		213.0	211.9	OT	OT		88.2	92.0	54.5	112.7
100,000		26.1	23.3	147.1	85.1		210.5	205.6	OT	OT		72.5	74.5	43.0	113.1
200,000		21.4	18.9	149.0	85.8		202.5	199.8	OT	OT		52.3	50.2	29.8	112.8
500,000		17.4	15.7	150.7	85.3		OT	OT	OT	OT		25.5	22.4	21.1	111.4
1,000,000		15.8	14.7	149.5	86.0		OT	OT	OT	OT		13.0	15.8	19.0	112.4
2,000,000		14.5	12.4	147.3	85.5		OT	OT	OT	OT		OT	OT	OT	111.6

munin

n	algorithm	method				algorithm	method				algorithm	method			
		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI		Natori	Steck	G^2	CMI
10,000	RAI	524	507	1786.7	812.8	MMHC	OT	OT	OT	OT	PC	OT	OT	OT	OT
20,000		476.7	504.1	1754.3	793.5		OT	OT	OT	OT		OT	OT	OT	OT
50,000		447.2	527.4	OT	798.3		OT	OT	OT	OT		OT	OT	OT	OT
100,000		444.6	OT	OT	801.8		OT	OT	OT	OT		OT	OT	OT	OT
200,000		439.9	OT	OT	780.9		OT	OT	OT	OT		OT	OT	OT	OT

Steck) with the PC algorithm provided the best performances for a large sample size because the PC algorithm performs all high-order CI tests and removes unnecessary edges exactly. However, it cannot learn a huge network structure because the run-times increase rapidly. In fact, it was unable to learn the andes and the munin networks. In the RAI algorithm, Bayes factor (Natori) provided the best performance aside from the andes network. Additionally, the run-time of Bayes factor (Natori) is surprisingly faster than Bayes factor (BDeu) does. The reason is that Natori's CI tests are more accurate than those of BDeu. In the andes network, Bayes factor (Steck) showed slightly more accurate results. Therefore, Bayes factor (Natori) can be expected that the learning accuracy can be improved by increasing

the sample size. However, the learning accuracies of the MMHC algorithm were somehow unstable. In this experiment, Bayes factor (Natori, Steck) decreased SHD value only for cancer and andes network. The learning accuracies of G^2 and CMI showed that the learning accuracy is worse than those of Bayes factor (Natori, Steck). Particularly the learning accuracies of CMI were the worst in all results. The reason that G^2 test entails the difficulty that the type I error converges to a user-determined significance level as the sample size increases. The learning accuracies of the conditional mutual information strongly depend on the threshold setting.

6. Conclusions

In this study, we demonstrated that a constraint-based learning Bayesian network using Bayes factor can learn a surprisingly huge network with thousands of variables. Additionally, we evaluated the run-time and learning accuracy to evaluate the efficiency of the proposed method by Natori et al. (2015). Results show that the proposed method improved the learning efficiency remarkably. Particularly, we demonstrated that the proposed method can complete learning faster and more accurately than any other method.

The results also suggest that the learning accuracy might depend on the hyperparameter values. Actually, Ueno (2010, 2011) theoretically showed that the optimal hyperparameter of Dirichlet prior becomes larger for smaller sample sizes.

Our future work will assess a method of adjusting optimal values of the hyperparameter for various tradeoff situations between the network size and the data size. Additionally, we will improve the proposed learning algorithm to reduce the computational cost dynamically.

References

J. Abellán, M. Gomez-olmedo, and S. Moral. Some Variations on the PC Algorithm. In *Probabilistic Graphical Models (PGM)*, pages 1–8, 2006.

A. Agresti and M. Kateri. *Categorical Data Analysis*. Springer, 2011.

W. Buntine. Theory Refinement on Bayesian Networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 52–60, 1991.

J. Cheng, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. In *Conference on Information and Knowledge Management (CIKM)*, pages 325–331, 1997.

D. M. Chickering. Learning Bayesian Networks is NP-Complete. In *Learning from Data: Artificial Intelligence and Statistics*, volume V, pages 121–130. Springer, 1996.

B. S. Clarke and A. R. Barron. Jeffreys' prior is asymptotically least favorable under entropy risk. *Journal of Statistical Planning and Inference*, 41(1):37–60, 1994.

J. Cussens. Bayesian network learning with cutting planes. In *Uncertainty in Artificial Intelligence (UAI)*, pages 153–160. AUAI Press, 2011.

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

R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90: 773–795, 1995.

C. Li and M. Ueno. An extended depth-first search algorithm for optimal triangulation of Bayesian networks. *International Journal of Approximate Reasoning*, 80:294–312, 2017.

B. Malone, C. Yuan, E. Hansen, and S. Bridges. Improving the Scalability of Optimal Bayesian Network Learning with External-Memory Frontier Breadth-First Branch and Bound Search. In *Uncertainty in Artificial Intelligence (UAI)*, pages 479–488. AUAI Press, 2011.

K. P. Murphy. The Bayes Net Toolbox for MATLAB. *Computing Science and Statistics*, 33, 2001.

K. Natori, M. Uto, Y. Nishiyama, S. Kawano, and M. Ueno. Constraint-based learning Bayesian networks using Bayes factor. In *Advanced Methodologies for Bayesian Networks (AMBN) 2015*, volume LNAI 9505, pages 15–31. Springer, 2015.

M. Scanagatta, C. P de Campos, G. Corani, and M. Zaffalon. Learning Bayesian Networks with Thousands of Variables. In *Neural Information Processing Systems (NIPS 2015)*, pages 1864–1872, 2015.

M. Scutari. Learning Bayesian Networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2011.

T. Silander and P. Myllymaki. A simple approach for finding the globally optimal Bayesian network structure. In *Uncertainty in Artificial Intelligence (UAI)*, pages 445–452. AUAI Press, 2006.

T. Silander, P. Kontkanen, and P. Myllymaki. On Sensitivity of the MAP Bayesian Network Structure to the Equivalent Sample Size Parameter. In *Uncertainty in Artificial Intelligence (UAI)*, 2007.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT press, 2000.

H. Steck and T.S. Jaakkola. On the Dirichlet Prior and Bayesian Regularization. In *Neural Information Processing Systems (NIPS 2002)*, pages 697–704. MIT Press, 2002.

J. Suzuki. A Theoretical Analysis of the BDeu Scores in Bayesian Network Structure Learning. *Behaviormetrika*, 1, 2016.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

M. Ueno. Learning likelihood-equivalence Bayesian networks using an empirical Bayesian approach. *Behaviormetrika*, 35((2)):115–135, 2008.

M. Ueno. Learning networks determined by the ratio of prior and data. In *Uncertainty in Artificial Intelligence (UAI)*, pages 598–605. AUAI Press, 2010.

M. Ueno. Robust learning Bayesian networks for prior belief. In *Uncertainty in Artificial Intelligence (UAI)*, pages 698–707, 2011.

R. Yehezkel and B. Lerner. Bayesian Network Structure Learning by Recursive Autonomy Identification. *Journal of Machine Learning Research*, 10:1527–1570, 2009.

C. Yuan, B. Malone, and W. Xiaojian. Learning Optimal Bayesian Networks Using A* Search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2186–2191, 2011.