An Efficient Approach for Parameters Learning of Bayesian Network with Multiple Latent Variables Using Neural Networks and P-EM

Kaiyu Song, Kun Yue(✉), Xinran Wu, and Jia Hao

School of Information Science and Engineering, Yunnan University, Kunming 650500, China
kyue@ynu.edu.cn

Abstract. Bayesian network with multiple latent variables (BNML) is used to model realistic problems with unobservable features, such as diagnosing diseases and preference modeling. However, EM based parameter learning for BNML is challenging if there is a large amount of intermediate results due to missing values in the training dataset. To address this issue, we propose the clustering and P-EM based method to improve the performance of parameter learning. First, an innovative layer of neural network is defined based on Recurrent Neural Network (RNN) by incorporating the structural information of BNML into the Mixture of Generative Adversarial Network (MGAN), which can reduce the number of parameters by enabling clustering in an unsupervised manner. We then propose a Parabolic acceleration of the EM (P-EM) algorithm to improve the efficiency of convergence of parameter learning. In our method, the geometry knowledge is adopted to obtain an approximation of the parameters. Experimental results show the efficiency and effectiveness of our proposed methods.

Keywords: Bayesian network · Latent variable · Generate adversarial network · Recurrent neural network · Parameter learning · Expectation maximization

1 Introduction

With the rapid development of Web2.0, unprecedented amount of user behavioral data can be collected and analyzed to facilitate the development of more realistic applications in economic and medical systems [6, 23]. In general, the features implied in user behavioral data are multi-dimensional. For instance, a user rates the movie, "*Titanic*", with a 5 star rating as he/she prefers *English* movies and *love* stories, shown as Fig. 1. Specifically, 75% of *female* users consider the *genre* first, and out of them, 88% of the users might prefer *love* stories. As it is seen in this example, multi-dimensional user preferences are often unobservable. Furthermore, various associations with uncertainties exist among users' properties, the attributes of the movies, the observed ratings, as well as other latent preferences on various other aspects of the movies. To enable the analysis of realistic applications, it is desirable to construct a human-like model to achieve interpretable inferences, in which the unobservable features and uncertain associations are represented.

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 357–372, 2021.
https://doi.org/10.1007/978-3-030-67537-0_22

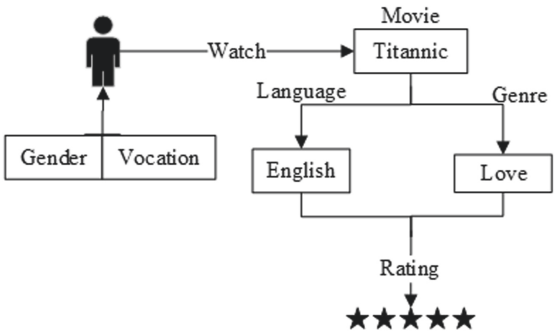

Fig. 1. An example of movie rating.

Probabilistic graphical model (PGM) is an effective framework to describe the dependency relationships among variables, and further quantify the above mentioned relationships. As an important PGM, Bayesian Network (BN) can be used to represent and infer the dependency relationships by a directed acyclic graph (DAG), combining with conditional probability tables (CPTs) [12]. In this paper, we adopt a BN with multiple latent variables, abbreviated as BNML, to model the unobserved features with uncertainties from user behavioral data. Oriented to preference modeling in movie rating as in Fig. 1, the corresponding BNML is illustrated in Fig. 2, where the latent variable is shown with dashed borders.

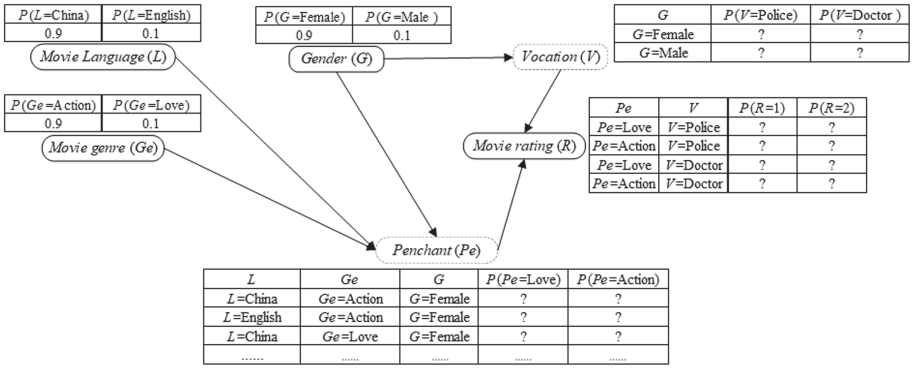

Fig. 2. An example of BNML.

The learning of BNML includes parameter learning and structure learning [12]. Generally, the structural expectation maximization (SEM) algorithm [22] is used for BNML learning. However, parameter learning in the SEM is highly complex due to the presence of missing values of latent variables. Expectation maximization (EM) [1] can be also adopted to fulfill parameter learning, where all possible combinations of the values of latent variables are generated for each incomplete sample. Then, the conditional probabilities of each complete sample are iteratively computed to obtain the optimal parameters. The complexity of parameter learning is $\Theta(pc^s)$, where p and s are the

number of iterations and that of latent variables respectively. c is a constant number greater than 1, related to the number of parameters. Therefore, EM based parameter learning is also inefficient due to the large amount of intermediate results.

The execution time of parameter learning can be exponentially reduced by decreasing the number of latent variables, which is, however, not applicable to most cases of realistic problems. Therefore, it is essential to further investigate the method of learning parameters of BNML. To address this issue, we propose the technique by reducing the number of parameters to speed up the convergence of parameter learning.

We first propose a clustering based method to process the original dataset, which can significantly reduce the number of parameters. In this paper, Mixture of GAN (MGAN) [30] is adopted to implement clustering with an embedding process by generating a new distribution for each class in an unsupervised manner. Moreover, Recurrent Neural Network [15] (RNN) is the special neural network using the recurrent mechanism to pass the information from prior step, which could be used to process the information with structural characteristics. In this paper, RNN based MGAN (RMGAN) is proposed to cluster the observed features into several classes, by which the original samples are replaced by the outputs of RMGAN to reduce the number of parameters. Specifically, MGAN uses GAN to train the Gaussian mixture model [30] to fulfill clustering, in which a dense layer with the softmax activation function [29] is incorporated to generate the labels associated to the samples. By using this approach, we can build multiple generators and discriminators to use labels and approximate the hyperparameter of distribution for each class.

Then, we propose the RNN layer to refine the dense layer. In the proposed RNN layer, the values from the prior RNN cell and the RNN cells associated with the next RNN cell according to the DAG of BNML could be considered. This enables one-by-one correspondence of each feature to the RNN cells. Thus, the values of the prior and related cells can be used in the recurrent mechanism if there is an edge in the DAG of BNML between the two cells. By replacing the dense layer in MGAN with our proposed RNN layer, RMGAN can use the structural information of BNML to fulfill clustering and avoid eliminating useful information as much as possible. This establishes the relationship between neural network and BNML.

To further guarantee the efficiency of parameter learning, we use parabolic acceleration of the EM algorithm (P-EM) [4] to refine the EM based parameter learning. P-EM adopts an innovative geometrical method to increase the convergence of EM algorithm by estimating the approximated value of the maximum likelihood function [12]. Thus, a parameter learning method is proposed to reduce the number of iterations by using the updating function of P-EM to replace that of the EM based method.

Generally, the contributions of this paper are as follows:

- We propose RMGAN by rebuilding the recurrent mechanism in the RNN based on MGAN. Thus, clustering could be achieved and the information in the DAG of BNML could be incorporated. Further, the outputs of RMGAN could be used to reduce the number of parameters.
- We present an improved method for parameter learning based on P-EM to reduce the number of iterations and improve the efficiency of convergence.

- We conduct experiments on Alarm, MovieLens, and Cardiovascular datasets. Experimental results show that our proposed methods improve the efficiency of parameter learning by reducing the number of parameters and iterations.

2 Related Work

BN is widely used in real applications, such as medical analysis [6] and recommendation systems [23]. However, it is not easy to calculate the parameters of BN in presence of missing values. Several methods were proposed to perform parameter learning based on incomplete data, including EM, Robust Bayesian estimation, Monte-Carlo and Gaussian approximations [18].

To overcome the efficiency bottleneck in parameter learning, some optimization techniques were proposed, among which the concept of clique tree was incorporated to avoid repeated computations using shared computing [27]. Quasi Newton [11] was also used to make EM algorithm more efficient by adopting Quasi Newton function to generate an approximation instead of an accurate estimation. The Damped Anderson acceleration method [10] was further proposed by accelerating the iterations. Constraints were used to reduce the search space during the learning of parameters and structure [7, 8]. MapReduce was adopted to improved parameter learning by accelerating EM [2, 26]. However, few previous works were used for BNML, due to the large amount of intermediate results generated by missing values after adding multiple latent variables.

Dimensionality reduction achieved by embedding has been widely adopted as the effective means to improve the efficiency of machine learning algorithms such as categorization in computer vision [30]. In particular, several unsupervised methods can be used to implement clustering to finish categorization, e.g., k-means [20] and GAN [9]. Recently, ClusterGAN [21] was presented to fulfill clustering based on the potential feature space. Few previous works were used to process structural information, since it is difficult to generate structural information for most plications. In this paper, we are to implement clustering by incorporating the structural information from DAG.

3 RMGAN for Clustering

3.1 Preparation

Definition 1. BNML is a BN with more than one variable, denoted as $B(G, \theta)$, where $G = (V, E)$ is the DAG of B. V, O and L are the sets of all variables, observed variables and latent variables, respectively, $V = O \cup L$. E is the set of directed edges. θ is the set of probability parameters consisting of CPTs, and

- $\pi(V_i)$ denotes the set of parent nodes for the i^{th} variables, V_i in G.
- θ_{ihk} denotes the parameter of V_i, where the value of V_i is equal to h and the value of its parent nodes take the k^{th} combination.

Inputs. To avoid eliminating useful information as much as possible, the inputs of RMGAN include the features of some observed variables and the relationships in G

among these observed variables. First, some observed variables are chosen as the inputs of RMGAN. Then, we build the relationships among the chosen variables. For instance, it is reasonable to build a relationship between *gender* and *vocation.*

Then, we choose the adjacent matrix, denoted as G_{mm}, to save these relationships in G and then feed G_{mm} into RMGAN, where G_{mm}^{ij} denotes the values of E_{ij}, and E_{ij} denotes the relationship from V_i and V_j ($V_i, V_j \in V, i \neq j$). G_{mm}^{ij} is set to 1 if there is a relationship between V_i and V_j, otherwise, G_{mm}^{ij} is equal to 0. In addition, we keep the relationships among the chosen variables unchangeable in the structure part of SEM, such that the change of G in SEM does not affect RMGAN.

3.2 Building the Structure of RMGAN

Our clustering model built by RMGAN is based on MGAN [30]. The structure of MGAN includes several generators, discriminators and a dense layer. We propose a novel RNN layer, referred to as DRNN, to refine the dense layer. The generator and discriminator are the same as those in MGAN. The hypothesis of the whole DRNN is defined as $H(x) = f_D(Flatten(f_R(X, G_{mm})))$, where $f_R(X, G_{mm})$, *Flatten* and f_D are the hypothesis of RNN, flatten layer and dense layer, respectively.

RNN Layer. We propose the new component, called state gate, between two RNN cells to control the values generated by the recurrent mechanism. Specifically, we generate the number of RNN cells with the same number of the chosen variables in G_{mm}. This achieves one-by-one matching between features of the chosen variables and the RNN cells. For instance, the feature of the n^{th} variable is fed to the n^{th} RNN cells. In $E_{ij} = 1$ for $i = n$ and $j = n - 1$, the value of the $n - 1^{\text{th}}$ RNN cell is considered during recurrent. This can be achieved as follows:

$$\begin{cases} f(x^n) = f(x^{n-1}), n = 2 \\ f(x^n) = f(f^{n-1}(x^{n-1}) + S(x^n)), 2 < n < a \end{cases} \tag{1}$$

where $f(x)$ is the hypothesis of each RNN cell, $S(x)$ is the function of the state gate, and a is the number of observed variables in O. Details of building $f(x)$ refer to [15]. Matrix $\boldsymbol{y}$ is adopted to record the result of the RNN cell in the state gate, which is updated after each RNN cell transmits the result to the state gate as given in Eq. 2.

$$y = [f(x^1), f(x^2), \ldots\ldots f(x^n)] \tag{2}$$

If the RNN cell in the n^{th} time step sends the result $f(x^n)$ to the state gate, the value of $\boldsymbol{y}[n]$ is updated by $f(x^n)$ as in Eq. 1. To calculate the values in the recurrent mechanism, we define the new function $S(x)$ in Eq. 3.

$$S\left(x^n\right) = <G_{mm}[n], y> + f(x^{n-1}) \tag{3}$$

where $G_{mm}[n]$ denotes the values of the n^{th} column of G_{mm}. y is the values in $\boldsymbol{y}$ and the inner product is dot product [28].

Therefore, if G_{mm}^{ij} between the current and other cells is equal to 1, where $i = n$ and $j = n - 1$, the values of hypothesis of related cells and prior cells are both considered during calculating the hypothesis of the current cell.

Flatten Layer. The flatten layer collects the features from RNN and sends them to the dense layer with softmax activation.

Dense Layer. The dense layer is similar to that in MGAN, composed of several fully connected layers. We set the activation of the output layer in the dense layer as softmax [29].

Therefore, DRNN can calculate the relationships in G shown as Fig. 3, where R_i is the i^{th} RNN cell, T is the state gate, F is the flatten layer and M is the output layer of the dense layer that sets the activation function as softmax. We then put DRNN into MGAN and use the generators as well as discriminators to fulfill clustering. The training strategy of RMGAN is the same as the original training strategy in MGAN [30]. In addition, the outputs of RMGAN are the parameters of distributions and labels.

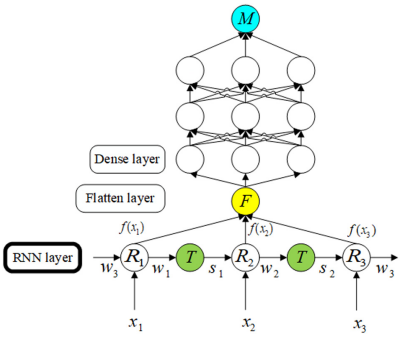

Fig. 3. Structure of the DRNN.

Finally, to reduce the number of combinations among variables, we rebuild the dataset by the outputs of RMGAN, used to generate new values for all samples and replace the original values. In other words, RMGAN generates new distributions $Para^i_{new}$ by using $Distri^*_i$ for the i^{th} class, and $Para^i_{new}$ is the parameter of new distribution for the chosen variables. Then, $Para^i_{new}$ is used to replace the original sample values belonging to the i^{th} class in the dataset. This reduces the number of combinations among the clustered variables in the rebuilt dataset to the number of classes, which is less than the original number of combinations. Correspondingly, the number of parameters in the CPTs is reduced [12]. Thus, the value of c is also accordingly reduced, since c is related to the number of parameters in the CPTs.

4 P-EM Based Parameter Learning

4.1 Preparation

Inputs. The inputs of PPL contain the DAG of BNML and the rebuilt dataset, denoted as $D = (D_1, D_2, \ldots, D_z)$ and $D_z = (O_1, O_2, ..., O_a)$. Note that replacement by generating D reduces the number of combinations in V, and further cleans the preliminary data cleaning, since the data is clustered into the same class to eliminate unnecessary information in the original dataset. Therefore, the efficiency of parameter learning is further improved by using the outputs of RMGAN.

EM Based Parameter Learning. EM based parameter learning (EPL) uses the EM based fragment updating method (EMFU) [12]. The whole process of EPL is based on SEM in which the DAG structure, G, is complete, since SEM can provide a complete G during parameter learning part. Therefore, $G^{''}$ denotes as the complete G generated by SEM and contains all information of E among V. EMFU is composed of E-step, and M-step. In the E-step, EMFU extends D by all possible values of L to generate the extended dataset $D^e = (D_1^e, D_2^e, \ldots, D_z^e)$, where $D_z^e = (O_1 \ldots O_a, L_1 \ldots L_s)$. Then, the set of weights is denoted as W and calculated as follows:

$$m_{ihk} = \sum_{l=1}^{z} P(V_i = h, \pi(V_i) = k | (O_l, L_l)) \tag{4}$$

In the initial E-step, the set of initial parameters θ^0 is also generated randomly. In the M-step, θ_{ihk} is calculated based on D^e and the weights by Eq. 5.

$$LL(\theta_{ihk}) = \begin{cases} \frac{m_{ihk}}{\sum_{h=1}^{r_i} m_{ihk}}, & if \sum_{h=1}^{r_i} m_{ihk} > 0 \\ \frac{1}{r_i}, & else \end{cases} \tag{5}$$

where r_i is the cardinality of V_i. The EPL then turns to the E-step to repeat the above process until θ satisfies the terminating condition [12].

4.2 P-EM Based Parameter Learning

Our proposed PPL is based on EMFU by using the function in Eq. 6 based on Bézier curve [4] to calculate the approximate values of maximum likelihood estimation.

Note that PPL needs 3 seeds to calculate the approximate values. In the E-step of the proposed PPL, the weights are calculated based on θ^0 by E-step in EPL. In the M-step of the proposed PPL, θ^1, θ^2 and θ^3 are also calculated based on the weights using EPL to generate the required seeds, denoted by P^0, P^1 and P^2 respectively, where $P^0 = (\theta^0, LL(\theta^0)) = (\theta^0, \theta^1)$, $P^1 = (\theta^1, LL(\theta^1)) = (\theta^1, \theta^2)$ and $P^2 = (\theta^2, LL(\theta^2)) = (\theta^2, \theta^3)$. Then, $P^{new} = (\theta^{new}, LL(\theta^{new}))$ can be obtained using Eq. 6 and Eq. 7.

$$P^{new} = (1-t)^2 * P^0 + 2 * t * (1-t) * P^1 + t^2 * P^2 \tag{6}$$

where

$$t = 1 + u^{\delta} g \tag{7}$$

where δ is the number of iterations of Eq. 6, u and g are two constants. Then, the new value of P^0 is replaced by P^{new} to generate the new seeds P^1 and P^2 by the same way to repeat the above process until δ is equal to the maximum number of iterations. As the terminating condition, we consider Euclidean distance [13] between $LL(\theta_{ihk}^{\text{new}}))$ and $LL(\theta_{ihk}^{2}))$ to reduce the number of parameters to be updated in each iteration.

$$\left\| LL(\theta_{ihk}^{new}) - LL(\theta_{ihk}^{2}) \right\|_2 < \sigma \tag{8}$$

where σ is the threshold of terminating condition, which is the same as that of EMFU, and the E-step of PPL is the same as the EMFU. We use vectorization to optimize the updating process of our proposed M-step. This enables all parameters to be calculated at the same time. First, the seeds are converted into the vectors and stored in a $3 \times \xi$ matrix, denoted as *Vec*, where ξ is the number of parameters in θ corresponding to θ_{ihk}. Then, Eq. 9 can be obtained by incorporating the vectorization into Eq. 6.

$$P^{new} = (1-t)^2 * Vec[0] + 2 * t * (1-t) * Vec[1] + t^2 * Vec[2] \tag{9}$$

Note that θ_{ihk} will not be updated if Eq. 8 is satisfied. Our proposed M-step of PPL is given in Algorithm 1. Iterator, denoted as *iter*, is built to iterate the values for all parameters saved in P^{new}, P^2 and *Vec*. By using PPL, the number of iterations required for convergence is reduced comparing to EMFU. Moreover, reducing the iterating times during the M-step can further improve the fitting of parameters and SEM. The value of p in the complexity of P-EM is less than that in the EM based parameter learning. Therefore, the total number of iterations in SEM is reduced.

Algorithm 1 M-step of PPL based on vectorization

Input:

G'', DAG of BNML generated by SEM

D^e, the extended D in E-step

W, the weights based on θ^0

u and g, two constants in Eq. 7

σ, threshold of the terminating condition

maxiter, the maximum number of iterations

Output: θ, the set of parameters of CPTs in BNML

Steps:

1. $\delta \leftarrow 0$; //number of iterations of Eq. 6
2. generating θ randomly;
3. generating seeds P^0, P^1 and P^2 based on W, G'' and D^e by Eq. 5;
4. $Vec[0] \leftarrow P^0$; $Vec[1] \leftarrow P^1$; $Vec[2] \leftarrow P^2$; //vectorization
5. $\theta \leftarrow Vec[2]$; $\xi \leftarrow \theta.length$; //number of parameters in θ
6. $flag \leftarrow [0, \ldots, 0]_{1\times N}$ //label updated
7. **while** $\xi > 0$ **do** //number of parameters need to be updated
8. $t \leftarrow 1+u^{\delta}g$; //by Eq. 7
9. $P^{new} \leftarrow (1-t)^2 * Vec[0]+2*t*(1-t)*Vec[1]+t^2*Vec[2]$; //by Eq. 9
10. $iter \leftarrow 0$; $iter.length \leftarrow \xi - 1$;//generate iterator for all θ_{ihk} in θ
11. **for** each $LL(\theta^{\text{new}}_{iter}))$ and $LL(\theta^2_{iter}))$ in P^{new} and P^2 **do**
12. **if** $\| LL(\theta^{new}_{iter}) - LL(\theta^2_{iter}) \|_2 < \sigma$ **and** $flag_{iiter}=0$ **then**
13. $\theta_{iiter} \leftarrow \theta^{new}_{iiter}$; $flag_{iter} \leftarrow 1$; $\xi \leftarrow \xi - 1$; $iter \leftarrow iter+1$; //$iter<iter.length$
14. **end if**
15. **end for**
16. $\theta^0 \leftarrow \theta^{new}$;
17. generating seeds P^0, P^1 and P^2 based on θ^0, G'', and D^e by Eq. 5;
18. $Vec[0] \leftarrow P^0$; $Vec[1] \leftarrow P^1$; $Vec[2] \leftarrow P^2$; $\delta \leftarrow \delta+1$;
19. **if** $\delta \geq maxiter$ **do**
20. $iter \leftarrow 0$;
21. **for** each $flag_{iter}$ in $flag$ **do**
22. updating θ_{iiter} by $Vec[2][iter]$; $iter \leftarrow iter+1$;
23. **end for**
24. **return** θ;
25. **end if**
26. **end while**

5 Experiments

5.1 Experimental Setup

Datasets. We adopted 3 publicly available datasets to evaluate the proposed RMGAN and P-EM accelerate methods, including Alarm[1], MovieLens[2], and Cardiovascular[3].

- **Alarm** contains 37 feature. The size of this dataset was set as 500, 1000 and 2000. We used this dataset to evaluate the training speed of our proposed method with many observed and latent variables in the DAG.
- **MovieLens** contains 7 features. The size of this dataset was set as 10000, 20000 and 40000. We used this dataset to evaluate the feasibility of proposed method for illustrating the validity of the BNML.
- **Cardiovascular** contains 12 features. The size of this dataset was set as 10000, 20000 and 40000. We used this dataset to evaluate the performance of BNML that uses P-EM combining with RMGAN to fulfill parameter learning in different tasks comparing with MovieLens.

Parameter Settings. For all datasets, the parameters of Adam optimizer [14] were fixed to 5e−5. In the improved updating P-EM, the threshold, u, g and the maximum number of iterations were set to 0.01, 1.5, 0.2, and 50, respectively.

Environment. Our experiments were carried out on a PC with RTX 2070 GPU, e1230 v5 CPU and 8G RAM. All codes were developed in Python 3.6. The neural network framework was tensorflow2.0.

Competing Models. The classical EM was used as the baseline to measure the efficiency. PNN [24] was used to measure the feasibility of the neural network model and BNML combining with our training methods. The SVD matrix factorization (MF) [16] was also used to measure the feasibility of the BNML combining with our proposed RMGAN and P-EM.

Evaluation Metrics. The execution time was used to evaluate the efficiency of parameter learning based on RMGAN and P-EM. RMSE, MSE and MAE were adopted to measure the accuracy of prediction made by BNML trained by several methods and P-EM, defined as $RMSE = \sqrt[2]{\sum_{i=1}^{\pi}(y_i - y_i')^2/\pi}$, $MSE = \sum_{i=1}^{\pi}(y_i - y_i')^2/\pi$, and $MAE = \sum_{i=1}^{\pi}|y_i - y_i'|/\pi$ respectively, where y_i is the true value of the i^{th} sample, π is the number of the samples, and y_i' is the predicted value.

[1] https://rdrr.io/cran/bnlearn/man/alarm.html.
[2] https://grouplens.org/datasets/movielens/1m/.
[3] https://www.kaggle.com/sulianova/cardiovascular-disease-dataset/data.

5.2 Experimental Results

Efficiency. To evaluate the efficiency of RMGAN combining with P-EM based parameter learning, we compared the execution time of the parameter learning using different methods. The comparisons were made for different sizes of Alarm dataset and different number of latent variables shown in Figs. 4, 5 and 6.

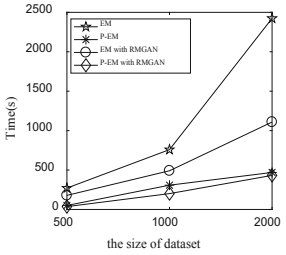

Fig. 4. Execution time on different sized datasets with 4 latent variables.

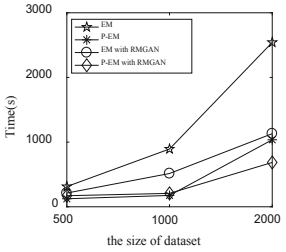

Fig. 5. Execution time on different sized datasets with 5 latent variables.

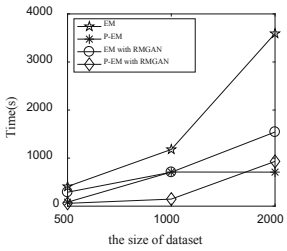

Fig. 6. Execution time on different sized datasets with 6 latent variables.

We chose 7 observed features and used RMGAN to fulfill the clustering and replacement. It can be seen from Fig. 4 that EM combining with RMGAN and P-EM reduce the execution time by 31.73% and 73.8% on average comparing with EM. It can also be seen in Figs. 5 and 6 that EM combining with RMGAN reduces almost 30% of execution time on average with 5 and 6 latent variables. This demonstrates that the proposed clustering method based on RMGAN improves the efficiency of parameter learning in BNML and P-EM. Moreover, we also made comparisons between P-EM combining with RMGAN and P-EM. As shown in Figs. 4, 5 and 6, P-EM combining with RMGAN reduces 74.79% of execution time on average compared with the EM in all conditions. It is shown that P-EM combining with RMGAN outperforms P-EM.

Feasibility. To evaluate the feasibility of BNML, we compared RMSE, MSE and MAE by different models on different sized MovieLens datasets, shown as Figs. 7, 8 and 9 and Table 1. The models include BNML using different methods to fulsill parameter learning, PNN and MF, in which the methods of parameter learning of BNML include P-EM with RMGAN, P-EM, EM, and EM with RMGAN. We then set the number of latent variables as 2 and chose the 2 observed features. 20% of the available dataset was considered as the test set and the remaining was used as training set. It can be seen from Table 1 that EM combining with RMGAN to finish parameter learning, on average, improves RMSE by 2.98%, MSE by 5.70%, and MAE by 3.39% respectively. This means the validity of clustering and the efficiency of RMGAN. As shown in Figs. 7, 8 and 9, the RMSE, MSE, and MAE based on BNML using P-EM to finish parameter learning are the same as those based on BNML using EM, which shows that P-EM achieves the same optimal values as EM.

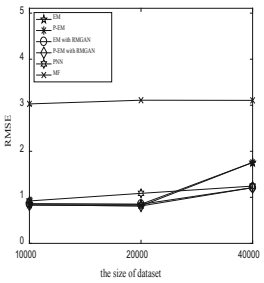

Fig. 7. RMSE on MovieLens.

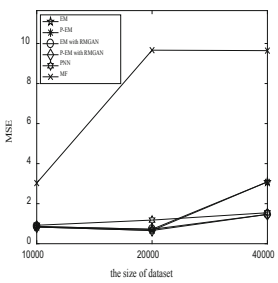

Fig. 8. MSE on MovieLens.

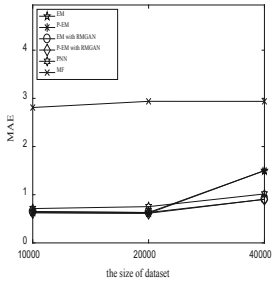

Fig. 9. MAE on MovieLens.

Table 1. Comparison of RMSE, MES and MAE for MovieLens.

	Size	Method	RMSE	MSE	MAE
BNML	10000	EM	0.85848	0.85848	0.64900
		EM with RMGAN	0.85848	0.85848	0.64900
		P-EM	**0.83366**	**0.83366**	**0.62700**
		P-EM with RMGAN	**0.83366**	**0.83366**	**0.62700**
	20000	EM	0.84897	0.72075	0.63075
		EM with RMGAN	**0.81319**	**0.72075**	**0.63075**
		P-EM	0.84897	0.66127	0.61489
		P-EM with RMGAN	**0.81319**	**0.66127**	**0.61489**
	40000	EM	1.75741	3.08850	1.50025
		EM with RMGAN	**1.21017**	**1.46450**	**0.90513**
		P-EM	1.75741	3.08850	1.50025
		P-EM with RMGAN	**1.21017**	**1.46450**	**0.90513**
PNN	10000	None	0.92168	0.92168	0.71250
	20000		1.08630	1.18000	0.75265
	40000		1.24358	1.54650	1.01600
MF	10000	None	3.02869	3.02869	2.81000
	20000		3.10913	9.66670	2.93820
	40000		3.10547	9.64397	2.93747

Additionally, we made comparisons between P-EM combining with RMGAN and P-EM, shown in Figs. 7, 8 and 9. The RMSE, MSE, and MAE for BNML using P-EM combining with RMGAN are lower than those for BNML using P-EM. This shows that P-EM combining with RMGAN outperforms P-EM in parameter learning. It can also be seen from Table 1 that BNML by our proposed method improves the performance of rating prediction by 15% and 80% comparing with PNN and MF respectively. Therefore,

these results suggest that BNML is an effective model for realistic applications such as rating prediction.

To further illustrate the feasibility of BNML, we compared the RMSE, MSE and MAE based on different models on different sized Cardiovascular datasets and presented the results in Figs. 10, 11 and 12 and Table 2. The models include BNML using different methods to fulfill parameter learning, and PNN, in which the methods of parameter learning of BNML are the same as the experiment on MovieLens.

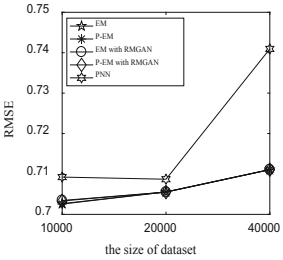

Fig. 10. RMSE on Cardiovascular.

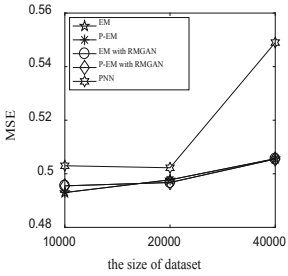

Fig. 11. MSE on Cardiovascular.

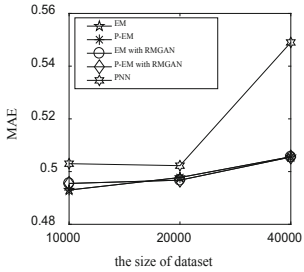

Fig. 12. MAE on Cardiovascular.

Table 2. Comparison of RMSE, MSE and MAE in Cardiovascular.

	Size	Method	RMSE	MSE	MAE
BNML	10000	**EM**	**0.70265**	**0.49300**	**0.49300**
		EM with RMGAN	0.70343	0.49550	0.49550
		P-EM	**0.70265**	**0.49300**	**0.49300**
		P-EM with RMGAN	0.70343	0.49550	0.49550
	20000	EM	0.70551	0.49775	0.49775
		EM with RMGAN	**0.70480**	**0.49675**	**0.49675**
		P-EM	0.70551	0.49775	0.49775
		P-EM with RMGAN	**0.70480**	**0.49675**	**0.49675**
	40000	**EM**	**0.71107**	**0.50562**	**0.50562**
		EM with RMGAN	**0.71107**	**0.50562**	**0.50562**
		P-EM	**0.71107**	**0.50562**	**0.50562**
		P-EM with RMGAN	**0.71107**	**0.50562**	**0.50562**
PNN	10000	None	0.70922	0.50300	0.50300
	20000		0.70869	0.50225	0.50225
	40000		0.74094	0.54900	0.54900

We first set one latent variable and chose 3 observed features to fulfill clustering. Since the features in MovieLens are discrete, we dropped the continuous features in

Cardiovascular. It can be seen from Table 2 that the performance of BNML using P-EM and RMGAN is, on average, improved by 0.6% in terms of RMSE, MSE and MAE. It can also be seen that our proposed method of parameter learning is effective on different types of datasets. Moreover, the BNML learnt by all methods in RMSE, MSE and MAE in Table 2 is, on average, improved by 0.7% comparing with PNN. By increasing the size of the dataset in Figs. 10, 11 and 12, BNML using different methods to finish parameter learning can achieve better RMSE, MSE and MAE comparing with PNN. This suggests that BNML is also feasible in Cardiovascular.

6 Conclusions and Future Work

We proposed a novel method based on MGAN, combining RNN and P-EM to accelerating the convergence of parameter learning of BNML. In our method BNML can be modeled for different types of applications with multiple latent variables. Incorporating the neural network into BNML is the main innovative property of our proposed method. Experimental results also demonstrated that the proposed method successfully achieves clustering and further improves the performance of BNML. There is however a disadvantage in selecting the clustering features. In our future research endeavors, we will explore a method based on GAN to address the feature selection problem. Another possible dimension for extending this work is to adopt Graph Neural Network to finish embedding instead of clustering.

Acknowledgments. This paper was supported by the National Natural Science Foundation of China (U1802271), the Science Foundation for Distinguished Young Scholars of Yunnan Province (2019FJ011), the Cultivation Project of Donglu Scholar of Yunnan University, and the Foundation for Undergraduates of Educational Department of Yunnan Province (2020Y0010).

References

1. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B (Methodol.) **39**(1), 1–38 (1977)
2. Basak, A., Brinster, I., Ma, X., Mengshoel, O.J.: Accelerating Bayesian network parameter learning using Hadoop and MapReduce. In: 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, pp. 101–108. ACM, Beijing (2012)
3. Becker, A., Geiger, D.: A sufficiently fast algorithm for finding close to optimal clique trees. Artif. Intell. **125**(1–2), 3–17 (2001)
4. Berlinet, A., Roland, C.: Parabolic acceleration of the EM algorithm. Stat. Comput. **19**(1), 35–47 (2009)
5. Binder, J., Koller, D., Russell, S., Kanazawa, K.: Adaptive probabilistic networks with hidden variables. Mach. Learn. **29**(2–3), 213–244 (1997)
6. Bradley, A., Van Der Meer, R.: Upfront surgery versus neoadjuvant therapy for resectable pancreatic cancer: systematic review and Bayesian network meta-analysis. Sci. Rep. **9**(1), 1–7 (2019)
7. De Campos, C.P., Ji, Q.: Improving Bayesian network parameter learning using constraints. In: 19th International Conference on Pattern Recognition, pp. 1–4. IEEE, Tampa (2008)

8. Feelders, A., Van der Gaag, L.C.: Learning Bayesian network parameters under order constraints. Int. J. Approx. Reason. **42**(1–2), 37–53 (2006)
9. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680. MIT Press, Montreal (2014)
10. Henderson, N.C., Varadhan, R.: Damped Anderson acceleration with restarts and monotonicity control for accelerating EM and EM-like algorithms. J. Comput. Graph. Stat. **28**(4), 834–846 (2019)
11. Jamshidian, M., Jennrich, R.I.: Acceleration of the EM algorithm by using quasi-Newton methods. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **59**(3), 569–587 (1997)
12. Jensen, F.V.: An Introduction to Bayesian Networks. UCL Press, London (1996)
13. Ji, Z., Xia, Q., Meng, G.: A review of parameter learning methods in Bayesian network. In: Huang, D.-S., Han, K. (eds.) ICIC 2015. LNCS (LNAI), vol. 9227, pp. 3–12. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22053-6_1
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, pp. 7–9. ICLR, San Diego (2015)
15. Klapper-Rybicka, M., Schraudolph, N.N., Schmidhuber, J.: Unsupervised learning in LSTM recurrent neural networks. In: Dorffner, G., Bischof, H., Hornik, K. (eds.) ICANN 2001. LNCS, vol. 2130, pp. 684–691. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44668-0_95
16. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
17. Langseth, H., Bangsø, O.: Parameter learning in object-oriented Bayesian networks. Ann. Math. Artif. Intell. **32**(1–4), 221–243 (2001)
18. Lauritzen, S.L.: The EM algorithm for graphical association models with missing data. Comput. Stat. Data Anal. **19**(2), 191–201 (1995)
19. Liao, W., Ji, Q.: Learning Bayesian network parameters under incomplete data with domain knowledge. Pattern Recogn. **42**(11), 3046–3056 (2009)
20. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern Recogn. **36**(2), 451–461 (2003)
21. Mukherjee, S., Asnani, H., Lin, E., Kannan, S.: Clustergan: latent space clustering in generative adversarial networks. In: 33th AAAI Conference on Artificial Intelligence, pp. 4610–4617. AAAI Press, Honolulu (2019)
22. Friedman, N.: The Bayesian structural EM algorithm. In: 14th Conference on Uncertainty in Artificial Intelligence (UAI 1998), pp. 129–138. Morgan Kaufmann Publishers Inc., San Francisco (1998)
23. Ono, C., Kurokawa, M., Motomura, Y., Asoh, H.: A context-aware movie preference model using a Bayesian network for recommendation and promotion. In: Conati, C., McCoy, K., Paliouras, G. (eds.) UM 2007. LNCS (LNAI), vol. 4511, pp. 247–257. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73078-1_28
24. Qu, Y., et al.: Product-based neural networks for user response prediction. In: 16th International Conference on Data Mining, pp. 1149–1154. IEEE, Barcelona (2016)
25. Reed, E., Mengshoel, O.J.: Bayesian network parameter learning using EM with parameter sharing. In: 11th UAI Bayesian Modeling Applications Workshop co-located with the 30th Conference on Uncertainty in Artificial Intelligence, pp. 48–59. CEUR-WS.org, Quebec (2014)
26. Reed, E.B., Mengshoel, O.J.: Scaling Bayesian network parameter learning with expectation maximization using mapreduce. Proc. Big Learn. Algorithms Syst. Tools **3**, 1 (2012)
27. Shafer, G.R., Shenoy, P.P.: Probability propagation. Ann. Math. Artif. Intell. **2**(1–4), 327–351 (1990)
28. Wikipedia Contributors Inner product space. https://en.wikipedia.org/wiki/Inner_product_space. Accessed 07 Feb 2020

29. Wikipedia Contributors Softmax function. https://en.wikipedia.org/wiki/Softmax_function. Accessed 16 Aug 2020
30. Yu, Y., Zhou, W.J.: Mixture of GANs for clustering. In: 27th International Joint Conference on Artificial Intelligence, pp. 3047–3053. IJCAI.org, Stockholm (2018)