Multiple DAGs Learning with Non-negative Matrix Factorization

Yun Zhou ZHOUYUN@NUDT.EDU.CN
Jiang Wang WANG1988JIANG@FOXMAIL.COM
Cheng Zhu ZHUCHENG@NUDT.EDU.CN
Weiming Zhang WMZHANG@NUDT.EDU.CN
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha (China)

Abstract

Probabilistic graphical models, e.g., Markov network and Bayesian network have been well studied in the past two decades. However, it is still difficult to learn a reliable network structure, especially with limited data. Recent works found multi-task learning can improve the robustness of the learned networks by leveraging data from related tasks. In this paper, we focus on the estimation of Direct Acyclic Graph (DAG) of Bayesian network. Most existing multi-task or transfer learning algorithms for Bayesian network use the DAG relatedness as an inductive bias in the optimization of multiple structures. More specifically, some works firstly find shared hidden structures among related tasks, and then treat them as the structure penalties in the learning step. However, current works omit the setting that the shared hidden structure comes from different parts of different DAGs. Thus, in this paper, the Non-negative Matrix Factorization (NMF) is employed to learn a parts-based representation to mediate this problem. Theoretically, we show the plausibility of our approach. Empirically, we show that compared to single task learning, multi-task learning is better able to positively identify true edges with synthetic data and real-world landmine data.

Keywords: Multiple DAGs learning, Structure penalties, Non-negative matrix factorization

1. Introduction

The discovery of direct acyclic graphs in Bayesian networks is of great interest in scientific domains such as medical risk analysis and risk assessment. The goal is to understand the relationships among causes and symptoms in a disease, such as the classical Asia network (Lauritzen and Spiegelhalter, 1988) which can be used to diagnose the risk of getting cancer or tuberculosis. However, the data collected is often done in several separate but related experiments. Therefore, the full dataset is actually composed of several distinct, but related, subsets of data - called tasks in multi-task learning(Caruana, 1997). For each task there may not be enough samples to learn a robust model. This problem can be ameliorated by introducing multi-task learning setting, which leverages information among tasks to smooth learned models. These smoothed models tend to be more robust to sample noise and generalize to holdout data better than models learned individually. Furthermore, in unsupervised learning, the set of models learned among tasks will share many structure patterns in common, easing interpretation of the models.

Algorithms for Bayesian network structure learning generally include two broad classes[1]: 1) score-based algorithm searches over the entire structure space to find a proper one that describes the observed data the most, it searches the small local changes to the learned graph structure (typically, the addition, removal or reversal of a single edge); 2) constraint-based algorithm uses constraints such as independence relations that we may know exist in the data, to reconstruct the structure. This paper only focus the multi-task setting of score-based algorithms. Firstly, this algorithm randomly generates initial structures. Then it defines quality metric (like AIC metric (Akaike, 1998), MDL metric (Bouckaert, 1993), Bayesian metric, K2 metric (Cooper and Herskovits, 1992), BDe metric (Heckerman et al., 1995) and etc.), which can be used to measure the quality of these candidate network structures that reflect the extent to which structure would fit the dataset. After that, the search algorithms (like K2 (Cooper and Herskovits, 1992), Hill climbing (Buntine, 1996), Repeated hill climbing, Max-Min hill climbing (Tsamardinos et al., 2006), Simulated annealing (Heckerman et al., 1995), Tabu search, Genetic search (Larrañaga et al., 1996) and etc.) are applied to identify the structure with the maximal score. For a detailed discussion and recent finding on structure heuristic search algorithms, please refer to the work (Fan and Yuan, 2015).

Niculescu-mizil and Caruana (2007) firstly introduced the multi-task setting in learning Bayesian network structures. For each two tasks, they penalize the edges existed in one structure but not in the other, and their algorithm searches the best configuration for all the tasks, which is NP-hard to find the global optimal. Because the key objective in Bayesian network multi-task learning algorithms is to implement mechanisms for learning the possible structure underlying the tasks. Finding this shared structure is important because it allows pooling information across the tasks, a property which is particularly appealing when there are many tasks but only few data per task. Thus, recent works introduce the ideas of task relatedness and shared hidden structures (Oyen and Lane, 2012; Oates et al., 2016), which are used as penalty terms for guiding the estimation of DAGs of different tasks(Figure 1a). However, the limitation is they cannot address the situation that the shared hidden structure comes from different parts of different DAGs (Figure 1b, the middle DAG consists of three parts, represented by black, blue and green colored nodes), and no previous works have addressed the problem that how to accurately use the corresponding part of the shared hidden structure as a penalty in different learning tasks.

The main challenge, therefore, is to incorporate different parts of the hidden structure for different learning tasks. The contribution of this paper is threefold. First, we give a novel formulation of multi-task DAGs discovery that uses the Non-negative Matrix Factorization (NMF) (Lee and Seung, 1999) to find the hidden factors and their weights to each task. Second, we further show that how to incorporate such hidden factors as the structure penalties in each DAG estimation. Third, we verify the superiority of our method compared with conventional DAG learning and state-of-the-art multi-task DAG learning algorithms on synthetic and real datasets. The experiments indicate that our method could improve

1. For detailed discussions we direct the reader to the books (Koller and Friedman, 2009; Barber, 2012).

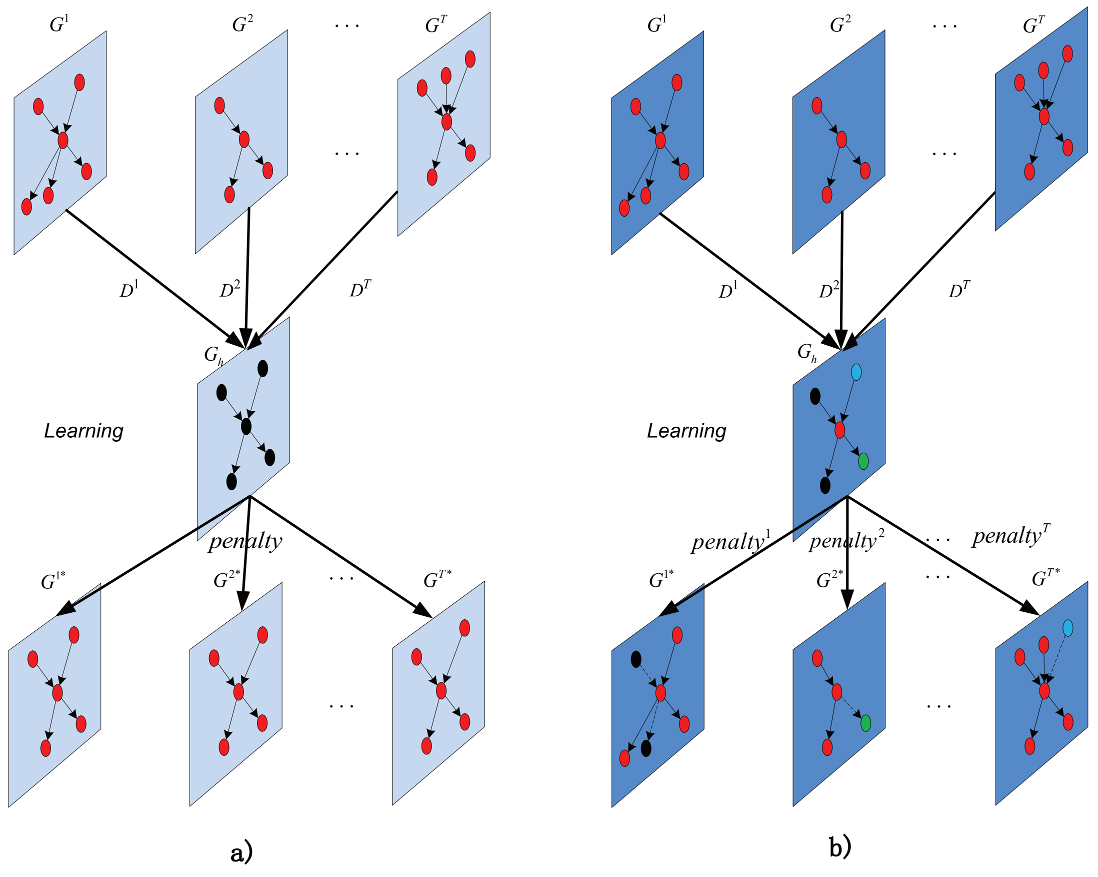


Figure 1: An illustrative example of DAGs $(G^1, G^2, ..., G^T)$ learning under two conditions: a) treat the shared hidden structure G_h as a single penalty; b) treat the part of the shared hidden structure as a penalty.

the learning performance with synthetic and real world data. This also highlight that our approach can be used for structure transfer learning.

2. Structure Learning of the Bayesian Network

In this section, we review the BN structure learning estimation problem (Cooper and Herskovits, 1992; Heckerman et al., 1995) and some previous extensions to multiple datasets/tasks (Oyen and Lane, 2012; Oates et al., 2016). We also summarize mathematical notations used throughout the paper in Table 1.

2.1 The Bayesian network model

Let θ denote a set of numerical parameters of the categorical random variables in some set U, and let G represent a Directed Acyclic Graph (DAG), whose nodes $X_1, X_2, X_3, ..., X_n$

Table 1: Mathematical notations used in this paper.

Index	Notation	Description
1	$G^t = (U^t, E^t)$	The t-th DAG contains random variable set U^t and edge set E^t.
2	$U^t = \{X_i^t\}_{i=1}^K$	The t-th variable set U^t contains K variables.
3	π_i	The set of parents of variable X_i in DAG G.
4	$A^t = \{a_{ij}^t\}_{i,j=1}^n$	A $n \times n$ adjacent matrix, whose entry $a_{ij}^t = 1$ represents an edge from X_i^t to X_j^t, and $a_{ij}^t = 0$ represents no edge between these two nodes.
5	$D^t = \{d_l^t\}_{l=1}^L$	The t-th dataset D^t contains L data records.

correspond to the random variables in V, and whose arcs represent the direct dependencies between these variables. Here $\theta = \{\theta_{ijk}\}$, and $\theta_{ijk} = p(X_i = k|\pi_i = j)$ represents a parameter for which X_i takes its k-th value and its parent set π_i takes its j-th value. As there is a one-to-one correspondence between nodes and variables, the terms 'node' and 'variable' are used interchangeably in this paper.

The Conditional Probability Table (CPT) associated with every variable contains the conditional probability of each value of the variable given each instantiation of its parents in G, which is also referred to as a CPT parameter θ_{ijk}. A CPT column[2] $p(X_i|\pi_i = j)$ denotes the discrete probability distribution of X_i given the j-th state configuration of its parents ($\pi_i = j$).

We call (G, θ) a Bayesian network if (G, θ) satisfies the *Local Markov condition* - a variable X_i is conditionally independent of its non-descendants given its parents π_i. Taking advantage of this property, one can obtain a factor representation of the joint probability distribution over all the random variables. That means a BN G encodes a simplified joint probability distribution over U given by:

$$p(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} p(X_i|\pi_i) \tag{1}$$

Structure learning is the task of automatically learning the DAG of a Bayesian Network given a dataset of observed cases, which is a NP hard to find the global optimum (Koller and Friedman, 2009).

2.2 Structure learning

Assuming global/local parameter independence, and parameter/structure modularity, the search-and-score approach (Maximum likelihood score) makes use of certain heuristics to find an optimal DAG that describes the observed data D^t the most over the entire space in the t-th task.

$$G^{t*} = \arg\max_{G^t \in A^t} \ell(G^t, \theta^t, D^t) \tag{2}$$

2. Note in some other works, e.g., Netica BN software, each CPT row represents a discrete probability distribution given a parent configuration.

Here, $\ell(G^t, \theta^t, D^t)$ is a log-likelihood of how the data fit the distribution given the structure. In a finite dataset, maximize the log-likelihood will result in a complete graph[3], which states that every pair of nodes is conditionally dependent.

To mediate this problem, the major scope of structure learning studies is how to avoid this unfavorable result from the maximum likelihood estimation and infer a sparse graph structure. Specifically, additional term is introduced, i.e. the total number of parameter values (P_{G^t}, Akaike's information criterion (AIC) metric (Akaike, 1998)) encoded in a DAG. Thus the resulting adjacent matrix has some zero entries owing to the effect of this regularization term. This estimation problem is defined as:

$$G^{t*} = \arg \max_{G^t \in A^t} \ell(G^t, \theta^t, D^t) - \eta \tag{3}$$

where η is the penalty term, $\eta = P_{G^t}$ and $\eta = \frac{P_{G^t}}{2} \cdot \log |D^t|$ in AIC metric and Minimum Description Length (MDL) metric (Bouckaert, 1993) respectively. By applying different hyperparameter (α, aka. equivalent sample size) settings in the Dirichlet prior distribution of each parameter in a BN, people introduced the K2 metric ($\alpha = 1$) (Cooper and Herskovits, 1992), BDe metric ($\alpha = |D^t| \cdot p(X_i^t = k|\pi_i^t = j)$) (Heckerman et al., 1995) and etc.

Taking advantage of regularization term is useful for improving the estimation performance of the resulting DAG. Moreover, if we know the hidden structure G_h^t of the task, which could be elicited from the domain expert, we can get the DAG that maximize the data log-likelihood and minimize its difference to G_h^t:

$$G^{t*} = \arg \max_{G^t \in A^t} \ell(G^t, \theta^t, D^t) - \eta - penalty(G^t, G_h^t) \tag{4}$$

where the $penalty(G^t, G_h^t)$ penalize the difference between the learned structure G and the hidden structure G_h^t.

3. Learning a set of DAGs with TRAM

The ordinary single task BN structure learning problem (equation 4) aims to learn one DAG from a single dataset. The natural extension of this framework to multiple datasets is introducing the similarity of tasks in DAGs learning. Previous work (Oyen and Lane, 2012) proposed the Task-Relatedness Aware Multi-task (TRAM) learning algorithm by incorporating the information sharing between different pairs of tasks. Specifically, this algorithm aims to estimate T structures and associated parameters $G^1, G^2, \ldots, G^T$ from T datasets $D^1, D^2, \ldots, D^T$ with additional penalty on task-relatedness $penalty(G^{1:t})$:

$$\begin{gathered} \max_{\{G^t \in A^t\}_{t=1}^T} \textstyle\sum_{t=1}^T (\ell(G^t, \theta^t, D^t) - \eta - penalty(G^{1:t})) \\ penalty(G^{1:t}) = I_{t \geq 2} \textstyle\prod_{j=1}^{t-1} \frac{1}{Z_{tj}} (1-\alpha)^{\Delta(G^t, G^j)} \end{gathered} \tag{5}$$

where Z_{tj} is the normalize term and $\Delta(\cdot)$ evaluate the task relatedness[4]. By default, $\Delta(\cdot)$ employs the graph edit distance, which is used to measure the minimum number of graph

3. According to the definition of the log-likelihood equation, adding an edge to a network will never reduce the log-likelihood score.
4. The task relatedness can also be set with empirical domain knowledge.

edit operations (insert and delete) to transform one graph to another. The $\alpha \in [0,1]$ is a positive parameter which balances the importance between the strength of fit to data and the bias toward similar DAGs. When $\alpha = 0$, the objective function is equivalent to learning the tasks independently. When $\alpha = 1$, identical learnt DAGs are the only feasible estimations. For each task, this algorithm explores all the learnt DAGs in previous tasks, and uses it to regularize the current estimation. The limitation is different learning order will produce different learning results. Additionally, in practice, the parameter of $\Delta(\cdot)$ needs to be tuned with specific domain knowledge.

4. Our Method

4.1 Learning a set of DAGs with a single or multiple hidden factor

To mediate this problem, we can introduce an EM-style framework, where T structures and associated parameters $G^1, G^2, \ldots, G^T$ are estimated in each iteration given the T datasets $D^1, D^2, \ldots, D^T$ and expected average DAG in the last iteration. The M step is defined as:

$$\max_{\{G^t \in A^t\}_{t=1}^T} \sum_{t=1}^{T} (\ell(G^t, \theta^t, D^t) - \eta - \Delta(G^t, G^t_h)) \tag{6}$$

where the G^t_h is the shared hidden structure over the entire tasks. In E step, the typical choice of the shared hidden DAG is $G^t_h = \frac{\sum_{t=1}^T \hat{G}^t}{T}$, where the $(\hat{G}^t\ )$ is estimated DAG of the t-th task in the last iteration. We refer to this problem (equation 6) as Multi-task Structure Learning with Single Hidden factor (MSL-SHF) in the remainder of the paper.

When introducing the shared hidden structure, we expect that there may exist more than one hidden structure among the tasks. Oates et al. (2016) considered the datasets collected from related but non-identical BNs whose DAGs may differ but are likely to share many features. Thus they discussed two cases of the hidden structure: known/unknown G^t_h. In the unknown case, they assume there are K hidden DAGs $G^t_{h1}, G^t_{h2}, \ldots, G^t_{hK}$, and an analogue of K-means clustering method is applied for finding the shared hidden DAGs. Thus, the entire estimation problem is reformulated as:

$$\max_{\{G^t \in A^t\}_{t=1}^T} \sum_{t=1}^{T} (\ell(G^t, \theta^t, D^t) - \eta - \Delta(G^t, G^t_{hk})) \tag{7}$$

where $G^t_{hk} = \arg\min_{G^t_{hk} \in \{G^t_{hk}\}_{k=1}^K} \Delta(G^t, G^t_{hk})$ is the closest hidden structure to G^t selected in t-th task. The difference between two DAGs is measured by graph edit distance. Similar to section 3, the G^t_k can be estimated as the average of the DAGs in the k-th cluster. We refer to this problem (equation 7) as Multi-task Structure Learning with Multiple Hidden factor (MSL-MHF) in the remainder of the paper.

4.2 Learning a set of DAGs with parts-based factors

In real world BNs construction, people usually combine some expert knowledge, e.g., BN idioms (Neil et al., 2000) and BN fragments (Laskey, 2008) to handcraft the final DAG. Follow this idea, in multi-task setting, there may also exist some parts-based hidden representations, which can be used to reconstruct each task. In order to verify this, we can introduce the NMF method (Lee and Seung, 1999), which is a matrix factorization algorithm distinguished from the other methods by its use of non-negativity constraints. These constraints lead to a parts-based representation because they allow only additive, not subtractive, combinations. For these reasons, the non-negativity constraints are compatible with the intuitive notion of combining parts to form a whole, which is how NMF learns a parts-based representation.

Given a set of estimated $\{G^t\}_{t=1}^T$, we convert each correspond adjacent matrix A^t into a column vector $v^t(n^2 \times 1)$. Thus, group the column vectors over all the tasks, we can have a original matrix $V = [v^1, v^2, \ldots, v^T] \in \Re^{n^2 \times T}$. NMF aims to find two non-negative matrices $W = [w_1, w_2, \ldots, w_K] \in \Re^{n^2 \times K}$ and $H = [h_1; h_2; \ldots; h_K] \in \Re^{K \times T}$ whose product can well approximate the original matrix V.

$$V \approx W \times H \tag{8}$$

Thus, each data vector v^t is approximated by a linear combination of the columns of W, weighted by the components of H, $v^t \approx \sum_{k=1}^K w_k h_k$. In reality, we have $K \ll n^2$ and $K \ll T$. Since relatively few basis vectors are used to represent many data vectors, a good approximation can only be achieved if the basis vectors discover structure that is latent in the data. Here, the entire multi-task estimation problem is defined as:

$$\begin{array}{c} \max_{\{G^t \in \Omega\}_{t=1}^T} \sum_{t=1}^T (\ell(G^t, \theta^t, D^t) - \eta - \Delta(G^t, f(G_h^t)) \\ f(G_h^t) = reshape(W \times W^{tr} \times v^t, n \times n) \end{array} \tag{9}$$

where the W^{tr} is the transpose of the matrix W. Here the original input v^t is firstly encoded by the hidden feature vector: $W^{tr} \times v^t$. Then a decoder reconstructs the input from the feature vector: $W \times W^{tr} \times v^t$. The final reconstructed input is used as the hidden structure in each estimation task.

5. Experiments

In this section, we mainly compare single task learning with multi-task learning in order to better understand their differences in terms of accuracy and robustness of the results.

5.1 Baseline methods and evaluation measurements

In the simulation, we adopt STL, MSL-TRAM, MSL-SHF and MSL-MHF as baseline methods to compare with MSL with non-negative matrix factorization (MSL-NMF). As the ground truth DAGs are known in all the tasks. The learning performance is measured by the graph edit distance between the learnt DAG and the original DAG in each task.

5.2 Learning results of the Asia network

We firstly briefly summarize the data generation procedure for our simulations. For the synthetic data, we need T different DAGs with same number of variables. We start from a standard Asia BN, and then randomly insert or delete one edge of this BN to make T new BNs and their correspond adjacent matrices $(A^1, A^2, \ldots, A^T)$. Meanwhile, the CPTs for the children of the inserted and deleted edges are updated by marginalizing over the deleted parents. Finally, we generate T datasets $D^1, D^2, \ldots, D^T$ from the corresponding CPTs using forwards sampling.

Table 2: Learning results of Asia network under different settings (*Samples*=200, 350 and 500; T=20 and 50), the learning performances are evaluated by the graph edit distance.

BN	Samples	STL	MSL-TRAM	MSL-SHF	MSL-MHF	MSL-NMF
Asia20	200	6.0±3.1	4.8±1.1	**3.4±1.3**	4.4±1.5	4.6±2.5
	350	4.5±1.9	4.3±1.0	**3.3±1.6**	4.0±1.5	3.9±1.4
	500	3.8±2.5	3.4±0.7	3.1±1.0	2.5±2.7	**1.8±1.5**
Asia50	200	5.8±2.8	5.2±1.0	3.9±2.1	**3.8±2.2**	4.8±2.1
	350	4.8±2.5	4.4±0.8	3.7±1.6	**3.0±2.0**	4.2±1.7
	500	4.4±2.9	3.5±0.7	3.6±1.6	3.9±1.7	**2.1±1.7**

We test two sizes of tasks: Asia20 ($T = 20$) and Asia50 ($T = 50$), and construct a dataset for each task by sampling 500 data samples from the correspond DAG. We then chose the first 200, 350 and 500 samples to create three data sizes. The edit distance in each estimation method is reported with the average value over the all tasks. The number of hidden factor is set as 2 ($K = 2$) in MSL-MHF and MSL-NMF. The relatedness parameter in MSL-TRAM is set as 0.5, $\alpha = 0.5$. We summarize the results in Table 2.

As shown in Table 2, the multi-task algorithms greatly improved the estimation performance compared with the conventional single task learning. This verifies the superiority of multi-task setting in the situation of multiple relevant datasets and limited data. In Asia network with 20 tasks, our EM-style multi-task DAG estimation algorithms always win the MSL-TRAM. Specifically, the MSL with single hidden factor has best performance in dataset sizes 200 and 350, and the MSL-NMF achieves the tremendous improvement compared with other algorithms in dataset size 500. However, with the increase of the total number of tasks (in $T = 50$), only the novel MSL-NMF constantly outperforms the MSL-TRAM in all data sizes. This verifies the robustness of the MSL-NMF.

5.3 Learning results of the real-world landmine problem

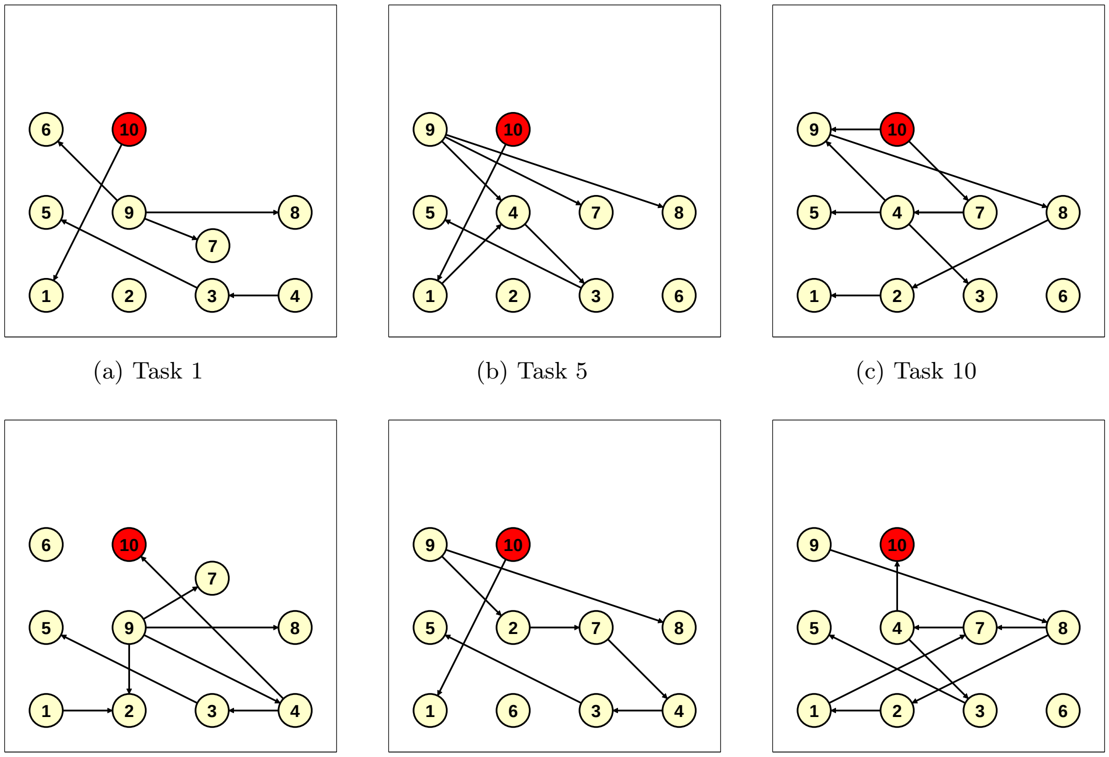
The landmine detection [5] is a binary classification problem, whose objective is to learn a classifier from the limited labelled actual synthetic-aperture radar data, with the goal of providing an accurate prediction of existence of landmines. There are 29 landmine fields and their correspond datasets in different sizes (shown in Table 3). Each data row is a 9-dimensional feature vector.

We treat the binary classification problem in each dataset as a BN learning task. Thus, we have 29 tasks with different data. We use the proposed MSL-NMF and baselines to learn 29 BNs with training data and infer the output of the landmine class (which is a binary node that 1 for landmine and 0 for clutter). To simplify the structure learning process, all 9 features are discretized into two values by a standard K-means algorithm. For each dataset of a task, we use half of data for training and half of data for testing.

5. `http://amll.pratt.duke.edu/research/landmine-detection`

Table 3: Number of data rows in each task for the Landmine problem.

Task ID	1	2	3	4	5	6	7	8	9	10
Number of data	690	690	689	508	509	509	510	511	508	509
Task ID	11	12	13	14	15	16	17	18	19	20
Number of data	689	688	508	510	507	445	449	448	449	449
Task ID	21	22	23	24	25	26	27	28	29	
Number of data	451	454	447	449	445	448	448	454	449	



(a) Task 1 (b) Task 5 (c) Task 10

(d) Task 15 (e) Task 20 (f) Task 25

Figure 2: DAGs learnt by the MSL-NMF method (due to the space limitation, only 6 of them are shown here).

To give an intuitive feel for the landmine problem, we present 6 DAGs (Task 1, 5, 10, 15, 20, 25), which are learnt from correspond training data by the MSL-NMF method. Details can be found in Figure 2. As we can see, each DAG contains 10 nodes, where the node with red color (*node* 10) is the landmine class node. Moreover, we can find some shared fragments/parts in these DAGs, e.g., "*node* 5 ← *node* 3 ← *node* 4" in DAGs 1, 5, 15, 20 and 25, "*node* 7 ← *node* 9 → *node* 8" in DAGs 1, 5, and 15, and isolated *node* 6 in DAGs 5, 10, 15, 20 and 25. This verifies the existence of shared parts among DAGs in different tasks.

Table 4: The average AUC value for 29 tasks in Landmine problem.

Dataset	STL	MSL-TRAM	MSL-SHF	MSL-MHF	MSL-NMF
Landmine	0.650±0.103	0.652±0.086	0.651±0.099	0.652±0.086	**0.657±0.091**

The classification performance is measured by the average AUC value over AUCs of all 29 tasks. As shown in Table 4, our MSL-NMF outperforms the baselines and state-of-art MSL-TRAM. Although the improvement is limited, it is nonetheless impressive given the difficulty of the problem.

6. Conclusion

When multiple datasets are existed, purely single task BN learning might be less accurate than learning with the help of other tasks. In this paper, we studied the new problem of DAGs multi-task estimation with parts-based factors. We proposed an EM-style learning framework for MSL-SHF and MSL-NMF, and examined their empirical performances at initial stage. Results show the proposed algorithms are effective at some point. However, more experiments on real world data are expected in the future to test the proposed method. Also, it would be interesting to extend this method to solve BN transfer learning problem.

7. Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 61703416 and 71571186) and MOE Research Center for Online Education (Grant No. 2017YB119).

References

Hirotugu Akaike. A Bayesian analysis of the minimum AIC procedure. In *Selected Papers of Hirotugu Akaike*, pages 275–280. Springer, 1998.

D. Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

Remco R Bouckaert. Probabilistic network construction using the minimum description length principle. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 41–48. Springer, 1993.

Wray L. Buntine. A guide to the literature on learning probabilistic networks from data. *Knowledge and Data Engineering, IEEE Transactions on*, 8(2):195–210, 1996.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Gregory F Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

Xiannian Fan and Changhe Yuan. An improved lower bound for bayesian network structure learning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 3526–3532, 2015.

David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques.* MIT Press, 2009.

Pedro Larrañaga, Mikel Poza, Yosu Yurramendi, Roberto H. Murga, and Cindy M. H. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):912–926, 1996.

Kathryn Blackmond Laskey. Mebn: A language for first-order bayesian knowledge bases. *Artificial intelligence*, 172(2-3):140–178, 2008.

Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Martin Neil, Norman Fenton, and Lars Nielson. Building large-scale Bayesian networks. *The Knowledge Engineering Review*, 15(03):257–284, 2000.

Alexandru Niculescu-mizil and Rich Caruana. Inductive transfer for Bayesian network structure learning. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pages 1–8, 2007.

Chris J Oates, Jim Q Smith, Sach Mukherjee, and James Cussens. Exact estimation of multiple directed acyclic graphs. *Statistics and Computing*, 4(26):797–811, 2016.

Diane Oyen and Terran Lane. Leveraging domain knowledge in multitask Bayesian network structure learning. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1091–1097, 2012.

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