Transfer Learning for Bayesian Networks

Roger Velásquez, L. Enrique Sucar, and Eduardo F. Morales

National Institute of Astrophysics, Optics and Electronics,
Computer Science Department,
Luis Enrique Erro 1, 72840 Tonantzintla, México
{roger,esucar,emorales}@inaoep.mx
http://ccc.inaoep.mx

Abstract. In several domains it is common to have data from different, but closely related problems. For instance, in manufacturing many products follow the same industrial process but with different conditions; or in industrial diagnosis, where there is equipment with similar specifications. In these cases, it is common to have plenty of data for some scenarios but very little for other. In order to learn accurate models for rare cases, it is desirable to use data and knowledge from similar cases; a technique known as "transfer learning". In this paper, we propose a transfer learning method for Bayesian networks, that considers both, structure and parameter learning. For structure learning, we use conditional independence tests, by combining measures from the target domain with those obtained from one or more auxiliary domains, using a weighted sum of the conditional independence measures. For parameter learning, we compared two techniques for probability aggregation that combine probabilities estimated from the target domain with those obtained from the auxiliary data. To validate our approach, we used three Bayesian networks models that are commonly used for evaluating learning techniques, and generated variants of each model by changing the structure as well as the parameters. We then learned one of the variants with a small data set and combined it with information from the other variants. The experimental results show a significant improvement in terms of structure and parameters when we transfer knowledge from similar problems.

1 Introduction

For many machine learning applications, it is assumed that a sufficiently large data set is available from which a reliable model can be induced. In some domains, however, it is difficult to gather enough data, for instance, in manufacturing some products are rarely produced. Experts, when confronted with a problem in a novel domain, use their experience from related domains to solve the problem. Recently, there has been an increasing interest in the machine learning community for using data from related domains, in particular when the available data is not enough [2,11,1], an approach known as *transfer learning*. However, this has not been explored for learning Bayesian networks. In this paper we propose transfer learning for Bayesian networks.

Learning a Bayesian network (BN) involves two main aspects: structure learning and parameter learning. Our structure learning algorithm combines the dependency measure obtained from data in the target domain, with those obtained from data in the auxiliary domains. We propose a combination function that takes into account the consistency between these measures, based on a variant of the PC algorithm [9].

The parameter learning algorithm uses an aggregation process, combining the parameters estimated from the target domain, with those estimated from the auxiliary data. Based on previous linear combination techniques, we propose two variants: (i) Distance–based linear pool (DBLP), which takes into account the distance of the auxiliary parameters to the target parameters, and (ii) Local linear pool (LoLP), which only includes auxiliary parameters that are *close* to the target one, weighted by the amount of data in each auxiliary domain. In the experiments we compare both methods, and also the basic linear pool as a baseline.

We evaluated experimentally the structure and parameter transfer learning techniques and compared the results vs. using only the data from the target domain. For this we considered 3 BNs commonly used for benchmarking learning algorithms (Alarm, Boblo and Insurance), and generated auxiliary models by changing the structure and parameters of the original model. We generated data from the original net and its variants, and then tested our method by combining these data sets. We evaluated both, the structure (in terms of edit distance) and parameters (in terms of the average square error), comparing the models obtained against the original model. Both aspects, structure and parameters, show a significant improvement when the amount of data for the target network is *small*. Also, the amount of improvement increases as the size of the auxiliary data increases. These results indicate that transfer learning can be a very useful approach for learning BNs.

The paper is structured as follows. Section 2 provides an overview of learning techniques for Bayesian networks. Section 3 describes relevant related work. Sections 4 and 5 introduce the structure and the parameter learning algorithms, respectively. The experiments and results are presented in Section 6. Finally, conclusions and future research directions are given in Section 7.

2 Learning Bayesian Networks

A Bayesian network (BN) [7] represents the joint distribution of a set of n (discrete) variables, $x_1, x_2, \ldots, x_n$, as a directed acyclic graph and a set of conditional probability tables (CPTs). Each node, that corresponds to a variable, has an associated CPT that contains the probability of each state of the variable given its parents in the graph. The structure of the network implies a set of conditional independence assertions, which give its power to this representation.

Learning a BN includes two aspects: learning the structure and learning the parameters. When the structure is known, parameter learning consists on estimating the CPTs from the data. For structure learning there are two main types of methods: (i) search and score, and (ii) conditional independence tests. The first class of methods [3] perform a heuristic search over the space of network structures, starting for some initial structure, and generating variation of the structure at each step. The *best* structure is selected based on a score that measures how well the model represents the data, common scores are BIC [3] and MDL [5]. The second class of methods are based on testing conditional independence between variables, and in this way adding or deleting arcs in the graph. A popular method of this type is the PC algorithm [9].

The PC algorithm starts from a fully connected undirected graph, and measures conditional independence of each pair of variables given some subset of the other variables, using the conditional cross entropy measure. If this measure is below a limit set according to certain confidence level, the arc between the pair of variables is eliminated. These tests are iterated until no more arcs can be eliminated. The direction of the remaining arcs are set based on conditional independence tests between variable triplets.

Both structure learning methods, including their variants, require *enough* data to produce accurate results. For instance, the PC algorithm assumes that there is sufficient data for accurate statistical tests; and the scores such as BIC and MDL require also a representative data set to provide good estimates.

3 Related Approaches

An alternative to compensate for the lack of data is to incorporate expert knowledge. A recent approach in this direction is proposed in [8], and considers the combination of multiple experts. They developed a Bayesian approach in which the knowledge of several experts is encoded as the prior distribution of possible structures, and the data is used to refine this structure and learn the parameters. The expert knowledge is represented as a "meta" Bayesian network, where the expert assertions are codified as the probability that an arc exists (and its direction) between two variables. In this way, the experts' knowledge is used to obtain a probability distribution over structures. The *best* structure is obtained by using a hill–climbing search method. Then, the parameters for this structure are estimated from data. In our work, we only use data and we do not include expert knowledge. In their work they combine several "weak" experts to obtain, in principle, a "strong" model; while we transfer knowledge (based on data) from a strong model(s) to improve a weak one.

Other work [6] considers the problem of learning Bayesian nets structures for related tasks. They consider that there are k data sets for related problems, and they propose a score and search method to learn simultaneously the BNs structures, one for each task. Assuming that the parameters of the different nets are independent, they define the joint probability distribution of the k structures, given the k data sets. The prior is defined such that it penalizes structures that are different for each other. Based on this score, they define a greedy search algorithms to find the best structures, where a new configuration is obtained by adding/deleting/reversing links for each structure. In contrast to this approach, our proposal is based on independence tests that are obtained separately for each data set, and then combined, resulting in a simpler and more efficient method. Again, the goal is different, we use data from one domain to improve the model in other, related domain; while they learn simultaneously the models for different problems. Our approach is divided into two phases, structure learning (Section 4) and parameter learning (Section 5), as described below.

4 Structure Learning

In this paper we propose an algorithm for structure learning of BN that incorporates information from auxiliary data sets in related domains, implementing a knowledge transfer approach based on dependency tests. The method can be seen as an extension of the PC algorithm for learning a BN when we have a small data set, and auxiliary data from related problems. Let us assume that there is a domain of interest with a *small* data set, D_0; and n larger data sets, $D_1, \ldots, D_n$, from auxiliary domains. The set of variables is the same for all the domains, $X_1, \ldots, X_m$. This is a reasonable assumption for manufacturing process, such as steel tube production, where each tube follows the same manufacturing process but with different furnace conditions or chemical components. As future work we will explore how to combine data with different, although similar, sets of variables.

Associated with the domain of interest there is a BN model, BN_0, with structure G_0 and parameters P_0. Then the problem is to obtain BN_0 from $D_0, D_1, \ldots, D_n$ so that it approximates the model that we will obtain if we had a *large* data set D_0. We start by learning the structure of the BN.

Following the PC algorithm we start with a fully connected undirected graph, and measure conditional independence values of each pair of variables given some subset of the other variables, using the conditional cross entropy measure. We obtain a set of measures for each pair of variables in the target domain, I_0, and in a similar way for each of the auxiliary domains, $I_1, \ldots, I_n$. Then, we combine these measures to build the structure for the target domain, G_0.

Before we see how we combine the results from the independency tests, we define a confidence measure for these tests. The cross entropy measure used in the PC algorithm depends on the size of the data set; it can be shown that the error of this test is asymptotically proportional to $logN/N$, where N is the size of the data set [10]. Based on this, we define the following function to estimate the confidence of the independency test between two variables X and Y given the conditioning set S:

$$\alpha(X, Y \mid S) = 1 - (logN/2 \times N) \times T \tag{1}$$

where $T = |X| \times |Y| \times |S|$. If the difference becomes negative, we set $\alpha = 0.005$. This function is proportional to the confidence of the test (inversely proportional to the size of the data set). We use this term to quantify the independency measures for the target and auxiliary data, before we combine them.

First we estimate how similar are the independency tests of each auxiliary data set with the target domain. The most similar domain is selected, say I_j. Then we combine the independence measures of this domain with those of the target domain using the following equation:

$$I_c(X, Y|S) = sgn(I_0(X, Y|S)) \times \alpha_0(X, Y|S) + w \times sgn(I_j(X, Y|S)) \times \alpha_j(X, Y|S) \tag{2}$$

where $sgn(I)$ is $+1$ if the independence test is positive (the variables are independent) and -1 otherwise. w is 1 if the result of the test is the same for the auxiliary data and the target data, 0.5 otherwise. Finally we use the results of the combined independency measure (I_c) to build the structure of the BN.

For efficiency reasons we currently restrict the conditioning set S to dimension one. Even with this restriction we obtain good results as described in the experiments. Once we have the structure of the model, we estimate the parameters also using the information from the auxiliary domains.

5 Parameter Learning

Given a Bayesian network structure, we need to specify all the conditional probability tables. The idea in this paper is to fill all the conditional probability tables using aggregation functions between several sources. There are several cases to consider when we want to combine conditional probabilities from different Bayesian networks:

- Combine CPTs that have the same variables, i.e., the same substructure. This is the simplest case and we do not need to do any transformation in the CPTs to apply an aggregation function.
- Combine CPTs with more parents in the auxiliary substructures. In this case we can marginalize over the additional variable to obtain the required conditional probability value in the target substructure. E.g., if we want to combine $P(X|Y, Z)$ of the target network with $P(X|Y, Z, W)$ of an auxiliary network, we can obtain $P(X|Y, Z)$ from the auxiliary substructure by marginalizing over W, i.e., $P(X|Y, Z) = \sum_i P(X|Y, Z, W_i)P(W_i)$.
- Combine CPTs with less parents in the auxiliary substructure. In this case, we can duplicate the CPTs of the auxiliary substructure for all the values of the additional variable in the target network.

Once we have a set of CPTs in terms of the CPTs of the target network, i.e., involving the same variables, we can proceed to combine them. There are several aggregation functions that have been proposed in the literature. Two commonly used functions are:

- Linear aggregation (lineal pool): a weighted sum of the different conditional probabilities, expressed as follows:

$$P(X) = k \times \sum_{i=1}^{n} w_i P_i(X)$$

where $P_i(X)$ represents the conditional probability of the i-th network involving X variables, w_i is the weight associated with that probability and k is a normalization factor.

– Logarithmic aggregation: a weighted product of the different conditional probabilities, expressed as follows:

$$P(X) = k \times \prod_{i=1}^{n} P_i(X)^{w_i}$$

The associated weight of the conditional probabilities, in our approach, depends on the confidence assigned to that probability. We proposed two novel aggregation methods: (i) using a weighted average and (ii) using a weighted average only over similar probabilities.

Our first aggregation method, called DBLP, involves the following steps:

1. Obtain the average probabilities of all the data sets:

$$\overline{P} = \frac{1}{n} \sum_{i=1}^{n} P_i$$

2. Obtain the minimum (d_{min}) and maximum (d_{min}) difference between the probability of the target data set and the above average.
3. Evaluate the new conditional probabilities of the target network as follows:

$$P_{target} = (1 - c_i) \times P_{target} + c_i \times \overline{P}$$

where the aggregation coefficients, c_i, express how much to consider from the CPTs of the other networks. The coefficients c_i basically express the following: if the CPTs of the target network are similar to the average of all the CPTs, then give more weight to the average, otherwise give more weight to the target CPT. This is expressed as follows:

$$c_i = (d_i - d_{min}) \times \left(\frac{c_{max} - c_{min}}{d_{max} - d_{min}} \right) + c_{min}$$

where c_{max} and c_{min} are parameters to indicate how close we want to consider the influence of the other CPTs. In our case we used $c_{max} = 0.75$ and $c_{min} = 0.25$.

The idea in our other aggregation function, called LoLP, is to use only the most similar or local probabilities and weight them according to the amount of data used in the target data set. This strategy has the following steps:

1. Obtain the average of the probabilities of the large data sets, but only between the most similar probabilities, in our case, those that are within the difference between the target probability and the overall average probability:

$$\overline{P}_{local} = \frac{1}{n} \sum_{i=1}^{n} P_i \ \forall P_i \text{ s.t. } P_i \in \{P_{target} \pm (P_{target} - \overline{P})\}$$

2. Obtain the new conditional probabilities of the target network as follows:

$$P_{target} = (1 - e_i) \times P_{target} + e_i \times \overline{P}_{local}$$

where e_i represents how much error we can expect from the CPTs.

$$e_i = \begin{cases} 1 - \frac{log(c_f)}{c_f} & \text{if } c_f \geq 3 \\ 1 - \frac{c_f \times log(3)}{3} & \text{if } c_f < 3 \end{cases}$$

where $c_f = \frac{N}{T_{i,j} \times 2}$ gives us a confidence value over the CPTs, that depends on $T_{i,j}$, the size of the CPT (number of possible values) and on N, the number of cases.

6 Experiments

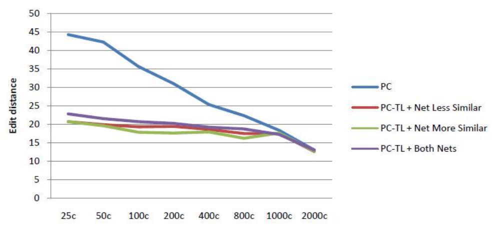
The aim of the experiments is to show that by considering additional data sets from related problems it is possible to improve the performance of a model constructed with a small data set.

We consider in the experiments three commonly used data sets for evaluating Bayesian networks, namely, *Alarm*, *Boblo* and *Insurance*. Table 1 provides a short description of them.

Table 1. Description of the data sets used in the experiments

Name	Description	Num. Attrib.
Alarm	Dataset in medical diagnosis for monitoring patients in intensive care units	37
Boblo	Data to identify cattle using blood type information	23
Insurance	Data used to classify insurance policies	27

We perform two main types of experiments to show the improvements in the structure/parameter learning algorithms when information from related problems is taken into account. The original data set is taken as our target network. We first build a Bayesian network with all the data using the PC algorithm as implemented in Elvira [4] with a 90% of significance value. We created new related problems by changing the original network and introducing Gaussian noise into the CPTs. We deleted links from the original network and altered the CPTs by introducing Gaussian noise with mean = 0 and different values of standard deviation. These altered networks are used as our auxiliary networks to try to reproduce the original network using only a subset of the original data. We created two related networks: (i) a similar network by randomly adding 5% of the links followed by randomly deleting 5% of the links and then introducing 5% of noise into the CPTs by multiplying them by a random noise, and (ii) a less similar network by adding and deleting 20% of the links and introducing 20% of



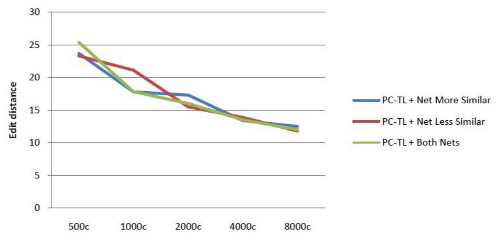


Fig. 1. Comparison between PC and PC-TL considering different number of cases for the Alarm data set with two auxiliary data sets. With increasing number of cases of the target network (left) and with increasing number of cases on the auxiliary networks (right).

noise into the CPTs. All the experiments were repeated 15 times and the figures show only the average values.

Figure 1 shows to the left, the behavior of the structure learning algorithm (PC-TL) considering auxiliary data sets against PC as more cases are considered from the the Alarm data set. The number of examples from the auxiliary data sets is fixed to 10,000. The figure on the right shows the behavior of the structure learning algorithm (PC-TL) considering auxiliary data sets against PC as more cases are considered from the auxiliary data set. The number of examples from the target data set is fixed to 100. Similarly the results for the Boblo data set are shown in Figure 2. Due to space limitations we do not present all the results from all the data sets, but they all show similar behavior.

From the figures it can be seen the expected behavior. The proposed approach performs better than PC, specially with small data sets, and they tend to converge to the same results as we increase the size of the training data set. Additional tests are required to observe how the algorithm degrades with very different data sets.

Similarly, we performed experiments for the parameter learning algorithms. Figure 3 shows a comparison between the errors in the CPTs from the conditional probabilities constructed only with the target data (Base), with the linear aggregation algorithm (LP) and with two proposed algorithms (DBLP and LoLP). The figure to the left shows the behavior of the algorithms when increasing the number of data of the target network. The figure to the right shows the behavior with an increasing number of data on the auxiliary data sets. Here, the number of examples from the target data set is fixed to 100.

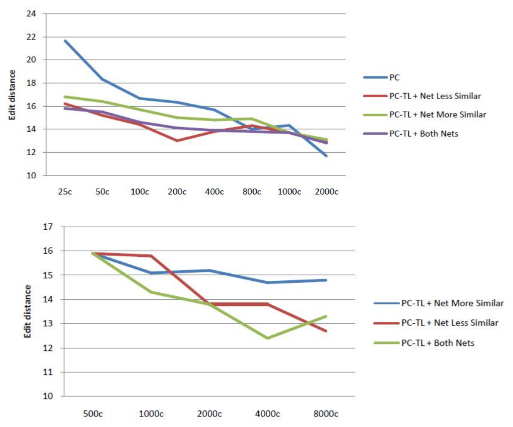


Fig. 2. Comparison between PC and PC-TL considering different number of cases for the Boblo data set with two auxiliary data sets. With increasing number of cases of the target network (left) and with increasing number of cases on the auxiliary networks (right).

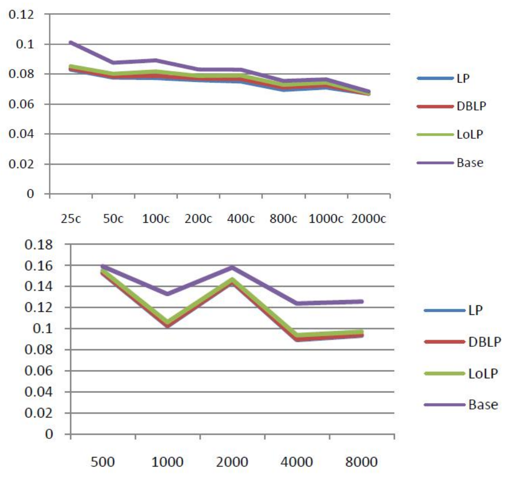


Fig. 3. Comparison between parameter learning considering different number of cases for the Alarm data set with two auxiliary data sets. Increasing the number of cases from the target data set (left) and increasing the number of cases of the auxiliary data sets (right).

From the figures we can see that in general the proposed algorithms perform better than the base case, so the use of additional data sets can also improve the parameters of the Bayesian networks. There is however, a very slight improvement over the lineal aggregation approach. As future work we will study more carefully the proposed aggregation functions to try to obtain bigger improvements.

7 Conclusions and Future Work

In this paper we have presented three algorithms for transfer learning in Bayesian networks, one for structure learning and two for parameter learning. The idea behind this research is to use information from related data sets in order to improve the performance of networks constructed with small data sets. This is common is several domains, where there can be rare cases, that nevertheless share some similarities with more common cases. From the experiments it is shown that using additional information can improve the accuracy of the constructed model. As future work, we plan to apply this technique to a manufacturing problem that originate the ideas developed in this work. We will also study the effect of increasingly dissimilar databases in the performance of the algorithms.

Acknowledgments

This work was supported in part by CONACyT, project 47968.

References

1. Baxter, J.: A bayesian/information theoretic model of learning to learn via multiple task sampling. Machine Learning 28(1), 7–39 (1997)
2. Caruana, R.: Multitask learning. Machine Learning 28(1), 41–75 (1997)
3. Cooper, G., Herskovitz, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning 9(4), 309–348 (1992)
4. Elvira: Elvira: An environment for creating and using probabilistic graphical models. In: Gámez, J.A., Salmerón, A. (eds.) First European Workshop on Probabilistic Graphical Models (2002)
5. Lam, W., Bacchus, F.: Learning bayesian belief networks: An approach based on the mdl principle. Computational Intelligence 10, 269–293 (1994)
6. Marina, M., Shen, X. (eds.): Inductive Transfer for Bayesian Network Structure Learning, vol. 2 (2007)
7. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, San Francisco (1988)
8. Richardson, M., Domingos, P.: Learning with knowledge from multiple experts. In: Fawcett, T., Mishra, N. (eds.) Proc. of the Twentieth Intl. Machine Learning Conf (ICML 2003), pp. 624–631. AAAI Press, Menlo Park (2003)
9. Spirtes, P., Glymour, C., Scheines, R.: Causation, prediction, and search. Springer, Berlin (1993)
10. Su, J., Zhang, H.: Full bayesian network classifiers. In: Cohen, W.W., Moore, A. (eds.) Proc. Twenty-Third Intl. Machine Lerning Conference (ICML 2006), vol. 148, pp. 897–904. ACM, New York (2006)
11. Thrun, S.: Is learning the n-th thing any easier than learning the first? In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems, vol. 8, pp. 640–646. The MIT Press, Cambridge (1996)