Mach Learn
DOI 10.1007/s10994-009-5160-4

Inductive transfer for learning Bayesian networks

Roger Luis · L. Enrique Sucar · Eduardo F. Morales

Received: 28 February 2009 / Revised: 9 October 2009 / Accepted: 22 October 2009

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 others. 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 an inductive 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 task with those obtained from one or more auxiliary tasks, using a novel weighted sum of the conditional independence measures. For parameter learning, we propose two variants of the linear pool for probability aggregation, combining the probability estimates from the target task with those from the auxiliary tasks. 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 dataset 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 tasks. We also evaluated the method with real-world data from a manufacturing process considering several products, obtaining an improvement in terms of log-likelihood between the data and the model when we do transfer learning from related products.

Keywords Inductive transfer · Bayesian networks · Structure learning · Parameter learning

Editors: Nicolo Cesa-Bianchi, David R. Hardoon, and Gayle Leen.

R. Luis · L.E. Sucar (✉) · E.F. Morales
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Luis Enrique Erro 1,
Sta. Ma. Tonantzintla, Puebla, Mexico
e-mail: esucar@inaoep.mx

Published online: 22 December 2009

1 Introduction

For many machine learning applications, it is assumed that a sufficiently large dataset 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 that are rarely produced or in medicine where there are some rare diseases. Experts, when confronted with a problem in a novel task, use their experience from related tasks to solve the problem. Recently, there has been an increasing interest in the machine learning community for using data from related tasks, in particular when the available data is scarce (Thrun 1996; Caruana 1997; Baxter 1997; Silver et al. 2008), an approach known as *transfer learning*. In many domains it is common to have uncertainty and Bayesian networks have proved to be an adequate and powerful technique to deal with it; however, there is limited previous work on transfer learning for Bayesian networks (BNs). Previous work considers learning simultaneously multiples tasks by combining data (Niculescu-Mizil and Caruana 2007) or expert knowledge (Richardson and Domingos 2003). Although our work is related to these methods, the main focus is different; we are interested on learning a model for a task with limited data, taking advantage of data from related tasks. Another important difference is that our method is based on independence tests, while previous approaches are based on search and score techniques. The learning methods based on independence tests tend to degrade more severely when there is not enough data, so in this case transfer learning has a greater impact.

In this paper we propose a transfer learning method for Bayesian networks that induces a model for a *target* task, from data from this task and from other related *auxiliary* tasks. The method includes both, structure and parameter learning. The structure learning method is based on the PC algorithm (Spirtes et al. 1993), and it combines the dependency measures obtained from data in the target task, with those obtained from data in the auxiliary tasks. We propose a combination function that takes into account the reliability and consistency between these measures.

The parameter learning algorithm uses an aggregation process, combining the parameters estimated from the target task, 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 task. In the experiments we compare both methods, and also the basic linear pool as baseline.

We evaluated experimentally the structure and parameter transfer learning techniques and compared the results versus using only the data from the target task. We performed two sets of experiments. In the first set we evaluated how well it can recover the structure and the parameters of a BN when we know the original model. For this we consider three commonly used BNs for benchmarking learning algorithms (ALARM in Beinlich et al. 1989, BOBLO in Rasmussen 1992 and INSURANCE in Binder et al. 1997), 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 datasets. We performed tests varying the amount of data for the target model and from the auxiliary tasks. We also performed tests with different auxiliary datasets obtained from auxiliary networks with different degrees of edit distances. We evaluated the structure in terms of edit distance, and both, structure and parameters, in terms of the average squared 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.

An important observation is that the results improve as we increase the number of auxiliary tasks, even if some of these are not very closely related to the target task. We also performed tests between our proposed method and the PC algorithm using the data from the target task concatenated with the data from the auxiliary tasks. In this case the PC algorithm with all the available data is only competitive to our approach when all the datasets are fairly similar.

In the second set of experiments we evaluated the transfer learning method with data from a manufacturing process, a complex real-world problem that motivated this work. We learned a model for the process variables of a product with relatively few data, and used, as auxiliary tasks, data from other similar products. In this case the evaluation was made in terms of the log-likelihood between the model and the data, as we do not have a reference BN. In general, the results show an improvement when we use auxiliary data with our approach; while if we simply concatenate the data of all the related domains and apply the PC algorithm, the quality in terms of log-likelihood decreases.

The results from both sets of experiments show that transfer learning is a very useful approach for learning BNs when we can incorporate data from related tasks, in particular when we have limited data. The proposed methods take advantage of data from related tasks to improve the structure and parameters of the target model, even if the auxiliary tasks are not so similar to the target.

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

2 Learning Bayesian networks

A Bayesian network (BN) (Pearl 1988) 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 gives 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 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 (Cooper and Herskovits 1992; Lam and Bacchus 1994) performs a heuristic search over the space of network structures, starting from 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 (Cooper and Herskovits 1992) and MDL (Lam and Bacchus 1994). The second class of methods are based on testing conditional independence between variables, and in this way adding or deleting arcs in the graph. The most well known variant of this approach is the PC algorithm (Spirtes et al. 1993).

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 dataset to provide good estimates.

Given that our structure learning method is based on the PC algorithm, we next give a brief overview of the basic algorithm.

Algorithm 1 The PC algorithm

Require: Set of variables $\mathbf{X}$, Independence test I
Ensure: Directed Acyclic Graph G

```
Initialize a complete undirected graph G'
i = 0
repeat
  for X ∈ X do
    for Y ∈ ADJ(X) do
      for S ⊆ ADJ(X) − {Y}, |S| = i do
        if I(X, Y | S) then
          Remove the edge X − Y from G'
        end if
      end for
    end for
  end for
  i = i + 1
until |ADJ(X)| ≤ i, ∀X
Orient edges in G'
Return G
```

2.1 The PC algorithm

The PC algorithm (Spirtes et al. 1993) first recovers the skeleton (underlying undirected graph) of the BN, and then it determines the orientation of the edges.

To determine the skeleton, it starts from a fully connected undirected graph, and determines the conditional independence of each pair of variables given some subset of the other variables. For this it assumes that there is a procedure that can determine if two variables, X, Y, are independent given a subset of variables, $\mathbf{S}$, that is, $I(X, Y \mid \mathbf{S})$. An alternative for this procedure is the conditional cross entropy measure. If this measure is below a threshold value set according to certain confidence level, the edge between the pair of variables is eliminated. These tests are iterated for all pairs of variables in the graph.

In the second phase the direction of the edges are set based on conditional independence tests between variable triplets. It proceeds by looking for substructures in the graph of the form $X - Z - Y$ such that there is no edge $X - Y$. If X, Y are not independent given Z, it orients the edges creating a V-structure $X \rightarrow Z \leftarrow Y$. Once all the V-structures are found, it tries to orient the other edges based on independence tests and avoiding cycles. Algorithm 1 summarizes the basic procedure.[1]

If the set of independencies are faithful to a graph and the independence tests are perfect, the algorithm produces a graph equivalent to the original one (Spirtes et al. 1993). However, the statistical tests for independence based on sample data have errors, and the number of errors increases when the sample is small. In this work we propose to incorporate data from related tasks to compensate for the lack of data in the task of interest.

[1] $ADJ(X)$ is the set of nodes adjacent to X in the graph.

3 Related approaches

Recently there has been an increasing interest in inductive transfer learning for different types of representations (Thrun 1996; Caruana 1997; Baxter 1997; Silver et al. 2008). One of the earliest approaches for transfer learning is described in Wu and Dietterich (2004). Here the authors proposed the use of auxiliary data to improve the accuracy of training data in the problem of classifying images of tree leaves. They introduced two approaches, one based on k-nearest neighbors and another in support vector machines (SVM). For SVM the auxiliary data was used as potential support vectors, as constraints in the optimization problem or both, with promising results. In the rest of this section we will focus on the related work for learning Bayesian networks.

In Dai et al. (2007) the authors proposed a transfer learning algorithm for text classification. The idea is to estimate the initial probabilities of a naïve Bayes network using a labeled dataset, and then use an EM-based algorithm to revise the model using unlabeled data from a different distribution. They use a metric based on the Kullback–Leibler divergence measure to estimate the relevance between the prior distributions. Their work is based on naïve Bayes classifiers, while we are interested in transfer learning for general Bayesian networks and, consequently, we are also interested in structural transfer learning.

Roy and Kaelbling (2007) developed an alternative method for transfer learning for the naïve Bayes classifier. The basic idea is to partition the dataset in a number of clusters, such that the data for each cluster for all tasks has the same distribution. They train one classifier for each partition, which are combined using a Dirichlet process. They test this approach in a multi-task meeting classification problem. As in the previous case, this technique is restricted to Bayesian classifiers, and in particular it does not address the problem of structure learning.

One alternative to compensate for the lack of data in a task is to incorporate expert knowledge. A recent approach in this direction is proposed in Richardson and Domingos (2003), which considers the combination of multiple experts. They develop 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 using a hill-climbing search method. Then, the parameters for this structure are estimated from data. In our work, we only use data and do not include expert knowledge. In their work, they combine several *weak* models to obtain, in principle, a *strong* model; while we transfer knowledge (based on data) from a strong model(s) to improve a weak one. Although knowledge transfer from strong to weak tasks has been done before for other learning techniques (Caruana 1997; Silver et al. 2008), this is not the case for learning Bayesian networks.

Other work (Niculescu-Mizil and Caruana 2007) considers the problem of learning Bayesian network 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 datasets. The prior is defined such that it penalizes structures that are different for each other. Based on this score, they define a greedy search algorithm to find the best structures, where a new configuration is obtained by adding/deleting/reversing arcs for each structure. In contrast to this approach, our proposal is based on independence tests that are obtained

separately for each dataset, and then combined, resulting in a simpler method. Again, the goal is different, we use data from one task to improve the model in another, related task; while they learn simultaneously the models for different problems.

Previous approaches for inductive transfer for learning BNs consider the combination of data or expert knowledge to learn simultaneously several related tasks, and are based on global score to compare alternative models. In contrast, our approach solves a different although related problem, to learn a model for a task with limited data, using additional data from related tasks; and is based on local independence tests. The method is divided into two phases, structure learning (Sect. 4) and parameter learning (Sect. 5), as described below.

4 Structure learning

In this paper we propose an algorithm for structure learning of BNs that incorporates information from auxiliary datasets in related tasks, implementing a knowledge transfer approach based on independence tests. The method can be seen as an extension of the PC algorithm for learning a BN when we have a small dataset, and auxiliary data from related problems.

Let us assume that we have a target task represented by a set of n discrete random variables, $\mathbf{X_0} = X_1, X_2, \ldots, X_n$, and m auxiliary tasks represented by their corresponding set of discrete random variables, $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_m}$. We assume that these sets of variables are the same for all tasks.[2] We consider that associated to the target task there is a *small* dataset, D_0; and m additional datasets, $D_1, \ldots, D_m$, for the auxiliary tasks, that are sufficiently large for learning BNs models for $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_m}$ to the desired level of accuracy.

The objective is to build a Bayesian network model for the target task, BN_0, with structure G_0 and parameters P_0 from $D_0, D_1, \ldots, D_m$, which approximates the model that we will obtain if we had a *large* dataset for the target task, D_0. We start by learning the structure of the model.

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

The transfer structure learning algorithm (*PC-TL*) follows basically the PC algorithm, the main difference is the way the independence tests are evaluated. For each pair of variables, X, Y, the independence measure is a linear combination of the estimation from the target task with that of the *closest* auxiliary task, where closest is determined in terms of a local and global similarity measure. In this linear combination each term is weighted by a factor that defines a *confidence* measure for the tests depending on the data. A general outline of the method is given in Algorithm 2, and below we detail the combination function and the associated measures.

4.1 Confidence measure

Before we see how we combine the results from the independence tests, we define a confidence measure for these tests. The cross entropy measure used in the PC algorithm depends

[2]The method can be directly extended to include auxiliary tasks with different variables, as long as there is a common subset of variables with the target task.

Algorithm 2 The PC-TL algorithm

Require: Set of variables **X**
Require: Target data D_0; and auxiliary data $D_1, \ldots, D_m$
Require: Independence tests: target task, I_0, and auxiliary tasks, $I_1, \ldots, I_m$
Ensure: Directed Acyclic Graph G

```
Initialize a complete undirected graph G'
i = 0
repeat
    for X ∈ X do
        for Y ∈ ADJ(X) do
            for S ⊆ ADJ(X) − {Y}, |S| = i do
                Find the most similar auxiliary task, k, and its similarity measure Sk_XY
                Determine the confidence measures α(X, Y | S) for target and auxiliary tasks
                Obtain the combined independence measure I_F(X, Y | S)
                if I_F(X, Y | S) then
                    Remove the edge X − Y from G'
                end if
            end for
        end for
    end for
    i = i + 1
until |ADJ(X)| ≤ i, ∀X
Orient edges in G'
Return G
```

on the size of the dataset; it can empirically be shown that the error of this test is asymptotically proportional to $\frac{\log N}{2N}$, where N is the size of the dataset (Friedman and Yakhini 1996). 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:

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

where $T = |X| \times |Y| \times |S|$, $|X|$ is the size of X, $|Y|$ of Y, and $|S|$ of 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 dataset). We use this term to quantify the independence measures for the target and auxiliary data, before we combine them.

4.2 Similarity measure

In general we want to transfer information from tasks that are closer to the target task, so we define a *similarity measure* that considers both, global and local similarity. This similarity measure is defined in terms of the number of shared independencies between the target and auxiliary tasks. The global similarity considers all pair-wise conditional independencies of the form $I(X, Y \mid S)$ in the models, while the local similarity includes only those for a specific pair of variables. These measures are based on the PC algorithm, as the similarity measure can not be seen independent of the learning technique (Thrun 1996, p. 2). That is, given that the basic learning algorithm is based on independence tests, it seems appropriate

to use the same type of tests for measuring similarity. This makes also the process more efficient, as these tests have been already done at least for the target task.

The global similarity measure is defined as:

$$Sg_{Dj} = dep_j + ind_j \tag{2}$$

where dep_j is the number of common conditional dependencies between all pairs of variables in the target task and the j auxiliary task, and ind_j is the number of common conditional independencies between all pairs of variables in the target task and the j auxiliary task. The conditional (in)dependency tests, $I(X, Y \mid S)$, are obtained from the target or auxiliary datasets using the cross entropy measure, using certain threshold.[3]

The local similarity measure for the variables X, Y given a conditioning subset S is defined as:

$$Sl_{Dj}(X, Y) = \begin{cases} 1.0, & \text{if } I_0(X, Y \mid S) = I_{Dj}(X, Y \mid S) \\ 0.5, & \text{if } I_0(X, Y \mid S) \neq I_{Dj}(X, Y \mid S) \end{cases} \tag{3}$$

where $I_0(X, Y \mid S)$ is the result of the independence test in the target task, and $I_{Dj}(X, Y \mid S)$ is the result of the test in the j auxiliary task. The constants in this equation (1.0 and 0.5) give a different weight to auxiliary structures that have the same or different local structure, giving more weight to similar structures. Different influence of the auxiliary tasks over the target task can be obtained by changing the constants in the SL_{Dj} function. We experimented with different values of these parameters, see Sect. 6, and in general there is not a significant difference in the performance; for all other experiments we used the above values.

Thus, for a pair of variables X, Y and conditioning set S, the combined similarity measure for the auxiliary task j, Dj_{XY}, is:

$$Sj_{XY} = Sg_{Dj} \times Sl_{Dj}(X, Y) \tag{4}$$

4.3 Combination function

Based on the similarity measure, for each pair of variables X, Y the *closest* auxiliary task is combined with the target task to obtain the combined independence measure. The independence measure transferred from the most similar auxiliary task k is weighted by the combined similarity measure, which is:

$$Sk^*_{XY} = Sg_{Dk} \times Sl_{Dk}(X, Y) \tag{5}$$

So the combined independence measure $I_F(X, Y \mid S)$ is a linear weighted combination of the independence measures in the target and auxiliary tasks, considering the confidence and similarity measures:

$$\begin{aligned} I_F(X, Y \mid S) = {} & \alpha_0(X, Y \mid S) \times \operatorname{sgn}(I_0(X, Y \mid S)) \\ & + Sk^*_{XY} \left(\alpha_{D_{XY}}(X, Y \mid S) \times \operatorname{sgn}(I_{D_{XY}}(X, Y \mid S))\right) \end{aligned} \tag{6}$$

where $\operatorname{sgn}(I)$ is $+1$ if the independence test is positive (X, Y are independent given S) and -1 otherwise. $\alpha_0(X, Y \mid S)$ is the confidence measure of the target task, and $\alpha_{D_{XY}}(X, Y \mid S)$

[3]The (in)dependency tests can be performed with a threshold value. In this work, we use α, defined in (1), as our significance threshold as it allows to obtain higher confidence when more data is available.

is the confidence measure for the most similar auxiliary task, both for the variable pair $\{X, Y\}$ conditioned on S.

In this way we do transfer learning via the independence measures, considering the similarity of the auxiliary tasks, and the confidence of the independence tests. Note that the global similarity measure needs to be estimated only once and then used for all the local tests.

Once we have the structure of the model, we estimate the parameters also using the information from the auxiliary tasks.

5 Parameter learning

Given a Bayesian network structure, to complete the model the conditional probability tables (CPTs) for each variable given its parents need to be estimated. Again, accurate estimates can be obtained from large datasets, however, poor estimates are normally obtained with small datasets. The idea in this paper is to fill all the conditional probability tables using aggregation functions between several sources, using transfer learning from the auxiliary tasks.

To combine the CPT for a variable X from an auxiliary task j, it is necessary that the variable has the same parents in the target task, that is $Pa_j(X) = Pa_0(X)$, where $Pa_i(X)$ is the set of parents of X in network i.[4] If they do not have the same parent set, we transform the substructure in the auxiliary task to match the target one. For this, there are several cases to consider:

1. Combine CPTs that have the same variables, i.e., the same substructure. This is the simplest case and no transformation is required to apply an aggregation function.
2. Combine CPTs with more parents in the auxiliary substructures. In this case we marginalize over the additional variable(s) to obtain the required substructure. E.g., if we want to combine $P(X \mid Y, Z)$ of the target network with $P(X \mid Y, Z, W)$ of an auxiliary network, we can obtain $P(X \mid Y, Z)$ from the auxiliary substructure by marginalizing over W, i.e., $P(X \mid Y, Z) = \sum_i P(X \mid Y, Z, W_i)P(W_i)$.
3. Combine CPTs with less parents in the auxiliary substructure. In this case, we duplicate the CPTs of the auxiliary substructure for all the values of the additional variable(s) in the target network. E.g., if we want to combine $P(X \mid Y, Z)$ of the target network with $P(X \mid Y)$ of an auxiliary network, we can obtain $P(X \mid Y, Z)$ from the auxiliary substructure by duplicating $P(X \mid Y)$ for all values of Z, i.e. $P(X \mid Y, Z_j) = P(X \mid Y), \forall j$.[5]
4. Combine CPTs with additional and missing variables in the auxiliary substructures (case 2 and 3 combined). In this case we first marginalize over the additional variable(s) and then duplicate over the missing variable(s).

The previous procedure is not necessary if we have data for all the tasks, in this case we can estimate the CPTs directly from the data. However, we consider that in some cases we might have only the models for some tasks. Note that in the case of structure learning it is also not necessary to have access to the data, only the sufficient statistics to obtain the independence measures.

[4] We assume that all the variables are discrete and have the same nominal values, or intervals if they were discretized.

[5] By duplicating we mean just repeating the probabilities, for example, if Z is binary and $P(X_1 \mid Y_1) = 0.7$, this value will be replicated for both values of Z: $P(X_1 \mid Y_1, Z_1) = 0.7$ and $P(X_1 \mid Y_1, Z_2) = 0.7$.

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 Genest and Zidek (1986), Chang and Chen (1996), Chen et al. (1996). Two commonly used functions are:

– Linear aggregation (lineal pool), also known as weighted mean. The consensus probability, $P(X)$, is a weighted sum of the probabilities from the target and auxiliary tasks, 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 model involving X, w_i is the weight associated with that probability and k is a normalization factor.

– Logarithmic aggregation. The consensus probability, $P(X)$, is a weighted geometric mean (weighted product) of the probabilities from the target and auxiliary tasks, expressed as follows:

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

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

We proposed two novel aggregation methods: (i) a distance based weighted average and (ii) a weighted average only over similar probabilities.

5.1 Distance based linear pool (DBLP)

Our first aggregation method, called DBLP, considers the confidence of the probability estimates from the auxiliary tasks, and the similarity with the target estimates. The idea is to give a higher weight to the probability estimates that have a higher confidence and that are more similar to the target. For this we first obtain a weighted average of the probabilities estimated from the data of the auxiliary tasks, where the weight is based on the size of each dataset. Then we combine this average to the target probability estimate, weighted by a factor that depends on its similarity to the target probability.

DBLP involves the following steps:[6]

1. Obtain the average probabilities of all the datasets discounted according to a confidence factor:

$$\overline{p} = k \sum_{i=1}^{n} (f_i \times p_i) \tag{7}$$

where f_i is a confidence factor for each probability estimate and k is a normalization factor. The confidence factor depends on the size of the dataset used to estimate it and it

[6] In the following descriptions of the proposed aggregation algorithms, $p = P(X)$ denotes each individual probability in the CPTs.

is defined as follows:

$$f_i = \begin{cases} 1 - \frac{\log(c_f)}{c_f}, & \text{if } c_f \geq 3 \\ 1 - \frac{c_f \times \log(3)}{3}, & \text{if } c_f < 3 \end{cases} \tag{8}$$

where $c_f = \frac{N}{T \times 2}$ is proportional to the expected error that depends on T, the number of entries in the CPT, and N is the number of cases in the data base. Since function f_i is not continuous in the range of interest, it is composed of two functions. For $c_f \geq 3$ it is given in a logarithmic scale, and when $c_f < 3$ it is given as a linear interpolation in the interval $[0, 3]$.

2. Obtain the minimum ($d_{\min}$) and maximum ($d_{\max}$) distance between the probability of the target dataset and the above average.
3. Estimate the new conditional probabilities of the target network as follows:

$$p'_{\text{target}} = (1 - c_i)\, p_{\text{target}} + c_i \overline{p} \tag{9}$$

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 CPTs. This is expressed as follows:

$$c_i = (d_i - d_{\min}) \times \left(\frac{c_{\max} - c_{\min}}{d_{\max} - d_{\min}} \right) + c_{\min} \tag{10}$$

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 $c_{\min} = 0.25$.

This aggregation function favors probabilities that are closer to the target probability, and weights them according to their confidence. The next function we propose considers only those that are similar to the target.

5.2 Local linear pool (LoLP)

The idea in our second aggregation function, called LoLP, is to use only the most similar or local probabilities and weight them also according to their confidence based on the amount of data. The procedure is similar to DBLP, but in this case the average of the probabilities obtained from the data of the auxiliary tasks, only includes those estimates that are *closer* to the target, the other ones are not considered. Then this average is combined with the target estimate weighting each by a confidence factor based on the amount of data.

LoLP has the following steps:

1. Obtain the average of the probabilities of the auxiliary datasets, 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:

$$\overline{p}_{\text{local}} = \frac{1}{n} \sum_{i=1}^{n} p_i \quad \forall p_i \quad \text{s.t.} \quad p_i \in \{p_{\text{target}} \pm (p_{\text{target}} - \overline{p})\} \tag{11}$$

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

$$p'_{\text{target}} = f_{\text{target}} \times p_{\text{target}} + (1 - f_{\text{target}}) \times \overline{p}_{\text{local}} \tag{12}$$

where f_{target} gives a confidence factor for the CPTs based on (8).

Table 1 Description of the networks used in the experiments showing the number of nodes (n) and arcs (α) of the original networks

Name	Description	n	α
Alarm	Diagnosis for monitoring intensive care patients	37	46
Boblo	Identification of blood for identification of cattle	23	24
Insurance	Classifier for car insurance applications	27	52

This aggregation function restricts the transfer to only those parameters that are close to the target estimates.

Given that the proposed aggregation functions are based on the lineal pool, this is used as a baseline for comparison in the experiments. Next we present the experimental evaluation of our inductive transfer techniques for structure and parameter learning.

6 Experiments

The purpose of the experiments is to show how a model induced from scarce data of a particular task can be improved with data from similar tasks. We performed two sets of experiments. First we use known BNs in which the structure and parameters are known, so we can use them as *gold standard* to evaluate the learned models. We also used data from a real-world application, in which we do not know the model, so the evaluation is done using the log-likelihood between the model and the data. We start by describing the first set of experiments.

We used three Bayesian networks: ALARM (Beinlich et al. 1989), BOBLO (Rasmussen 1992) and INSURANCE (Binder et al. 1997), briefly described in Table 1, commonly used to evaluate BN learning algorithms.

We adopted the evaluation criteria proposed in (Niculescu-Mizil and Caruana 2007) to evaluate our proposed methods for structure and parametric learning. We created new related problems from the original networks described in Table 1, by adding/removing links and changing the CPTs using Gaussian noise with $mean = 1$ and different values of standard deviations. From each network we generated data samples of different sizes. The data used in the experiments was sampled using probabilistic logic sampling as implemented in Elvira (2002) from the altered networks. These samples were used in conjunction to a subset of the original data to try to reconstruct the original network. Each experiment was repeated 10 times with independent data samples, in the results we show only the average.

We performed three types of experiments. In the first set of experiments, we evaluated how the proposed methods for structure and parameter learning are affected by the size of the data samples taken from the target task, and from related tasks. The second set of experiments evaluates how the structure learning method is affected by changing the degree of similarity between the target and the auxiliary tasks. In the third set of experiments we evaluated the difference between our proposed method and the PC algorithm with all the available data (target and auxiliary tasks). We also evaluated the sensitivity of the method to some of the parameters.

6.1 Experiments with different sizes of data samples

We created two different BNs with different degrees of similarity to the original model. One is a more similar network (MSN) built by randomly adding 5% of new links followed by

Table 2 Characteristics of the auxiliary networks used in the first set of experiments for the Alarm network

		Added	Removed	Inverted	Edit-distance	Noise in TCP (%)	MSE
Alarm	More similar net	2	1	0	3	5%	0.019
	Less similar net	6	8	0	14	20%	0.041

removing 5% of the current links and introducing 5% of random noise in the CPTs obtained from 100,000 examples of the original network. The second one is a less similar network (LSN) built by adding 20% of links followed by removing 20% of the current links and introducing 20% of random noise to the TPCs obtained from 100,000 cases sampled from the original network. Table 2 shows the differences between the two auxiliary networks and the target network in the structure: number of each basic edit operation and total edit distance; and in the parameters: percentage of noise and resulting mean square error (MSE). These models were used as auxiliary tasks in the experiments by generating samples via logic sampling.

In the first experiments we started with 1,000 samples of the auxiliary tasks and changed the number of samples of the target task from 25 to 2,000. In the second experiments we fixed the samples of the target task (100) and changed the number of samples in the auxiliary tasks from 500 until 8,000 to see how much improvement can be achieved with transfer learning from different sample sizes from the auxiliary tasks. We used the PC algorithm as implemented in Elvira (2002) as the base case with a significance of 0.90 for the independence tests.

In both experiments we evaluated the learned models by comparing them to the original network; the structure in terms of edit distance, and the structure and parameters in terms of MSE.

6.1.1 Results

The experiments were performed for the three BNs: Alarm, Boblo, and Insurance. There results are very similar in the three cases, and have the same qualitative behavior, so we discuss in detail only the Alarm case, but present results for the three domains.

The transfer learning algorithm was evaluated by using each auxiliary task separately (MSN and LSN), and then both at the same time. Figure 1 shows the behavior of PC-TL against the basic PC algorithm using samples from the networks described in Table 2 when the number of samples in the target task are increased while the samples of the auxiliary tasks remain constant (1,000 samples). Figure 2 shows analogous tests, but when the number of samples in the auxiliary tasks are varied while the samples of the target task remain constant (100 samples).

Figure 1 shows the expected behavior. The proposed method has a better performance than PC, especially when the number of cases from the target task is small. Both tend to converge to the same structure as the number of samples increases. There is not much difference between the performance of PC-TL with the different auxiliary tasks. The results in the graph are the average of 10 runs, so if we take into account the variance of each data point, the differences for the more/less similar tasks or the combination, are not significant. This could be because there is not so much difference between both auxiliary models. To analyze how the performance degrades with more distant networks we performed another set of experiments that are described in the next section. Figure 2 shows that the network structures induced in conjunction with the auxiliary tasks improve as the number of samples

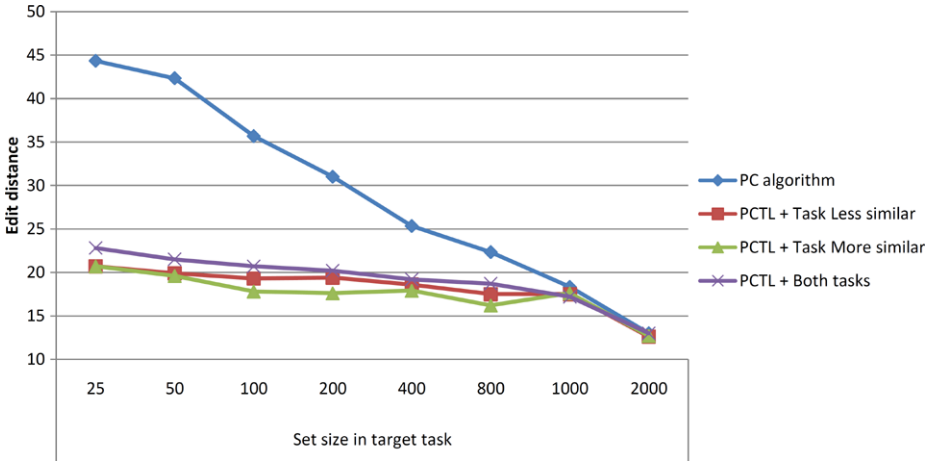


Fig. 1 Behavior of the proposed method (PC-TL) as the number of samples of the target task increases (Alarm network)

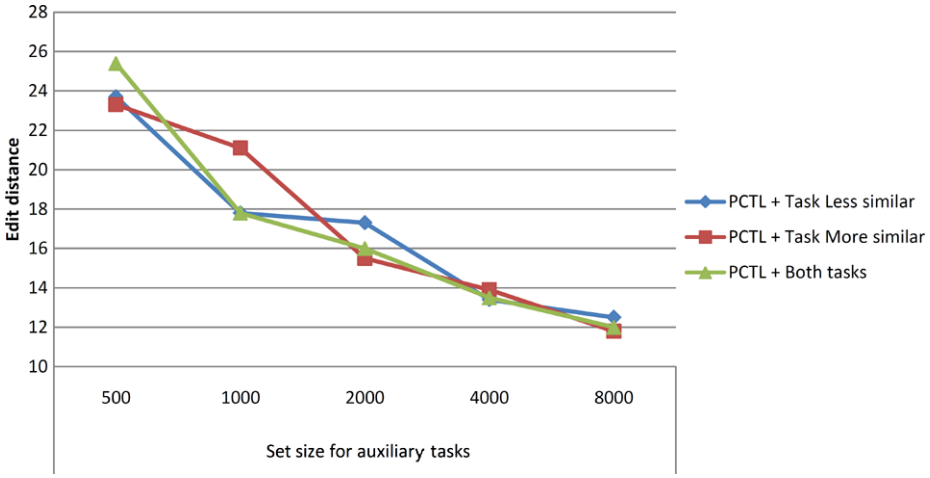


Fig. 2 Behavior of the proposed method (PC-TL) as the number of samples of the auxiliary tasks increases (Alarm network)

of the auxiliary tasks increases. In general, better results are obtained when more than one auxiliary task is used.

Next we present the results for parameter learning, which also take into account the structure. In these experiments, for each sample size we first learn the structure and then the parameters, so we are really evaluating both aspects. Due to this, the MSE might not be the same even for the same sample size. All the data points in the graphs are the average of 10 runs.

Figure 3 shows the performance of the parameter learning algorithms, in terms of mean square error, measured against the true parameters using data from the target task (base), the linear aggregation algorithm (LP) and the proposed algorithms (DBLP and LoLP). The top figure shows the behavior when the auxiliary data comes from the MSN, the middle

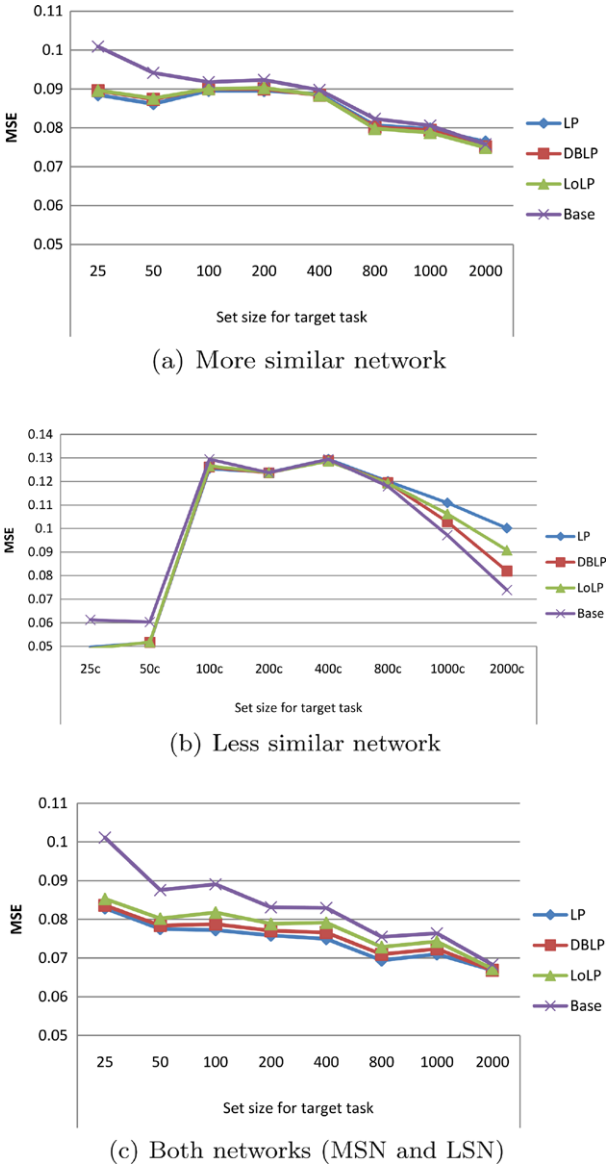
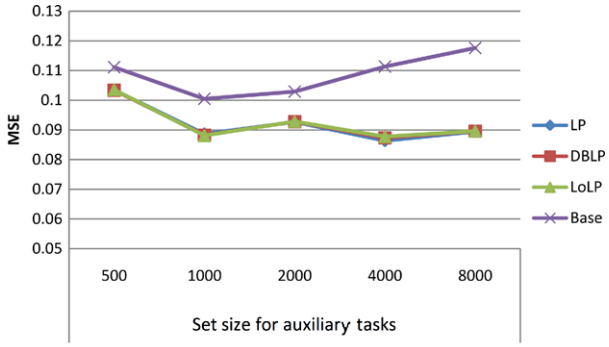


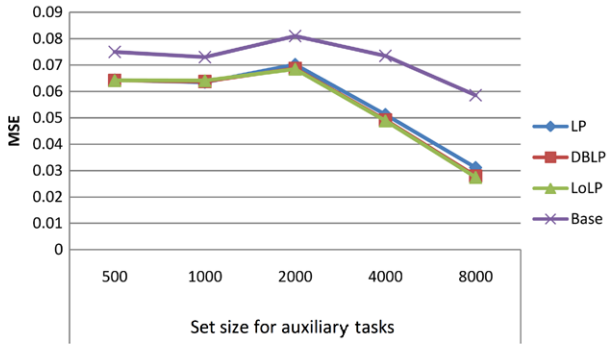
Fig. 3 Comparison between the parameter learning algorithms with different numbers of instances of the target task for the Alarm network, with different auxiliary tasks. *Top*: with data from the MSN. *Middle*: with data from the LSN. *Bottom*: using data from both networks

figure shows the behavior with the LSN, and the bottom figure shows the behavior with both networks. The number of samples for the auxiliary tasks was fixed at 1,000. Figures 4(a), (b), (c) show the behavior when the number of instances of the auxiliary tasks are varied. The number of samples for the target task was fixed at 100.

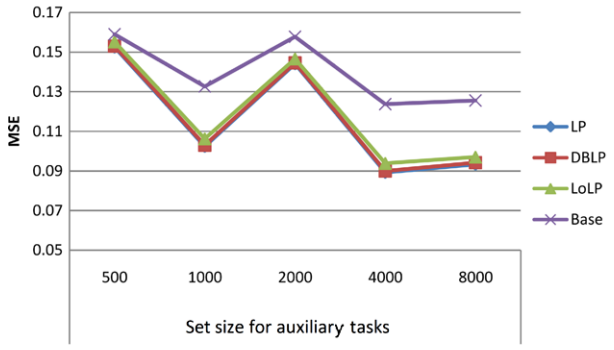
Figure 3 shows, as expected, in general a reduction in error as we increase the number of instances in the target task. For the case with the MSN there is a *strange* behavior for the first two sample sizes (25 and 50) that could be because the structure of the model is being learned from a very small dataset, which frequently implies having very few arcs. In this case the MSE is estimated based on a few CPTs and this could give an apparent low error.



(a) More similar network



(b) Less similar network



(c) Both networks (MSN and LSN)

Fig. 4 Comparison between the parameter learning algorithms with different numbers of instances of the auxiliary tasks for the Alarm network, with different auxiliary tasks. *Top*: with data from the MSN. *Middle*: with data from the LSN. *Bottom*: using data from both networks

In the other experiments, see Fig. 4, we observe a significant error reduction when we use data from auxiliary tasks, and in general the error is reduced as we increase the sample size of the auxiliary tasks. In some cases the results are slightly better with the DBLP and LoLP methods, but in general there is no significant difference from the basic LP.

The differences between the graphs and the high variability (Figs. 3 and 4) are due to the different network structures obtained for each case, as we varied the sample sizes. That is, for each experiment the structure of the network changes, so even with the same number of instances for target and auxiliary tasks, the MSE may vary. We consider that the

Table 3 Characteristics of the networks used in second set of experiments for the Alarm network

		Added	Removed	Inverted	Edit-distance	MSE
Alarm	Similar net 1	4	5	0	9	0.018
	Similar net 2	9	11	0	20	0.033
	Similar net 3	8	14	0	22	0.027
	Similar net 4	10	22	0	32	0.047
	Similar net 5	11	33	0	44	0.055
	Similar net 6	12	52	0	64	0.032

Table 4 Characteristics of the networks used in third set of experiments for the Alarm network

		Added	Removed	Inverted	Edit-distance	MSE
Alarm	Similar net 1	3	0	0	3	0.018
	Similar net 2	10	0	0	10	0.044
	Similar net 3	13	0	0	13	0.050
	Similar net 4	16	0	0	16	0.043
	Similar net 5	24	0	0	24	0.079
	Similar net 6	29	0	0	29	0.088
	Similar net 7	32	0	0	32	0.094
	Similar net 8	40	0	0	40	0.094
	Similar net 9	43	0	0	43	0.100

strange behavior observed when we vary the data from the auxiliary tasks with both networks, Fig. 4(c), is also due to changes in the structure of the models.

6.2 Experiments with different auxiliary tasks

The objective of this set of experiments is to evaluate how more distinct auxiliary tasks affect the structure learning algorithm. In these experiments we created a larger number of auxiliary networks with different degrees of edit distances. In the second set of experiments we created six auxiliary tasks by adding 10%–60% of links followed by the removal of 10%–60% of the original links and with the parameters obtain from 100,000 instances sampled from the original network. The idea was to test when transfer learning starts to damage the recovery of the target network. In the third set of experiments we created nine networks, starting from the original network, by randomly removing from 10% to 90% of the original links. The description of these networks is given in Tables 3 and 4.

In order to measure the effect of the transfer learning approach independently from the used data, in a fourth set of experiments we compared the performance of the PC algorithm using all the available data (both from the target task and the auxiliary tasks) with our PC-TL algorithm.

6.2.1 Results

Again we discuss in detail only the results for the Alarm network, the results for the other two BNs (Boblo and Insurance) are very similar and are presented below.

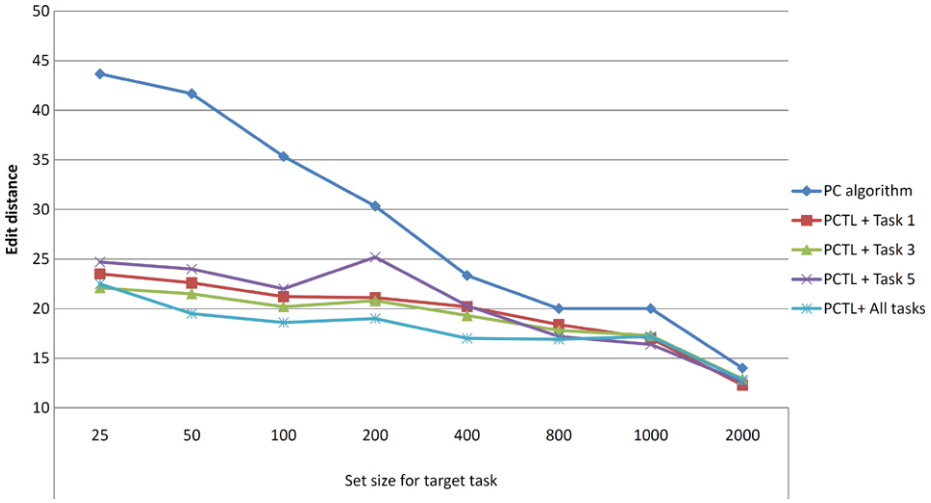


Fig. 5 Behavior of PC-TL when the similarity between the target and the auxiliary task changes (for the Alarm network)

Figures 5 and 6 summarize the results for the second set of auxiliary networks with arcs added and removed. Figure 5 shows the behavior of PC-TL vs. PC in terms of edit distance, with different auxiliary tasks (three of the models from the first set), changing the number of instances in the target task (between 25–2,000) with a fixed number of instances (1,000) in the auxiliary tasks. We observe that even with tasks that are significantly different, as task five with an edit distance of 44, there is a significant improvement in the structure by incorporating data from the auxiliary task. Figure 6 shows the results for the six auxiliary models in the second set, varying the instances (between 500–8,000) in the auxiliary tasks with a fixed number of instances (100) in the target task. We confirm that in general there is a greater improvement as we increase the amount of data of the auxiliary tasks, and also as the tasks are closer to the target. However, the best results are obtained by using all the auxiliary data, even when some come for quite different models. There are some cases in which a less similar network gives a better result, this could be due to the variance in the experiments, as we show only the average. But in general, as we see in the other experiments below, the results are better with more similar networks and in particular when all the auxiliary tasks are considered.

In the third experiment we analyzed the effect of auxiliary networks that are very different from the target BN. In this case we removed arcs up to a high percentage (90%) of the original model. Figure 7 shows the behavior of the algorithm when the auxiliary tasks differ from the target task in the number of links (10%–90% less links in the auxiliary tasks). We sampled 100 instances from the target task and between 500–8,000 samples from the auxiliary task. We observe that the error increases as the auxiliary model is more different from the target, and the last two models have more error than just using the data from the target task (PC algorithm). However, we can observe that again the best results are with all the auxiliary tasks.

Figures 5, 6 and 7 show that in general PC-TL degrades its performance as the structure of the auxiliary tasks departs from the target task, and also in general the best performance is obtained by combining all the auxiliary tasks. Figure 7 shows how PC-TL is able to recover a close structure of the target task when a larger number of auxiliary tasks is used. Figures 6

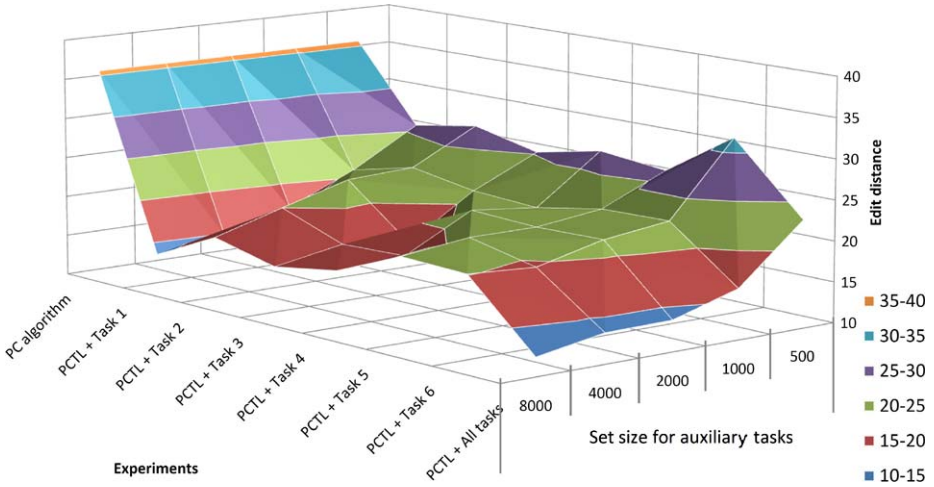


Fig. 6 Behavior of the PC-TL algorithm with different similarities between the target and auxiliary tasks and with different number of instances in the auxiliary tasks (Alarm network)

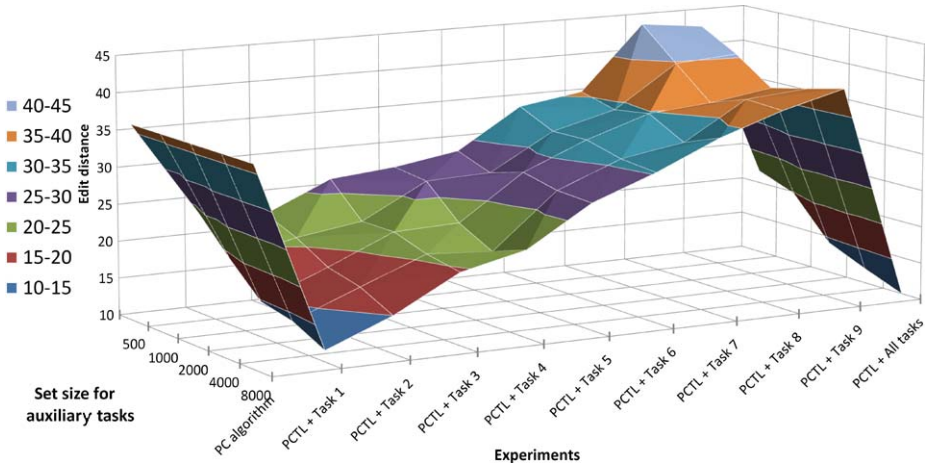


Fig. 7 Behavior of PC-TL with auxiliary tasks constructed by deleting links from the target task (Alarm network)

and 7 show that the target structure improves as more data is considered from the auxiliary tasks.

These experiments demonstrate that in general the structure improves as the similarity or the data from the auxiliary tasks increases, and combining all the auxiliary tasks in general gives the best results. There are some variations in these tendencies due to the variance in the experiments, as all results are the average of 10 repetitions with data generated randomly from the target and auxiliary models.

From these last sets of experiments, Figs. 6 and 7, we observe that using data from not so similar tasks can still improve learning of the target task. This could be due to: (i) more data points, even if some are not relevant or (ii) the transfer learning method, which takes into

Table 5 Characteristics of the networks used in fourth set of experiments for the Alarm network

		Added	Removed	Inverted	Edit-distance
Alarm	Net 1	20	20	0	40
	Net 2	33	33	0	65
	Net 3	40	40	0	80
	Net 4	43	43	0	86

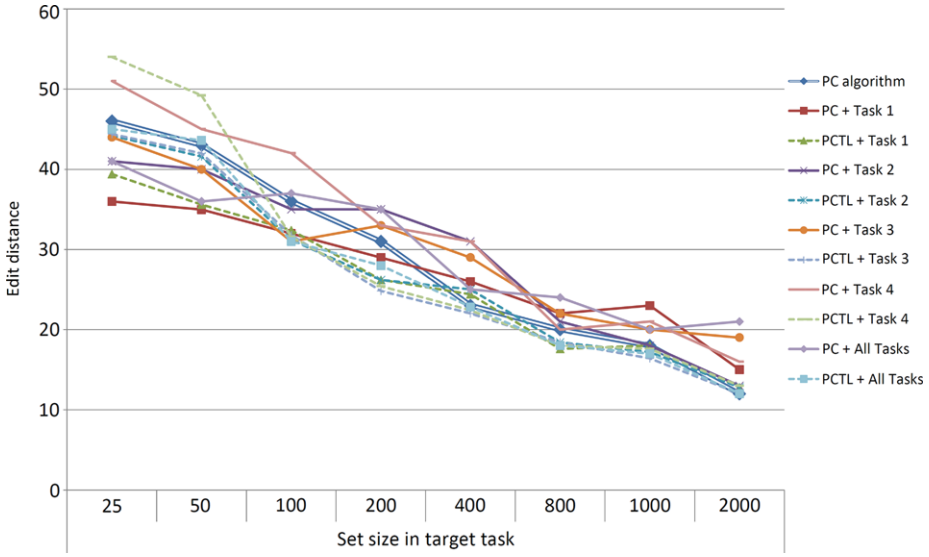


Fig. 8 A comparison of the PC algorithm with all the available (target and auxiliary) data and the PC-TL algorithm, for 4 different auxiliary tasks, and with different dataset sizes for the target task (Alarm network)

account the similarity of the auxiliary tasks. To distinguish between these two hypotheses, and show the effectiveness of our method, we performed additional experiments described below.

6.3 Experiments with PC using all the available data

In the last set of experiments we compared the performance of the PC algorithm using all the available data (both from the target task and the auxiliary tasks) with our PC-TL algorithm. In preliminary experiments we noticed that if we used very similar (target and auxiliary) datasets the performance of PC-TL and PC with all the available data is very similar. In the experiments of this section, we used the data described in Table 5, where there is a significant difference between the auxiliary and the target tasks. We consider 4 different auxiliary tasks and generated 1,000 data points for each task, varying the amount of data for the target task.

6.3.1 Results

The results are shown in Fig. 8, where we gradually increased the size of the target data and learned a model with data from four different auxiliary tasks. The figure shows with

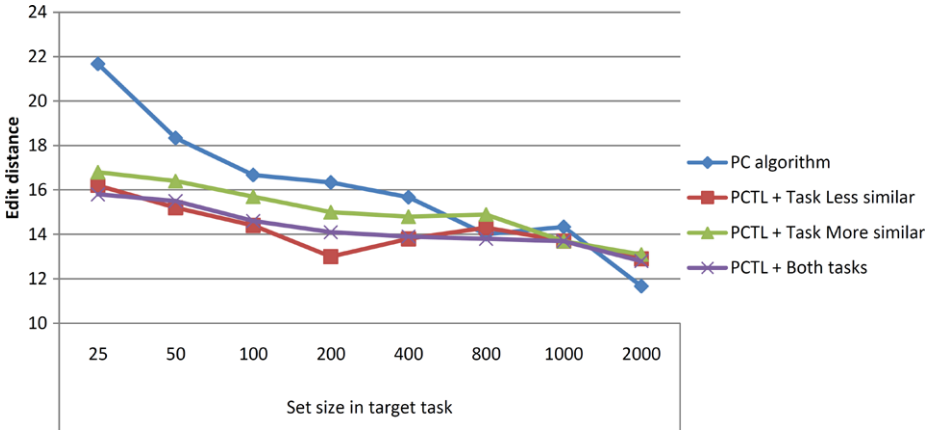


Fig. 9 Behavior of the proposed method (PC-TL) as the number of samples of the target task increases (Boblo network)

dotted lines the behavior of the PC-TL algorithm. As it can be seen from the figure, the PC algorithm with all the available data tends to produce larger errors than the PC-TL algorithm with these datasets. This is expected as the PC algorithm fits a model according to the available data, so it produces large errors when the auxiliary data comes from a task that is not very similar to the target task. When the amount of data available for the target task increases, PC with all the data tends to produce larger errors than PC with just the target data. On the other hand, the PC-TL algorithm has the advantage of weighting the contribution of the auxiliary data according to their similarity, thus producing more accurate models and a more useful transfer learning approach. This is particularly clear when we combine several auxiliary tasks with PC-TL, obtaining in general the best results.

Thus, we conclude that using all the available data with the basic PC algorithm could work just in the case that the auxiliary tasks are very similar to the target; so in general it will be risky to use this simple approach. In contrast, PC-TL works well with similar and not so similar auxiliary tasks, so it is a more robust and effective method.

6.4 Results for the Boblo and Insurance datasets

We performed the same experiments for the Boblo and Insurance data sets. Here we present some of the most representative results. For Boblo, Figs. 9, 10 and 11 show the behavior of the proposed method (PC-TL) as the number of samples of the target task increases, as the number of samples of the auxiliary tasks increases, and the behavior with auxiliary tasks constructed by deleting links from the target task, respectively.

Similarly for the Insurance dataset, Figs. 12, 13 and 14 show the behavior of the proposed method (PC-TL) as the number of samples of the target task increases, as the number of samples of the auxiliary tasks increases, and the behavior with auxiliary tasks constructed by deleting links from the target task, respectively.

We can observe in both sets of figures a very similar behavior with the experiments of the Alarm network.

6.5 Sensitivity analysis

To evaluate the sensitivity of the method to certain parameters we performed some additional tests. In particular, we measured the sensitivity to the constants in (3), which control the

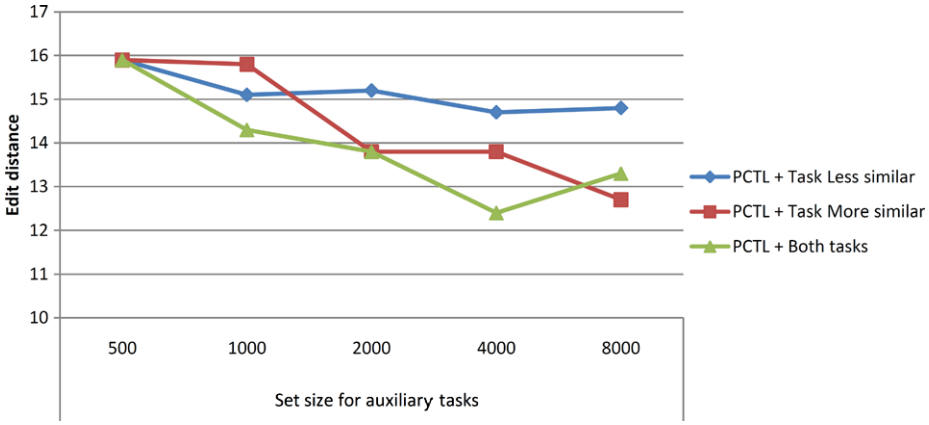


Fig. 10 Behavior of the proposed method (PC-TL) as the number of samples of the auxiliary tasks increases (Boblo network)

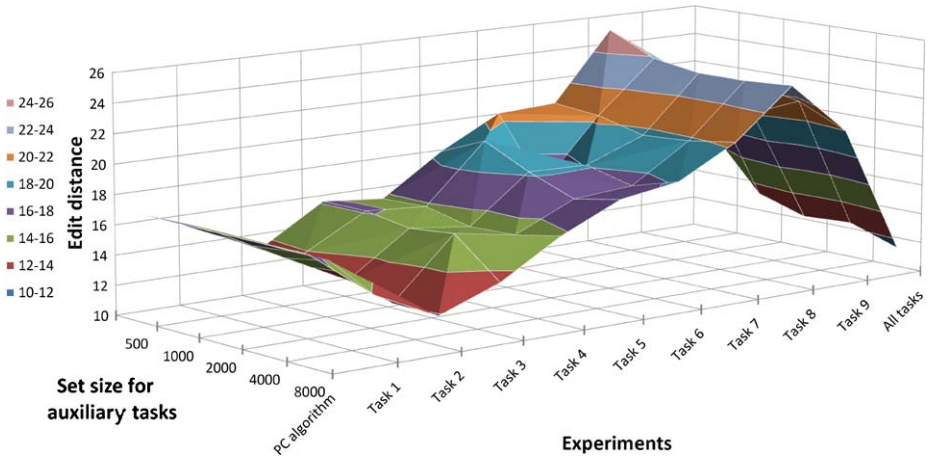


Fig. 11 Behavior of PC-TL with auxiliary tasks constructed by deleting links from the target task (Boblo network)

weight given to auxiliary tasks according to the local similarity. The values used in our previous experiments were $(1, 0.5)$, which gave two times more weight to auxiliary tasks with the same independence values versus those with different independence results. We compared the performance for the *Alarm* task, by given approximately the same weight $(0.6, 0.4)$ and by making the difference larger, with a ratio of three $(1.5, 0.5)$. The results in terms of edit distance for different auxiliary tasks (the same ones used in the first experiments in Sect. 6.2.) for the Alarm BN, and varying the data from the auxiliary tasks, are given in Fig. 6 for the original parameters, and in Fig. 15 for the other two set of parameters.

We observe that there is not a significant difference in performance in the three cases, which gives evidence that the method is not very sensitive to these parameters. In general the results are slightly better when we give more weight to the auxiliary tasks that are locally similar in terms of the independence tests.

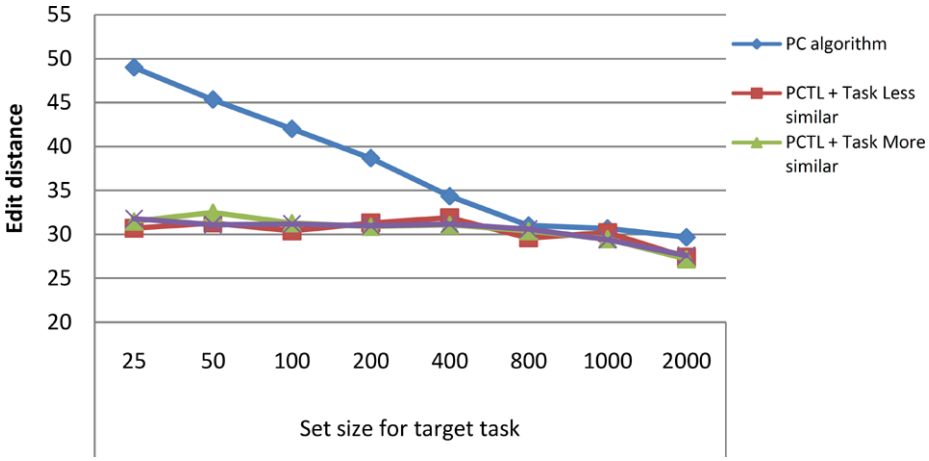


Fig. 12 Behavior of the proposed method (PC-TL) as the number of samples of the target task increases (Insurance network)

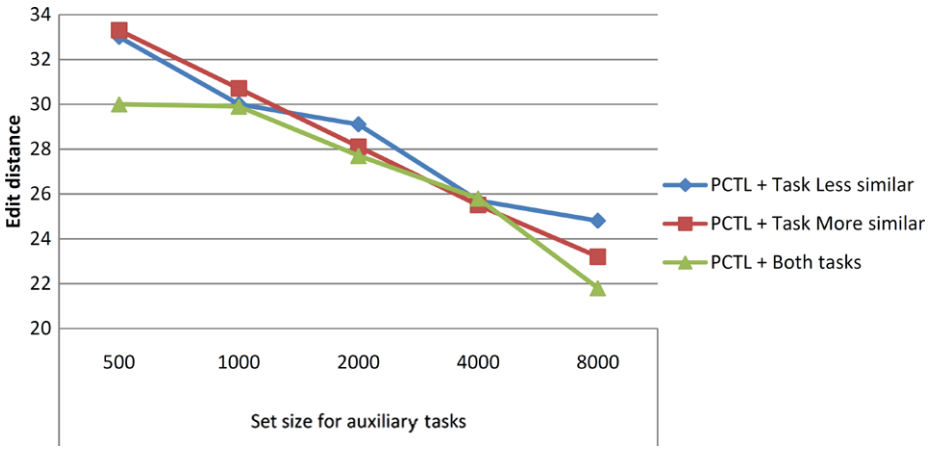


Fig. 13 Behavior of the proposed method (PC-TL) as the number of samples of the auxiliary tasks increases (Insurance network)

6.6 Experiments in a *real-world* problem

We have also evaluated our method in a *real-world* industrial process in which several, similar, products are manufactured.[7] The process is automated and different variables (such as temperature, pressure, flow, etc.) at different stages of the process are measured and stored in a database, one for each type of product. The datasets are from products that follow the same stages (production line) but under different production conditions. Namely, they are produced at different temperatures, different pressures, they spend different times at the

[7] Due to a confidentiality agreement with the company, we can not disclose the details of the process and the actual names of the variables.

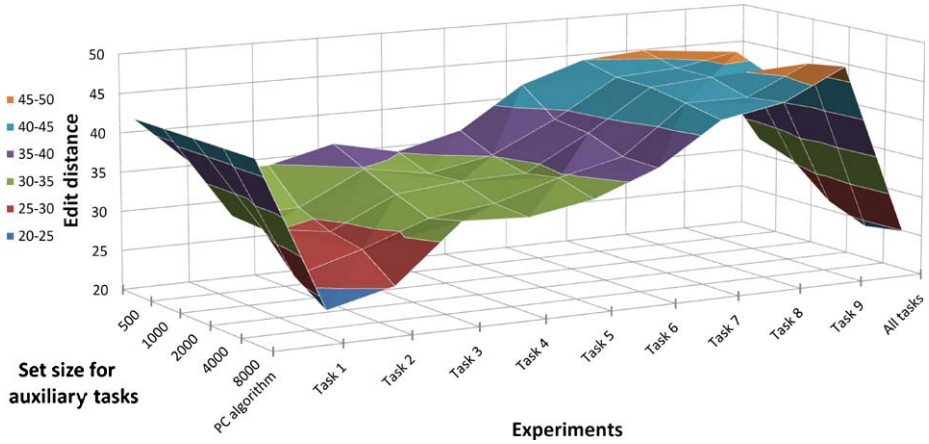


Fig. 14 Behavior of PC-TL with auxiliary tasks constructed by deleting links from the target task (Insurance network)

Table 6 Results for the a real-world manufacturing process considering a similar product as auxiliary task

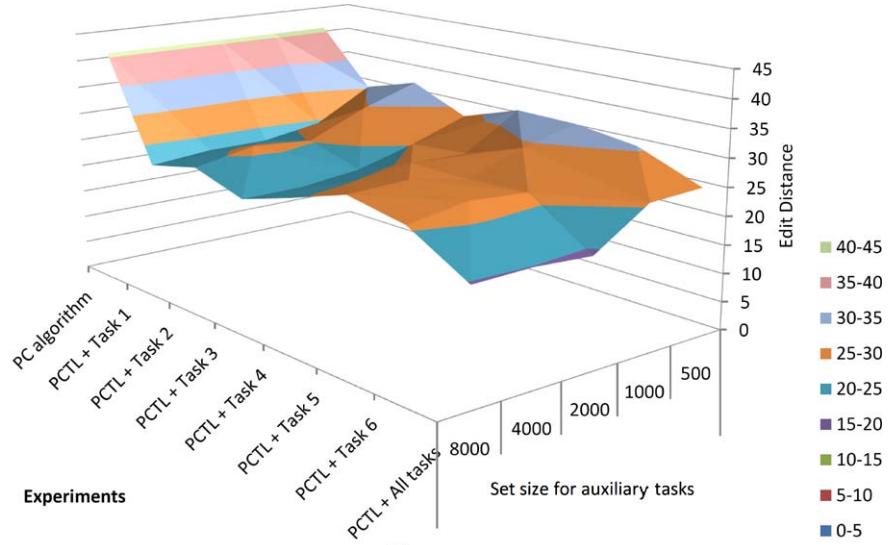
Model	Log-likelihood
Target data only (PC)	−2.589
Target and auxiliary data (PC-TL)	−2.325

production stages and they have different composition. However, the same variables are recorded for these products as they follow the same production line. Some products are manufactured in larger quantities while others are very rarely produced, so the amount of data for each product varies significantly. The company is interested in building a model of the process that could help to improve the quality of the products. Given that for some products there is a relatively small dataset, we are interested in using inductive transfer from similar products to obtain a better model for rare products.

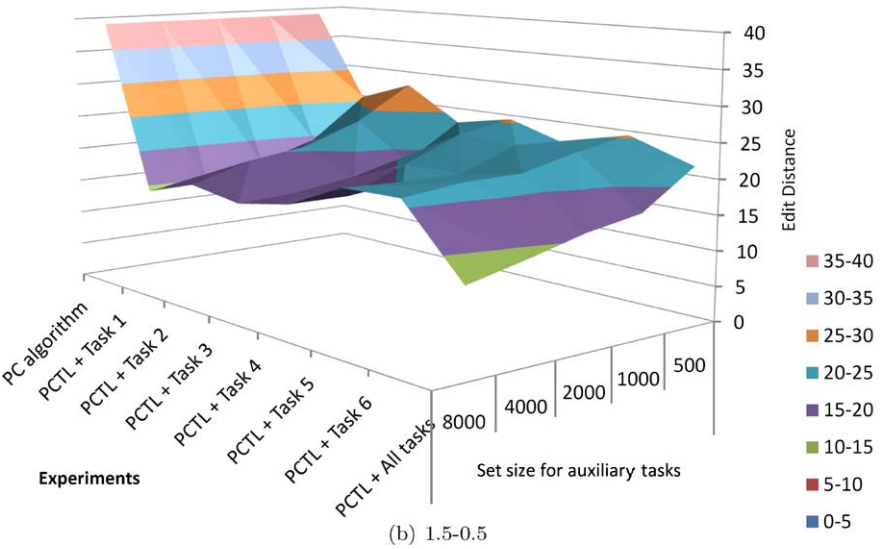
We have applied our methodology for structure and parameter learning for inducing a BN for one of the products, using data from this product and other similar products. The model was built for part of the process including 15 continuous variables that were previously discretized. We compared the BN obtained combining the data from the target product (with only 260 instances for this manufacturing process) and other similar product using PC-TL, with the one obtained using data only from the target product using PC. We considered one auxiliary task (5,191 instances from another, similar product). Figure 16 depicts the structure obtained only with the target data, and Fig. 17 shows the structure using inductive transfer.

Given that we do not know the *reference* model, we compared them in terms of the log-likelihood (*ll*) between the target dataset and the model. The results are summarized in Table 6.

We observe a significant improvement in terms of the log-likelihood in comparison with using only data for the target product. We have tested with other related products with similar results, in particular when the dataset from the auxiliary task is significantly larger; if the size of the dataset is similar to the target domain, there is some improvement but not very significant.



(a) 0.6-0.4



(b) 1.5-0.5

Fig. 15 Sensibility analysis of the constants in (3) that balance how much influence have the auxiliary tasks over the target task (Alarm network)

We performed an additional experiment using the PC algorithm with all the available data from another target product (67 samples) and from a related product (5,191 samples), and our PC-TL algorithm.[8] The results are shown in Fig. 18 with different sizes of the auxiliary data. As can be seen from the figure, there is a significant difference between the PC algorithm that uses all the data (target and auxiliary task), and our approach (PC-TL);

[8]Notice that in this and the previous experiment, we consider different target tasks but the same auxiliary task.

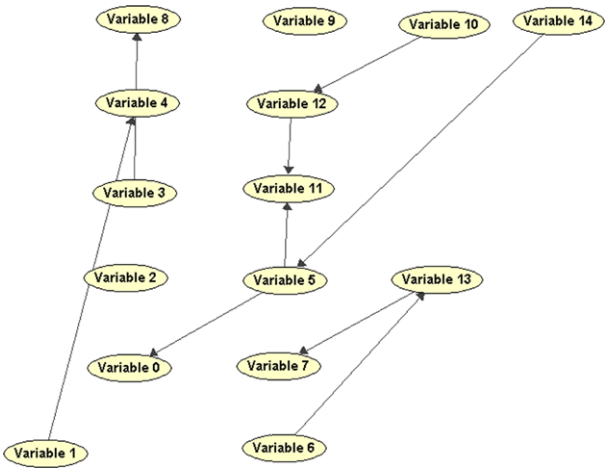


Fig. 16 Structure obtained by the PC algorithm for the real-world manufacturing task, using only data from the target product

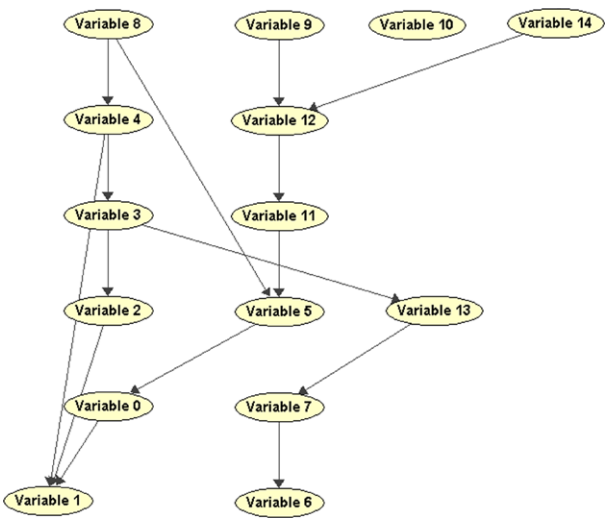


Fig. 17 Structure obtained by the PC-TL algorithm for the real-world manufacturing task, using data for the target product and a similar product

which produces better results for all sets sizes of the auxiliary task. In this case PC-TL produces similar results to PC based only on the target data. The poor performance of using all the data with the basic PC algorithm could be due to the large difference in sizes of the datasets. This problem does not occur with PC-TL, which in the worst case gives results similar to those using PC only with the target data, and in most cases in our experiments tends to improve the model using transfer learning.

6.7 Discussion

In general we can draw the following conclusions from these experiments:

– Transfer learning from related tasks improves in general the structure and parameters of the learned models (using as base algorithm PC); these improvements are more significant as we reduce the data sample of the target task.

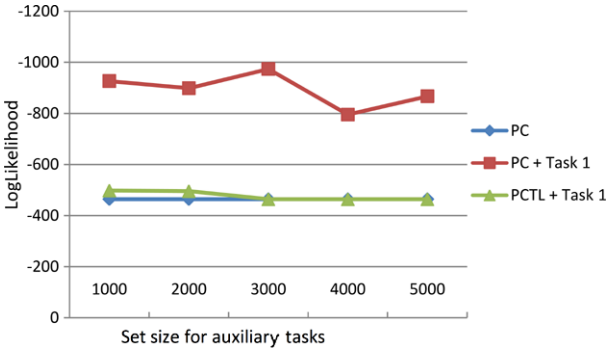


Fig. 18 Behavior of PC with all the available (target and auxiliary) data and the PC-TL with different dataset sizes of the auxiliary task for the manufacturing task

- In general, as we have a larger data sample from the auxiliary tasks, there is a greater improvement in both, structure and parameters.
- The error reductions (in terms of edit distance) in the structure depend on the similarity of the auxiliary BNs, however even with structures that are not very close the reductions are significant.
- Sometimes a less similar model gives an apparently better structure, this is due to two factors: (i) the variance of the experiments (we show the averages of 10 runs with data generated randomly); (ii) the models are relatively close. With more dissimilar auxiliary tasks the difference in favor of the closer ones is cleaer.
- Transferring from several auxiliary models is in general better, with a larger improvement than that from each model individually; as our method takes into account the similarity between the auxiliary tasks and the target.
- Using PC with all the available data is only useful when the datasets are very similar.
- The MSE in the parameters is reduced by using data from auxiliary tasks via a linear combination with the data from the target task, and this reduction is greater as we increase the sample size in the auxiliary tasks.
- In some cases the proposed parameter learning methods, DBPL and LoLP, improve the basic LP, however in general there is no significant difference.

Additional details of these experiments are described in Luis-Velázquez (2009).

7 Conclusions and future work

In this paper we have presented a novel methodology for transfer learning in Bayesian networks, including structure learning and parameter learning. The idea behind this research is to use information from related datasets in order to improve the performance of networks constructed with small datasets. This is common is several domains, where there can be rare cases, that nevertheless share some similarities with more common cases. Learning a Bayesian network involves two steps, learning the network structure and learning its parameters. For structure learning we extended the PC algorithm to consider information from related tasks. The PC-TL algorithm considers the confidence of the independence tests, and the similarity of the auxiliary tasks to the target in the combination function. For parameter learning we defined two extensions to the linear combination technique, that also consider the expected error and similarity. To test our method we used three Bayesian networks models, 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. The improvement increases in proportion to the amount of data from the auxiliary tasks, and in general the results are better with more similar tasks; although we observe that even not so similar tasks can improve the model, and the best results are obtained when we combine all the auxiliary tasks. We compared the proposed transfer learning method with PC using all the data (target and auxiliary tasks), the results show that using PC with all the data could be an alternative only when the tasks are very similar, and our method is more robust producing good results under different conditions. We also evaluated the method with real-world data from a manufacturing process considering several products, showing an improvement in terms of log-likelihood when using data from similar products.

As future work, we would like to explore how to combine several related tasks at the same time considering more complex conditional independence tests between the tasks. The PC algorithm depends on the amount of available data, so we would also like to know how to deal with small datasets both in the target and the related tasks.

Acknowledgements The authors acknowledge to CONACyT the support provided through the grant for MSc. studies number 207086 and project No. 47968. We thank the anonymous referees and the editor, Gayle Leen, for their thoughtful and constructive suggestions.

References

Baxter, J. (1997). A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, *28*(1), 7–39.

Beinlich, I. A., Suermondt, H. J., Chavez, R. M., & Cooper, G. F. (1989). The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks. In *Proceedings of the second European Conference on Artificial Intelligence in Medicine*. Berlin: Springer.

Binder, J., Koller, D., Russell, S., & Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, *29*(2–3), 213–244.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.

Chang, C.-S., & Chen, A. L. P. (1996). Aggregate functions over probabilisitic data. *Informing Science*, *88*(1–4), 15–45.

Chen, A. L. P., Chiu, J.-S., & Tseng, F. S.-C. (1996). Evaluating aggregate operations over imprecise data. *IEEE Transactions on Knowledge and Data Engineering*, *8*(2), 273–284.

Cooper, G. F., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*(4), 309–347.

Dai, W., Xue, G., Yang, Q., & Yu, Y. (2007). Trasfer naive Bayes classifiers for text classification. In *Proceedings of the twenty-second AAAI conference on artificial intelligence (AAAI-07)* (pp. 540–545). Menlo Park: AAAI Press.

Elvira (2002). Elvira: an environment for creating and using probabilistic graphical models. In J. A. Gámez & A. Salmerón (Eds.), *First European workshop on probabilistic graphical models*, 6–8 November, 2002, Cuenca (Spain). Electronic Proceedings.

Friedman, N., & Yakhini, Z. (1996). On the sample complexity of learning Bayesian networks. In E. J. Horvitz & F. V. Jensen (Eds.), *Proceedings of the 12th conference on Uncertainty in Artificial Intelligence* (pp. 274–282). San Mateo: Morgan-Kaufmann.

Genest, C., & Zidek, J. V. (1986). Combining probability distributions: a critique and an annotated bibliography. *Statistical Science*, *1*(1), 114–148.

Lam, W., & Bacchus, F. (1994). Learning Bayesian belief networks: an approach based on the MDL principle. *Computational Intelligence*, *10*, 269–293.

Luis-Velázquez, R. (2009). *Aprendizaje por transferencia de redes Bayesianas* (Technical Report). Instituto Nacional de Atrofísica, Óptica y Electrónica, Mexico.

Niculescu-Mizil, A., & Caruana, R. (2007). Inductive transfer for Bayesian network structure learning. In *Proceedings of the 11th international conference on AI and statistics (AISTATS '07)*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Mateo: Morgan Kaufmann.

Rasmussen, L. (1992). *Blood groups determination of danish jersey cattlein the f-blood group system* (Technical Report Dina Res. Rep. No. 8). Research Centre Foulum, Denmark.

Richardson, M., & Domingos, P. (2003). Learning with knowledge from multiple experts. In T. Fawcett & N. Mishra (Eds.), *Machine learning, Proceedings of the Twentieth International Conference (ICML 2003)*, August 21–24, 2003, Washington, DC, USA (pp. 624–631). Menlo Park: AAAI Press.

Roy, D., & Kaelbling, L. (2007). Efficient Bayesian task-level transfer learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)* (pp. 2599–2604). Menlo Park: AAAI Press.

Silver, D., Poirier, R., & Currie, D. (2008). Inductive transfer with context-sensitive neural networks. *Machine Learning*, *73*(3), 313–336.

Spirtes, P., Glymour, C., & Scheines, R. (1993). *Causation, prediction, and search*. Berlin: Springer.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems* (vol. 8, pp. 640–646). Cambridge: MIT Press.

Wu, P., & Dietterich, T. G. (2004). Improving SVM accuracy by training on auxiliary data sources. In *ICML '04: Proceedings of the Twenty-first International Conference on Machine Learning* (p. 110). New York: ACM.