JMLR: Workshop and Conference Proceedings 27:167–181, 2012 Workshop on Unsupervised and Transfer Learning

Inductive Transfer for Bayesian Network Structure Learning

Alexandru Niculescu-Mizil ALEX@NEC-LABS.COM
NEC Laboratories America, 4 Independence Way, Princeton, NJ 08540

Rich Caruana RCARUANA@MICROSOFT.COM
Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399

Editor: I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver

Abstract

We study the multi-task Bayesian Network structure learning problem: given data for multiple related problems, learn a Bayesian Network structure for each of them, sharing information among the problems to boost performance. We learn the structures for all the problems *simultaneously* using a *score and search* approach that encourages the learned Bayes Net structures to be similar. Encouraging similarity promotes information sharing and prioritizes learning structural features that explain the data from all problems over features that only seem relevant to a single one. This leads to a significant increase in the accuracy of the learned structures, especially when training data is scarce.[1]

1. Introduction

Bayes Nets (Pearl, 1988) provide a compact description of the dependency structure of a domain by using a directed acyclic graph to encode probabilistic dependencies between variables. The ability to learn this structure from data makes Bayes Nets an appealing data analysis tool, as the learned structure can convey, in an intuitive manner, a wealth of information about the domain at hand.

Until now, Bayes Net structure learning research has focused on learning *a single* structure for a single problem (task) in isolation (e.g. learn the structure for only one species of yeast from the gene expression data from that one species alone) (e.g. Cooper and Hersovits, 1992; Heckerman, 1999; Spirtes et al., 2000; Teyssier and Koller, 2005). In many situations, however, we are faced with multiple problems (tasks) that are related in some way (e.g. learn about the gene regulatory structure of several species of yeast, not just one). In these cases, rather than learning the Bayes Net structure for each problem in isolation, and ignoring the relationships with the other tasks, it would be beneficial to learn all the structure for all the problems jointly. Indeed, the transfer learning literature (e.g. Caruana, 1997; Baxter, 1997; Thrun, 1996) suggests that significant benefits can be obtained by *transferring* relevant information among the related problems.

In this paper we present a transfer learning approach that jointly learns multiple Bayesian Network structures from multiple related datasets. We follow a score and search approach, where the search is performed over *sets of DAGs* rather than over single DAGs as in case of traditional structure learning. We derive a principled measure of the quality of a *set of*

1. A version of this paper appeared in (Niculescu-Mizil and Caruana, 2007). The work was done while both authors were at Cornell University.

© 2012 A. Niculescu-Mizil & R. Caruana.

structures that rewards both a good fit of the training data as well as high similarity between the structures in the set. This score is then used to guide a greedy hill climbing procedure in a properly defined search space to find a high quality set of Bayes Net structures.

We evaluate the proposed technique on problems generated from the benchmark ALARM (Beinlich et al., 1989) and INSURANCE (Binder et al., 1997) networks, as well as on a real bird ecology problem. The results of the empirical evaluation show that learning the Bayes Net structures jointly in a multi-task manner does indeed yield a boost in performance and leads to learning significantly more accurate structures than when learning each structure independently. As with other transfer learning techniques, the benefit is especially large when the training data is scarce.

2. Background: Learning the Bayes Net Structure for a Single Problem

A Bayesian Network $\mathcal{B} = \{G, \theta\}$ compactly encodes the joint probability distribution of a set of n random variables $X = \{X_1, X_2, ..., X_n\}$. It is specified by a directed acyclic graph (DAG) G and a set of conditional probability functions parametrized by θ (Pearl, 1988). The Bayes Net *structure*, G, encodes the probabilistic dependencies in the data: the presence of an edge between two variables means that there exists a direct dependency between them. An appealing feature of Bayes Nets is that the dependency graph G is easy to interpret and can be used to aid understanding the problem domain.

Given a dataset $D = \{x^1, ..., x^m\}$ where each x^i is a complete assignment of variables $X_1, ..., X_n$, it is possible to learn both the structure G and the parameters θ (Cooper and Hersovits, 1992; Heckerman, 1999; Spirtes et al., 2000). In this paper we will focus on structure learning, and more specifically on the score and search approach to it.

Following the Bayesian paradigm, the posterior probability of the structure given the data is estimated via Bayes rule:

$$P(G|D) \propto P(G)P(D|G) \tag{1}$$

The prior, $P(G)$, indicates the belief, before seeing any data, that the structure G is correct. If there is no reason to prefer one structure over another, one should assign the same probability to all structures. If there exists a known ordering on the nodes in G such that all the parents of a node precede it in the ordering, a prior can be assessed by specifying the probability that each of the $n(n-1)/2$ possible arcs is present in the correct structure (Buntine, 1991). Alternately, when there is access to a structure believed to be close to the correct one (e.g. from an expert), $P(G)$ can be specified by penalizing each difference between G and the given structure by a constant factor (Heckerman et al., 1995).

The marginal likelihood, $P(D|G)$, is computed by integrating over all parameter values:

$$P(D|G) = \int P(D|G, \theta)P(\theta|G)d\theta \tag{2}$$

When the local conditional probability distributions are from the exponential family, the parameters θ_i are mutually independent, we have conjugate priors for these parameters, and the data is complete, $P(D|G)$ can be computed in closed form (Heckerman, 1999).

Treating the posterior, $P(G|D)$, as a score, one can search for a high scoring network using heuristic search (Heckerman, 1999). Greedy search, for example, starts from an initial structure, evaluates the score of all the *neighbors* of that structure and moves to the

neighbor with the highest score. A common definition of the *neighbor* of a structure G is a DAG obtained by removing or reversing an existing arc in G, or by adding an arc that is not present in G. The search terminates when the current structure is better than all it's neighbors. Because it is possible to get stuck in a local minima, this procedure is usually repeated a number of times starting from different initial structures.

3. Learning Bayes Net Structures for Multiple Related Problems

In the previous section we reviewed how to learn the Bayes Net structure for a single problem. What if we have data for a number of related problems (e.g., gene expression data for several species) and we want to jointly learn Bayes Net structures for each of them?

Given k data-sets, $D_1, ..., D_k$, defined on overlapping but not necessarily identical sets of variables, we want to learn the structures of the Bayes Nets $\mathcal{B}_1 = \{G_1, \theta_1\}$, ..., $\mathcal{B}_k = \{G_k, \theta_k\}$. In what follows, we will use the term *configuration* to refer to a set of structures $(G_1, ..., G_k)$.

From Bayes rule, the posterior probability of a configuration given the data is:

$$P(G_1, ..., G_k | D_1, ..., D_k) \propto P(G_1, ..., G_k) P(D_1, ..., D_k | G_1, ..., G_k) \tag{3}$$

The marginal likelihood $P(D_1, ..., D_k | G_1, ..., G_k)$ is computed by integrating over all parameter values for all the k networks:

$$\begin{aligned} P(D_1, ..., D_k | G_1, ..., G_k) &= \int P(D_1, ..., D_k | G_1, ..., G_k, \theta_1, ..., \theta_k) \cdot P(\theta_1, ..., \theta_k | G_1, ..., G_k) d\theta_1 ... d\theta_k \\ &= \int P(\theta_1, ..., \theta_k | G_1, ..., G_k) \prod_{p=1}^{k} P(D_p | G_p, \theta_p) d\theta_1 ... d\theta_k \end{aligned} \tag{4}$$

If we make the parameters of different networks independent *a priori* (i.e. $P(\theta_1, ..., \theta_k | G_1, ..., G_k) = P(\theta_1 | G_1) ... P(\theta_k | G_k)$), the marginal likelihood factorizes into the product of the marginal likelihoods of each data set given its network structure. In this case the posterior probability of a configuration is:

$$P(G_1, .., G_k | D_1, .., D_k) \propto P(G_1, .., G_k) \prod_{p=1}^{k} P(D_p | G_p) \tag{5}$$

Making the parameters independent *a priori* is unfortunate, and contradicts the intuition that related problems should have similar parameters, but it is needed in order to make the learning efficient (see Section 3.3). Note that this is not a restriction on the model. Unlike Naive Bayes for instance, where the attribute independence assumption restricts the class of models that can be learned, here the learned parameters will be similar if the data supports it. The only downside of making the parameters independent *a priori* is that it prevents multi-task structure learning from taking advantage of the similarities between the parameters during the structure learning phase. After the structures have been learned, however, such similarities could be leveraged to learn more accurate parameters. Finding ways to allow for some *a priori* parameter dependence while still maintaining computational efficiency is an interesting direction for future work.

3.1. The Prior

The prior knowledge of how related the different problems are and how similar their structures should be is encoded in the prior $P(G_1, ..., G_k)$. If there is no reason to believe that the structures for each task should be related, then $G_1, ..., G_k$ should be made independent *a priori* (i.e. $P(G_1, ..., G_k) = P(G_1) \cdot ... \cdot P(G_k)$). In this case the structure-learning can be performed independently on each problem.

At the other extreme, if the structures for all the different tasks should be identical, the prior $P(G_1, ..., G_k)$ should put zero probability on any configuration that contains non-identical structures. In this case one can efficiently learn the same structure for all tasks by creating a new data set with attributes $X_1, ..., X_n, TSK$, where TSK encodes the problem each case is coming from.[2] Then learn the structure for this new data set under the restriction that TSK is always the parent of all the other nodes. The common structure for all the problems is exactly the learned structure, with the node TSK and all the arcs connected to it removed. This approach, however, does not easily generalize to the case where the problems have only partial overlap in their attributes.

Between these two extremes, the prior should encourage configurations with similar network structures. One way to generate such a prior for two structures is to penalize each arc (X_i, X_j) that is present in one structure but not in the other by a constant $\delta \in [0, 1]$:

$$P(G_1, G_2) = Z_\delta \cdot (P(G_1)P(G_2))^{\frac{1}{1+\delta}} \prod_{\substack{(X_i,X_j)\in \\ G_1 \Delta G_2}} (1 - \delta) \tag{6}$$

where Z_δ is a normalization constant and $G_1 \Delta G_2$ represents the symmetric difference between the edge sets of the two DAGs (in case some variables are only present in one of the tasks, arcs connected to such variables are not counted).

If $\delta = 0$ then $P(G_1, G_2) = P(G_1)P(G_2)$, so the structures are learned independently. If $\delta = 1$ then $P(G_1, G_2) = \sqrt{P(G)P(G)} = P(G)$ for $G_1 = G_2 = G$ and $P(G_1, G_2) = 0$ for $G_1 \neq G_2$, leading to learning identical structures for all problems. For δ between 0 and 1, the higher the penalty, the higher the probability of more similar structures. The advantage of this prior is that $P(G_1)$ and $P(G_2)$ can be any structure priors that are appropriate for the task at hand.

One way to interpret the above prior is that it penalizes by δ each *edit* (i.e. arc addition, arc removal or arc reversal) that is necessary to make the two structures identical (arc reversals can count as one or two edits). This leads to a natural extension to more than two tasks: penalizes each edit that is necessary to obtain a set of identical structures:

$$P(G_1, ..., G_k) = Z_{\delta,k} \cdot \prod_{1 \leq s \leq k} P(G_s)^{\frac{1}{1+(k-1)\delta}} \times \prod_{i,j} (1 - \delta)^{edits_{i,j}} \tag{7}$$

where $edits_{i,j}$ is the minimum number of edits necessary to make the arc between X_i and X_j the same in all the structures. We will call this prior the *Edit* prior. The exponent $1/(1 + (k - 1)\delta)$ is used to transition smoothly between the case where structures should be independent (i.e. $P(G_1, ..., G_k) = (P(G_1)...P(G_k))^1$ for $\delta = 0$) and the case where

2. This is different from pooling the data, which would mean that not only the structures, but also the parameters will be identical for all problems.

structures should be identical (i.e. $P(G,..,G) = (P(G)...P(G))^{1/k}$ for $\delta = 1$). This prior can be easily generalized by using different penalties for different edges, and/or different penalties for different edit operations.

Another way to specify a prior in configurations for more than two tasks is to multiply the penalties incurred between all pairs of structures:

$$P(G_1,...,G_k) = Z_{\delta,k} \cdot \prod_{1 \le s \le k} P(G_s)^{\frac{1}{1+(k-1)\delta}} \times \prod_{1 \le s < t \le k} \left(\prod_{\substack{(X_i,X_j) \in \\ G_s \Delta G_t}} (1-\delta) \right)^{\frac{1}{k-1}} \quad (8)$$

We will call this prior the *Paired* prior. The exponent $1/(k-1)$ is used because each individual structure is involved in $k-1$ terms (one for each other structure).

One advantage that the Paired prior has over the Edit prior is that it can be generalized by specifying different penalties between different pairs of structures. This can handle situations where there is reason to believe that Task1 is related to Task2, and Task2 is related to Task3, but the relationship to between Task1 and Task3 is weaker.

There are, of course, other priors that encourage finding similar networks for each task in different ways. In particular, if the process that generated the related tasks is know, it might be possible to design a suitable prior.

3.2. Greedy Structure Learning

Treating $P(G_1,...,G_k|D_1,...,D_k)$ as a score, we can search for a high scoring configuration using an heuristic search algorithm. If we choose to use greedy search for example, we start from an initial configuration, compute the scores of the neighboring configurations, then move to the configuration that has the highest score. The search ends when no neighboring configuration has a higher score than the current one.

One question remains: what do we mean by the neighborhood of a configuration? An intuitive definition of a neighbor is the configuration obtained by modifying a single arc in a single DAG in the configuration, such that the resulting graph is still a DAG. With this definition, the size of the neighborhood of a configuration is $O(k * n^2)$ for k problems and n variables. Unfortunately, this definition introduces a lot of local minima in the search space and leads to significant loss in performance. Consider for example the case where there is a strong belief that the structures should be similar (i.e. the penalty parameter of the prior, δ, is near one resulting in a prior probability near zero when the structures in the configuration differ). In this case it would be difficult to take any steps in the greedy search since modifying a single edge for a single DAG would make it different from the other DAGs, resulting in a very low posterior probability (score).

To correct this problem, we have to allow all structures to change at the same time. Thus, we will define the neighborhood of a configuration to be the set of all configurations obtained by changing the same arc in any subset of the structures. Examples of such changes are removing an existing arc from all the structures, or just removing it from half of the structures, or removing it from one structure, reverse it in another, and leave it unchanged in the rest. This way we avoid creating local minimas in the search space while still ensuring that every possible configuration can be reached. Given this definition,

the size of a neighborhood is $O(n^2 3^k)$, which is exponential in the number of problems, but only quadratic in the number of nodes.[3] When setting $\delta = 1$, leading to learning identical structures, multi-task learning with this definition of neighborhood will find the same structures as the specialized algorithm described in Section 3.1.

3.3. Searching for the Best Configuration

At each iteration, the greedy procedure described in the previous section must find the best scoring configuration from a set $\mathcal{N}$ of neighboring configurations. In a naive implementation, the score of every configuration in $\mathcal{N}$ has to be computed to find the best one, which can quickly become computationally infeasible given our definition of neighborhood.

In this section we show how one can use branch-and-bound techniques to find the best scoring configuration without evaluating all configurations in $\mathcal{N}$. Let a partial configuration of order l, $\mathcal{C}_l = (G_1, .., G_l)$, be a configuration where only the *first* l structures are specified and the rest of $k - l$ structures are not specified. We say that a configuration $\mathcal{C}$ matches a partial configuration $\mathcal{C}_l$ if the first l structures in $\mathcal{C}$ are the same as the structures in $\mathcal{C}_l$.

A search strategy for finding the best scoring configuration in $\mathcal{N}$ can be represented via a search tree of depth k with the following properties: a) each node at level l contains a valid partial configuration of order l; b) all nodes in the subtree rooted at node $\mathcal{C}_l$ contain only (partial) configurations that match $\mathcal{C}_l$ (i.e. the first l structures are the same as in $\mathcal{C}_l$.)

If, given a partial configuration, the score of any complete configuration that matches it can be efficiently upper bounded, and the upper bound is lower than the current best score, then the entire subtree rooted at the respective partial configuration can be pruned.

Let $edits_{l,i,j}$ be the minimum number of edits necessary to make the arc between X_i and X_j the same in the *first* l structures, and let $Best_q = \max_{G_q}\{P(G_q)^{\frac{1}{1+(k-1)\delta}} P(D_q|G_q)\}$. If the marginal likelihood of a configuration factorizes in the product of the marginal likelihoods of the individual structures, as in equation 5, then the score of any configuration that matches the partial configuration $\mathcal{C}_l = (G_1, ..., G_l)$ can be upper bounded by:

$$U^E_{\mathcal{N}}(\mathcal{C}_l) = \left(\prod_{i,j}(1-\delta)^{edits_{l,i,j}}\right) \cdot \left(\prod_{1\le p\le l} P(G_p)^{\frac{1}{1+(k-1)\delta}} P(D_p|G_p)\right) \cdot \left(\prod_{l+1\le q\le k} Best_q\right) \quad (9)$$

if using the Edit prior (equation 7), and by

$$U^P_{\mathcal{N}}(\mathcal{C}_l) = \left(\prod_{1\le s<t\le l}\ \prod_{\substack{(X_i,X_j)\in \\ G_s \Delta G_t}} (1-\delta)\right)^{\frac{1}{k-1}} \cdot \left(\prod_{1\le p\le l} P(G_p)^{\frac{1}{1+(k-1)\delta}} P(D_p|G_p)\right) \cdot \left(\prod_{l+1\le q\le k} Best_q\right) \quad (10)$$

if using the Paired prior (equation 8).

3. The restriction that changes, if any, have to be made to the same arc in all structures could be dropped, but this would lead to a neighborhood that is exponential in both n and k. Considering the assumption that the structures should be similar, such a restriction is not inappropriate.

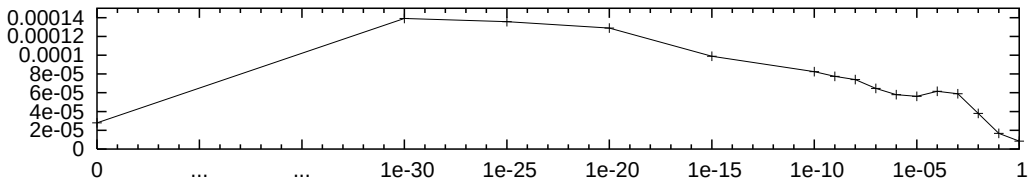

Figure 1: Fraction of partial configurations evaluated as a function of the penalty

This branch and bound search significantly reduces the number of partial configurations (and consequently complete configurations) that need to be explored. As an example of how much the branch and bound search can help, Figure 1 shows the fraction of configurations that are evaluated by branch and bound as the multi-task penalty parameter δ is varied, for a problem with five tasks and thirty seven variables. In this case, branch and bound evaluates four orders of magnitude less configurations than a naive search would.

4. Experimental Results

We evaluate the performance of multi-task structure learning using multi-task problems generated by perturbing the ALARM (Beinlich et al., 1989) and INSURANCE (Binder et al., 1997) networks, and on a real problem in bird ecology.

Multi-task structure learning is compared to single-task structure learning, and learning identical structures for all tasks. Single-task structure learning uses greedy hill-climbing with 100 restarts and tabu lists to learn the structure of each task independently of the others. The learning of identical structures is performed via the algorithm presented in Section 3.1 and also uses greedy hillclimbing with 100 restarts and tabu lists.[4] Multi-task structure learning uses the greedy algorithm described in Section 3.2 with the solution found by single-task learning as the starting point.[5] The penalty parameter of the multi-task prior, δ, is selected from a logarithmic grid to maximize the mean log-likelihood of a small validation set. For all methods, the Bayes net parameters are learned using Bayesian updating ((see e.g. Cooper and Hersovits, 1992)) independently for each problem.

The goal is to recover as closely as possible the true Bayes Net structures for all the related tasks, so the main measure of performance we use is average edit distance[6] between the true structures and learned structures. Edit distance directly measures the quality of the learned structures, independently of the parameters of the Bayes Net. We also measure the average empirical KL-divergence (computed on a large test set) between the distributions encoded by the true networks and the learned ones. Since KL-Divergence is

4. Learning identical structures and single-task structure learning can be viewed as learning an augmented naive Bayesian network and a Bayesian multi-net (Friedman et al., 1997) respectively, where the "class" of each example is the task it belongs to . Unlike in the usual setting, however, here we are not interested in predicting to which task an example belongs to. We are only interested in recovering accurate network structures for each task.
5. Initializing MTL search with the STL solution does not provide an advantage to MTL, but makes the search more efficient.
6. Edit distance measures how many edits (arc additions, deletions or reversals) are needed to get from one structure to the other.

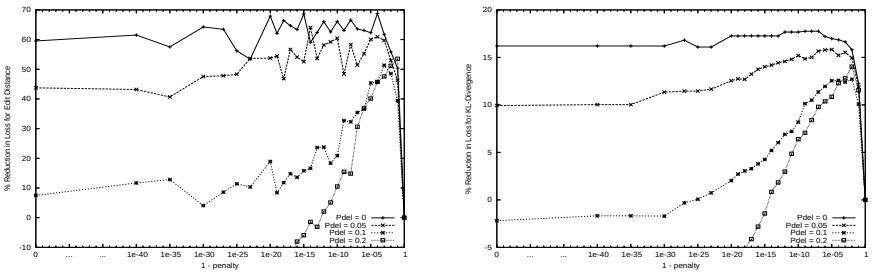

Figure 2: Reduction in edit distance (left) and KL-Divergence (right) for ALARM

also sensitive to the parameters of the Bayes Net it does not measure directly the quality of the learned structures, but, in general, more accurate structures should lead to models with lower KL-Divergence. For the bird ecology problem, where the true networks are unknown, we measure performance in terms of mean log likelihood on a large independent test set.

4.1. The ALARM and INSURANCE problems

For the experiments with the ALARM and INSURANCE networks, we generate five related tasks by perturbing the original structures. We use two qualitatively different methods for perturbing the networks: randomly deleting edges with some probability, and changing entire subgraphs. In the first case, we create five related tasks by starting with the original network and deleting arcs with probability P_{del}. This way, the structures of the five tasks can be made more or less similar by varying P_{del} (For $P_{del} = 0$ all the structures are identical. Given the restriction we imposed in Section 3 that parameters for different tasks should be independent *a priori*, we want to investigate the performance of multi-task structure learning in settings where the parameters are indeed independent between tasks (ALARM-IND and INSURANCE-IND), as well as in settings where the parameters are actually correlated between tasks (ALARM and INSURANCE).

We also experiment with a qualitatively different way of generating related tasks (ALARM-COMP). We split the ALARM network in 4 subgraphs with disjoint sets of nodes. For each of the five tasks, we randomly change the structure and parameters of zero, one or two of the subgraphs, while keeping the rest of the Bayes net (including parameters) unchanged. This way parts of the structures are shared between tasks while other parts are completely unrelated.

Figures 2 and 3 show the average percent reduction in loss, in terms of edit distance and KL-divergence, achieved by multi-task learning over single-task learning for a training set of 1000 points on the ALARM and INSURANCE-IND problems. The figures for the ALARM-IND, ALARM-COMP, and INSURANCE problems are similar and are not included. On the x-axis we vary the penalty parameter of the multi-task prior on a log-scale. Note that the x-axis plots $1-\delta$. The higher the penalty (the lower $1-\delta$), the more similar the learned structures will be, with all the structures being identical for a penalty of one ($1-\delta = 0$, left end of graphs). Each line in the figure corresponds to a particular value of P_{del}. Error bars are omitted to maintain the figure readable.

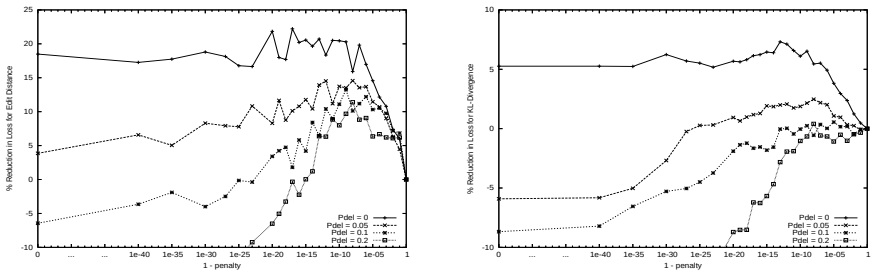

Figure 3: Reduction in edit distance (left) and KL-Divergence (right) for INSURANCE-IND

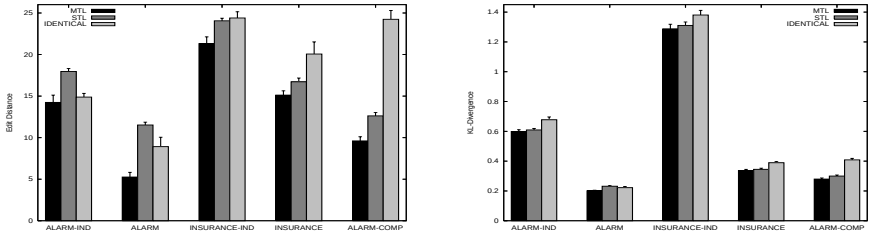

Figure 4: Edit distance (left) and KL-Div (right) for STL, learning identical structures and MTL

The trends in the graphs are exactly as expected. For all values of P_{del}, as the penalty increases, the performance increases because the learning algorithm takes into account information from the other tasks when deciding whether to add a new arc or not. If the penalty is too high, however, the algorithm loses the ability to find true differences between tasks and the performance drops. As the tasks become more similar (lower values of P_{del}), the best performance is obtained at higher penalties. Also as the tasks become more similar, more information can be extracted from the related tasks, so usually multi-task learning provides more benefit. Multi-task learning provides similar benefits whether the tasks have highly correlated parameters (ALARM and INSURANCE problems) or independent parameters (ALARM-IND and INSURANCE-IND problems). This shows that making the parameters independent *a priori* (see Section 3) does not hurt the performance of multi-task learning.

One thing to note is that multi-task structure learning provides a larger relative improvement in edit distance than in KL-divergence. This happens because multi-task structure learning helps to correctly identify the arcs that encode weaker dependencies (or independences) which have a smaller effect on KL-divergence. The arcs that encode strong dependencies, and have the biggest effect on KL-divergence, can be easily learned without help from the other tasks.

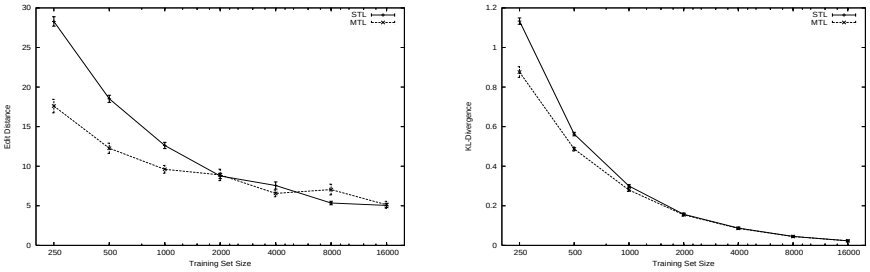

Figure 5: Edit distance (left) and KL-Divergence (right) vs. train set size for ALARM-COMP.

Figure 4 shows the edit distance and KL-Divergence performance for single task learning (STL), learning identical networks (IDENTICAL), and multi-task learning (MTL). The training set has 1000 instances with 50 instances used to select the penalty parameter for the multi-task prior. Single-task learning and identical structure learning use all the data for learning since they do not have free parameters. The figure shows that multi-task learning yields a 10%-54% reduction in edit distance and a 2% - 13% reduction in KL-divergence when compared to single task structure learning. All differences except for KL-divergence on ALARM-IND and INSURANCE-IND problems are .95 significant according to paired T-tests. When compared to learning identical structures, multi-task learning reduces the KL-divergence 7% - 32% and the number of incorrect arcs in the learned structures by 4% - 60%. All differences are .95 significant, except for edit distance on the ALARM-IND problem. Since the five tasks for the ALARM, INSURANCE, and ALARM-COMP problems share a large number of their parameters, one might believe that simply pooling the data would work well. This is, however, not the case. Except for the ALARM problem, where it achieves about the same edit distance as learning identical structures, pooling the data has much worse performance both in terms of edit distance and in terms of KL-divergence.

Figure 5 shows the performance of single and multi-task learning as the train set size varies from 250 to 16000 cases (MTL uses 5% of the training points as a validation set to select the penalty parameter). As expected, the benefit from multi-task learning is larger when the data is scarce and it diminishes as more training data is available. This is consistent with the behavior of multi-task learning in other learning setting (see e.g. (Caruana, 1997)). For smaller training set sizes multi-task learning needs about half as much data as single-task learning to achieve the same edit distance. In terms of KL-divergence, multi-task learning provides smaller savings in sample size. One reason for this is that, as discussed before, multi-task learning yields lower improvements in KL-divergence than in edit distance. For the most part however, the smaller savings in sample size are due to the fact that more training data leads not only to more accurate structures, but also to more accurate parameters. Since multi-task structure learning only improves the structure and not the parameters, it is not able to *make up* for the loss of large amounts of training data.

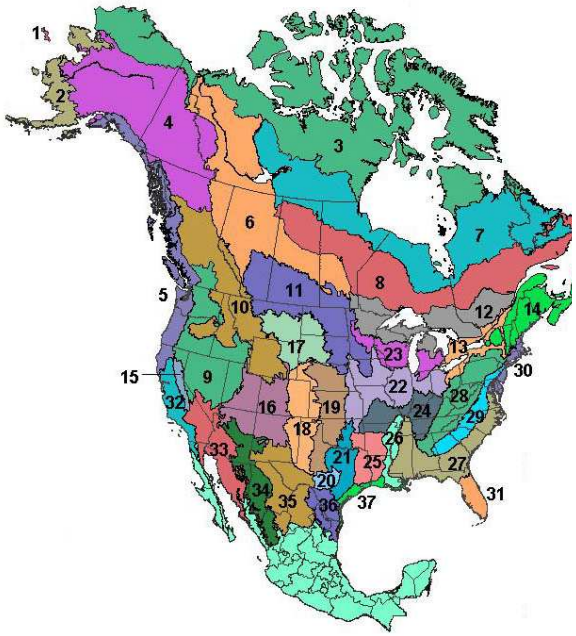

Figure 6: North American Bird Conservation Regions.

4.2. The Bird Ecology Problem

We also evaluate the performance of multi-task structure learning on a real bird ecology problem. The data for this problem comes from Project FeederWatch (PFW)[7], a winter-long survey of North American birds observed at bird feeders. Each PFW submission is described by multiple attributes, which can be roughly grouped into features related to which birds have been observed, observer effort, weather during the observation period, and attractiveness of the location and neighborhood area for birds. The goal is to gain a better understanding of the various bird species by identifying environmental factors that attract or detract certain bird species, as well as how different bird species interact with each other.

Ecologists have divided North America into a number of ecologically distinct Bird Conservation Regions (BCRs; see Figure 6). This division naturally splits the data into multiple tasks, one task per BCR. For the results in this section we use six related tasks corresponding to BCRs 30, 29, 28, 22, 13 and 23. Because each bird species lives in some BCRs but not in others, this is an instance of a problem where the different tasks are not defined over identical sets of variables.

The results on the BIRD problem mimic the ones in the previous section. Figure 7 shows the average (across the 6 BCRs/tasks) mean log likelihood on a large independent test set for multi-task structure learning as a function of the penalty parameter of the

7. http://birds.cornell.edu/pfw

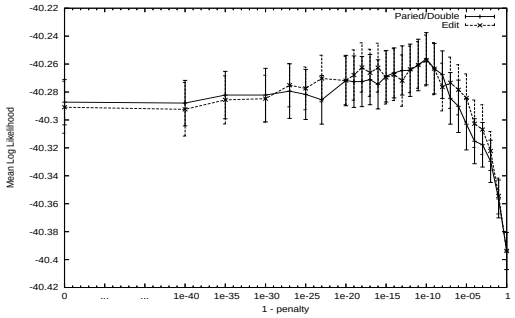

Figure 7: Average mean log likelihood vs. the penalty parameter for multi-task structure learning on the BIRD problem.

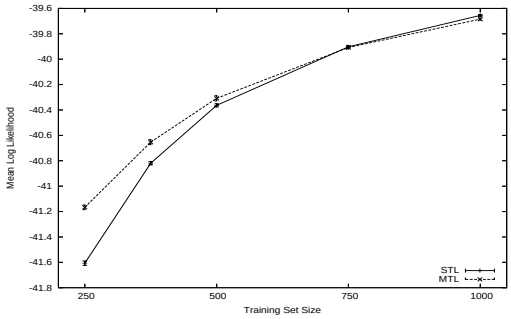

Figure 8: Average mean log likelihood vs. training set size for the BIRD problem.

multi-task prior. Each line corresponds to a different type of multi-task prior. The x-axis plots $1 - \delta$, so the right most point corresponds to no penalty (single task learning) and the leftmost point corresponds to a penalty of one (learning identical structures). Higher mean log likelihood represents better performance. As the penalty parameter increases ($1 - \delta$ decreases), information starts to be transferred between the different tasks and the performance quickly increases. After reaching a peak, the performance starts to decrease slowly as the penalty increases further. Since the tasks are not all defined on the same set of variables, the algorithm for learning identical structures for all tasks from Section 3.1 can not be easily applied. Our algorithm on the other hand can handle this situation and learns a set of identical structures for all tasks that performs reasonably well (left end of the plot). The type of multi-task prior does not have a significant impact on the performance for this problem.

Figure 8 shows the average mean log likelihood performance of multi-task structure learning and single task structure learning as a function of the training set size. Multi-task learning uses 5% of the training data to select the penalty parameter for the multi-task

prior. Again, the benefit from multi-task learning is largest for smaller training set sizes. As the training size increases single-task learning catches up and eventually outperforms multi-task learning. Unfortunately, since we do not know the real network structures for this problem, we can not directly asses the quality of the learned structures. The results on the ALARM and INSURANCE problems, however, suggest that the improvement provided by multi-task learning in terms of structural accuracy (edit distance) would probably be even larger than the improvement in terms of average mean log likelihood.

5. Conclusions and Discussion

Learning the structure of Bayes Nets from data has received a lot of attention in the literature and numerous techniques have been developed to solve this problem (e.g. (Cooper and Hersovits, 1992; Heckerman, 1999; Buntine, 1996; Spirtes et al., 2000)). In this paper, we have focused on, arguably, the most basic one: score-and-search using greedy hill-climbing in the space of network structures (DAG-search), and extended this technique to the multi-task learning scenario. The key ingredients in achieving this have been: defining a principled scoring function that takes into account the data from all the tasks and encourages the learning of similar structures, defining a suitable search space, and devising a branch and bound procedure that enables efficient moves in this search space. We experimented with perturbed ALARM and INSURANCE networks and a real bird ecology problem, and showed that the multi-task structure learning technique yields significantly more accurate Bayes Net structures, especially when training data is scarce.

Even though in the paper we have focused on DAG-search, one can straightforwardly obtain multi-task Bayes Net structure learning algorithms based on other techniques such as greedy search in the space of equivalence classes (Chickering, 1996), obtaining confidence measures on the structural features of the configurations via bootstrap analysis (Friedman et al., 1999), and structure learning from incomplete datasets via the structural EM algorithm (Friedman, 1998). Other extensions such as obtaining a sample from the posterior distribution via MCMC methods might be more problematic. Because of the larger search space, MCMC methods might not converge in reasonable time. Evaluating different MCMC schemes is a direction for future work.

Another open question is whether we can relax the requirement that the parameters of the Bayes Nets for the different related tasks are independent *a priori*. Relaxing this requirement might further improve the performance of multi-task learning since the task would be able to share not only the structures but also the parameters, thus having more opportunities for inductive transfer. Further improvement is also possible by eliminating the need for the user to specify the penalty parameter δ. At this point, one has to rely on cross-validation to determine a reasonable value for this parameter, which leads to a loss in performance and an increase in computational time. It would be very desirable to find techniques to infer δ directly from the data, or integrate over it in a bayesian manner.

Multi-task structure learning might also prove useful in learning Bayesian multi-nets (Friedman et al., 1997). In Bayesian multi-nets a special attribute is selected (usually the class attribute), and a separate network is learned for each value of that attribute. To the best of our knowledge, all work in learning Bayesian multi-nets treats each separate network as an independent learning problem, in a single-task manner. Since it is reasonable

to assume that the networks for the different values of the class attribute should be similar, learning all the networks jointly using multi-task structure learning might yield improved performance.

References

J. Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Mach. Learn.*, 28(1):7–39, 1997. ISSN 0885-6125.

I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. 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*, 1989.

J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29, 1997.

W. Buntine. Theory refinement on bayesian networks. In *Proc. 7th Conference on Uncertainty in Artificial Intelligence (UAI '91)*, 1991.

W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Trans. On Knowledge and data Engineering*, 8:195–210, 1996.

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

D. Chickering. Learning equivalence classes of Bayesian network structures. In *Proc. 12th Conference on Uncertainty in Artificial Intelligence (UAI'96)*, 1996. ISBN 1-55860-412-X.

G. Cooper and E. Hersovits. A bayesian method for the induction of probabilistic networks from data. *Maching Learning*, 9:309–347, 1992.

N. Friedman. The Bayesian structural EM algorithm. In *Proc. 14th Conference on Uncertainty in Artificial Intelligence (UAI '98)*, 1998.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(2):131–163, 1997.

N. Friedman, M. Goldszmidt, and A. J. Wyner. Data analysis with bayesian networks: A bootstrap approach. In *Proc. 15th Conference on Uncertainty in Artificial Intelligence*, 1999.

D. Heckerman. A tutorial on learning with bayesian networks. *Learning in graphical models*, pages 301–354, 1999.

D. Heckerman, A. Mamdani, and M.P. Wellman. Real-world applications of Bayesian networks. *Communications of the ACM*, 38(3):24–30, 1995.

A. Niculescu-Mizil and R. Caruana. Inductive transfer for bayesian network structure learning. In *Proc. 11th International Conf. on AI and Statistics*, 2007.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, Cambridge, MA, second edition, 2000.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-first Conference on Uncertainty in AI (UAI)*, pages 584–590, Edinburgh, Scottland, UK, July 2005.

S. Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, 1996.