Learning the Structure of Related Tasks

Alexandru Niculescu-Mizil
Department of Computer Science
Cornell University
Ithaca, NY 14853
alexn@cs.cornell.edu

Rich Caruana
Department of Computer Science
Cornell University
Ithaca, NY 14853
caruana@cs.cornell.edu

Abstract

We consider the problem of learning Bayes Net structures for related tasks. We present a formalism for learning related Bayes Net structures that takes advantage of the similarity between tasks by biasing toward learning similar structures for each task. Heuristic search is used to find a high scoring *set* of structures (one for each task), where the *score* for a set of structures is computed in a principled way. Experiments on synthetic problems generated from the ALARM network show that learning the related structures using the proposed method yields better results than learning the structures independently.

1 Introduction

Bayes Nets [1] provide a way to understand data by revealing the statistical dependence between variables. Bayes Nets provide a compact, intuitive description of the dependency structure of a domain by using a directed acyclic graph to encode probabilistic dependencies between variables. This dependency structure can be automatically learned from data, and can be very useful for gaining a better understanding of the problem at hand. For example, Friedman et al. [2] used Bayes Nets learned from gene expression level data to discover regulatory interactions between genes for one type of yeast.

In this paper we focus on simultaneously learning Bayes Net structures for multiple problems. Multi-task learning [3, 4, 5] suggests that if different problems are related, learning structures for the problems together should provide an advantage because what is learned for one problem may be useful for the other problems. For example, suppose that gene expression data is available for a number of different species of yeast. It is reasonable to assume that the regulatory interactions between genes for all yeast are similar, but not identical. Learning that there is an interaction between two genes for one species of yeast should provide evidence for interaction between the two genes in other species of yeast.

2 Learning the Structure of Related Tasks

A Bayesian Network $\mathcal{B} = \{G, \theta\}$ that encodes the joint probability distribution of a set of n random variables $X = \{X_1, X_2, ..., X_n\}$ is specified by a directed acyclic graph (DAG) G and a set of conditional probability functions parametrized by θ [1]. 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 understand and can be used to gain better understanding of the problem. In this section we propose an algorithm for learning the Bayes Net structure for related tasks

Given the i.i.d data-sets $D_1, ..., D_k$, we want to simultaneously learn the structures of the Bayes Nets $\mathcal{B}_1 = \{G_1, \theta_1\}, ..., \mathcal{B}_k = \{G_k, \theta_k\}$. We will use the term *configuration* to refer to an ordered set of structures $(G_1, ..., G_k)$. The posterior probability of a configuration is:

$$P(G_1, ..., G_k|D_1, ..., D_k) \propto P(G_1, ..., G_k)P(D_1, ..., D_k|G_1, ..., G_k) \tag{1}$$

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:

$$\begin{aligned} &P(D_1, ..., D_k|G_1, ..., G_k) = \\ &= \int P(D_1, ..., D_k|G_1, ..., G_k, \theta_1, ..., \theta_k)P(\theta_1, ..., \theta_k|G_1, ..., G_k)d\theta_1...d\theta_k \\ &= \int P(\theta_1, ..., \theta_k|G_1, ..., G_k)\prod_{p=1}^{k} P(D_p|G_p, \theta_p)d\theta_1...d\theta_k \end{aligned} \tag{2}$$

If we make the simplifying assumption that the parameters of different networks are independent apriori (i.e. $P(\theta_1, ..., \theta_k|G_1, ..., G_k) = P(\theta_1|G_1)...P(\theta_k|G_k)$), the marginal likelihood is just the product of the marginal likelihoods of each data set given its network structure. [1] In this case the posterior can be written as:

$$P(G_1, ..., G_k|D_1, ..., D_k) \propto P(G_1, ..., G_k)\prod_{p=1}^{k} P(D_p|G_p) \tag{3}$$

For simplicity, we also assume that all tasks have the same attributes $X_1, ..., X_n$. Everything that follows can be easily extended to handle tasks that do not have all attributes in common.

2.1 The Prior

The prior knowledge of how related the different tasks are and how similar their structures should be is encoded in the prior $P(G_1, ..., G_k)$. If there is reason to believe that the structures for each task are related, the prior should encourage finding similar network structures for the tasks. The prior can be seen as penalizing structures that deviate from each other, so that deviation will occur only if it is supported by enough evidence in the data.

One way to generate such a prior for two structures is to independently penalize each arc (X_i, X_j) that is present in one structure but not in the other by a constant $\delta_{ij} \in [0, 1]$:

$$P(G_1, G_2) = Z_\delta P(G_1)P(G_2) \prod_{\substack{(X_i, X_j)\in \\ G_1 \Delta G_2}} (1 - \delta_{ij}) \tag{4}$$

Assuming that the penalty for each arc is the same, only one parameter, δ, needs to be specified. If $\delta = 0$ the structures are independent. If $\delta = 1$ only configurations with identical structures have probability mass. 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 kind of structure prior that is appropriate for the task at hand. Z_δ is a normalization factor that is absorbed in the proportionality constant of equation 3.

In the case of more than two tasks an intuitive prior is:

$$P(G_1, ..., G_k) = Z \left(\prod_{1\le i<j\le k} P(G_i, G_j) \right)^{\frac{1}{k-1}} \tag{5}$$

2.2 Greedy Structure Learning

As in the case of learning the structure of a single network [6], we treat $P(G_1, ..., G_k|D_1, ..., D_k)$ as a score, and search for a high scoring configuration using

[1] While this assumption is clearly not true, we need it in order to make structure learning efficient.

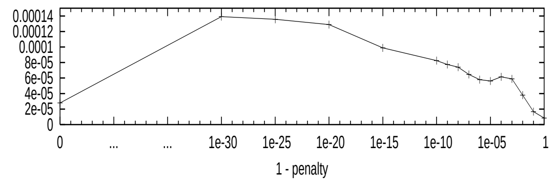

Figure 1: Fraction of partial configurations evaluated as a function of the penalty

some heuristic search algorithm. If we chose to use greedy search for example, we start from a 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? Let the extended neighborhood of a single structure be the set of all the DAGs that can be obtained by removing or reversing an existing arc in G, or by adding an arc that is not present in G, plus the structure itself. Then the neighborhood of a configuration could be defined as the cross-product of the extended neighborhoods of each structure in the configuration.

This definition of neighborhood is appealing because in the special cases when the structures for all the tasks are considered independent (zero penalty for diverging structures) multi-task structure learning will produce the same results as if the structures are learned independently. Unfortunately the size of such neighborhoods is of order of $O((n^2)^k)$ for k tasks and n attributes. Computing the score of all configurations in this neighborhood quickly becomes prohibitive when either n or k is large. In practice however, since we are only interested in the best scoring configuration in this neighborhood, we do not need to compute the scores of configurations that have no chance of having the best score, so only a (hopefully) small fraction of configurations needs to be evaluated. If the parameters for each task are mutually independent apriori so the score of a configuration has the form in equation 3, and the prior over configurations has the form in equation 5 it is possible to significantly reduce the number of configurations that need to be evaluated. For example, on the problem described in the next section the number of evaluated structures was reduced by a factor of 10^4. See Figure 1. Moreover, if a configuration score has the form in equation 3, we can precompute the marginal likelihoods for all structures in $\mathcal{G}_1, ..., \mathcal{G}_k$, avoiding this expensive computation when evaluating different configurations.

3 Experimental Results

This section provides a proof of concept for the proposed method. To create related tasks with similar structures, we generate five similar Bayes Nets. For each net, we start from the the ALARM [7] network and randomly delete arcs with probability P_{del}. We then generate random parameters for each of the five Bayes Nets. This yields five related tasks with independent parameters whose structures can be made more or less similar by varying P_{del} (For $P_{del} = 0$ all the structures are identical). Five datasets (one for each task) are generated from the joint distribution encoded by each of the five Bayes Nets.

The goal is to recover as closely as possible the structure of all five Bayes Nets We measure performance in terms of average hamming distance between the true structures and learned structures. Structures are learned with the algorithm described in Section 2.2. We use the prior over configurations described in equation 5 with the same penalty for all arcs.

The left graph in figure 2 shows the average improvement of multi-task structure learning over learning the structures individually for a train set size of 1000 points. A +4 on the y-axis means that on average 4 more arcs are correctly learned by the multi-task method than by single task structure learning. On the x-axis we vary the penalty parameter of the prior. A penalty of zero (right end of graphs) is equivalent to learning the structures individually. A penalty of one (left end of graphs) means that the structures for all tasks must be identical. Each line in the figure corresponds to a particular value of P_{del}. The graphs for all values of P_{del} have similar behavior. As the penalty increases, the performance increases because the learning algorithm takes into

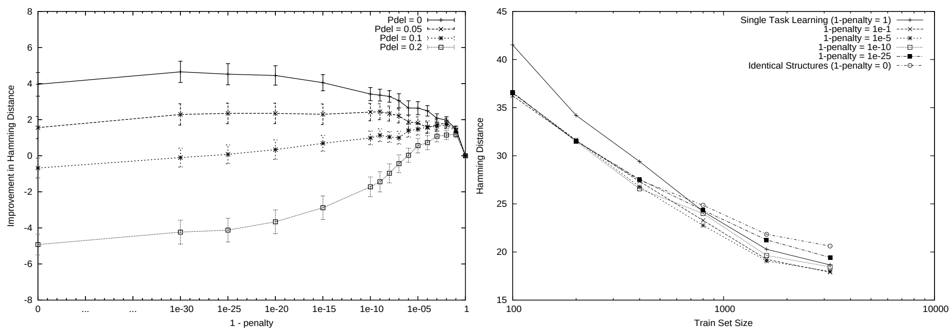

Figure 2: Improvement over single task learning (Left); Learning curves for different penalty factors (Right)

account information from the other tasks when deciding whether to add a new arc or not. If he penalty is to high, however, the algorithm loses the ability of finding true differences between the different tasks and the performance drops (unless of course the $P_{del} = 0$). The trends in the graphs are exactly as expected. As the tasks become more similar, 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 multi-task learning provides more benefit.

The right graph in figure 2 shows learning curves for different penalty values when $P_{del} = 0.1$. The smaller the training set the more benefit from multi-task learning, with single task learning catching up for higher train set sizes. At small training set sizes there is too little data to overcome the penalty, so the learned structures will be very similar for any penalty. As the training set increases, there is enough evidence to overcome the prior and learn differences between the structures for each task. With enough data, high penalties perform worse because they are not flexible enough to allow true differences between the structures to be learned.

4 Summary

We present a method for learning Bayes Net structures of related tasks. The approach assumes that the structures of related tasks are similar: the presence or absence of arcs in some of the structures provides evidence for the presence or absence of those arcs in the other structures. When this assumption is true, learning the structures together provides an advantage over learning a structure for each task individually. Similarity between learned structures is controlled via a prior. The posterior probability of a set of structures given the data is used by heuristic search to efficiently find high-scoring sets of structures. Experiments with synthetic data from perturbed ALARM networks show that learning related structures simultaneously yields 1-6 additional correct arcs in the learned Bayes Nets when compared to structures learned separately.

References

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

[2] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *J. Comput. Biol.*, 7(3-4):601–620, 2000.

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

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

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

[6] D. Heckerman. A tutorial on learning with bayesian networks, 1995.

[7] 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.