Finding Optimal Bayesian Networks with Local Structure

Topi Talvitie TOTALVIT@CS.HELSINKI.FI
Ralf Eggeling EGGELING@CS.HELSINKI.FI
Mikko Koivisto MKHKOIVI@CS.HELSINKI.FI
Department of Computer Science, University of Helsinki, 00014 Helsinki, Finland

Abstract

The idea of using decision trees as local models in Bayesian networks is revisited. A class of dyadic decision trees—proposed previously only for continuous conditioning variables—is augmented by incorporating categorical variables with arbitrary context-specific recursive splitting of their state spaces. It is shown that the resulting model class admits computationally feasible maximization of a Bayes score in a range of moderate-size problem instances. In particular, it enables global optimization of the Bayesian network structure, including the local structure, using state-of-the-art exact algorithms. The paper also introduces a related model class that extends ordinary conditional probability tables to continuous variables by employing an adaptive discretization approach. The two model classes are compared empirically by learning Bayesian networks from benchmark real-world and synthetic data sets. The relative strengths of the model classes are discussed.

Keywords: Bayesian networks; decision trees; exact algorithms; structure learning.

1. Introduction

Conditional probability distributions (CPDs) play a central role in probabilistic approaches to classification, regression, and reasoning under uncertainty in general. In particular, they are a key ingredient in Bayesian networks (BNs), which compose a multivariate distribution as a product of univariate CPDs. For statistically efficient learning of a CPD, it is crucial that the number of free model parameters adapts to the amount of the data. The notion of context-specific independence (Boutilier et al., 1996) offers an appealing way to achieve this: given certain configuration for a subset of the *predictor* variables, the *response* variable is independent of the remaining predictors.

Among the most popular models of context-specific independence is the *classification and regression tree (CART)* model (Breiman et al., 1984). It employs a decision tree to partition the predictor space into regions, in each of which the CPD is constant, letting the model adapt to the complexity of the data. Unfortunately, learning CART models is computationally intractable, and so researchers and practitioners have resorted to greedy algorithms (Buntine, 1992; Quinlan, 1993) or local search (Chipman et al., 1998). While such algorithms perform well in many cases, they lack performance guarantees. This implies uncontrolled uncertainty concerning the learning results, which can hamper decision making in applications as well as methods research.

Here, we aim to remove the uncontrolled uncertainty in learning CART models, restricting ourselves to a moderate number of predictors. While such models do not apply to high-dimensional classification and regression tasks (without a separate feature selection phase), they can be valuable as local models in BNs, in which CPDs typically contain just a handful of predictors (the parents of a node). The idea was put forward by Friedman and Goldszmidt (1996), who however resorted to greedy algorithms for both finding high-scoring local tree models and global network structures, i.e.,

directed acyclic graphs (DAGs). Being able to find optimal BN structures with local CART models would allow more confident comparison of CART against ordinary *conditional probability tables (CPTs)*, and bringing CART to settings where it does boost the statistical power of BNs, without compromising exact computations. Exact algorithms not only remove the uncertainty on the quality, but also tend to perform better in structure learning and density estimation (Malone et al., 2015).

The present work stems from two observations. First, existing exact algorithms can nowadays find optimal DAGs over several dozens of variables in some hours of computation, provided that pre-computed local scores for a relatively small number of candidate parent sets are given as input; see Malone et al. (2018) and the references therein. Thus it remains to be investigated what classes of local models admit learning within about the same time budget. Second, it is known that a restriction of the CART model to so-called *dyadic CART* essentially inherits the statistical power of CART and yet enables significantly faster global optimization (Donoho, 1997; Scott and Nowak, 2006; Blanchard et al., 2007). Building on these observations, we present an exact algorithm to learn BNs with local structure, applicable to mixed sets of categorical and continuous variables. Specifically, we introduce the *partition–dyadic CART (PCART)* model; here "partition" refers to (recursive) splitting of the state spaces of categorical variables—to the best of our knowledge, categorical predictors have not been allowed in previous works on dyadic CART.

When all variables are categorical, one can compare the performance of PCART against standard CPTs in a straightforward manner. To compare PCART against analogous "full-table" models also in the presence of continuous variables, we introduce a *full-table (FT)* variant of PCART: it discretizes the involved continuous predictors (but not the response). While FT adapts the number of states per variable to the data at hand, it renders global optimization computationally feasible, unlike some more sophisticated discretization schemes (Chen et al., 2017).

Armed with these methodological developments—which are a main contribution of the paper—we also study empirically the following two questions: What kind of problem instances, in terms of model complexity and data dimensions, can be solved in a reasonable time? Does PCART boost the statistical efficiency of structure learning in practice, that is, on benchmark data sets? One could hypothesize that, thanks to a parsimonius parameterization, PCART enables more powerful detection of arcs. Our results will support this hypothesis.

2. Preliminaries

We begin by reviewing some basics of BNs and decision trees and learning them from data. We focus on the key terminology and notation used in later sections; for a more comprehensive treatment, the reader is referred to the literature on graphical models (Koller and Friedman, 2009, Chs. 3, 5, and 18) and score-based learning of decision trees (Buntine, 1992; Chipman et al., 1998).

2.1 Bayesian Networks

Consider modeling a multivariate probability distribution $p(X_1, X_2, \ldots, X_n)$, where each variable X_v takes values from some set S_v. Write V for the index set $\{1, 2, \ldots, n\}$ and index by subsets: e.g., if $P \subseteq V$ denote by X_P the tuple $(X_v)_{v \in P}$ and by S_P the Cartesian product $\times_{v \in P} S_v$.

A distribution p and a directed acyclic graph (DAG) $G = (V, A)$ form a *Bayesian network (BN)* if the joint distribution factorizes into a product of the conditional distributions (CPDs) $p(X_v | X_{G_v})$, where $G_v = \{u : (u, v) \in A\}$ is the set of *parents* of v in G. Of the various ways to parameterize a

CPD $p(X_v|X_P)$, with $P \subseteq V \setminus \{v\}$, we focus on decision trees; a special case of that is the simple conditional probability table, which applies when all ranges are finite (see Section 3.2).

2.2 Classification and Regression Trees

A decision tree of a multidimensional space is a recursive, axis-parallel, binary partition of the space. More formally, let T be a rooted binary tree where each node R is a subset of S_P of the form $\times_{u \in P} R_u$. We call T a *decision tree* of S_P if its root is S_P and every inner node R and its two children R' and R'' differ in exactly one dimension $u \in P$ for which $\{R'_u, R''_u\}$ is a partition of R_u. The "decision" associated with node R is whether the variable X_u belongs to R'_u or R''_u. Consequently, the leaves of T partition S_P. A decision tree T_v of S_P and a CPD $p(X_v|X_P)$ are *compatible* with each other if, for all leaves R, X_v is independent of X_P conditionally on the event $X_P \in R$. In other words, the CPD is constant over each leaf R and piecewise constant over S_P.

In order to parameterize the distribution of the response variable X_v in each leaf, we distinguish between a discrete and a continuous distribution. In this work, we focus on two standard cases:

Categorical distribution: If S_v is finite, we let θ_x be the probability that $X_v = x$, for each $x \in S_v$.

Normal distribution: If S_v is the set of reals, X_v is normally distributed with mean μ, variance σ^2.

We understand that the parameters are distinct for all v and all leaves of T_v, and omit emphasizing this in the notation. Depending on whether the response is categorical or continuous we call the decision tree equipped with the CPD a *classification tree* or a *regression tree*, generally a *CART*.

2.3 Structure Learning

To learn the model structure from data, we take a Bayesian score-based approach. We only note in passing that the derivations of Sections 3 and 4 also apply to other scoring schemes, especially penalized maximum-likelihood scores.

Consider a set of data points $X^i = (X_1^i, X_2^i, \ldots, X_n^i)$, $i = 1, 2, \ldots, N$. We model the data points as independent draws from a distribution p whose structure, namely a DAG $G = (G_v)_{v \in V}$ with a decision tree T_v of S_{G_v} for each $v \in V$, are unknown; the parameters of the CPDs are unknown as well. We will describe a modular prior of the structure and the parameters and obtain the score as the posterior probability of the structure, by marginalizing out the parameters.

Consider a decision tree T_v of S_P. For a single leaf of T_v, we use standard conjugate priors:

Dirichlet: Specify hyper-parameters $\alpha = (\alpha_x)_{x \in S_v}$ and let $\theta \sim \mathrm{Dirichlet}(\alpha)$.

Normal-inverse-gamma: Specify hyper-parameters ν, λ, $\bar{\mu}$, and a, and let $\sigma^2 \sim \mathrm{IG}(\nu/2, \nu\lambda/2)$ and $\mu|\sigma^2 \sim \mathrm{N}(\bar{\mu}, \sigma^2/a)$.

While we could set the hyper-parameters separately for each X_v, our experiments will focus on the case where all are set to constant values: $\alpha = 1/2$ (Jeffreys's prior), $\nu = 1$, $\lambda = 1$, $\bar{\mu} = 0$, $a = 1$.

We let the parameters of all leaves of T_v be independent. For a leaf R, let $I_R = \{i : X_P^i \in R\}$ be the index set of the data points that belong to R. If X_v is categorical, we obtain the marginal likelihood of T_v as

$$\ell_v(T_v) = \prod_{R \text{ leaf of } T_v} \frac{\Gamma(\sum_x \alpha_x)}{\Gamma(N_R + \sum_x \alpha_x)} \prod_x \frac{\Gamma(N_{Rx} + \alpha_x)}{\Gamma(\alpha_x)}$$

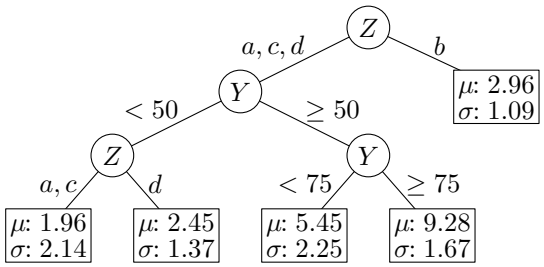

Figure 1: An example of a PCART model. It represents the conditional probability density $p(X|Y, Z)$, where $X \in (0, 10)$, $Y \in (0, 100)$, and $Z \in \{a, b, c, d\}$.

where $N_R = |I_R|$ and $N_{Rx} = |\{i \in I_R : X_v^i = x\}|$. If X_v is continuous, we get

$$\ell_v(T_v) = \prod_{R \text{ leaf of } T_v} \pi^{-N_R/2} (\lambda\nu)^{\nu/2} \frac{\sqrt{a}}{\sqrt{N_R + a}} \frac{\Gamma((N_R + \nu)/2)}{\Gamma(\nu/2)} (s_R + t_R + \nu\lambda)^{-(N_R+\nu)/2} ,$$

where $s_R = (N_R - 1)\sigma_R^2$, $t_R = N_R a(\mu_R - \bar{\mu})^2/(N_R + a)$, μ_R is the sample mean and σ_R^2 the sample variance of the N_R values X_v^i with $i \in I_R$ (Chipman et al., 1998).

We let the parameters be independent for each v, so that the marginal likelihood of the structure $(G_v, T_v)_{v \in V}$ factorizes into a product of the local marginal likelihoods $\ell_v(T_v)$. We also assign a modular structure prior

$$p\big((G_v, T_v)_{v \in V}\big) \propto \prod_{v \in V} \pi_v(G_v, T_v) ,$$

where each π_v is some (unnormalized) distribution. Various priors of decision trees have been proposed (Buntine, 1992; Chipman et al., 1998); see Section 3 for the choice in our experiments.

The structure learning problem now becomes that of finding a structure $(G_v, T_v)_{v \in V}$ so as to maximize the posterior probability

$$\text{constant} \times \prod_{v \in V} \pi_v(G_v, T_v)\, \ell_v(T_v) . \tag{1}$$

As the unspecified (normalizing) constant is the same for all structures, it can be ignored.

3. Partition–Dyadic CART

Partition–dyadic CART (PCART) is a subclass of the CART model, which we obtain by restricting the way in which the range of a continuous predictor variable is recursively partitioned by the decision tree. Specifically, we assume the range is an interval and only allow halving it in its midpoint, whence the name dyadic splits. For the number of recursive dyadic splits per variable we set an upper bound, *the maximum number of splits*, we will denote by s. In contrast, for a categorical predictor we allow arbitrary (binary) partitioning. Figure 1 shows an example of a PCART model.

PCART generalizes *dyadic decision trees* (Scott and Nowak, 2006), which assume all predictors be continuous (or binary) and the response be categorical. The motivation for dyadic splits stems from their ability to guarantee both statistical and computational efficiency: dyadic decision trees enable computationally feasible global optimization on data sets with up to around a dozen of predictor variables, and they achieve (in a minimax sense) nearly optimal rates of convergence for a range of classification problems (Scott and Nowak, 2006; Blanchard et al., 2007).

3.1 Structure Prior

Having fixed the family of decision trees, we proceed to specify a prior distribution over the family. To conform to the modular structure of the likelihood function, we construct a prior where the probability $p(T)$ of a tree T factorizes over the leaves of T. This yields a decomposable scoring function, which property is crucial for the optimization algorithm we give in the next section.

We consider priors that assign $p(T) \propto \gamma^{l(T)}$, where $l(T)$ is the number of leaves of T and $0 < \gamma \leq 1$ is a constant independent of T. Importantly, we do not set γ to any fixed value, but we instead let it depend on the number of splits possible in each inner node. Without such dependency, the prior could distribute nearly all probability to very large decision trees, as their number overwhelms the number of small trees.

More precisely, we set $\gamma = 1/(4C)$, where C is the total number of possible splits at an inner node (ignoring splits made in other inner nodes). We have that C is a sum where each continuous predictor contributes 1 and each categorical predictor with k possible values contributes $2^{k-1} - 1$. The factor 4 comes from the number of ordered full binary trees with i inner nodes, given by the ith Catalan number and equalling to 4^i up to a factor polynomial in i. We can interpret the prior as first drawing the number of leaves from a nearly uniform distribution, then drawing a random full binary tree with the given number of leaves, drawing a random split independently for each inner node of the tree, and accepting the tree if it is a valid decision tree.

3.2 PCART As a Local Model

To use PCART as local models in BNs, we need to specify the components $\pi_v(G_v, T_v)$ of the structure prior. Let $P = G_v \subseteq V \setminus \{v\}$ with $|P| \leq D$; the *maximum indegree* parameter D controls the computational and statistical complexity of the model. For brevity, write T for T_v. We use the factorization

$$\pi_v(P, T) = \pi_v(P)\, \pi_v(T|P)\, c(P)\,,$$

where $c(P)$ normalizes $\pi_v(T|P)$ into a probability distribution for T. It remains to specify $\pi_v(P)$ and $\pi_v(T|P)$. We consider two different choices for both.

Our first choice is to set $\pi_v(P) = 1$, i.e., the prior is *nearly uniform over parent sets* of size at most D. This is the simplest and most "ignorant" prior, which treats all DAGs equally a priori; for a deeper discussion of graph priors we refer to Angelopoulos and Cussens (2008) and references therein. We get two models:

PCART (partition–dyadic CART): Let $\pi_v(T|P) = (4C)^{-l(T)}$ for all decision trees T of S_P with at most s dyadic splits per continuous variable.

FT (conditional probability table): Supposing all predictor variables are categorical, let F be a maximal decision tree of S_P where each leaf is a singleton $\{x_P\}$ with $x_P \in S_P$. We let $\pi_v(T|P) = 1$ if $T = F$ and $\pi_v(T|P) = 0$ otherwise.

To make the model applicable to continuous predictors, we discretize them into some small number of bins k based on quantiles of the empirical distribution. If fact, we let each continuous predictor $u \in P$ have a dedicated number of bins k_u, which takes a value between 2 and some maximum M (we used $M = 7$ in our experiments).

Our second choice is given by $\pi_v(P) = 1/\binom{n-1}{|P|}$, i.e., the prior is *nearly uniform over the sizes of the parent sets*.[1] This prior avoids concentrating almost all prior probability on the densest graphs; see, e.g., Friedman and Koller (2003) for usage and discussion of this prior. We get the *penalized* variants *PCARTp* and *FTp*.

Note that BNs with FT and FTp specialize to the standard BNs with CPT (with a uniform and a non-uniform prior, respectively) when all variables are categorical and have at most M states.

4. Algorithms

The state-of-the-art algorithms for finding a globally optimal BN structure have two phases:

Phase I (Candidate parent set scoring and pruning): For each $v \in V$ and $P \subseteq V \setminus \{v\}$ with $|P| \leq D$ compute the local score $g_v(P)$, defined in terms of the local model and the data (e.g., prior and likelihood). Prune every set that has a subset with at least as large score. Return the remaining parent sets and their scores.

Phase II (DAG search): Return a DAG G that maximizes the total score $\prod_{v \in V} g_v(G_v)$.

For phase II we may use existing complete solvers, e.g., ones based on integer linear programming (Bartlett and Cussens, 2017) or A* (Yuan and Malone, 2013). In our experiments we used the constraint programming based solver *CPBayes* (van Beek and Hoffmann, 2015), since in our preliminary study it solved all instances of interest in less than 24 hours.

Our concern is phase I. By the definition of the scoring function (1), the local scores are set by

$$g_v(P) = \max_T \left\{ \pi_v(P, T)\, \ell_v(T) \right\},$$

where T runs through all local structures on the predictors P. Indeed, it is easy to see that this assignment quarantees that the algorithm will find a globally optimal structure:

Proposition 1 (Optimality) *Let G be a DAG that maximizes $\prod_{v \in V} g_v(G_v)$. Then, for each $v \in V$, there exists a T_v such that the structure $(G_v, T_v)_{v \in V}$ maximizes the scoring function (1).*

In the case of FT (maximal decision tree), it suffices to enumerate the possible joint discretizations of the continuous predictors. In the case of PCART each local score requires optimization over a large number of decision trees. Moreover, the number of required local scores grows rapidly with the total number of variables n, being $n \sum_{d=0}^{D} \binom{n-1}{d}$ for maximum indegree D. In the remainder of this section we detail a dynamic programming algorithm to solve this optimization problem.[2]

4.1 PCART Structure Optimization

We give a dynamic programming algorithm to find an optimal PCART structure for a response v and a set of predictors $P \subseteq V \setminus \{v\}$. The algorithm applies to any objective function that factorizes into a product of local scores $f(R)$ over the leaves R of the tree. Our Bayes score is clearly of this form as both the prior and the likelihood factorize over the leaves. For every possible decision tree node $R = \times_{u \in P} R_u$, the algorithm computes, in a top-down fashion, the best score $f(R)$ for the subtree below R. Thus the score of the root, $f(S_P)$, is the maximum score over all PCART structures.

1. The prior is not exactly uniform because sparser directed graphs are acyclic with higher probability.
2. We have developed a similar algorithm for computing the normalizing term $c(P)$, which is a *sum* over decision trees.

The best subtree score for node R is obtained by considering all possibilities for the node: the node may be a leaf node, or it is an inner node with two children R' and R''. The score of a leaf is computed from the training data points contained in R as specified in Sections 2 and 3. For an inner node the score is computed recursively as the product $f(R')f(R'')$ of the scores of the child subtrees. The recursion caches the already computed subtree scores to avoid computing the same score multiple times. After computing the scores, an optimal decision tree is extracted by tracing the splits that gave the optimal scores.

With our choice of scores, it happens that a node R that contains no training data points has to always be a leaf node, and the recursion does not need to advance further. Furthermore, if for some categorical variable u there are some values in R_u that do not appear in the training data that is contained in R, we can simply consider a smaller node with the unused values removed. These optimizations significantly reduce the number of nodes the algorithm has to consider.

We next give an asymptotic upper bound for the number of nodes K the algorithm considers and for the running time of the algorithm. The bounds will be rather loose: they fail to account for all the optimizations and special properties of the data sets. For the bounds, let us denote the number of categorical and continuous predictor variables by A and B, respectively, the maximum number of categories by k, and the maximum number of recursive splits per continuous variable by s.

Counting the possible nodes simply as the product of at most 2^k possible value subsets per categorical variable and at most 2^{s+1} subsets per continuous variable, gives an upper bound $K = O(2^{Ak+B(s+1)})$. However, in the case when the number of training data points N is low, we get a better bound similarly to Blanchard et al. (2007) (who assume all variables be continuous): Since the algorithm only considers nodes that contain at least one training data point, we can bound the number of nodes by the total number of containing nodes for each data point. The number of value subsets that contain a given value is at most 2^{k-1} for each categorical variable and $s+1$ for each continuous variable, yielding an upper bound $K = O(N2^{A(k-1)}(s+1)^B)$.

Now, for every node the algorithm considers all possible splits—at most 2^k for each categorical variable and one for each continuous variable—and finds the score from an associative array data structure in $O(\log K)$ time. We get the following result.

Proposition 2 (Time complexity) *The running time of the PCART optimization algorithm is* $O(n(2^k A + B)K \log K)$*, where* $K = O\big(\min\big\{N2^{A(k-1)}(s+1)^B,\, 2^{Ak+B(s+1)}\big\}\big)$.

4.2 Implementation Tricks

The dynamic programming algorithm for finding an optimal PCART structure is easily implemented using recursion, as described in Section 4.1. However, by carefully tuning the memory representation of nodes and arranging the algorithm such that we can use a simple sorted list instead of dynamic associative array data structures, we are able to optimize our `C++` implementation[3] to run an order of magnitude faster.

Our implementation stores node R as a bit vector that is a concatenation of fixed-length bitvectors, each of which stores the value subset R_u for a variable $u \in P$. For each continuous variable with maximum number of splits s, the $2^{s+1} - 1$ possible intervals are stored as $(s+1)$-bit binary numbers. For each categorical variable u, the subset of values $R_u \subseteq S_u$ is stored using $|S_u|$ bits. An example of the representation is shown in Figure 2. Using this representation, the operations

3. The code is publicly available at `http://www.tba`.

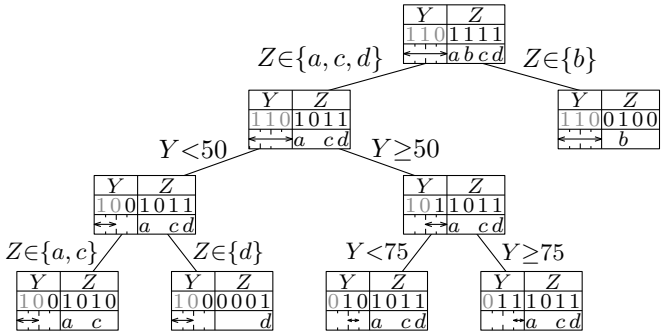

Figure 2: An illustration of the binary node representation. The example corresponds to the PCART in Figure 1, and consists of two parts: 3 bits for the continuous variable Y (here the maximum number of splits is 2) and 4 bits for the categorical variable Z. For the continuous variable, the length of the leading 1-bits and the following 0-bit (in gray) indicates the length of the interval, and the remaining bits encode the position of the interval.

performed by the algorithm on the nodes translate into efficient bit operations on 64-bit machine words. The numerical ordering of these bit representations is bottom-up in the dynamic programming, which allows us to iteratively build a sorted list of the node-score-pairs instead of using recursion and dynamic associative array data structures.

5. Empirical Studies

How does PCART compare to its full-table variants when they are used as local models in Bayesian network learning? We next study this question from the aspects of (i) capability of recovering a data-generating DAG, and (ii) the time requirement of phase I (local score computation) and phase II (DAG optimization). For phase I we use our implementation described in Section 4; for phase II we use CPBayes (van Beek and Hoffmann, 2015).

5.1 Benchmark Networks

We investigated the capability of the different models to recover a data-generating network structure. For this purpose, we use the networks Alarm, Insurance, and Water [4], which contain 37, 27, and 32 variables respectively, and are thus suitable for finding the globally optimal DAG with modern solvers. With a given ground-truth network, we generate a data set of size N and used it to learn DAGs using the four local models described in Section 3, setting the maximum indegree to $D = 4$. We next compared the learned DAGs to the generating DAG using the structural Hamming distance (SHD) (Tsamardinos et al., 2006). For each network and each sample size $N = 50 \times 2^i$, for $i = 0, 1, \ldots, 9$, we repeated the procedure ten times and averaged the resulting SHDs (Figure 3).

We observe that both PCART variants yield a smaller SHD than FT for all generating networks and sample sizes. When adding the nonuniform DAG prior to FT, it becomes competitive for Alarm. A possible explanation is that Alarm has the smallest edge/node ratio of only 1.24, whereas Insurance and Water have 1.93 and 2.06, respectively. A small average indegree leaves little potential for

4. All data sets were obtained from `http://www.bnlearn.com/bnrepository`

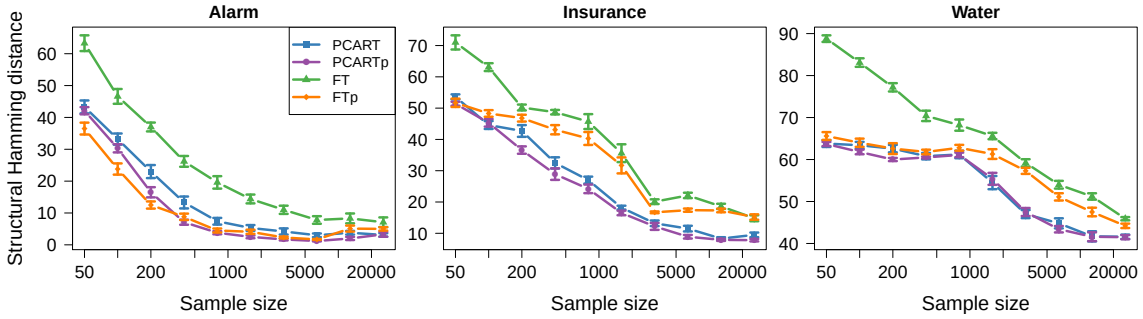

Figure 3: Structure recovery of benchmark networks.

structured CPDs as the majority of CPTs can be accurately represented with a (small) probability table. For Insurance and Water the PCART variants yield faster convergence than the FT models.

We also performed a Wilcoxon signed rank test, with level $\alpha = 0.01$, to compare the SHD populations of PCARTp and FTp for each network and sample size. For Alarm, the difference is indeed statistically insignificant: for each sample size the p-value exceeds 0.01. For Insurance and Water, PCARTp yields a significantly lower SHD than FTp for $N \leq 400$ and $3200 \leq N \leq 16400$, respectively. These findings coincide well with visual inspection of the learning curves in Figure 3.

5.2 Real Data

All variables in the three benchmark networks are discrete. To evaluate the structure learning performance in the presence of continuous variables, we thus need to utilize real-world data sets, e.g., from the UCI Machine Learning Repository (Lichman, 2013).

For evaluating structure learning based on a data set alone, we used a very recent method called Intersection-Validation (Viinikka et al., 2018), which allows to draw learning curves along the lines of Figure 3 even when no ground-truth DAG is available. The method checks which structural features all included learning methods agree upon when learning from the entire data set, and treats them as surrogate ground truth. Structures learned at smaller subsamples of the data are compared against this partial network to compute an approximation of SHD, called *partial Hamming distance*.

Naturally, we may expect Intersection-Validation be the more reliable, the larger the partial network is, that is, the larger the available complete data set is. Therefore, we picked the four UCI data sets are (i) suitable to exact structure recovery due to a moderate number of variables

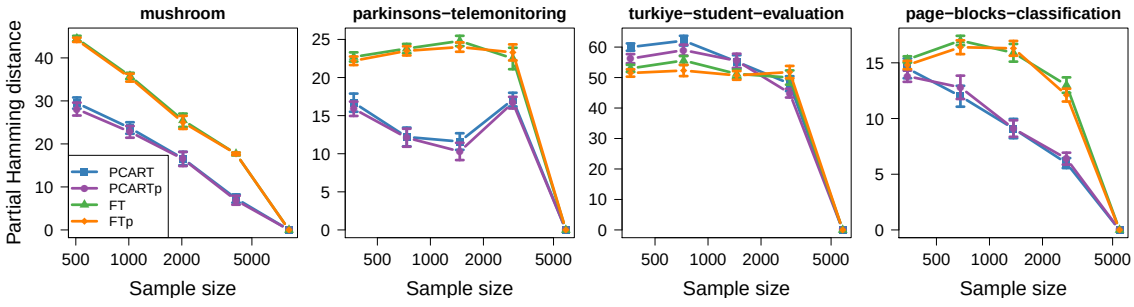

Figure 4: Structure learning from UCI data sets, evaluated using the intersection–validation method.

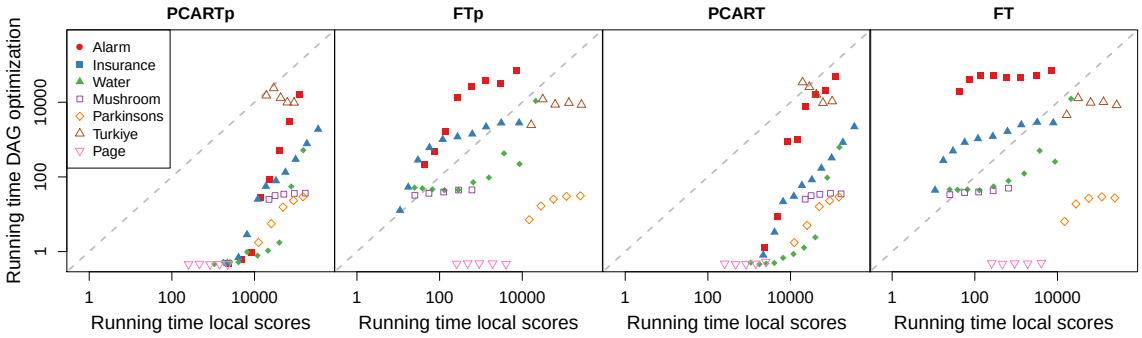

Figure 5: Running times (in seconds) for computing local scores and for DAG optimization. For each benchmark network and data set, different measurements correspond to different sample sizes.

and (ii) contain more than 5000 data points: mushroom (0/22 continuous variables), parkinsons-telemonitoring (20/21), turkiye-student-evaluation (31/33), and page-blocks-classification (10/11). For each data set and learning method, we learned networks from subsamples of $1/2$, $1/4$, $1/8$, and $1/16$ of the original sample size and repeated the subsampling process ten times.

The resulting averaged learning curves for the four methods in comparison are shown in Figure 4. For turkiye-student-evaluation the methods differ only slightly; while FT and FTp achieve slightly lower partial Hamming distances, none of the methods performs particularly well. In the remaining three cases, PCART and PCARTp always outperform the FT variants, and in 8 of 12 combinations of data set and sample size by a large relative margin, passing the Wilcoxon signed rank test with $\alpha = 0.01$. This demonstrates that structured CPDs can improve structure recovery not only for purely discrete networks, but also in the presence of continuous variables.

5.3 Running Times

Figure 5 summarizes the running times of the algorithms corresponding to the experiments in the previous two sections. Plotted is the time needed for computing the local scores against the time needed by CPBayes to find an optimal DAG given the local scores. Each individual point in the plots corresponds to the average running time for a concrete sample size.

We note that the difference between the penalized and unpenalized variants are small and thus focus on the comparison of PCARTp with FTp in the following. Only when the number of variables n is very small (page-blocks-classification), computing the local scores cleary dominates running time, as DAG optimization is very fast. The picture changes when n and the sample size grow: DAG optimization becomes more expensive and its time consumption will eventually surpass that of the local score computation, which grows only polynomially in n for any bounded maximum indegree D. Moreover, the running time for CPBayes with PCARTp is often faster than CPBayes with FTp, probably because the former allows a more effective pruning of the DAG search space.

6. Concluding Remarks

We have advanced the idea of using decision trees as local models in Bayesian networks (Friedman and Goldszmidt, 1996). We introduced a new model class, PCART, which (i) can handle both categorical and continuous predictor and response variables, (ii) takes a Bayesian approach, and

(iii) admits computationally feasible global optimization in a range of problem sizes. Armed with this, we empirically compared decision trees against their full-table counterparts.

In our experiments we set the parameters that control the computational complexity of the model (maximum indegree, number of splits) so that the computations of the local scores (phase I) could be completed within a few hours for each data set. This was motivated by the fact that for some of the data sets, the state-of-the-art solvers require hours to find an optimal DAG (phase II). There are two simple ways to expedite phase I, if needed: by setting the complexity parameters to lower values, and by running the computations in parallel.

Another direction for future research is to more systematically examine the effect of different parameter priors in PCART and the full-table variants; the choice of the hyperparameter values is known to affect the learned model for standard CPTs (Silander et al., 2007), and our setting is even more complex. We suspect that our simple choice to use same constant hyperparameter values over all data sets and variables could be improved by more local, empirical Bayes-like methods.

A related question is, whether the power of PCART in structure learning transfers to density estimation, i.e., predictive performance. We have looked at this, and it appeared that the differences between PCART and the full-table variants are small. We find this observation unsurprising, given that density estimation is, in a sense, an easier task than structure learning (Viinikka et al., 2018). One might hypothesize that in density estimation PCART's ability to learn "weak arcs" is less important and, moreover, selecting a single (optimal) local structure for each node could lead to over-fitting. Due to space constraints, we omit presenting and discussing the results in full here. In addition, PCART and the full-table variants warrant further investigation as alternatives to other approaches to hybrid Bayesian networks, such as hybrid copulas (Karra and Mili, 2016), iterative discretization schemes (Chen et al., 2017), or some commonly used simpler discretization methods that only consider the marginal or pairwise distributions.

Acknowledgments

This work was supported in part by the Academy of Finland, Grant 276864.

References

N. Angelopoulos and J. Cussens. Bayesian learning of Bayesian networks with informative priors. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):53–98, 2008.

M. Bartlett and J. Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

G. Blanchard, C. Schäfer, Y. Rozenholc, and K. Müller. Optimal dyadic decision trees. *Machine Learning*, 66(2-3):209–241, 2007.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. UAI*, 1996.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

W. Buntine. Learning classification trees. *Statistics and Computing*, 2(2):63–73, 1992.

Y. Chen, T. Wheeler, and M. Kochenderfer. Learning discrete Bayesian networks from continuous data. *Journal of Artificial Intelligence Research*, 59:103–132, 2017.

H. Chipman, E. George, and R. McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

D. Donoho. CART and best-ortho-basis: A connection. *Annals of Statistics*, 25(5):1870–1911, 1997.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Proc. UAI*, 1996.

N. Friedman and D. Koller. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1-2):95–125, 2003.

K. Karra and L. Mili. Hybrid copula Bayesian networks. In *Proc. PGM*, pages 240–251, 2016.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

M. Lichman. UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science, 2013. URL `[http://archive.ics.uci.edu/ml]`.

B. Malone, M. Järvisalo, and P. Myllymäki. Impact of learning strategies on the quality of Bayesian networks: An empirical evaluation. In *Proc. UAI*, pages 562–571, 2015.

B. Malone, K. Kangas, M. Järvisalo, M. Koivisto, and P. Myllymäki. Empirical hardness of finding optimal Bayesian network structures: algorithm selection and runtime prediction. *Machine Learning*, 107(1):247–283, 2018.

J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

C. Scott and R. Nowak. Minimax optimal classification with dyadic decision trees. *IEEE Transactions on Information Theory*, 52(4):1335–1353, 2006.

T. Silander, P. Kontkanen, and P. Myllymäki. On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In *Proc. UAI*, pages 360–367, 2007.

I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

P. van Beek and H. Hoffmann. Machine learning of Bayesian networks using constraint programming. In *Proc. CP*, volume 9255 of *Lecture Notes in Computer Science*, pages 429–445. Springer, 2015.

J. Viinikka, R. Eggeling, and M. Koivisto. Intersection-validation: A method for evaluating structure learning without ground truth. In *Proc. AISTATS*, pages 1570–1578, 2018.

C. Yuan and B. Malone. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, 2013.