Machine Learning, 59, 1–32, 2005
2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

TAN Classifiers Based on Decomposable Distributions

JESÚS CERQUIDES cerquide@maia.ub.es
Dept. de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via, 585, 08007 Barcelona, Spain

RAMON LÓPEZ DE MÁNTARAS mantaras@iiia.csic.es
Institut d'Investigació en Intel.ligència Artificial, CSIC, Campus UAB, 08190 Bellaterra, Spain

Editors: Pedro Larrañaga, Jose A. Lozano, Jose M. Peña and Iñaki Inza

Abstract. In this paper we present several Bayesian algorithms for learning Tree Augmented Naive Bayes (TAN) models. We extend the results in Meila & Jaakkola (2000a) to TANs by proving that accepting a prior decomposable distribution over TAN's, we can compute the exact Bayesian model averaging over TAN structures and parameters in polynomial time. Furthermore, we prove that the *k*-maximum a posteriori (MAP) TAN structures can also be computed in polynomial time. We use these results to correct minor errors in Meila & Jaakkola (2000a) and to construct several TAN based classifiers. We show that these classifiers provide consistently better predictions over Irvine datasets and artificially generated data than TAN based classifiers proposed in the literature.

Keywords: Bayesian networks, Bayesian network classifiers, naive Bayes, tree augmented naive Bayes, decomposable distributions , Bayesian model averaging

1. Introduction

Bayesian network classifiers such as *naive Bayes* (Langley, Iba, & Thompson, 1992) or *Tree Augmented Naive Bayes* (TAN) (Friedman, Geiger, & Goldszmidt, 1997) have shown excellent performance in spite of their simplicity and heavy underlying independence assumptions.

In their seminal work (Chow & Liu, 1968), Chow and Liu proved that we can find the maximum likelihood tree structure and parameters in polynomial time. In 1997, Friedman et al. introduced TAN in Friedman, Geiger, and Goldszmidt (1997) by extending Chow and Liu results from tree structures to TAN structures. Meila and Jaakkola (2000a) defined decomposable distributions over tree belief networks and claimed that the exact Bayesian model averaging over tree structures and parameters was computable in polynomial time. In this paper we correct Meila and Jaakkola results and extend them to TANs.

The paper relevance is twofold. First, it presents a new, careful, coherent and theoretically appealing development of TAN based classifiers rooted in Bayesian probability theory concepts. Second, it establishes through empirical tests the practical usefulness of these results.

We start the paper by presenting the notation to be used and shortly reviewing previous work in TAN classifiers.

In Section 3, we define decomposable distributions over TAN models and we prove several properties for them. After that, in Section 4, we show that decomposable distributions over trees can be seen as a particular case of decomposable distributions over TANs and correct (Meila & Jaakkola, 2000a) results.

In Section 5, we use decomposable distributions over TANs to construct four classifiers.

In Section 6 we compare the empirical results obtained by the four classifier introduced, with the softened TAN classifier proposed in Friedman, Geiger, and Goldszmidt (1997) which we name STAN. The comparison is made both over a set of Irvine datasets and over a set of randomly generated Bayesian networks with different characteristics. We show that all our classifiers provide consistently better predictions than STAN in a statistically significant way. We also show that averaging over TANs also improves over selecting a single TAN structure when little data is available.

Finally, we provide some conclusions and future work in Section 7.

2. Trees and tree augmented Naive Bayes

Tree Augmented Naive Bayes (TAN) appears as a natural extension to the *naive Bayes* classifier (Kontkanen et al., 1998; Langley, Iba, & Thompson, 1992; Domingos & Pazzani, 1997). TAN models result from adding a tree of dependencies between the attributes to a naive Bayes model.

In this section we start introducing the notation to be used in the rest of the paper. After that we discuss the TAN induction algorithm presented in Friedman, Geiger, and Goldszmidt (1997).

2.1. Formalization and notation

The notation used in the paper is an effort to put together the different notations used in Cerquides (1999), Heckerman, Geiger, and Chickering (1995), Friedman, Geiger, and Goldszmidt (1997) Meila and Jaakkola (2000a) and some conventions in the machine learning literature.

2.1.1. The discrete classification problem. A *discrete attribute* is a finite set, for example we can define attribute *Pressure* as $Pressure = \{Low, Medium, High\}$. A *discrete domain* is a finite set of discrete attributes. We will note $\Omega = \{X_1, \dots, X_m\}$ for a discrete domain, where $X_1, \dots, X_m$ are the attributes in the domain. A *classified discrete domain* is a discrete domain where one of the attributes is distinguished as "class". We will use $\Omega_C = \{A_1, \dots, A_n, C\}$ for a classified discrete domain. We will use i and j as values of an attribute and u and v as indexes over attributes in a domain. We refer to an attribute either as X_u (when it is considered part of a discrete domain), A_u (when it is considered part of a classified discrete domain and it is not the class) C (when it is the class of a classified discrete domain). We note $X_{-C} = \{A_1, \dots, A_n\}$ the set that contains all the attributes in a classified discrete domain except the class attribute.

Given an attribute X, we note $\#X$ as the number of different values of X. We define

$$\#\Omega = \prod_{u=1}^{m} \#X_u \quad \text{and} \quad \#\Omega_C = \#C \prod_{u=1}^{n} \#A_u.$$

We note the maximum number of different values for an attribute in a domain

$$r = \max_{u \in \Omega} \#X_u = \max\left(\max_{u \in X_{-C}} \#A_u, \#C \right).$$

An *observation* x in a classified discrete domain Ω_C is an ordered tuple $x = (x_1, \dots, x_n, x_C) \in A_1 \times \cdots \times A_n \times C$. An *unclassified observation* x_{-C} in Ω_C is an ordered tuple $x_{-C} = (x_1, \dots, x_n) \in A_1 \times \cdots \times A_n$. To be homogeneous we will abuse this notation a bit noting x_C for a possible value of the class for x_{-C}. A *dataset* $\mathcal{D}$ in Ω_C is a multiset of classified observations in Ω_C.

We will note N for the number of observations in the dataset. We will also note $N_u(i)$ for the number of observations in $\mathcal{D}$ where the value for A_u is i, $N_{u,v}(i, j)$ the number of observations in $\mathcal{D}$ where the value for A_u is i and the value for A_v is j and similarly for $N_{u,v,w}(i, j, k)$ and so on.

A *classifier* in a classified discrete domain Ω_C is a procedure that given a dataset $\mathcal{D}$ in Ω_C and an unclassified observation x_{-C} in Ω_C assigns a class to x_{-C}.

2.1.2. Bayesian networks for discrete classification. Bayesian networks offer a solution for the discrete classification problem. The approach is to define a random variable for each attribute in Ω (the class is included but not distinguished at this time). We will note $\mathbf{U} = \{\mathcal{X}_1, \dots, \mathcal{X}_m\}$ where each $\mathcal{X}_u$ is a random variable over its corresponding attribute X_u. We extend the meaning of this notation to $\mathcal{A}_u, \mathcal{C}$ and $\mathcal{X}_{-C}$. A *Bayesian network* over $\mathbf{U}$ is a pair $B = \langle G, \Theta \rangle$. The first component, G, is a directed acyclic graph whose vertices correspond to the random variables $\mathcal{X}_1, \dots, \mathcal{X}_m$ and whose edges represent direct dependencies between the variables. The graph G encodes independence assumptions: each variable $\mathcal{X}_u$ is independent of its non-descendants given its parents in G. The second component of the pair, namely Θ, represents the set of parameters that quantifies the network. It contains a parameter $\theta_{u|\Pi_u}(x_u, \Pi_{x_u}) = P_B(x_u \mid \Pi_{x_u})$ for each $x_u \in X_u$ and $\Pi_{x_u} \in \Pi_{X_u}$, where Π_{X_u} denotes the Cartesian product of every X_u such that $\mathcal{X}_u$ is a parent of $\mathcal{X}_u$ in G. Π_u is the list of parents of $\mathcal{X}_u$ in G. We will note $\bar{\Pi}_u = \mathbf{U} - \{\mathcal{X}_u\} - \Pi_u$. A Bayesian network defines a unique joint probability distribution over $\mathbf{U}$ given by

$$P_B(x_1, \dots, x_m) = \prod_{u=1}^{m} P_B\big(x_u \mid \Pi_{x_u}\big) = \prod_{u=1}^{m} \theta_{u|\Pi_u}\big(x_u \mid \Pi_{x_u}\big). \tag{1}$$

The application of Bayesian networks for classification can be very simple. For example suppose we have an algorithm that, given a classified discrete domain Ω_C and a dataset $\mathcal{D}$ over Ω_C, returns a Bayesian network B over $\mathbf{U} = \{\mathcal{A}_1, \dots, \mathcal{A}_n, \mathcal{C}\}$ where each $\mathcal{A}_u$ (resp. $\mathcal{C}$) is a random variable over A_u (resp. C). Then if we are given a new unclassified observation x_{-C} we can easily classify x_{-C} into class $argmax_{x_C \in C}(P_B(x_{-C}, x_C))$.

```
procedure Construct-TAN (ProbabilityDistribution $P$)
    foreach $A_u$, $A_v$
        Compute $I_P(A_u; A_v|C) = \sum_{\substack{i \in A_u \\ j \in A_v \\ c \in C}} P(i,j,c) log(\frac{P(i,j|c)}{P(i|c)P(j|c)})$
    end
    $UT$ = MST of the clique that has $I_P$ as weigth matrix
    Construct $T$ from $UT$ by selecting a root
    Add edges to $T$ connecting $C$ to every $A_u$
    Fix $T$ parameters using equation 2
```

Algorithm 1. Maximum likelihood TAN construction procedure.

This simple mechanism allows us to see any Bayesian network learning algorithm as a classifier.

2.1.3. Notation for learning with TANs. Given a classified domain Ω_C we will note $\mathcal{E}$ the set of undirected graphs E over $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ such that E is a tree. We will use $u, v \in E$ instead of $(\mathcal{A}_u, \mathcal{A}_v) \in E$ for simplicity. We note $\bar{E}$ a directed tree for E and $Or(E)$ the set of directed trees with undirected structure E. We note $\rho_{\bar{E}}$ the root of a directed tree $\bar{E}$. Note that the choice of the root uniquely determines the orientation of edges of an undirected tree. Every $\bar{E}$ uniquely determines the structure of a Tree Augmented Naive Bayes classifier, because from $\bar{E}$ we can construct $\bar{E}^* = \bar{E} \cup \{(\mathcal{C}, \mathcal{A}_u) \mid 1 \leq u \leq n\}$.

We note $\Theta_{\bar{E}^*}$ the set of parameters that quantify the Bayesian network $B = \langle \bar{E}^*, \Theta_{\bar{E}^*} \rangle$. More concretely:

$$\Theta_{\bar{E}^*} = (\boldsymbol{\theta}_C, \boldsymbol{\theta}_{\rho_{\bar{E}}|C}, \{\boldsymbol{\theta}_{v|u,C} \mid u, v \in \bar{E}\}).$$
$$\boldsymbol{\theta}_C = \{\theta_C(c) \mid c \in C\} \text{ where } \theta_C(c) = P(\mathcal{C} = c \mid B).$$
$$\boldsymbol{\theta}_{\rho_{\bar{E}}|C} = \{\theta_{\rho_{\bar{E}}|C}(i, c) \mid i \in A_{\rho_{\bar{E}}}, c \in C\} \quad \text{where}$$
$$\theta_{\rho_{\bar{E}}|C}(i, c) = P(\mathcal{A}_{\rho_{\bar{E}}} = i \mid \mathcal{C} = c, B).$$

For each $u, v \in \bar{E}$:

$$\boldsymbol{\theta}_{v|u,C} = \{\theta_{v|u,C}(j, i, c) \mid j \in A_v, i \in A_u, c \in C\} \quad \text{where}$$
$$\theta_{v|u,C}(j, i, c) = P(\mathcal{A}_v = j \mid \mathcal{A}_u = i, \mathcal{C} = c, B).$$

2.2. Learning maximum likelihood TAN

The results for finding maximum likelihood trees in Chow and Liu (1968) were extended to TANs in Friedman, Geiger, and Goldszmidt (1997). We recall here the procedure and theorem for learning maximum likelihood TANs.

Theorem 1 (Friedman et al., 1997.) *Let $\mathcal{D}$ be a dataset over Ω_C. The procedure* `Construct − TAN`*(f) (where f is the probability distribution induced by the frequencies in $\mathcal{D}$) builds a TAN B_T that maximizes the likelihood given $\mathcal{D}$ and has time complexity $O((N + r^3) \cdot n^2)$.*

Algorithm 1 fixes the model parameters by using:

$$\begin{aligned}\theta_C(c) &= P(\mathcal{C} = c)\\ \theta_{\rho_{\bar{E}}|C}(i, c) &= P\big(\mathcal{A}_{\rho_{\bar{E}}} = i \mid \mathcal{C} = c\big)\\ \theta_{v|u,C}(j, i, c) &= P(\mathcal{A}_v = j \mid \mathcal{A}_u = i, \mathcal{C} = c)\end{aligned} \tag{2}$$

where P is the probability distribution that it receives as parameter. It has been shown (Friedman, Geiger, & Goldszmidt, 1997) that Eq. (2) leads to models that overfit the data. As an alternative, Friedman et al. propose to set the parameters as follows:

$$\begin{aligned}\theta_C(c) &= P(\mathcal{C} = c)\\ \theta_{\rho_{\bar{E}}|C}(i, c) &= \frac{N \cdot P(\mathcal{A}_{\rho_{\bar{E}}} = i, \mathcal{C} = c) + N^0 \cdot P(\mathcal{A}_{\rho_{\bar{E}}} = i)}{N \cdot P(\mathcal{C} = c) + N^0}\\ \theta_{v|u,C}(j, i, c) &= \frac{N \cdot P(\mathcal{A}_v = j, \mathcal{A}_u = i, \mathcal{C} = c) + N^0 \cdot P(\mathcal{A}_v = j)}{N \cdot P(\mathcal{A}_u = i, \mathcal{C} = c) + N^0}\end{aligned} \tag{3}$$

and suggest the use of $N^0 = 5$ based on empirical results. The classifier resultant from finding the maximum likelihood TAN structure and adjusting the parameters as in Eq. (3) will be referred in the rest of the paper as STAN.

3. Decomposable distributions over TANs

In Meila and Jaakkola (2000a), introduced *decomposable priors*: a family of priors over structure and parameters of tree belief networks, that is, a family of probability distributions over the space of tree belief network models. We will use the term decomposable distribution over trees (resp. TANs) instead of decomposable prior, and we will use prior and posterior as they are commonly used in Bayesian analysis. In the following, we define decomposable distributions over TAN models (a generalization of decomposable distributions over trees) and we prove several properties for them. We start by introducing decomposable distributions over TAN structures and parameters. After that we demonstrate that, given a decomposable distribution over TANs, it is possible to efficiently compute the probability of an observation and that, given a prior decomposable distribution over TANs, the posterior distribution is also a decomposable distribution over TANs. Furthermore, we demonstrate that it is possible to find the MAP and k-MAP TAN structures efficiently.

3.1. Definition

Decomposable distributions over TANs are a family of probability distributions in the space $\mathcal{B}$ of TAN models. They are constructed in two steps. In the first step, a distribution over the set of different TAN structures is defined. In the second step, a distribution over the set of parameters is defined. In the following, ξ represents the hypothesis of a decomposable distribution over TAN structures and parameters. $P(B \mid \xi)$, the probability for a model

$B = \langle \bar{E}^*, \Theta_{\bar{E}^*} \rangle$ (a TAN with fixed tree structure $\bar{E}^*$ and fixed parameters $\Theta_{\bar{E}^*}$) under this hypothesis is determined by:

$$P(B \mid \xi) = P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \xi) = P(\bar{E}^* \mid \xi) P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi). \tag{4}$$

3.1.1. Decomposable distributions over TAN structures. Recalling that n is the number of attributes when talking about a classified discrete domain, a decomposable distribution over TAN structures is determined by an $n \times n$ hyperparameter matrix $\boldsymbol{\beta} = (\beta_{u,v})$ such that $\forall u, v : 1 \leq u, v \leq n$ we have that $\beta_{u,v} = \beta_{v,u} \geq 0$ and $\beta_{v,v} = 0$. We can interpret $\beta_{u,v}$ as a measure of how possible is under ξ that the edge $(\mathcal{A}_u, \mathcal{A}_v)$ is contained in the TAN model underlying the data.

We say that $P(\bar{E}^* \mid \xi)$ follows a decomposable distribution over TAN structures with hyperparameter set $\boldsymbol{\beta}$ iff:

$$P(\bar{E}^* \mid \xi) = P(\bar{E} \mid E, \xi) P(E \mid \xi) \tag{5}$$

$$P(E \mid \xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{6}$$

where Z_β is a normalization constant with value:

$$Z_\beta = \sum_{E \in \mathcal{E}} \prod_{u,v \in E} \beta_{u,v}. \tag{7}$$

It is worth noting that $P(\bar{E} \mid E, \xi)$ is left unconstrained. In fact it will be shown to be irrellevant (see Eq. (69) in Section B.1.2) if parameters follow a decomposable distribution over TAN parameters.

3.1.2. Decomposable distributions over TAN parameters. A decomposable distribution over TAN parameters follows the equation

$$P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi) = P\big(\boldsymbol{\theta}_C \mid \bar{E}^*, \xi\big) P(\boldsymbol{\theta}_{\rho_{\bar{E}}|C} \mid \bar{E}^*, \xi) \prod_{u,v \in \bar{E}} P(\boldsymbol{\theta}_{v|u,C} \mid \bar{E}^*, \xi). \tag{8}$$

Furthermore, a decomposable distribution over TAN parameters has a hyperparameter set $\mathbf{N}' = \{N'_{v,u,C}(j,i,c) \mid 1 \leq u \neq v \leq n; j \in A_v; i \in A_u; c \in C\}$ with the constraints that $N'_{v,u,C}(j,i,c) > 0$ and that there exist $N'_{u,C}(i,c)$, $N'_C(c)$, N' such that for every u, v:

$$N'_{u,C}(i,c) = \sum_{j \in A_v} N'_{v,u,C}(j,i,c) \tag{9}$$

$$N'_C(c) = \sum_{i \in A_u} N'_{u,C}(i,c) \tag{10}$$

$$N' = \sum_{c \in C} N'_C(c). \tag{11}$$

We say that $P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi)$ follows a decomposable distribution over TAN parameters with hyperparameter set $\mathbf{N}'$ iff

1. $P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi)$ fulfills Eq. (8)
2. $\mathbf{N}'$ fulfills the conditions appearing in Eqs. (9)–(11)
3. the following equations are also satisfied:

$$P(\boldsymbol{\theta}_C \mid \bar{E}^*, \xi) = D(\theta_C(.); N'_C(.)) \tag{12}$$

$$P\big(\boldsymbol{\theta}_{\rho_{\bar{E}}|C} \mid \bar{E}^*, \xi\big) = \prod_{c \in C} D\big(\theta_{\rho_{\bar{E}}|C}(., c); N'_{\rho_{\bar{E}},C}(., c)\big) \tag{13}$$

$$P(\boldsymbol{\theta}_{v|u,C} \mid \bar{E}^*, \xi) = \prod_{c \in C} \prod_{i \in A_u} D(\theta_{v|u,C}(., i, c); N'_{v,u,C}(., i, c)) \tag{14}$$

where D is the Dirichlet distribution (see Appendix A.2).

3.1.3. Decomposable distributions over TAN structures and parameters. $P(B \mid \xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$ iff the conditions in Eqs. (4)–(14) hold.

3.2. Calculating probabilities under decomposable distributions over TANs

Assume that the data is generated by a TAN model and that $P(B \mid \xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$. We can calculate the probability of an observation x given ξ by averaging over the set of TAN models (noted $\mathcal{B}$):

$$P(\mathcal{X} = x \mid \xi) = \int_{B \in \mathcal{B}} P(\mathcal{X} = x \mid B) P(B \mid \xi). \tag{15}$$

For any real $n \times n$ matrix $\boldsymbol{\tau}$ we define $Q(\boldsymbol{\tau})$ as the first $n - 1$ lines and columns of the matrix $\bar{Q}(\boldsymbol{\tau})$ where

$$\bar{Q}_{u,v}(\boldsymbol{\tau}) = \bar{Q}_{v,u}(\boldsymbol{\tau}) = \begin{cases} -\tau_{u,v} & 1 \leq u < v \leq n \\ \displaystyle\sum_{v'=1}^{n} \tau_{v',v} & 1 \leq u = v \leq n. \end{cases} \tag{16}$$

The integral for $P(\mathcal{X} = x \mid \xi)$ can be calculated in closed form by applying the matrix tree theorem (see Appendix A.1) and the result can be expressed in terms of the previously introduced Q as:

$$P(\mathcal{X} = x \mid \xi) = h_0^x |Q(\boldsymbol{\beta}\mathbf{h}^x)| \tag{17}$$

where

$$h_0^x = \frac{1}{Z_\beta} \frac{1}{N'} \prod_{A_u \in X_{-C}} N'_{u,C}(x_u, x_C) \tag{18}$$

$$\mathbf{h}^x = \left(h_{u,v}^x\right) \quad \text{where} \quad h_{u,v}^x = \frac{N'_{v,u,C}(x_v, x_u, x_C)}{N'_{u,C}(x_u, x_C) N'_{v,C}(x_v, x_C)}. \tag{19}$$

The proof for this result appears in Appendix B.1.

3.3. Learning under decomposable distributions over TANs

Assume that the data is generated by a TAN model and that $P(B \mid \xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$. Then, $P(B \mid \mathcal{D}, \xi)$, the posterior probability distribution after observing a dataset $\mathcal{D}$ (containing independent identically distributed observations) is a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}^*$ and $\mathbf{N}'^*$ given by:

$$\beta_{u,v}^* = \beta_{u,v} W_{u,v} \tag{20}$$

$$N'^*_{u,v,C}(j, i, c) = N'_{u,v,C}(j, i, c) + N_{u,v,C}(j, i, c) \tag{21}$$

where

$$W_{u,v} = \prod_{c \in C} \left(\prod_{i \in A_u} \frac{\Gamma(N'_{u,C}(i, c))}{\Gamma(N'^*_{u,C}(i, c))} \prod_{j \in A_v} \frac{\Gamma(N'_{v,C}(j, c))}{\Gamma(N'^*_{v,C}(j, c))} \right) \times \prod_{c \in C} \prod_{i \in A_u} \prod_{j \in A_v} \frac{\Gamma(N'^*_{v,u,C}(j, i, c))}{\Gamma(N'_{v,u,C}(j, i, c))}. \tag{22}$$

The proof appears in Appendix B.2.

3.4. Classifying with decomposable distributions over TANs given an undirected structure

Assume that the data is generated by a TAN model and that $P(B \mid \xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$. Then, $P(\mathcal{C} = x_C \mid \mathcal{X}_{-C} = x_{-C}, E, \xi)$, the probability of a class x_C, given an unclassified instance x_{-C} and an undirected TAN structure E, fulfills

$$P(\mathcal{C} = x_C \mid \mathcal{X}_{-C} = x_{-C}, E, \xi) \propto h_0^x \prod_{u,v \in E} h_{u,v}^x \tag{23}$$

where h_0^x and $h_{u,v}^x$ are defined as in Eqs. (18) and (19). The proof for this result follows from the result demonstrated in Appendix B.1.1

In fact, given an undirected TAN structure E, it is easy to see that the probability distribution $P(\mathcal{C} = x_C \mid \mathcal{X}_{-C} = x_{-C}, E, \xi)$ can be represented as a TAN model with structure $\bar{E}^*$, such that its undirected version coincides with E and its parameter set is given by

$$\begin{aligned} \theta_{v|\rho_{\bar{E}},C}\left(x_v, x_{\rho_{\bar{E}}}, x_C\right) &= \frac{N'_{v,\rho_{\bar{E}},C}\left(x_v, x_{\rho_{\bar{E}}}, x_C\right)}{N'_{\rho_{\bar{E}},C}\left(x_{\rho_{\bar{E}}}, x_C\right)} \\ \theta_{\rho_{\bar{E}}|C}\left(x_{\rho_{\bar{E}}}, x_C\right) &= \frac{N'_{\rho_{\bar{E}},C}\left(x_{\rho_{\bar{E}}}, x_C\right)}{N'_C(x_C)} \\ \theta_C(x_C) &= \frac{N'_C(x_C)}{N'}. \end{aligned} \tag{24}$$

This means that under decomposable distributions over TANs, the Bayesian model averaging over parameters given a fixed undirected tree structure E can be represented as a single TAN model.

3.5. Calculating the most probable undirected tree structure under a decomposable distribution over TANs

From the definition of decomposable distribution over TAN structures, concretely from Eq. (6), it is easy to see that the most probable undirected tree given a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$, is given by

$$MPT(\boldsymbol{\beta}, \mathbf{N}') = \operatorname*{argmax}_{E \in \mathcal{E}} \prod_{u,v \in E} \beta_{u,v}. \tag{25}$$

We can see that $MPT(\boldsymbol{\beta}, \mathbf{N}')$ does not depend on $\mathbf{N}'$. Furthermore, assuming that $\forall u, vu \neq v$ we have that $\beta_{u,v} > 0$, we can take the logarithm of the right hand side having

$$MPT(\boldsymbol{\beta}, \mathbf{N}') = \operatorname*{argmax}_{E \in \mathcal{E}} \sum_{u,v \in E} \log(\beta_{u,v}). \tag{26}$$

Considering the matrix $\log(\boldsymbol{\beta})$ as an adjacency matrix, $MPT(\boldsymbol{\beta}, \mathbf{N}')$ is the MST (maximum spanning tree) for the graph represented by that adjacency matrix. The complexity of the MST algorithm for a complete graph is $\mathcal{O}(n^2)$ (Pettie & Ramachandran, 2002).

Combining this result with the one in Section 3.3, we can conclude that given a decomposable distribution over TANs as prior, we can find the MAP undirected tree given a dataset $\mathcal{D}$ in $\mathcal{O}((N + r^3) \cdot n^2)$ time.

3.6. *Calculating the k MAP undirected tree structures and their relative probability weights given a prior decomposable distribution over TANs*

The problem of computing the k MST in order is well known and can be solved in $\mathcal{O}(k \cdot n^2)$ for a complete graph (Katoh, Ibaraki, & Mine, 1981). It is easy to see that if instead of computing the MST we compute the k MST and their relative weights then we have an algorithm that finds the k MAP undirected tree structures and their relative probabilities. The time complexity of the new algorithm is $\mathcal{O}((N + r^3 + k) \cdot n^2)$.

4. Decomposable distributions over tree belief networks

Decomposable distributions over tree belief networks were introduced in Meila and Jaakkola (2000a). In this section we see that the results for decomposable distributions over TANs can be applied to decomposable distributions over trees. We start redefining decomposable distributions over trees by means of decomposable distributions over TANs. After that, we provide particularizations of the results over TANs for the case of trees and doing so we correct some expressions that were wrong in Meila and Jaakkola (2000a). It is worth noting that the errors in Meila and Jaakkola (2000a) expressions do not invalidate neither the definition of decomposable distribution over trees nor any of the qualitative conclusions of that paper.

4.1. *Connection between TANs and trees*

Decomposable distributions over trees can be defined (Meila & Jaakkola, 2000a) in a similar way to the previous definition of decomposable distributions over TANs. However, it is easy to see that if we are given a domain Ω, we can easily trasform it to a classified domain Ω_{C_0} by adding a class attribute C_0 with a unique value. And that whatever we define in Ω_{C_0} can be easily projected into Ω. Hence, given a domain Ω, we can define a decomposable distribution over trees as the projection to Ω of a decomposable distribution over TANs in Ω_{C_0}.

4.2. *Meila and Jaakkola results and corrections*

In Meila and Jaakkola (2000a), some important results about decomposable distributions over trees are stated. Unfortunately, some of these results were not stated correctly. We review these results and provide corrected versions whenever needed.

4.2.1. Bayesian learning with decomposable distributions over trees. Meila and Jaakkola claimed that if we assume a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$, the posterior distribution, given a dataset $\mathcal{D}$, follows a decomposable distribution over trees with hyperparameters given by

$$\beta^*_{u,v} = \beta_{u,v} W_{u,v} \tag{27}$$

$$N'^*_{v,u}(j,i) = N'_{v,u}(j,i) + N_{v,u}(j,i) \tag{28}$$

where

$$W_{u,v} = \prod_{i \in X_u} \prod_{j \in X_v} \frac{\Gamma(N'^{*}_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))}. \tag{29}$$

Unfortunately, the last result is incorrect. In fact, the result was corrected in Meila and Jaakkola (2000b). We can obtain the corrected result by particularizing the result in Section 3.3 to Ω_{C_0} and projecting to Ω:

$$W_{u,v} = \prod_{i \in X_u} \frac{\Gamma(N'_u(i))}{\Gamma(N'^{*}_u(i))} \prod_{j \in X_v} \frac{\Gamma(N'_v(j))}{\Gamma(N'^{*}_v(j))} \prod_{i \in X_u} \prod_{j \in X_v} \frac{\Gamma(N'^{*}_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))}. \tag{30}$$

4.2.2. Computation of probabilities from the posterior. Meila and Jaakkola claimed that if we assume a decomposable prior distribution over trees with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$, the posterior probability of a new data observation conditioned to a dataset $\mathcal{D}$ is given by

$$P(\mathcal{X} = x \mid \mathcal{D}, \xi) = \frac{w_0(x)|Q(\boldsymbol{\beta}\mathbf{w}(x))|}{|Q(\boldsymbol{\beta}\mathbf{W}|} \tag{31}$$

where

$$w_0(x) = \frac{1}{N'^{*}} \prod_{X_u \in X_{-C}} \left[N'^{*}_{pa(u)}(x_{pa(u)})\right] \tag{32}$$

where $pa(u)$ is the parent of attribute $\mathcal{X}_u$ in some dependency tree and

$$\mathbf{w}(x) = (w_{u,v}(x)) \quad \text{where} \quad w_{u,v}(x) = \frac{N'^{*}_{v,u}(x_v, x_u)}{(N'^{*}_u(x_u))(N'^{*}_v(x_v))} \tag{33}$$

where Q is defined as in Section 3.2.

The previous result is also incorrect since they assume $\mathbf{W}$ defined as in Eq. (29). Furthermore, they also claim that $w_0(x)$ is a structure independent factor. In fact this is not so. To see why just consider the case where our domain contains two attributes, namely X_1 and X_2. For the tree $X_1 \rightarrow X_2$ we have that $w(x) = N'_1(x_1) + N_1(x_1)$ while for the tree $X_2 \rightarrow X_1$ we have that $w(x) = N'_2(x_2) + N_2(x_2)$. Since we have seen that the posterior is also a decomposable distribution over trees, we think it is simpler to have a result regarding the probability of a new observation given a decomposable distribution over trees, since such a result can be applied to any decomposable distribution over trees, including the posterior. Hence, particularizing the result in Section 3.2 to Ω_{C_0} and projecting to Ω we find the following result:

Assume that $P(B \mid \xi)$ follows a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$. The probability of an observation x given ξ is given by

$$P(\mathcal{X} = x \mid \xi) = h_0^x |Q(\boldsymbol{\beta}\mathbf{h}^x)| \tag{34}$$

where

$$h_0^x = \frac{1}{Z_\beta}\frac{1}{N'} \prod_{X_u \in X_{-C}} N'_u(x_u) \tag{35}$$

$$\mathbf{h}^x = \left(h_{u,v}^x\right) \quad \text{where} \quad h_{u,v}^x = \frac{N'_{v,u}(x_v, x_u)}{N'_u(x_u)N'_v(x_v)}. \tag{36}$$

It is easy to see that it can be particularized for the posterior by using the corrected result for Bayesian learning with decomposable distributions over trees given in Section 4.2.1

5. TAN classifiers based on decomposable distributions

In this section we define four classifiers based on decomposable distributions over TANs: TBMATAN, MAPTAN, MAPTAN+BMA and SSTBMATAN. We start introducing TBMATAN as the theoretically optimal classifier based on decomposable distributions over TANs. We explain that a direct implementation of TBMATAN presents some problems with floating point accuracy and propose an approximation to overcome them: SSTBMATAN. After that, we introduce MAPTAN and MAPTAN+BMA, classifiers based on the assumption that the posterior probability distribution over models is concentrated around its most probable structures. Finally, we introduce the prior distribution that we will use when no better information from the classification domain is available.

5.1. TBMATAN*: Exact Bayesian model averaging over TANs*

Putting together the results from Sections 3.2 and 3.3 we can easily design an optimal classifier based on decomposable distributions over TANs. The classifier works as follows: when given a dataset $\mathcal{D}$, it assumes that the data is generated from a TAN model and assumes a decomposable distribution over TANs as prior over the set of models. Applying the result from Section 3.3, the posterior distribution over the set of models is also a decomposable distribution over TANs, and applying the result of Section 3.2 this decomposable posterior distribution over TANs can be used to calculate the probability of any observation x. When given an unclassified observation x_{-C}, it can just calculate the probability $P(\mathcal{X}_{-C} = x_{-C}, \mathcal{C} = x_C \mid \mathcal{D}, \xi)$ for each possible class $x_C \in C$ and classify x_{-C} in the class with highest probability.

The learning time complexity for TBMATAN is bounded by the counting step, that is, $\mathcal{O}((N + r^3) \cdot n^2)$. The classification time complexity requires the computation of $\#C \leq r$ determinants. Since computing the determinant of a $n \times n$ matrix is $\mathcal{O}(n^3)$, the classification time complexity for TBMATAN is bounded by $\mathcal{O}(n^3 \cdot r)$. In comparison with STAN complexity, we have exactly the same learning time complexity, but a higher classification time complexity. This higher classification time complexity comes as a byproduct of the fact that

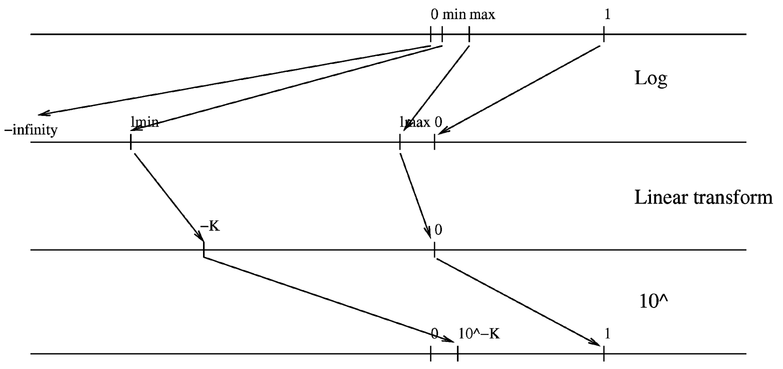


Figure 1. Transformation of weights for SSTBMATAN.

in TBMATAN we are performing exact Bayesian model averaging over an overexponential number of trees.

A straightforward implementation of TBMATAN, even when accomplishing the before mentioned complexity bounds, will not yield accurate results, specially for large datasets. This is due to the fact that the calculations that need to be done in order to classify a new observation include the computation of a determinant (in Eq. (17)) that happens to be ill-conditioned. Even worse, the determinant gets more and more ill-conditioned as the number of observations in the dataset increases. This forces the floating point accuracy, that we have to use to calculate these determinants, to depend on the dataset size. We would like to note that this problem is due to the straightforward implementation of the formulas. If it were possible to compute quotients of determinants of similar matrices accurately, the problem would be solved. To the best of our knowledge, such accurate computation does not exist. Therefore, we have used a brute force solution to accurately implement TBMATAN. More concretely, we have calculated the determinants by means of NTL (Shoup, 2003), a library that allows us to calculate determinants with the desired precision arithmetic. This solution makes the time for classifying a new observation grow faster than $\mathcal{O}(n^3 \cdot r)$, and hence makes the practical application of the algorithms difficult in situations where it is required to classify a large set of unclassified data.

5.2. SSTBMATAN*: A solution to* TBMATAN *computational problems*

We analyzed what makes the determinant in TBMATAN ill-conditioned and concluded that it is due to the $W_{u,v}$ factors given by Eq. 22. The factor $W_{u,v}$ could be interpreted as "how much the dataset $\mathcal{D}$ has changed the belief in that there is a link between u and v in the TAN model generating the data". The problems relies in the fact that $W_{u,v}$ are easily in the order of 10^{-200} for a dataset with 1500 observations. Furthermore, the factors $\frac{W_{u,v}}{W_{u',v'}}$ for such a dataset can be around 10^{-20}, providing the ill-condition of the determinant. In order to overcome this problem, we propose to postprocess the factors $W_{u,v}$ computed by Eq. (22) by means of a transformation that limits them to lie in the interval $[10^{-K}, 1]$ where K is a constant that has to be fixed depending on the floating point accuracy of the machine. In our implementation we have used $K = 5$.

The transformation works as depicted in Figure 1 and is described in detail by the following equations:

$$l\max = \log_{10} \max_{\substack{u \in X_{-C} \\ v \in X_{-C} \\ u \neq v}} W_{u,v} \tag{37}$$

$$l\min = \log_{10} \min_{\substack{u \in X_{-C} \\ v \in X_{-C} \\ u \neq v}} W_{u,v} \tag{38}$$

$$a = \begin{cases} \dfrac{K}{l\max - l\min} & l\max - l\min > K \\ 1 & \text{otherwise} \end{cases} \tag{39}$$

$$b = -K - a * l\min \tag{40}$$

$$\widetilde{W}_{u,v} = 10^{a \log_{10}(W_{u,v}) + b}. \tag{41}$$

Using $\widetilde{W}_{u,v}$ instead of $W_{u,v}$ to calculate the posterior hyperparameters $\beta^*_{u,v}$ has the following properties:

1. It is harder to get ill-conditioned determinants, because for all u, $v\widetilde{W}_{u,v}$ is bound to the interval $[10^{-K}, 1]$.
2. It preserves the relative ordering of the $W_{u,v}$. That is, if $W_{u,v} > W_{u',v'}$ then $\widetilde{W}_{u,v} > \widetilde{W}_{u',v'}$.
3. It does not exaggerate relative differences in belief. That is, for all u, v, u', v' we have that

   –If $\frac{W_{u,v}}{W_{u',v'}} \geq 1$ then $\frac{W_{u,v}}{W_{u',v'}} \geq \frac{\widetilde{W}_{u,v}}{\widetilde{W}_{u',v'}}$.
   –If $\frac{W_{u,v}}{W_{u',v'}} \leq 1$ then $\frac{W_{u,v}}{W_{u',v'}} \leq \frac{\widetilde{W}_{u,v}}{\widetilde{W}_{u',v'}}$.

The posterior hyperparameters $\beta^*_{u,v}$ can be interpreted as a representation of the a posteriori belief in the existence of an edge (u, v) in the TAN structure. Using $\widetilde{W}_{u,v}$, given the properties stated, means being more conservative in the structure learning process, because the beliefs will be confined to the interval $[10^{-K}, 1]$ which impedes the representation of extreme probability differences between edges. We can interpret the transformation as applying some stubbornness to the structure learning process. Applying this transformation allows us to implement an approximation of TBMATAN that does not require the use of special floating point accuracy computations. We will refer to this approximation of TBMATAN as SSTBMATAN (from Structure Stubborn TBMATAN).

It is worth noting that the problem described in this section does only affect the classification time complexity. The learning process for TBMATAN does not need high precision arithmetics. The learning time complexity for TBMATAN, $\mathcal{O}((N + r^3) \cdot n^2)$, is the same as the one for STAN and SSTBMATAN.

When we have enough data, it is likely that the posterior over models is concentrated around their highest values. If we accept that, we can approximate the results of TBMATAN by selecting a small set of models and their relative probability weights or, in the extreme case, by selecting a single model. We introduce now MAPTAN (Maximum a Posteriori TAN) and MAPTAN+BMA based on these ideas and on the results described in Section 3.

5.3. MAPTAN*: Learning a single TAN*

The learning steps for the MAPTAN classifier consist in:

1. Assume a decomposable distribution over TANs as prior
2. Find the undirected tree E underlying the MAP TAN structure given a dataset $\mathcal{D}$.
3. Choose any root, create a directed tree $\bar{E}$ and from it a directed TAN structure $\bar{E}^*$.
4. Use Eq. (24) applied over the posterior to fix the TAN parameters.

For classifying an unclassified observation, we have to apply the TAN that has been learned for each of the $\#C$ classes to construct a probability distribution over the values of the class C and then choose the most probable class. This classification algorithm runs in $\mathcal{O}((N + r^3) \cdot n^2)$ learning time and $\mathcal{O}(nr)$ classification time.

5.4. MAPTAN+BMA*: Learning an ensemble of TANs*

The learning steps for MAPTAN+BMA classifier consist in:

1. Assume a decomposable distribution over TANs as prior
2. Find the k undirected trees underlying the k MAP TAN structures and their relative probability weights given a dataset $\mathcal{D}$.
3. Generate a TAN model for each of the undirected tree structures as we did in MAPTAN.
4. Assign to each TAN model the weight of its corresponding undirected tree.

The resulting probabilistic model will be a mixture of TANs.

For classifying an unclassified observation, we have to apply the k TAN models for the $\#C$ classes and calculate the weighted average to construct a probability distribution over the values of the class C and then choose the most probable class.

This classification algorithm runs in $\mathcal{O}((N + r^3 + k) \cdot n^2)$ learning time and $\mathcal{O}(nrk)$ classification time.

5.5. *The prior*

The four classifiers we have presented assume a decomposable distribution over TANs as prior. Ideally, this prior will be fixed by an expert that knows the classification domain. Otherwise, we have to provide the classifier with a way of fixing the prior distribution hyperparameters without knowledge about the domain. In this case the prior should be as "non-informative" as possible in order for the information coming from $\mathcal{D}$ to dominate the posterior by the effects of Eqs. (20) and (21). We have translated this requisite into Eqs. (42) and (43):

$$\forall u, v; 1 \leq u \neq v \leq n; \beta_{u,v} = 1 \tag{42}$$

$$\forall u, v; 1 \leq u \neq v \leq n; \forall j \in A_v; \forall i \in A_u; \forall c \in C;$$
$$N'_{v,u,C}(j, i, c) = \frac{\lambda}{\#C\#A_u\#A_v}. \tag{43}$$

Defining $\boldsymbol{\beta}$ as in Eq. (42) means that we have the same amount of belief for any edge being in the TAN structure underlying the data. For fixed u, v, Eq. (43) assigns the same probability to any (j, i, c) such that $j \in A_v$, $i \in A_u$ and $c \in C$.

The hyperhyperparameter λ is an "equivalent sample size" for the prior in the sense of Heckerman, Geiger, and Chickering (1995). Experimental tests have shown that the algorithms are relatively stable to the choice of λ provided that every $N'_{v,u,C}(j, i, c) \geq 1$. In our experiments we have selected the minimal λ such that this condition is fulfilled.

5.6. *Comments*

We have shown that decomposable distributions over TANs can be used to construct classifiers based on TAN models. Comparing with the STAN classifier introduced in Friedman, Geiger, and Goldszmidt (1997), classifiers based on decomposable distributions show some theoretical advantages.

First, STAN disregards uncertainty over models. If the posterior distribution over models is not concentrated around its peak, this could affect the accuracy of the classifier. We can foresee two main situations where the posterior is likely not to be concentrated. The first one is when there is little data available. The second one is when the distribution underlying the data is not a TAN model. In these cases, classifiers that take into account uncertainty over models, such as TBMATAN, SSTBMATAN or MAPTAN+BMA are likely to obtain better results.

Second, regarding the determination of the TAN structure, STAN relies on the maximum likelihood principle. This gives the algorithm a theoretical guarantee that it is assymptotically correct. That is, the algorithm will have a good performance given enough data. On the other hand, no guarantee is provided when not enough data is available, and we are given no indication on the number of instances needed for the convergence of the algorithm. This contrasts with the result for finding the MAP TAN structure provided in Section 3.5. The result given there guarantees that the TAN structure determined is the most probable, independently of the amount of data at our disposal. This means that, again when little data is available, MAPTAN is likely to obtain better results.

Third, regarding the way of fixing the parameters, whilst STAN is inspired on Bayesian principles, MAPTAN and MAPTAN+BMA rely on a well founded result showing that there is a single TAN model that predicts as the Bayesian model averaging over parameters of models with a fixed structure. In our opinion this is more appealing from a theoretical point of view, and gives a reasonable explanation on why it makes sense to soften the parameters instead of choosing the parameters resulting from the application of the maximum likelihood principle.

Fourth, regarding prior information, STAN algorithm does not take into account any, while TAN classifiers based on decomposable distributions allow the use of some form of prior information if available, specially structure related information. For example, if we have expert knowledge that tells us that one of the edges of the tree is much more (or much less) likely than the others it is very easy to incorporate this knowledge when fixing the prior hyperparameter matrix $\boldsymbol{\beta}$. Evidently, as was pointed out in Meila and Jaakkola (2000a), decomposable distributions do not allow the expression of some types of prior information such as "if edge (u, v) exists then edge (w, z) is very likely to exist".

The first of these four advantages requires a higher computational complexity from the classifier, but the others come at no price. Concretely it is worth pointing out that MAPTAN has exactly the same computational complexity as STAN.

6. Empirical comparison

In order to evaluate the classifiers we performed two sets of experiments. In the first one we tested their performances over Irvine datasets. In the second we tested them over randomly generated Bayesian networks with different sets of parameters. In the following sections, we explain the experiment setups and then show the results and draw some conclusions.

6.1. General setup

The algorithms compared are STAN, MAPTAN, MAPTAN+BMA, SSTBMATAN and TBMATAN. STAN refers to the algorithm presented in Friedman, Geiger, and Goldszmidt (1997) as TAN^s. For the learning algorithms based on decomposable distribution the prior is assumed to be the one described in Section 5.5 with λ as prescribed in that section. For MAPTAN+BMA the number of trees in the mixture was fixed to $k = 10$, because in our implementation this value provided a classification time between the ones of MAPTAN and SSTBMATAN. For SSTBMATAN, K was fixed to 5.

The measure used to compare the performance of the algorithms is the area under the ROC curve (Fawcett, 2003) which we will refer to as AUC. When the class has more than two values, we use the formula provided in Hand and Till (2001), that is based on the idea that if we can compute the AUC for two classes i, j (let us denote this by $A(i, j)$), then we can compute an extension of AUC for any arbitrary number of classes by choosing all the possible pairs (1 vs. 1). Since $A(i, j) = A(j, i)$, this can be simplified as shown in the following expression:

$$AUC = \frac{2}{\#C(\#C-1)} \sum_{i<j} A(i, j). \tag{44}$$

6.2. *Irvine setup*

We run each of the algorithms over 23 Irvine datasets: adult*, australian, breast, car, chess*, cleve, crx, flare, glass, glass2, hep, iris, letter*, liver, lymphography, mushroom*, pima, nursery*, primary-tumor, shuttle-small*, soybean, vehicle and votes. Due to computational constraints, for the larger datasets marked with a*, we used 10 fold CV (cross validation) and we did not run TBMATAN. For the datasets which are not marked, we repeated 5 times 10 fold CV. For each learning fold, we learnt with 10%, 50%, and 100% of the learning data to evaluate how the classifier improves with increasing data.

6.3. *Random Bayesian networks setup*

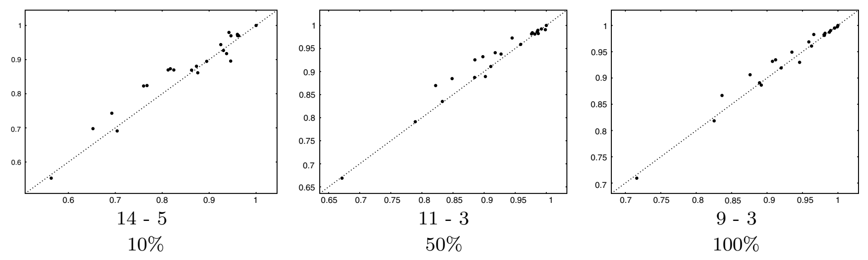
We wanted to evaluate our algorithms over artificial domains that could resemble real life datasets. In order to do that we generated random Bayesian networks with different characteristics, using BNGenerator (Ide & Cozman, 2003). We varied three characteristics: the number of attributes of the dataset, the number of maximum values of an attribute and the maximum induced width of the network. In Ide and Cozman (2003), it is argued that a network with low induced width "looks like" real networks in the literature and hence, the most appropriate parameter to control a network density when generating Bayesian networks is the induced width.

We compared our algorithms over networks varying the number of attributes in $\{5,10,20,40\}$, the number of maximum values of an attribute in $\{2,5,10\}$ and the maximum induced width in $\{2,3,4\}$. For each configuration of parameters we generated randomly 100 Bayesian networks. For each Bayesian network we obtained 4 learning samples of sizes $\{25,100,400,1600\}$ and a testing sample of size 100. For TBMATAN evaluation we dropped the bigger learning sample, due to computational constraints.

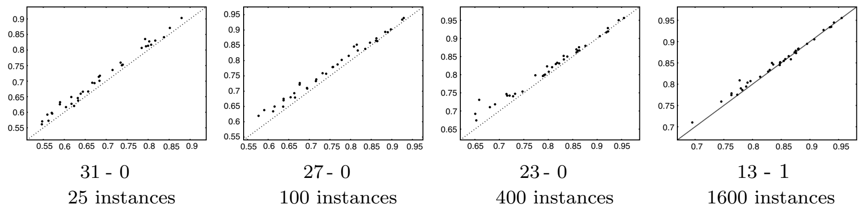
We also compared the algorithms when the underlying model is a random TAN, varying attributes as described before, except for the induced width, that was not controlled.

6.4. *Analysis of results*

In this section we draw some conclusions from the analysis of the results of the experimental work. We start comparing STAN with MAPTAN, because both classifiers have the same computational complexity. We will see that MAPTAN provides consistently better predictions than STAN, and that the improvement increases as the amount of data gets smaller or the number of different values of the attributes grows. Then we will compare SSTBMATAN, our proposed approximation to avoid floating point accuracy problems to the optimal TBMATAN. We will see that SSTBMATAN most of the times improves over TBMATAN, and will argue that this is due to the fact that the assumptions under TBMATAN are not exactly fulfilled, and hence the increased softening provided by SSTBMATAN is benefitial. Finally, we will compare MAPTAN, as a classifier learning a single model, to MAPTAN+BMA and SSTBMATAN, classifiers based on Bayesian model averaging. We will see that performing Bayesian model averaging usually provides significant improvements, specially for the case of SSTBMATAN.



(a) On 23 Irvine datasets



(b) On random Bayesian networks

Figure 2. MAPTAN VS. STAN. AUC comparison across learning data size.

***6.4.1.* STAN *versus* MAPTAN.** The relationship between STAN and MAPTAN as data increases appears in Figure 2. We present two sets of plots. In Figure 2(a) we can see from left to right the evolution of the scatter plot of the AUC of STAN and MAPTAN for Irvine datasets as we increase the amount of data in the learning set. Below each scatter plot we present statistical significance results showing the number of datasets where each classifier outperforms the other at a level of significance of 95% (*t*-test). Below the significance test results we can see the percentage of training data that was used. In Figure 2(b) we follow the same schema for randomly generated Bayesian networks, presenting scatter plots and significance tests figures when we use samples of 25, 100, 400 and 1600 instances for learning. Both over Irvine and artificial datasets we can see that when the amount of data at our disposal is small, MAPTAN outperforms STAN, and as data size grows the difference diminishes. We performed an additional experiment in order to test whether, following this tendency, STAN finally outperforms MAPTAN. In order to do that we selected the smallest sampling size space (5 attributes and 2 values per attribute) and increased the data size up to more than 100.000 instances. Under this setting, MAPTAN results were usually over STAN results and STAN was never found to be statistically significantly better than MAPTAN, reinforcing our belief that both classifiers converge to the same model given enough data.

In Figure 3 we can see that MAPTAN improves significantly over STAN and that this improvement is more significant as the number of maximum values of the attributes grows. This increase can not be noticed as the number of attributes grows. This is understandable, because the size of the sampling space is $\mathcal{O}(r^n)$, and hence is much more sensible to an increase in r than to an increase in n. The difference for both algorithms was stable along the different maximum induced width values we tested.

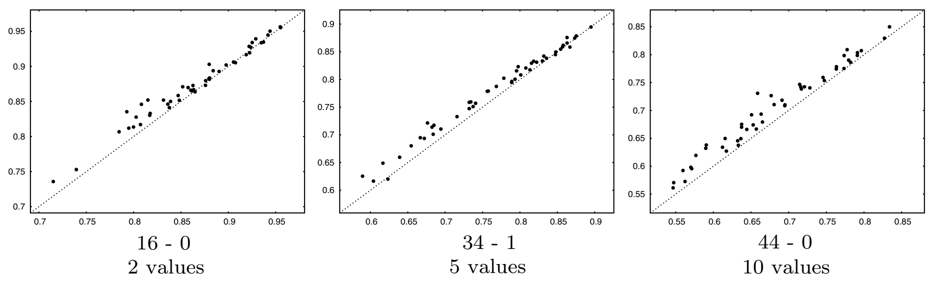


Figure 3. MAPTAN VS. STAN. AUC comparison across number of values of the attributes.

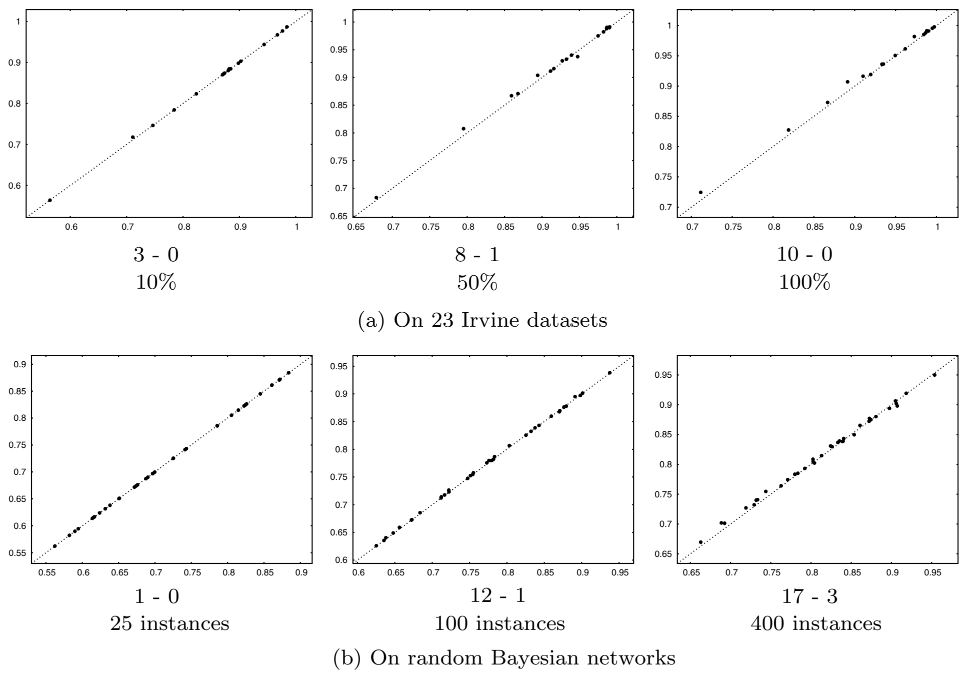


Figure 4. SSTBMATAN VS. TBMATAN. AUC comparison accross learning data size.

***6.4.2.* SSTBMATAN *versus* TBMATAN.** In Figure 4 we can see that both classifiers give approximately the same results when provided a small sample, but as we increase the number of observations, SSTBMATAN significantly improves TBMATAN in many cases. The difference also increases as the maximum number of values per attribute grows (see Figure 5). The difference between SSTBMATAN and TBMATAN sligthly benefits SSTBMATAN on a stable way along the different maximum induced width values and the different number of attributes we tested.

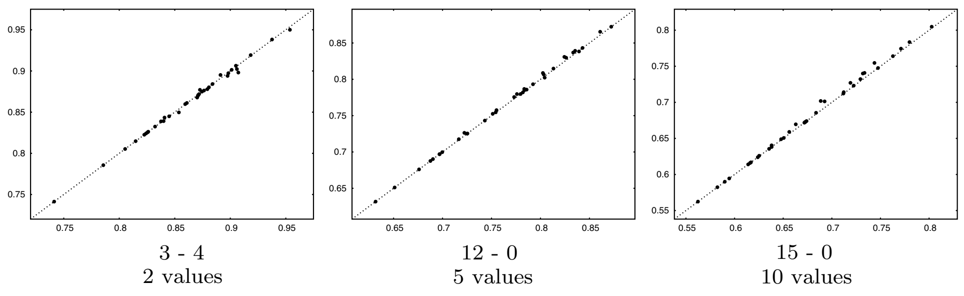


Figure 5. SSTBMATAN VS. TBMATAN. AUC comparison across number of values of the attributes.

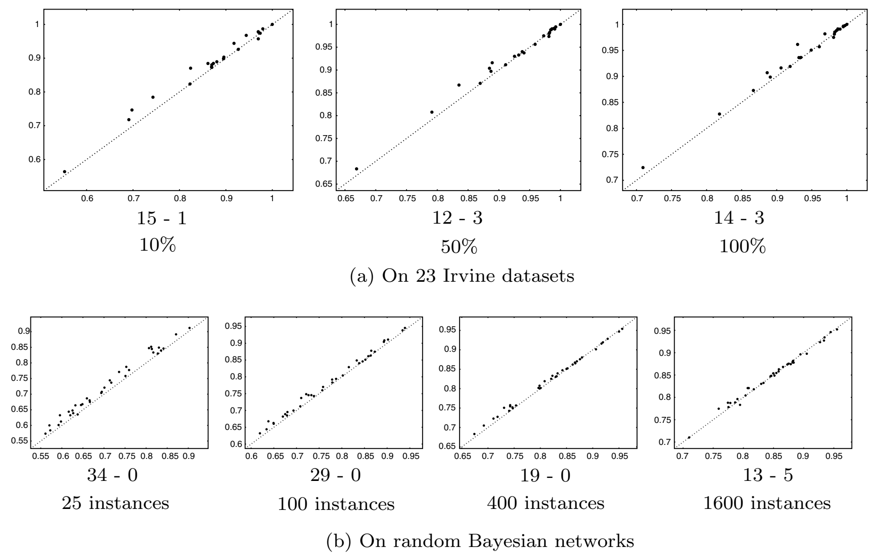


Figure 6. SSTBMATAN VS. MAPTAN. AUC comparison across learning data size.

The fact that SSTBMATAN improves over the theoretically optimal TBMATAN can be understood if we analyze on what do the two algorithms differ. The only difference is that SSTBMATAN is more conservative (stubborn) in terms of changing its probability distribution over structures. In the datasets we are analyzing the underlying distributions are not TAN distributions and hence the assumptions for TBMATAN are not fulfilled. Being more conservative regarding what is known about the structure turns out to be good in this setting. In order to double check that claim, we ran additional experiments were data was sampled from TAN distributions with sample sizes in $\{25,100,400\}$. On those experiments both algorithms returned almost equivalent results:TBMATAN provided sligthly better AUC, but the biggest difference in AUC was in the order of 0.001.

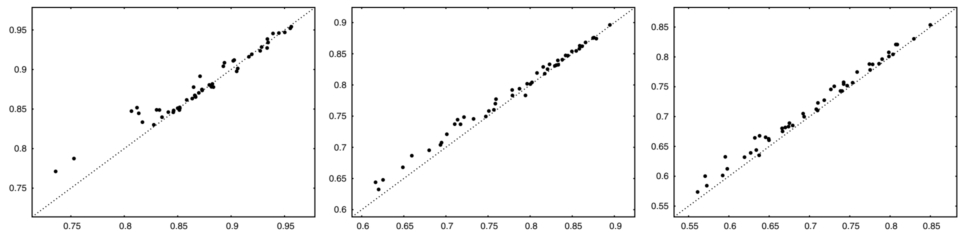


Figure 7. SSTBMATAN VS. MAPTAN. AUC comparison across number of values of the attributes.

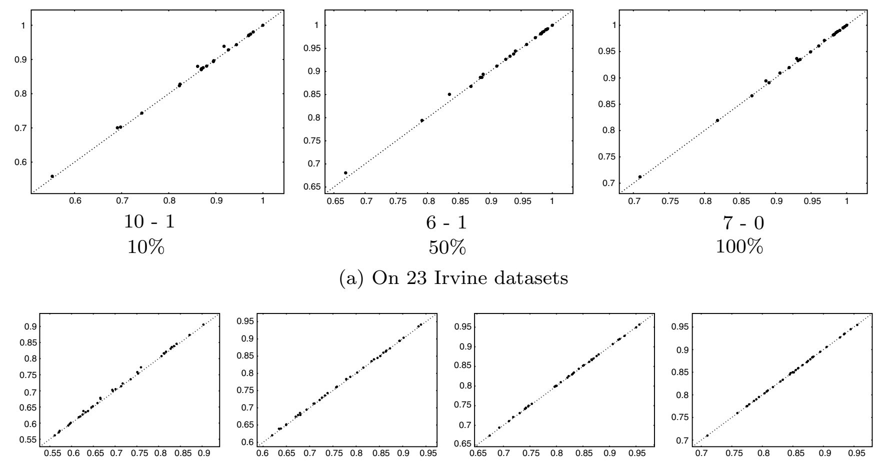


Figure 8. MAPTAN+BMA. VS MAPTAN. AUC comparison across learning data size.

***6.4.3.* SSTBMATAN *versus* MAPTAN.** We can see in Figure 6(b) that SSTBMATAN improves significantly over MAPTAN for many datasets and that as we increase the amount of data, the difference between SSTBMATAN and MAPTAN decreases. The opposite happens as the number of maximum values per attribute grows (see Figure 7). This can be understood because as the amount of data grows, our uncertainty over the set of models decreases, and so does the improvement obtained by taking this uncertainty into account.

On Irvine datasets. (Figure 6(a)), SSTBMATAN improves significantly over MAPTAN for many datasets, but no clear tendency is appreciated as we increase the amount of data.

***6.4.4.* MAPTAN+BMA *versus* MAPTAN.** MAPTAN+BMA improves over MAPTAN in a statistically significant way over some datasets, specially when the amount of learning data is small (see Figure 8). However, the differences are much smaller than for SSTBMATAN and do not seem very rellevant from a practical point of view. No significant direct dependence on the number of attributes, maximum number of values of the attributes, or maximum induced width has been detected.

7. Conclusions and future work

In this paper, we have focused on improving Bayesian network classifiers based on trees and TANs. We have done that by following Bayesian probability theory, that is by defining a conjugate distribution for these families of models: decomposable distributions. We have introduced decomposable distributions over TANs, extending the results of Meila and Jaakkola to TANs. Then, we have corrected the results for decomposable distributions over trees. After that, we have proposed four classifiers based on decomposable distributions over TANs. Finally, we have shown that these classifiers provide clearly significant improvements, specially when data is scarce. Furthermore, our classifiers allow the user to provide the classifier with some prior information, if such is available.

Of the four classifiers we have introduced, TBMATAN should be discarded for practical use due to computational reasons. There are three variables to take into account in order to select which of the other three classifiers should be applied. The first one is the value ratio between efficiency and accuracy (how much are we willing to pay in computing time for a given increase in accuracy). When this ratio is small we should use MAPTAN and as it grows we should switch to MAPTAN+BMA, and then SSTBMATAN. The second one is the posterior level of uncertainty in the models. Again, when it is small we should use MAPTAN and as it grows we should switch to MAPTAN+BMA, and then SSTBMATAN. The third variable to take into account is the amount of the sampling space covered by our learning data. When our learning data size is small compared to our sampling space, we should use SSTBMATAN, and as it grows we should switch to MAPTAN+BMA and then to MAPTAN.

Three future lines of work arise. The first one is the design of a classifier that is "conscious" of the relevance of the uncertainty in models and hence able to choose between SSTBMATAN, TBMATAN, and MAPTAN. The second one is constructing an algorithm that instead of learning the MAP TAN structure learns a probably MAP TAN structure, that is a TAN structure that coincides with the MAP TAN structure with probability $1 - \delta$. Such an algorithm could be used in learning from infinite sequences of data. The third one is the extension of decomposable distributions to other tree based families (see Friedman, Geiger, and Goldszmidt, 1997).

Appendix A: Preliminaries

In this appendix we introduce two results that will be needed in the further development and then in Appendix B we prove the results in Sections 3.2 and 3.3.

A.1. The matrix tree theorem for decomposable distributions

Let $P(E)$ be a distribution over spanning tree structures defined by Eqs. (6) and (7). Then the normalization constant Z_β is equal to $|Q(\boldsymbol{\beta})|$ where Q is defined as in Section 3.2.

Proof: See Meila and Jaakkola (2000b). $\square$

A.2. A useful result about Dirichlet distributions

A Dirichlet distribution is defined as

$$D(\theta_1, \ldots, \theta_k; N_1, \ldots, N_k) = \frac{\Gamma\left(\sum_{i=1}^{k} N_i\right)}{\prod_{i=1}^{k} \Gamma(N_i)} \prod_{i=1}^{k} {\theta_i}^{N_i - 1}. \tag{45}$$

Let $D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r)$ be a Dirichlet distribution. We have that:

$$\begin{aligned} & D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r) \prod_{i=1}^{r} {\theta_i}^{n_i} \\ & = \frac{\Gamma\left(\sum_{i=1}^{r} n'_i\right)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i + n_i)}{\Gamma\left(\sum_{i=1}^{r} n'_i + n_i\right)} D(\theta_1, \ldots, \theta_r; n'_1 + n_1, \ldots, n'_r + n_r) \end{aligned} \tag{46}$$

and since the Dirichlet distribution is normalized you have that

$$\int \cdots \int_{\theta_1, \ldots, \theta_r} D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r) \prod_{i=1}^{r} {\theta_i}^{n_i} = \frac{\Gamma\left(\sum_{i=1}^{r} n'_i\right)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i + n_i)}{\Gamma\left(\sum_{i=1}^{r} n'_i + n_i\right)} \tag{47}$$

Appendix B: Detailed development for decomposable distributions over TANs results

In this appendix we provide the proofs for the results in Sections 3.2 and 3.3.

B.1. Calculating probabilities under decomposable distributions over TANs

Knowing that $P(B \mid \xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$ we need to calculate

$$P(\mathcal{X} = x \mid \xi) = \int_{B \in \mathcal{B}} P(\mathcal{X} = x \mid B, \xi) P(B \mid \xi). \tag{48}$$

This integral can be calculated by first obtaining the probability of each structure and then performing an addition over the set of structures.

$$P(\mathcal{X} = x \mid \xi) = \sum_{\bar{E} \in \bar{\mathcal{E}}} P(\mathcal{X} = x \mid \bar{E}, \xi) P(\bar{E} \mid \xi). \tag{49}$$

B.1.1. Calculating $P(\mathcal{X} = x \mid \bar{E}, \xi)$. Since $\bar{E}$ uniquely determines $\bar{E}^*$ we have that

$$P(\mathcal{X} = x \mid \bar{E}, \xi) = \int \cdots \int_{\Theta_{\bar{E}^*}} P(\mathcal{X} = x \mid \bar{E}^*, \Theta_{\bar{E}^*}) P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi) d\Theta_{\bar{E}^*}. \tag{50}$$

$P(\mathcal{X} = x \mid \bar{E}^*, \Theta_{\bar{E}^*})$ is determined by the expansion of Eq. (1) taking into account the TAN structure

$$P(\mathcal{X} = x \mid \bar{E}^*, \Theta_{\bar{E}^*}) = \theta_C(x_C) \theta_{\rho_{\bar{E}}|C}\left(s_{\rho_{\bar{E}}}, x_C\right) \prod_{u,v \in \bar{E}} \theta_{v|u,C}(x_v, x_u, x_C). \tag{51}$$

$P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi)$ can be expanded from Eqs. (8), (12), (13) and (14) into

$$\begin{aligned} P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi) = {} & D(\theta_C(.); N'_C(.)) \\ & \times \prod_{c \in C} D\left(\theta_{\rho_{\bar{E}}|C}(., c); N'_{\rho_{\bar{E}},C}(., c)\right) \\ & \times \prod_{c \in C} \prod_{u,v \in \bar{E}} \prod_{i \in A_u} D(\theta_{v|u,C}(., i, c); N'_{v,u,C}(., i, c)). \end{aligned} \tag{52}$$

Now we need to calculate the integral in Eq. (50). In order to do this we define:

$$1^x_C(c) = \begin{cases} 1 & c = x_C \\ 0 & \text{otherwise} \end{cases} \tag{53}$$

$$1^x_{u,C}(i, c) = \begin{cases} 1 & i = x_u \wedge c = x_C \\ 0 & \text{otherwise} \end{cases} \tag{54}$$

$$1^x_{v,u,C}(j, i, c) = \begin{cases} 1 & j = x_v \wedge i = x_u \wedge c = x_C \\ 0 & \text{otherwise} \end{cases} \tag{55}$$

$$N'^x{}_C(c) = N'{}_C(c) + 1^x_C(c) \tag{56}$$

$$N'^x{}_{u,C}(i, c) = N'{}_{u,C}(i, c) + 1^x_{u,C}(i, c) \tag{57}$$

$$N'^x{}_{v,u,C}(j, i, c) = N'{}_{v,u,C}(j, i, c) + 1^x_{v,u,C}(j, i, c). \tag{58}$$

It is easy to see that:

$$\sum_{j\in A_v} 1^x_{v,u,C}(j,i,c) = 1^x_{u,C}(i,c) \tag{59}$$

$$\sum_{i\in A_u} 1^x_{u,C}(i,c) = 1^x_C(c) \tag{60}$$

$$\sum_{c\in C} 1^x_C(c) = 1. \tag{61}$$

We can use this notation to expand the product

$$P(\mathcal{X} = x \mid \bar{E}^*, \Theta_{\bar{E}^*}) P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi) \tag{62}$$

by substituting Eqs. (51) and (52) giving:

$$\begin{aligned} P(\mathcal{X} = x \mid \bar{E}^*, \Theta_{\bar{E}^*}) P(\Theta_{\bar{E}^*} \mid \bar{E}^*, \xi) &= D(\theta_C(.); N'_C(.) N'_C(.)) \prod_{c\in C} \theta_C(c)^{1^x_C(c)} \\ \times \prod_{c\in C} \Bigg[D\big(\theta_{\rho_{\bar{E}}|C}(.,c); N'_{\rho_{\bar{E}},C}(.,c)\big) &\prod_{i\in A_{\rho_{\bar{E}}}} \theta_{\rho_{\bar{E}}|C}(i,c)^{1^x_{\rho_{\bar{E}},C}(i,c)} \Bigg] \\ \times \prod_{c\in C} \prod_{u,v\in\bar{E}} \prod_{i\in A_u} \Bigg[D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) &\prod_{j\in A_v} \theta_{v|u,C}(j,i,c)^{1^x_{v,u,C}(j,i,c)} \Bigg]. \end{aligned} \tag{63}$$

By analyzing Eq. (63) we can see that the integral in Eq. (50) can be calculated by applying the result in Eq. (47) three times:

$$\begin{aligned} P(\mathcal{X} = x \mid \bar{E}, \xi) &= \frac{\Gamma\big(\sum_{c\in C} N'_C(c)\big)}{\prod_{c\in C} \Gamma(N'_C(c))} \frac{\prod_{c\in C} \Gamma(N'^x_C(c))}{\Gamma\big(\sum_{c\in C} N'^x_C(c)\big)} \\ &\times \prod_{c\in C} \left[\frac{\Gamma\big(\sum_{i\in A_{\rho_{\bar{E}}}} N'_{\rho_{\bar{E}},C}(i,c)\big)}{\prod_{i\in A_{\rho_{\bar{E}}}} \Gamma(N'_{\rho_{\bar{E}},C}(i,c))} \frac{\prod_{i\in A_{\rho_{\bar{E}}}} \Gamma(N'^x_{\rho_{\bar{E}},C}(i,c))}{\Gamma\big(\sum_{i\in A_{\rho_{\bar{E}}}} N'^x_{\rho_{\bar{E}},C}(i,c)\big)} \right] \\ &\times \prod_{c\in C} \prod_{u,v\in\bar{E}} \prod_{i\in A_u} \left[\frac{\Gamma\big(\sum_{j\in A_v} N'_{v,u,C}(j,i,c)\big)}{\prod_{j\in A_v} \Gamma(N'_{v,u,C}(j,i,c))} \frac{\prod_{i\in A_v} \Gamma(N'^x_{v,u,C}(j,i,c))}{\Gamma\big(\sum_{i\in A_v} N'^x_{v,u,C}(j,i,c)\big)} \right]. \end{aligned} \tag{64}$$

This expression can be simplified by applying Eqs. (9)–(11),(59)–(61) and reorganizing:

$$P(\mathcal{X}=x \mid \bar{E},\xi) = \frac{\Gamma(N')}{\Gamma(N'+1)} \prod_{c\in C} \prod_{i\in A_{\rho_{\bar{E}}}} \frac{\Gamma(N'^{x}_{\rho_{\bar{E}},C}(i,c))}{\Gamma(N'_{\rho_{\bar{E}},C}(i,c))}$$
$$\times \prod_{u,v\in\bar{E}} \prod_{c\in C} \prod_{i\in A_u} \left[\frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'^{x}_{u,C}(i,c))} \prod_{i\in A_v} \frac{\Gamma(N'^{x}{}_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))} \right]. \tag{65}$$

Since the quotient $\frac{\Gamma(\overset{x}{N'}_*(*))}{\Gamma(N'_*(*))}$ is $N'_*(*)$ if the condition is satisfied and 1 otherwise we have that:

$$P(\mathcal{X}=x \mid \bar{E},\xi) = \frac{1}{N'} N'_{\rho_{\bar{E}},C}(x_{\rho_{\bar{E}}},x_C) \prod_{u,v\in\bar{E}} \left[\frac{N'_{v,u,C}\big(x_v,x_u,x_C\big)}{N'_{u,C}(x_u,x_C)} \right]. \tag{66}$$

Defining h_0^x and $h_{u,v}^x$ as in Eq. (18) and (19) it is easy to see that multiplying and dividing in Eqs. (66) by the factor:

$$\prod_{v\in X_{-C}-\{\rho_{\bar{E}}\}} N'_{v,C}(x_v,x_C) \tag{67}$$

and rearranging we get an expression that depends only on the undirected structure of the tree, allowing the definition of $P(\mathcal{X}=x \mid E,\xi)$:

$$P(\mathcal{X}=x \mid E,\xi) = P(\mathcal{X}=x \mid \bar{E},\xi) = h_0^x Z_\beta \prod_{u,v\in E} h_{u,v}^x. \tag{68}$$

B.1.2. Adding over tree structures The summation over directed structures can be calculated by means of a summation over undirected structures where for each undirected structure we add over the directed structures that can be derived from it.

$$\begin{aligned} P(\mathcal{X}=x \mid \xi) &= \sum_{E\in\mathcal{E}} \left(\sum_{\bar{E}\in Or(E)} P(\mathcal{X}=x \mid \bar{E},\xi) P(\bar{E} \mid E,\xi) \right) P(E \mid \xi) \\ &= \sum_{E\in\mathcal{E}} P(\mathcal{X}=x \mid E,\xi) \left(\sum_{\bar{E}\in Or(E)} P(\bar{E} \mid E,\xi) \right) P(E \mid \xi) \\ &= \sum_{E\in\mathcal{E}} P(\mathcal{X}=x \mid E,\xi) P(E \mid \xi). \end{aligned} \tag{69}$$

Combining Eqs. (6) and (68) we get

$$P(\mathcal{X} = x \mid E, \xi)P(E \mid \xi) = h_0^x \prod_{u,v \in E} \left(\beta_{u,v} h_{u,v}^x\right). \tag{70}$$

Calculating the summation over structures using the matrix tree theorem for decomposable distributions gives the desired result.

$$P(\mathcal{X} = x \mid \xi) = h_0^x |Q(\boldsymbol{\beta}\mathbf{h}^x)|. \tag{71}$$

B.2. Learning under decomposable distributions over TANs

Given that $P(B \mid \xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$ we want to calculate $P(B \mid \mathcal{D}, \xi)$ where $\mathcal{D}$ is an i.i.d. dataset sampled from a TAN distribution.Using Bayes rule we get:

$$P(B \mid \mathcal{D}, \xi) = P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \mathcal{D}, \xi) = \frac{P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \xi) P(\mathcal{D} \mid \bar{E}^*, \Theta_{\bar{E}^*}, \xi)}{Z_{\mathcal{D}}}. \tag{72}$$

The prior $P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \xi)$ is calculated combining Eqs. (4)–(6), and (52) giving:

$$\begin{aligned} P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \xi) = {} & P(\bar{E} \mid E) \frac{1}{Z_\beta} \prod_{u,v \in \bar{E}} \beta_{u,v} D(\theta_C(.); N'_C(.)) \\ & \times \prod_{c \in C} D\left(\theta_{\rho_{\bar{E}}|C}(., c); N'_{\rho_{\bar{E}},C}(., c)\right) \\ & \times \prod_{c \in C} \prod_{u,v \in \bar{E}} \prod_{i \in A_u} D(\theta_{v|u,C}(., i, c); N'_{v,u,C}(., i, c)). \end{aligned} \tag{73}$$

$P(\mathcal{D} \mid \bar{E}^*, \Theta_{\bar{E}^*}, \xi)$ is the probability that the model generates the data in $\mathcal{D}$. Since $\mathcal{D}$ contains i.i.d. observations, we have that

$$\begin{aligned} P(\mathcal{D} \mid \bar{E}^*, \Theta_{\bar{E}^*}, \xi) = {} & \prod_{c \in C} \theta_C(c)^{N_C(c)} \prod_{c \in C} \prod_{i \in A_{\rho_{\bar{E}}}} \theta_{\rho_{\bar{E}}|C}(i, c)^{N_{\rho_{\bar{E}},C}(i,c)} \\ & \times \prod_{c \in C} \prod_{u,v \in \bar{E}} \prod_{i \in A_u} \prod_{j \in A_v} \theta_{v|u,C}(j, i, c)^{N_{v,u,C}(j,i,c)}. \end{aligned} \tag{74}$$

Substituting Eqs. (73) and (74) into (72) and then applying the result in Eq. (46) for all the Dirichlets and the notation in Eq. (21),

$$\begin{aligned}
P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \mathcal{D}, \xi) = {} & P(\bar{E} \mid E)\frac{1}{Z_\beta}\frac{1}{Z_{\mathcal{D}}}\prod_{u,v\in\bar{E}}\beta_{u,v} \\
& \times \frac{\Gamma\left(\sum_{c\in C} N'_C(c)\right)}{\prod_{c\in C}\Gamma(N'_C(c))}\frac{\prod_{c\in C}\Gamma(N'^*_C(c))}{\Gamma\left(\sum_{c\in C} N'^*_C(c)\right)} \\
& \times \prod_{c\in C}\left[\frac{\Gamma\left(\sum_{i\in A_{\rho_{\bar{E}}}} N'_{\rho_{\bar{E}},C}(i,c)\right)}{\prod_{i\in A_{\rho_{\bar{E}}}}\Gamma(N'_{\rho_{\bar{E}},C}(i,c))}\frac{\prod_{i\in A_{\rho_{\bar{E}}}}\Gamma(N'^*_{\rho_{\bar{E}},C}(i,c))}{\Gamma\left(\sum_{i\in A_{\rho_{\bar{E}}}} N'^*_{\rho_{\bar{E}},C}(i,c)\right)}\right] \\
& \times \prod_{c\in C}\prod_{u,v\in\bar{E}}\prod_{i\in A_u}\left[\frac{\Gamma\left(\sum_{j\in A_v} N'_{v,u,C}(j,i,c)\right)}{\prod_{j\in A_v}\Gamma(N'_{v,u,C}(j,i,c))}\frac{\prod_{j\in A_v}\Gamma(N'^*_{v,u,C}(j,i,c))}{\Gamma\left(\sum_{j\in A_v} N'^*_{v,u,C}(j,i,c)\right)}\right] \\
& \times D(\theta_C(.); N'^*_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\bar{E}}|C}(.,c); N'^*_{\rho_{\bar{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\bar{E}}\prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'^*_{v,u,C}(.,i,c)).
\end{aligned} \tag{75}$$

This expression can be simplified by applying Eqs. (9)–(11) (and similar ones for N'^*) and reorganizing:

$$\begin{aligned}
P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \mathcal{D}, \xi) = {} & P(\bar{E} \mid E)\frac{1}{Z_\beta}\frac{1}{Z_{\mathcal{D}}}\frac{\Gamma(N')}{\Gamma(N'^*)}\prod_{u,v\in\bar{E}}\beta_{u,v} \\
& \times \prod_{c\in C}\prod_{i\in A_{\rho_{\bar{E}}}}\frac{\Gamma(N'^*_{\rho_{\bar{E}},C}(i,c))}{\Gamma(N'_{\rho_{\bar{E}},C}(i,c))} \\
& \times \prod_{u,v\in\bar{E}}\prod_{c\in C}\prod_{i\in A_u}\left[\frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'^*_{u,C}(i,c))}\prod_{j\in A_v}\frac{\Gamma(N'^*_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))}\right] \\
& \times D(\theta_C(.); N'^*_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\bar{E}}|C}(.,c); N'^*_{\rho_{\bar{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\bar{E}}\prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'^*_{v,u,C}(.,i,c)).
\end{aligned} \tag{76}$$

Defining $W_{u,v}$ as appears in Eq. (22), it is easy to see that multiplying and dividing in Eq. (76) by the factor:

$$\prod_{v \in X_{-C} - \{\rho_{\bar{E}}\}} \prod_{c \in C} \prod_{i \in A_v} \frac{\Gamma(N'^*_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \tag{77}$$

and rearranging we get:

$$\begin{aligned} P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \mathcal{D}, \xi) = {} & P(\bar{E} \mid E) \frac{1}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \frac{\Gamma(N')}{\Gamma(N'^*)} \prod_{u,v \in \bar{E}} W_{u,v} \beta_{u,v} \\ & \times \prod_{c \in C} \prod_{v \in X_{-C}} \prod_{i \in A_v} \frac{\Gamma(N'^*_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \\ & \times D(\theta_C(.); N'^*_C(.)) \\ & \times \prod_{c \in C} D(\theta_{\rho_{\bar{E}}|C}(.,c); N'^*_{\rho_{\bar{E}},C}(.,c)) \\ & \times \prod_{c \in C} \prod_{u,v \in \bar{E}} \prod_{i \in A_u} D(\theta_{v|u,C}(.,i,c); N'^*_{v,u,C}(.,i,c)). \end{aligned} \tag{78}$$

In order to have $P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \mathcal{D}, \xi)$ completely determined we need to calculate $Z_{\mathcal{D}}$. Since we know that

$$\int_{B \in \mathcal{B}} P(B \mid \mathcal{D}, \xi) = \sum_{E \in \mathcal{E}} \left(\sum_{\bar{E} \in Or(E)} \int \cdots \int_{\Theta_{\bar{E}^*}} P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \mathcal{D}, \xi) \right) = 1. \tag{79}$$

We can do this by integrating over the parameters, then summing over the tree structures and finally solving for $Z_{\mathcal{D}}$. The first step is easy, because Dirichlet distributions are normalized and integrate to 1 giving:

$$\begin{aligned} \int \cdots \int_{\Theta_{\bar{E}^*}} P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \mathcal{D}, \xi) = {} & P(\bar{E} \mid E) \frac{1}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \frac{\Gamma(N')}{\Gamma(N'^*)} \prod_{u,v \in \bar{E}} W_{u,v} \beta_{u,v} \\ & \times \prod_{c \in C} \prod_{v \in X_{-C}} \prod_{i \in A_v} \frac{\Gamma(N'^*_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))}. \end{aligned} \tag{80}$$

Since $W_{u,v}\beta_{u,v} = W_{v,u}\beta_{v,u}$, we can add out $P(\bar{E} \mid E)$ as we did before (see Eq. (69) in Section B.1.2) and the addition over undirected structures can be calculated by means of the matrix tree theorem for decomposable priors, giving

$$\begin{aligned} \sum_{E \in \mathcal{E}} \int \cdots \int_{\Theta_{\bar{E}^*}} P(\bar{E}^*, \Theta_{\bar{E}^*} \mid \mathcal{D}, \xi) = {} & \frac{\mid Q(\boldsymbol{\beta}\mathbf{W}) \mid}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \frac{\Gamma(N')}{\Gamma(N'^*)} \\ & \times \prod_{c \in C} \prod_{v \in X_{-C}} \prod_{i \in A_v} \frac{\Gamma(N'^*_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} = 1. \end{aligned} \tag{81}$$

Solving for $Z_{\mathcal{D}}$, recalling that $Z_\beta = \mid Q(\boldsymbol{\beta}) \mid$ we have that

$$Z_{\mathcal{D}} = \frac{\mid Q(\boldsymbol{\beta}\mathbf{W}) \mid}{\mid Q(\boldsymbol{\beta}) \mid} \frac{\Gamma(N')}{\Gamma(N'^{*})} \prod_{c \in C} \prod_{v \in X_{-C}} \prod_{i \in A_v} \frac{\Gamma(N'^{*}_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))}. \tag{82}$$

Finally, substituting the result for $Z_{\mathcal{D}}$ in Eq. (78) we can see that the posterior is a decomposable distribution with the parameters updated as given by Eqs. (20)–(22):

$$\begin{aligned} P(\bar{E}^{*}, \Theta_{\bar{E}^{*}} \mid \mathcal{D}, \xi) = \frac{1}{\mid Q(\boldsymbol{\beta}\mathbf{W}) \mid} & \prod_{u,v \in \bar{E}} W_{u,v} \beta_{u,v} D(\theta_C(.); N'^{*}_C(.)) \\ & \times \prod_{c \in C} D(\theta_{\rho_{\bar{E}}|C}(.,c); N'^{*}_{\rho_{\bar{E}},C}(.,c)) \\ & \times \prod_{c \in C} \prod_{u,v \in \bar{E}} \prod_{i \in A_u} D(\theta_{v|u,C}(.,i,c); N'^{*}_{v,u,C}(.,i,c)). \end{aligned} \tag{83}$$

Acknowledgments

We would like to acknowledge the Dynamical Systems Group of the department of Matemàtica Aplicada i Anàlisi, Universitat de Barcelona for allowing the use of the cluster to run the experiments here reported. We would also like to thank the anonymous reviewers for an excellent and helpful work.

References

Cerquides, J. (1999). Applying general Bayesian techniques to improve TAN induction. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining, KDD99* (pp. 292–296).

Chow, C., & Liu, C. (1968). Aproximatting discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory, 14*, 462–497.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning, 29*, 103–130.

Fawcett, T. (2003). Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Laboratories Palo Alto.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29*, 131–163.

Hand, D. & Till, R. (2001). A simple generalization of the area under the roc curve to multiple class classification problems. *Machine Learning, 45:2*, 171–86.

Heckerman, D., Geiger, D., & Chickering, D. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*, 197–243.

Ide, J., & Cozman, F. (2003). Generation of random bayesian networks with constraints on induced width, with applications to the average analysis od d-connectivity, quasi-random sampling, and loopy propagation. Technical report, University of Sao Paulo.

Katoh, N., Ibaraki, T., & Mine, H. (1981). An algorithm for finding k minimum spanning trees. *SIAM J. Comput., 10:2*, 247–55.

Kontkanen, P., Myllymaki, P., Silander, T., & Tirri H. (1998). Bayes Optimal Instance-Based Learning. In *Machine Learning: ECML-98, Proceedings of the 10th European Conference*, volume 1398 of *Lecture Notes in Artificial Intelligence* (pp. 77–88). Springer-Verlag.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223–228). AAAI Press and MIT Press.

Meila, M., & Jaakkola, T. (2000a). Tractable bayesian learning of tree belief networks. In *Proc. of the Sixteenth Conference on Uncertainty in Artificial Intelligence* (pp. 380–388).

Meila, M., & Jaakkola, T. (2000b). Tractable bayesian learning of tree belief networks. Technical Report CMU-RI-TR-00-15, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Pettie, S., & Ramachandran, V. (2002). An optimal minimum spanning tree algorithm. *Journal of the ACM (JACM), 49:1*, 16–34.

Shoup, V. (2003). NTL: A library for doing number theory. http://www.shoup.net/ntl.

Received March 1, 2004
Revised November 26, 2004
Accepted November 26, 2004