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
T: +34 667756217
F: +34 934021601

Ramon López de Màntaras (mantaras@iiia.csic.es)
Institut d'Investigació en Intel.ligència Artificial - CSIC
Campus UAB
08190 Bellaterra, Spain

Abstract. In this paper we present several Bayesian algorithms for learning Tree Augmented Naive Bayes (TAN) models. First we correct Meila and Jaakkola (Meila and Jaakkola, 2000a) results for Bayesian learning with tree belief networks. Then we show that these results can be extended 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 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 et al., 1992) or *Tree Augmented Naive Bayes* (TAN) (Friedman et al., 1997) have shown excellent performance in spite of their simplicity and heavy underlying independence assumptions.

In their seminal work (Chow and 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 et al., 1997) by extending Chow and Liu results from tree structures to TAN structures. In (Meila and Jaakkola, 2000a), Meila and Jaakkola 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 reviewing previous work in TAN classifiers and presenting the notation to be used.

In section 3, we define decomposable distributions over trees as in (Meila and Jaakkola, 2000a). We show that some results in (Meila and Jaakkola, 2000a) are not correct and we provide the corrected results with their detailed proofs. Then we define decomposable distributions over TAN models and we prove that they have four relevant properties. First, that they allow the computation of the exact Bayesian model averaging over TAN structures and parameters in polynomial time. Second, that they allow the computation of the single MAP TAN structure in polynomial time. Third, that they allow the computation of the k *maximum a posteriori* (MAP) TAN structures and their relative probability weights in polynomial time and that the increase in complexity is small with respect to computing the single MAP TAN structure. Finally we show that, given a fixed TAN structure, the result of performing exact Bayesian model averaging over parameters can be represented as a single TAN model.

Once we have presented and proved these properties , we use them to construct three classifiers. We construct the MAPTAN classifier by computing the MAP TAN structure and then performing Bayesian model averaging (BMA) over parameters. We construct MAPTAN+BMA by computing the k MAP TAN structures (and their relative probability weigths) and constructing an ensemble classifier based on that. Finally, we construct TBMATAN (Tractable Bayesian Model Averaging TAN) by computing the exact Bayesian model averaging over TAN structures and parameters. Furthermore, we argue that special care should be taken when implementing TBMATAN, because the algorithm requires the computation of the determinat of a matrix that frequently is ill-conditioned. In order to deal with this problem we introduce a fourth classifier, SSTBMATAN (Structure Sttuborn TBMATAN), as an approximation to TBMATAN.

In section 6 we compare the empirical results obtained by the four classifiers introduced, with the softened TAN classifier proposed in (Friedman et al., 1997) which we name STAN. The comparison is made 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.

This paper improves and extends significatively the results presented in (Cerquides and López de Màntaras, 2003). Concretely, the paper provides a more detailed background, corrected results for decomposable distributions over trees, detailed proofs and introduces and discusses two new classifiers (MAPTAN and MAPTAN+BMA). Also the experimental work has extended by using the area under the ROC curve as measure of performance and performing experimental work over randomly generated Bayesian networks. This allows the experimental results to be far more conclusive than the ones presented in (Cerquides and López de Màntaras, 2003).

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 et al., 1992; Domingos and Pazzani, 1997). TAN models are a restricted family of Bayesian networks in which the class variable has no parents and each attribute has as parents the class variable and at most one other attribute. An example of TAN model can be seen in Figure 2(c).

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 et al., 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 et al., 1995; Friedman et al., 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 $\Omega = \{X_1, \ldots, X_m\}$ for a discrete domain, where $X_1, \ldots, X_m$ 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 $\Omega_C = \{A_1, \ldots, A_n, C\}$ for a classified discrete domain. In the rest of the paper we will refer to an attribute either as X_i (when it is considered part of a discrete

domain), A_i (when it is considered part of a classified discrete domain and it is not the class) and C (when it is the class of a classified discrete domain). We note $V = \{A_1, \ldots, A_n\}$ 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 $\#\Omega = \prod_{i=1}^{m} \#X_i$ and $\#\Omega_C = \#C \prod_{i=1}^{n} \#A_i$. We note the maximum number of different values for an attribute in a domain $r = \max_{i \in \Omega} \#X_i = \max(\max_{i \in V} \#A_i, \#C)$.

An *observation* x in a classified discrete domain Ω_C is an ordered tuple $x = (x_1, \ldots, x_n, x_C) \in A_1 \times \ldots \times A_n \times C$. An *unclassified observation* S in Ω_C is an ordered tuple $S = (s_1, \ldots, s_n) \in A_1 \times \ldots \times A_n$. To be homogeneous we will abuse this notation a bit noting s_C for a possible value of the class for S. 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 $N_i(x_i)$ for the number of observations in $\mathcal{D}$ where the value for A_i is x_i, $N_{i,j}(x_i, x_j)$ the number of observations in $\mathcal{D}$ where the value for A_i is x_i and the value for A_j is x_j and similarly for $N_{i,j,k}(x_i, x_j, x_k)$ and so on. We note similarly $f_i(x_i), f_{i,j}(x_i, x_j), \ldots$ the frequencies in $\mathcal{D}$. It is worth noticing that f defines a probability distribution over $A_1 \times \ldots \times A_n \times C$.

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 S in Ω_C assigns a class to S.

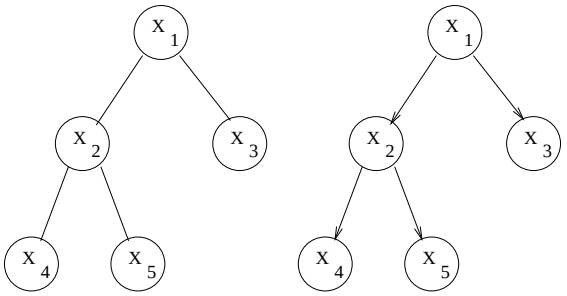
2.1.2. *Bayesian Networks for Discrete Classification*

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 $\mathbf{U} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ where each $\mathcal{X}_i$ is a random variable over its corresponding attribute X_i. We extend the meaning of this notation to $\mathcal{A}_i$, $\mathcal{C}$ and $\mathcal{V}$. 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 $\mathcal{X}_1, \ldots, \mathcal{X}_m$ and whose edges represent direct dependencies between the variables. The graph G encodes independence assumptions: each variable $\mathcal{X}_i$ is independent of its nondescendants 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 $\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = P_B(x_i|\Pi_{x_i})$ for each $x_i \in X_i$ and $\Pi_{x_i} \in \Pi_{X_i}$, where Π_{X_i} denotes the Cartesian product of every X_j such that $\mathcal{X}_j$ is a parent of $\mathcal{X}_i$ in G. Π_i is the list of parents of $\mathcal{X}_i$ in G. We will note $\overline{\Pi_i} = \mathbf{U} - \{\mathcal{X}_i\} - \Pi_i$. A Bayesian network defines a unique

joint probability distribution over $\mathbf{U}$ given by

$$P_B(x_1, \ldots, x_m) = \prod_{i=1}^{m} P_B(x_i|\Pi_{x_i}) = \prod_{i=1}^{m} \theta_{i|\Pi_i}(x_i|\Pi_{x_i}) \qquad (1)$$

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 $\mathbf{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{C}\}$ where each $\mathcal{A}_i$ (resp. $\mathcal{C}$) is a random variable over A_i (resp. C). Then if we are given a new unclassified observation S we can easily classify S into class $\underset{s_C \in C}{argmax}(P_B(s_1, \ldots, s_n, s_C))$. This simple mechanism allows us to see any Bayesian network learning algorithm as a classifier.



(a) E (b) $\overline{E}$

Figure 1. Notation for learning with trees

2.2. Learning with trees

Given an unclassified domain Ω we will note $\mathcal{E}$ the set of undirected graphs E over $\{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ such that E is a tree (has no cycles). We will use $u, v \in E$ instead of $(\mathcal{X}_u, \mathcal{X}_v) \in E$ for simplicity. We note $\overline{E}$ a directed tree for E. We note $\rho_{\overline{E}}$ the root of a directed tree $\overline{E}$ (i.e. in Figure 1(b) we have that $\rho_{\overline{E}} = X_1$).

We will note $\Theta_{\overline{E}}$ the set of parameters that quantify the Bayesian network $M = \langle \overline{E}, \Theta_{\overline{E}} \rangle$. More concretely:

$\Theta_{\overline{E}} = (\boldsymbol{\theta}_{\rho_{\overline{E}}}, \{\boldsymbol{\theta}_{v|u} | u, v \in \overline{E}\})$

$\boldsymbol{\theta}_{\rho_{\overline{E}}} = \{\theta_{\rho_{\overline{E}}}(i) | i \in X_{\rho_{\overline{E}}}\}$ where $\theta_{\rho_{\overline{E}}}(i) = P(\mathcal{X}_{\rho_{\overline{E}}} = i|M)$

For each $u, v \in \overline{E}$:
$\boldsymbol{\theta}_{v|u} = \{\theta_{v|u}(j, i) | j \in X_v, i \in X_u\}$ where
$\theta_{v|u}(j, i) = P(\mathcal{X}_v = j | \mathcal{X}_u = i, M)$.

```
procedure Construct-Tree (ProbabilityDistribution P)
    var
       WeightMatrix I_P;
       UndirectedGraph UG;
       UndirectedTree UT;
       DirectedTree T;
    foreach X_i, X_j
        Compute I_P(X_i, X_j) = sum_{x in X_i, y in X_j} P(x,y) log(P(x,y) / (P(x)P(y)))
    end
    UG = ConstructUndirectedGraph(I_P);
    UT = MaximumWeightedSpanningTree(UG);
    T = MakeDirected(UT);
    Fix T parameters using equation 3
    return T;
```

Algorithm 1: Maximum likelihood tree construction procedure

2.2.1. *Learning Maximum Likelihood trees*
One of the measures used to learn Bayesian networks is the *log likelihood*:

$$LL(B|\mathcal{D}) = \sum_{x \in \mathcal{D}} \log(P_B(x)) \tag{2}$$

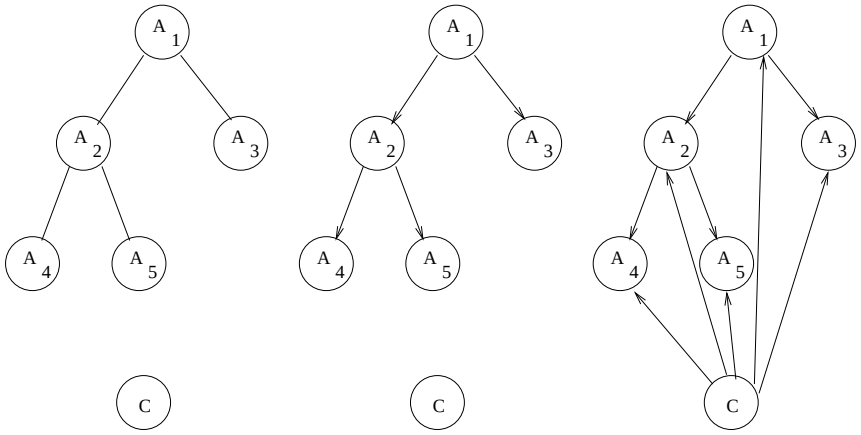
An interesting property of tree probability distributions is that we have an efficient procedure (Chow and Liu, 1968) for identifying the structure of the tree which maximizes likelihood. The procedure, that can be seen in algorithm 1, fixes the value of the tree parameters as follows:

$$\begin{aligned} \theta_{\rho_{\overline{E}}}(i) &= P(\mathcal{X}_{\rho_{\overline{E}}} = i) \\ \theta_{v|u}(j,i) &= \frac{P(\mathcal{X}_v = j, \mathcal{X}_u = i)}{P(\mathcal{X}_{\rho_{\overline{E}}} = i)} \end{aligned} \tag{3}$$

where P is the probability distribution that it receives as parameter.

The theorem that guarantees that algorithm 1 finds the maximum likelihood tree is stated below, where f is the probability distribution induced by the frequencies in $\mathcal{D}$ and r,n and N are defined in section 2.1.1.

THEOREM 1 (Chow & Liu, 1968). *Let $\mathcal{D}$ be a dataset over Ω. The procedure* `Construct-Tree(`f`)` *builds a tree B_T that maximizes $LL(B_T|\mathcal{D})$ and has time complexity $O((N + r^2) \cdot n^2)$.*



(a) E (b) $\overline{E}$ (c) $\overline{E}^*$

Figure 2. Notation for learning with TANs

2.3. 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 exclude the class attribute from $\mathcal{E}$). We will use $u, v \in E$ instead of $(\mathcal{A}_u, \mathcal{A}_v) \in E$ for simplicity. We note $\overline{E}$ a directed tree for E. Every $\overline{E}$ uniquely determines the structure of a Tree Augmented Naive Bayes classifier, because from $\overline{E}$ we can construct $\overline{E}^* = \overline{E} \cup \{(\mathcal{C}, \mathcal{A}_i) | 1 \leq i \leq n\}$ as can be seen in an example in Figure 2. We note the root of a directed tree $\overline{E}$ as $\rho_{\overline{E}}$ (i.e. in Figure 2(b) we have that $\rho_{\overline{E}} = A_1$).

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

$$\Theta_{\overline{E}^*} = (\boldsymbol{\theta}_C, \boldsymbol{\theta}_{\rho_{\overline{E}}|C}, \{\boldsymbol{\theta}_{v|u,C} | u, v \in \overline{E}\})$$

$$\boldsymbol{\theta}_C = \{\theta_C(c) | c \in C\} \text{ where } \theta_C(c) = P(\mathcal{C} = c | M)$$

$$\boldsymbol{\theta}_{\rho_{\overline{E}}|C} = \{\theta_{\rho_{\overline{E}}|C}(i, c) | i \in A_{\rho_{\overline{E}}}, c \in C\} \text{ where}$$
$$\theta_{\rho_{\overline{E}}|C}(i, c) = P(\mathcal{A}_{\rho_{\overline{E}}} = i | \mathcal{C} = c, M)$$

For each $u, v \in \overline{E}$:
$$\boldsymbol{\theta}_{v|u,C} = \{\theta_{v|u,C}(j, i, c) | j \in A_v, i \in A_u, c \in C\} \text{ where}$$
$$\theta_{v|u,C}(j, i, c) = P(\mathcal{A}_v = j | \mathcal{A}_u = i, \mathcal{C} = c, M).$$

```
procedure Construct-TAN (ProbabilityDistribution P)
    var
       WeightMatrix I_P;
       UndirectedGraph UG;
       UndirectedTree UT;
       DirectedTree T;
       DirectedGraph TAN;
    foreach A_i, A_j
        Compute I_P(A_i; A_j|C) = sum_{x∈A_i, y∈A_j, z∈C} P(x,y,z) log( P(x,y|z) / (P(x|z)P(y|z)) )
    end
    UG = ConstructUndirectedGraph(I_P);
    UT = MaximumWeightedSpanningTree(UG);
    T = MakeDirected(UT);
    TAN = AddClass(T);
    Fix TAN parameters using equation 4
    return TAN;
```

Algorithm 2: Maximum likelihood TAN construction procedure

2.3.1. *Learning Maximum Likelihood TAN*
The results for trees reviewed in section 2.2.1 were extended to TANs in (Friedman et al., 1997). We call here the procedure and theorem for learning maximum likelihood TANs.

THEOREM 2 (Friedman, Geiger & Goldszmidt, 1997). *Let $\mathcal{D}$ be a dataset over Ω_C. The procedure* `Construct-TAN(f)` *builds a TAN B_T that maximizes $LL(B_T|D)$ and has time complexity $O((N + r^3) \cdot n^2)$.*

Algorithm 2 fixes the model parameters by using:

$$\begin{aligned} \theta_C(c) &= P(\mathcal{C} = c) \\ \theta_{\rho_{\overline{E}}|C}(i,c) &= P(\mathcal{A}_{\rho_{\overline{E}}} = i|\mathcal{C} = c) \\ \theta_{v|u,C}(j,i,c) &= P(\mathcal{A}_v = j|\mathcal{A}_u = i, \mathcal{C} = c) \end{aligned} \tag{4}$$

where P is the probability distribution that it receives as parameter. It has been shown (Friedman et al., 1997) that equation 4 leads to models that overfit the data. As an alternative, Friedman et al. propose to set

the parameters as follows:

$$\begin{aligned}
\theta_C(c) &= P(\mathcal{C} = c) \\
\theta_{\rho_{\overline{E}}|C}(i,c) &= \frac{N \cdot P(\mathcal{A}_{\rho_{\overline{E}}} = i, \mathcal{C} = c) + N^0 \cdot P(\mathcal{A}_{\rho_{\overline{E}}} = i)}{N \cdot P(\mathcal{C} = c) + N^0} \\
\theta_{v|u,C}(j,i,c) &= \frac{N \cdot P(\mathcal{A}_v = j, \mathcal{A}_u = i, \mathcal{C} = c) + N^0 \cdot P(\mathcal{A}_v = j)}{N \cdot P(\mathcal{A}_u = i, \mathcal{C} = c) + N^0}
\end{aligned} \tag{5}$$

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 equation 5 will be referred to in the rest of the paper as STAN. Using equation 5 to fix the parameters improves the accuracy of the classifier. However, they are making some further assumptions (they are assuming that parameters follow the Dirichlet distribution) in order to derive equation 5. The question arises on whether those assumptions (or similar ones) could be used in the determination of the TAN structure. In section 4.5 we will see how this can be done, but first we have to develop the results for the simpler case of trees.

3. Decomposable distributions over tree belief networks

In this section we review and correct results for Bayesian learning of tree belief networks. Following (Meila and Jaakkola, 2000a), we define decomposable distributions over trees in section 3.1 and after that we prove that the exact Bayesian model averaging can be computed in polynomial time.

3.1. Definition

In (Meila and Jaakkola, 2000b), Meila and Jaakkola 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. In the following we will use the term decomposable distribution over trees instead of decomposable priors, and we will use prior and posterior as they are commonly used in Bayesian analysis.

Decomposable distributions over trees are the product of a distribution over tree structures and a distribution over tree parameters, that is, assuming that, given ξ, the probability distribution over the set of tree belief network models follows a decomposable distribution over trees, we have that

$$P(M|\xi) = P(\overline{E}, \Theta_{\overline{E}}|\xi) = P(\overline{E}|\xi)P(\Theta_{\overline{E}}|\overline{E}, \xi) \tag{6}$$

where we recall that $\overline{E}$ is the directed tree structure and $\Theta_{\overline{E}}$ its parameters. The definition of decomposable distribution will be done by specifying its two components, $P(\overline{E}|\xi)$, the decomposable distribution over tree structures and $P(\Theta_{\overline{E}}|\overline{E},\xi)$, the decomposable distribution over tree parameters.

3.1.1. *Decomposable distributions over tree structures*
Recalling that m is the number of variables when talking about an unclassified discrete domain, a decomposable distribution over tree structures is determined by a $m \times m$ hyperparameter matrix $\boldsymbol{\beta} = (\beta_{u,v})$ such that $\forall u, v : 1 \leq u, v \leq m$ 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 $(\mathcal{X}_u, \mathcal{X}_v)$ is contained in the tree model underlying the data.

We say that $P(\overline{E}|\xi)$ follows a decomposable distribution over tree structures with hyperparameter set $\boldsymbol{\beta}$ iff:

$$P(\overline{E}|\xi) = \frac{P(E|\xi)}{m} \tag{7}$$

being

$$P(E|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{8}$$

where Z_β is a normalization constant with value:

$$Z_\beta = \sum_{E \in \mathcal{E}} \prod_{u,v \in E} \beta_{u,v} \tag{9}$$

It is worth noting that $P(\overline{E}|\xi)$ depends only on the underlying undirected tree structure E.

3.1.2. *Decomposable distributions over tree parameters*
A decomposable distribution over tree parameters follows the equation:

$$P(\Theta_{\overline{E}}|\overline{E},\xi) = P(\boldsymbol{\theta}_{\rho_{\overline{E}}}|\overline{E},\xi) \prod_{u,v \in \overline{E}} P(\boldsymbol{\theta}_{v|u}|\overline{E},\xi) \tag{10}$$

where we recall that $\rho_{\overline{E}}$ is the root of $\overline{E}$. Furthermore, a decomposable distribution over tree parameters has a hyperparameter set $\mathbf{N}' = \{N'_{v,u}(j,i) | 1 \leq u \neq v \leq m \ ; \ j \in X_v \ ; \ i \in X_u\}$ with the constraints that $N'_{v,u}(j,i) > 0$ and that there exist $N'_u(i)$, N' such that for every u,v:

$$N'_u(i) = \sum_{j \in X_v} N'_{v,u}(j,i) \tag{11}$$

$$N' = \sum_{i \in X_u} N'_u(i) \tag{12}$$

We say that $P(\Theta_{\overline{E}}|\overline{E}, \xi)$ follows a decomposable distribution over tree parameters with hyperparameter set $\mathbf{N}'$ iff

1. $P(\Theta_{\overline{E}}|\overline{E}, \xi)$ fulfills equation 10
2. $\mathbf{N}'$ fulfills the conditions appearing in equations 11 and 12.
3. the following two equations are also satisfied

$$P(\boldsymbol{\theta}_{\rho_{\overline{E}}}|\overline{E}, \xi) = D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \tag{13}$$

$$P(\boldsymbol{\theta}_{v|u}|\overline{E}, \xi) = \prod_{i \in X_u} D(\theta_{v|u}(., i); N'_{v,u}(., i)) \tag{14}$$

where D stands for the Dirichlet distribution.

3.1.3. *Decomposable distributions over tree structures and parameters*
We say that $P(M|\xi)$ follows a decomposable distribution over trees with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$ iff

1. $P(M|\xi)$ fulfills equation 6
2. $P(\overline{E}|\xi)$ follows a decomposable distribution over tree structures with hyperparameter set $\boldsymbol{\beta}$
3. $P(\Theta_{\overline{E}}|\overline{E}, \xi)$ follows a decomposable distribution over tree parameters with hyperparameter set $\mathbf{N}'$

that is, if the conditions in equations 6, 7, 8, 9, 10, 11, 12, 13 and 14 hold.

3.2. Meila and Jaakkola results and corrections

In (Meila and Jaakkola, 2000b), 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.

3.2.1. *Assumptions needed for decomposable distributions over tree parameters*
Meila and Jaakkola demonstrated that if we have a probability distribution over tree belief networks satisfying equation 6, for which the support graph is connected and its parameter set is strictly positive,

then under the assumptions of likelihood equivalence, parameter independence, parameter modularity and connectivity, for any tree in any directed representation, the parameters are distributed following a set of Dirichlets as imposed by equations 10, 11, 12, 13 and 14. This means that decomposable distributions over tree parameters are the result of a fairly reasonable set of assumptions widely used for learning Bayesian networks as can be seen in (Heckerman et al., 1995).

3.2.2. *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

$$\beta^*_{u,v} = \beta_{u,v} W_{u,v} \tag{15}$$

$$N'^*_{v,u}(j,i) = N'_{v,u}(j,i) + N_{v,u}(j,i) \tag{16}$$

where

$$W_{u,v} = \prod_{i \in X_u} \prod_{j \in X_v} \frac{\Gamma(N'_{v,u}(j,i) + N_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))} \tag{17}$$

3.2.3. *Corrected Bayesian learning with decomposable distributions over trees*
Unfortunately, the last result is incorrect. After careful derivation, it can be proven that if we assume a decomposable prior 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 equations 15 and 16 but $W_{u,v}$ are given by:

$$\begin{aligned} W_{u,v} = & \prod_{i \in X_u} \frac{\Gamma(N'_u(i))}{\Gamma(N'_u(i) + N_u(i))} \\ & \times \prod_{j \in X_v} \frac{\Gamma(N'_v(j))}{\Gamma(N'_v(j) + N_v(j))} \\ & \times \prod_{i \in X_u} \prod_{j \in X_v} \frac{\Gamma(N'_{v,u}(j,i) + N_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))} \end{aligned} \tag{18}$$

The demonstration can be seen in appendix B.2.

3.2.4. *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 point conditioned to the observation of a dataset $\mathcal{D}$ is given by

$$P(\mathcal{X} = x|\mathcal{D}, \xi) = \frac{w_0(x)|Q(\boldsymbol{\beta}\ \mathbf{w}(x))|}{|Q(\boldsymbol{\beta}\mathbf{W})|} \tag{19}$$

where

$$w_0(x) = \frac{1}{N' + N} \prod_{X_u \in V} [N'_{pa(u)}(x_{pa(u)}) + N_{pa(u)}(x_{pa(u)})] \tag{20}$$

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

$$\mathbf{w}(x) = (w_{u,v}(x)) \text{ where } w_{u,v}(x) = \frac{N'_{v,u}(s_v, s_u) + N_{v,u}(s_v, s_u)}{(N'_u(s_u) + N_u(s_u))(N'_v(s_v) + N_v(s_v))} \tag{21}$$

and for any real $m \times m$ matrix $\boldsymbol{\tau}$ we define $Q(\boldsymbol{\tau}) : \mathbb{R}^{m \times m} \to \mathbb{R}^{m-1 \times m-1}$ as the first $m - 1$ lines and columns of the matrix $\overline{Q}(\boldsymbol{\tau})$ where

$$\overline{Q}_{u,v}(\boldsymbol{\tau}) = \overline{Q}_{v,u}(\boldsymbol{\tau}) = \begin{cases} -\tau_{u,v} & 1 \leq u < v \leq m \\ \sum_{v'=1}^{m} \tau_{v',v} & 1 \leq u = v \leq m \end{cases} \tag{22}$$

3.2.5. *Corrected computation of probabilities from a decomposable distribution over trees*

The previous result is also incorrect since they assume $\mathbf{W}$ defined as in equation 17. 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 $X_1 \to X_2$ we have that $w(x) = N'_1(x_1) + N_1(x_1)$ while for the tree $X_2 \to X_1$ we have that $w(x) = N'_2(x_2) + N_2(x_2)$. In fact, 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 data point given a decomposable distribution over trees, since such a result can be applied to any decomposable distribution over trees, including the posterior. Hence, we state the corrected result as follows:

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

$$P(\mathcal{X} = x|\xi) = h_0^x |Q(\boldsymbol{\beta}\ \mathbf{h}^x)| \tag{23}$$

where

$$h_0^x = \frac{1}{Z_\beta} \frac{1}{N'} \prod_{X_u \in V} N'_u(x_u) \tag{24}$$

$$\mathbf{h}^x = \left(h^x_{u,v}\right) \text{ where } h^x_{u,v} = \frac{N'_{v,u}(x_v, x_u)}{N'_u(x_u)N'_v(x_v)} \tag{25}$$

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

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 3.2.3.

4. Decomposable distributions over TANs

In the previous section we have reviewed and corrected the results in (Meila and Jaakkola, 2000b) for trees. In this section we will extend them to TAN models. We start by introducing decomposable distributions over TAN structures and parameters, built upon the already presented idea of decomposable priors over trees. After that we demonstrate that, given a decomposable distribution over TANs, it is possible to compute the probability of an unseen observation and that, given a prior decomposable distribution over TANs, the posterior distribution after observing a dataset is also a decomposable distribution over TANs.

4.1. Definition

In this section we introduce decomposable distributions over TANs, which are a family of probability distributions in the space $\mathcal{M}$ of TAN models.

Decomposable distributions over TANs are constructed in two steps. In the first step, a distribution over the set of different undirected tree structures is defined. Every directed tree structure is defined to have the same probability as its undirected equivalent. In the second step, a distribution over the set of parameters is defined. If $P(M|\xi)$ follows a decomposable distribution over TANs then the probability for a model $M = \langle \overline{E}^*, \Theta_{\overline{E}^*} \rangle$ (a TAN with fixed tree structure $\overline{E}^*$ and fixed parameters $\Theta_{\overline{E}^*}$) is determined by:

$$P(M|\xi) = P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) = P(\overline{E}^*|\xi)P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) \tag{26}$$

In the following sections we specify the value of a decomposable distribution over the two components of a TAN model, namely its structure and its parameters. That is, $P(\overline{E}^*|\xi)$ (decomposable distribution over TAN structures) and $P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)$ (decomposable distribution over TAN parameters).

4.1.1. *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 $P(\overline{E}^*|\xi)$ follows a decomposable distribution over TAN structures with hyperparameter set $\boldsymbol{\beta}$ iff:

$$P(\overline{E}^*|\xi) = \frac{P(E|\xi)}{n} \tag{27}$$

$$P(E|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{28}$$

where Z_β is a normalization constant with value:

$$Z_\beta = \sum_{E \in \mathcal{E}} \prod_{u,v \in E} \beta_{u,v} \tag{29}$$

It is worth noticing that $P(\overline{E}^*|\xi)$ depends only on the underlying undirected tree structure E.

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

$$P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) = P(\boldsymbol{\theta}_C|\overline{E}^*, \xi)\ P(\boldsymbol{\theta}_{\rho_{\overline{E}}|C}|\overline{E}^*, \xi) \prod_{u,v \in \overline{E}} P(\boldsymbol{\theta}_{v|u,C}|\overline{E}^*, \xi) \tag{30}$$

Furthermore, a decomposable distribution over TAN parameters has a hyperparameter set $\mathbf{N}' = \{N'_{v,u,C}(j,i,c) | 1 \leq u \neq v \leq n\ ;\ j \in A_v\ ;\ i \in A_u\ ;\ c \in C\}$ 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:

$$N'_{u,C}(i,c) = \sum_{j \in A_v} N'_{v,u,C}(j,i,c) \tag{31}$$

$$N'_C(c) = \sum_{i \in A_u} N'_{u,C}(i,c) \tag{32}$$

$$N' = \sum_{c \in C} N'_C(c) \tag{33}$$

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

1. $P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)$ fulfills equation 30
2. $\mathbf{N}'$ fulfills the conditions appearing in equations 31, 32 and 33
3. the following equations are also satisfied:

$$P(\boldsymbol{\theta}_C|\overline{E}^*, \xi) = D(\theta_C(.); N'_C(.)) \tag{34}$$

$$P(\boldsymbol{\theta}_{\rho_{\overline{E}}|C}|\overline{E}^*, \xi) = \prod_{c \in C} D(\theta_{\rho_{\overline{E}}|C}(., c); N'_{\rho_{\overline{E}},C}(., c)) \tag{35}$$

$$P(\boldsymbol{\theta}_{v|u,C}|\overline{E}^*, \xi) = \prod_{c \in C} \prod_{i \in A_u} D(\theta_{v|u,C}(., i, c); N'_{v,u,C}(., i, c)) \tag{36}$$

4.1.3. *Decomposable distributions over TAN structures and parameters*

We say that $P(M|\xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$ iff

1. $P(M|\xi)$ fulfills equation 26
2. $P(\overline{E}^*|\xi)$ follows a decomposable distribution over TAN structures with hyperparameter set $\boldsymbol{\beta}$
3. $P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)$ follows a decomposable distribution over TAN parameters with hyperparameter set $\mathbf{N}'$

that is if the conditions in equations 26, 27, 28, 29, 30, 31, 32, 33, 34, 35 and 36 hold.

4.2. Calculating probabilities under decomposable distributions over TANs

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

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \int_{M \in \mathcal{M}} P(\mathcal{V} = S, \mathcal{C} = s_C|M) P(M|\xi) \tag{37}$$

The integral for $P(\mathcal{V} = S, \mathcal{C} = s_C|\xi)$ can be calculated in closed form by applying the matrix tree theorem (see Appendix A.1 and (Meila and Jaakkola, 2000b)) and the result can be expressed in terms of the previously introduced Q as:

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = h_0^{S,s_C} |Q(\boldsymbol{\beta}\ \mathbf{h}^{S,s_C})| \tag{38}$$

where

$$h_0^{S,s_C} = \frac{1}{Z_\beta}\frac{1}{N'}\prod_{A_u \in V} N'_{u,C}(s_u, s_C) \tag{39}$$

$$\mathbf{h}^{S,s_C} = \left(h_{u,v}^{S,s_C}\right) \text{ where } h_{u,v}^{S,s_C} = \frac{N'_{v,u,C}(s_v, s_u, s_C)}{N'_{u,C}(s_u, s_C)N'_{v,C}(s_v, s_C)} \tag{40}$$

The proof for this result appears in appendix C.1.

4.3. Learning under decomposable distributions over TANs

Assume that the data is generated by a TAN model and that $P(M|\xi)$ follows a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}$, $\mathbf{N}'$. Then, $P(M|\mathcal{D}, \xi)$, 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:

$$\beta^*_{u,v} = \beta_{u,v} W_{u,v} \tag{41}$$

$$N'^*_{u,v,C}(j,i,c) = N'_{u,v,C}(j,i,c) + N_{u,v,C}(j,i,c) \tag{42}$$

where

$$\begin{aligned} W_{u,v} = & \prod_{c\in C}\prod_{i\in A_u} \frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'_{u,C}(i,c) + N_{u,C}(i,c))} \\ & \times \prod_{c\in C}\prod_{j\in A_v} \frac{\Gamma(N'_{v,C}(j,c))}{\Gamma(N'_{v,C}(j,c) + N_{v,C}(j,c))} \\ & \times \prod_{c\in C}\prod_{i\in A_u}\prod_{j\in A_v} \frac{\Gamma(N'_{v,u,C}(j,i,c) + N_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))} \end{aligned} \tag{43}$$

The proof appears in appendix C.2.

4.4. Classifying with Decomposable Distributions over TANs given an undirected structure

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

$$P(\mathcal{C} = s_C|\mathcal{V} = S, E, \xi) \propto h_0^{S,s_C} \prod_{u,v\in E} h_{u,v}^{S,s_C} \tag{44}$$

where h_0^{S,s_C} and $h_{u,v}^{S,s_C}$ are defined as in equations 39 and 40. The proof for this result follows from the result demonstrated in appendix C.1.1

In fact, given an undirected TAN structure E, it is easy to see that the probability distribution $P(\mathcal{C} = s_C|\mathcal{V} = S, E, \xi)$ can be represented as a TAN model with structure $\overline{E}^*$, such that its undirected version coincides with E and its parameter set is given by

$$\begin{aligned} \theta_{u|v,C}(s_u, s_v, s_C) &= \frac{N'_{u,v,C}(s_u,s_v,s_C)}{N'_{v,C}(s_v,s_C)} \\ \theta_{u|C}(s_u, s_C) &= \frac{N'_{u,C}(s_u,s_C)}{N'_{C}(s_C)} \\ \theta_{C}(s_C) &= \frac{N'_{C}(s_C)}{N'} \end{aligned} \tag{45}$$

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. A similar result in the case of decomposable distribution over trees can also be found in (Meila and Jordan, 2000).

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

From the definition of decomposable distribution over TAN structures, concretely from equation 28, 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

$$MPT(\boldsymbol{\beta}, \mathbf{N}') = \underset{E \in \mathcal{E}}{argmax} \prod_{u,v \in E} \beta_{u,v} \tag{46}$$

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

$$MPT(\boldsymbol{\beta}, \mathbf{N}') = \underset{E \in \mathcal{E}}{argmax} \sum_{u,v \in E} \log(\beta_{u,v}) \tag{47}$$

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. Hence, if we are given a decomposable distribution over TANs with hyperparameter $\boldsymbol{\beta}$, we can find the most probable undirected tree by computing the logarithm of every element in the matrix and then running any algorithm for finding the MST. The complexity of the MST algorithm for a complete graph is $\mathcal{O}(n^2)$ (Pettie and Ramachandran, 2002).

```
procedure MAPTANStructure (Dataset 𝒟,Matrix β,CountingSet N′)
    var
       CountingSet N′;
       Matrix lβ*;
    begin
        N′* = CalcN'PosteriorTAN(𝒟,N′);
        lβ* = CalcLogBetaPosteriorTAN(β,N′,N′*);
        return MST(lβ*);

procedure CalcN'PosteriorTAN (Dataset 𝒟,CountingSet N′)
    var
       CountingSet N′*;
    begin
        foreach attribute u
            foreach attribute v < u
                foreach value x_u ∈ A_u
                    foreach value x_v ∈ A_v
                        foreach value c ∈ C
                            N′*_{u,v,C}(x_u, x_v, c) = N′_{u,v}(x_u, x_v, c);
        foreach attribute x ∈ 𝒟
            foreach attribute u
                foreach attribute v < u
                    N′*_{u,v,C}(x_u, x_v, x_C) = N′*_{u,v,C}(x_u, x_v, x_C) + 1;
        return N′*;

procedure CalcLogBetaPosteriorTAN (Matrix β,CountingSet N′, N′*)
    var
       Matrix lβ*;
    begin
        foreach attribute u
            foreach attribute v < u
                lβ*_{u,v} = log β_{u,v} + CalcLogWTAN(N′,N′*,u,v);
        return lβ*;

procedure CalcLogWTAN (CountingSet N′, N′*, int u, v)
    begin
        w = 0;
        foreach value c ∈ C
            foreach value x_u ∈ A_u
                w = w + logΓ(N′_{u,C}(x_u, c)) - logΓ(N′*_{u,C}(x_u, c));
            foreach value x_v ∈ A_v
                w = w + logΓ(N′_{v,C}(x_v, c)) - logΓ(N′*_{v,C}(x_v, c));
            foreach value x_u ∈ A_u
                foreach value x_v ∈ A_v
                    w = w + logΓ(N′*_{u,v,C}(x_u, x_v, c))
                          - logΓ(N′_{u,v,C}(x_u, x_v, c));
        return w;
```

Algorithm 3: Computation of the MAP TAN structure

4.6. Calculating the MAP undirected tree structure given a prior decomposable distribution over TANs

In section 4.3 we claimed that if we assume a decomposable prior distribution over TANs with hyperparameters $\boldsymbol{\beta}$ and $\mathbf{N}'$ the posterior distribution given a dataset $\mathcal{D}$ follows a decomposable distribution over TANs with hyperparameters given by equations 41, 42 and 43. Since the posterior is a decomposable distribution over TANs, we can apply the former result for finding the most probable undirected tree over it and we get the MAP tree. We can translate this result into algorithm 3, that finds the MAP undirected tree given a dataset $\mathcal{D}$ and prior hyperparameters $\boldsymbol{\beta}$,$\mathbf{N}'$. Since the computation of MST is $\mathcal{O}(n^2)$, the time complexity of `MAPTANStructure` is bounded by `CalcN'PosteriorTAN`, which has complexity $\mathcal{O}((N + r^3) \cdot n^2)$.

4.7. 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 $\mathcal{O}((\log(\beta(n^2, n)) + k) \cdot n^2)$ for a complete graph (Katoh et al., 1981). It is easy to see that if in the last step of `MAPTANStructure` instead of computing the MST we compute the k MST and their relative weights as shown in algorithm 4, then the algorithm will return the k MAP undirected tree structures and their relative probabilities. The time complexity of the new algorithm is simply the addition of the complexity of `CalcN'Posterior` with that of computing the k MAP trees and that of computing the weights, giving $\mathcal{O}((N + r^3 + \log(\beta(n^2, n)) + k) \cdot n^2)$ which can be understood as $\mathcal{O}((N + r^3 + k) \cdot n^2)$ for all practical purposes.

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.

```
procedure k-MAPTANs (Dataset 𝒟,Matrix β,CountingSet N′, int k)
    var
       CountingSet N′;
       WeightedTreeSet WTS;
       Matrix lβ*;
    begin
        N′* = CalcN'PosteriorTAN(𝒟,N′);
        lβ* = CalcLogBetaPosteriorTAN(β,N′,N′*);
        WTS = k-MST(lβ*,k);
        CalcTreeWeights(WTS,lβ*);
        return WTS;
```

Algorithm 4: Computation of the k MAP TANs

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 4.2 and 4.3 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 4.3, the posterior distribution over the set of models is also a decomposable distribution over TANs, and applying the result of section 4.2 this decomposable posterior distribution over TANs can be used to calculate the probability of any observation S, s_C. When given an unclassified observation S, it can just calculate the probability $P(\mathcal{V} = S, \mathcal{C} = s_C|\mathcal{D}, \xi)$ for each possible class $s_C \in C$ and classify S 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 $\#C < r$ 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 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 accu-

rate 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 equation 38) 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 equation 43. 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 equation 43 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 5.2 and is described in detail by the following equations:

$$lmax = \log_{10} \max_{\substack{u \in V \\ v \in V \\ u \neq v}} W_{u,v} \tag{48}$$

$$lmin = \log_{10} \min_{\substack{u \in V \\ v \in V \\ u \neq v}} W_{u,v} \tag{49}$$

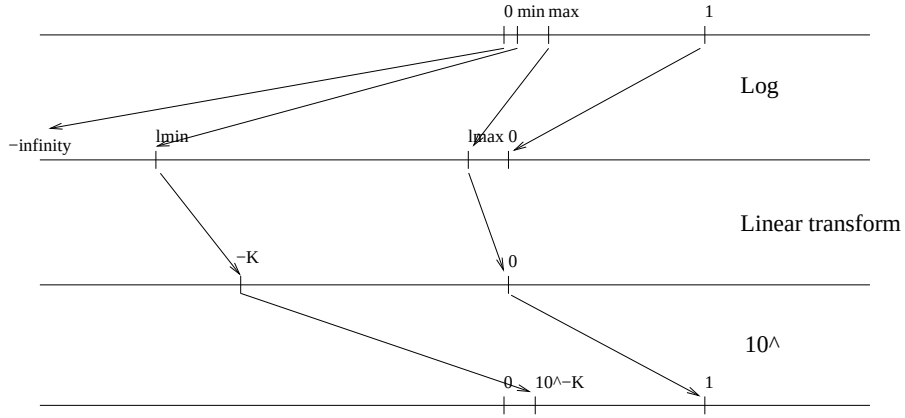


Figure 3. Transformation of weights for SSTBMATAN

$$a = \begin{cases} \frac{K}{lmax - lmin} & lmax - lmin > K \\ 1 & otherwise \end{cases} \tag{50}$$

$$b = -K - a * lmin \tag{51}$$

$$\widetilde{W}_{u,v} = 10^{a \log_{10}(W_{u,v}) + b} \tag{52}$$

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 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 allow 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 4

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. Apply algorithm 3 to find the undirected tree E underlying the MAP TAN structure given a dataset $\mathcal{D}$.
3. Randomly choose a root, create a directed tree $\overline{E}$ and from it a directed TAN structure $\overline{E}^*$.
4. Use equation 45 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. Apply algorithm 4 to 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 $\mathcal{O}((N + r^3 + \log(\beta(n^2, n)) + k) \cdot n^2)$ 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 equations 41 and 42. We have translated this requisite into equations 53 and 54:

$$\forall u, v \ ; \ 1 \leq u \neq v \ \leq n \ ; \ \beta_{u,v} = 1 \tag{53}$$

$$\begin{aligned} \forall u, v; 1 \leq u \neq v \leq n; \forall j \in A_v; \forall i \in A_u; \forall c \in C; \\ N'_{v,u,C}(j, i, c) = \frac{\lambda}{\#C \#A_u \#A_v} \end{aligned} \tag{54}$$

Defining $\boldsymbol{\beta}$ as in equation 53 means that we have the same amount of belief for any edge being in the TAN structure underlying the data. For fixed u, v, equation 54 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 et al., 1995). Experimental tests have shown that the algorithms are 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 et al., 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 4.5. 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, 2000b), decomposable distri-

butions 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 the others come at no price. Concretely it is worth pointing out that MAPTAN has exactly the same computation complexity as STAN.

6. Empirical comparison

In order to evaluate the classifiers we performed two sets of experiments. On the first one we tested its performance against Irvine datasets. On the second we tested it against 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 et al., 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 is multivalued, 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 function:

$$AUC = \frac{2}{\#C(\#C - 1)} \sum_{i<j} A(i, j) \tag{55}$$

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, 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 and 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 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 improves STAN consistently, 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.

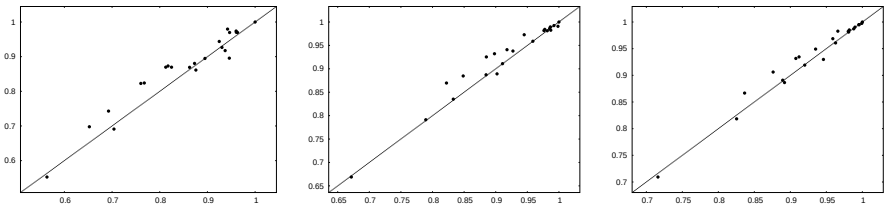
6.4.1. STAN *against* MAPTAN

The relationship between STAN and MAPTAN as data increases appears in figure 4. We present two sets of plots. In figure 4.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 5%. Below the significance test results we can see the percentage of training data that was used. In figure 4.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 5 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.

6.4.2. SSTBMATAN *against* TBMATAN

In figure 6 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 7). The difference between SSTBMATAN and TBMATAN sligthly benefits SSTBMATAN on a stable way along the

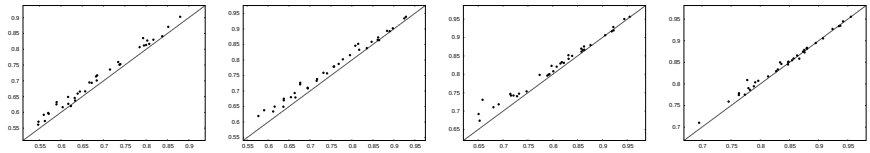


14 - 5
10%

11 - 3
50%

9 - 3
100%

(a) On Irvine datasets



31 - 0
25 instances

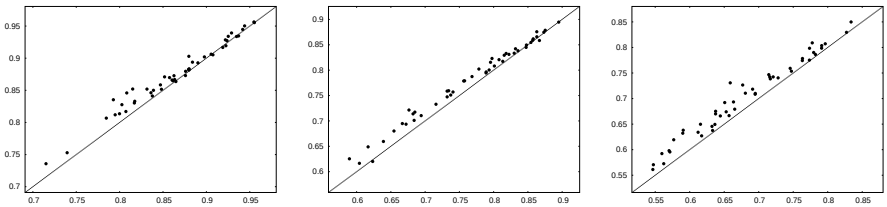
27 - 0
100 instances

23 - 0
400 instances

13 - 1
1600 instances

(b) On random Bayesian networks

Figure 4. MAPTAN vs. STAN. AUC comparison across learning data size



16 - 0
2 values

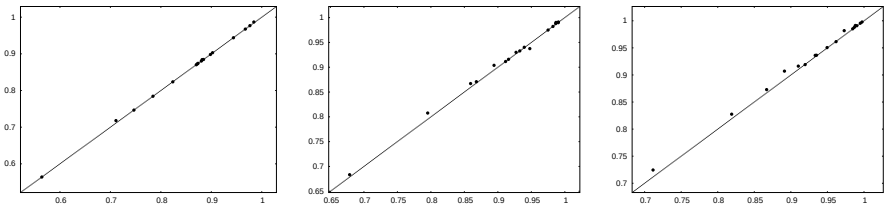
34 - 1
5 values

44 - 0
10 values

Figure 5. MAPTAN vs. STAN. AUC comparison across number of values of the attributes

different maximum induced width values and the different number of attributes we tested.

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

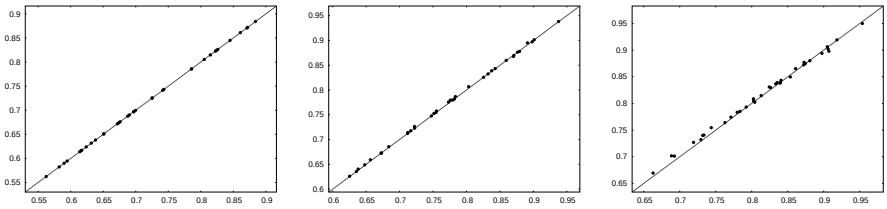


3 - 0
10%

8 - 1
50%

10 - 0
100%

(a) On Irvine datasets



1 - 0
25 instances

12 - 1
100 instances

17 - 3
400 instances

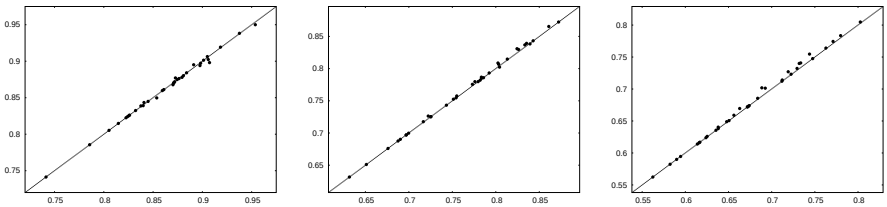
(b) On random Bayesian networks

Figure 6. SSTBMATAN vs. TBMATAN. AUC comparison across learning data size

are not fulfilled. Being more conservative regarding 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.

6.4.3. SSTBMATAN *against* MAPTAN

We can see in figure 8.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 9). 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.

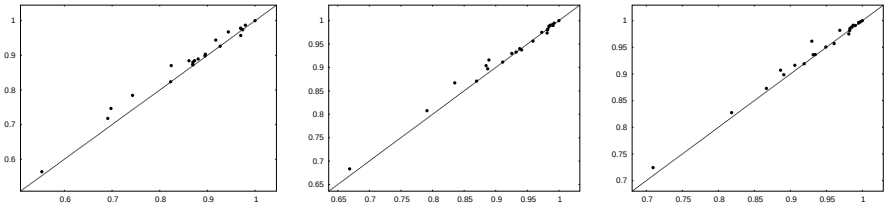


3 - 4
2 values

12 - 0
5 values

15 - 0
10 values

Figure 7. SSTBMATAN vs. TBMATAN. AUC comparison across number of values of the attributes

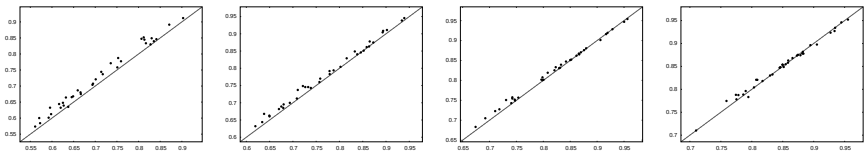


15 - 1
10%

12 - 3
50%

14 - 3
100%

(a) On Irvine datasets



34 - 0
25 instances

29 - 0
100 instances

19 - 0
400 instances

13 - 5
1600 instances

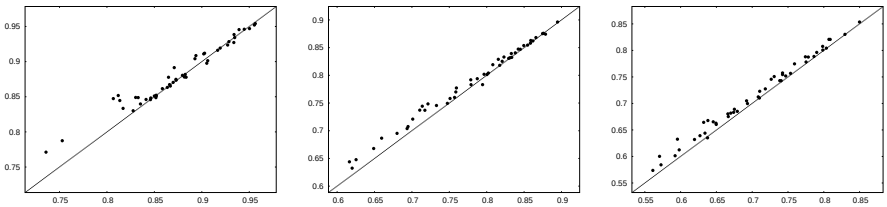
(b) On random Bayesian networks

Figure 8. SSTBMATAN vs. MAPTAN. AUC comparison across learning data size

On Irvine datasets (figure 8.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 *against* 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 10). However, the differences are much smaller than for

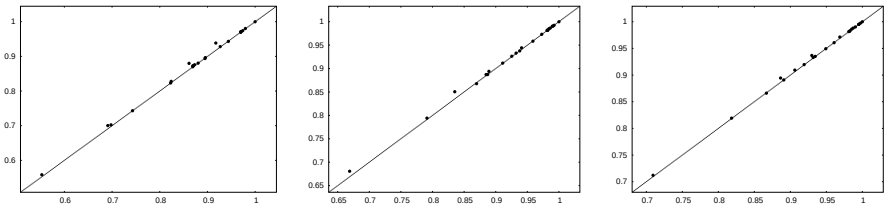


18 - 4
2 values

33 - 1
5 values

44 - 0
10 values

Figure 9. SSTBMATAN vs. MAPTAN. AUC comparison across number of values of the attributes

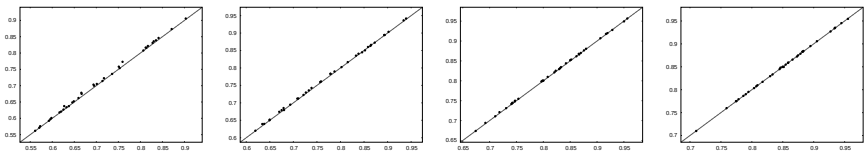


10 - 1
10%

6 - 1
50%

7 - 0
100%

(a) On Irvine datasets



22 - 0
25 instances

24 - 0
100 instances

19 - 0
400 instances

5 - 1
1600 instances

(b) On random Bayesian networks

Figure 10. MAPTAN+BMA vs. MAPTAN. AUC comparison across learning data size

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 corrected Meila and Jaakkola results for decomposable distributions over trees. Then we have introduced decomposable distributions over TANs, extending the results to TANs. 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 et al., 1997)).

8. Acknowledgements

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.

Appendix

A. Preliminaries

In this appendix we introduce three results that will be needed in the further development and then in appendix C we prove the results in sections 4.2 and 4.3.

A.1. The matrix tree theorem

Let $G = (V, E)$ be a multigraph and denote by $a_{u,v} = a_{v,u}$ the number of undirected edges between vertices u and v. Then the number of all spanning trees of G is given by the value of the determinant obtained from the following matrix by removing row u and column v.

$$A = \begin{bmatrix} \deg v_1 & -a_{1,2} & -a_{1,3} & \dots & a_{1,n} \\ -a_{2,1} & \deg v_2 & -a_{2,3} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ -a_{n,1} & -a_{n,2} & -a_{n,3} & \dots & \deg v_n \end{bmatrix} \tag{56}$$

Proof: See (West, 1999; Rubey, 2000).

□

A.2. The matrix tree theorem for decomposable distributions

Let P(E) be a distribution over spanning tree structures defined by equations 8 and 9. Then the normalization constant Z_β is equal to $|Q(\boldsymbol{\beta})|$ with $Q(\boldsymbol{\beta})$ being the first (n-1) lines and columns of the matrix $\overline{Q}(\boldsymbol{\beta})$ given by:

$$\overline{Q}_{u,v}(\boldsymbol{\beta}) = \overline{Q}_{v,u}(\boldsymbol{\beta}) = \begin{cases} -\beta_{u,v} & 1 \le u < v \le n \\ \sum\limits_{v'=1}^{n} \beta_{v',v} & 1 \le u = v \le n \end{cases} \tag{57}$$

Proof: See (Meila and Jaakkola, 2000a).

□

A.3. A useful result about Dirichlet distributions

A Dirichlet distribution is defined as

$$D(\theta_1,\ldots,\theta_k;N_1,\ldots,N_k)=\frac{\Gamma(\sum_{i=1}^{k}N_i)}{\prod_{i=1}^{k}\Gamma(N_i)}\prod_{i=1}^{k}{\theta_i}^{N_i-1} \tag{58}$$

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

$$\begin{aligned}&D(\theta_1,\ldots,\theta_r;n'_1,\ldots,n'_r)\prod_{i=1}^{r}{\theta_i}^{n_i}=\\&\frac{\Gamma(\sum_{i=1}^{r}n'_i)}{\prod_{i=1}^{r}\Gamma(n'_i)}\frac{\prod_{i=1}^{r}\Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^{r}n'_i+n_i)}D(\theta_1,\ldots,\theta_r;n'_1+n_1,\ldots,n'_r+n_r)\end{aligned} \tag{59}$$

and since the Dirichlet distribution is normalized you have that

$$\int_{\theta_1,\ldots,\theta_r}\cdots\int D(\theta_1,\ldots,\theta_r;n'_1,\ldots,n'_r)\prod_{i=1}^{r}{\theta_i}^{n_i}=\frac{\Gamma(\sum_{i=1}^{r}n'_i)}{\prod_{i=1}^{r}\Gamma(n'_i)}\frac{\prod_{i=1}^{r}\Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^{r}n'_i+n_i)} \tag{60}$$

Proof: By expanding the Dirichlet distribution by means of its definition in equation 58, grouping again into a Dirichlet and considering that the Dirichlet distribution is normalized distribution and hence integrates to one, we have that:

$$\int_{\theta_1,\ldots,\theta_r}\cdots\int D(\theta_1,\ldots,\theta_r;n'_1,\ldots,n'_r)\prod_{i=1}^{r}{\theta_i}^{n_i} \tag{61}$$

$$=\int_{\theta_1,\ldots,\theta_r}\cdots\int\frac{\Gamma(\sum_{i=1}^{r}n'_i)}{\prod_{i=1}^{r}\Gamma(n'_i)}\prod_{i=1}^{r}{\theta_i}^{n'_i+n_i-1} \tag{62}$$

$$=\int_{\theta_1,\ldots,\theta_r}\cdots\int\frac{\Gamma(\sum_{i=1}^{r}n'_i)}{\prod_{i=1}^{r}\Gamma(n'_i)}\frac{\prod_{i=1}^{r}\Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^{r}n'_i+n_i)}D(\theta_1,\ldots,\theta_r;n'_1+n_1,\ldots,n'_r+n_r) \tag{63}$$

$$=\frac{\Gamma(\sum_{i=1}^{r}n'_i)}{\prod_{i=1}^{r}\Gamma(n'_i)}\frac{\prod_{i=1}^{r}\Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^{r}n'_i+n_i)}\int_{\theta_1,\ldots,\theta_r}\cdots\int D(\theta_1,\ldots,\theta_r;n'_1+n_1,\ldots,n'_r+n_r) \tag{64}$$

$$=\frac{\Gamma(\sum_{i=1}^{r}n'_i)}{\prod_{i=1}^{r}\Gamma(n'_i)}\frac{\prod_{i=1}^{r}\Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^{r}n'_i+n_i)} \tag{65}$$

□

B. Detailed development for decomposable distributions over trees results

In this appendix we provide the proofs for the results in section 3.2.

B.1. Calculating probabilities under decomposable distributions over trees

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

$$P(\mathcal{X} = x|\xi) = \int\limits_{M \in \mathcal{M}} P(\mathcal{X} = x|M, \xi) P(M|\xi) \tag{66}$$

We can calculate the integral over the set of models by calculating the probability of each structure and then performing an addition over the set of structures. In fact, since we have assumed likelihood equivalence and our distribution over directed structures is uniform given the undirected structure, we can work over the set of undirected structures, that is

$$P(\mathcal{X} = x|\xi) = \sum_{E \in \mathcal{E}} P(\mathcal{X} = x|E, \xi) P(E|\xi) \tag{67}$$

where $P(E|\xi)$ comes given by:

$$P(E|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{68}$$

In order to calculate $P(\mathcal{X} = x|\xi)$ we have to calculate $P(\mathcal{X} = x|E, \xi)$ and then calculate the summmation in equation 67.

B.1.1. *Calculating* $P(\mathcal{X} = x|E, \xi)$

Using again likelihood equivalence we can express $P(\mathcal{X} = x|E, \xi)$ as the integral over any directed structure $\overline{E}$ which undirected structure coincides with E:

$$P(\mathcal{X} = x|E, \xi) = \int\limits_{\Theta_{\overline{E}}} \cdots \int P(\mathcal{X} = x|\overline{E}, \Theta_{\overline{E}}) P(\Theta_{\overline{E}}|\overline{E}, \xi) d\Theta_{\overline{E}} \tag{69}$$

$P(\mathcal{X} = x|\overline{E}, \Theta_{\overline{E}})$ is determined by the expansion of equation 1 taking into account the tree structure.

$$P(\mathcal{X} = x|\overline{E}, \Theta_{\overline{E}}) = \theta_{\rho_{\overline{E}}}(x_{\rho_{\overline{E}}}) \prod_{u,v \in \overline{E}} \theta_{v|u}(x_v, x_u) \tag{70}$$

$P(\Theta_{\overline{E}}|\overline{E},\xi)$ can be expanded from equations 10, 13 and 14 into

$$P(\Theta_{\overline{E}}|\overline{E},\xi) = D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i)) \quad (71)$$

Now we need to calculate the integral in equation 69 We define:

$$1_i^x(k) = \begin{cases} 1 & k = x_i \\ 0 & \text{otherwise} \end{cases} \quad (72)$$

$$1_{i,j}^x(k,l) = \begin{cases} 1 & k = x_i \wedge l = x_j \\ 0 & \text{otherwise} \end{cases} \quad (73)$$

It is easy to see that:

$$\sum_{j\in A_v} 1_{v,u}^x(j,i) = 1_u^x(i) \quad (74)$$

$$\sum_{i\in A_u} 1_u^x(i) = 1 \quad (75)$$

We can use this notation to expand the product

$$P(\mathcal{X} = x|\overline{E},\Theta_{\overline{E}})P(\Theta_{\overline{E}}|\overline{E},\xi) \quad (76)$$

by substituting equations 70 and 71 giving:

$$P(\mathcal{X} = x|\overline{E},\Theta_{\overline{E}})P(\Theta_{\overline{E}}|\overline{E},\xi) = D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \prod_{i\in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}}(i)^{1^x_{\rho_{\overline{E}}}(i)}$$
$$\times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[D(\theta_{v|u}(.,i); N'_{v,u}(.,i)) \prod_{j\in A_v} \theta_{v|u}(j,i)^{1^x_{v,u}(j,i)} \right] \quad (77)$$

By analyzing equation 77 we can see that the integral in equation 69 can be calculated by applying the result in equation 60 twice. This gives:

$$P(\mathcal{X} = x|E,\xi) = \frac{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}}}(i))}{\prod_{i\in A_{\rho_{\overline{E}}}} \Gamma(N'_{\rho_{\overline{E}}}(i))} \frac{\prod_{i\in A_{\rho_{\overline{E}}}} \Gamma(N'_{\rho_{\overline{E}}}(i) + 1^x_{\rho_{\overline{E}}}(i))}{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}}}(i) + 1^x_{\rho_{\overline{E}}}(i))}$$
$$\times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[\frac{\Gamma(\sum_{j\in A_v} N'_{v,u}(j,i))}{\prod_{j\in A_v} \Gamma(N'_{v,u}(j,i))} \frac{\prod_{i\in A_v} \Gamma(N'_{v,u}(j,i) + 1^x_{v,u}(j,i))}{\Gamma(\sum_{i\in A_v} N'_{v,u}(j,i) + 1^x_{v,u}(j,i))} \right] \quad (78)$$

This expression can be simplified by applying equations 11,12,74 and 75 and reorganizing:

$$P(\mathcal{X}=x|E,\xi)=\frac{\Gamma(N')}{\Gamma(N'+1)}\prod_{i\in A_{\rho_{\overline{E}}}}\frac{\Gamma(N'_{\rho_{\overline{E}}}(i)+1^x_{\rho_{\overline{E}}}(i))}{\Gamma(N'_{\rho_{\overline{E}}}(i))}$$
$$\times\prod_{u,v\in\overline{E}}\prod_{i\in A_u}\left[\frac{\Gamma(N'_u(i))}{\Gamma(N'_u(i)+1^x_u(i))}\prod_{i\in A_v}\frac{\Gamma(N'_{v,u}(j,i)+1^x_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))}\right] \tag{79}$$

Since the quotient $\frac{\Gamma(N'_*(*)+1^x_*(*))}{\Gamma(N'_*(*))}$ is $N'_*(*)$ if the condition expressed by the $1^x_*(*)$ is satisfied and 1 otherwise we have that:

$$P(\mathcal{X}=x|E,\xi)=\frac{1}{N'}N'_{\rho_{\overline{E}}}(x_{\rho_{\overline{E}}})\prod_{u,v\in\overline{E}}\left[\frac{N'_{v,u}(x_v,x_u)}{N'_u(x_u)}\right] \tag{80}$$

Defining h^x_0 and $h^x_{u,v}$ as in equations 24 and 25 it is easy to see that multiplying and dividing in equation 80 by the factor:

$$\prod_{v\in\Omega-\{\rho_{\overline{E}}\}}N'_v(x_v) \tag{81}$$

and rearranging we get:

$$P(\mathcal{X}=x|E,\xi)=h^x_0\ Z_\beta\prod_{u,v\in E}h^x_{u,v} \tag{82}$$

and the expression depends only of the undirected structure of the tree.

B.1.2. *Adding over Tree Structures*
Combining equations 68, 82 we get

$$P(\mathcal{X}=x|E,\xi)P(E|\xi)=h^x_0\prod_{u,v\in E}\beta_{u,v}h^x_{u,v} \tag{83}$$

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

$$P(\mathcal{X}=x|\xi)=h^x_0|Q(\boldsymbol{\beta}\ \mathbf{h}^x)| \tag{84}$$

□

B.2. Learning under decomposable distributions over trees

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

$$P(M|\mathcal{D},\xi) = P(\overline{E},\Theta_{\overline{E}}|\mathcal{D},\xi) = \frac{P(\overline{E},\Theta_{\overline{E}}|\xi)P(\mathcal{D}|\overline{E},\Theta_{\overline{E}},\xi)}{Z_{\mathcal{D}}} \tag{85}$$

The prior $P(\overline{E},\Theta_{\overline{E}}|\xi)$ is calculated combining equations 6,68, 71 giving:

$$\begin{aligned} P(\overline{E},\Theta_{\overline{E}}|\xi) = \frac{1}{Z_\beta} & \prod_{u,v\in\overline{E}} \beta_{u,v} \\ & \times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \\ & \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i)) \end{aligned} \tag{86}$$

$P(\mathcal{D}|\overline{E},\Theta_{\overline{E}},\xi)$ is the probability that the model generates the data in $\mathcal{D}$. Since $\mathcal{D}$ contains independent identically distributed observations, we have that

$$\begin{aligned} P(\mathcal{D}|\overline{E},\Theta_{\overline{E}},\xi) = & \prod_{i\in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}}(i)^{N_{\rho_{\overline{E}}}(i)} \\ & \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \prod_{j\in A_v} \theta_{v|u}(j,i)^{N_{v,u}(j,i)} \end{aligned} \tag{87}$$

Substituting equations 86 and 87 into 85 we get

$$\begin{aligned} P(\overline{E},\Theta_{\overline{E}}|\mathcal{D},\xi) = & \frac{1}{Z_\beta}\frac{1}{Z_{\mathcal{D}}} \prod_{u,v\in\overline{E}} \beta_{u,v} \\ & \times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.)) \prod_{i\in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}}(i)^{N_{\rho_{\overline{E}}}(i)} \\ & \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[D(\theta_{v|u}(.,i); N'_{v,u}(.,i)) \prod_{j\in A_v} \theta_{v|u}(j,i)^{N_{v,u}(j,i)} \right] \end{aligned} \tag{88}$$

Applying the result in equation 59 for all the Dirichlets we have that

$$
\begin{aligned}
P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}} \prod_{u,v\in\overline{E}} \beta_{u,v} \\
& \times \frac{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}}}(i))}{\prod_{i\in A_{\rho_{\overline{E}}}} \Gamma(N'_{\rho_{\overline{E}}}(i))} \frac{\prod_{i\in A_{\rho_{\overline{E}}}} \Gamma(N'_{\rho_{\overline{E}}}(i) + N_{\rho_{\overline{E}}}(i))}{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}}}(i) + N_{\rho_{\overline{E}}}(i))} \\
& \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[\frac{\Gamma(\sum_{j\in A_v} N'_{v,u}(j,i))}{\prod_{j\in A_v} \Gamma(N'_{v,u}(j,i))} \frac{\prod_{j\in A_v} \Gamma(N'_{v,u}(j,i) + N_{v,u}(j,i))}{\Gamma(\sum_{j\in A_v} N'_{v,u}(j,i) + N_{v,u}(j,i))} \right] \\
& \times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.) + N_{\rho_{\overline{E}}}(.)) \\
& \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i) + N_{v,u}(.,i))
\end{aligned}
\tag{89}
$$

This expression can be simplified by applying equations 11 and 12 (and similar ones for N) and reorganizing:

$$
\begin{aligned}
P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}} \frac{\Gamma(N')}{\Gamma(N'+N)} \prod_{u,v\in\overline{E}} \beta_{u,v} \\
& \times \prod_{i\in A_{\rho_{\overline{E}}}} \frac{\Gamma(N'_{\rho_{\overline{E}}}(i) + N_{\rho_{\overline{E}}}(i))}{\Gamma(N'_{\rho_{\overline{E}}}(i))} \\
& \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[\frac{\Gamma(N'_u(i))}{\Gamma(N'_u(i) + N_u(i))} \prod_{j\in A_v} \frac{\Gamma(N'_{v,u}(j,i) + N_{v,u}(j,i))}{\Gamma(N'_{v,u}(j,i))} \right] \\
& \times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.) + N_{\rho_{\overline{E}}}(.)) \\
& \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i) + N_{v,u}(.,i))
\end{aligned}
\tag{90}
$$

Defining $W_{u,v}$ as appears in equation 18, it is easy to see that multiplying and dividing in equation 90 by the factor:

$$
\prod_{v\in\Omega-\{\rho_{\overline{E}}\}} \prod_{i\in A_v} \frac{\Gamma(N'_v(i) + N_v(i))}{\Gamma(N'_v(i))} \tag{91}
$$

and rearranging we get:

$$\begin{aligned} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) &= \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)} \\ &\times \prod_{v\in\Omega}\prod_{i\in A_v}\frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} \\ &\times \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \\ &\times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.) + N_{\rho_{\overline{E}}}(.)) \\ &\times \prod_{u,v\in\overline{E}}\prod_{i\in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i) + N_{v,u}(.,i)) \end{aligned} \tag{92}$$

It is worth noting that if we use the definition of $W_{u,v}$ given by Meila and Jaakkola (see equation 17), our expression will keep a factor that depends on the directed tree structure (concretely on the root) and it would not be possible to continue with our development further on.

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

$$\int_{M\in\mathcal{M}} P(M|\mathcal{D},\xi) = \sum_{E\in\mathcal{E}}\int\cdots\int_{\Theta_{\overline{E}}} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D},\xi) = 1 \tag{93}$$

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:

$$\begin{aligned} \int\cdots\int_{\Theta_{\overline{E}}} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D},\xi) &= \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)} \\ &\times \prod_{v\in\Omega}\prod_{i\in A_v}\frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} \\ &\times \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \end{aligned} \tag{94}$$

The addition over structures can be calculated by means of the matrix tree theorem for decomposable priors, giving

$$\begin{aligned} \sum_{E\in\mathcal{E}}\int\cdots\int_{\Theta_{\overline{E}}} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D},\xi) &= \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)} \\ \times \prod_{v\in\Omega}\prod_{i\in A_v}\frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} &= 1 \end{aligned} \tag{95}$$

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

$$Z_{\mathcal{D}} = \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{|Q(\boldsymbol{\beta})|} \frac{\Gamma(N')}{\Gamma(N'+N)} \prod_{v\in\Omega} \prod_{i\in A_v} \frac{\Gamma(N'_v(i)+N_v(i))}{\Gamma(N'_v(i))} \tag{96}$$

Finally, substituting the result for $Z_{\mathcal{D}}$ in equation 92 we can see that the posterior is a decomposable distribution over trees with the hyperparameters updated as given by equations 15, 16 and 18:

$$\begin{aligned} P(\overline{E}, \Theta_{\overline{E}}|\mathcal{D}, \xi) = & \frac{1}{|Q(\boldsymbol{\beta}\mathbf{W})|} \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \\ & \times D(\theta_{\rho_{\overline{E}}}(.); N'_{\rho_{\overline{E}}}(.) + N_{\rho_{\overline{E}}}(.)) \\ & \times \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u}(.,i); N'_{v,u}(.,i) + N_{v,u}(.,i)) \end{aligned} \tag{97}$$

□

C. Detailed development for decomposable distributions over TANs results

In this appendix we provide the proofs for the results in sections 4.2 and 4.3.

C.1. Calculating probabilities under decomposable distributions over TANs

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

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \int_{M\in\mathcal{M}} P(\mathcal{V} = S, \mathcal{C} = s_C|M, \xi)P(M|\xi) \tag{98}$$

The development will be parallel to the one in section B.1. In this case, we can also work over the set of undirected structures having:

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\xi) = \sum_{E\in\mathcal{E}} P(\mathcal{V} = S, \mathcal{C} = s_C|E, \xi)P(E|\xi) \tag{99}$$

where $P(E|\xi)$ comes given by:

$$P(E|\xi) = \frac{1}{Z_\beta} \prod_{u,v\in E} \beta_{u,v} \tag{100}$$

C.1.1. *Calculating* $P(\mathcal{V} = S, \mathcal{C} = s_C|E, \xi)$
Using likelihood equivalence we can express $P(\mathcal{X} = x|E, \xi)$ as the integral over any directed TAN structure $\overline{E}^*$ which undirected structure coincides with E:

$$\begin{aligned} &P(\mathcal{V} = S, \mathcal{C} = s_C|E, \xi) = \\ &\quad = \int_{\Theta_{\overline{E}^*}} \cdots \int P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*}) P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) d\Theta_{\overline{E}^*} \end{aligned} \quad (101)$$

$P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*})$ is determined by the expansion of equation 1 taking into account the TAN structure.

$$P(\mathcal{V} = S, \mathcal{C} = s_C|\overline{E}^*, \Theta_{\overline{E}^*}) = \theta_C(s_C)\, \theta_{\rho_{\overline{E}}|C}(s_{\rho_{\overline{E}}}, s_C) \prod_{u,v \in \overline{E}} \theta_{v|u,C}(s_v, s_u, s_C) \quad (102)$$

$P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)$ can be expanded from equations 30, 34, 35 and 36 into

$$\begin{aligned} P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) = &D(\theta_C(.); N'_C(.)) \\ &\times \prod_{c \in C} D(\theta_{\rho_{\overline{E}}|C}(., c); N'_{\rho_{\overline{E}}, C}(., c)) \\ &\times \prod_{c \in C} \prod_{u,v \in \overline{E}} \prod_{i \in A_u} D(\theta_{v|u,C}(., i, c); N'_{v,u,C}(., i, c)) \end{aligned} \quad (103)$$

Now we need to calculate the integral in equation 101. We define:

$$1_C^{S,s_C}(c) = \begin{cases} 1 & c = s_C \\ 0 & \text{otherwise} \end{cases} \quad (104)$$

$$1_{i,C}^{S,s_C}(k, c) = \begin{cases} 1 & k = s_i \wedge c = s_C \\ 0 & \text{otherwise} \end{cases} \quad (105)$$

$$1_{i,j,C}^{S,s_C}(k, l, c) = \begin{cases} 1 & k = s_i \wedge l = s_j \wedge c = s_C \\ 0 & \text{otherwise} \end{cases} \quad (106)$$

It is easy to see that:

$$\sum_{j \in A_v} 1_{v,u,C}^{S,s_C}(j, i, c) = 1_{u,C}^{S,s_C}(i, c) \quad (107)$$

$$\sum_{i \in A_u} 1_{u,C}^{S,s_C}(i, c) = 1_C^{S,s_C}(c) \quad (108)$$

$$\sum_{c \in C} 1_C^{S,s_C}(c) = 1 \quad (109)$$

We can use this notation to expand the product

$$P(\mathcal{V}=S,\mathcal{C}=s_C|\overline{E}^*,\Theta_{\overline{E}^*})P(\Theta_{\overline{E}^*}|\overline{E}^*,\xi) \tag{110}$$

by substituting equations 102 and 103 giving:

$$\begin{aligned}
&P(\mathcal{V}=S,\mathcal{C}=s_C|\overline{E}^*,\Theta_{\overline{E}^*})P(\Theta_{\overline{E}^*}|\overline{E}^*,\xi) = \\
&\quad = D(\theta_C(.);N'_C(.))\prod_{c\in C}\theta_C(c)^{1_C^{S,s_C}(c)} \\
&\quad \times \prod_{c\in C}\left[D(\theta_{\rho_{\overline{E}}|C}(.,c);N'_{\rho_{\overline{E}},C}(.,c))\prod_{i\in A_{\rho_{\overline{E}}}}\theta_{\rho_{\overline{E}}|C}(i,c)^{1^{S,s_C}_{\rho_{\overline{E}},C}(i,c)}\right] \\
&\quad \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i\in A_u}\left[D(\theta_{v|u,C}(.,i,c);N'_{v,u,C}(.,i,c))\prod_{j\in A_v}\theta_{v|u,C}(j,i,c)^{1^{S,s_C}_{v,u,C}(j,i,c)}\right]
\end{aligned} \tag{111}$$

By analyzing equation 111 we can see that the integral in equation 101 can be calculated by applying the result in equation 60 three times. This gives:

$$\begin{aligned}
P(\mathcal{V}=S,\mathcal{C}=s_C|E,\xi) &= \frac{\Gamma(\sum_{c\in C}N'_C(c))}{\prod_{c\in C}\Gamma(N'_C(c))}\frac{\prod_{c\in C}\Gamma(N'_C(c)+1_C^{S,s_C}(c))}{\Gamma(\sum_{c\in C}N'_C(c)+1_C^{S,s_C}(c))} \\
&\times \prod_{c\in C}\left[\frac{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}}N'_{\rho_{\overline{E}},C}(i,c))}{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c))}\frac{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c)+1^{S,s_C}_{\rho_{\overline{E}},C}(i,c))}{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}}N'_{\rho_{\overline{E}},C}(i,c)+1^{S,s_C}_{\rho_{\overline{E}},C}(i,c))}\right] \\
&\times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i\in A_u}\left[\frac{\Gamma(\sum_{j\in A_v}N'_{v,u,C}(j,i,c))}{\prod_{j\in A_v}\Gamma(N'_{v,u,C}(j,i,c))}\frac{\prod_{i\in A_v}\Gamma(N'_{v,u,C}(j,i,c)+1^{S,s_C}_{v,u,C}(j,i,c))}{\Gamma(\sum_{i\in A_v}N'_{v,u,C}(j,i,c)+1^{S,s_C}_{v,u,C}(j,i,c))}\right]
\end{aligned} \tag{112}$$

This expression can be simplified by applying equations 31,32,33,107,108 and 109 and reorganizing:

$$P(\mathcal{V}=S,\mathcal{C}=s_C|E,\xi)=\frac{\Gamma(N')}{\Gamma(N'+1)}$$
$$\times\prod_{c\in C}\prod_{i\in A_{\rho_{\overline{E}}}}\frac{\Gamma(N'_{\rho_{\overline{E}},C}(i,c)+1^{S,s_C}_{\rho_{\overline{E}},C}(i,c))}{\Gamma(N'_{\rho_{\overline{E}},C}(i,c))}$$
$$\times\prod_{u,v\in\overline{E}}\prod_{c\in C}\prod_{i\in A_u}\left[\frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'_{u,C}(i,c)+1^{S,s_C}_{u,C}(i,c))}\prod_{i\in A_v}\frac{\Gamma(N'_{v,u,C}(j,i,c)+1^{S,s_C}_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))}\right] \tag{113}$$

Since the quotient $\frac{\Gamma(N'_*(*)+1^{S,s_C}_*(*))}{\Gamma(N'_*(*))}$ is $N'_*(*)$ if the condition expressed by the $1^{S,s_C}_*(*)$ is satisfied and 1 otherwise we have that:

$$\begin{aligned}P(\mathcal{V}=S,\mathcal{C}=s_C|E,\xi)&=\frac{1}{N'}\\&\times N'_{\rho_{\overline{E}},C}(s_{\rho_{\overline{E}}},s_C)\\&\times\prod_{u,v\in\overline{E}}\left[\frac{N'_{v,u,C}(s_v,s_u,s_C)}{N'_{u,C}(s_u,s_C)}\right]\end{aligned} \tag{114}$$

Defining h_0^{S,s_C} and $h_{u,v}^{S,s_C}$ as in equations 39 and 40 it is easy to see that multiplying and dividing in equation 114 by the factor:

$$\prod_{v\in V-\{\rho_{\overline{E}}\}}N'_{v,C}(s_v,s_C) \tag{115}$$

and rearranging we get:

$$P(\mathcal{V}=S,\mathcal{C}=s_C|E,\xi)=h_0^{S,s_C}\ Z_\beta\prod_{u,v\in E}\left(h_{u,v}^{S,s_C}\right) \tag{116}$$

and the expression depends only on the undirected structure of the tree.

C.1.2. *Adding over Tree Structures*
Combining equations 100, 116 we get

$$P(\mathcal{V}=S,\mathcal{C}=s_C|E,\xi)P(E|\xi)=h_0^{S,s_C}\prod_{u,v\in E}\left(\beta_{u,v}h_{u,v}^{S,s_C}\right) \tag{117}$$

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

$$P(\mathcal{V}=S,\mathcal{C}=s_C|\xi)=h_0^{S,s_C}|Q(\boldsymbol{\beta}\ \mathbf{h}^{S,s_C})| \tag{118}$$

□

C.2. Learning under decomposable distributions over TANs

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

$$P(M|\mathcal{D},\xi) = P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D},\xi) = \frac{P(\overline{E}^*, \Theta^*_{\overline{E}}|\xi)P(\mathcal{D}|\overline{E}^*, \Theta_{\overline{E}^*},\xi)}{Z_{\mathcal{D}}} \quad (119)$$

The prior $P(\overline{E}, \Theta_{\overline{E}}|\xi)$ is calculated combining equations 26,100, 103 giving:

$$\begin{aligned} P(\Theta_{\overline{E}^*}\overline{E}^*|\xi) = \frac{1}{Z_\beta} & \prod_{u,v\in\overline{E}} \beta_{u,v} \\ & \times D(\theta_C(.); N'_C(.)) \\ & \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \\ & \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \end{aligned} \quad (120)$$

$P(\mathcal{D}|\overline{E}^*, \Theta_{\overline{E}^*},\xi)$ is the probability that the model generates the data in $\mathcal{D}$. Since $\mathcal{D}$ contains independent identically distributed observations, we have that

$$\begin{aligned} P(\mathcal{D}|\overline{E}^*, \Theta_{\overline{E}^*},\xi) = & \prod_{c\in C} \theta_C(c)^{N_C(c)} \\ & \times \prod_{c\in C} \prod_{i\in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}|C}(i,c)^{N_{\rho_{\overline{E}},C}(i,c)} \\ & \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \prod_{j\in A_v} \theta_{v|u,C}(j,i,c)^{N_{v,u,C}(j,i,c)} \end{aligned} \quad (121)$$

Substituting equations 120 and 121 into 119 we get

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}} \prod_{u,v\in\overline{E}} \beta_{u,v} \\
& \times D(\theta_C(.); N'_C(.)) \prod_{c\in C} \theta_C(c)^{N_C(c)} \\
& \times \prod_{c\in C} \left[D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \prod_{i\in A_{\rho_{\overline{E}}}} \theta_{\rho_{\overline{E}}|C}(i,c)^{N_{\rho_{\overline{E}},C}(i,c)} \right] \\
& \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \prod_{j\in A_v} \theta_{v|u,C}(j,i,c)^{N_{v,u,C}(j,i,c)} \right]
\end{aligned}
\tag{122}
$$

Applying the result in equation 59 for all the Dirichlets we have that

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}} \prod_{u,v\in\overline{E}} \beta_{u,v} \\
& \times \frac{\Gamma(\sum_{c\in C} N'_C(c))}{\prod_{c\in C}\Gamma(N'_C(c))} \frac{\prod_{c\in C}\Gamma(N'_C(c)+N_C(c))}{\Gamma(\sum_{c\in C} N'_C(c)+N_C(c))} \\
& \times \prod_{c\in C} \left[\frac{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}},C}(i,c))}{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \frac{\prod_{i\in A_{\rho_{\overline{E}}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c)+N_{\rho_{\overline{E}},C}(i,c))}{\Gamma(\sum_{i\in A_{\rho_{\overline{E}}}} N'_{\rho_{\overline{E}},C}(i,c)+N_{\rho_{\overline{E}},C}(i,c))} \right] \\
& \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} \left[\frac{\Gamma(\sum_{j\in A_v} N'_{v,u,C}(j,i,c))}{\prod_{j\in A_v}\Gamma(N'_{v,u,C}(j,i,c))} \frac{\prod_{j\in A_v}\Gamma(N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))}{\Gamma(\sum_{j\in A_v} N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))} \right] \\
& \times D(\theta_C(.); N'_C(.)+N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))
\end{aligned}
\tag{123}
$$

This expression can be simplified by applying equations 31,32 and 33 (and similar ones for N) and reorganizing:

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)}\prod_{u,v\in\overline{E}}\beta_{u,v} \\
& \times \prod_{c\in C}\prod_{i\in A_{\rho_{\overline{E}}}}\frac{\Gamma(N'_{\rho_{\overline{E}},C}(i,c)+N_{\rho_{\overline{E}},C}(i,c))}{\Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \\
& \times \prod_{u,v\in\overline{E}}\prod_{c\in C}\prod_{i\in A_u}\left[\frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'_{u,C}(i,c)+N_{u,C}(i,c))}\prod_{j\in A_v}\frac{\Gamma(N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))}\right] \\
& \times D(\theta_C(.); N'_C(.)+N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))
\end{aligned}
\tag{124}
$$

Defining $W_{u,v}$ as appears in equation 43, it is easy to see that multiplying and dividing in equation 124 by the factor:

$$
\prod_{v\in V-\{\rho_{\overline{E}}\}}\prod_{c\in C}\prod_{i\in A_v}\frac{\Gamma(N'_{v,C}(i,c)+N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \tag{125}
$$

and rearranging we get:

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)} \\
& \times \prod_{c\in C}\prod_{v\in V}\prod_{i\in A_v}\frac{\Gamma(N'_{v,C}(i,c)+N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \\
& \times \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \\
& \times D(\theta_C(.); N'_C(.)+N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))
\end{aligned}
\tag{126}
$$

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

$$\int_{M \in \mathcal{M}} P(M|\mathcal{D}, \xi) = \sum_{E \in \mathcal{E}} \int \cdots \int_{\Theta_{\overline{E}^*}} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = 1 \tag{127}$$

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:

$$\begin{aligned} \int \cdots \int_{\Theta_{\overline{E}^*}} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) &= \frac{1}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \frac{\Gamma(N')}{\Gamma(N'+N)} \\ &\times \prod_{c \in C} \prod_{v \in V} \prod_{i \in A_v} \frac{\Gamma(N'_{v,C}(i,c) + N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \\ &\times \prod_{u,v \in \overline{E}} W_{u,v} \beta_{u,v} \end{aligned} \tag{128}$$

The addition over structures can be calculated by means of the matrix tree theorem for decomposable priors, giving

$$\begin{aligned} \sum_{E \in \mathcal{E}} \int \cdots \int_{\Theta_{\overline{E}^*}} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) &= \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \frac{\Gamma(N')}{\Gamma(N'+N)} \\ \times \prod_{c \in C} \prod_{v \in V} \prod_{i \in A_v} \frac{\Gamma(N'_{v,C}(i,c) + N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} &= 1 \end{aligned} \tag{129}$$

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

$$Z_{\mathcal{D}} = \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{|Q(\boldsymbol{\beta})|} \frac{\Gamma(N')}{\Gamma(N'+N)} \prod_{c \in C} \prod_{v \in V} \prod_{i \in A_v} \frac{\Gamma(N'_{v,C}(i,c) + N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \tag{130}$$

Finally, substituting the result for $Z_{\mathcal{D}}$ in equation 126 we can see that the posterior is a decomposable distribution with the parameters updated as given by equations 41, 42 and 43:

$$\begin{aligned} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) &= \frac{1}{|Q(\boldsymbol{\beta}\mathbf{W})|} \prod_{u,v \in \overline{E}} W_{u,v} \beta_{u,v} \\ &\times D(\theta_C(.); N'_C(.) + N_C(.)) \\ &\times \prod_{c \in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c) + N_{\rho_{\overline{E}},C}(.,c)) \\ &\times \prod_{c \in C} \prod_{u,v \in \overline{E}} \prod_{i \in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c) + N_{v,u,C}(.,i,c)) \end{aligned} \tag{131}$$

□

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*.

Cerquides, J. and López de Màntaras, R. (2003). Tractable bayesian learning of tree augmented naive bayes classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 75–82.

Chow, C. and Liu, C. (1968). Aproximatting Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory*, 14:462–467.

Domingos, P. and 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., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29:131–163.

Hand, D. and Till, R. (2001). A simple generalization of the area under the roc curve to multiple class classification problems. *Machine Learning*, 45(2):171–186.

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

Ide, J. and 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., and Mine, H. (1981). An algorithm for finding k minimum spanning trees. *SIAM J. Comput.*, 10(2):247–255.

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

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

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

Meila, M. and Jaakkola, T. (2000b). Tractable bayesian learning of tree belief networks. In *Proc. of the Sixteenth Conference on Uncertainty in Artificial Intelligence*.

Meila, M. and Jordan, M. I. (2000). Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48.

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

Rubey, M. (2000). Counting spanning trees. Diplomarbeit.

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

West, D. (1999). *Introduction to Graph Theory, Second Edition.* Prentice Hall.