INSTITUT D'INVESTIGACIÓ EN
INTEL.LIGÈNCIA ARTIFICIAL

CSIC

TECHNICAL REPORT TR-2003-02

http://www.iiia.csic.es/~mantaras/ReportIIIA-TR-2003-02.pdf

Tractable Bayesian Learning of Tree Augmented Naive Bayes Classifiers

Jesús Cerquides
Ramon López de Màntaras

JANUARY 2003

Abstract

Bayesian classifiers as *Naive Bayes* or *Tree Augmented Naive Bayes* (TAN) have shown excellent performance given their simplicity and heavy underlying independence assumptions. In this paper we show that the expression resulting from the Bayesian Model Averaging of TAN classifiers can be integrated into closed form if we assume as prior probability distribution a decomposable distribution. This result allows for the construction of a classifier with a smaller learning time and a larger classification time than TAN. Empirical results show that, as expected, the classifier is more accurate than TAN.

1 Introduction

Bayesian classifiers as *Naive Bayes* [16] or *Tree Augmented Naive Bayes* (TAN) [7] have shown excellent performance given their simplicity and heavy underlying independence assumptions.

Furthermore, it has been shown [4, 14] that *Naive Bayes* predictions and probability estimations can benefit from incorporating uncertainty in model selection. In [14] Kontkanen et al. introduce an approach named Bayesian Instance-Based Learning that can be seen as a version of Bayesian Model Averaging [9] and demonstrate that it improves predictions and probability estimates. A slightly improved development of the same idea by the use of a prior given by the principle of indifference is presented in [4].

In the case of TAN, a development inspired in the same idea is presented in [2], where to overcome the difficulty of exactly calculating the averaged classifier the idea of Local Bayesian Model Averaging is introduced to calculate an approximation. In this case predictions are also improved.

In this paper we show that the Bayesian Model Averaging of TAN can be integrated in closed form and that it leads to improved classification performance. The paper is organized as follows. In section 2 we introduce *Tree Augmented Naive Bayes* and the notation that we will use in the rest of the paper. After that, in section 3 we develop the closed expression for TAN. We start by introducing Bayesian Model Averaging, then we explain decomposable distributions over tree structures and parameters built upon the idea of decomposable priors as proposed by Meila and Jaakola [17] to end up showing that given a decomposable distribution is is possible to calculate the probability of an unseen observation and that given a prior decomposable distribution, our posterior distribution after observing a set of data is also a decomposable distribution. This results allow us to provide a closed expression for the Bayesian Model Averaging of TAN which we will name TBMATAN. In section 4 we notice that TBMATAN has a major drawback that makes its usage difficult for large datasets due to the fact that it implies the calculation of an ill-conditioned determinant that requires that the floating point precision increases with the dataset size and hence the computing time required for the algorithm. To solve this drawback we introduce SSTBMATAN as a way of approximating TBMATAN. In section 5 we study the empirical characteristics of TBMATAN and show that it leads to improving classification accuracy and to a better approximation of the class probabilities with respect to TAN. We also show that the empirical results for SSTBMATAN do not differ significantly from the ones obtained by TBMATAN. We end up with some conclusions and future work in section 6.

2 Tree Augmented Naive Bayes

Tree Augmented Naive Bayes (TAN) appears as a natural extension to the *Naive Bayes* classifier. *Naive Bayes* [14, 16, 6] is a very simple classifier that performs very well on small and not-so-small datasets. The assumption made by *Naive Bayes* is that all the attributes in the dataset are conditionally independent given the value of the class. This is a very strong assumption that is very likely not to be fulfilled, but the classifier works well in practice even when strong dependencies hold in the dataset. Furthermore, it has been shown to be optimal under zero-one loss in a larger subspace [6]. Given these facts, the general idea is that if we somehow relax the assumptions that are made and keep the "way of reasoning", we can get a more accurate classifier. This has been tried in different ways [7, 11, 12, 13, 15, 19]. From our point of view TAN are the more coherent and best performing enhancement to *Naive Bayes* up to now. TAN 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 can be seen in Figure 1(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 [7]. Finally we present the improvements introduced to TAN in [2, 3].

2.1 Formalization and notation

The notation followed in the paper is an effort to put together the different notations used in [2, 8, 7, 17] 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 will note as $V = \{A_1, \ldots, A_n\}$ the set of attributes in a classified discrete domain that are not the class.

Given an attribute A, we will note $\#A$ as the number of different values of A. We define $\#\Omega = \prod_{i=1}^{m} \#X_i$ and $\#\Omega_C = \#C \prod_{i=1}^{n} \#A_i$.

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 $S = (s_1, \ldots, s_n) \in A_1 \times \ldots \times A_n$. 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. 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

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

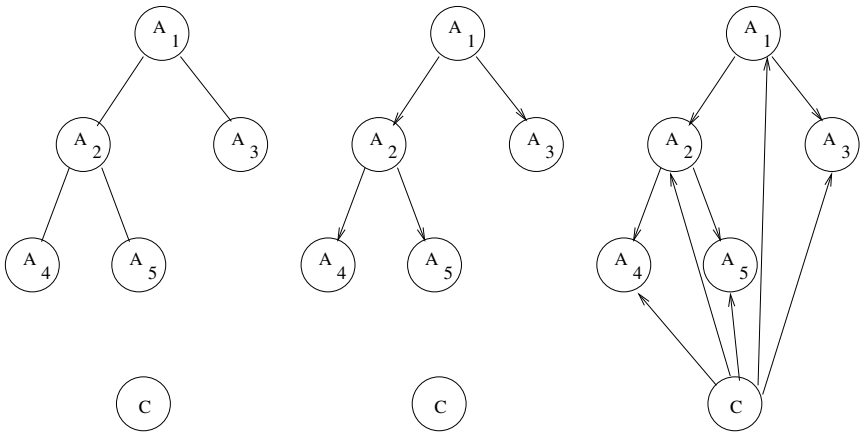
The application of bayesian networks for classification can be very simple. For example suppose we have an algorithm that given a classified discrete domain Ω_C and a dataset $\mathcal{D}$ over Ω_C returns a bayesian network B over $\mathbf{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{C}\}$ where $\mathcal{A}_i$ (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 $\underset{c \in C}{argmax}(P_B(s_1, \ldots, s_n, c))$. This simple mechanism allows us to see any bayesian network learning algorithm as a classifier.

2.1.3 Dirichlet distributions

The Dirichlet probability distribution is frequently used in bayesian networks because it is closed under multinomial sampling [8]. It is defined as:

$$D(\theta_1, \ldots, \theta_k; N_1, \ldots, N_k) = \frac{\Gamma(\sum_{i=1}^{k} N_i)}{\prod_{i=1}^{k} \Gamma(N_i)} \prod_{i=1}^{k} {\theta_i}^{N_i - 1} \tag{2}$$



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

Figure 1: Notation for learning with trees

2.1.4 Learning with Trees

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 (has no cycles). We will note as $\overline{E}$ a directed tree for E. We will use $u, v \in E$ instead of $(\mathcal{A}_u, \mathcal{A}_v) \in E$ for compactness. 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 1. We note the root of a directed tree $\overline{E}$ as $\rho_{\overline{E}}$ (i.e. in Figure 1(b) we have that $\rho_{\overline{E}} = A_1$).

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

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

$\boldsymbol{\theta}_C = \{\theta_C(c) | c \in C\}$ where $\theta_C(c) = p(S_C = c|M)$

$\boldsymbol{\theta}_{\rho_{\overline{E}}|C} = \{\theta_{\rho\overline{E}|C}(i,c) | i \in A_{\rho(\overline{E}}, c \in C\}$ where
$\theta_{\rho_{\overline{E}}|C}(i,c) = p(S_{\rho_{\overline{E}}} = i | S_C = c, M)$

For each $u, v \in \overline{E}$: $\boldsymbol{\theta}_{v|u,C} = \{\theta_{v|u,C}(j,i,c) | j \in A_v, i \in A_u, c \in C\}$ where
$\theta_{v|u,C}(j,i,c) = p(S_v = j | S_u = i, S_C = c, M)$.

2.2 Learning maximum likelihood TAN

One of the measures used to learn bayesian networks is the *log likelihood*:

$$LL(B|\mathcal{D}) = \sum_{x \in \mathcal{D}} \log(P_B(x)) \tag{3}$$

An interesting property of the TAN family is that we have an efficient procedure [7] for identifying the structure of the network which maximizes likelihood. The procedure and the theorem are given below.

Theorem 1 (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 $O(N \cdot n^2)$.*

To learn the maximum likelihood TAN we should use Theorem 1 to determine the structure and the following equation to compute the parameters.

$$\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = \frac{N_{i,\Pi_i}(x_i, \Pi_{x_i})}{N_{\Pi_i}(\Pi_{x_i})} \tag{4}$$

It has been shown [7] that equation 4 leads to "overfitting" the model. Also in [7] Friedman et al. propose to use the parameters as given by

$$\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = \frac{N_{i,\Pi_i}(x_i, \Pi_{x_i})}{N_{\Pi_i}(\Pi_{x_i}) + N^0_{i|\Pi_i}} + \frac{N^0_{i|\Pi_i}}{N_{\Pi_i}(\Pi_{x_i}) + N^0_{i|\Pi_i}} \frac{N_i(x_i)}{N} \tag{5}$$

and suggest setting $N^0_{i|\Pi_i} = 5$ based on empirical results. Using equation 5 to fix the parameters improves the accuracy of the classifier. In our opinion, no well founded justification is given for the improvement. In the following section we revisit the results in [3] and show that we can get an alternative parameter fixing equation with a well founded theoretical support and equivalent classification accuracy.

```
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) = Σ_{x∈A_i, y∈A_j, z∈C} P(x,y,z) log( P(x,y|z) / (P(x|z)P(y|z)) )
    end
    G = ConstructUndirectedGraph(I_P);
    UT = MaximumWeightedSpanningTree(G);
    T = MakeDirected(UT);
    TAN = AddClass(T);
    return TAN;
```

Algorithm 1: TAN construction procedure

2.3 Learning multinomial sampling TAN

In [3] we introduced an alternative approach to learning bayesian networks which we named "multinomial sampling approach". This alternative approach is based on changing the usual statement that describes what is learning a bayesian network from statement 1 to statement 2 (notation has been changed on both statements to agree with the one used in this paper).

Statement 1 (Friedman, Geiger & Goldszmidt, 1997) *Given a dataset $\mathcal{D}$ of instances of* **U** *find the network B that best matches $\mathcal{D}$.*

Statement 2 *Given a dataset $\mathcal{D}$, interpret $\mathcal{D}$ as a sample of a probability distribution P^* and find the network B that best matches P^*.*

The main difference between both statements is that statement 2 gives a well founded solution for the problem of "overfitting". In [2, 3] this multinomial approach has been applied to TAN with the result that we should approximate probabilities by:

$$P^*(x_1, \ldots, x_m) = \frac{N_{1,\ldots,m}(x_1, \ldots, x_m) + \frac{\lambda}{\#\Omega}}{N + \lambda} \tag{6}$$

where λ is a hyperparameter.

Instead of maximizing likelihood, under the multinomial sampling approach we should minimize the Kullback-Leibler divergence between P^* and the probability distribution represented by our network. In order to do that we should estimate the parameters using:

$$\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = \frac{N_{i,\Pi_i}(x_i, \Pi_{x_i}) + \lambda \frac{\#\overline{\Pi_i}}{\#\Omega}}{N_{\Pi_i}(\Pi_{x_i}) + \lambda \frac{\#X_i \#\overline{\Pi_i}}{\#\Omega}} \tag{7}$$

where $\#\overline{\Pi_i} = \prod_{\mathcal{X}_j \in \overline{\Pi_i}} \#X_j$ and we remind that $\overline{\Pi_i}$ stands for the set of variables which are not parents of $\mathcal{X}_i$ in the network excluding $\mathcal{X}_i$.

In [2, 3] the usage of $\lambda = 10$ was suggested as a good value after empirical tests and the multinomial sampling approach was compared to the maximum likelihood (equation 4) and softened maximum likelihood (equation 5) parameter estimations. The results were that multinomial sampling is clearly better than maximum likelihood. When compared to softened maximum likelihood, it was observed that multinomial sampling provides an equivalent classification accuracy but improves the quality of the probabilities assigned to the class.

3 Development of the Averaged Tree Augmented Naive Bayes

3.1 BMA classification

We are faced with the problem of defining a good classifier for a classified dataset. If we accept that there is a probabilistic model behind our dataset, we have two alternatives:

1. We know the model M (both structure and parameters) that is generating the data in advance. In this case it is a matter of probabilistic computation. We should be able to calculate $p(S_C = c_i|S, M)$ and to choose the class with the highest probability. No learning is performed, because we knew the model in advance.

2. We are given a set of possible models $\mathcal{M}$. This the situation, for instance, when learning decision trees, neural networks or bayesian classifiers. The usual approach followed is to let an algorithm choose the model M that fits the data best. This can give good results, if the model selected accounts for a good share of the posterior probability

distribution function over the set of models or if its predictions coincide with the ones given by the majority of the models. In spite of that, probability theory tell us we should take a weighted average where each model prediction is weighted by the probability of the model given the data. Formally:

$$p(S, S_C|\mathcal{D}, \xi) = \int_{M \in \mathcal{M}} p(S, S_C|M) p(M|\mathcal{D}, \xi) \tag{8}$$

Applying this equation is commonly known as Bayesian Model Averaging [9]. In practice, the problem is that for most models it is very hard to find a closed form for the integral. This has led to the appearance of methods such as Local Bayesian Model Averaging [2, 3], that approximate the integral over a subset of highly probable models.

In the following we prove that if we fix our set of models $\mathcal{M}$ to TAN models and assume a decomposable distribution as prior probability distribution over the set of models, the integral for $p(S, S_C|\mathcal{D}, \xi)$ in equation 8 can be integrated in closed form. In section 3.2 we present decomposable priors over structures and parameters. In sections 3.3 and 3.4 we give results that allow us to calculate the probability of an unclassified observation under a decomposable distribution and to refine a prior decomposable distribution into a posterior decomposable distribution given a dataset. Both results are developed into detail in Appendix B. After presenting the results we conclude the development by putting them together to construct a classifier in section 3.5.

3.2 Decomposable distributions over TANs

In order to apply Bayesian Model Averaging, it is necessary to have a prior probability distribution over the set of models $\mathcal{M}$. Decomposable priors were introduced by Meila and Jaakola in [17] where it was demonstrated for tree belief networks that if we assume a decomposable prior, the posterior probability is also decomposable and can be completely determined analytically in polynomial time.

In this section we introduce decomposable distributions over TANs, which are probability distributions in the space $\mathcal{M}$ of TAN models and an adaptation of decomposable priors, as they appear in [17], to the task of learning TAN. As can be seen in [17] decomposable priors are based on four assumptions: likelihood equivalence, parameter independence, parameter modularity and connectivity. These four assumptions are also on the basis of the development of decomposable distributions over TANs. Specially significant in

order to understand the developments in this paper is likelihood equivalence. This assumption states that in all possible parameterizations consistent with a given undirected tree E the distribution will assign the same probability mass to any measurable subset in parameter space. This provides us with a very valuable tool when integrating over parameters, because it allows us to integrate over the parameters of any directed tree $\overline{E}$ obtained from E because for all of them the result of the integration should be the same.

Decomposable distributions 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 so that it is also independent on the structure. In the rest of the paper we will assume ξ implies a decomposable distribution over $\mathcal{M}$ with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ (these hyperparameters will be explained along the development). Under this assumption, 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:

$$P(M|\xi) = P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) = P(\overline{E}^*|\xi)P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) \tag{9}$$

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

3.2.1 Decomposable distribution over TAN structures

One of the hyperparameters of a decomposable distribution is an $n \times n$ matrix $\boldsymbol{\beta} = (\beta_{u,v})$ such that $\forall u, v : 1 \leq u, v \leq n \ : \beta_{u,v} = \beta_{v,u} \geq 0 \ ; \ \beta_{v,v} = 0$. 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.

Given ξ, the probability of a TAN structure $\overline{E}^*$ is defined as:

$$p(\overline{E}^*|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \tag{10}$$

where Z_β is a normalization constant with value:

$$Z_\beta = \sum_{E \in \mathcal{E}} \prod_{u,v \in E} \beta_{u,v} \tag{11}$$

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

3.2.2 Decomposable distribution over TAN parameters

Applying equation 1 to the case of TAN we have that

$$P(\Theta_{\overline{E}^*}|\overline{E}^*,\xi) = P(\boldsymbol{\theta}_C|\overline{E}^*,\xi)\ P(\boldsymbol{\theta}_{\rho_{\overline{E}}|C}|\overline{E}^*,\xi)\ \prod_{u,v\in\overline{E}} P(\boldsymbol{\theta}_{v|u,C}|\overline{E}^*,\xi) \tag{12}$$

A decomposable distribution has a hyperparameter $\mathbf{N}' = \{N'_{v,u,C}(j,i,c)|1 \leq u \neq v \leq n\ ;\ j \in A_v\ ;\ i \in A_u\ ;\ c \in C\}$. We define $N'_{u,C}(i,c)$,$N'_C(c)$ and N' as:

$$N'_{u,C}(i,c) = \sum_{j\in A_v} N'_{v,u,C}(j,i,c) \tag{13}$$

$$N'_C(c) = \sum_{i\in A_u} N'_{u,C}(i,c) \tag{14}$$

$$N' = \sum_{c\in C} N'_C(c) \tag{15}$$

Given ξ, a decomposable probability distribution over parameters with hyperparameter $\mathbf{N}'$ is defined by equation 12 and the following set of Dirichlet distributions:

$$P(\boldsymbol{\theta}_C|\overline{E},\xi) = D(\theta_C(.); N'_C(.)) \tag{16}$$

$$P(\boldsymbol{\theta}_{\rho_{\overline{E}}|C}|\overline{E},\xi) = \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \tag{17}$$

$$P(\boldsymbol{\theta}_{v|u,C}|\overline{E},\xi) = \prod_{c\in C}\prod_{i\in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \tag{18}$$

If the conditions in equations 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18 hold, we will say that $P(M|\xi)$ follows a decomposable distribution with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$.

3.3 Calculating probabilities with decomposable distributions

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

$$P(S, S_C|\xi) = \int_{M\in\mathcal{M}} P(S, S_C|M)P(M|\xi) \tag{19}$$

Let $Q : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n-1 \times n-1}$. For any real $n \times n$ matrix $\boldsymbol{\tau}$ we define $Q(\boldsymbol{\tau})$ as the first $n-1$ lines and columns of the matrix $\overline{Q}(\boldsymbol{\tau})$ where

$$\overline{Q}_{u,v}(\boldsymbol{\tau}) = \overline{Q}_{v,u}(\boldsymbol{\tau}) = \begin{cases} -\tau_{u,v} & 1 \leq u < v \leq n \\ \sum_{v'=1}^{n} \tau_{v',v} & 1 \leq u = v \leq n \end{cases} \tag{20}$$

The integral for $P(S, S_C|\xi)$ can be calculated in closed form by applying the matrix tree theorem (see Appendix A.1 and [17]) and expressed in terms of the previously introduced Q as:

$$P(S, S_C|\xi) = h_0^{S,S_C} |Q(\boldsymbol{\beta}\ \mathbf{h}^{S,S_C})| \tag{21}$$

where

$$h_0^{S,S_C} = \frac{1}{Z_\beta} \frac{1}{N'} \prod_{A_u \in V} N'_{u,C}(S_u, S_C) \tag{22}$$

$$\mathbf{h}^{S,S_C} = \left(h_{u,v}^{S,S_C}\right) \text{ where } h_{u,v}^{S,S_C} = \frac{N'_{v,u,C}(S_v, S_u, S_C)}{N'_{u,C}(S_u, S_C) N'_{v,C}(S_v, S_C)} \tag{23}$$

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

3.4 Learning with decomposable distributions

Assume that our data is generated by a TAN model and that $P(M|\xi)$ follows a decomposable distribution with hyperparameters $\boldsymbol{\beta}$,$\mathbf{N}'$. Then, $P(M|\mathcal{D}, \xi)$, the posterior probability distribution after observing a dataset $\mathcal{D}$ is a decomposable distribution with parameters $\boldsymbol{\beta}^*$,$\mathbf{N}'^*$ given by:

$$\beta^*_{u,v} = \beta_{u,v} W_{u,v} \tag{24}$$

$$N'^*_{u,v,C}(j, i, c) = N'_{u,v,C}(j, i, c) + N_{u,v,C}(j, i, c) \tag{25}$$

where

$$\begin{aligned} W_{u,v} = & \prod_{c \in C} \prod_{i \in A_u} \frac{\Gamma(N'_{u,C}(i, c))}{\Gamma(N'_{u,C}(i, c) + N_{u,C}(i, c))} \\ & \prod_{c \in C} \prod_{j \in A_v} \frac{\Gamma(N'_{v,C}(j, c))}{\Gamma(N'_{v,C}(j, c) + N_{v,C}(j, c))} \\ & \prod_{c \in C} \prod_{i \in A_u} \prod_{j \in A_v} \frac{\Gamma(N'_{v,u,C}(j, i, c) + N_{v,u,C}(j, i, c))}{\Gamma(N'_{v,u,C}(j, i, c))} \end{aligned} \tag{26}$$

The proof appears in in appendix B.2.

3.5 Putting it all together

Putting together the results from sections 3.3 and 3.4 we can easily design a 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 as prior over the set of models. Applying the result from section 3.4, the posterior distribution over the set of models is also a decomposable distribution and applying the result of section 3.3 this decomposable posterior distribution 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 $p(S, S_C|\mathcal{D}, \xi)$ for each possible class $S_C \in C$ and classify S in the class with highest probability.

We have mentioned that the classifier assumes a decomposable distribution 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 for 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 24 and 25. We have translated this requisite into equations 27 and 28:

$$\forall u, v \ ; \ 1 \leq u \neq v \ \leq n \ ; \ \beta_{u,v} = 1 \tag{27}$$

$$\forall u, v; 1 \leq u \neq v \leq n; \forall j \in A_v; \forall i \in A_u; \forall c \in C; N'_{v,u,C}(j, i, c) = \frac{\lambda}{\#C\#A_u\#A_v} \tag{28}$$

Defining $\boldsymbol{\beta}$ as in equation 27 means that we have the same amount of belief for any edge being in the TAN structure underlying the data. Fixed u, v, equation 28 assigns the same probability to any (j, i, c) such that $j \in A_v$,$i \in A_u$ and $c \in C$ and the value assigned is coherent with the multinomial sampling approach. λ is an "equivalent sample size" for our prior in the sense of Heckerman et al. in [8]. In our experiments we have fixed $\lambda = 10$. In the following TBMATAN will refer to the classifier described in this section.

4 Approximating TBMATAN

TBMATAN can theoretically be implemented by an algorithm with $\mathcal{O}(N \cdot n^2)$ learning time and $\mathcal{O}(\#C \cdot n^3)$ time for classifying a new observation. In spite of that, a straightforward implementation of TBMATAN, even when accomplishing these complexity bounds, will not yield accurate results, specially

for large datasets. This is due to the fact that the calculations that need to be done in order to classify a new observation include the computation of a determinant (in equation 21) 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 matrixes 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 [21], 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 $\mathcal{O}(\#C \cdot n^3)$, and hence makes the practical application of the algorithms difficult in situations where it is required to classify a large set of unclassified data.

Faced with this problem, we analyzed what makes the determinant being ill-conditioned and concluded that it is due to the $W_{u,v}$ factors given by equation 26. The factor $W_{u,v}$ could be interpreted as "how much the dataset $\mathcal{D}$ has changed our 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 26 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 a $K = 5$. The transformation works as depicted in figure 2 and described in detail by the following

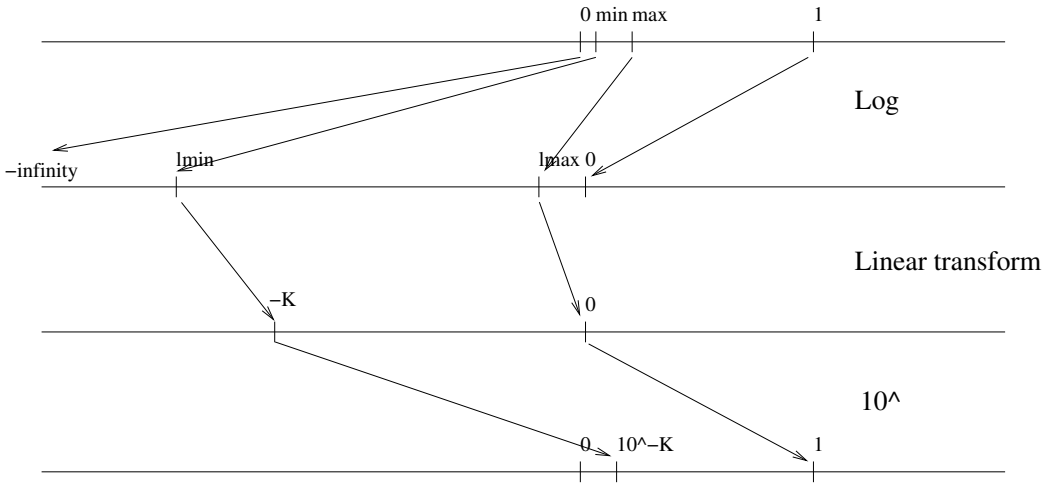


Figure 2: Transformation of weights for SSTBMATAN

equations:

$$lmax = \log_{10} \max_{\substack{u \in V \\ v \in V \\ u \neq v}} W_{u,v} \tag{29}$$

$$lmin = \log_{10} \min_{\substack{u \in V \\ v \in V \\ u \neq v}} W_{u,v} \tag{30}$$

$$a = \begin{cases} \frac{K}{lmax - lmin} & lmax - lmin > K \\ 1 & otherwise \end{cases} \tag{31}$$

$$b = -K - a * lmin \tag{32}$$

$$\widetilde{W}_{u,v} = 10^{a \log_{10}(W_{u,v}) + b} \tag{33}$$

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 our 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 our 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. The learning process for TBMATAN does not need high precision arithmetics. The learning time complexity for TBMATAN, $\mathcal{O}(N \cdot n^2)$, is the same as the one for TAN. In spite of that, in practice, TAN learning time would be somewhat larger because the learning stage for TBMATAN (calculating every $N_{v,u,C}(j, i, c)$) is only the first step of the TAN learning process.

5 Empirical results

We tested four algorithms over 16 datasets from the Irvine repository [1]. The dataset characteristics are described in Table 1. To discretize continuous attributes we used equal frequency discretization with 5 intervals. For each dataset and algorithm we tested both accuracy and *LogScore*. *LogScore* is calculated by adding the minus logarithm of the probability assigned by the classifier to the correct class and gives an idea of how well the classifier is estimating probabilities (the smaller the score the better the result). If we name our test set $\mathcal{D}'$ we have

$$LogScore(M, \mathcal{D}') = \sum_{(S,S_C)\in\mathcal{D}'} -\log(P(S_C|S, M)) \tag{34}$$

The focus of most research in machine learning algorithms is on improving accuracy. There are many cases in the real life application of these algorithms, where not only accuracy is interesting, but also the quality of the probability estimations for each class is very important. Two common examples are the selection of people likely to respond to mailing campaigns and when the results of the learning process should lead to decisions being

taken by a human who has to have an estimate of the decision risk. That is why we included *LogScore* in our testing. For the evaluation of both error rate and *LogScore* we used 10 fold cross validation. We tested the algorithm with the 10%, 50% and 100% of the learning data for each fold, in order to get an idea of the influence of the amount of data in the behaviours of both error rate and *LogScore* for the algorithm.

The error rates appear in Tables 2,4,6 with the best method for each dataset boldfaced. *LogScore*'s appear in Tables 3,5,7. The columns of the tables are the induction methods and the rows are the datasets. The meaning of the column headers are:

- IndifferentNB is the Naive Bayes algorithm (implemented as in [4]).
- SSTBMATAN is the method described in section 4.
- TAN+MS is the maximum likelihood TAN induction using the *multinomial sampling approach* [3].
- TBMATAN, is the method described in section 3.5.

TBMATAN classification times are very large for datasets with a large number of instances. For datasets over 5000 instances we have skipped the execution of TBMATAN. This is represented as a - sign in the corresponding table entries. We have skipped those datasets also when drawing comparison graphs for TBMATAN.

Dataset	Attributes	Instances	Classes	Missing
ADULT	14	48842	2	some
BREAST	10	699	2	16
CAR	6	1728	4	no
CHESS	36	3196	2	no
CLEVE	13	303	2	some
CRX	15	690	2	few
FLARE	10	323	4	no
GLASS	10	214	2	none
HEP	19	155	2	some
IRIS	4	150	3	none
LETTER	16	20000	26	none
MUSHROOM	22	8124	2	some
NURSERY	8	12960	5	no
PIMA	8	768	2	no
SOYBEAN	35	316	19	some
VOTES	16	435	2	few

Table 1: Datasets information

Dataset	IndifferentNB	SSTBMATAN	TAN+MS	TBMATAN
ADULT	19.08 ± 0.82	**16.69 ± 0.72**	17.63 ± 0.82	-
BREAST	**4.67 ± 1.31**	12.84 ± 1.69	20.88 ± 2.05	15.67 ± 1.78
CAR	19.75 ± 1.84	**18.87 ± 1.51**	19.03 ± 2.08	19.85 ± 1.54
CHESS	14.92 ± 1.55	10.61 ± 1.51	8.81 ± 1.43	**8.50 ± 1.05**
CLEVE	**22.59 ± 2.26**	28.37 ± 2.32	31.64 ± 2.69	28.37 ± 2.32
CRX	**16.33 ± 1.83**	18.26 ± 1.77	33.28 ± 1.99	19.20 ± 2.26
FLARE	28.14 ± 2.12	**23.46 ± 1.94**	24.17 ± 1.77	23.84 ± 1.89
GLASS	40.12 ± 3.37	**38.27 ± 3.56**	55.13 ± 3.99	39.60 ± 3.67
HEP	34.31 ± 3.94	**32.63 ± 3.91**	49.13 ± 3.69	**32.63 ± 3.91**
IRIS	**16.31 ± 2.78**	24.80 ± 3.24	27.95 ± 3.67	26.13 ± 3.60
LETTER	33.75 ± 0.89	**23.81 ± 1.04**	40.89 ± 1.69	-
MUSHROOM	7.14 ± 1.14	0.30 ± 0.34	**0.30 ± 0.37**	-
NURSERY	10.05 ± 0.90	**7.88 ± 1.09**	8.82 ± 1.08	-
PIMA	**29.00 ± 2.29**	31.88 ± 1.99	33.85 ± 1.85	33.26 ± 1.61
SOYBEAN	35.32 ± 2.27	**25.39 ± 2.48**	29.83 ± 2.36	25.45 ± 2.27
VOTES	12.38 ± 1.73	**10.17 ± 2.12**	10.51 ± 2.01	10.17 ± 1.96

Table 2: Averages and standard deviations of error rate using 10% of the learning data

Dataset	IndifferentNB	SSTBMATAN	TAN+MS	TBMATAN
ADULT	682.41 ± 5.16	**523.85 ± 4.38**	626.09 ± 5.47	-
BREAST	**4.54 ± 1.75**	9.91 ± 1.64	24.31 ± 2.62	15.17 ± 1.92
CAR	39.14 ± 2.29	**34.82 ± 2.34**	39.42 ± 2.76	37.10 ± 2.49
CHESS	46.53 ± 2.50	35.57 ± 2.09	30.88 ± 1.96	**29.43 ± 1.58**
CLEVE	**7.99 ± 1.67**	9.94 ± 1.91	14.25 ± 1.85	10.06 ± 1.95
CRX	**13.94 ± 2.17**	13.98 ± 1.96	42.37 ± 3.00	16.46 ± 2.31
FLARE	46.21 ± 2.91	**45.12 ± 3.18**	53.95 ± 3.44	46.79 ± 3.33
GLASS	**10.71 ± 1.95**	12.44 ± 2.71	27.25 ± 3.83	14.76 ± 2.88
HEP	2.45 ± 1.31	1.82 ± 1.05	4.15 ± 1.46	**1.82 ± 1.05**
IRIS	**2.35 ± 0.89**	3.45 ± 0.95	4.00 ± 1.11	3.45 ± 0.95
LETTER	**1284.49 ± 5.90**	1361.88 ± 7.94	5076.85 ± 26.18	-
MUSHROOM	100.85 ± 4.67	2.68 ± 1.17	**1.71 ± 1.25**	-
NURSERY	167.10 ± 3.07	**112.84 ± 2.85**	117.21 ± 3.02	-
PIMA	**21.40 ± 1.68**	26.86 ± 2.21	33.88 ± 2.66	29.90 ± 2.47
SOYBEAN	80.80 ± 4.72	65.67 ± 5.31	81.62 ± 4.65	**65.02 ± 5.36**
VOTES	11.44 ± 2.19	**4.66 ± 1.52**	5.77 ± 1.81	4.66 ± 1.51

Table 3: Averages and standard deviations of *LogScore* using 10% of the learning data

Dataset	IndifferentNB	SSTBMATAN	TAN+MS	TBMATAN
ADULT	18.67 ± 0.85	**16.33 ± 0.67**	16.78 ± 0.75	-
BREAST	**4.07 ± 1.29**	5.73 ± 1.50	8.94 ± 1.66	7.64 ± 1.93
CAR	15.64 ± 1.74	**8.04 ± 1.53**	9.04 ± 1.47	8.32 ± 1.58
CHESS	12.48 ± 1.04	9.15 ± 1.10	8.13 ± 1.21	**7.81 ± 1.21**
CLEVE	**20.19 ± 2.21**	22.35 ± 2.25	25.41 ± 2.08	24.80 ± 2.36
CRX	15.22 ± 2.05	**14.93 ± 2.24**	20.52 ± 2.27	15.52 ± 2.22
FLARE	26.22 ± 1.98	**20.67 ± 1.51**	20.79 ± 1.76	20.73 ± 1.92
GLASS	25.97 ± 3.13	**25.21 ± 3.14**	36.79 ± 3.40	25.86 ± 3.25
HEP	23.53 ± 3.15	**21.51 ± 2.79**	29.71 ± 3.14	22.34 ± 2.77
IRIS	14.97 ± 2.28	15.92 ± 3.34	**14.94 ± 3.08**	15.92 ± 3.34
LETTER	27.40 ± 0.75	**12.18 ± 0.76**	15.16 ± 0.91	-
MUSHROOM	5.43 ± 1.03	0.22 ± 0.27	**0.15 ± 0.22**	-
NURSERY	9.96 ± 0.80	**6.92 ± 0.83**	7.25 ± 0.87	-
PIMA	26.42 ± 1.71	**24.94 ± 1.82**	26.47 ± 1.51	26.58 ± 1.69
SOYBEAN	13.34 ± 1.61	**7.26 ± 1.44**	11.74 ± 1.98	7.84 ± 1.66
VOTES	11.83 ± 1.64	8.22 ± 1.84	9.07 ± 2.00	**7.93 ± 1.90**

Table 4: Averages and standard deviations of error rate using 50% of the learning data

Dataset	IndifferentNB	SSTBMATAN	TAN+MS	TBMATAN
ADULT	667.22 ± 4.35	**497.55 ± 3.54**	536.37 ± 3.84	-
BREAST	7.47 ± 2.51	**5.85 ± 2.04**	10.46 ± 1.99	10.18 ± 2.48
CAR	26.13 ± 2.10	16.27 ± 1.95	16.88 ± 2.03	**16.10 ± 1.96**
CHESS	40.45 ± 1.85	31.33 ± 1.88	26.17 ± 1.81	**26.12 ± 1.85**
CLEVE	6.51 ± 1.47	**6.28 ± 1.58**	9.14 ± 2.03	6.94 ± 1.70
CRX	13.20 ± 2.19	**10.98 ± 2.32**	24.41 ± 3.13	12.30 ± 2.45
FLARE	42.45 ± 2.86	**37.65 ± 3.21**	41.34 ± 3.36	39.23 ± 3.27
GLASS	6.16 ± 1.44	**5.62 ± 1.38**	15.10 ± 2.36	9.47 ± 1.96
HEP	1.50 ± 1.14	**1.13 ± 0.82**	2.34 ± 1.16	1.29 ± 0.86
IRIS	**1.54 ± 0.88**	1.56 ± 0.97	1.60 ± 1.07	1.66 ± 1.04
LETTER	1046.72 ± 5.38	**425.20 ± 5.34**	1082.94 ± 7.63	-
MUSHROOM	59.90 ± 3.87	1.34 ± 0.98	**0.21 ± 0.69**	-
NURSERY	150.01 ± 2.88	99.09 ± 2.46	**92.48 ± 2.58**	-
PIMA	**18.59 ± 1.24**	18.72 ± 1.74	20.55 ± 1.99	20.38 ± 1.93
SOYBEAN	25.63 ± 3.11	6.22 ± 1.63	14.58 ± 2.72	**5.07 ± 1.32**
VOTES	11.57 ± 2.32	**3.48 ± 1.76**	3.95 ± 1.80	3.69 ± 1.81

Table 5: Averages and standard deviations of *LogScore* using 50% of the learning data

Dataset	IndifferentNB	SSTBMATAN	TAN+MS	TBMATAN
ADULT	18.55 ± 0.82	**16.34 ± 0.64**	16.50 ± 0.83	-
BREAST	**3.93 ± 1.34**	5.05 ± 1.25	6.15 ± 1.48	5.48 ± 1.36
CAR	14.29 ± 1.53	6.66 ± 1.43	6.03 ± 1.26	**6.02 ± 1.37**
CHESS	12.79 ± 1.30	9.81 ± 0.79	**7.53 ± 1.12**	8.08 ± 1.12
CLEVE	**19.19 ± 2.14**	20.82 ± 2.24	26.30 ± 2.05	21.66 ± 2.31
CRX	15.14 ± 2.01	14.54 ± 2.03	19.06 ± 2.13	**13.52 ± 1.89**
FLARE	25.17 ± 2.08	**20.23 ± 1.60**	20.77 ± 1.66	20.38 ± 1.64
GLASS	22.01 ± 3.31	**17.37 ± 2.59**	28.46 ± 2.65	21.21 ± 2.46
HEP	22.62 ± 3.01	**18.65 ± 2.75**	22.08 ± 3.06	**18.65 ± 2.75**
IRIS	15.04 ± 2.18	**11.19 ± 2.17**	13.51 ± 2.71	12.30 ± 2.40
LETTER	26.40 ± 0.78	**9.74 ± 0.76**	12.16 ± 0.79	-
MUSHROOM	4.66 ± 0.90	0.21 ± 0.28	**0.12 ± 0.05**	-
NURSERY	9.71 ± 0.76	7.02 ± 0.71	**6.72 ± 0.78**	-
PIMA	26.26 ± 2.09	**23.38 ± 1.36**	23.43 ± 1.47	24.22 ± 1.46
SOYBEAN	11.20 ± 1.42	**6.36 ± 1.38**	7.80 ± 1.58	6.44 ± 1.48
VOTES	11.76 ± 1.51	8.10 ± 1.96	7.85 ± 2.01	**7.58 ± 1.99**

Table 6: Averages and standard deviations of error rate using 100% of the learning data

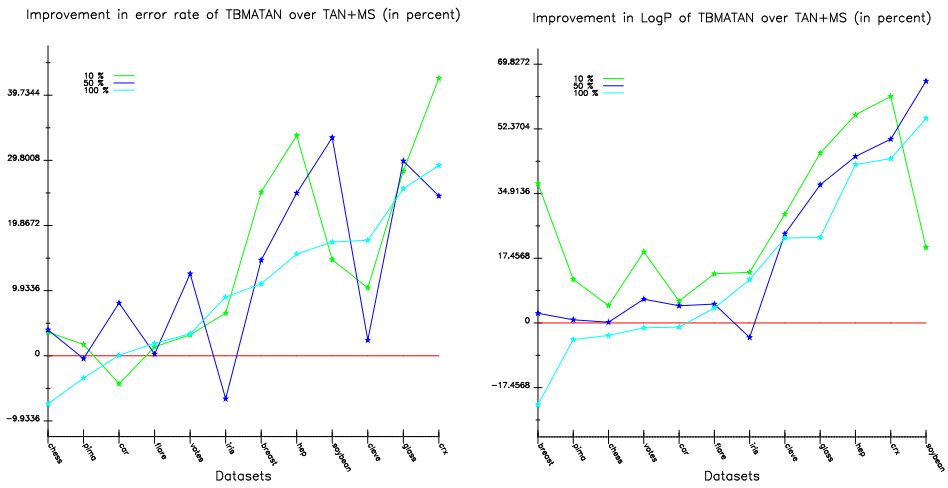
Dataset	IndifferentNB	SSTBMATAN	TAN+MS	TBMATAN
ADULT	666.75 ± 4.16	**493.66 ± 3.41**	513.33 ± 4.30	-
BREAST	8.04 ± 2.66	**5.46 ± 1.93**	6.54 ± 2.14	7.98 ± 2.42
CAR	24.68 ± 1.93	14.38 ± 1.49	**13.92 ± 1.40**	14.08 ± 1.46
CHESS	40.64 ± 1.74	31.15 ± 1.64	**25.26 ± 1.56**	26.11 ± 1.49
CLEVE	6.44 ± 1.49	**5.68 ± 1.41**	8.25 ± 1.42	6.36 ± 1.59
CRX	12.59 ± 2.12	**10.57 ± 2.13**	20.36 ± 2.92	11.33 ± 2.13
FLARE	41.23 ± 2.78	**33.71 ± 2.91**	36.55 ± 2.93	35.09 ± 2.94
GLASS	4.41 ± 1.11	**3.99 ± 1.42**	10.22 ± 2.11	7.86 ± 1.96
HEP	1.84 ± 1.28	1.13 ± 0.97	1.85 ± 1.45	**1.06 ± 0.95**
IRIS	1.44 ± 0.95	**1.08 ± 0.83**	1.29 ± 1.00	1.14 ± 0.85
LETTER	999.94 ± 5.59	**299.40 ± 4.91**	640.74 ± 6.35	-
MUSHROOM	48.17 ± 3.57	1.03 ± 0.92	**0.00 ± 0.02**	-
NURSERY	148.09 ± 2.84	97.56 ± 2.37	**88.98 ± 2.34**	-
PIMA	18.00 ± 1.21	**17.17 ± 1.50**	17.25 ± 1.49	18.02 ± 1.50
SOYBEAN	24.06 ± 3.41	4.76 ± 1.62	7.56 ± 2.18	**3.39 ± 1.38**
VOTES	11.79 ± 2.37	**3.25 ± 1.80**	3.39 ± 1.79	3.43 ± 1.87

Table 7: Averages and standard deviations of *LogScore* using 100% of the learning data

5.1 Interpretation of the results

We can see in figures 3(a) and 3(b) that in most cases TBMATAN improves both accuracy and *LogScore* with respect to TAN+MS. The average improvement is around 10% for error rate and slightly higher for *LogScore*. The percentage of improvement is higher as we reduce the amount of learning data. This is understandable, because it is reasonable to think that if we have enough data, the posterior is likely to be concentrated around the tree learned by TAN+MS.

Comparing SSTBMATAN and TBMATAN by looking at figure 4(a) and 4(b) we see that SSTBMATAN is even slightly better than TBMATAN for many datasets, so we can accept that the approximation introduced in section



(a) Error rate

(b) Log score

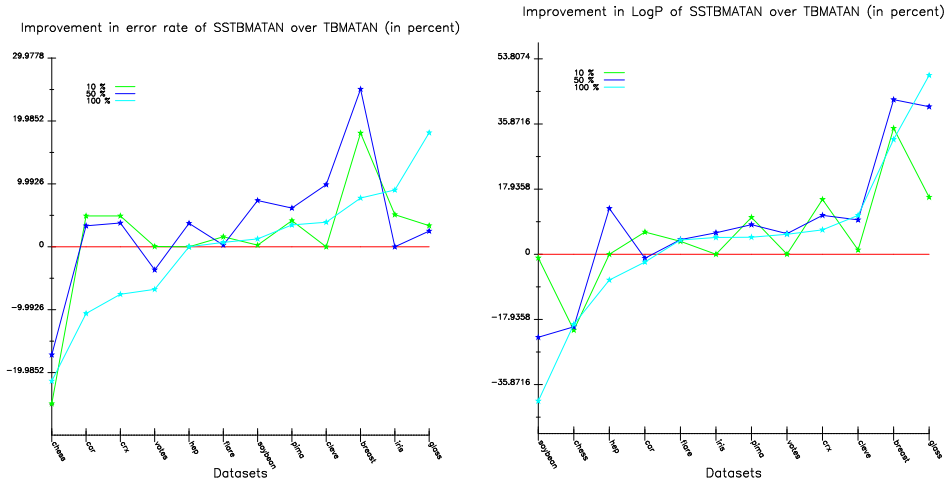
Figure 3: Comparison of TBMATAN and TAN+MS

4 is good for datasets of this size. Finally, if we compare SSTBMATAN and TAN+MS by means of figures 5(a) and 5(b), we can see that its relative behaviour is very similar to the one of TBMATAN with TAN+MS. The only exception is the MUSHROOM dataset, where there is a huge relative difference. This can easily be understood by looking at the tables. TAN+MS is able to learn the dataset almost exactly. Even when SSTBMATAN only has an error rate of 0.21%, the relative difference turns out to be very high.

6 Conclusions and future work

We have introduced TBMATAN a classifier based on TAN, decomposable distributions and bayesian model averaging. We have seen that its implementation leads to the calculation of ill-conditioned determinants and have proposed to use an approximated implementation: SSTBMATAN.

SSTBMATAN is, to the best of our knowledge, the most accurate classifier reported with a learning time linear on the number of observations of the dataset. The accuracy increase comes at the price of increasing the classification time, making it cubic on the number of attributes. The algorithm is anytime and incremental: as long as the dataset observations are processed randomly, we can stop the learning stage anytime we need, perform some classifications and then continue learning at the only (obvious) cost of the



(a) Error rate

(b) Log score

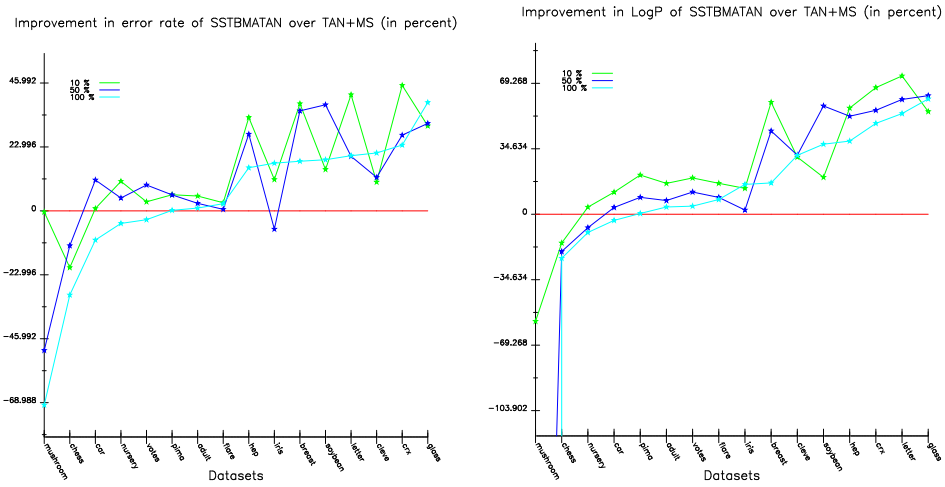
Figure 4: Comparison of SSTBMATAN and TBMATAN

lower accuracy of the classifications performed in the middle of the learning process. These characteristics make the algorithm very suitable for huge databases.

If we were able to determine beforehand the impact of working with a sample in the accuracy of the predictions, in the line of Chernoff-Hoeffding bounds, we could speed up considerably the algorithm by accepting a small deviation in accuracy. In this sense, finding a result in the line of [10] remains as future work.

Being able to calculate some measure of the concentration of the posterior distribution around the TAN learned by TAN+MS (that is, some sort of "variance") will probably allow us to determine beforehand whether TBMATAN will provide significant improvement over TAN+MS in a dataset.

Finally, we think that all of the classifiers reviewed in [7] that are based on the Chow and Liu algorithm [5] can benefit from an improvement similar to the one seen here by the use of decomposable distributions and bayesian model averaging. Formalizing the development for these classifiers and performing the empirical tests remains as future work.



(a) Error rate

(b) Log score

Figure 5: Comparison of SSTBMATAN and TAN+MS

A Preliminaries

In this appendix we introduce three results that will be needed in the further development and then in appendix B we prove the results in sections 3.3 and 3.4.

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.

$$A = \begin{bmatrix} \deg v_1 & -a_{1,2} & -a_{1,3} & \dots & a_{1,n} \\ -a_{2,1} & \deg v_2 & -a_{2,3} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ -a_{n,1} & -a_{n,2} & -a_{n,3} & \dots & \deg v_n \end{bmatrix} \tag{35}$$

Proof: See [22, 20].

□

A.2 The matrix tree theorem for decomposable distributions

Let P(E) be a distribution over spanning tree structures defined by equations 10 and 11. 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:

$$\overline{Q}_{u,v}(\boldsymbol{\beta}) = \overline{Q}_{v,u}(\boldsymbol{\beta}) = \begin{cases} -\beta_{u,v} & 1 \leq u < v \leq n \\ \sum_{v'=1}^{n} \beta_{v',v} & 1 \leq u = v \leq n \end{cases} \tag{36}$$

Proof: See [17].

□

A.3 A useful result about Dirichlet distributions

Let $D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r)$ be a Dirichlet distribution defined as in equation 2. We have that:

$$\begin{aligned} &D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r) \prod_{i=1}^{r} {\theta_i}^{n_i} = \\ &\frac{\Gamma(\sum_{i=1}^{r} n'_i)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i + n_i)}{\Gamma(\sum_{i=1}^{r} n'_i + n_i)} D(\theta_1, \ldots, \theta_r; n'_1 + n_1, \ldots, n'_r + n_r) \end{aligned} \tag{37}$$

and since the Dirichlet distribution is normalized you have that

$$\int_{\theta_1} \cdots \int_{\ldots,\theta_r} D(\theta_1, \ldots, \theta_r; n'_1, \ldots, n'_r) \prod_{i=1}^{r} {\theta_i}^{n_i} = \frac{\Gamma(\sum_{i=1}^{r} n'_i)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i + n_i)}{\Gamma(\sum_{i=1}^{r} n'_i + n_i)} \tag{38}$$

Proof: By expanding the Dirichlet distribution by means of its definition in equation 2, grouping again into a Dirichlet and considering that the Dirichlet distribution is normalized distribution and hence integrates to one, we have

that:

$$\int_{\theta_1,\ldots,\theta_r} \cdots \int D(\theta_1,\ldots,\theta_r; n'_1,\ldots,n'_r) \prod_{i=1}^{r} {\theta_i}^{n_i} \tag{39}$$

$$= \int_{\theta_1,\ldots,\theta_r} \cdots \int \frac{\Gamma(\sum_{i=1}^{r} n'_i)}{\prod_{i=1}^{r} \Gamma(n'_i)} \prod_{i=1}^{r} {\theta_i}^{n'_i+n_i-1} \tag{40}$$

$$= \int_{\theta_1,\ldots,\theta_r} \cdots \int \frac{\Gamma(\sum_{i=1}^{r} n'_i)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^{r} n'_i+n_i)} D(\theta_1,\ldots,\theta_r; n'_1+n_1,\ldots,n'_r+n_r) \tag{41}$$

$$= \frac{\Gamma(\sum_{i=1}^{r} n'_i)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^{r} n'_i+n_i)} \int_{\theta_1,\ldots,\theta_r} \cdots \int D(\theta_1,\ldots,\theta_r; n'_1+n_1,\ldots,n'_r+n_r) \tag{42}$$

$$= \frac{\Gamma(\sum_{i=1}^{r} n'_i)}{\prod_{i=1}^{r} \Gamma(n'_i)} \frac{\prod_{i=1}^{r} \Gamma(n'_i+n_i)}{\Gamma(\sum_{i=1}^{r} n'_i+n_i)} \tag{43}$$

□

B Detailed development

B.1 Calculating probabilities with decomposable distributions

We need to calculate

$$P(S, S_C|\xi) = \int_{M\in\mathcal{M}} P(S, S_C|M, \xi) P(M|\xi) \tag{44}$$

that is:

$$p(S, S_C|\xi) = \sum_{E\in\mathcal{E}} \int_{\Theta_{\overline{E}^*}} \cdots \int P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*}) P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) d\Theta_{\overline{E}^*} \tag{45}$$

We have to develop $P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*})$ and $P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi)$ multiply them and integrate over $\Theta_{\overline{E}^*}$ to get the desired result.

B.1.1 Calculating $P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*})$

It is determined by the expansion of equation 1 taking into account the TAN structure.

$$P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*}) = \theta_C(S_C)\ \theta_{\rho_{\overline{E}}|C}(S_{\rho_{\overline{E}}}, S_C) \prod_{u,v\in\overline{E}} \theta_{v|u,C}(S_v, S_u, S_C) \tag{46}$$

B.1.2 Calculating $P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi)$

The prior $P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi)$ can be expanded from equations 9, 10, 12, 16, 17 and 18 into

$$\begin{aligned} P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) = \frac{1}{Z_\beta} & \prod_{u,v\in\overline{E}} \beta_{u,v} \\ & \times D(\theta_C(.); N'_C(.)) \\ & \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(., c); N'_{\rho_{\overline{E}},C}(., c)) \\ & \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i=1}^{\#u} D(\theta_{v|u,C}(., i, c); N'_{v,u,C}(., i, c)) \end{aligned} \tag{47}$$

B.1.3 Integrating over $\Theta_{\overline{E}^*}$

We want to calculate

$$\int_{\Theta_{\overline{E}^*}} \cdots \int P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*}) P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) d\Theta_{\overline{E}^*} \tag{48}$$

We define:

$$\delta_C^{S,S_C}(c) = \begin{cases} 1 & c = S_C \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

$$\delta_{i,C}^{S,S_C}(k, c) = \begin{cases} 1 & k = S_i \wedge c = S_C \\ 0 & \text{otherwise} \end{cases} \tag{50}$$

$$\delta_{i,j,C}^{S,S_C}(k, l, c) = \begin{cases} 1 & k = S_i \wedge l = S_j \wedge c = S_C \\ 0 & \text{otherwise} \end{cases} \tag{51}$$

It is easy to see that:

$$\sum_{j\in v} \delta_{v,u,C}^{S,S_C}(j, i, c) = \delta_{u,C}^{S,S_C}(i, c) \tag{52}$$

$$\sum_{i \in u} \delta_{u,C}^{S,S_C}(i,c) = \delta_C^{S,S_C}(c) \tag{53}$$

$$\sum_{c \in C} \delta_C^{S,S_C}(c) = 1 \tag{54}$$

We can use this notation to expand the product $P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*})P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi)$ by substituting equations 46 and 47 giving:

$$\begin{aligned}
P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*})P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) &= \frac{1}{Z_\beta} \prod_{u,v \in \overline{E}} \beta_{u,v} \\
&\times D(\theta_C(.); N'_C(.)) \prod_{c \in C} \theta_C(c)^{\delta_C^{S,S_C}(c)} \\
&\times \prod_{c \in C} \left[D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \prod_{i=1}^{\#\rho_{\overline{E}}} \theta_{\rho_{\overline{E}}|C}(i,c)^{\delta^{S,S_C}_{\rho_{\overline{E}},C}(i,c)} \right] \\
&\times \prod_{c \in C} \prod_{u,v \in \overline{E}} \prod_{i=1}^{\#u} \left[D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \prod_{j=1}^{\#v} \theta_{v|u,C}(j,i,c)^{\delta^{S,S_C}_{v,u,C}(j,i,c)} \right]
\end{aligned} \tag{55}$$

By analyzing equation 55 we can see that the integral in equation 48 can be calculated by applying the result in equation 38 three times. This gives:

$$\begin{aligned}
\int_{\Theta_{\overline{E}^*}} \cdots \int P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*})P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) d\Theta_{\overline{E}^*} &= \frac{1}{Z_\beta} \prod_{u,v \in \overline{E}} \beta_{u,v} \\
&\times \frac{\Gamma(\sum_{c \in C} N'_C(c))}{\prod_{c \in C} \Gamma(N'_C(c))} \frac{\prod_{c \in C} \Gamma(N'_C(c) + \delta_C^{S,S_C}(c))}{\Gamma(\sum_{c \in C} N'_C(c) + \delta_C^{S,S_C}(c))} \\
&\times \prod_{c \in C} \left[\frac{\Gamma(\sum_{i=1}^{\#\rho_{\overline{E}}} N'_{\rho_{\overline{E}},C}(i,c))}{\prod_{i=1}^{\#\rho_{\overline{E}}} \Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \frac{\prod_{i=1}^{\#\rho_{\overline{E}}} \Gamma(N'_{\rho_{\overline{E}},C}(i,c) + \delta^{S,S_c}_{\rho_{\overline{E}},C}(i,c))}{\Gamma(\sum_{i=1}^{\#\rho_{\overline{E}}} N'_{\rho_{\overline{E}},C}(i,c) + \delta^{S,S_c}_{\rho_{\overline{E}},C}(i,c))} \right] \\
&\times \prod_{c \in C} \prod_{u,v \in \overline{E}} \prod_{i=1}^{\#u} \left[\frac{\Gamma(\sum_{j=1}^{\#v} N'_{v,u,C}(j,i,c))}{\prod_{j=1}^{\#v} \Gamma(N'_{v,u,C}(j,i,c))} \frac{\prod_{i=1}^{\#v} \Gamma(N'_{v,u,C}(j,i,c) + \delta^{S,S_c}_{v,u,C}(j,i,c))}{\Gamma(\sum_{i=1}^{\#v} N'_{v,u,C}(j,i,c) + \delta^{S,S_c}_{v,u,C}(j,i,c))} \right]
\end{aligned} \tag{56}$$

This expression can be simplified by applying equations 13,14,15,52,53 and 54 and reorganizing:

$$\begin{aligned}\int_{\Theta_{\overline{E}^*}} \cdots \int P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*}) P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) d\Theta_{\overline{E}^*} &= \frac{1}{Z_\beta} \prod_{u,v\in\overline{E}} \beta_{u,v} \\ &\times \frac{\Gamma(N')}{\Gamma(N'+1)} \\ &\times \prod_{-} c \in C \prod_{i=1}^{\#\rho_{\overline{E}}} \frac{\Gamma(N'_{\rho_{\overline{E}},C}(i,c) + \delta^{S,S_c}_{\rho_{\overline{E}},C}(i,c))}{\Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \\ &\times \prod_{u,v\in\overline{E}} \prod_{c\in C} \prod_{i=1}^{\#u} \left[\frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'_{u,C}(i,c) + \delta^{S,S_c}_{u,C}(i,c))} \prod_{i=1}^{\#v} \frac{\Gamma(N'_{v,u,C}(j,i,c) + \delta^{S,S_c}_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))} \right]\end{aligned} \tag{57}$$

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

$$\begin{aligned}\int_{\Theta_{\overline{E}^*}} \cdots \int P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*}) P(\overline{E}^*, \Theta_{\overline{E}^*}\xi) d\Theta_{\overline{E}^*} &= \frac{1}{Z_\beta} \prod_{u,v\in\overline{E}} \beta_{u,v} \\ &\times \frac{1}{N'} \\ &\times N'_{\rho_{\overline{E}},C}(S_{\rho_{\overline{E}}}, S_C) \\ &\times \prod_{u,v\in\overline{E}} \left[\frac{N'_{v,u,C}(S_v, S_u, S_C)}{N'_{u,C}(S_u, S_C)} \right]\end{aligned} \tag{58}$$

B.1.4 Adding over tree structures

We have to calculate

$$p(S|D,\xi) = \sum_{E\in\mathcal{E}} \int_{\Theta_{\overline{E}^*}} \cdots \int P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*}) P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) d\Theta_{\overline{E}^*} \tag{59}$$

Defining h_0^{S,S_C} and $h_{u,v}^{S,S_C}$ as in equations 22 and 23 it is easy to see that multiplying and dividing in equation 58 by the factor:

$$\prod_{v\in V-\{\rho_{\overline{E}}\}} N'_{v,C}(S_v, S_C) \tag{60}$$

and rearranging we get:

$$\int_{\Theta_{\overline{E}^*}} \cdots \int P(S, S_C|\overline{E}^*, \Theta_{\overline{E}^*}) P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) d\Theta_{\overline{E}^*} = h_0^{S,S_C} \prod_{u,v \in \overline{E}} \left(\beta_{u,v} h_{u,v}^{S,S_C}\right) \quad (61)$$

And then calculating the summation over structures using the matrix tree theorem for decomposable distributions gives the desired result.

$$P(S, S_C|\xi) = h_0^{S,S_C} |Q(\boldsymbol{\beta}\ \mathbf{h}^{S,S_C})| \quad (62)$$

B.2 Learning with decomposable distributions

We want to calculate $P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi)$. Using Bayes rule we get:

$$P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = \frac{P(\overline{E}^*, \Theta^*_{\overline{E}}|\xi) P(\mathcal{D}|\overline{E}^*, \Theta_{\overline{E}^*}, \xi)}{Z_{\mathcal{D}}} \quad (63)$$

The prior $P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi)$ comes given by equation 47. $P(\mathcal{D}|\overline{E}^*, \Theta_{\overline{E}^*}, \xi)$ is the probability that the model generates the data in $\mathcal{D}$. Since $\mathcal{D}$ contains independent identically distributed observations, we have that

$$\begin{aligned} P(\mathcal{D}|\overline{E}^*, \Theta_{\overline{E}^*}, \xi) = & \prod_{c \in C} \theta_C(c)^{N_C(c)} \\ & \times \prod_{c \in C} \prod_{i=1}^{\#\rho_{\overline{E}}} \theta_{\rho_{\overline{E}}|C}(i, c)^{N_{\rho_{\overline{E}},C}(i,c)} \\ & \times \prod_{c \in C} \prod_{u,v \in \overline{E}} \prod_{i=1}^{\#u} \prod_{j=1}^{\#v} \theta_{v|u,C}(j, i, c)^{N_{v,u,C}(j,i,c)} \end{aligned} \quad (64)$$

Substituting equations 47 and 64 into 63 we get

$$\begin{aligned} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \prod_{u,v \in \overline{E}} \beta_{u,v} \\ & \times D(\theta_C(.); N'_C(.)) \prod_{c \in C} \theta_C(c)^{N_C(c)} \\ & \times \prod_{c \in C} \left[D(\theta_{\rho_{\overline{E}}|C}(., c); N'_{\rho_{\overline{E}},C}(., c)) \prod_{i=1}^{\#\rho_{\overline{E}}} \theta_{\rho_{\overline{E}}|C}(i, c)^{N_{\rho_{\overline{E}},C}(i,c)} \right] \\ & \times \prod_{c \in C} \prod_{u,v \in \overline{E}} \prod_{i=1}^{\#u} \left[D(\theta_{v|u,C}(., i, c); N'_{v,u,C}(., i, c)) \prod_{j=1}^{\#v} \theta_{v|u,C}(j, i, c)^{N_{v,u,C}(j,i,c)} \right] \end{aligned} \quad (65)$$

Applying the result in equation 37 for all the Dirichlets we have that

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}} \prod_{u,v\in\overline{E}} \beta_{u,v} \\
& \times \frac{\Gamma(\sum_{c\in C} N'_C(c))}{\prod_{c\in C}\Gamma(N'_C(c))} \frac{\prod_{c\in C}\Gamma(N'_C(c)+N_C(c))}{\Gamma(\sum_{c\in C} N'_C(c)+N_C(c))} \\
& \times \prod_{c\in C}\left[\frac{\Gamma(\sum_{i=1}^{\#\rho_{\overline{E}}} N'_{\rho_{\overline{E}},C}(i,c))}{\prod_{i=1}^{\#\rho_{\overline{E}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \frac{\prod_{i=1}^{\#\rho_{\overline{E}}}\Gamma(N'_{\rho_{\overline{E}},C}(i,c)+N_{\rho_{\overline{E}},C}(i,c))}{\Gamma(\sum_{i=1}^{\#\rho_{\overline{E}}} N'_{\rho_{\overline{E}},C}(i,c)++N_{\rho_{\overline{E}},C}(i,c))}\right] \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i=1}^{\#u}\left[\frac{\Gamma(\sum_{j=1}^{\#v} N'_{v,u,C}(j,i,c))}{\prod_{j=1}^{\#v}\Gamma(N'_{v,u,C}(j,i,c))} \frac{\prod_{i=1}^{\#v}\Gamma(N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))}{\Gamma(\sum_{i=1}^{\#v} N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))}\right] \\
& \times D(\theta_C(.); N'_C(.)+N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i=1}^{\#u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))
\end{aligned}
\tag{66}
$$

This expression can be simplified by applying equations 13,14 and 15 (and similar ones for N) and reorganizing:

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)}\prod_{u,v\in\overline{E}}\beta_{u,v} \\
& \times \prod_{c\in C}\prod_{i=1}^{\#\rho_{\overline{E}}}\frac{\Gamma(N'_{\rho_{\overline{E}},C}(i,c)+N_{\rho_{\overline{E}},C}(i,c))}{\Gamma(N'_{\rho_{\overline{E}},C}(i,c))} \\
& \times \prod_{u,v\in\overline{E}}\prod_{c\in C}\prod_{i=1}^{\#u}\left[\frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'_{u,C}(i,c)+N_{u,C}(i,c))}\prod_{i=1}^{\#v}\frac{\Gamma(N'_{v,u,C}(j,i,c)+N_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))}\right] \\
& \times D(\theta_C(.); N'_C(.)+N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i=1}^{\#u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))
\end{aligned}
\tag{67}
$$

Defining $W_{u,v}$ as appears in equation 26, it is easy to see that multiplying and dividing in equation 67 by the factor:

$$
\prod_{v\in V-\{\rho_{\overline{E}}\}}\prod_{c\in C}\prod_{i=1}^{\#v}\frac{\Gamma(N'_{v,C}(i,c)+N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \tag{68}
$$

and rearranging we get:

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{Z_\beta}\frac{1}{Z_\mathcal{D}}\frac{\Gamma(N')}{\Gamma(N'+N)} \\
& \times \prod_{c\in C}\prod_{v\in V}\prod_{i=1}^{\#v}\frac{\Gamma(N'_{v,C}(i,c)+N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \\
& \times \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \\
& \times D(\theta_C(.); N'_C(.)+N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)+N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C}\prod_{u,v\in\overline{E}}\prod_{i=1}^{\#u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)+N_{v,u,C}(.,i,c))
\end{aligned}
\tag{69}
$$

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

$$\int_{M \in \mathcal{M}} P(M|\mathcal{D}, \xi) = \sum_{E \in \mathcal{E}} \int \cdots \int_{\Theta_{\overline{E}^*}} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = 1 \tag{70}$$

We need to integrate over the parameters, then sum over the tree structures and finally solve for $Z_{\mathcal{D}}$. The first step is easy, because Dirichlet distributions are normalized and integrate to 1 giving:

$$\begin{aligned} \int \cdots \int_{\Theta_{\overline{E}^*}} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) &= \frac{1}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \frac{\Gamma(N')}{\Gamma(N'+N)} \\ &\times \prod_{c \in C} \prod_{v \in V} \prod_{i=1}^{\#v} \frac{\Gamma(N'_{v,C}(i,c) + N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \\ &\times \prod_{u,v \in \overline{E}} W_{u,v} \beta_{u,v} \end{aligned} \tag{71}$$

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

$$\begin{aligned} \sum_{E \in \mathcal{E}} \int \cdots \int_{\Theta_{\overline{E}^*}} P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) &= \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{Z_\beta} \frac{1}{Z_{\mathcal{D}}} \frac{\Gamma(N')}{\Gamma(N'+N)} \\ &\times \prod_{c \in C} \prod_{v \in V} \prod_{i=1}^{\#v} \frac{\Gamma(N'_{v,C}(i,c) + N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} = 1 \end{aligned} \tag{72}$$

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

$$Z_{\mathcal{D}} = \frac{|Q(\boldsymbol{\beta}\mathbf{W})|}{|Q(\boldsymbol{\beta})|} \frac{\Gamma(N')}{\Gamma(N'+N)} \prod_{c \in C} \prod_{v \in V} \prod_{i=1}^{\#v} \frac{\Gamma(N'_{v,C}(i,c) + N_{v,C}(i,c))}{\Gamma(N'_{v,C}(i,c))} \tag{73}$$

Finally, substituting the result for $Z_{\mathcal{D}}$ in equation 69 we can see that the posterior is a decomposable distribution with the parameters updated as

given by equations 24, 25 and 26:

$$
\begin{aligned}
P(\overline{E}^*, \Theta_{\overline{E}^*}|\mathcal{D}, \xi) = & \frac{1}{|Q(\boldsymbol{\beta}\mathbf{W})|} \prod_{u,v\in\overline{E}} W_{u,v}\beta_{u,v} \\
& \times D(\theta_C(.); N'_C(.) + N_C(.)) \\
& \times \prod_{c\in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c) + N_{\rho_{\overline{E}},C}(.,c)) \\
& \times \prod_{c\in C} \prod_{u,v\in\overline{E}} \prod_{i=1}^{\#u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c) + N_{v,u,C}(.,i,c))
\end{aligned} \tag{74}
$$

References

[1] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases, 1998.

[2] Jesús Cerquides. Applying General Bayesian Techniques to Improve TAN Induction. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining, KDD99*, 1999.

[3] Jesús Cerquides. Applying General Bayesian Techniques to Improve TAN Induction. Long Version. Technical report, Ubilab, http://www.ubilab.org, 1999.

[4] Jesús Cerquides and Ramon López de Màntaras. The indifferent naive bayes classifier. Technical Report IIIA-2003-01, Institut d'Investigació en Intel.ligència Artificial, http://www.iiia.csic.es/∼mantaras/ReportIIIA-TR-2003-01.pdf, 2003.

[5] C. K. Chow and C. N. Liu. Aproximatting Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory*, 14:462–467, May 1968.

[6] Pedro Domingos and Michael Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29:103–130, 1997.

[7] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

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

[9] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging. Technical Report 9814, Department of Statistics. Colorado State University, 1998.

[10] Geoff Hulten and Pedro Domingos. Mining complex models from arbitrarily large databases in constant time. In *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[11] Eamonn J. Keogh and Michael Pazzani. Learning augmented bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Uncertainty 99: The Seventh International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 1999.

[12] Ron Kohavi and George H. John. Wrappers for Feature Subset Selection. *AI Journal*, Special Issue on Relevance.

[13] I. Kononenko. Semi-naive bayesian classifier. In Y. Kodratoff, editor, *Proc. Sixth European Working Session on Learning*, pages 206–219. Berlin: Springer-Verlang, 1991.

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

[15] P. Langley and S. Sage. Induction of selective bayesian classifiers. In R. López de Màntaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406. San Francisco,CA: Morgan Kaufmann, 1994.

[16] Pat Langley, Wayne Iba, and Kevin Thompson. An Analysis of Bayesian Classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223–228. AAAI Press and MIT Press, 1992.

[17] M. Meila and T. Jaakkola. Tractable bayesian learning of tree belief networks. Technical report, Carnegie Mellon University Robotics Institute, 2000.

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

[19] Michael Pazzani. Searching for dependencies in bayesian classifiers. In D. Fisher and H. Lenz, editors, *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 1995.

[20] Martin Rubey. Counting spanning trees. Diplomarbeit, 2000.

[21] Victor Shoup. NTL: A library for doing number theory. http://www.shoup.net/ntl.

[22] D. West. *Introduction to Graph Theory, Second Edition.* Prentice Hall, 1999.