Maximum a Posteriori Tree Augmented Naive Bayes Classifiers

Jesús Cerquides[1] and Ramon Lòpez de Màntaras[2]

[1] Departament de Matemàtica Aplicada i Anàlisi
Universitat de Barcelona
Gran Via 585, Barcelona 08007
[2] Institut d'Investigació en Intel.ligència Artificial (IIIA)
Consejo Superior de Investigaciones Científicas(CSIC)
Campus UAB, Bellaterra 08193

Abstract. Bayesian classifiers such 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 prove that under suitable conditions it is possible to efficiently compute the maximum a posterior TAN model. Furthermore, we prove that it is also possible to efficiently calculate a weighted set with the k maximum a posteriori TAN models. This allows efficient TAN ensemble learning and accounting for model uncertainty. These results can be used to construct two classifiers. Both classifiers have the advantage of allowing the introduction of prior knowledge about structure or parameters into the learning process. Empirical results show that both classifiers lead to an improvement in error rate and accuracy of the predicted class probabilities over established TAN based classifiers with equivalent complexity. Keywords: Bayesian networks, Bayesian network classifiers, Naive Bayes, decomposable distributions, Bayesian model averaging.

1 Introduction

Bayesian classifiers as *Naive Bayes* [11] or *Tree Augmented Naive Bayes* (TAN) [7] have shown excellent performance in spite of their simplicity and heavy underlying independence assumptions.

In our opinion, the TAN classifier, as presented in [7], has two weak points: not taking into account model uncertainty and lacking a theoretically well founded explanation for the use of softening of the induced model parameters (see section 2.2).

In [2] an alternative classifier based on empirical local Bayesian model averaging was proposed as a possible improvement for the first weak point. Furthermore, in [4] the fact that decomposable distributions over TANs allow the tractable calculation of the model averaging integral was used to construct sstb-matan, a classifier that takes into account model uncertainty in a theoretically well founded way and that provides improved classification accuracy.

In [2] an alternative softening is proposed with a theoretically more appealing derivation based on multinomial sampling.

In this paper both weak points are addressed. A computationally more efficient alternative to the first weak point is introduced and a well founded softening alternative is proposed that solves the second weak point. More concretely, we show that under the assumption of decomposable distributions over TANs, we can efficiently compute the TAN model with a maximum a posteriori (MAP) probability. This result allows the construction of MAPTAN, a classifier that provides a well founded alternative to the softening proposed in [7] and improves its error rate and the accuracy of the predicted class probabilities. Furthermore, we will also prove that under this assumption we can efficiently compute the k most probable TAN models and their relative probabilities. This result allows the construction of MAPTAN+BMA, a classifier that takes into consideration model uncertainty and improves in time complexity and accuracy over its equivalent presented in [2]. Furthermore, established TAN classifiers do not easily allow the introduction of prior knowledge into the learning process. Being able to compute MAP TAN structures means that we can easily incorporate prior knowledge, whenever our prior knowledge can be represented as a decomposable distribution over TANs.

These results point out the relevance of decomposable distribution over TANs, which are conjugate to TAN models, for the construction of classifiers based on the TAN model.

The paper is structured as follows. In section 2 Tree Augmented Naive Bayes is presented and the notation to be used in the rest of the paper is introduced. In section 3 we present decomposable distributions over TANs. In section 4 we give the main results for finding MAP TAN structures. In section 5 we construct MAPTAN and MAPTAN+BMA, using the previously stated results. In section 6 we provide the empirical results showing that our classifiers improve over established TAN classifiers. We end up with some conclusions and future work in section 7.

2 Tree Augmented Naive Bayes

Tree Augmented Naive Bayes (TAN) appears as a natural extension to the *Naive Bayes* classifier [10, 11, 6]. TAN models are a restricted family of Bayesian networks in which the class variable has no parents, each attribute has as parent the class variable and additionally there is a tree of dependencies between non-class attributes. An example of TAN model 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].

2.1 Formalization and Notation

The notation used in the paper is an effort to put together the different notations used in [2, 8, 7, 13] and some conventions in the machine learning literature.

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 use $\Omega = \{X_1, \ldots, X_m\}$ 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 use $V = \{A_1, \ldots, A_n\}$ for the set of attributes in a classified discrete domain that are not the class.

Given an attribute A, we will use $\#A$ 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 $S = (s_1, \ldots, s_n) \in A_1 \times \ldots \times A_n$. 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 use 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.

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.

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 use $\mathbf{U} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ 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 non-descendants given its parents in G. The second component of the pair, namely Θ, represents the set of parameters that quantifies the network. It contains a parameter $\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = P_B(x_i|\Pi_{x_i})$ 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 use $\overline{\Pi_i} = \mathbf{U} - \{\mathcal{X}_i\} - \Pi_i$. A Bayesian network defines a unique joint probability distribution over $\mathbf{U}$ given by

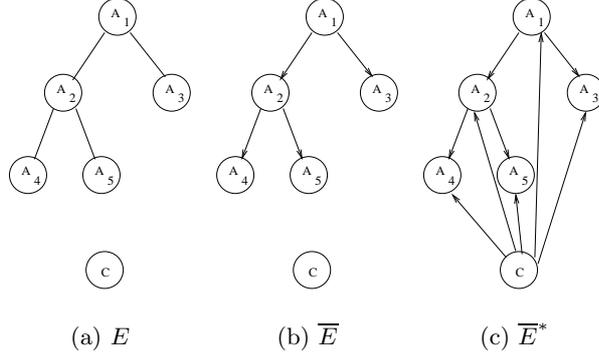$$P_B(x_1, \ldots, x_m) = \prod_{i=1}^{m} P_B(x_i|\Pi_{x_i}) = \prod_{i=1}^{m} \theta_{i|\Pi_i}(x_i|\Pi_{x_i}) \qquad (1)$$

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

Fig. 1. Notation for learning with trees

Learning with Trees Given a classified domain Ω_C we will use $\mathcal{E}$ 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 use $u, v \in E$ instead of $(\mathcal{A}_u, \mathcal{A}_v) \in E$ for simplicity. We will use $\overline{E}$ 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 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 use $\Theta_{\overline{E}^*}$ the set of parameters that quantify the Bayesian network $B = \langle \overline{E}^*, \Theta_{\overline{E}^*} \rangle$. 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 $\theta_C(c) = P(\mathcal{C} = c | B)$
$\boldsymbol{\theta}_{\rho_{\overline{E}}|C} = \{\theta_{\rho_{\overline{E}}|C}(i, c) | i \in A_{\rho_{\overline{E}}}, c \in C\}$ where
$\quad \theta_{\rho_{\overline{E}}|C}(i, c) = P(\mathcal{A}_{\rho_{\overline{E}}} = i | \mathcal{C} = c, B)$
For each $u, v \in \overline{E}$:
$\quad \boldsymbol{\theta}_{v|u,C} = \{\theta_{v|u,C}(j, i, c) | j \in A_v, i \in A_u, c \in C\}$ where
$\quad \theta_{v|u,C}(j, i, c) = P(\mathcal{A}_v = j | \mathcal{A}_u = i, \mathcal{C} = c, B)$.

2.2 Learning Maximum Likelihood TAN

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

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

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. To learn the maximum likelihood TAN we should use the following equation to compute the parameters.

$$\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = \frac{N_{i,\Pi_i}(x_i, \Pi_{x_i})}{N_{\Pi_i}(\Pi_{x_i})} \tag{3}$$

where $N_{i,\Pi_i}(x_i, \Pi_{x_i})$ stands for the number of times in the dataset that attribute i has value x_i and its parents have values Π_{x_i}. Equivalently, $N_{\Pi_i}(\Pi_{x_i}$ is the number of times in the dataset that the parents of attribute i have values Π_{x_i}.

It has been shown [7] that equation 3 leads to "overfitting" the model. Also in [7] Friedman et al. propose to use the parameters as given by

$$\theta_{i|\Pi_i}(x_i, \Pi_{x_i}) = \frac{N_{i,\Pi_i}(x_i, \Pi_{x_i})}{N_{\Pi_i}(\Pi_{x_i}) + N^0_{i|\Pi_i}} + \frac{N^0_{i|\Pi_i}}{N_{\Pi_i}(\Pi_{x_i}) + N^0_{i|\Pi_i}} \frac{N_i(x_i)}{N} \qquad (4)$$

and suggest setting $N^0_{i|\Pi_i} = 5$ based on empirical results. Using equation 4 to fix the parameters improves the accuracy of the classifier. In our opinion, no well founded justification is given for the improvement. In the next section we introduce decomposable distribution over TANs, a family of probability distributions over the space of TAN models that allow to derive a well founded softening alternative.

3 Decomposable Distributions over TANs

Decomposable priors were introduced by Meila and Jaakola in [13] 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 [13], to the task of learning TAN.

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

$$P(B|\xi) = P(\overline{E}^*, \Theta_{\overline{E}^*}|\xi) = P(\overline{E}^*|\xi)P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) \qquad (5)$$

In the following sections we specify the value of $P(\overline{E}^*|\xi)$ (decomposable distribution over structures) and $P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi)$ (decomposable distribution over parameters).

3.1 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:

$$P(\overline{E}^*|\xi) = \frac{1}{Z_\beta} \prod_{u,v \in E} \beta_{u,v} \qquad (6)$$

where Z_β is a normalization constant

3.2 Decomposable Distribution over TAN Parameters

Applying equation 1 to the case of TAN we have that

$$P(\Theta_{\overline{E}^*}|\overline{E}^*, \xi) = P(\boldsymbol{\theta}_C|\overline{E}^*, \xi) P(\boldsymbol{\theta}_{\rho_{\overline{E}}|C}|\overline{E}^*, \xi) \times \prod_{u,v \in \overline{E}} P(\boldsymbol{\theta}_{v|u,C}|\overline{E}^*, \xi) \qquad (7)$$

A decomposable distribution 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 constraint that exist $N'_{u,C}(i,c)$, $N'_C(c)$, N' such that for every u,v:

$$N'_{u,C}(i,c) = \sum_{j \in A_v} N'_{v,u,C}(j,i,c) \qquad (8)$$

$$N'_C(c) = \sum_{i \in A_u} N'_{u,C}(i,c) \qquad (9)$$

$$N' = \sum_{c \in C} N'_C(c) \qquad (10)$$

Given ξ, a decomposable probability distribution over parameters with hyperparameter $\mathbf{N}'$ is defined by equation 7 and the following set of Dirichlet distributions:

$$P(\boldsymbol{\theta}_C|\overline{E}, \xi) = D(\theta_C(.); N'_C(.)) \qquad (11)$$

$$P(\boldsymbol{\theta}_{\rho_{\overline{E}}|C}|\overline{E}, \xi) = \prod_{c \in C} D(\theta_{\rho_{\overline{E}}|C}(.,c); N'_{\rho_{\overline{E}},C}(.,c)) \qquad (12)$$

$$P(\boldsymbol{\theta}_{v|u,C}|\overline{E}, \xi) = \prod_{c \in C} \prod_{i \in A_u} D(\theta_{v|u,C}(.,i,c); N'_{v,u,C}(.,i,c)) \qquad (13)$$

If the conditions in equations 5, 6, 7, 8, 9, 10, 11, 12 and 13 hold, we will say that $P(M|\xi)$ follows a decomposable distribution with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$.

3.3 Learning with Decomposable Distributions

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

$$\beta^*_{u,v} = \beta_{u,v} W_{u,v} \qquad (14)$$

$$N'^{*}_{u,v,C}(j,i,c) = N'_{u,v,C}(j,i,c) + N_{u,v,C}(j,i,c) \tag{15}$$

where

$$W_{u,v} = \prod_{c \in C} \prod_{i \in A_u} \frac{\Gamma(N'_{u,C}(i,c))}{\Gamma(N'^{*}_{u,C}(i,c))} \prod_{c \in C} \prod_{j \in A_v} \frac{\Gamma(N'_{v,C}(j,c))}{\Gamma(N'^{*}_{v,C}(j,c))} \times$$
$$\times \prod_{c \in C} \prod_{i \in A_u} \prod_{j \in A_v} \frac{\Gamma(N'^{*}_{v,u,C}(j,i,c))}{\Gamma(N'_{v,u,C}(j,i,c))} \tag{16}$$

The proof appears in [5].

3.4 Classifying with Decomposable Distributions given an undirected structure

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

$$P(\mathcal{C} = s_C|\mathcal{V} = S, E, \xi) \propto h_0^{S,s_C} \prod_{u,v \in E} h_{u,v}^{S,s_C} \tag{17}$$

where

$$h_0^{S,s_C} = \frac{1}{Z_\beta} \frac{1}{N'} \prod_{A_u \in V} N'_{u,C}(s_u, s_C) \tag{18}$$

$$h_{u,v}^{S,s_C} = \frac{N'_{v,u,C}(s_v, s_u, s_C)}{N'_{u,C}(s_u, s_C)N'_{v,C}(s_v, s_C)} \tag{19}$$

The proof appears in [3].

4 Maximum a Posteriori results for decomposable distributions over TANs

In this section we show that if we assume a decomposable distribution over TANs as prior over the set of models, the undirected tree structure underlying the MAP TAN can be found in $\mathcal{O}((N+r^3) \cdot n^2)$ time where $r = \max(\max_{i \in V} \#A_i, \#C)$. Furthermore, we also show that we can find the k undirected tree structures underlying the k MAP TAN models and their relative weights in $\mathcal{O}((N + r^3 + k) \cdot n^2)$.

Both results are supported by the next result, that shows that computing the most probable undirected tree structure under a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ can be reduced to calculating the maximum spanning tree (MST) for the graph with adjacency matrix $\log(\boldsymbol{\beta})$.

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

From equation 6, assuming that $\forall u, v \ u \neq v; \ \beta_{u,v} > 0$, we can characterize the most probable undirected tree given a decomposable distribution over TANs with hyperparameters $\boldsymbol{\beta}, \mathbf{N}'$ is given by

$$MPT(\boldsymbol{\beta}, \mathbf{N}') = \underset{E \in \mathcal{E}}{argmax} \sum_{u,v \in E} \log(\beta_{u,v}) \tag{20}$$

Considering the matrix $\log(\boldsymbol{\beta})$ as an adjacency matrix, $MPT(\boldsymbol{\beta}, \mathbf{N}')$ is the MST 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 calculating 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 $\mathcal{O}(n^2)$ [15].

4.2 Calculating the MAP TAN structure given a prior decomposable distribution over TANs

If we assume a decomposable prior distribution over TANs the posterior distribution after a dataset $\mathcal{D}$ follows a decomposable distribution over TANs (section 3.3). Since the posterior is a decomposable distribution over trees, 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 1.

4.3 Calculating the k MAP TAN structures and their relative probability weights given a prior decomposable distribution over TANs

The problem of computing the k MST in order is well known and can be solved in $\mathcal{O}(k \cdot n^2)$ for a complete graph [9]. It is easy to see that if in the last step of `MAPTreeStructure` instead of calculating the MST we calculate the k MST and their relative weights as shown in algorithm 2, the algorithm will return the k MAP TANs and their relative probabilities. The time complexity of the new algorithm is $\mathcal{O}((N + r^3 + k) \cdot n^2)$.

5 Constructing the MAPTAN and MAPTAN+BMA classifiers

From the result in section 3.4 it is easy to see that given an undirected TAN structure E, 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

$$
\begin{aligned}
\theta_{u|v,C}(s_u, s_v, s_C) &= \frac{N'_{u,v,C}(s_u, s_v, s_C)}{N'_{v,C}(s_v, s_C)} \\
\theta_{u|C}(s_u, s_C) &= \frac{N'_{u,C}(s_u, s_C)}{N'_C(s_C)} \\
\theta_C(s_C) &= \frac{N'_C(s_C)}{N'}
\end{aligned}
\tag{21}
$$

```
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 1: Computation of the MAP TAN

A similar result in the case of decomposable distribution over trees can also be found in [14]. Given a decomposable prior we can calculate the decomposable posterior using the result in section 3.3 and then apply the result we have just enunciated to the posterior.

The posterior probability distribution $P(\mathcal{C} = s_C|\mathcal{V} = S, E, \mathcal{D}, \xi)$ 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

$$
\theta_{u|v,C}(s_u, s_v, s_C) = \frac{N'^*_{u,v,C}(s_u,s_v,s_C)}{N'^*_{v,C}(s_v,s_C)}
$$

$$
\theta_{u|C}(s_u, s_C) = \frac{N'^*_{u,C}(s_u,s_C)}{N'^*_C(s_C)}
$$

$$
\theta_C(s_C) = \frac{N'^*_C(s_C)}{N'^*}
$$

(22)

In [4] we argued that

$$\forall u,v \ ; \ 1 \leq u \neq v \ \leq n \ ; \ \beta_{u,v} = 1 \qquad (23)$$

$$\forall u,v; 1 \leq u \neq v \leq n; \forall j \in A_v; \forall i \in A_u; \forall c \in C$$
$$N'_{v,u,C}(j,i,c) = \frac{\lambda}{\#C\#A_u\#A_v} \qquad (24)$$

where λ is an equivalent sample size, provide a reasonable choice of the hyperparameters if no information from the domain is available.

5.1 MAPTAN **classifier**

After fixing the prior hyperparameters, the learning step for MAPTAN classifier consists in:

1. Applying algorithm 1 to find the undirected tree E underlying the MAP TAN structure given a dataset $\mathcal{D}$.
2. Randomly choose a root, create a directed tree $\overline{E}$ and from it a directed TAN structure $\overline{E}^*$.
3. Use equation 22 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.

```
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 2: Computation of the k MAP TANs

5.2 MAPTAN+BMA **classifier**

After fixing the prior hyperparameters, the learning stage for MAPTAN+BMA classifier consists in:

1. Applying algorithm 2 to find the k undirected trees underlying the k MAP TAN structures and their relative probability weights given a dataset $\mathcal{D}$.
2. Generate a TAN model for each of the undirected tree structures as we did in MAPTAN.

3. Assign to each TAN model the weight of its corresponding undirected tree.

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

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

5.3 Relevant characteristics of MAPTAN and MAPTAN+BMA

We have shown that decomposable distributions over TANs can be used to construct two well founded classifiers: MAPTAN and MAPTAN+BMA. In the introduction we highlighted two possible ways in which the TAN classifier, as presented in [7], could be improved: by taking into account model uncertainty and by providing a theoretically well founded explanation for the use of softening.

We have seen that MAPTAN+BMA provides a theoretically well founded way of dealing with model uncertainty. Its learning time complexity regarding N is almost equivalent to that of STAN, and it grows polynomially on k. This is much more efficient than the algorithm for learning k TAN models proposed in [2]. MAPTAN+BMA has a classification time complexity, $\mathcal{O}(nrk)$ reasonably higher than that of STAN. Furthermore, we can use k as an *effort knob*, in the sense of [16], hence providing a useful feature for data mining users that allows them to decide how much computational power they want to spend in the task. In our opinion, MAPTAN+BMA provides a good complexity tradeoff to deal with model uncertainty when learning TAN.

Both MAPTAN and MAPTAN+BMA can be interpreted as using softening in both the structure search and the parameter fixing. This softening appears, in a natural way, as the result of assuming a decomposable distribution over TANs as the prior over the set of models. In our opinion MAPTAN is theoretically more appealing than STAN.

Both MAPTAN and MAPTAN+BMA share the relevant characteristic of allowing the use of some form of prior information if such is available, specially structure related information. For example, if we have expert knowledge that tell us that one of the edges of the tree is much more (equiv. 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 [12], decomposable distributions do not allow the expression of some types of prior information such as "if edge (u, v) exists then edge (w, z) is very likely to exist".

6 Empirical results

We tested four algorithms over 17 datasets from the Irvine repository [1]. To discretize continuous attributes we used equal frequency discretization with 5 intervals. For each dataset and algorithm we tested both error rate 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 the test set $\mathcal{D}'$ we have

$$LogScore(B, \mathcal{D}') = \sum_{(S,s_C) \in \mathcal{D}'} -\log(P(\mathcal{C} = s_C | \mathcal{V} = S, B)) \qquad (25)$$

For the evaluation of both error rate and *LogScore* we used 10 fold 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 behaviors of both error rate and *LogScore* for the algorithm.

The error rates appear in Tables 1, 3 and 5 with the best method for each dataset boldfaced. *LogScore*'s appear in Tables 2, 4 and 6. The columns of the tables are the induction methods and the rows are the datasets. The meaning of the column headers are:

- STAN is the softened TAN induction algorithm as presented in [7].
- STAN+BMA is the classifier resulting from applying local Bayesian model averaging (see [2]) to STAN.
- MAPTAN, is the classifier based on the MAP TAN model described in section 5.
- MAPTAN+BMA is the classifier based on the weighted average of the k MAP TAN models described also in section 5.

6.1 Interpretation of the results

Summarizing the empirical results in the tables, we can conclude that:

- MAPTAN improves STAN error rate for most datasets and has a similar *LogScore*.
- MAPTAN+BMA improves MAPTAN's *LogScore* for most datasets. When little data is available, it also improves its error rate.
- MAPTAN+BMA improves STAN+BMA error rate and *LogScore* for many datasets.

7 Conclusions

We have seen that under a decomposable distribution over TANs it is possible to efficiently determine the MAP undirected TAN structure and the set of k MAP TAN structures and their relative probability weights. We used these results to construct two new classifiers: MAPTAN and MAPTAN+BMA. We have provided empirical results showing that both classifiers improve over established TAN based classifiers (in the case of MAPTAN+BMA reducing also severely the computational complexity). Our results give also a satisfying theoretical explanation for the use of softening in TAN based classifiers (as the result of Bayesian model averaging over parameters). These results highlight the relevance of decomposable distributions over TANs for the construction of TAN based classifiers.

Dataset	MAPTAN	MAPTAN+BMA	sTAN	sTAN+BMA
ADULT	**17.18 ± 0.68**	17.19 ± 0.71	17.60 ± 0.82	17.60 ± 0.80
AUSTRALIAN	19.91 ± 1.14	**19.62 ± 1.13**	25.39 ± 1.18	24.96 ± 1.13
BREAST	17.23 ± 1.21	16.89 ± 1.28	8.73 ± 0.87	**7.73 ± 0.93**
CAR	17.19 ± 1.04	**16.50 ± 0.84**	19.38 ± 0.95	17.60 ± 0.77
CHESS	9.55 ± 0.80	**9.48 ± 0.86**	10.89 ± 0.56	10.91 ± 0.53
CLEVE	**28.12 ± 1.68**	28.14 ± 1.59	32.37 ± 1.00	31.89 ± 1.27
CRX	19.77 ± 0.91	**19.16 ± 1.00**	25.14 ± 0.87	24.18 ± 0.98
FLARE	23.50 ± 1.09	23.16 ± 1.09	19.94 ± 0.85	**19.92 ± 0.88**
GLASS	47.02 ± 1.66	**45.72 ± 1.59**	59.19 ± 1.78	58.54 ± 1.83
GLASS2	33.69 ± 1.74	**32.87 ± 1.82**	37.75 ± 1.39	36.63 ± 1.37
IRIS	28.67 ± 2.33	26.27 ± 2.30	25.87 ± 3.07	**24.80 ± 2.96**
LETTER	30.22 ± 0.96	**30.19 ± 0.97**	36.11 ± 1.39	34.68 ± 1.37
LIVER	45.52 ± 1.26	44.96 ± 1.06	42.39 ± 0.94	**41.24 ± 1.37**
NURSERY	7.87 ± 1.03	**7.57 ± 1.04**	8.88 ± 1.12	8.50 ± 1.12
PRIMARY-TUMOR	74.52 ± 1.73	74.28 ± 1.66	**71.67 ± 1.54**	71.73 ± 1.44
SOYBEAN	26.53 ± 1.30	**26.51 ± 1.33**	30.79 ± 1.28	30.82 ± 1.33
VOTES	**9.61 ± 0.94**	9.67 ± 0.99	14.14 ± 0.93	14.13 ± 0.71

Table 1. Averages and standard deviations of error rate using 10% of the learning data

Dataset	MAPTAN	MAPTAN+BMA	sTAN	sTAN+BMA
ADULT	562.25 ± 3.75	**561.39 ± 3.71**	567.09 ± 3.92	567.64 ± 4.00
AUSTRALIAN	18.54 ± 0.95	17.68 ± 0.96	17.85 ± 0.64	**17.06 ± 0.60**
BREAST	23.59 ± 1.67	18.24 ± 1.56	8.12 ± 0.69	**7.56 ± 0.65**
CAR	34.89 ± 1.02	**32.79 ± 0.98**	38.55 ± 0.91	36.52 ± 0.86
CHESS	32.50 ± 0.89	**32.25 ± 0.91**	35.39 ± 0.58	35.40 ± 0.59
CLEVE	11.15 ± 1.06	10.09 ± 0.96	8.49 ± 0.74	**8.23 ± 0.76**
CRX	19.44 ± 1.06	18.30 ± 1.00	17.84 ± 1.05	**16.89 ± 1.00**
FLARE	51.12 ± 1.17	**49.48 ± 1.15**	24332.38 ± 56.59	24332.03 ± 56.59
GLASS	20.49 ± 1.45	**17.14 ± 1.40**	11713.24 ± 72.91	11713.00 ± 72.91
GLASS2	6.45 ± 0.79	5.49 ± 0.64	4.68 ± 0.57	**4.57 ± 0.54**
IRIS	4.58 ± 0.68	4.06 ± 0.69	4.04 ± 0.67	**3.96 ± 0.70**
LETTER	3535.93 ± 12.92	3495.14 ± 13.52	1385.73 ± 8.95	**1300.23 ± 8.38**
LIVER	18.71 ± 0.95	15.87 ± 0.92	12.62 ± 0.79	**11.71 ± 0.65**
NURSERY	112.72 ± 2.47	**111.95 ± 2.47**	3126.39 ± 77.45	3123.62 ± 77.45
PRIMARY-TUMOR	71.74 ± 2.08	**69.08 ± 2.05**	75927.03 ± 123.39	75926.94 ± 123.39
SOYBEAN	68.52 ± 1.77	**65.29 ± 1.55**	41125.59 ± 108.25	41125.46 ± 108.25
VOTES	5.66 ± 0.66	**5.17 ± 0.60**	6.09 ± 0.50	6.03 ± 0.48

Table 2. Averages and standard deviations of *LogScore* using 10% of the learning data

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. *Improving Bayesian network classifiers*. PhD thesis, Technical University of Catalonia, 2003.
4. Jesús Cerquides and Ramon López de Màntaras. Tractable bayesian learning of tree augmented naive bayes classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 75–82, 2003.
5. Jesús Cerquides and Ramon López de Màntaras. Tractable bayesian learning of tree augmented naive bayes classifiers. long version. Technical Report IIIA-2003-04, Institut d'Investigació en Intel.ligència Artificial, 2003.
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.

Dataset	MAPTAN	MAPTAN+BMA	sTAN	sTAN+BMA
ADULT	**16.26 ± 0.75**	16.28 ± 0.77	16.46 ± 0.78	16.45 ± 0.83
AUSTRALIAN	15.36 ± 0.94	**15.13 ± 1.09**	18.14 ± 0.91	17.74 ± 0.80
BREAST	5.92 ± 0.74	5.84 ± 0.78	5.26 ± 0.84	**4.75 ± 0.72**
CAR	7.62 ± 0.75	**7.55 ± 0.76**	8.68 ± 0.68	8.09 ± 0.58
CHESS	**7.87 ± 0.44**	7.90 ± 0.44	8.25 ± 0.49	8.15 ± 0.49
CLEVE	**19.82 ± 1.30**	20.27 ± 1.27	24.01 ± 1.31	23.57 ± 1.28
CRX	15.47 ± 1.01	**15.30 ± 1.02**	18.12 ± 0.92	17.68 ± 0.85
FLARE	19.83 ± 0.72	19.81 ± 0.65	18.55 ± 0.62	**18.54 ± 0.72**
GLASS	24.02 ± 1.22	**23.31 ± 1.48**	33.79 ± 1.14	33.86 ± 0.97
GLASS2	23.69 ± 1.62	22.81 ± 1.61	**22.38 ± 1.53**	23.40 ± 1.54
IRIS	11.60 ± 1.22	11.07 ± 1.08	8.40 ± 1.00	**8.27 ± 0.82**
LETTER	**14.79 ± 0.78**	**14.79 ± 0.78**	15.62 ± 0.91	15.31 ± 0.83
LIVER	37.33 ± 1.16	36.90 ± 1.15	36.73 ± 1.60	**35.17 ± 1.34**
NURSERY	6.39 ± 0.92	6.37 ± 0.89	7.09 ± 0.80	**6.03 ± 0.97**
PRIMARY-TUMOR	59.15 ± 1.67	**59.09 ± 1.63**	60.23 ± 1.17	59.87 ± 1.33
SOYBEAN	6.64 ± 0.77	**6.50 ± 0.85**	7.88 ± 0.71	7.80 ± 0.82
VOTES	**6.22 ± 0.83**	6.25 ± 0.84	7.63 ± 0.93	7.76 ± 0.93

Table 3. Averages and standard deviations of error rate using 50% of the learning data

Dataset	MAPTAN	MAPTAN+BMA	sTAN	sTAN+BMA
ADULT	507.82 ± 3.82	**507.52 ± 3.81**	520.03 ± 3.93	518.82 ± 3.91
AUSTRALIAN	12.86 ± 0.82	**12.57 ± 0.84**	14.79 ± 0.76	14.41 ± 0.59
BREAST	10.95 ± 0.67	9.20 ± 0.69	5.17 ± 0.64	**4.40 ± 0.62**
CAR	15.96 ± 0.44	**15.90 ± 0.40**	20.44 ± 0.51	19.73 ± 0.48
CHESS	26.70 ± 0.66	**26.66 ± 0.68**	27.32 ± 0.73	27.12 ± 0.79
CLEVE	6.83 ± 0.69	**6.70 ± 0.66**	7.38 ± 0.66	7.15 ± 0.63
CRX	13.28 ± 0.89	**12.93 ± 0.85**	15.62 ± 1.11	15.21 ± 1.07
FLARE	39.81 ± 1.16	**39.45 ± 1.15**	4233.42 ± 41.82	4233.31 ± 41.82
GLASS	11.05 ± 0.73	**9.37 ± 0.84**	309.52 ± 24.49	309.25 ± 24.49
GLASS2	5.06 ± 0.73	4.64 ± 0.64	3.86 ± 0.53	**3.68 ± 0.51**
IRIS	1.87 ± 0.35	1.77 ± 0.34	1.52 ± 0.37	**1.48 ± 0.34**
LETTER	1030.65 ± 9.50	1030.65 ± 9.50	574.47 ± 6.13	**559.56 ± 6.17**
LIVER	13.03 ± 0.89	12.21 ± 0.77	10.78 ± 0.74	**10.39 ± 0.71**
NURSERY	96.60 ± 2.40	**96.52 ± 2.42**	1596.96 ± 67.06	1594.32 ± 67.06
PRIMARY-TUMOR	44.24 ± 1.25	**43.00 ± 1.23**	12028.93 ± 51.79	12028.74 ± 51.79
SOYBEAN	6.79 ± 0.83	**6.47 ± 0.74**	907.34 ± 42.43	907.27 ± 42.43
VOTES	3.66 ± 0.58	**3.54 ± 0.52**	5.04 ± 0.80	4.50 ± 0.69

Table 4. Averages and standard deviations of *LogScore* using 50% of the learning data

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

10. 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.

11. 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.

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

13. Marina Meila and Tommi Jaakkola. Tractable bayesian learning of tree belief networks. Technical Report CMU-RI-TR-00-15, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2000.

Dataset	MAPTAN	MAPTAN+BMA	sTAN	sTAN+BMA
ADULT	**16.35 ± 0.73**	16.35 ± 0.73	16.46 ± 0.68	16.42 ± 0.72
AUSTRALIAN	13.68 ± 0.75	**13.65 ± 0.74**	16.49 ± 0.65	16.43 ± 0.72
BREAST	4.75 ± 0.53	4.63 ± 0.48	4.29 ± 0.66	**3.72 ± 0.45**
CAR	**5.76 ± 0.52**	5.78 ± 0.45	6.23 ± 0.55	6.16 ± 0.53
CHESS	7.71 ± 0.25	**7.67 ± 0.21**	7.89 ± 0.38	7.68 ± 0.44
CLEVE	18.74 ± 1.15	**18.53 ± 1.18**	19.99 ± 1.26	19.73 ± 1.18
CRX	13.67 ± 0.53	**13.53 ± 0.58**	15.71 ± 0.66	15.79 ± 0.74
FLARE	19.71 ± 0.49	19.71 ± 0.55	18.46 ± 0.30	**18.31 ± 0.24**
GLASS	**18.46 ± 1.20**	18.74 ± 1.29	26.58 ± 1.22	25.99 ± 1.28
GLASS2	19.81 ± 0.85	20.25 ± 1.39	19.61 ± 1.42	**18.06 ± 1.43**
IRIS	7.73 ± 1.70	7.47 ± 1.66	8.13 ± 1.44	**7.20 ± 1.43**
LETTER	**11.49 ± 0.74**	11.49 ± 0.74	12.69 ± 0.77	12.48 ± 0.83
LIVER	34.35 ± 0.86	33.99 ± 0.77	33.36 ± 0.98	**33.19 ± 1.10**
NURSERY	6.33 ± 0.89	6.26 ± 0.91	6.62 ± 0.75	**4.81 ± 0.76**
PRIMARY-TUMOR	55.09 ± 1.24	**54.68 ± 1.02**	56.74 ± 1.09	56.32 ± 0.93
SOYBEAN	5.47 ± 0.62	**5.27 ± 0.62**	5.97 ± 0.50	5.94 ± 0.49
VOTES	**5.89 ± 0.74**	5.89 ± 0.72	6.26 ± 0.81	6.34 ± 0.56

Table 5. Averages and standard deviations of error rate using 100% of the learning data

Dataset	MAPTAN	MAPTAN+BMA	sTAN	sTAN+BMA
ADULT	495.88 ± 3.68	**495.70 ± 3.67**	508.10 ± 3.07	508.01 ± 3.07
AUSTRALIAN	10.65 ± 0.46	**10.47 ± 0.44**	12.90 ± 0.65	12.66 ± 0.61
BREAST	8.96 ± 0.87	7.89 ± 0.61	4.85 ± 0.50	**4.28 ± 0.54**
CAR	**14.11 ± 0.40**	14.12 ± 0.40	16.29 ± 0.39	16.31 ± 0.41
CHESS	26.12 ± 0.40	**26.09 ± 0.32**	26.46 ± 0.46	26.22 ± 0.36
CLEVE	6.10 ± 0.43	**6.05 ± 0.38**	6.51 ± 0.44	6.29 ± 0.51
CRX	11.34 ± 0.62	**11.05 ± 0.60**	13.97 ± 0.68	13.76 ± 0.58
FLARE	35.82 ± 0.92	**35.61 ± 0.90**	1532.39 ± 0.62	1532.22 ± 0.65
GLASS	8.53 ± 1.05	7.50 ± 1.03	7.40 ± 0.59	**7.12 ± 0.52**
GLASS2	4.20 ± 0.56	3.91 ± 0.54	3.20 ± 0.39	**3.08 ± 0.37**
IRIS	1.29 ± 0.53	1.22 ± 0.53	1.18 ± 0.44	**1.16 ± 0.44**
LETTER	612.99 ± 7.71	612.99 ± 7.71	441.94 ± 5.61	**433.37 ± 5.84**
LIVER	10.79 ± 0.63	10.61 ± 0.66	**9.59 ± 0.44**	9.72 ± 0.60
NURSERY	94.59 ± 2.45	94.57 ± 2.43	91.52 ± 2.41	**89.41 ± 2.30**
PRIMARY-TUMOR	35.28 ± 0.99	**34.64 ± 0.94**	6327.87 ± 38.33	6327.64 ± 38.33
SOYBEAN	3.45 ± 0.50	**3.38 ± 0.49**	4.49 ± 0.51	4.45 ± 0.48
VOTES	3.74 ± 0.59	**3.57 ± 0.58**	3.96 ± 0.55	3.76 ± 0.46

Table 6. Averages and standard deviations of *LogScore* using 100% of the learning data

14. Marina Meila and Michael I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.
15. Seth Pettie and Vijaya Ramachandran. An optimal minimum spanning tree algorithm. *Journal of the ACM (JACM)*, 49(1):16–34, 2002.
16. Kurt Thearling. Some thoughts on the current state of data mining software applications. In *Keys to the Commercial Success of Data Mining, KDD'98 Workshop*, 1998.