Hindawi Publishing Corporation
EURASIP Journal on Bioinformatics and Systems Biology
Volume 2007, Article ID 90947, 11 pages
doi:10.1155/2007/90947

Research Article

NML Computation Algorithms for Tree-Structured Multinomial Bayesian Networks

Petri Kontkanen, Hannes Wettig, and Petri Myllymäki

Complex Systems Computation Group (CoSCo), Helsinki Institute for Information Technology (HIIT), P.O. Box 68 (Department of Computer Science), FIN-00014 University of Helsinki, Finland

Received 1 March 2007; Accepted 30 July 2007

Recommended by Peter Grünwald

Typical problems in bioinformatics involve large discrete datasets. Therefore, in order to apply statistical methods in such domains, it is important to develop efficient algorithms suitable for discrete data. The minimum description length (MDL) principle is a theoretically well-founded, general framework for performing statistical inference. The mathematical formalization of MDL is based on the normalized maximum likelihood (NML) distribution, which has several desirable theoretical properties. In the case of discrete data, straightforward computation of the NML distribution requires exponential time with respect to the sample size, since the definition involves a sum over all the possible data samples of a fixed size. In this paper, we first review some existing algorithms for efficient NML computation in the case of multinomial and naive Bayes model families. Then we proceed by extending these algorithms to more complex, tree-structured Bayesian networks.

1. INTRODUCTION

Many problems in bioinformatics can be cast as *model class selection* tasks, that is, as tasks of selecting among a set of competing mathematical explanations the one that best describes a given sample of data. Typical examples of this kind of problem are DNA sequence compression [1], microarray data clustering [2–4] and modeling of genetic networks [5]. The *minimum description length* (MDL) principle developed in the series of papers [6–8] is a well-founded, general framework for performing model class selection and other types of statistical inference. The fundamental idea behind the MDL principle is that any regularity in data can be used to *compress* the data, that is, to find a description or *code* of it, such that this description uses less symbols than it takes to describe the data literally. The more regularities there are, the more the data can be compressed. According to the MDL principle, learning can be equated with finding regularities in data. Consequently, we can say that the more we are able to compress the data, the more we have learned about them.

MDL model class selection is based on a quantity called *stochastic complexity* (SC), which is the description length of a given data relative to a model class. The stochastic complexity is defined via the normalized maximum likelihood (NML) distribution [8, 9]. For multinomial (discrete) data, this definition involves a normalizing sum over all the possible data samples of a fixed size. The logarithm of this sum is called the *regret* or *parametric complexity*, and it can be interpreted as the amount of complexity of the model class. If the data is continuous, the sum is replaced by the corresponding integral.

The NML distribution has several theoretical optimality properties, which make it a very attractive candidate for performing model class selection and related tasks. It was originally [8, 10] formulated as the unique solution to a minimax problem presented in [9], which implied that NML is the minimax optimal universal model. Later [11], it was shown that NML is also the solution to a related problem involving expected regret. See Section 2 and [10–13] for more discussion on the theoretical properties of the NML.

Typical bioinformatic problems involve large discrete datasets. In order to apply NML for these tasks one needs to develop suitable NML computation methods since the normalizing sum or integral in the definition of NML is typically difficult to compute directly. In this paper, we present algorithms for efficient computation of NML for both one- and multidimensional discrete data. The model families used in the paper are so-called *Bayesian networks* (see, e.g., [14]) of varying complexity. A Bayesian network is a graphical representation of a joint distribution. The structure of the graph

corresponds to certain conditional independence assumptions. Note that despite the name, having Bayesian network models does not necessarily imply using Bayesian statistics, and the information-theoretic approach of this paper cannot be considered Bayesian.

The problem of computing NML for discrete data has been studied before. In [15] a linear-time algorithm for the one-dimensional multinomial case was derived. A more complex case involving a multidimensional model family, called naive Bayes, was discussed in [16]. Both these cases are also reviewed in this paper.

The paper is structured as follows. In Section 2, we discuss the basic properties of the MDL principle and the NML distribution. In Section 3, we instantiate the NML distribution for the multinomial case and present a linear-time computation algorithm. The topic of Section 4 is the naive Bayes model family. NML computation for an extension of naive Bayes, the so-called Bayesian forests, is discussed in Section 5. Finally, Section 6 gives some concluding remarks.

2. PROPERTIES OF THE MDL PRINCIPLE AND THE NML MODEL

The MDL principle has several desirable properties. Firstly, it automatically protects against overfitting in the model class selection process. Secondly, this statistical framework does not, unlike most other frameworks, assume that there exists some underlying "true" model. The model class is only used as a technical device for constructing an efficient code for describing the data. MDL is also closely related to the Bayesian inference but there are some fundamental differences, the most important being that MDL does not need any prior distribution; it only uses the data at hand. For more discussion on the theoretical motivations behind the MDL principle see, for example, [8, 10–13, 17].

The MDL model class selection is based on minimization of the stochastic complexity. In the following, we give the definition of the stochastic complexity and then proceed by discussing its theoretical properties.

2.1. Model classes and families

Let $\mathbf{x}^n = (x_1, \ldots, x_n)$ be a data sample of n outcomes, where each outcome x_j is an element of some space of observations $\mathcal{X}$. The n-fold Cartesian product $\mathcal{X} \times \cdots \times \mathcal{X}$ is denoted by $\mathcal{X}^n$, so that $\mathbf{x}^n \in \mathcal{X}^n$. Consider a set $\Theta \subseteq \mathbb{R}^d$, where d is a positive integer. A class of parametric distributions indexed by the elements of Θ is called a *model class*. That is, a model class $\mathcal{M}$ is defined as

$$\mathcal{M} = \{P(\cdot \mid \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\} \tag{1}$$

and the set Θ is called the *parameter space*.

Consider a set $\Phi \subseteq \mathbb{R}^e$, where e is a positive integer. Define a set $\mathcal{F}$ by

$$\mathcal{F} = \{\mathcal{M}(\boldsymbol{\varphi}) : \boldsymbol{\varphi} \in \Phi\}. \tag{2}$$

The set $\mathcal{F}$ is called a *model family*, and each of the elements $\mathcal{M}(\boldsymbol{\varphi})$ is a model class. The associated parameter space is denoted by $\Theta_{\boldsymbol{\varphi}}$. The model class selection problem can now be defined as a process of finding the parameter vector $\boldsymbol{\varphi}$, which is optimal according to some predetermined criteria. In Sections 3–5, we discuss three specific model families, which will make these definitions more concrete.

2.2. The NML distribution

One of the most theoretically and intuitively appealing model class selection criteria is the *stochastic complexity*. Denote first the maximum likelihood estimate of data $\mathbf{x}^n$ for a given model class $\mathcal{M}(\boldsymbol{\varphi})$ by $\hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\varphi}))$, that is, $\hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\varphi})) = \arg\max_{\boldsymbol{\theta} \in \Theta_{\boldsymbol{\varphi}}} \{P(\mathbf{x}^n \mid \boldsymbol{\theta})\}$. The *normalized maximum likelihood* (NML) distribution [9] is now defined as

$$P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(\boldsymbol{\varphi})) = \frac{P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\varphi})))}{\mathcal{C}(\mathcal{M}(\boldsymbol{\varphi}), n)}, \tag{3}$$

where the normalizing term $\mathcal{C}(\mathcal{M}(\boldsymbol{\varphi}), n)$ in the case of discrete data is given by

$$\mathcal{C}(\mathcal{M}(\boldsymbol{\varphi}), n) = \sum_{\mathbf{y}^n \in \mathcal{X}^n} P(\mathbf{y}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{y}^n, \mathcal{M}(\boldsymbol{\varphi}))) \tag{4}$$

and the sum goes over the space of data samples of size n. If the data is continuous, the sum is replaced by the corresponding integral.

The stochastic complexity of the data $\mathbf{x}^n$, given a model class $\mathcal{M}(\boldsymbol{\varphi})$, is defined via the NML distribution as

$$\begin{aligned} &\mathrm{SC}(\mathbf{x}^n \mid \mathcal{M}(\varphi)) \\ &\quad = -\log P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(\boldsymbol{\varphi})) \\ &\quad = -\log P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\varphi}))) + \log \mathcal{C}(\mathcal{M}(\boldsymbol{\varphi}), n) \end{aligned} \tag{5}$$

and the term $\log \mathcal{C}(\mathcal{M}(\boldsymbol{\varphi}), n)$ is called the *(minimax) regret* or *parametric complexity*. The regret can be interpreted as measuring the logarithm of the number of essentially different (distinguishable) distributions in the model class. Intuitively, if two distributions assign high likelihood to the same data samples, they do not contribute much to the overall complexity of the model class, and the distributions should not be counted as different for the purposes of statistical inference. See [18] for more discussion on this topic.

The NML distribution (3) has several important theoretical optimality properties. The first is that NML provides a unique solution to the minimax problem

$$\min_{\hat{P}} \max_{\mathbf{x}^n} \log \frac{P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\varphi})))}{\hat{P}(\mathbf{x}^n \mid \mathcal{M}(\varphi))}, \tag{6}$$

as posed in [9]. The minimizing $\hat{P}$ is the NML distribution, and the minimax regret

$$\log P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\varphi}))) - \log \hat{P}(\mathbf{x}^n \mid \mathcal{M}(\boldsymbol{\varphi})) \tag{7}$$

is given by the parametric complexity $\log \mathcal{C}(\mathcal{M}(\boldsymbol{\varphi}), n)$. This means that the NML distribution is the *minimax optimal universal model*. The term universal model in this context means

that the NML distribution represents (or mimics) the behavior of all the distributions in the model class $\mathcal{M}(\boldsymbol{\varphi})$. Note that the NML distribution itself does not have to belong to the model class, and typically it does not.

A related property of NML involving expected regret was proven in [11]. This property states that NML is also a unique solution to

$$\max_{g} \min_{q} E_g \log \frac{P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\varphi)))}{q(\mathbf{x}^n \mid \mathcal{M}(\boldsymbol{\varphi}))}, \tag{8}$$

where the expectation is taken over $\mathbf{x}^n$ with respect to g and the minimizing distribution q equals g. Also the maximin expected regret is thus given by $\log \mathcal{C}(\mathcal{M}(\boldsymbol{\varphi}), n)$.

3. NML FOR MULTINOMIAL MODELS

In the case of discrete data, the simplest model family is the *multinomial*. The data are assumed to be one-dimensional and to have only a finite set of possible values. Although simple, the multinomial model family has practical applications. For example, in [19] multinomial NML was used for histogram density estimation, and the density estimation problem was regarded as a model class selection task.

3.1. The model family

Assume that our problem domain consists of a single discrete random variable X with K values, and that our data $\mathbf{x}^n = (x_1, \ldots, x_n)$ is multinomially distributed. The space of observations $\mathcal{X}$ is now the set $\{1, 2, \ldots, K\}$. The corresponding model family $\mathcal{F}_{\mathrm{MN}}$ is defined by

$$\mathcal{F}_{\mathrm{MN}} = \{\mathcal{M}(\boldsymbol{\varphi}) : \boldsymbol{\varphi} \in \Phi_{\mathrm{MN}}\}, \tag{9}$$

where $\Phi_{\mathrm{MN}} = \{1, 2, 3, \ldots\}$. Since the parameter vector $\boldsymbol{\varphi}$ is in this case a single integer K we denote the multinomial model classes by $\mathcal{M}(K)$ and define

$$\mathcal{M}(K) = \{P(\cdot \mid \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta_K\}, \tag{10}$$

where Θ_K is the simplex-shaped parameter space,

$$\Theta_K = \{(\pi_1, \ldots, \pi_K) : \pi_k \geq 0,\ \pi_1 + \cdots + \pi_K = 1\} \tag{11}$$

with $\pi_k = P(X = k)$, $k = 1, \ldots, K$.

Assume the data points x_j are independent and identically distributed (i.i.d.). The NML distribution (3) for the model class $\mathcal{M}(K)$ is now given by (see, e.g., [16, 20])

$$P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(K)) = \frac{\prod_{k=1}^{K} (h_k/n)^{h_k}}{\mathcal{C}(\mathcal{M}(K), n)}, \tag{12}$$

where h_k is the frequency (number of occurrences) of value k in $\mathbf{x}^n$, and

$$\mathcal{C}(\mathcal{M}(K), n) = \sum_{\mathbf{y}^n} P(\mathbf{y}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{y}^n, \mathcal{M}(K))) \tag{13}$$

$$= \sum_{h_1 + \cdots + h_K = n} \frac{n!}{h_1! \cdots h_K!} \prod_{k=1}^{K} \left(\frac{h_k}{n}\right)^{h_k}. \tag{14}$$

To make the notation more compact and consistent in this section and the following sections, $\mathcal{C}(\mathcal{M}(K), n)$ is from now on denoted by $\mathcal{C}_{\mathrm{MN}}(K, n)$.

It is clear that the maximum likelihood term in (12) can be computed in linear time by simply sweeping through the data once and counting the frequencies h_k. However, the normalizing sum $\mathcal{C}_{\mathrm{MN}}(K, n)$ (and thus also the parametric complexity $\log \mathcal{C}_{\mathrm{MN}}(K, n)$) involves a sum over an exponential number of terms. Consequently, the time complexity of computing the multinomial NML is dominated by (14).

3.2. The quadratic-time algorithm

In [16, 20], a recursion formula for removing the exponentiality of $\mathcal{C}_{\mathrm{MN}}(K, n)$ was presented. This formula is given by

$$\begin{aligned} \mathcal{C}_{\mathrm{MN}}(K, n) = &\sum_{r_1 + r_2 = n} \frac{n!}{r_1! r_2!} \left(\frac{r_1}{n}\right)^{r_1} \left(\frac{r_2}{n}\right)^{r_2} \\ &\cdot \mathcal{C}_{\mathrm{MN}}(K^*, r_1) \cdot \mathcal{C}_{\mathrm{MN}}(K - K^*, r_2), \end{aligned} \tag{15}$$

which holds for all $K^* = 1, \ldots, K - 1$. A straightforward algorithm based on this formula was then used to compute $\mathcal{C}_{\mathrm{MN}}(K, n)$ in time $\mathcal{O}(n^2 \log K)$. See [16, 20] for more details. Note that in [21, 22] the quadratic-time algorithm was improved to $\mathcal{O}(n \log n \log K)$ by writing (15) as a convolution-type sum and then using the fast Fourier transform algorithm. However, the relevance of this result is unclear due to severe numerical instability problems it easily produces in practice.

3.3. The linear-time algorithm

Although the previous algorithms have succeeded in removing the exponentiality of the computation of the multinomial NML, they are still superlinear with respect to n. In [15], a linear-time algorithm based on the mathematical technique of generating functions was derived for the problem.

The starting point of the derivation is the generating function B defined by

$$B(z) = \frac{1}{1 - T(z)} = \sum_{n \geq 0} \frac{n^n}{n!} z^n, \tag{16}$$

where T is the so-called *Cayley's tree function* [23, 24]. It is easy to prove (see [15, 25]) that the function B^K generates the sequence $((n^n/n!)\mathcal{C}_{\mathrm{MN}}(K, n))_{n=0}^{\infty}$, that is,

$$\begin{aligned} B^K(z) &= \sum_{n \geq 0} \frac{n^n}{n!} \cdot \sum_{h_1 + \cdots + h_K = n} \frac{n!}{h_1! \cdots h_K!} \prod_{k=1}^{K} \left(\frac{h_k}{n}\right)^{h_k} z^n \\ &= \sum_{n \geq 0} \frac{n^n}{n!} \cdot \mathcal{C}_{\mathrm{MN}}(K, n) z^n, \end{aligned} \tag{17}$$

which by using the tree function T can be written as

$$B^K(z) = \frac{1}{(1 - T(z))^K}. \tag{18}$$

The properties of the tree function T can be used to prove the following theorem.

Theorem 1. *The $C_{MN}(K,n)$ terms satisfy the recurrence*

$$C_{MN}(K+2,n) = C_{MN}(K+1,n) + \frac{n}{K}\cdot C_{MN}(K,n). \quad (19)$$

Proof. See the appendix. □

It is now straightforward to write a linear-time algorithm for computing the multinomial NML $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(K))$ based on Theorem 1. The process is described in Algorithm 1. The time complexity of the algorithm is clearly $\mathcal{O}(n+K)$, which is a major improvement over the previous methods. The algorithm is also very easy to implement and does not suffer from any numerical instability problems.

3.4. Approximating the multinomial NML

In practice, it is often not necessary to compute the exact value of $C_{\mathrm{MN}}(K,n)$. A very general and powerful mathematical technique called *singularity analysis* [26] can be used to derive an accurate, constant-time approximation for the multinomial regret. The idea of singularity analysis is to use the analytical properties of the generating function in question by studying its singularities, which then leads to the asymptotic form for the coefficients. See [25, 26] for details.

For the multinomial case, the singularity analysis approximation was first derived in [25] in the context of memoryless sources, and later [20] re-introduced in the MDL framework. The approximation is given by

$$\begin{aligned}\log C_{\mathrm{MN}}(K,n) &= \frac{K-1}{2}\log\frac{n}{2} + \log\frac{\sqrt{\pi}}{\Gamma(K/2)} + \frac{\sqrt{2}K\cdot\Gamma(K/2)}{3\Gamma(K/2-1/2)}\cdot\frac{1}{\sqrt{n}} \\ &\quad + \left(\frac{3+K(K-2)(2K+1)}{36} - \frac{\Gamma^2(K/2)\cdot K^2}{9\Gamma^2(K/2-1/2)}\right)\cdot\frac{1}{n} \\ &\quad + \mathcal{O}\left(\frac{1}{n^{3/2}}\right).\end{aligned} \quad (20)$$

Since the error term of (20) goes down with the rate $\mathcal{O}(1/n^{3/2})$, the approximation converges very rapidly. In [20], the accuracy of (20) and two other approximations (Rissanen's asymptotic expansion [8] and Bayesian information criterion (BIC) [27]) were tested empirically. The results show that (20) is significantly better than the other approximations and accurate already with very small sample sizes. See [20] for more details.

4. NML FOR THE NAIVE BAYES MODEL

The one-dimensional case discussed in the previous section is not adequate for many real-world situations, where data are typically multidimensional, involving complex dependencies between the domain variables. In [16], a quadratic-time algorithm for computing the NML for a specific multivariate model family, usually called the naive Bayes, was derived. This model family has been very successful in practice in mixture modeling [28], clustering of data [16], case-based reasoning [29], classification [30, 31], and data visualization [32].

4.1. The model family

Let us assume that our problem domain consists of m primary variables $X_1,\ldots,X_m$ and a special variable X_0, which can be one of the variables in our original problem domain or it can be latent. Assume that the variable X_i has K_i values and that the extra variable X_0 has K_0 values. The data $\mathbf{x}^n = (\mathbf{x}_1,\ldots,\mathbf{x}_n)$ consist of observations of the form $\mathbf{x}_j = (x_{j0}, x_{j1},\ldots,x_{jm}) \in \mathcal{X}$, where

$$\mathcal{X} = \{1,2,\ldots,K_0\}\times\{1,2,\ldots,K_1\}\times\cdots\times\{1,2,\ldots,K_m\}. \quad (21)$$

The naive Bayes model family $\mathcal{F}_{\mathrm{NB}}$ is defined by

$$\mathcal{F}_{\mathrm{NB}} = \{\mathcal{M}(\boldsymbol{\varphi}) : \boldsymbol{\varphi} \in \Phi_{\mathrm{NB}}\} \quad (22)$$

with $\Phi_{\mathrm{NB}} = \{1,2,3,\ldots\}^{m+1}$. The corresponding model classes are denoted by $\mathcal{M}(K_0,K_1,\ldots,K_m)$:

$$\mathcal{M}(K_0,K_1,\ldots,K_m) = \{P_{\mathrm{NB}}(\cdot \mid \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta_{K_0,K_1,\ldots,K_m}\}. \quad (23)$$

The basic naive Bayes assumption is that given the value of the special variable, the primary variables are independent. We have consequently

$$\begin{aligned}&P_{\mathrm{NB}}(X_0 = x_0, X_1 = x_1,\ldots,X_m = x_m \mid \boldsymbol{\theta}) \\ &\quad = P(X_0 = x_0 \mid \boldsymbol{\theta})\cdot\prod_{i=1}^{m} P(X_i = x_i \mid X_0 = x_0, \boldsymbol{\theta}).\end{aligned} \quad (24)$$

Furthermore, we assume that the distribution of $P(X_0 \mid \boldsymbol{\theta})$ is multinomial with parameters $(\pi_1,\ldots,\pi_{K_0})$, and each $P(X_i \mid X_0 = k, \boldsymbol{\theta})$ is multinomial with parameters $(\sigma_{ik1},\ldots,\sigma_{ikK_i})$. The whole parameter space is then

$$\begin{aligned}&\Theta_{K_0,K_1,\ldots,K_m} \\ &= \{(\pi_1,\ldots,\pi_{K_0}),(\sigma_{111},\ldots,\sigma_{11K_1}),\ldots,(\sigma_{mK_01},\ldots,\sigma_{mK_0K_m}) : \\ &\quad \pi_k \geq 0,\ \sigma_{ikl} \geq 0,\ \pi_1+\cdots+\pi_{K_0} = 1, \\ &\quad \sigma_{ik1}+\cdots+\sigma_{ikK_i} = 1,\ i = 1,\ldots,m,\ k = 1,\ldots K_0\},\end{aligned} \quad (25)$$

and the parameters are defined by $\pi_k = P(X_0 = k)$, $\sigma_{ikl} = P(X_i = l \mid X_0 = k)$.

Assuming i.i.d., the NML distribution for the naive Bayes can now be written as (see [16])

$$\begin{aligned}&P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(K_0,K_1,\ldots,K_m)) \\ &\quad = \frac{\prod_{k=1}^{K_0}(h_k/n)^{h_k}\prod_{i=1}^{m}\prod_{l=1}^{K_i}(f_{ikl}/h_k)^{f_{ikl}}}{C(\mathcal{M}(K_0,K_1,\ldots,K_m),n)},\end{aligned} \quad (26)$$

where h_k is the number of times X_0 has value k in $\mathbf{x}^n$, f_{ikl} is the number of times X_i has value l when the special variable has value k, and $C(\mathcal{M}(K_0,K_1,\ldots,K_m),n)$ is given by (see [16])

$$\begin{aligned}&C(\mathcal{M}(K_0,K_1,\ldots,K_m),n) \\ &\quad = \sum_{h_1+\cdots+h_{K_0}=n}\frac{n!}{h_1!\cdots h_{K_0}!}\prod_{k=1}^{K_0}\left(\frac{h_k}{n}\right)^{h_k}\prod_{i=1}^{m}C_{\mathrm{MN}}(K_i,h_k).\end{aligned} \quad (27)$$

To simplify notations, from now on we write $C(\mathcal{M}(K_0,K_1,\ldots,K_m),n)$ in an abbreviated form $C_{\mathrm{NB}}(K_0,n)$.

1: Count the frequencies $h_1, \ldots, h_K$ from the data $\mathbf{x}^n$
2: Compute the likelihood $P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(K))) = \prod_{k=1}^{K} (h_k/n)^{h_k}$
3: Set $\mathcal{C}_{\mathrm{MN}}(1, n) = 1$
4: Compute $\mathcal{C}_{\mathrm{MN}}(2, n) = \sum_{r_1+r_2=n} (n!/r_1!r_2!)(r_1/n)^{r_1}(r_2/n)^{r_2}$
5: **for** $k = 1$ to $K - 2$ **do**
6: Compute $\mathcal{C}_{\mathrm{MN}}(k+2, n) = \mathcal{C}_{\mathrm{MN}}(k+1, n) + (n/k) \cdot \mathcal{C}_{\mathrm{MN}}(k, n)$
7: **end for**
8: Output $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(K)) = P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(K)))/\mathcal{C}_{\mathrm{MN}}(K, n)$

Algorithm 1: The linear-time algorithm for computing $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(K))$.

4.2. The quadratic-time algorithm

It turns out [16] that the recursive formula (15) can be generalized to the naive Bayes model family case.

Theorem 2. *The terms $\mathcal{C}_{NB}(K_0, n)$ satisfy the recurrence*

$$\mathcal{C}_{NB}(K_0, n) = \sum_{r_1+r_2=n} \frac{n!}{r_1!r_2!} \left(\frac{r_1}{n}\right)^{r_1} \left(\frac{r_2}{n}\right)^{r_2} \cdot \mathcal{C}_{NB}(K^*, r_1) \cdot \mathcal{C}_{NB}(K_0 - K^*, r_2), \tag{28}$$

where $K^* = 1, \ldots, K_0 - 1$.

Proof. See the appendix. □

In many practical applications of the naive Bayes, the quantity K_0 is unknown. Its value is typically determined as a part of the model class selection process. Consequently, it is necessary to compute NML for model classes $\mathcal{M}(K_0, K_1, \ldots, K_m)$, where K_0 has a range of values, say, $K_0 = 1, \ldots, K_{\max}$. The process of computing NML for this case is described in Algorithm 2. The time complexity of the algorithm is $\mathcal{O}(n^2 \cdot K_{\max})$. If the value of K_0 is fixed, the time complexity drops to $\mathcal{O}(n^2 \cdot \log K_0)$. See [16] for more details.

5. NML FOR BAYESIAN FORESTS

The naive Bayes model discussed in the previous section has been successfully applied in various domains. In this section we consider, tree-structured Bayesian networks, which include the naive Bayes model as a special case but can also represent more complex dependencies.

5.1. The model family

As before, we assume m variables $X_1, \ldots, X_m$ with given value cardinalities $K_1, \ldots, K_m$. Since the goal here is to model the *joint* probability distribution of the m variables, there is no need to mark a special variable. We assume a data matrix $\mathbf{x}^n = (x_{ji}) \in \mathcal{X}^n$, $1 \le j \le n$, and $1 \le i \le m$, as given.

A Bayesian network structure $\mathcal{G}$ encodes independence assumptions so that if each variable X_i is represented as a node in the network, then the joint probability distribution factorizes into a product of local probability distributions, one for each node, conditioned on its parent set. We define a *Bayesian forest* to be a Bayesian network structure $\mathcal{G}$ on the node set $X_1, \ldots, X_m$ which assigns at most one parent $X_{\mathrm{pa}(i)}$ to any node X_i. Consequently, a *Bayesian tree* is a connected Bayesian forest and a Bayesian forest breaks down into component trees, that is, connected subgraphs. The root of each such component tree lacks a parent, in which case we write $\mathrm{pa}(i) = \varnothing$.

The parent set of a node X_i thus reduces to a single value $\mathrm{pa}(i) \in \{1, \ldots, i-1, i+1, \ldots, m, \varnothing\}$. Let further $\mathrm{ch}(i)$ denote the set of children of node X_i in $\mathcal{G}$ and $\mathrm{ch}(\varnothing)$ denote the "children of none," that is, the roots of the component trees of $\mathcal{G}$.

The corresponding model family $\mathcal{F}_{\mathrm{BF}}$ can be indexed by the network structure $\mathcal{G}$ and the corresponding attribute value counts $K_1, \ldots, K_m$:

$$\mathcal{F}_{\mathrm{BF}} = \{\mathcal{M}(\boldsymbol{\varphi}) : \boldsymbol{\varphi} \in \Phi_{\mathrm{BF}}\} \tag{29}$$

with $\Phi_{\mathrm{BF}} = \{1, \ldots, |\mathcal{G}|\} \times \{1, 2, 3, \ldots\}^m$, where $\mathcal{G}$ is associated with an integer according to some enumeration of all Bayesian forests on $(X_1, \ldots, X_m)$. As the K_i are assumed fixed, we can abbreviate the corresponding model classes by $\mathcal{M}(\mathcal{G}) := \mathcal{M}(\mathcal{G}, K_1, \ldots, K_m)$.

Given a forest model class $\mathcal{M}(\mathcal{G})$, we index each model by a parameter vector $\boldsymbol{\theta}$ in the corresponding parameter space $\Theta_{\mathcal{G}}$:

$$\Theta_{\mathcal{G}} = \left\{ \boldsymbol{\theta} = (\theta_{ikl}) : \theta_{ikl} \ge 0, \ \sum_{l} \theta_{ikl} = 1, \ i = 1, \ldots, m, \ k = 1, \ldots, K_{\mathrm{pa}(i)}, \ l = 1, \ldots, K_i \right\}, \tag{30}$$

where we define $K_{\varnothing} := 1$ in order to unify notation for root and non-root nodes. Each such θ_{ikl} defines a probability

$$\theta_{ikl} = P(X_i = l \mid X_{\mathrm{pa}(i)} = k, \mathcal{M}(\mathcal{G}), \boldsymbol{\theta}), \tag{31}$$

where we interpret $X_{\varnothing} = 1$ as a null condition.

The joint probability that a model $M = (\mathcal{G}, \boldsymbol{\theta})$ assigns to a data vector $\mathbf{x} = (x_1, \ldots, x_m)$ becomes

$$P(\mathbf{x} \mid \mathcal{M}(\mathcal{G}), \boldsymbol{\theta}) = \prod_{i=1}^{m} P(X_i = x_i \mid X_{\mathrm{pa}(i)} = x_{\mathrm{pa}(i)}, \mathcal{M}(\mathcal{G}), \boldsymbol{\theta}) = \prod_{i=1}^{m} \theta_{i, x_{\mathrm{pa}(i)}, x_i}. \tag{32}$$

1: Compute $\mathcal{C}_{\mathrm{MN}}(k, j)$ for $k = 1, \ldots, V_{\max}$, $j = 0, \ldots, n$, where $V_{\max} = \max\{K_1, \ldots, K_m\}$
2: **for** $K_0 = 1$ to $K_{\max}$ **do**
3: Count the frequencies $h_1, \ldots, h_{K_0}, f_{ik1}, \ldots, f_{ikK_i}$ for $i = 1, \ldots, m$, $k = 1, \ldots, K_0$ from the data $\mathbf{x}^n$
4: Compute the likelihood:
$P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(K_0, K_1, \ldots, K_m))) = \prod_{k=1}^{K_0} (h_k/n)^{h_k} \prod_{i=1}^{m} \prod_{l=1}^{K_i} (f_{ikl}/h_k)^{f_{ikl}}$
5: Set $\mathcal{C}_{\mathrm{NB}}(K_0, 0) = 1$
6: **if** $K_0 = 1$ **then**
7: Compute $\mathcal{C}_{\mathrm{NB}}(1, j) = \prod_{i=1}^{m} \mathcal{C}_{\mathrm{MN}}(K_i, j)$ for $j = 1, \ldots, n$
8: **else**
9: Compute $\mathcal{C}_{\mathrm{NB}}(K_0, j) = \sum_{r_1+r_2=j} (j!/r_1!r_2!)(r_1/j)^{r_1}(r_2/j)^{r_2} \cdot \mathcal{C}_{\mathrm{NB}}(1, r_1) \cdot \mathcal{C}_{\mathrm{NB}}(K_0 - 1, r_2)$ for $j = 1, \ldots, n$
10: **end if**
11: Output $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(K_0, K_1, \ldots, K_m)) = P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(K_0, K_1, \ldots, K_m)))/\mathcal{C}_{\mathrm{NB}}(K_0, n)$
12: **end for**

Algorithm 2: The algorithm for computing $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(K_0, K_1, \ldots, K_m))$ for $K_0 = 1, \ldots, K_{\max}$.

For a sample $\mathbf{x}^n = (x_{ji})$ of n vectors $\mathbf{x}_j$, we define the corresponding frequencies as

$$f_{ikl} := \left|\{j : x_{ji} = l \wedge x_{j,\mathrm{pa}(i)} = k\}\right|, \qquad f_{il} := \left|\{j : x_{ji} = l\}\right| = \sum_{k=1}^{K_{\mathrm{pa}(i)}} f_{ikl}. \tag{33}$$

By definition, for any component tree root X_i, we have $f_{il} = f_{i1l}$. The probability assigned to a sample $\mathbf{x}^n$ can then be written as

$$P(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G}), \boldsymbol{\theta}) = \prod_{i=1}^{m} \prod_{k=1}^{K_{\mathrm{pa}(i)}} \prod_{l=1}^{K_i} \theta_{ikl}^{f_{ikl}}, \tag{34}$$

which is maximized at

$$\hat{\theta}_{ikl}(\mathbf{x}^n, \mathcal{M}(\mathcal{G})) = \frac{f_{ikl}}{f_{\mathrm{pa}(i),k}}, \tag{35}$$

where we define $f_{\varnothing,1} := n$. The maximum data likelihood thereby is

$$\hat{P}(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G})) = \prod_{i=1}^{m} \prod_{k=1}^{K_{\mathrm{pa}(i)}} \prod_{l=1}^{K_i} \left(\frac{f_{ikl}}{f_{\mathrm{pa}(i),k}}\right)^{f_{ikl}}. \tag{36}$$

5.2. The algorithm

The goal is to calculate the NML distribution $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G}))$ defined in (3). This consists of calculating the maximum data likelihood (36) and the normalizing term $\mathcal{C}(\mathcal{M}(\mathcal{G}), n)$ given in (4). The former involves frequency counting, one sweep through the data, and multiplication of the appropriate values. This can be done in time $\mathcal{O}(n + \sum_i K_i K_{\mathrm{pa}(i)})$. The latter involves a sum exponential in n, which clearly makes it the computational bottleneck of the algorithm.

Our approach is to break up the normalizing sum in (4) into terms corresponding to subtrees with given frequencies in either their root or its parent. We then calculate the complete sum by sweeping through the graph once, bottom-up. Let us now introduce some necessary notation.

Let $\mathcal{G}$ be a given Bayesian forest. Then for any node X_i denote the subtree rooting in X_i, by $\mathcal{G}_{\mathrm{sub}(i)}$ and the forest built up by all descendants of X_i by $\mathcal{G}_{\mathrm{dsc}(i)}$. The corresponding data domains are $\mathcal{X}_{\mathrm{sub}(i)}$ and $\mathcal{X}_{\mathrm{dsc}(i)}$, respectively. Denote the sum over all n-instantiations of a subtree by

$$\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n) := \sum_{\mathbf{x}^n_{\mathrm{sub}(i)} \in \mathcal{X}^n_{\mathrm{sub}(i)}} P(\mathbf{x}^n_{\mathrm{sub}(i)} \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{\mathrm{sub}(i)}), \mathcal{M}(\mathcal{G}_{\mathrm{sub}(i)})) \tag{37}$$

and for any vector $\mathbf{x}_i^n \in X_i^n$ with frequencies $\mathbf{f}_i = (f_{i1}, \ldots, f_{iK_i})$, we define

$$\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i) := \sum_{\mathbf{x}^n_{\mathrm{dsc}(i)} \in \mathcal{X}^n_{\mathrm{dsc}(i)}} P(\mathbf{x}^n_{\mathrm{dsc}(i)}, \mathbf{x}_i^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{\mathrm{dsc}(i)}, \mathbf{x}_i^n), \mathcal{M}(\mathcal{G}_{\mathrm{sub}(i)})) \tag{38}$$

to be the corresponding sum with fixed root instantiation, summing only over the attribute space spanned by the descendants on X_i.

Note that we use $\mathbf{f}_i$ on the left-hand side, and $\mathbf{x}_i^n$ on the right-hand side of the definition. This needs to be justified. Interestingly, while the terms in the sum depend on the ordering of $\mathbf{x}_i^n$, the sum itself depends on $\mathbf{x}_i^n$ only through its frequencies $\mathbf{f}_i$. To see this pick, any two representatives $\mathbf{x}_i^n$ and $\bar{\mathbf{x}}_i^n$ of $\mathbf{f}_i$ and find, for example, after lexicographical ordering of the elements, that

$$\left\{\left(\mathbf{x}_i^n, \mathbf{x}^n_{\mathrm{dsc}(i)}\right) : \mathbf{x}^n_{\mathrm{dsc}(i)} \in \mathcal{X}^n_{\mathrm{dsc}(i)}\right\} = \left\{\left(\bar{\mathbf{x}}_i^n, \mathbf{x}^n_{\mathrm{dsc}(i)}\right) : \mathbf{x}^n_{\mathrm{dsc}(i)} \in \mathcal{X}^n_{\mathrm{dsc}(i)}\right\}. \tag{39}$$

Next, we need to define corresponding sums over $\mathcal{X}_{\mathrm{sub}(i)}$ with the frequencies at the subtree root parent $X_{\mathrm{pa}(i)}$ given.

For any $\mathbf{f}_{\mathrm{pa}(i)} \sim \mathbf{x}^n_{\mathrm{pa}(i)} \in X^n_{\mathrm{pa}(i)}$ define

$$\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_{\mathrm{pa}(i)}) := \sum_{\mathbf{x}^n_{\mathrm{sub}(i)} \in \mathcal{X}^n_{\mathrm{sub}(i)}} P\left(\mathbf{x}^n_{\mathrm{sub}(i)} \mid \mathbf{x}^n_{\mathrm{pa}(i)}, \hat{\boldsymbol{\theta}}\left(\mathbf{x}^n_{\mathrm{sub}(i)}, \mathbf{x}^n_{\mathrm{pa}(i)}\right), \mathcal{M}\left(\mathcal{G}_{\mathrm{sub}(i)}\right)\right). \tag{40}$$

Again, this is well defined since any other representative $\bar{\boldsymbol{x}}^n_{\mathrm{pa}(i)}$ of $\mathbf{f}_{\mathrm{pa}(i)}$ yields summing the same terms modulo their ordering.

After having introduced this notation, we now briefly outline the algorithm and in the following subsections give a more detailed description of the steps involved. As stated before, we go through $\mathcal{G}$ bottom-up. At each inner node X_i, we receive $\mathcal{L}_j(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i)$ from each child X_j, $j \in \mathrm{ch}(i)$. Correspondingly, we are required to send $\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_{\mathrm{pa}(i)})$ up to the parent $X_{\mathrm{pa}(i)}$. At each component tree root X_i, we then calculate the sum $\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n)$ for the whole connectivity component and then combine these sums to get the normalizer $\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n)$ for the complete forest $\mathcal{G}$.

5.2.1. Leaves

For a leaf node X_i we can calculate the $\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_{\mathrm{pa}(i)})$ without listing its own frequencies $\mathbf{f}_i$. As in (27), $\mathbf{f}_{\mathrm{pa}(i)}$ splits the n data vectors into $K_{\mathrm{pa}(i)}$ subsets of sizes $f_{\mathrm{pa}(i),1}, \ldots, f_{\mathrm{pa}(i),K_{\mathrm{pa}(i)}}$ and each of them can be modeled independently as a multinomial; we have

$$\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_{\mathrm{pa}(i)}) = \prod_{k=1}^{K_{\mathrm{pa}(i)}} \mathcal{C}_{\mathrm{MN}}(K_i, f_{\mathrm{pa}(i),k}). \tag{41}$$

The terms $\mathcal{C}_{\mathrm{MN}}(K_i, n')$ (for $n' = 0, \ldots, n$) can be precalculated using recurrence (19) as in Algorithm 1.

5.2.2. Inner nodes

For inner nodes X_i we divide the task into two steps. First, we collect the child messages $\mathcal{L}_j(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i)$ sent by each child $X_j \in \mathrm{ch}(i)$ into partial sums $\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i)$ over $\mathcal{X}_{\mathrm{dsc}(i)}$, and then "lift" these to sums $\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_{\mathrm{pa}(i)})$ over $\mathcal{X}_{\mathrm{sub}(i)}$ which are the messages to the parent.

The first step is simple. Given an instantiation $\mathbf{x}^n_i$ at X_i or, equivalently, the corresponding frequencies $\mathbf{f}_i$, the subtrees rooting in the children $\mathrm{ch}(i)$ of X_i become independent of each other. Thus we have

$$\begin{aligned}
&\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i) \\
&\quad = \sum_{\mathbf{x}^n_{\mathrm{dsc}(i)} \in \mathcal{X}^n_{\mathrm{dsc}(i)}} P(\mathbf{x}^n_{\mathrm{dsc}(i)}, \mathbf{x}^n_i \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{\mathrm{dsc}(i)}, \mathbf{x}^n_i), \mathcal{M}(\mathcal{G}_{\mathrm{sub}(i)})) \qquad (42) \\
&\quad = P(\mathbf{x}^n_i \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{\mathrm{dsc}(i)}, \mathbf{x}^n_i), \mathcal{M}(\mathcal{G}_{\mathrm{sub}(i)})) \\
&\qquad \times \left(\sum_{\mathbf{x}^n_{\mathrm{dsc}(i)} \in \mathcal{X}^n_{\mathrm{dsc}(i)}} \prod_{j \in \mathrm{ch}(i)} P(\mathbf{x}^n_{\mathrm{dsc}(i)|\mathrm{sub}(j)} \mid \mathbf{x}^n_i, \right. \qquad (43) \\
&\qquad \left. \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{\mathrm{dsc}(i)}, \mathbf{x}^n_i), \mathcal{M}(\mathcal{G}_{\mathrm{sub}(i)})) \right) \\
&\quad = P(\mathbf{x}^n_i \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{\mathrm{dsc}(i)}, \mathbf{x}^n_i), \mathcal{M}(\mathcal{G}_{\mathrm{sub}(i)})) \\
&\qquad \times \prod_{j \in \mathrm{ch}(i)} \left(\sum_{\mathbf{x}^n_{\mathrm{sub}(j)} \in \mathcal{X}^n_{\mathrm{sub}(j)}} P(\mathbf{x}^n_{\mathrm{sub}(j)} \mid \mathbf{x}^n_i, \right. \qquad (44) \\
&\qquad \left. \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{\mathrm{dsc}(i)}, \mathbf{x}^n_i), \mathcal{M}(\mathcal{G}_{\mathrm{sub}(i)})) \right) \\
&\quad = \prod_{l=1}^{K_i} \left(\frac{f_{il}}{n}\right)^{f_{il}} \prod_{j \in \mathrm{ch}(i)} \mathcal{L}_j(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i), \qquad (45)
\end{aligned}$$

where $\mathbf{x}^n_{\mathrm{dsc}(i)|\mathrm{sub}(j)}$ is the restriction of $\mathbf{x}_{\mathrm{dsc}(i)}$ to columns corresponding to nodes in $\mathcal{G}_j$. We have used (38) for (42), (32) for (43) and (44), and finally (36) and (40) for (45).

Now we need to calculate the outgoing messages $\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_{\mathrm{pa}(i)})$ from the incoming messages we have just combined into $\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i)$. This is the most demanding part of the algorithm, for we need to list all possible conditional frequencies, of which there are $\mathcal{O}(n^{K_i K_{\mathrm{pa}(i)} - 1})$ many, the -1 being due to the sum-to-n constraint. For fixed i, we arrange the conditional frequencies f_{ikl} into a matrix $\mathbf{F} = (f_{ikl})$ and define its marginals

$$\begin{aligned}
\boldsymbol{\rho}(\mathbf{F}) &:= \left(\sum_k f_{ik1}, \ldots, \sum_k f_{ikK_i} \right), \\
\boldsymbol{\gamma}(\mathbf{F}) &:= \left(\sum_l f_{i1l}, \ldots, \sum_l f_{iK_{\mathrm{pa}(i)}l} \right)
\end{aligned} \tag{46}$$

to be the vectors obtained by summing the rows of $\mathbf{F}$ and the columns of $\mathbf{F}$, respectively. Each such matrix then corresponds to a term $\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n \mid \boldsymbol{\rho}(\mathbf{F}))$ and a term $\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \boldsymbol{\gamma}(\mathbf{F}))$. Formally, we have

$$\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_{\mathrm{pa}(i)}) = \sum_{\mathbf{F}:\boldsymbol{\gamma}(\mathbf{F}) = \mathbf{f}_{\mathrm{pa}(i)}} \mathcal{C}_i(\mathcal{M}(\mathcal{G}), n \mid \boldsymbol{\rho}(\mathbf{F})). \tag{47}$$

5.2.3. Component tree roots

For a component tree root $X_i \in \mathrm{ch}(\varnothing)$ we do not need to pass any message upward. All we need is the complete sum over the component tree

$$\mathcal{C}_i(\mathcal{M}_{\mathcal{G}}, n) = \sum_{\mathbf{f}_i} \frac{n!}{f_{i1}! \cdots f_{iK_i}!} \mathcal{C}_i(\mathcal{M}_{\mathcal{G}}, n \mid \mathbf{f}_i), \tag{48}$$

where the $\mathcal{C}_i(\mathcal{M}_{\mathcal{G}}, n \mid \mathbf{f}_i)$ are calculated from (45). The summation goes over all nonnegative integer vectors $\mathbf{f}_i$ summing to n. The above is trivially true since we sum over all instantiations $\mathbf{x}_i$ of X_i and group like terms, corresponding to the same frequency vector $\mathbf{f}_i$, while keeping track of their respective count, namely $n!/f_{i1}! \cdots f_{iK_i}!$.

5.2.4. The algorithm

For the complete forest $\mathcal{G}$ we simply multiply the sums over its tree components. Since these are independent of each

1: Count all frequencies f_{ikl} and f_{il} from the data $\mathbf{x}^n$
2: Compute $\hat{P}(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G})) = \prod_{i=1}^{m} \prod_{k=1}^{K_{\mathrm{pa}(i)}} \prod_{l=1}^{K_i} (f_{ikl}/f_{\mathrm{pa}(i),k})^{f_{ikl}}$
3: **for** $k = 1, \ldots, K_{\max} := \max_{i:X_i \text{ is a leaf}} \{K_i\}$ and $n' = 0, \ldots, n$ **do**
4: Compute $\mathcal{C}_{\mathrm{MN}}(k, n')$ as in Algorithm 1
5: **end for**
6: **for** each node X_i in some bottom-up order **do**
7: **if** X_i is a leaf **then**
8: **for** each frequency vector $\mathbf{f}_{\mathrm{pa}(i)}$ of $X_{\mathrm{pa}(i)}$ **do**
9: Compute $\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_{\mathrm{pa}(i)}) = \prod_{k=1}^{K_{\mathrm{pa}(i)}} \mathcal{C}_{\mathrm{MN}}(K_i, \mathbf{f}_{\mathrm{pa}(i)k})$
10: **end for**
11: **else if** X_i is an inner node **then**
12: **for** each frequency vector $\mathbf{f}_i X_i$ **do**
13: Compute $\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i) = \prod_{l=1}^{K_i} (f_{il}/n)^{f_{il}} \prod_{j \in \mathrm{ch}(i)} \mathcal{L}_j(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i)$
14: **end for**
15: initialize $\mathcal{L}_i \equiv 0$
16: **for** each non-negative $K_i \times K_{\mathrm{pa}(i)}$ integer matrix $\mathbf{F}$ with entries summing to n **do**
17: $\mathcal{L}_i(\mathcal{M}(\mathcal{G}), n \mid \boldsymbol{\gamma}(\mathbf{F})) \mathrel{+}= \mathcal{C}_i(\mathcal{M}(\mathcal{G}), n \mid \boldsymbol{\rho}(\mathbf{F}))$
18: **end for**
19: **else if** X_i is a component tree root **then**
20: Compute $\mathcal{C}_i(\mathcal{M}(\mathcal{G}), n) = \sum_{\mathbf{f}_i} \prod_{l=1}^{K_i} (f_{il}/n)^{f_{il}} \prod_{j \in \mathrm{ch}(i)} \mathcal{L}_j(\mathcal{M}(\mathcal{G}), n \mid \mathbf{f}_i)$
21: **end if**
22: **end for**
23: Compute $\mathcal{C}(\mathcal{M}(\mathcal{G}), n) = \prod_{i \in \mathrm{ch}(\varnothing)} \mathcal{C}_i(\mathcal{M}(\mathcal{G}), n)$
24: Outpute $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G})) = \hat{P}(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G}))/\mathcal{C}(\mathcal{M}(\mathcal{G}), n)$

Algorithm 3: The algorithm for computing $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G}))$ for a Bayesian forest $\mathcal{G}$.

other, in analogy to (42)–(45) we have

$$\mathcal{C}(\mathcal{M}_{\mathcal{G}}, n) = \prod_{i \in \mathrm{ch}(\varnothing)} \mathcal{C}_i(\mathcal{M}_{\mathcal{G}}, n). \tag{49}$$

Algorithm 3 collects all the above into a pseudocode.

The time complexity of this algorithm is $\mathcal{O}(n^{K_i K_{\mathrm{pa}(i)} - 1})$ for each inner node, $\mathcal{O}(n(n + K_i))$ for each leaf, and $\mathcal{O}(n^{K_i - 1})$ for a component tree root of $\mathcal{G}$. When all $m' < m$ inner nodes are binary, it runs in $\mathcal{O}(m' n^3)$, independently of the number of values of the leaf nodes. This is polynomial with respect to the sample size n, while applying (4) directly for computing $\mathcal{C}(\mathcal{M}(\mathcal{G}), n)$ requires exponential time. The order of the polynomial depends on the attribute cardinalities: the algorithm is exponential with respect to the number of values a non-leaf variable can take.

Finally, note that we can speed up the algorithm when $\mathcal{G}$ contains multiple copies of some subtree. Also we have $\mathcal{C}_i/\mathcal{L}_i(\mathcal{M}_{\mathcal{G}}, n \mid \mathbf{f}_i) = \mathcal{C}_i/\mathcal{L}_i(\mathcal{M}_{\mathcal{G}}, n \mid \pi(\mathbf{f}_i))$ for any permutation π of the entries of $\mathbf{f}_i$. However, this does not lead to considerable gain, at least in order of magnitude. Also, we can see that in line 16 of the algorithm we enumerate all frequency matrices $\mathbf{F}$, while in line 17 we sum the same terms whenever the marginals of $\mathbf{F}$ are the same. Unfortunately, computing the number of non-negative integer matrices with given marginals is a #P-hard problem already when the other matrix dimension is fixed to 2, as proven in [33]. This suggests that for this task there may not exist an algorithm that is polynomial in all input quantities. The algorithm presented here is polynomial as well in the sample size n as in the graph size m. For attributes with relatively few values, the polynomial is time tolerable.

6. CONCLUSION

The normalized maximum likelihood (NML) offers a universal, minimax optimal approach to statistical modeling. In this paper, we have surveyed efficient algorithms for computing the NML in the case of discrete datasets. The model families used in our work are Bayesian networks of varying complexity. The simplest model we discussed is the multinomial model family, which can be applied to problems related to density estimation or discretization. In this case, the NML can be computed in linear time. The same result also applies to a network of independent multinomial variables, that is, a Bayesian network with no arcs.

For the naive Bayes model family, the NML can be computed in quadratic time. Models of this type have been used extensively in clustering or classification domains with good results. Finally, to be able to represent more complex dependencies between the problem domain variables, we also considered tree-structured Bayesian networks. We showed how to compute the NML in this case in polynomial time with respect to the sample size, but the order of the polynomial depends on the number of values of the domain variables, which makes our result impractical for some domains.

The methods presented are especially suitable for problems in bioinformatics, which typically involve multidimensional discrete datasets. Furthermore, unlike the Bayesian methods, information-theoretic approaches such as ours do not require a prior for the model parameters. This is the most important aspect, as constructing a reasonable parameter prior is a notoriously difficult problem, particularly in bioinformatical domains involving novel types of data with little background knowledge. All in all, information theory has been found to offer a natural and successful theoretical framework for biological applications in general, which makes NML an appealing choice for bioinformatics.

In the future, our plan is to extend the current work to more complex cases such as general Bayesian networks, which would allow the use of NML in even more involved modeling tasks. Another natural area of future work is to apply the methods of this paper to practical tasks involving large discrete databases and compare the results to other approaches, such as those based on Bayesian statistics.

APPENDIX

PROOFS OF THEOREMS

In this section, we provide detailed proofs of two theorems presented in the paper.

Proof of Theorem 1 (multinomial recursion)

We start by proving the following lemma.

Lemma 3. *For the tree function $T(z)$ we have*

$$zT'(z) = \frac{T(z)}{1 - T(z)}. \tag{A.1}$$

Proof. A basic property of the tree function is the functional equation $T(z) = ze^{T(z)}$ (see, e.g., [23]). Differentiating this equation yields

$$\begin{aligned} T'(z) &= e^{T(z)} + T(z)T'(z) \\ zT'(z)(1 - T(z)) &= ze^{T(z)}, \end{aligned} \tag{A.2}$$

from which (A.1) follows. □

Now we can proceed to the proof of the theorem. We start by multiplying and differentiating (17) as follows:

$$z \cdot \frac{d}{dz} \sum_{n \geq 0} \frac{n^n}{n!} \mathcal{C}_{\mathrm{MN}}(K, n) z^n = z \cdot \sum_{n \geq 1} n \cdot \frac{n^n}{n!} \mathcal{C}_{\mathrm{MN}}(K, n) z^{n-1} \tag{A.3}$$

$$= \sum_{n \geq 0} n \cdot \frac{n^n}{n!} \mathcal{C}_{\mathrm{MN}}(K, n) z^n. \tag{A.4}$$

On the other hand, by manipulating (18) in the same way, we get

$$z \cdot \frac{d}{dz} \frac{1}{(1 - T(z))^K} \tag{A.5}$$

$$= \frac{z \cdot K}{(1 - T(z))^{K+1}} \cdot T'(z) \tag{A.6}$$

$$= \frac{K}{(1 - T(z))^{K+1}} \cdot \frac{T(z)}{1 - T(z)} \tag{A.7}$$

$$= K \left(\sum_{n \geq 0} \frac{n^n}{n!} \mathcal{C}_{\mathrm{MN}}(K + 2, n) z^n - \sum_{n \geq 0} \frac{n^n}{n!} \mathcal{C}_{\mathrm{MN}}(K + 1, n) z^n \right), \tag{A.8}$$

where (A.6) follows from Lemma 3. Comparing the coefficients of z^n in (A.4) and (A.8), we get

$$n \cdot \mathcal{C}_{\mathrm{MN}}(K, n) = K \cdot \left(\mathcal{C}_{\mathrm{MN}}(K + 2, n) - \mathcal{C}_{\mathrm{MN}}(K + 1, n) \right), \tag{A.9}$$

from which the theorem follows. □

Proof of Theorem 2 (naive Bayes recursion)

We have

$$\begin{aligned} &\mathcal{C}_{\mathrm{NB}}(K_0, n) \\ &= \sum_{h_1 + \cdots + h_{K_0} = n} \frac{n!}{h_1! \cdots h_{K_0}!} \prod_{k=1}^{K_0} \left(\frac{h_k}{n} \right)^{h_k} \prod_{i=1}^{m} \mathcal{C}_{\mathrm{MN}}(K_i, h_k) \\ &= \sum_{h_1 + \cdots + h_{K_0} = n} \frac{n!}{n^n} \prod_{k=1}^{K_0} \frac{h_k^{h_k}}{h_k!} \prod_{i=1}^{m} \mathcal{C}_{\mathrm{MN}}(K_i, h_k) \\ &= \sum_{\substack{h_1 + \cdots + h_{K^*} = r_1 \\ h_{K^*+1} + \cdots + h_{K_0} = r_2 \\ r_1 + r_2 = n}} \frac{n!}{n^n} \frac{r_1^{r_1}}{r_1!} \frac{r_2^{r_2}}{r_2!} \left(\frac{r_1!}{r_1^{r_1}} \prod_{k=1}^{K^*} \frac{h_k^{h_k}}{h_k!} \cdot \frac{r_2!}{r_2^{r_2}} \prod_{k=K^*+1}^{K_0} \frac{h_k^{h_k}}{h_k!} \right) \\ &\qquad \cdot \prod_{i=1}^{m} \prod_{k=1}^{K^*} \mathcal{C}_{\mathrm{MN}}(K_i, h_k) \prod_{k=K^*+1}^{K_0} \mathcal{C}_{\mathrm{MN}}(K_i, h_k) \\ &= \sum_{\substack{h_1 + \cdots + h_{K^*} = r_1 \\ h_{K^*+1} + \cdots + h_{K_0} = r_2 \\ r_1 + r_2 = n}} \frac{n!}{r_1! r_2!} \left(\frac{r_1}{n} \right)^{r_1} \left(\frac{r_2}{n} \right)^{r_2} \\ &\qquad \cdot \left(\frac{r_1!}{h_1! \cdots h_{K^*}!} \prod_{k=1}^{K^*} \left(\frac{h_k}{r_1} \right)^{h_k} \prod_{i=1}^{m} \mathcal{C}_{\mathrm{MN}}(K_i, h_k) \right) \\ &\qquad \cdot \left(\frac{r_2!}{h_{K^*+1}! \cdots h_{K_0}!} \prod_{k=K^*+1}^{K_0} \left(\frac{h_k}{r_2} \right)^{h_k} \prod_{i=1}^{m} \mathcal{C}_{\mathrm{MN}}(K_i, h_k) \right) \\ &= \sum_{r_1 + r_2 = n} \frac{n!}{r_1! r_2!} \left(\frac{r_1}{n} \right)^{r_1} \left(\frac{r_2}{n} \right)^{r_2} \cdot \mathcal{C}_{\mathrm{NB}}(K^*, r_1) \cdot \mathcal{C}_{\mathrm{NB}}(K_0 - K^*, r_2), \end{aligned} \tag{A.10}$$

and the proof follows.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and Jorma Rissanen for useful comments. This work was supported in part by the Academy of Finland under the project Civi and by the Finnish Funding Agency for Technology and Innovation under the projects Kukot and PMMA. In addition, this work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

REFERENCES

[1] G. Korodi and I. Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression," *ACM Transactions on Information Systems*, vol. 23, no. 1, pp. 3–34, 2005.

[2] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and B. Brown, "Clustering methods for the analysis of DNA microarray data," Tech. Rep., Department of Health Research and Policy, Stanford University, Stanford, Calif, USA, 1999.

[3] W. Pan, J. Lin, and C. T. Le, "Model-based cluster analysis of microarray gene-expression data," *Genome Biology*, vol. 3, no. 2, pp. 1–8, 2002.

[4] G. J. McLachlan, R. W. Bean, and D. Peel, "A mixture model-based approach to the clustering of microarray expression data," *Bioinformatics*, vol. 18, no. 3, pp. 413–422, 2002.

[5] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young, "Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks," in *Proceedings of the 6th Pacific Symposium on Biocomputing (PSB '01)*, pp. 422–433, The Big Island of Hawaii, Hawaii, USA, January 2001.

[6] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[7] J. Rissanen, "Stochastic complexity," *Journal of the Royal Statistical Society, Series B*, vol. 49, no. 3, pp. 223–239, 1987, with discussions, 223–265.

[8] J. Rissanen, "Fisher information and stochastic complexity," *IEEE Transactions on Information Theory*, vol. 42, no. 1, pp. 40–47, 1996.

[9] Yu M. Shtarkov, "Universal sequential coding of single messages," *Problems of Information Transmission*, vol. 23, no. 3, pp. 175–186, 1987.

[10] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2743–2760, 1998.

[11] J. Rissanen, "Strong optimality of the normalized ML models as universal codes and information in data," *IEEE Transactions on Information Theory*, vol. 47, no. 5, pp. 1712–1717, 2001.

[12] P. Grünwald, *The Minimum Description Length Principle*, The MIT Press, Cambridge, Mass, USA, 2007.

[13] J. Rissanen, *Information and Complexity in Statistical Modeling*, Springer, New York , NY, USA, 2007.

[14] D. Heckerman, "A tutorial on learning with Bayesian networks," Tech. Rep. MSR-TR-95-06, Microsoft Research, Advanced Technology Division, One Microsoft Way, Redmond, Wash, USA, 98052, 1996.

[15] P. Kontkanen and P. Myllymäki, "A linear-time algorithm for computing the multinomial stochastic complexity," *Information Processing Letters*, vol. 103, no. 6, pp. 227–233, 2007.

[16] P. Kontkanen, P. Myllymäki, W. Buntine, J. Rissanen, and H. Tirri, "An MDL framework for data clustering," in *Advances in Minimum Description Length: Theory and Applications*, P. Grünwald, I. J. Myung, and M. Pitt, Eds., The MIT Press, Cambridge, Mass, USA, 2006.

[17] Q. Xie and A. R. Barron, "Asymptotic minimax regret for data compression, gambling, and prediction," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 431–445, 2000.

[18] V. Balasubramanian, "MDL, Bayesian inference, and the geometry of the space of probability distributions," in *Advances in Minimum Description Length: Theory and Applications*, P. Grünwald, I. J. Myung, and M. Pitt, Eds., pp. 81–98, The MIT Press, Cambridge, Mass, USA, 2006.

[19] P. Kontkanen and P. Myllymäki, "MDL histogram density estimation," in *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, (AISTATS '07)*, San Juan, Puerto Rico, USA, March 2007.

[20] P. Kontkanen, W. Buntine, P. Myllymäki, J. Rissanen, and H. Tirri, "Efficient computation of stochastic complexity," in *Proceedings of the 9th International Conference on Artificial Intelligence and Statistics*, C. Bishop and B. Frey, Eds., pp. 233–238, Society for Artificial Intelligence and Statistics, Key West, Fla, USA, January 2003.

[21] M. Koivisto, "Sum-Product Algorithms for the Analysis of Genetic Risks," Tech. Rep. A-2004-1, Department of Computer Science, University of Helsinki, Helsinki, Finland, 2004.

[22] P. Kontkanen and P. Myllymäki, "A fast normalized maximum likelihood algorithm for multinomial data," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI '05)*, Edinburgh, Scotland, August 2005.

[23] D. E. Knuth and B. Pittle, "A recurrence related to trees," *Proceedings of the American Mathematical Society*, vol. 105, no. 2, pp. 335–349, 1989.

[24] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the Lambert W function," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 329–359, 1996.

[25] W. Szpankowski, *Average Case Analysis of Algorithms on Sequences*, John Wiley & Sons, New York, NY, USA, 2001.

[26] P. Flajolet and A. M. Odlyzko, "Singularity analysis of generating functions," *SIAM Journal on Discrete Mathematics*, vol. 3, no. 2, pp. 216–240, 1990.

[27] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[28] P. Kontkanen, P. Myllymäki, and H. Tirri, "Constructing Bayesian finite mixture models by the EM algorithm," Tech. Rep. NC-TR-97-003, ESPRIT Working Group on Neural and Computational Learning (NeuroCOLT), Helsinki, Finland, 1997.

[29] P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri, "On Bayesian case matching," in *Proceedings of the 4th European Workshop Advances in Case-Based Reasoning (EWCBR '98)*, B. Smyth and P. Cunningham, Eds., vol. 1488 of *Lecture Notes In Computer Science*, pp. 13–24, Springer, Dublin, Ireland, September 1998.

[30] P. Grünwald, P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri, "Minimum encoding approaches for predictive modeling," in *Proceedings of the 14th International Conference on Uncertainty in Artificial Intelligence (UAI '98)*, G. Cooper and S. Moral, Eds., pp. 183–192, Morgan Kaufmann, Madison, Wis, USA, July 1998.

[31] P. Kontkanen, P. Myllymäki, T. Silander, H. Tirri, and P. Grünwald, "On predictive distributions and Bayesian networks," *Statistics and Computing*, vol. 10, no. 1, pp. 39–54, 2000.

[32] P. Kontkanen, J. Lahtinen, P. Myllymäki, T. Silander, and H. Tirri, "Supervised model-based visualization of high-dimensional data," *Intelligent Data Analysis*, vol. 4, no. 3-4, pp. 213–227, 2000.

[33] M. Dyer, R. Kannan, and J. Mount, "Sampling contingency tables," *Random Structures and Algorithms*, vol. 10, no. 4, pp. 487–506, 1997.