Fast NML Computation for Naive Bayes Models

Tommi Mononen and Petri Myllymäki

Complex Systems Computation Group (CoSCo)
Helsinki Institute for Information Technology (HIIT)
University of Helsinki & Helsinki University of Technology
P.O.Box 68 (Department of Computer Science)
FIN-00014 University of Helsinki, Finland
{Firstname}.{Lastname}@hiit.fi

Abstract. The *Minimum Description Length* (MDL) is an information-theoretic principle that can be used for model selection and other statistical inference tasks. One way to implement this principle in practice is to compute the *Normalized Maximum Likelihood* (NML) distribution for a given parametric model class. Unfortunately this is a computationally infeasible task for many model classes of practical importance. In this paper we present a fast algorithm for computing the NML for the Naive Bayes model class, which is frequently used in classification and clustering tasks. The algorithm is based on a relationship between powers of generating functions and discrete convolution. The resulting algorithm has the time complexity of $\mathcal{O}(n^2)$, where n is the size of the data.

1 Introduction

The information-theoretical *Minimum Description Length* (MDL) principle [15, 4, 17, 3] for model selection is based on the conceptually simple idea that given a data set, the best model for the data is the one which results in the shortest description for the data together with the model. Hence, we wish to select a model representing a balance between too simple models (in which case the code length for the data is large) and too complex models (in which case the code length for the data is small, but for the model itself large).

Consider a parametric probabilistic model class, i.e., a set of models each defining a probability distribution over all possible data sets. Let us call the shortest possible code length obtainable with the given set of models *stochastic complexity*. Consequently, given a data set, we can choose between alternative parametric models (model classes) with different number of parameters by comparing the corresponding stochastic complexities for the given data.

However, there remains the question of how to formally define the stochastic complexity for a model class. As each of the probability distributions in a probabilistic model class corresponds to a code length, it is obvious that no code can be shorter than all the other codes in the model class for all data sets, because no probability distribution can dominate another probability distribution over all data sets. A *universal model* is a model (code) which can imitate any model in a given parametric model class. The *normalized maximum likelihood (NML)* distribution [16, 18] is the *worst-case optimal* universal model giving a desired formal definition for the stochastic complexity (see the next Section).

Unfortunately, computing the NML is very difficult for many model classes of practical interest. In this paper we consider Bayesian networks, probabilistic model classes defined by acyclic directed graphs [14, 5]. The Naive Bayes model is a simple Bayesian network, which is continuously used with success in areas such as clustering and classification. It has been earlier shown [11] how to compute the NML for the Naive Bayes model family in $\mathcal{O}(n^2 \log L)$ time, where L denotes the number of values of the class variable of the Naive Bayes model. In this paper we introduce a faster $\mathcal{O}(n^2)$ algorithm for this task, based on generating functions.

2 The Problem

The normalized maximum likelihood (NML) distribution [16, 18] is defined by

$$P_{NML}(\mathbf{x}^n|\mathcal{M}) = \frac{P(\mathbf{x}^n|\hat{\theta}(\mathbf{x}^n, \mathcal{M}))}{\sum_{\mathbf{y}^n} P(\mathbf{y}^n|\hat{\theta}(\mathbf{y}^n, \mathcal{M}))}, \tag{1}$$

where the numerator is the maximum likelihood for the observed data $\mathbf{x}^n$ within the model class $\mathcal{M}$. The *normalizing term* in the denominator is the sum over maximum likelihoods of all possible data sets of size n, with respect to the model class. As shown in [18], this yields the worst case universal distribution with respect to the model class $\mathcal{M}$.

Although NML was defined as the worst-case optimal universal model, without considering model complexity regularization, it is interesting to note how it behaves as a model class selection criterion. Namely, if the model class is very complex, then the maximum likelihood for the given data (the numerator in (1)) is large, but so is also the denominator as a complex model gives a high maximum likelihood for many data sets. For simple model classes the sum in the denominator is small, but so is the numerator. Consequently, the denominator behaves a a regularization term, and the model class optimizing the stochastic complexity $-\log(P_{NML}(\mathbf{x}^n|\mathcal{M}))$ has to balance between model complexity and fit to the given data.

Let us now consider Naive Bayes models. The *Naive Bayes* is a Bayesian network with one root node and m leaf nodes attached to the root node. Variables related to nodes are multinomially distributed. The joint distribution corresponding to the Naive Bayes is defined by

$$P(\mathbf{x}) = P(x_0) \prod_{i=1}^{m} P(x_i|x_0), \tag{2}$$

where $\mathbf{x} = (x_0, x_1, \ldots, x_m)$ is a vector of variable value assignments and x_0 is the value in the root.

For Naive Bayes models, computing the numerator of (1) is trivial, but this is not the case with the denominator. In this paper we derive an efficient algorithm for computing the normalizing term for this model family and call it the *Naive Bayes normalizing term*.

3 Generating Functions and the Naive Bayes

The normalizing term for a single multinomial variable is called *multinomial normalizing term*. We can compute the multinomial normalizing term efficiently: the most efficient known method is proved using generating functions [9, 10]. We now use this same methodology in the Naive Bayes case. First we have to define the needed operations, which we use with generating functions, and then take a closer look at the Naive Bayes normalizing term.

3.1 Generating Functions

An *ordinary generating function* (OGF) of a sequence a_n is

$$F(z) = \sum_{n=0}^{\infty} a_n z^n = a_0 + a_1 z + a_2 z^2 + \cdots \, , \tag{3}$$

where $z \in \mathbb{C}$ [2]. We are only interested in coefficients a_n, not the value of the function $F(z)$ itself. The function $F(z)$ is only used for computation of some a_n coefficients or in derivation of recurrence formulas. With a *recurrence formula* we can compute the coefficient a_{n+1} with the help of the fixed and finite set of previous coefficients. A generating function may have a closed form, in which case manipulation is easier.

As a generating function is also a formal power series, all general formal power series operations are applicable. In the case of the multinomial normalizing term, however, we need the *exponential generating function* (EGF), which is of form

$$G(z) = \sum_{n=0}^{\infty} b_n \frac{z^n}{n!}. \tag{4}$$

We need to define for later use also two operations: a *coefficient extraction* from a formal power series and taking the power of the exponential generating function. The first operation is defined by

$$[z^n]G(z) = \frac{b_n}{n!}, \tag{5}$$

which means that $[z^n]$ gives us the coefficient of term z^n. The second operation defines what happens to coefficients when we exponentiate the generating function. Rising the generating function $G(z)$ to the power of two, denoted by

$$G^2(z) = \left(\sum_{n=0}^{\infty} b_n \frac{z^n}{n!} \right)^2 = \sum_{n=0}^{\infty} c_n \frac{z^n}{n!}, \tag{6}$$

corresponds to the binomial convolution

$$c_n = \sum_{h=0}^{n} \binom{n}{h} b_h b_{n-h} \tag{7}$$

in the level of coefficients. Similarly, the power of L gives

$$d_n = \sum_{h_1+\cdots+h_L=n} \left(\frac{n!}{h_1! h_2! \cdots h_L!} \right) b_{h_1} b_{h_2} \cdots b_{h_L}, \tag{8}$$

which is the multinomial convolution [2, 8]. This relation between the *expanded form* and the *power form* is the key feature for achieving a new, more efficient algorithm for computing the Naive Bayes normalizing term.

3.2 Naive Bayes Generating Function in the Power Form

First we have to define the generating function for the multinomial normalizing term. We do not give this function in the expanded form, but use a more compact notation: Lth power of a generating function [9, 10]. The power form is

$$\mathcal{B}^L(z) = \left(\sum_{n=0}^{\infty} n^n \frac{z^n}{n!} \right)^L . \tag{9}$$

We call the series inside the parentheses a *basic series*. The basic series here is of exponential type and formal power series coefficients are now $\frac{n^n}{n!}$. Coefficients of the exponential generating function are n^n. When we expand power L, we get an exponential generating function

$$\mathcal{B}^L(z) = \sum_{n=0}^{\infty} \mathcal{C}_{MN}(L,n) n^n \frac{z^n}{n!}, \tag{10}$$

where $\mathcal{C}_{MN}(L,n)$ is the multinomial normalizing term with L values and n data vectors [9, 10]. By a strict definition of generating functions this is not such a function, as it is not explicitly defined. However, we misuse the definition here slightly and in same way also later in the Naive Bayes case, because the implicit form is sufficient for our purposes. There are efficient ways to compute the term $\mathcal{C}_{MN}(L,n)$, and we will show one of them later.

Now we focus on the Naive Bayes normalizing term. The normalizing term is represented in the previous papers only using the expanded form. We denote the Naive Bayes normalizing term by $\mathcal{C}_{NB}(L, K_1, \ldots, K_m, n)$, where L is the number of values of the root variable and K_i is the number of values in leaf variable i. The following theorem shows the simple power form of the normalizing term.

Theorem 1.

$$\frac{n^n}{n!} \mathcal{C}_{NB}(L, K_1, \ldots, K_m, n) = [z^n] \left(\sum_{n=0}^{\infty} n^n \left(\prod_{i=1}^{m} \mathcal{C}_{MN}(K_i, n) \right) \frac{z^n}{n!} \right)^L .$$

Proof. A vector $(h_1, \ldots, h_L)$ is a sufficient statistics i.e. data counts of the root variable. The used formula for the Naive Bayes normalizing term is from the paper [11]. With standard manipulation we get

$$\begin{aligned}
&\frac{n^n}{n!}\mathcal{C}_{NB}(L, K_1, \ldots, K_m, n) && (11)\\
&\quad = \frac{n^n}{n!} \sum_{h_1+\cdots+h_L=n} \frac{n!}{h_1!\cdots h_L!} \left(\prod_{k=1}^{L} \left(\frac{h_k}{n}\right)^{h_k} \right) \prod_{i=1}^{m} \prod_{k=1}^{L} \mathcal{C}_{MN}(K_i, h_k) && (12)\\
&\quad = \frac{n^n}{n!} \sum_{h_1+\cdots+h_L=n} \frac{n!}{h_1!\cdots h_L!} \left(\frac{1}{n^n} \prod_{k=1}^{L} h_k^{h_k} \right) \prod_{k=1}^{L} \left(\prod_{i=1}^{m} \mathcal{C}_{MN}(K_i, h_k) \right) && (13)\\
&\quad = \frac{1}{n!} \sum_{h_1+\cdots+h_L=n} \frac{n!}{h_1!\cdots h_L!} \prod_{k=1}^{L} \left(h_k^{h_k} \prod_{i=1}^{m} \mathcal{C}_{MN}(K_i, h_k) \right) && (14)\\
&\quad = [z^n] \left(\sum_{n=0}^{\infty} n^n \left(\prod_{i=1}^{m} \mathcal{C}_{MN}(K_i, n) \right) \frac{z^n}{n!} \right)^L . && (15)
\end{aligned}$$

The last form is the power form, from where we can easily extract the basic series. We started from the expanded form and ended up with the power form. □

Let us compare the generating functions of the multinomial and the Naive Bayes normalizing terms. In the multinomial case we have

$$\left(\sum_{n=0}^{\infty} n^n \frac{z^n}{n!} \right)^L \tag{16}$$

and in the Naive Bayes case we have

$$\mathcal{E}^L = \left(\sum_{n=0}^{\infty} \mathcal{C}_{MN}(K_1, n) \cdots \mathcal{C}_{MN}(K_m, n) n^n \frac{z^n}{n!} \right)^L . \tag{17}$$

The two forms seem to be quite similar, except that in the Naive Bayes case we have additional multinomial normalizing terms inside the basic series terms. These extra multinomial normalizing terms makes the expanded form look quite ugly. However, despite of the complex terms, there exists an $\mathcal{O}(n^2 \log L)$ algorithm for computing the Naive Bayes normalizing term [11]. The basic idea is very simple: we can split the exponent L into two parts. Let's call these parts L^* and $L - L^*$. Then we get $\mathcal{E}^L = \mathcal{E}^{L^*}\mathcal{E}^{L-L^*}$. Now we can simply take the normal discrete convolution in the right hand side to get one term of the series in the left hand side. If we require that the result is also a normalizing term, we get the known recurrence formula

$$\begin{aligned}
\mathcal{C}_{NB}(L, K_1, \ldots, K_m, n) &= \sum_{k=0}^{n} \binom{n}{k} \left(\frac{k}{n}\right)^k \left(\frac{n-k}{n}\right)^{n-k} \\
&\quad \cdot \mathcal{C}_{NB}(L^*, K_1, \ldots, K_m, k)\ \mathcal{C}_{NB}(L - L^*, K_1, \ldots, K_m, n-k). \qquad (18)
\end{aligned}$$

So two lower exponents produce a higher one. To achieve the $\log L$ -term in the time complexity, we have to merge exponents wisely, so that we do not make any unnecessary steps. For example, if we want to compute $\mathcal{E}^L$ with $L = 16$, we first compute $\mathcal{E}^2$ and then compute $\mathcal{E}^2\mathcal{E}^2$ to get $\mathcal{E}^4$. In the same way we get the series $\mathcal{E}^8$ and finally the series $\mathcal{E}^{16}$. If the target value is not two to some power, then we have to do more complicated multiplications based on same idea. However, in the next section we present a novel, even more efficient way for computing the Naive Bayes normalizing term.

4 Powers of Formal Power Series

As basic series are formal power series, we can use some known powers of formal power series formula. One of these formulas is the *Miller formula* [6]. It is originally a result of Euler and it has time complexity of $\mathcal{O}(n^2)$ for any real number exponent, but of course only natural numbers are meaningful in our case.

The proof of the Miller formula has been sketched many times in history [7, 13, 6], but we were not able to find a detailed proof in the literature. The detailed proof is relatively straightforward and uses only standard manipulation. We complete below the missing parts of Knuth's proof for sake of clarity.

Theorem 2 (The Miller formula). *If two formal power series are* $V(z) = 1 + \sum_{k=1}^{\infty} v_k z^k$ *and* $W(z) = \sum_{k=0}^{\infty} w_k z^k$ *and* $W(z) = (V(z))^\alpha$, $\alpha \in \mathbb{R}$, *then* $w_0 = 1$ *and* $w_n = \sum_{k=1}^{n} \big((\frac{\alpha+1}{n})k - 1\big) v_k w_{n-k}$.

Proof. It is evident that $w_0 = 1$, since $v_0 = 1$ and $1 = 1^\alpha$. Next we derivate the basic equation of the theorem and get

$$W'(z) = \alpha V(z)^{\alpha-1} V'(z).$$

Then we multiply both sides with $V(z)$ and substitute $V(z)^\alpha = W(z)$, which gives us the equation

$$W'(z)V(z) = \alpha W(z)V'(z).$$

Let us now look at the z^{n-1}-coefficients of both sides:

$$[z^{n-1}]W'(z)V(z) = \alpha[z^{n-1}]W(z)V'(z) \tag{19}$$

$$\sum_{k=0}^{n} k w_k v_{n-k} = \alpha \sum_{k=0}^{n} (n-k) w_k v_{n-k} \tag{20}$$

$$\sum_{k=0}^{n-1} k w_k v_{n-k} + n w_n v_0 = \alpha \sum_{k=0}^{n-1} (n-k) w_k v_{n-k} + 0 \tag{21}$$

$$n w_n v_0 = \sum_{k=0}^{n-1} \big(\alpha(n-k) w_k v_{n-k} - k w_k v_{n-k}\big) \tag{22}$$

$$w_n = \frac{1}{n v_0} \sum_{k=0}^{n-1} \big((\alpha(n-k) - k) w_k v_{n-k}\big) \tag{23}$$

$$w_n = \sum_{k=0}^{n-1} \left(\frac{\alpha(n-k)-k}{n} \right) w_k v_{n-k} \tag{24}$$

$$w_n = \sum_{k=1}^{n} \left(\frac{\alpha(n-n+k)-n+k}{n} \right) w_{n-k} v_{n-n+k} \tag{25}$$

$$w_n = \sum_{k=1}^{n} \left(\left(\frac{\alpha+1}{n} \right) k - 1 \right) w_{n-k} v_k . \tag{26}$$

After some straightforward manipulation we get the result. □

We can obviously use this method for computing the normalizing term of the Naive Bayes model. It should be noted that while this result is elegant, if we use the discrete Fourier transform, we can achieve the time complexity $\mathcal{O}(n \log n)$ by using the basic identity $(V(z))^\alpha = \exp(\alpha \log(V(z)))$ and the *fast Fourier transform* (FFT). The FFT method involves utilization of Newton's method and is explained in the paper [1]. However, the usefulness of this approach is unclear as some earlier tests with the multinomial normalizing term [12] show that the used floating point numbers must have very high precision in practical cases. This is due to the fact that the values of the normalizing terms can be quite large, and consequently, as the data size increases, the precision of the floating point numbers must also increase. This means that increasing the precision will affect the efficiency of the algorithm, although the number of operations remains in principle the same.

5 Computation of the Naive Bayes Normalizing Term

The computation of the Naive Bayes normalizing term is quite straightforward given the results derived above. Now we collect these results in a form of an algorithm. As we did not describe earlier how to compute efficiently the multinomial normalizing term, we start by defining that.

5.1 Recurrence Formula for the Multinomial Normalizing Term

The multinomial normalizing term can be computed by using a recurrence formula [9, 10]. Initial values for this formula are

$$\mathcal{C}_{MN}(1, n) = 1 \quad \text{and} \tag{27}$$

$$\mathcal{C}_{MN}(2, n) = \sum_{k=0}^{n} \binom{n}{k} \left(\frac{k}{n} \right)^k \left(\frac{n-k}{n} \right)^{n-k} , \tag{28}$$

where $\mathcal{C}_{MN}(2, n)$ is a *binomial normalizing term*. After this we use the recurrence formula

$$\mathcal{C}_{MN}(L+2, n) = \mathcal{C}_{MN}(L+1, n) + \frac{n}{L} \mathcal{C}_{MN}(L, n) \tag{29}$$

to get the normalizing term with the wanted number of values in a multinomial variable. Time complexity of this whole method is $\mathcal{O}(n)$, as the number of values in a variable is usually much smaller than the number of the data points n. If we have to compute all normalizing terms between $[0, n]$ and we choose to use FFT, then binomial normalizing terms should be computed using (16). For the multiplication we can apply FFT.

5.2 The Algorithm

Now we have all the components we need for our algorithm. As said before, our main theorem is quite obvious given the earlier results (Theorems 1 and 2) and needs no proof.

Theorem 3. *The Naive Bayes normalizing term can be efficiently calculated in following way:*

1. *Compute first $n + 1$ binomial normalizing terms.*
2. *Use the recurrence formula to get the needed multinomial normalizing terms.*
3. *Compute the basic series $\sum_{k=0}^{n} \mathcal{C}_{MN}(K_1, k) \cdots \mathcal{C}_{MN}(K_m, k) k^k \frac{z^k}{k!}$.*
4. *Use the Miller formula to compute a new series, which is the basic series to the power of L.*
5. *Extract the Naive Bayes normalizing terms from the computed series by extracting coefficients and multiplying every coefficient so that the kth coefficient is multiplied by $\frac{k!}{k^k}$.*

Time complexity is $\mathcal{O}(n^2)$ for any exponent, because complexities of the steps are $\mathcal{O}(n^2)$, $\mathcal{O}(n \cdot \max(K_i))$, $\mathcal{O}(n \cdot m)$, $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$, respectively. This way we get all the Naive Bayes normalizing terms between $[0, n]$ in the given time, not just the nth of them. Notice that if the FFT approach could be used, time complexities of the first (explained in the Sect. 5.1) and the fourth steps would become $\mathcal{O}(n \log n)$. In this case the Miller formula in the fourth step is replaced with the algorithm mentioned in Section 4. Theorem 3 gives us actually a general framework for designing this kind of algorithms, as step 4 can be replaced with any exponentiation algorithm.

The method given in Theorem 3 is more efficient than the $\mathcal{O}(n^2 \log L)$-algorithm presented in [11]. This is easy to see, as the previous algorithm essentially performs in the fourth step at minimum $\log L$- power series multiplications instead of something which corresponds just one power series multiplication. In fact even when using the previous method, in some case it can be wise to compute series coefficients and not to require that all sub-results has to be normalizing terms. This way we can replace (18) just with normal convolution and achieve some more efficiency by omitting unnecessary multipliers present in the old formula. Furthermore, the old formula is applicable for values L greater than 2, but we can use normal convolution for values L greater than 1. In the fifth step we then convert wanted series coefficients into normalizing terms.

The new Miller method algorithm works perfectly fine with exact rational numbers. However our preliminary implementations show that in practice this is not necessarily the case with fixed precision floating point numbers and all formal power series: for some tested basic series, small errors in elementary operations tend to corrupt the normalizing terms very fast as n grows (because the algorithm uses iteratively previous values). Therefore with finite precision floating point numbers, using the previous, slower algorithm may be more advisable.

We have derived an efficient algorithm for computing the Naive Bayes normalizing term exactly. The computational complexity of computing the NML criterion for a Naive Bayes model is the same as for this algorithm, as the numerator of (1) is trivial to compute. Further information on computing the stochastic complexity for Naive Bayes models can be found in papers [11, 12].

6 Concluding Remarks

We presented an $\mathcal{O}(n^2)$ time algorithm for computing the normalizing term of the NML distribution exactly in the case of the Naive Bayes model. As the remaining term of the NML distribution is trivial to compute in this case, this result leads to a computationally efficient algorithm for computing the NML exactly for Naive Bayes models. We also defined a general framework for developing efficient algorithms for the NML computation in the Naive Bayes case and showed how the old $\mathcal{O}(n^2 \log L)$-algorithm can be seen as an special case of the framework, and how to make the algorithm more efficient.

We believe that it is not possible to do formal power series exponentiation in this case faster than $\mathcal{O}(n^2)$ without resorting to the Fast Fourier transform, which would easily lead to numerical problems, as discussed earlier. So unless the basic series for the Naive Bayes model reveals new hidden regularities with respect to exponentiation, our algorithm meets the lower limit of the time complexity for computing the NML exactly for Naive Bayes models.

Acknowledgements

The authors would like thank the anonymous reviewers for constructive 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] R. P. Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. In J. F. Traub, editor, *Analytic Computational Complexity*. Academic Press, New York, 1976.

[2] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. In preparation.

[3] P. Grünwald. *The Minimum Description Length Principle.* MIT Press, 2007.

[4] P. Grünwald, J. Myung, and M. Pitt, editors. *Advances in Minimum Description Length: Theory and Applications,.* MIT Press, 2005.

[5] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, September 1995.

[6] P. Henrici. Automatic computations with power series. *Journal of the ACM*, 3(1):11–15, January 1956.

[7] D.E. Knuth. *The Art of Computer Programming, vol. 2 / Seminumerical Algorithms (third edition).* Addison-Wesley, 1998.

[8] D.E. Knuth and B. Pittel. A recurrence related to trees. *Proceedings of the American Mathematical Society*, 105(2):335–349, 1989.

[9] P. Kontkanen and P. Myllymäki. Analyzing the stochastic complexity via tree polynomials. Technical Report 2005-4, Helsinki Institute for Information Technology (HIIT), 2005.

[10] P. Kontkanen and P. Myllymäki. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227–233, 2007.

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

[12] P. Kontkanen, H. Wettig, and P. Myllymäki. NML computation algorithms for tree-structured multinomial Bayesian networks. *EURASIP Journal on Bioinformatics and Systems Biology*, to appear.

[13] G. Nakos. Expansions of powers of multivariate formal power series. *Mathematica Journal*, 3(1):45–47, Winter 1993.

[14] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[15] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Company, New Jersey, 1989.

[16] J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January 1996.

[17] J. Rissanen. *Information and Complexity in Statistical Modeling.* Springer, 2007.

[18] Yu M. Shtarkov. Universal sequential coding of single messages. *Problems of Information Transmission*, 23:3–17, 1987.