Computing the Multinomial Stochastic Complexity in Sub-Linear Time

Tommi Mononen and Petri Myllymäki
Helsinki Institute for Information Technology (HIIT), Finland
{firstname}.{lastname}@hiit.fi

Abstract

Stochastic complexity is an objective, information-theoretic criterion for model selection. In this paper we study the stochastic complexity of multinomial variables, which forms an important building block for learning probabilistic graphical models in the discrete data setting. The fastest existing algorithms for computing the multinomial stochastic complexity have the time complexity of $\mathcal{O}(n)$, where n is the number of data points, but in this paper we derive sub-linear time algorithms for this task using a finite precision approach. The main idea here is that in practice we do not need exact numbers, but finite floating-point precision is sufficient for typical statistical applications of stochastic complexity. We prove that if we use only finite precision (e.g. double precision) and precomputed sufficient statistics, we can in fact do the computations in sub-linear time with respect to data size and have the overall time complexity of $\mathcal{O}(\sqrt{dn} + L)$, where d is precision in digits and L is the number of values of the multinomial variable. We present two fast algorithms based on our results and discuss how these results can be exploited in the task of learning the structure of a probabilistic graphical model.

1 Introduction

Stochastic complexity (SC) is an information-theoretic model selection criterion, which can be seen as a theoretical instantiation of the minimum description length (MDL) principle (Rissanen, 2007; Grünwald, 2007). Intuitively speaking, the basic idea is that the best model for the data is the one which results in the shortest description for the data together with the model. This principle gives us a non-informative, objective criterion for model selection, but there are many ways to define the stochastic complexity formally; one theoretically solid way is to use the *normalized maximum likelihood* (NML) distribution. Recent results suggest that this criterion performs very well in the task of learning Bayesian network structures (Roos et al., 2008).

In the following, let $\mathcal{M}$ denote a parametric probabilistic model, and $\hat{\theta}(\mathbf{x}^n)$ the maximum likelihood parameters of the model given a matrix of observations $\mathbf{x}^n$. The NML distribution is defined as

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

where in the numerator we have the maximum likelihood of our observed data and in the denominator we have the sum of maximum likelihoods (denoted in the sequel by $\mathcal{C}(\mathcal{M}, n)$) over all the discrete data sets of size n (Shtarkov, 1987).

Let us define the stochastic complexity as the negative logarithm of (1):

$$SC(\mathbf{x}^n \mid \mathcal{M}) = -\log \frac{P(\mathbf{x}^n \mid \hat{\theta}(\mathbf{x}^n), \mathcal{M})}{\mathcal{C}(\mathcal{M}, n)}. \quad (2)$$

The basic model selection task is to compute the value of this model selection criterion for parametric models of different complexity and choose the one for which this value is minimized, given the observed data.

The single multinomial variable model is an important building block for building more complex probabilistic graphical models for discrete data. For this reason we want to able to compute the NML for multinomial variables as efficiently as possible. In the following, we simplify our notation and leave out $\mathcal{M}$: the model is implicitly defined by the number of values of the multinomial variable, denoted

by L. The numerator is now

$$P(x^n \mid \hat{\theta}(x^n), L) = \prod_{k=1}^{L} \left(\frac{h_k}{n} \right)^{h_k}, \qquad (3)$$

where h_k is a number of data points assigned to the kth value. We expect in this paper that sufficient statistics is known and computing (3) takes therefore only time $\mathcal{O}(L)$. The denominator is

$$\mathcal{C}(L,n) = \sum_{h_1+\cdots+h_L=n} \frac{n!}{h_1! \cdots h_L!} \prod_{k=1}^{L} \left(\frac{h_k}{n} \right)^{h_k},$$

which is a sum of maximum likelihoods so that the summation goes over every possible data of length n.

Although using the definition directly for computing the *multinomial normalizing sum* in the denominator is not computationally feasible, several algorithms for doing this in $\mathcal{O}(n)$ time have been recently developed (Kontkanen and Myllymäki, 2007; Mononen and Myllymäki, 2008b). In this paper we show that our earlier theoretical results presented in (Mononen and Myllymäki, 2008b) can be used for constructing algorithms that compute $\mathcal{C}(L,n)$ in sub-linear time with any desired (finite) precision. We start by briefly reviewing the relevant earlier results and then show how they can be exploited in deriving new ultra-fast algorithms.

2 Known Properties of the Normalizing Sums

In Mononen and Myllymäki (2008b) we proved that the multinomial normalizing sum can be described as a confluent hypergeometric function evaluated at a certain point. We showed that we can write the hypergeometric presentation also in another simple form using falling and rising factorial polynomials. The *falling factorial polynomials* are of the form

$$x^{\underline{k}} = x(x-1)\cdots(x-k+1), \qquad (4)$$

and the *rising factorial polynomials* are

$$x^{\overline{k}} = x(x+1)\cdots(x+k-1). \qquad (5)$$

The *binomial normalizing sum* is then

$$\mathcal{C}(2,n) = \sum_{k=0}^{n} b_k = \sum_{k=0}^{n} \frac{n^{\underline{k}}}{n^k}, \qquad (6)$$

and the general multinomial normalizing sum can be written as

$$\mathcal{C}(L,n) = \sum_{k=0}^{n} m_k = \sum_{k=0}^{n} \frac{n^{\underline{k}} \, (L-1)^{\overline{k}}}{n^k k!}. \qquad (7)$$

As these forms spin off from hypergeometric forms, and a hypergeometric series has the property that there exist a simple ratio of consecutive terms, we know that also (6) and (7) have this property. We will introduce these ratios later in sections 3.1 and 4.1 and use them in the computations.

It is known that the binomial normalizing sum equals to the expectation of the *birthday problem* with the mapping: data size is equal to the number of days (Mononen and Myllymäki, 2008b). We will use an approximation derived for this expectation later in our proof.

There is a recurrence formula for computing the multinomial normalizing sum as the value of the corresponding binomial normalizing sum is known (Kontkanen and Myllymäki, 2007):

$$\mathcal{C}(L,n) = \mathcal{C}(L-1,n) + \frac{n \cdot \mathcal{C}(L-2,n)}{L-2}, \qquad (8)$$

and $\mathcal{C}(1,n)$ is defined to be 1 for every n. This formula can be effectively used for linear time computation of multinomial normalizing sums.

3 The Binomial Normalizing Sum

3.1 Properties of the Sum Terms

Let us start by plotting the terms of (6). We can immediately observe that the first terms of the sum give the greatest impact and most of the terms are very small (see Figure 1). All the terms are positive and getting closer to the zero, because the ratio of successive terms of (6) is

$$\frac{b_k}{b_{k-1}} = \frac{n-k+1}{n}. \qquad (9)$$

Now the natural question is, how many terms do we need, if we want to compute the normalizing sum and use for example double precision. We study this question more closely in section 3.2, but in loose terms the number of needed terms is proportional to the square root on n, which promises sub-linear time performance.

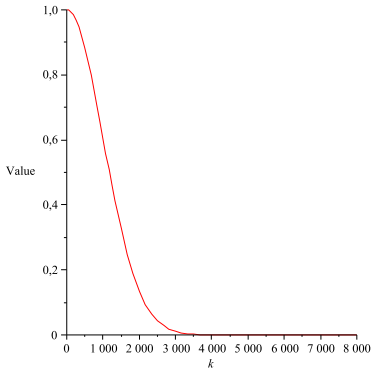

Figure 1: Magnitude of the first 8000 terms of the binomial normalizing sum when the data size (n) is one million.

However, first we have to quantify how to measure the precision. We measure in the standard way the error we make when pruning the sum: we compute the tail sum and compare it to the whole sum. Thus we compute a relative error. However, because setting the desired relative error directly is quite cumbersome, we rather compute it in digits: it much easier just to say that we need e.g. 7 digit precision. More precisely, we define that for positive real numbers p and q, where $p \leq q$ and $p, q \geq 1$, the precision in digits is

$$\left\lfloor -\log_{10}\left(\frac{q-p}{q}\right)\right\rfloor. \tag{10}$$

So if the target q is for example 1.100000 and the approximation p is 1.099999, the precision is $\lfloor 6.04139 \rfloor$. So although only the first digit is the same, the precision in digits according to the definition above is 6. But if we round p after the sixth digit, we get 1.10000. This means that although the precision in digits does not tell all the time how many correct digits we have, it is still very close to what we want. Next we do an upper bound approximation of the last required term of the binomial normalizing sum for achieving some fixed precision.

3.2 Proof of the Right Bound

There is no known closed form solution for (6). To compute the precision, we have to compute the sum starting from some b_r to all the way to the the term b_n. As said before, we are interested in this tail sum, because it tells us how big an error we make, if we stop computing the sum after the term b_{r-1}. Although the terms of the sum look simple, they are a bit tricky to handle, and therefore we perform several upper bound approximations for the terms. Upper bound approximations are of course required, because we want to be sure that whatever bound we get, it must give the promised result. We also move from the discrete sum to an integral presentation, because it makes things simpler in this case.

Next we give in a row four propositions needed for proving the index bound of the binomial sum. After the propositions we prove the mentioned bound, which we call the *right index bound* (in the multinomial case also the left bound exists).

Proposition 1. *We have the following upper bound approximation for the term b_k:*

$$\frac{n^{\underline{k}}}{n^k} \leq \left(1 - \frac{k-1}{2n}\right)^{k-1}, \textit{where} \quad 1 \leq k \leq n.$$

Proof. We prove that the ratio of both sides is bigger than one. Let us look at the ratio:

$$\frac{\left(1-\frac{k-1}{2n}\right)^{k-1}}{\frac{n^{\underline{k}}}{n^k}} = \frac{n \cdot n^{k-1}\left(\frac{n-\frac{1}{2}(k-1)}{n}\right)^{k-1}}{n^{\underline{k}}} \tag{11}$$

$$= \frac{(n-\frac{1}{2}(k-1))^{k-1}}{(n-1)\cdots(n-k+1)} \tag{12}$$

$$= \prod_{r=2}^{k} \frac{n-\frac{1}{2}(k-1)}{n-r+1} = \prod_{r=2}^{k} a_r. \tag{13}$$

Now we just look at the product of pairwise terms defined by

$$a_r a_{k-r+2} = \frac{(n-\frac{1}{2}(k-1))^2}{(n-r+1)(n-k+r-1)} = \frac{N_r}{D_r}.$$

We want to prove that each of these pairwise terms is bigger than 1. However, we can equivalently subtract the denominator from the numerator and require the result to be bigger than 0, because both are always positive in our range. The result is

$$N_r - D_r = n + \frac{1}{4}k^2 - rk + \frac{1}{2}k + r^2 - 2r + \frac{5}{4}. \tag{14}$$

We take the derivate of this difference with respect to k and get the minimum point $k = 2r - 1$. Substituting this in (14), we notice that the result is

$n-r+1$, which is always bigger than 0 in our range. This means that also the pairwise terms must be always bigger than 1, which implies that the proposition must be true for even number of terms a_r.

If we have an odd number of terms a_r, then we have to still prove that the median term is also bigger than 1. The median term is

$$a_{\frac{k}{2}+1} = \frac{2n-k+1}{2n-k}. \tag{15}$$

This cannot be smaller than 1, because $n \geq k \geq 0$. □

Proposition 2. *The following inequality is true for* $\frac{k}{2n} < 1$*:*

$$\left(1 - \frac{k}{2n}\right)^k \leq e^{-\frac{k^2}{2n}}.$$

Proof. Take the natural logarithm of both sides of the inequality and use the known logarithm inequality $\ln(1+x) \leq x$, which is valid for all $x > -1$. □

Proposition 3. *Because* b_k *is a continuous monotonically decreasing function inside the interval* $[0, n]$*, the discrete volume (the sum) corresponds to the upper Riemann sum with intervals of length one. Hence we can give the following inequality:*

$$\sum_{k=0}^{n} b_k \leq 1 + \int_{k=1}^{n} b_{k-1} \mathrm{d}k.$$

Proof. Our integral is always bigger than the given upper sum, because on the right hand side we shifted our function in such way that every discrete column is always entirely below the curve. The size of the first column is 1, which is the first term on the right hand side. □

Proposition 4. *The following inequality holds:*

$$\operatorname{erf}(x) \geq \sqrt{1 - e^{-x^2}},$$

when $x \geq 0$ *and*

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{t=0}^{x} e^{-t^2} \mathrm{d}t.$$

Proof. First suppose $x \geq 0$. Let us now modify the inequality:

$$\operatorname{erf}(x) \geq \sqrt{1 - e^{-x^2}} \tag{16}$$

$$(\operatorname{erf}(x))^2 + e^{-x^2} - 1 \geq 0 \tag{17}$$

Denote the left hand side by $h(x)$ and take the derivative of it:

$$h'(x) = \frac{2e^{-x^2}(2\operatorname{erf}(x) - x\sqrt{\pi})}{\sqrt{\pi}}. \tag{18}$$

We can see that all the roots in our range are solutions for the equation

$$2\operatorname{erf}(x) - x\sqrt{\pi} = 0, \tag{19}$$

$$\operatorname{erf}(x) = \frac{\sqrt{\pi}}{2} x. \tag{20}$$

As the error function is a convex function between 0 and infinity and the right hand side of (20) is a linear function, there can be only two solutions. The first one is $x = 0$, where $h(0) = 0$, and the other one is $x \approx 0.8982$. The second solution is a positive maximum point. As we have

$$\lim_{x \to \infty} h(x) = (1)^2 + 0 - 1 = 0, \tag{21}$$

this means that $h(x)$ must be always positive in the range, which completes our proof. □

Finally we are now ready to introduce our main theorem giving the right bound approximation:

Theorem 1. *Given precision in digits* (d) *and data size* n*, the right index bound* t *for the binomial normalizing sums is* $\left\lceil 2 + \sqrt{-2n \ln(2 \cdot 10^{-d} - 100^{-d})} \right\rceil$.

Proof. First we approximate the upper bound of the partial binomial normalizing sum from 0 to r:

$$\sum_{k=0}^{r} \frac{n^{\underline{k}}}{n^k} \leq 1 + \sum_{k=1}^{r} \left(1 - \frac{k-1}{2n}\right)^{k-1} \tag{22}$$

$$\leq 1 + \sum_{k=1}^{r} e^{-\frac{(k-1)^2}{2n}} \tag{23}$$

$$\leq 2 + \int_{k=2}^{r} e^{-\frac{(k-2)^2}{2n}} \tag{24}$$

$$= 2 + \sqrt{\frac{n\pi}{2}} \operatorname{erf}\left(\frac{r-2}{\sqrt{2n}}\right) = F(r) \tag{25}$$

Previous inequality steps follow easily from propositions 1, 2 and 3. Now we can express the precision in digits (d) with the equation

$$-\log_{10}\left(\frac{F(n) - F(r)}{\sqrt{\frac{n\pi}{2}}}\right) = d, \tag{26}$$

where the denominator inside the logarithm is a lower bound approximation for the binomial normalizing sum. For example Laplace's method gives for (6) the approximation (Flajolet and Sedgewick, 2005):

$$\mathcal{C}(2,n) = \sqrt{\frac{n\pi}{2}} + \frac{2}{3} + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right). \quad (27)$$

By omitting the constant term, we get our lower bound approximation for the denominator, which is valid for all data sizes (exact proof omitted). Thus we have

$$-\log_{10}\left(\frac{\sqrt{\frac{n\pi}{2}} - \sqrt{\frac{n\pi}{2}}\ \mathrm{erf}\left(\frac{r-2}{\sqrt{2n}}\right)}{\sqrt{\frac{n\pi}{2}}}\right) = d \quad (28)$$

$$-\log_{10}\left(1 - \ \mathrm{erf}\left(\frac{r-2}{\sqrt{2n}}\right)\right) = d. \quad (29)$$

We replaced the first error function in (26) with its supremum value 1 and got (28). If we solve r, we have

$$r = 2 + \sqrt{2n} \cdot R, \quad (30)$$

where

$$R = \mathrm{erf}^{-1}(1 - 10^{-d}). \quad (31)$$

The final task is now to approximate the inverse of the error function (Winitzky, 2008). We need this for approximating R to get a nice, clean and computable bound. First we compute the Taylor approximation:

$$g(x) = \ln(1 - \mathrm{erf}(x)^2) \approx -\frac{4}{\pi}x^2 + \mathcal{O}(x^4). \quad (32)$$

We need only the first non-zero term for our purpose. The approximation is then

$$\mathrm{erf}(x) = \sqrt{1 - e^{g(x)}} \approx \sqrt{1 - e^{-\frac{4x^2}{\pi}}}, \quad (33)$$

and it is good enough as our tests later show. This is not a lower bound approximation, but we need one, because we invert the approximating function. A little dirty trick solves our problem. We just change the multiplier 4 to π (Proposition 4). This new function can be easily inverted and the result therefore is

$$\mathrm{erf}^{-1}(u) = \sqrt{-\ln(1 - u^2)}. \quad (34)$$

The final step is to set $u = 1 - 10^{-d}$ and we have the result

$$R \approx \sqrt{-\ln(1 - (1 - 10^{-d})^2)} \quad (35)$$

$$= \sqrt{-\ln(2 \cdot 10^{-d} - 100^{-d})}. \quad (36)$$

□

We continue approximating R, because for the time complexity reasons, we need a simplified form to see the magnitude of this term. We easily see that

$$\sqrt{-\ln(2 \cdot 10^{-d} - 100^{-d})} \leq \sqrt{-\ln(10^{-d})} \quad (37)$$

$$= \sqrt{d\ln(10)}, \quad (38)$$

and therefore the required number of terms is $\mathcal{O}(\sqrt{dn})$.

The index bound seems to be quite good. In Figure 2 we have plotted optimal indexes and indexes given our bound with respect to data size n. We chose precisions so that they correspond approximately to single and double precision floating-point numbers. If n is one million, the index error is about $+250$ in both cases. The single precision error is a bit larger than the double precision error, because the index bound is getting tighter as precision increases.

4 The Multinomial Normalizing Sum

4.1 Properties of the Sum Terms

As we already saw, the ratio of the terms of the binomial normalizing sum is a simple rational function. In the multinomial case the ratio is the function

$$\frac{m_k}{m_{k-1}} = \frac{(n-k+1)(k+L-2)}{nk}. \quad (39)$$

Let us look at the terms of the multinomial normalizing sum. Figure 3 suggest that there is the biggest term and if we look at the term function, we see that it is unimodal. The next theorem and its proof give formal justifications for these claims.

Theorem 2. *The index of the biggest term of the multinomial normalizing sum is* $\left\lfloor \frac{1}{2}\left(3 - L + \sqrt{L^2 + (4n-2)L - 8n + 1}\right)\right\rfloor$.

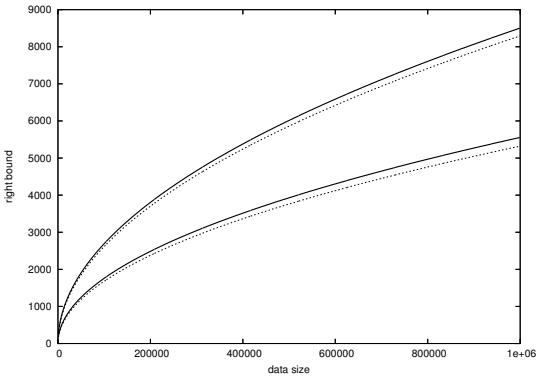

Figure 2: Terms needed for 16 (above) and 7 digit precisions with given data size. Actual approximations are shown as thick solid line. Thin dotted lines represent optimal index values.

Proof. We have to solve the equation

$$\frac{m_k}{m_{k-1}} = 1 \tag{40}$$

with respect to k. In other words, we are interested in values of real-valued k, where two consecutive sum terms have the same value. This requires solving roots of second order polynomial with respect to k. The other root is always negative and therefore is not in our range; let us denote the positive root by r. We are allowed to take the floor, because we know that the peak is between continuous index values $r-1$ and r. Outside this range all the sum term values are smaller than inside the range. Inside this range there are one or two integer points. If only one, then it is $\lfloor r \rfloor$. If there are two integer indexes, then $r-1$ and r must be these and they both give the same maximum value and thus $\lfloor r \rfloor$ gives the other of the two maximum values. □

This proved at the same time that the sum terms are getting bigger until they reach the peak, after which they start getting smaller. This unimodality gives us a great opportunity to construct simple and efficient algorithms.

4.2 About the Index Bounds

We can guess from the 15-nomial in Figure 3 that if we want to compute the sum in a fixed precision, we actually have to compute the sum terms from some $s \geq 0$ to some $t \leq n$. This means that the index of the first required term can be bigger than 0.

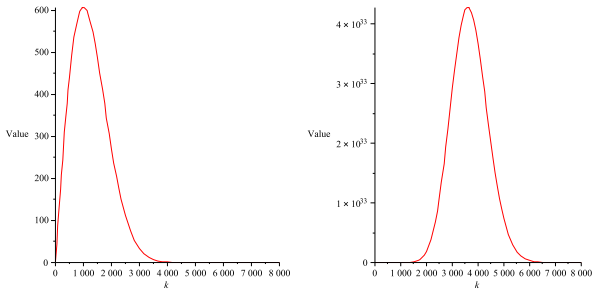

Figure 3: Magnitude of the first 8000 terms of the trinomial (left) and the 15-nomial (right) normalizing sums when data size (n) is one million.

However, we have to compute the factorials starting from 0 (to avoid any approximation), so we still have to start from index 0. Our empirical tests also show that we have to compute a lot more terms than in the binomial case. The effect is visible also by comparing figures 1 and 3. However, we can use (8) for computing the multinomial normalizing sum when the value of the binomial normalizing sum is known. This recurrence formula method seems to be much more efficient and therefore it is unnecessary to prove bounds in the multinomial case.

5 Sub-Linear Algorithms

First we present a simple algorithm (Algorithm 1), which is validated directly by unimodality. The basic idea is that the algorithm can sum the terms of a multinomial sum until the sum does not change anymore. Terms of the left slope always change the sum because each term is bigger than the previous one. After the peak, terms are monotonically getting smaller, and we know that the terms are never getting bigger anymore. Therefore the algorithm can stop after the sum has converged. All terms in the tail are so small and decaying so rapidly, that they cannot have a significant effect on the finite sum (we will return to this subject after the second algorithm). A precision of the result is determined by the precision of the used floating-point numbers.

Notice that the algorithm is using the recurrence

$$m_k = \frac{(n-k+1)(k+L-2)}{nk} \cdot m_{k-1} \tag{41}$$

while computing the sum terms. This way we can avoid huge factorial values and also floating-point

```
Compute_Multinomial(L,n){
  double sum = 1, previous_sum = -1, m = 1;
  int k = 1;

  for k from 1 to n by 1{
    m = (k+L-2) * (n-k+1) / (n*k) * m;
    sum = sum + m;

    if(sum is same as previous_sum){
      return sum;
    }

    previous_sum = sum;
  }

  return sum;
}
```

Algorithm 1: A simple algorithm for computing the multinomial normalizing sum.

```
Compute_Multinomial_With_Recurrence(L,n){
  double sum = 1, b = 1, old_sum, new_sum;
  int k,j;
  int bound = ceil( 2 + sqrt( -2*n*ln( 2*10^(-d)
                    -100^(-d) ) ) );

  for k from 1 to bound by 1{
    b = (n-k+1) / n * b;
    sum = sum + b;
  }

  old_sum = 1;

  for j from 3 to L by 1{
    new_sum = sum + (n * old_sum) / (j-2);
    old_sum = sum;
    sum = new_sum;
  }

  return sum;
}
```

Algorithm 2: A faster algorithm for computing the multinomial normalizing sum.

errors seem to be much lower. However the time complexity is not proved for this algorithm, as we did not compute the index bounds in the multinomial case. In addition, we cannot set the digit precision freely. For these reasons we present a second algorithm, which also seems to be twice as fast as the first one.

The intuitive idea of Algorithm 2 is to first compute the index bound, and then compute the binomial normalizing sum using the ratio of successive terms. After this, the algorithm uses the recurrence formula to compute the wanted multinomial normalizing sum. Variable `bound` is not directly computable as presented in the pseudocode, but we can use some standard logarithmic manipulation to avoid underflow. The time complexity of this second algorithm is $\mathcal{O}(\sqrt{dn} + L)$.

Now we can revise the question about the tail sum. Terms below the index bound do not affect the sum, but the first algorithm actually stops in the binomial case before the right bound is reached, because single terms do not change the sum. But if we take a sum starting from the stopping point all the way to the index bound, we can notice that this tail sum can affect the original sum. In our empirical tests we found out that the effect seems to be only on the few last digits, which is of the same magnitude as the floating point errors of our algorithms.

In Figure 4 we can see the average decay of the last digits with respect to data size. The variance is small between different numbers of same magnitude, so the figure gives a realistic impression. In the end we decided to keep the algorithms simple, because such small errors in the criterion hardly have any effect on the actual model selection task.

There are two operations that in fact increase precision. First we found out that the recurrence formula in Algorithm 2 tends to increase precision of those terms with an odd number of outcomes. Second we still have to take a logarithm of the normalizing sum, when computing actual stochastic complexity. The effect of the latter operation seem to be about one digit.

Precise values can be achieved using a simple trick: use a floating-point precision that is higher than the precision d, and crop the tail digits (e.g. quad precision for triple precision).

6 Conclusions

Stochastic complexity is an elegant, information-theoretic criterion for learning probabilistic graphical model structures. Although probabilistic in nature, it is fully objective and does not involve any hyperparameters which may be introduce problems in learning (Silander et al., 2007).

The fastest previously known algorithms for computing the multinomial stochastic complexity are $\mathcal{O}(n)$-algorithms. In this paper we showed that our previous hypergeometric representation of multinomial normalizing sums can be used for deriving sub-linear algorithms.

As the multinomial variable is an important building block in many more complex model classes, the results are directly applicable to model selec-

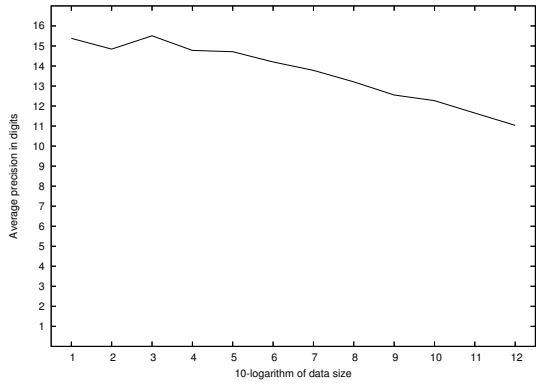

Figure 4: Average precision in the binomial normalizing sum with respect to data size using double precision floating-point numbers and Algorithm 2.

tion tasks in these cases as well. However, it should be noted that if the learning criterion is defined via the standard normalized maximum likelihood, the savings in the overall computational complexity are not necessarily very significant: for example, in the Naive Bayes case (classification or clustering tasks if the root node is hidden), the learning criterion involves a product of multinomial normalizing sums corresponding to the predictor variables, and these can be now computed in sub-linear time using the results above. Nevertheless, the real bottleneck of the computation process is a convolution operation, which still takes at least $\mathcal{O}(n^2 \log L)$ floating-point operations to compute (Mononen and Myllymäki, 2007). Similarly, when learning tree-structured Bayesian networks (Mononen and Myllymäki, 2008a), we can speed-up some parts of the computation, but not all.

On the other hand, there now exists a slightly different way to define the stochastic complexity, based on the factorized NML (fNML) criterion (Roos et al., 2008). The fNML criterion can be used in the Naive Bayes or in the Bayesian tree case, or even for learning Bayesian networks (DAGs) in general. In this case, the learning criterion factorizes into a product of multinomials, which means that the speed-up offered by our sub-linear algorithm is more apparent.

Acknowledgments

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.

References

P. Flajolet and R. Sedgewick. 2005. *Analytic Combinatorics*. Unpublished.

P. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

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

T. Mononen and P. Myllymäki. 2007. Fast NML computation for Naive Bayes models. In V. Corruble, M. Takeda, and E. Suzuki, editors, *Proceedings of the 10th International Conference on Discovery Science*, October.

T. Mononen and P. Myllymäki. 2008a. Computing the NML for Bayesian forests via matrices and generating polynomials. In *Proceedings of the IEEE Information Theory Workshop*, Porto, Portugal, May.

T. Mononen and P. Myllymäki. 2008b. On the multinomial stochastic complexity and its connection to the birthday problem. In *Proceedings of the International Conference on Information Theory and Statistical Learning*, Las Vegas, NV, July.

J. Rissanen. 2007. *Information and Complexity in Statistical Modeling*. Springer.

T. Roos, T. Silander, P. Kontkanen, and Myllymäki P. 2008. Bayesian network structure learning using factorized NML universal models. In *Proceedings of the Information Theory and Applications Workshop*, San Diego, CA, January.

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

T. Silander, P. Kontkanen, and P. Myllymäki. 2007. On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In R. Parr and L. van der Gaag, editors, *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 360–367. AUAI Press.

S. Winitzky. 2008. A handy approximation for the error function and its inverse. Unpublished.