On asymptotically optimal methods of prediction and adaptive coding for Markov sources

Boris Ya. Ryabko
Information Technology University, Copenhagen, Denmark

Flemming Topsøe
Department of Mathematics, University of Copenhagen, Denmark*

Summary. The problem of predicting a sequence $x_1, x_2, \cdots$ generated by a discrete source with unknown statistics is considered. Each letter x_{t+1} is predicted using information on the word $x_1 x_2 \cdots x_t$ only. In fact, this problem is a classical problem which has received much attention. Its history can be traced back to Laplace. To estimate the efficiency of a method of prediction, three quantities are considered: the *precision* as given by the Kullback-Leibler divergence, the *memory size* of the program needed to implement the method on a computer and the *time* required, measured by the number of binary operations needed at each time instant.

A method is presented for which the memory size and the average time is close to the minimum. The results can readily be translated to results about adaptive coding.

Keywords. Prediction, adaptive coding, universal coding, ε-capacity, ε-net, imaginary sliding window.

1 Introduction

The problem of prediction and the closely related problem of adaptive coding of time series is well known in information theory, probability theory and statistics. The problem can be traced back to Laplace (cf. Feller [8] where the problem is referred to as the *problem of succession*).

Presently, the problem of prediction is investigated by many researches because of its practical applications and importance for probability theory, statistics, pattern recognition, cybernetics and other theoretical sciences. An extensive review and list of references can be found in Algoet, [2].

[0]Research of both authors supported in part by the Carlsberg Foundation.

We shall investigate the relation between complexity and precision of prediction methods. This problem is interesting from a practical point of view. To realize this, imagine you want to predict x_{t+1} knowing the sequence $x_1 x_2 \cdots x_t$. Then you have to deal with a growing amount of data as t increases. This results in an increase of the time of calculation and of the amount of memory space required. Though practical methods have been devised and studyed empirically, there is a lack of studies where the relation between complexity and precision of prediction methods is investigated theoretically. Such theoretical results will be useful, e.g. when judging the efficiency of concrete prediction methods in experiments based on real sources.

Here, the apparently simplest classes of problems are considered as a first step towards an understanding of the connection between complexity and precision. Namely, we consider a source with unknown statistics which generates sequences $x_1 x_2 \cdots$ of letters from a finite alphabet $A = \{a_1, \cdots, a_n\}$ and the models we consider are either Bernoulli or Markov models (of fixed connectivity). The underlying true distribution, which is unknown except for the restriction given by the model assumption, is indicated by the letter p. We imagine that we have a computer at our disposal for solving the prediction problem. Now, let us have a specific method of prediction in mind. As input we consider any finite string $x_1 x_2 \cdots x_t$ of letters from A and as output we require that at each time instant t we receive non-negative numbers $p^*(a_1|x_1 \cdots x_t), \cdots, p^*(a_n|x_1 \cdots x_t)$ which are estimates of the unknown conditional probabilities $p(a_1|x_1 \cdots x_t), \cdots, p(a_n|x_1 \cdots x_t)$, i.e. of the probabilities $p(x_{t+1} = a_i|x_1 \cdots x_t)$; $i = 1, \cdots, n$. The set $p^*(a_i|x_1 \cdots x_t)$; $i \leq n$ is the *prediction*.

The *precision* of a prediction method is measured by the divergence between p and p^*, and the *complexity* of a method is characterized by two numbers: The *time* of calculation at each time instant

in bit operations and the *memory size* in bits of the computer which is necessary in order to execute the program defining the method. This approach is natural from a practical point of view and well known in Computer Science, see, for example, [1]. It is in conformity with methods used for prediction and for similar problems of learning and statistics which are based on the theory of finite state machines (cf. [7] and [3]).

The problem which Laplace considered (cf. also the recent contribution by Krichevskii, [13]) was to estimate the probability that the sun will rise tomorrow, given that it has risen every day since the creation. Using our terminology, we can say that Laplace estimated $p(r|rr\cdots r)$ and $p(\bar{r}|rr\cdots r)$, where $\{r,\bar{r}\}$ is the alphabet ("sun rises", "sun does not rise") and the length of $rr\cdots r$ is the number of days since the creation.

Instead of viewing the prediction $p^*(a|x_1\cdots x_t)$ as probabilities, one may view $p^*(a|x_1\cdots x_t)$ as a stake on the letter $a\in A$. This more game theoretical view also goes back to Laplace. Further considerations of a game theoretical nature were suggested by many authors (Kelly [10], Topsøe [20], Ryabko [16], Feder,Merhav and Gutman [7] and Rissanen [15]).

The problem of adaptive (and universal) coding is closely related to the prediction problem and was investigated in Krichevskii [12] and Ryabko [17]. From a mathematical point of view the problems are identical and can, therefore, be investigated together (cf. also further explanatory remarks in the next section).

We shall suggest two prediction methods for Markov sources, which are, respectively, asymptotically optimal (or near optimal) in average and asymptotically optimal with probability one. The one method is deterministic, whereas the other uses randomization. [1]

[1]Regarding randomization (cf. also [14]), it is important to realize that this device is considered as an external device, an *oracle*, which can be consulted from time to time. Therefore, the use of randomization does not in itself pose extra demands on the memory of the main computer. In order to implement the randomizing mechanism, a pseudorandom number generator will always be used and this demands separate devices (memory etc.). As stated, these demands are not considered to interfere with memory management etc. related to the main computer. Certain theoretical considerations demonstrate the possibility to generate efficient randomizing agents with low complexity, and this justifies the view taken. We acknowledge that in practice the *exterior* device and the methods in question are built together using the same computer.

2 Definitions

Consider an alphabet $A=\{a_1,\cdots,a_n\}$ with $n\geq 2$ letters and denote by A^t the set of words $x_1\cdots x_t$ of length t from A. Let p be a source which generates letters from A. Formally, p is a probability distribution on the set of words of infinite length or, more simply, $p=(p^t)_{t\geq 1}$ is a consistent set of probabilities over the sets $A^t;\ t\geq 1$. By $M_0(A)$ we denote the set of Bernoulli sources over A, and by $M_k(A)$ the set of Markov sources over A of connectivity (memory) $k;\ k\geq 1$.

We use M to denote the model under consideration. Formally, M could be any set of sources but for this paper we only consider the cases $M=M_0(A)$ and $M=M_k(A)$ with k a fixed natural number. In fact, we shall mainly focus on the case $M=M_0(A)$ as results for the general Markovian case can be deduced from results for the Bernoulli case.

Denote by $D(\cdot\|\cdot)$ the Kullback-Leibler divergence and consider the source p and a method γ of prediction. For a deterministic method of prediction, the *precision* is characterized by the divergence

$$r_{\gamma,p}(x_1\cdots x_t)=D\left(p(\cdot|x_1\cdots x_t)\|p^*_\gamma(\cdot|x_1\cdots x_t)\right)$$

$$=\sum_{a\in A}p(a|x_1\cdots x_t)\log\frac{p(a|x_1\cdots x_t)}{p^*_\gamma(a|x_1\cdots x_t)}.$$

Here, and in the sequel, log denotes natural logarithms (we also need logarithms to the base 2 which are denoted by $\log_2$).

As usual, high precision means divergence close to zero.

We will also consider methods of prediction which allow randomization. Then, we emphasize that $p^*_\gamma(a|x_1\ldots x_t);a\in A$ is a random distribution (even for fixed $x_1\cdots x_t$).

Thus, for these methods we define the precision as follows

$$\begin{array}{l} r_{\gamma,p}(x_1\ldots x_t)= \\ E_\gamma(D(p(\cdot|x_1\ldots x_t)\|p^*_\gamma(\cdot|x_1\ldots x_t))= \\ \sum_{a\in A}p(a|x_1\ldots x_t)E_\gamma\left(\log\frac{p(a|x_1\ldots x_t)}{p^*_\gamma(a|x_1\ldots x_t)}\right), \end{array} \qquad (1)$$

where E_γ denotes mean value.

Note that $r_{\gamma,p}$ may also be considered as the *redundancy* when the prediction is used for coding. Let us comment on the relation to coding in more detail. We use p to stand for the true conditional distribution $p(\cdot|x_1\cdots x_t)$ and p^* to stand for the corresponding prediction (possibly chosen after invoking randomization). An observer can construct a (prefix-free) code with codelength

$\kappa^*(a|x_1\cdots x_t) \approx -\log_2 p^*(a|x_1\cdots x_n)$ for any letter $a \in A$ (since Shannon's original research, it has been well known, cf. e.g. Gallager [9] or Cover and Thomas [4], that, using block codes with large block length or more modern methods of arithmetic coding, the approximation may be as accurate as you like). An ideal observer would base coding on the true distribution p and not on the prediction p^*. The difference in performance measured by average code length is given by

$$\sum_{a\in A} p(a|x_1\cdots x_t)(-\log_2 p^*(a|x_1\cdots x_t))$$

$$-\sum_{a\in A} p(a|x_1\cdots x_t)(-\log_2 p(a|x_1\cdots x_t))$$

$$=\sum_{a\in A} p(a|x_1\cdots x_t)\log_2\frac{p(a|x_1\cdots x_t)}{p^*(a|x_1\cdots x_t)}.$$

Thus this excess, the *redundancy*, is – apart from the unit in bits rather than in natural units – exactly the precision defined above. We shall in fact mostly refer to redundancy rather than divergence or precision in what follows.

In the final section with concluding remarks we shall return to coding aspects and include a discussion of adaptive coding.

For fixed t, $r_{\gamma,p}$ is a random variable. This has nothing to do with the use of randomization and only reflects that $x_1, x_2, \cdots, x_t$ are random variables.

We define the *average divergence* (at time t) by

$$D^t(p\|\gamma) = E_{p^t}\,(r_{\gamma,p}(\cdot)) = \sum_{x_1\cdots x_t\in A^t} p(x_1\cdots x_t)r_{\gamma,p}(x_1\cdots x_t) \qquad (2)$$

Related to this quantity we define the *maximum average divergence* (at time t) by

$$D^t(M\|\gamma) = \sup_{p\in M} D^t(p\|\gamma) \qquad (3)$$

and the *limiting maximum average divergence* by

$$D^\infty(M\|\gamma) = \limsup_{t\to\infty} D^t(M\|\gamma). \qquad (4)$$

We also refer to $D^\infty(M\|\gamma)$ simply as *limiting redundancy*. The dependence on the method and on the model can be emphasized by speaking of *limiting redundancy of the method γ under the model M*.

It is important to develop methods which, in principle, can be realized on any computer. Therefore, a method γ will also depend on a parameter relating to the computer available. This parameter could be the memory size. For the methods we shall discuss it is more convenient to use a parameter w expressing the size of a *window* $x_{t-w+1}\cdots x_t$ which is used for the prediction. The asymptotic properties which we shall study investigates the performance of a method as given by limiting redundancy when the method is realized on more and more powerful computers. Equivalently, and this will be our preferred view, one may study the necessary requirements on memory space and time in order to achieve a lower and lower limiting redundancy.

3 A method which is asymptotically optimal on the average for Bernoulli sources

We shall describe a method α_0 which is based on results from universal coding theory, cf.[12].

The method is based on the frequencies of the letters in A in a window $x_1\cdots x_w$ of size w. By $\nu_a(x_1\cdots x_w)$ we denote the *frequency* of a in $x_1\cdots x_w$, i.e. the number of $i \le w$ with $x_i = a$. For the method α_0, $x_1\cdots x_w$ is used to obtain the frequencies $\nu_a(x_1\cdots x_w)$ and these will then be the basis for prediction of $x_{w+1}, x_{w+2}, \cdots$ according to the formula

$$p^*_{\alpha_0}(a|x_1\cdots x_t) = \frac{\nu_a(x_1\cdots x_w)+1}{w+n}; \quad t \ge w. \qquad (5)$$

As we are only interested in limiting behaviour, it is not important how $x_1, \cdots, x_w$ are predicted.

In order to store the numbers $\nu_a(x_1\cdots x_w)$; $a \in A$ we could require $S_w = n\lceil\log_2(w+1)\rceil$ or just $(n-1)\lceil\log_2(w+1)\rceil$ bits. In principle, we could have been less demanding and only required a memory of $\lceil\log_2\binom{w+n-1}{n-1}\rceil$ bits. However, this might increase computing time since the actual frequencies $\nu_a(x_1,\cdots,x_w)$ may then not be so easy to access. We also point out that we consider n as fixed whereas larger and larger w will be involved. Therefore, the quantities suggested basically only differ by an additive constant. Furthermore, we are not interested in the fine details regarding actual implementations on a computer. Taking the above considerations into account, we simplify the discussion by taking

$$S(w) = (n-1)\log_2 w \qquad (6)$$

as the required memory size.

When the computer presents the results (5) of the prediction at a given time instant, we assume that the probabilities involved are represented (i.e. printed) as fractions. The time needed for this is $c \cdot n \cdot \lceil \log_2(w+1) \rceil$ counted in bit operations. Here c is a constant which is characteristic of the actual computer used. We again simplify and take

$$T(w) = O(\log w) \tag{7}$$

to be the time requirement.

We need some lemmas.

Lemma 1. For the method α_0, the limiting redundancy for the Bernoulli model $M_0(A)$ can be upper bounded as follows:

$$D^\infty(M_0(A)\|\alpha_0) \leq (n-1)/(w+1) \tag{8}$$

Proof. Though this is known from [17], we give the details of the proof for the convenience of the reader.

Consider a Bernoulli source p and an integer $t \geq w$. We employ the general inequality

$$D(\mu\|\eta) \leq -1 + \sum_{a \in A} \mu(a)^2/\eta(a),$$

valid for any distributions μ and η over A (follows from the elementary inequality $\log x \leq x - 1$), and find:

$$\begin{aligned} D^t(p\|\alpha_0) &= E_{p^t} D(p(\cdot \mid x_1 \cdots x_t)\|p^*(\cdot \mid x_1 \cdots x_t)) \\ &= E_{p^w}(D(p\|p^*(\cdot \mid x_1 \cdots x_w)) \\ &\leq -1 + \sum_{x_1 \cdots x_w \in A^w} p(x_1 \cdots x_w) \\ &\quad \sum_{a \in A} \frac{p(a)^2(w+n)}{\nu_a(x_1 \cdots x_w)+1} \\ &= -1 + \sum_{a \in A} \sum_{i=0}^{w} \frac{p(a)^2(w+n)}{i+1} \end{aligned}$$

$$\binom{w}{i} p(a)^i (1-p(a))^{w-i} = -1 + \frac{w+n}{w+1} \sum_{a \in A} p(a)$$

$$\sum_{i=0}^{w} \binom{w+1}{i+1} p(a)^{i+1}(1-p(a))^{w-i}$$

$$\leq -1 + \frac{w+n}{w+1} \sum_{a \in A} p(a)$$

$$\sum_{j=0}^{w+1} \binom{w+1}{j} p(a)^j (1-p(a))^{w+1-j} = \frac{n-1}{w+1}.$$

As this holds for any $t \geq w$ and any Bernoulli source p, (8) follows. □

It may be remarked that if, instead of (5), we define the method α' by the formula

$$p^*_{\alpha'}(a|x_1 \cdots x_t) = \frac{\nu_a(x_1 \cdots x_w) + \frac{1}{2}}{w + n/2}; \quad t \geq w,$$

then the redundancy for small values of w will be less than for α_0 (see [12], [17]), however, the asymptotic properties (for memory size going to infinity) will not be changed, whereas the analysis needed to replace Lemma 1 will be more complicated.

In order to reach a given (small) value of $r = D^\infty(M_0(A)\|\alpha_0)$, a certain size of the window is required as shown by (8). By (6) this imposes a condition on the memory size S_{α_0} demanded by the method. Thus, the required memory size may be considered to be a function of $r = D^\infty(M_0(A)\|\alpha_0)$ and we may write $S = S_{\alpha_0}(r)$. In the same manner, the time of prediction may be considered to be a function of r: $T = T_{\alpha_0}(r)$. Similar considerations apply to any method γ.

Lemma 1 and the considerations above suffice in order to establish the appropriate upper bounds for the complexity of the method α_0. However, we must also develop results that permit the derivation of lower bounds and these results must apply to any method. Otherwise, (near–) optimality of any particular method, such as e.g. α_0, cannot be ascertained.

The technique we shall use in the search for lower bounds uses the notions of ε-capacity and ε-nets developed by Kolmogorov and Tihomirov [11]. Let $M(A)$ denote the set of probability distributions over the alphabet A and consider an $\varepsilon > 0$. A subset $\Gamma \subseteq M(A)$ is a $2\varepsilon-$ *net* for $M(A)$ if, for every $p, q \in \Gamma$ with $p \neq q$ there does not exist a distribution $\lambda \in M(A)$ for which both $D(p\|\lambda) < \varepsilon$ and $D(q\|\lambda) < \varepsilon$ hold. By $C_\varepsilon(M(A))$ we denote the *ε-capacity* of $M(A)$ defined as the maximum of all numbers $\log N$ for which there exists a 2ε-net $\Gamma \subseteq M(A)$ with N elements.

Before we state the capacity bound we need, it is convenient to point out the following auxiliary result:

Lemma 2. Let $p = (p_1, \cdots, p_n), q = (q_1, \cdots, q_n)$ and $\pi = (\pi_1, \cdots, \pi_n)$ be probability distributions. Then

$$\max(D(p\|\pi), D(q\|\pi)) \geq \frac{1}{8}\|p - q\|^2 \tag{9}$$

with $\|\cdot\|$ denoting l_1-norm (total variation).

Proof. By Pinsker's inequality (cf. Csiszár and Körner [5]), $D(p\|\pi) \geq \frac{1}{2}\|p - \pi\|^2$ and $D(q\|\pi) \geq \frac{1}{2}\|q - \pi\|^2$. Then

$$\frac{1}{8}\|p - q\|^2 \leq \frac{1}{8}(\|p - \pi\| + \|q - \pi\|)^2 \leq$$

$$\frac{1}{8}(2\max\|p-\pi\|,\|q-\pi\|)^2 \leq$$

$$\max\frac{1}{2}\|q-\pi\|^2) \leq$$

$$\max(D(p\|\pi), D(q\|\pi)). \qquad \square$$

We can then prove a key lemma:

Lemma 3.

$$C_\varepsilon(M(A)) \geq \frac{n-1}{2}\log_2\frac{1}{\varepsilon} + O(1). \tag{10}$$

Proof. Put $\delta = \sqrt{8\varepsilon}$ and denote by $\Gamma \subseteq M(A)$ the set of distributions $p = (p_1, \cdots, p_n)$ such that all coordinates p_i with $i \leq n-1$ are of the form $p_i = k_i\delta$ with $k_1, \cdots, k_{n-1}$ non-negative integers. We show that Γ is a 2ε-net. Indeed, if $p \in \Gamma, q \in \Gamma$ and $p \neq q$, there exists $i \leq n-1$ such that $p_i \neq q_i$. Then $|p_i - q_i| \geq \delta$, hence we obtain from Lemma 2 that for any distribution π,

$$\max(D(p\|\pi), D(q\|\pi)) \geq \frac{1}{8}\delta^2 = \varepsilon,$$

and it follows that Γ is a 2ε-net. Thus $C_\varepsilon(M_0(A)) \geq \log|\Gamma|$ ($|\cdot|$ denoting "number of elements in").

Let $K = \lfloor\frac{1}{\delta}\rfloor$ and denote by $a_{n-1}(K)$ the number of solutions of the inequality $k_1 + \cdots + k_{n-1} \leq K$ with all the k_i's non-negative integers. It is then clear that $|\Gamma| \geq a_{n-1}(K)$. Now,

$$a_{n-1}(K) = \sum_{j=0}^{K}\binom{j+n-2}{n-2} \geq \frac{1}{(n-2)!}$$

$$\sum_{j=1}^{K+1} j^{n-2} \geq \frac{1}{(n-2)!}\int_0^{K+1} x^{n-2}dx$$

$$= \frac{(K+1)^{n-1}}{(n-1)!}.$$

Putting things together, we find that

$$C_\varepsilon(M_0(A)) \geq \log\frac{1}{\delta^{n-1}(n-1)!} =$$

$$\frac{n-1}{2}\log\frac{1}{\varepsilon} + O(1)$$

as claimed. $\square$

We may note that the $O(1)$-term in Lemma 3 is approximately $-n\log n$ (apply Stirling's formula).

The theorem below shows that the method α_0 is close to being optimal for $r \to 0$.

Theorem 1. (i). For $r \to 0$, where $r = D^\infty(M_0(A)\|\alpha_0)$, we have

$$S_{\alpha_0}(r) \leq (n-1)\log_2\frac{n-1}{r} =$$

$$(n-1)\log_2\frac{1}{r} + O(1) \tag{11}$$

and

$$T_{\alpha_0}(r) = O(\log\frac{1}{r}). \tag{12}$$

(ii). For any prediction method γ we have for $r \to 0$, where $r = D^\infty(M_0(A)\|\gamma)$,

$$S_\gamma(r) \geq \frac{n-1}{2}\log_2\frac{1}{r} + O(1), \tag{13}$$

$$T_\gamma(r) \geq c\left(\log\frac{1}{r}\right), \tag{14}$$

where c is a positive constant (also depending on n).

Proof. The proof of (i) was indicated above in connection with Lemma 1.

In order to establish the more difficult part (ii) of the theorem, consider an $r > 0$ and any method γ of prediction with $D^\infty(M_0(A)\|\gamma) = r$. We shall prove that (11) and (12) hold for any such method. In order to make the ideas of the proof clear, we first treat the simpler case when γ is a deterministic method. In that case the formula

$$D^t(p\|\gamma) = E_{p^t}D(p\|p^*_\gamma(\cdot|x_1\cdots x_t)) \tag{15}$$

holds for all t and $p \in M(A)$. Choose t so large that $D^t(p\|\gamma) \leq r$ for all $p \in M(A)$. Then, for each p, there must exist at least one string $x_1\cdots x_t$ such that $D(p\|p^*_\gamma(\cdot|x_1\cdots x_t)) \leq r$.

Now, let $\Gamma = \{p_1, \cdots, p_N\}$ be a $2r$-net. We can then find strings $x^i_1\cdots x^i_t; i \leq N$ such that

$$D(p_i\|p^*_\gamma(\cdot|x^i_1\cdots x^i_t)) \leq r\,; \quad i = 1, \cdots, N.$$

According to the definition of a $2r$-net, the distributions $p^*_\gamma(\cdot|x^i_1\cdots x^i_t);\ i \leq N$ must be distinct, hence the N input strings $x^i_1\cdots x^i_t;\ i \leq N$ must be distinct too and the computer must be able to distinguish between them. This means that the computer must have a memory of at least $\log_2 N$ bits. As N may be chosen equal to $\exp C_r(M(A))$, (13) now follows from (10).

Then we consider the general case of a method which may involve randomization. In that case we find from (1) and (2) that

$$D^t(p\|\gamma) = E_{p^t}E_\gamma D(p\|p^*_\gamma(\cdot|x_1\cdots x_t)). \tag{16}$$

By Jensen's inequality in the form $E(\log\frac{1}{x}) \geq \log\frac{1}{E(x)}$ we obtain

$$D^t(p\|\gamma) \geq E_{p^t}D(p\|\lambda_\gamma(\cdot|x_1\cdots x_t)) \tag{17}$$

where we have defined $\lambda_\gamma(\cdot|x_1\cdots x_t) \in M(A)$ by

$$\lambda_\gamma(a|x_1\cdots x_t) = E_\gamma p^*_\gamma(a|x_1\cdots x_t)\,; \quad a \in A.$$

Choose t so large that $D^t(p\|\gamma) \le r$ for all $p \in M_0(A)$. Again, let $\Gamma = \{p_1, \cdots, p_N\}$ be a $2r$-net. Let us consider a distribution $p_i \in \Gamma$. Then there exists at least one string $x_1^i \cdots x_t^i$ for which

$$E_\gamma D(p_i \| p_\gamma^*(\cdot|x_1^i \cdots x_t^i)) \le r.$$

From (17) and the assumption about the performance of the method γ we obtain the inequality

$$D(p_i \| \lambda_\gamma(\cdot|x_1^i \cdots x_t^i)) \le r.$$

We should note that

$$D(p_j \| \lambda_\gamma(\cdot|x_1^i \cdots x_t^i)) > r$$

for all $p_j \in \Gamma$, $j \ne i$ because Γ is a $2r$-net.

The (deterministic) distributions $\lambda_\gamma(\cdot|x_1^i \cdots x_t^i)$, $i \le N$ are thus all different, hence also the (random) distributions $p_\gamma^*(\cdot|x_1^i \cdots x_t^i)$, $i \le N$ are different. So we found N input strings such that the N output objects provided by the method γ (the above indicated random distribution) are all distinct. The computer must therefore be able to distinguish between these N input strings, and this requires a memory of at least $\log_2 N$ bits. The lower bound (13) now follows as before by reference to (10).

As to the lower bound (14), this is directly connected with the proof of (13). Indeed, every prediction method must be able to print the distribution which is predicted at each time instant. And, as we saw in the proof of (11), if $D^\infty(M_0(A)\|\gamma) \le r$, then, at some time instant t, the method could lead to any of $\exp C_r(M(A))$ many distributions (whether random or not) as the distribution predicted for x_{t+1}. So the computer must contain a code to distinguish between these distributions. Such a code must contain a codeword of bit length at least $\log_2 \exp C_r(M(A))$. If the corresponding distribution is to be printed – and the computer must be capable of doing that – then a look-up of the codeword in question is required, and this takes time measured in bit operations of at least the bit length of the codeword. Therefore, for a positive constant which is characteristic for the computer used,

$$T_\gamma(r) \ge c \cdot \log_2 \exp C_r(M(A))$$

and (14) follows from (10). □

Remark. An inspection of the proof shows that the given lower bounds hold in a slightly more general setting for which lim sup in the defining relation (4) is replaced by lim inf.

4 A method which is asymptotically optimal with probability one

First, we notice that the method α_0 has a serious shortage as the first w letters on which the method is based could exhibit a bad statistics. On first sight, it is tempting to improve on this by using the window $x_{t-w+1} \cdots x_t$ rather than $x_1 \cdots x_w$ for the prediction of x_{t+1}. With this change, the method will be good with probability one and not only in average. But then, after predicting every letter x_{t+1}, we would have to move the window: x_{t+1} should be included in the window and x_{t-w+1} removed. This will require $w \log_2 n$ bits rather than the $n \log_2 w$ bits which suffice for α_0. In more detail, the point is that for any of the w positions making up the window, we would have to know which letter was observed at that time instant , so that even if $n = 2$, we would need w bits of memory for this purpose. When the precision r goes to 0, the parameter w goes to infinity and the method with a sliding window would need *constant*$/r$ bits of memory as compared to $(n-1)\log(1/r)$ bits for α_0.

In order to preserve the relatively small demand for memory for α_0 while at the same time improving the prediction by using also the later letters, we propose to use the method β_0 of the *imaginary sliding window* from [19] (related methods are used in computer science, e.g. regarding so-called *paging*, cf. [14], Chapter 13).

At each time instant t (we need only worry about $t \ge w$) we keep track of certain numbers $\nu^t(a_1), \cdots, \nu^t(a_n)$, which we think of as frequencies, and use these to predict x_{t+1} according to the formula

$$p_{\beta_0}^*(a|x_1 \cdots x_t) = \frac{\nu^t(a) + 1}{w + n}; \quad t \ge w.$$

The frequencies $\nu^t(a)$ are random variables as they depend on $x_1, \cdots, x_t$. We shall now explain how these frequencies are defined.

The starting frequencies $\nu^w(a_1), \cdots, \nu^w(a_n)$ are the true frequencies in the window $x_1 \cdots x_w$. The new feature of β_0 is that after prediction of x_{t+1} $(t \ge w)$, we change the frequencies as follows: First, we consult the oracle and choose a letter at random according to the probabilities $\nu^t(a)/w$, $a \in A$. If a_j is chosen, we define the new frequency for this letter using the convention $\nu^{t+1}(a_j) = \nu^t(a_j) - 1$. After this, we add 1 to the frequency of x_{t+1} $(\nu^{t+1}(x_{t+1}) = \nu^t(x_{t+1}) + 1)$. If $a \ne x_{t+1}$ and $a \ne a_j$,

we do not change its frequency ($\nu^{t+1}(a) = \nu^t(a)$). The new frequencies $\nu^{t+1}(a_i); i \leq n$ are then used to predict x_{t+2}, and after this the frequencies are again changed, x_{t+3} is predicted and so on.

The memory size required for the method β_0 is $(n-1)\lceil \log_2(w+1) \rceil$ bits, and can more conveniently be taken to be

$$S_w = (n-1)\log_2 w \tag{18}$$

just as for the method α_0, cf. (6). However, in taking (18) as expression for the memory size, we ignore the (asymptotically negligible) demand for memory resulting from the use of a random number generator. This is also justifyed by the considerations detailed in the introduction.

When estimating the time of prediction, we likewise ignore the complexity of a random number generator and only consider the transformation to values with probabilities $\nu^t(a)/w; a \in A$. Using fast methods of transformation as described in [18], one finds that the average time of transformation per letter is $O(\log w)$ bit operations.

Using properties of the imaginary sliding window, see [19] and also the remarks below, and appealing to calculations similar to those for the proof of Theorem 1, we now obtain the following result:

Theorem 2. Let $r > 0$ be given, put $w = \lceil (n-1)\log_2 e/r \rceil$ and apply the method β_0 of prediction with this parameter value w. Then, for every $p \in M_0(A)$,

$$Pr\left(\limsup_{t\to\infty} r_{\beta_0,p}(x_1,\cdots,x_t) \leq r\right) = 1,$$

$$S_{\beta_0}(r) \leq (n-1)\log_2 \frac{1}{r} + O(1),$$

$$T_{\alpha_0}(r) = O\left(\log \frac{1}{r}\right).$$

Remarks. As the lower bounds (13) and (14) still apply, the result shows that the method β_0 is not all that far from being optimal. A remark on the proof is also in place. It is essential that the statistics of the numbers $\nu^t(a_i)$ really do behave as they should, i.e. that they converge to the corresponding true frequencies. We point out that the proof of this crucial fact is easily accomplished when transforming the problem in a natural way into a problem of calculating the invarient distribution associated with the Markov chain which models the updating method of the imaginary sliding window.

5 Markov sources

In §2 we indicated that extensions of the key results to cover the general Markov case are possible. We take this up now. The trick is to view a Markov source $p \in M_k(A)$ as resulting from $|A|^k$ Bernoulli sources. We illustrate this idea by an example.

So assume that $A = \{O, I\}$, $k = 2$ and assume that the source $p \in M_2(A)$ has generated the sequence

$$OOIOIIOOIIIOIO.$$

We represent this sequence by the following four subsequences:

$$**I*****I*****,$$

$$***O*I***I***O,$$

$$****I**O****I*,$$

$$******O***IO**.$$

These four subsequences contain letters which follow after OO, after OI, after IO and after II, respectively. By definition, $p \in M_k(A)$ if $p(a|x_1\cdots x_t) = p(a|x_{t-m+1}\cdots x_t)$, for all $k \leq m \leq t$, all $a \in A$ and all $x_1\cdots x_t \in A^t$. Therefore, each of the four generated subsequences may be considered to be generated by a Bernoulli source. Further, it is possible to reconstruct the original sequence if we know the four ($= |A|^k$) subsequences and the two ($= k$) first letters of the original sequence.

In order to predict, it is enough to store in the memory $|A|^k$ methods for Bernoulli sources (α_0 or β_0), one corresponding to each word in A^k. Thus, in the example, the letter x_3 which follows after OO is predicted based on the Bernoulli method corresponding to the OO- subsequence ($= II$), then x_4 is predicted based on the Bernoulli method corresponding to x_2x_3, i.e. to the OI- subsequence ($= OIIO$), and so forth. It will not be important how to predict x_1x_2 or, in general, $x_1\cdots x_k$.

The methods for $M_k(A)$ which are obtained in this way by using either α_0 for all $|A|^k$ subsequences or else β_0 for all these subsequences, we denote by α_k and β_k, respectively. For the associated memory size and the associated average time of calculation by letter we find the following result:

Theorem 3. (i). Denoting the limiting average divergence for the method α_k by r, then, as $r \to 0$,

$$S_{\alpha_k}(r) = n^k(n-1)\log_2 \frac{1}{r} + O(1) \tag{19}$$

and

$$T_{\alpha_k}(r) = O\left(\log \frac{1}{r}\right). \tag{20}$$

(ii). If, for the method β_k,

$$Pr\left(\limsup_{t\to\infty} r_{\beta_k,p}(x_1\cdots x_t) \leq r\right) = 1,$$

then, as $r \to \infty$,

$$S_{\beta_k}(r) \le n^k (n-1) \log_2 \frac{1}{r} + O(1) \qquad (21)$$

and

$$T_{\beta_k}(r) = O\left(\log \frac{1}{r}\right). \qquad (22)$$

(iii). For every prediction method γ for $M_k(A)$,

$$S_\gamma(r) \ge \frac{n^k (n-1)}{2} \log_2 \frac{1}{r} + O(1) \qquad (23)$$

and

$$T_\gamma(r) \ge c\left(\log \frac{1}{r}\right), \qquad (24)$$

where c is a positive constant (which also depends on n and k).

Proof. As it was shown above , every Markov source $p \in M_k(A)$ can be presented as resulting from $|A|^k$ Bernoulli sources.

So, the computer has to store $|A|^k$ tables in order to predict each letter x_t using the corresponding above described method for $M_0(A)$. That is why the memory size required for the suggested method is $|A|^k$ times more than for the corresponding method for a single Bernoulli source, but the time of prediction is asymptotically the same. It is not important how to predict the first letters $x_1, \cdots, x_k$, because we are interested in asymptotic behaviour of the method. For example, we can ascribe equal probabilities $|A|^{-k}$ to all strings $x_1 \cdots x_k$.

The same representation of a Markov source may be used in order to obtain the lower bounds . In fact, in the case of Markov sources, the dimension of the parameter space is equal to $(|A|-1)|A|^k$, unlike the Bernoulli case when the dimension is $|A|-1$. Using a similar estimation of the ε-capacity as before, we can obtain the lower bounds of the memory size. □

6 Concluding remarks

From a practical point of view, the β–methods have an extra advantage over the α–methods which is connected also with the fact that the β–methods are good with probability 1 whereas α–methods are only good in average. The point is that the very nature of the β–methods make them robust to changes in time of the statistics of the source.

Note that the methods developed (α as well as β) were designed solely with the aim of obtaining good methods for special stationary sources. Clearly, further research should broaden the scope. But even though we did not aim at developing methods for non-stationary sources it so happens that the notion of the (imaginary) sliding window is perfectly suited to handle sources with changes in time of the statistics. This paper is focused on theoretical considerations and has the primary aim to study performance in terms of complexity, especially we have endeavoured to initiate research which gives tight lower bounds of performance possibilities which are close to the optimally achievable.

On the more practical side, it would be interesting to study the performance of our methods and methods developed by other authors in real experiments based on real data.

Acknowledgements. The authors had helpfull discussions with Joe Suzuki in Novosibirsk, July 1998, and later with Jyrki Katajainen, Peter Andreasen and Peter Harremoës. Some of the comments received resulted in clarifications regarding the use of randomization. Thanks are due to two referees who requested certain expansions and rearrangements of the material which has made the paper more readable and self-contained. The meticulous work of the referees is gratefully acknowledged.

References

1. **Aho A.V., Hopcroft J.E. and Ullman J.D.**, *The design and analysis of computer algorithms.* Addison-Wesley, Cambridge, 1976.

2. **Algoet P.**, *Universal Schemes for Learning the Best Nonlinear Predictor Given the Infinite Past and Side Information*, IEEE Trans. Inform. Theory, v. 45, pp. 1165-1185, 1999.

3. **Cover T.M., Freedman A.M. and Hellman M.E.**, *Optimal Finite Memory Learning Algorithms for the Finite Sample Problem*, Information and Control, v. 30, pp. 49-85, 1976.

4. **Cover T.M. and Thomas J.A.**, *Elements of Information Theory.* John Wiley & Sons, New York, 1991.

5. **Csiszár I. and Körner J.**, *Information Theory, Coding Theorems for Discrete Memoryless Systems.* Akademiai Kiado, Budapest 1981.

6. **Feder M. and Merhav N.**, *Universal prediction*, IEEE Trans. Inform. Theory, v. 44, pp. 2124-2147, 1998.

7. **Feder M., Merhav N. and Gutman M.**, *Universal prediction of individual sequences*, IEEE Trans. Inform. Theory, v. 38, pp. 1258-1270, 1992.

8. **Feller W.**, *An Introduction to Probabability Theory and Its Applications*, vol.1. John Wiley & Sons, New York, 1970.

9. **Gallager R. G.**, *Information Theory and Reliable Communication*. John Wiley & Sons, New York, 1968.

10. **Kelly J.L.**, *A new interpretation of information rate*, Bell System Tech. J., v. 35, pp. 917-926, 1956.

11. **Kolmogorov A.N. and Tihomirov V.M.**, *Epsilon-entropy and epsilon-capacity of sets in metric spases*, Uspechi Math. Nauk. v. 14, pp. 2-86, 1959. (Russian original, translated into English).

12. **Krichevsky R.**, *Universal Compression and Retrival*. Kluver Academic Publishers, Dordrecht, 1994.

13. **Krichevskii R.**, *Laplace's Law of Succession and Universal Encoding*, IEEE Trans. Inform. Theory, v. 44, pp. 296-303, 1998.

14. **Motwani R. and Raghavan, P.**, *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995.

15. **Rissanen J.**, *Universal coding, information, prediction, and estimation*, IEEE Trans. Inform. Theory, v. 30 pp. 629-636, 1984.

16. **Ryabko B. Ya.**, *The complexity and effectiveness of prediction algorithms*, J. of Complexity, v. 10, pp. 281-295, 1994.

17. **Ryabko B. Ya.**, *Prediction of random sequences and universal coding*, Problems Inform. Transm. v. 24, pp. 87-96, 1988.

18. **Ryabko B. Ya.**, *A fast on-line adaptive code*, IEEE Trans. Inform. Theory, v. 38, pp. 1400-1404, 1992.

19. **Ryabko B. Ya.**, *Data compression by "an imaginary sliding window"*, Problems Inform. Transmission v. 32, pp. 156-163, 1996.

20. **Topsøe F,** *Information Theoretical Optimization Techniques*, Kybernetika, v. 15, pp. 8-27, 1979.

21. **Topsøe F.,** *Game Theoretical Equilibrium, Maximum Entropy and Minimum Information Discrimination*, in Mohammad-Djafari et al (eds.): Maximum Entropy and Bayesian Methods. Kluwer, Dordrecht, 1993.

22. **Verdu S.**, *Fifty years of Shannon Theory*, IEEE Trans. Inform. Theory, v. 44, pp. 2057-2078, 1998.