KYBERNETIKA — VOLUME 40 (2004), NUMBER 6, PAGES 703 - 714

EFFICIENCY OF SOME ALGORITHMS FOR PREDICTION IN FINITE STATIONARY TIME SERIES

PAVEL RANOCHA

Important characteristics of any algorithm are its complexity and speed of real calculations. From this point of view we analyze some algorithms for prediction in finite stationary time series. First, we review results developed by Bondon [1] and then we derive complexities of Levinson and innovations algorithm. It is shown that time needed for real calculations of prediction is proportional to theoretical complexity of the algorithm. Some practical recommendations for selection of the best algorithm are given.

Keywords: stationary time series, multistep prediction, Levinson's algorithm, innovations algorithm

AMS Subject Classification: 60G25

1. INTRODUCTION

Let $\{X_n, n \in \mathbb{N}\}$ be real-valued, (weakly) stationary process with zero mean and covariance function $\gamma(k)$ defined on probability space $(\Omega, \mathcal{A}, \mathsf{P})$. Let $L^2(\Omega, \mathcal{A}, \mathsf{P})$ denote the Hilbert space with inner product $\langle X, Y\rangle = \mathsf{E}XY$. Let $H\{X_n, n \in M\}$ be the Hilbert subspace of L^2 generated by variables $X_n, n \in M$. If $M = \{l, \dots, n\}$, we simply write $H\{X_n, n \in M\} = H_{l,n}$. We use the symbol $P_{l,n}$ for orthogonal projection operator onto $H_{l,n}$. Finally, assume that $H_{1,n} \subsetneq H_{1,n+1}$ for all $n \in \mathbb{N}$.

It is well-known that if we write the optimal linear prediction of variable X_{n+h} based on the knowledge of $X_1, \dots, X_n$ in the form

$$\hat{X}_{n+h}(n) = P_{1,n}X_{n+h} = \sum_{i=1}^{n} a_{n,i}^h X_{n+1-i},$$

then the prediction coefficients $\boldsymbol{a}_n^h = (a_{n,1}^h, \dots, a_{n,n}^h)^\mathsf{T}$ can be obtained by solving the system of linear equations

$$\boldsymbol{\Gamma}_n \boldsymbol{a}_n^h = \boldsymbol{\gamma}_{n,h}, \tag{1.1}$$

where

$$\boldsymbol{\Gamma}_n = \big(\gamma(i-j)\big)_{i,j=1}^n, \quad \boldsymbol{\gamma}_{n,h} = \big(\gamma(n+h-1), \dots, \gamma(h)\big)^\mathsf{T}. \tag{1.2}$$

Mean square error is given by

$$v_n^h = \mathsf{E}[X_{n+h} - \hat{X}_{n+h}(n)]^2 = \gamma(0) - \boldsymbol{\gamma}_{n,h}^{\mathsf{T}} \boldsymbol{\Gamma}_n^{-1} \boldsymbol{\gamma}_{n,h}.$$

The main disadvantage of the direct method is its high numerical complexity. It is necessary to find the solution of the system, the dimension of which is n, where n is usually large. Moreover, if we get a new observation, we have to repeat the whole procedure.

It is natural to ask whether it is possible to solve the described problem more efficiently. Probably, the first effective method derived for the construction of predictions with finite past was the Levinson algorithm (see [5]). It is based on the existence of relations between prediction coefficients recurrent with respect to number of observations which is equal to order of matrix $\boldsymbol{\Gamma}_n$. Innovations algorithm, which was derived later (see e. g. [4]), works with the properties of projection operator and orthogonal decomposition but it does not use the assumption of stationarity. That is why its complexity is still very high (see below).

The procedures using also recursion with respect to prediction step were deduced by Bondon in [1]. The author derived several methods enumerating prediction coefficients and mean square errors. (Similar relations for infinite time series were derived earlier, see e. g. [2].) Quite recently, Brockwell and Dahlhaus [3] deduced some recursive properties of orthogonal projections which lead to a variety of different prediction algorithms (e. g. Durbin–Levinson, Burg and Whittle algorithms).

For the calculation of their numerical complexity, Bondon supposed that multiplications (divisions) are much more time demanding than summations (subtractions). No other operations occur. With the help of our programme implementation we show that these simplifying assumptions do not effect the results significantly and theoretical complexities computed on their basis may be used to compare the effectiveness of the rated procedures. We use the same programme to measure time needed for the calculations. In conclusion we deduce the numerical complexity of Levinson and innovations algorithm and find out that their efficiency is far beyond Bondon's methods.

2. CLASSICAL METHODS

Levinson's algorithm (see [5]), which we describe now, can be generally used for solving the system of linear equations with so called Toeplitz matrix. It is a square matrix with elements $t_{i,j}$, for which there exist real numbers $u_{-n+1}, \dots, u_0, \dots, u_{n-1}$ such that $t_{i,j} = u_{i-j}$, $i, j = 1, \dots, n$. It is obvious that the matrix (1.2) satisfies this condition. The system (1.1) can be written in the form

$$\sum_{n=0}^{M} a_{M,n} \gamma(k-n) = \delta_k, \quad k = 0, 1, \dots, M,$$

where $\delta_k = \gamma(M + h - k)$. Its solution is given by

$$a_{0,0} = \frac{\delta_0}{\gamma(0)}, \tag{2.1}$$

$$a_{M+1,M+1} = \frac{\delta_{M+1} - \sum_{k=0}^{M} a_{M,k}\gamma(M+1-k)}{\gamma(0) - \sum_{k=0}^{M} C_k^M \gamma(M+1-k)} \tag{2.2}$$

and

$$a_{M+1,k} = a_{M,k} - C_k^M a_{M+1,M+1}, \quad k = 0, 1, \ldots, M. \tag{2.3}$$

The constants C_k^M are computed from

$$C_0^0 = \frac{\gamma(1)}{\gamma(0)}, \tag{2.4}$$

$$C_0^M = \frac{\gamma(M+1) - \sum_{k=1}^{M} C_{k-1}^{M-1}\gamma(k)}{\gamma(0) - \sum_{k=0}^{M-1} C_k^{M-1}\gamma(M-k)} \tag{2.5}$$

and

$$C_k^M = C_{k-1}^{M-1} - C_0^M C_{M-k}^{M-1}, \quad k = 1, 2, \ldots, M. \tag{2.6}$$

The next method, which was derived later, was the innovations algorithm (see e. g. [4]). If we write the prediction in the form

$$P_{1,n}X_{n+h} = \sum_{j=h}^{n+h-1} \theta_{n+h-1,j}(X_{n+h-i} - \hat{X}_{n+h-j}), \tag{2.7}$$

then the prediction coefficients $\theta_{n,j}$ are obtained from

$$v_0 = \kappa(1,1),$$

$$\theta_{n,n-k} = v_k^{-1}\left(\kappa(n+1,k+1) - \sum_{j=0}^{k-1} \theta_{k,k-j}\theta_{n,n-j}v_j\right), \quad k = 0, 1, \ldots, n-1, \tag{2.8}$$

and

$$v_n = \kappa(n+1,n+1) - \sum_{j=0}^{n-1} \theta_{n,n-j}^2 v_j, \tag{2.9}$$

where $\kappa(t,s) = \mathrm{cov}\,(X_t, X_s)$. Mean square error is computed using

$$v_n^h = \kappa(n+h,n+h) - \sum_{j=h}^{n+h-1} \theta_{n+h-1,j}^2 v_{n+h-j-1}. \tag{2.10}$$

The main disadvantage of this procedure is its high numerical complexity (see further). On the other hand, it can be used more generally, namely when predicting in non-stationary processes. It can also be easily modified for the ARMA processes in order to reduce the number of operations being made during the computations (see [4]).

3. BONDON'S PROCEDURES

In this section we introduce several propositions and some very effective algorithms, which were derived by Bondon, [1].

Proposition 3.1. For any step $h \geq 1$ and any $n \geq 1$,

$$a_{n,i}^h = a_{n+h-1,i+h-1}^1 + \sum_{j=1}^{h-1} a_{n+h-1,j}^1 a_{n,i}^{h-j}, \quad i = 1, \ldots, n. \tag{3.1}$$

Proof. See [1], Proposition 3.1. □

Proposition 3.2. For any step $h > 1$ and any $n \geq 1$,

$$a_{n,i}^h = a_{n+1,i+1}^{h-1} + a_{n+1,1}^{h-1} a_{n,i}^1, \quad i = 1, \ldots, n \tag{3.2}$$

and

$$v_n^h = v_{n+1}^{h-1} + (a_{n+1,1}^{h-1})^2 v_n^1. \tag{3.3}$$

Proof. See [1], Proposition 3.2. □

Proposition 3.3. For any step $h \geq 1$ and any $n \geq 1$,

$$v_0^h = \gamma(0),$$

$$a_{n,n}^h = \left[\gamma(n+h-1) - \sum_{i=1}^{n-1} a_{n-1,i}^1 \gamma(n+h-i-1)\right] (v_{n-1}^1)^{-1}, \tag{3.4}$$

$$a_{n,i}^h = a_{n-1,i}^h - a_{n,n}^h a_{n-1,n-i}^1, \quad i = 1, \ldots, n-1, \tag{3.5}$$

and

$$v_n^h = v_{n-1}^h - (a_{n,n}^h)^2 v_{n-1}^1. \tag{3.6}$$

Proof. See [1], Proposition 4.1. □

The first method described (denoted by A_1) is based on Proposition 3.3 only. Another possibility (A_2) consists of calculating $a^1_{n,i}$ for $1 \leq n \leq p+s-1$ from (3.4)–(3.6) and enumerating $a^h_{p,i}$ for $2 \leq h \leq s$ with the help of (3.1). The mean square errors v^h_p, $2 \leq h \leq s$ can be obtained from

$$v^h_p = \|X_{p+h} - P_{1,p}X_{p+h}\|^2 = \gamma(0) - \sum_{i=1}^{p} a^h_{p,i}\gamma(i+h-1). \tag{3.7}$$

Proposition 3.4. For any step $h > 1$ and any $n > 1$,

$$a^h_{n,i} = a^{h-1}_{n,i+1} + a^{h-1}_{n,1}a^1_{n-1,i} - a^h_{n,n}a^1_{n-1,n-i}, \quad i = 1, \ldots, n-1 \tag{3.8}$$

and

$$v^h_n = v^{h-1}_n + [(a^{h-1}_{n,1})^2 - (a^h_{n,n})^2]v^1_{n-1}. \tag{3.9}$$

Proof. See [1], Proposition 4.2. □

In the first step of A_3 coefficients $a^1_{n,i}$ and errors v^1_n, $1 \leq n \leq p$ are computed from (3.4)–(3.6). In the second stage we use the equalities (3.4), (3.8) and (3.9) to enumerate $a^h_{p,i}$ and v^h_p for $2 \leq h \leq s$.

The next alternative (A_4) is the procedure, in the first stage of which we calculate $a^1_{n,i}$ and v^1_n for $1 \leq n \leq p+s-1$ according to (3.4)–(3.6). These values are used in the second stage, when we get $a^h_{n,i}$ and v^h_n, $2 \leq h \leq s$, $p \leq n \leq p+s-h$ from (3.2) and (3.3).

The next proposition shows a different approach for calculating prediction coefficients based on the orthogonal decomposition of the space $H_{1,n}$.

Proposition 3.5. For any step $h \geq 1$ and any $n \geq 1$,

$$P_{1,n}X_{n+h} = \sum_{i=1}^{n} c^h_i(X_i - P_{1,i-1}X_i) \tag{3.10}$$

and

$$v^h_n = \gamma(0) - \sum_{i=1}^{n}(c^h_i)^2 v^1_{i-1} \tag{3.11}$$

where

$$c^h_i = \left[\gamma(n+h-i) - \sum_{j=1}^{i-1} a^1_{i-1,j}\gamma(n+h-i+j)\right](v^1_{i-1})^{-1}. \tag{3.12}$$

Proof. See [1], Remark 4.2. □

Proposition 3.5 is used to construct algorithm A_5. First, $a^1_{n,i}$ and v^1_n for $1 \leq n \leq p$ are computed from (3.4)–(3.6). The coefficients $a^h_{p,i}$ and errors v^h_p, $2 \leq h \leq s$, are then be obtained according to (3.10)–(3.12).

4. COMPARISON OF EFFICIENCIES

In this section we derive the complexity of both Bondon's and classical algorithms described above, namely innovations and Levinson, and compare them.

When using A_1, the number of multiplications and divisions needed for computation of all the prediction coefficients and mean square errors concerning $\hat{X}_{p+1}(p), \ldots, \hat{X}_{p+s}(p)$ is summarized in Table 1.

Table 1. The complexity of algorithm A_1.

Coefficients to be computed	Used relation	Number of multiplications	Range of indices
$a_{n,n}^h$	(3.4)	n	$n = 1, \ldots, p,\ h = 1, \ldots, s$
$a_{n,i}^h$	(3.5)	$n-1$	$n = 1, \ldots, p,\ h = 1, \ldots, s$
v_n^h	(3.6)	2	$n = 1, \ldots, p,\ h = 1, \ldots, s$

The total complexity of A_1 is

$$N_1 = s \sum_{n=1}^{p} (2n+1) = p^2 s + 2ps.$$

The numerical complexity of procedure A_2 is shown in Table 2.

The total complexity of the algorithm is

$$N_2 = p^2 + \frac{p}{2}(5s + s^2 - 2) + s^2 - 3.$$

The difference between A_1 and A_2 is

$$N_2 - N_1 = p^2(1-s) + \frac{p}{2}(s^2 + s - 2) + s^2 - 3.$$

We can see that the sign of the difference depends on the length of the series p and the maximum step s.

The complexity of operations made when A_3 is used can be found in Table 3.

The total complexity of the procedure is equal to

$$N_3 = p^2 + p(3s-1) + s - 1.$$

Comparing it with the complexity of A_1, we can see that

$$N_1 - N_3 = (s-1)(p^2 - p - 1) > 0,$$

for any $p > 1$. We come to the same conclusion when we compare A_3 and A_2 since for $s > 1$ we have

$$N_2 - N_3 = \frac{ps}{2}(s-1) + (s-2)(s+1) > 0.$$

Table 2. The complexity of algorithm A_2.

Coefficients to be computed	Used relation	Number of multiplications	Range of indices
$a^1_{n,n}$	(3.4)	n	$n = 1,\dots,p+s-1,\ h = 1,\dots,s$
$a^1_{n,i}$	(3.5)	$n-1$	$n = 1,\dots,p+s-1,\ h = 1,\dots,s$
v^h_n	(3.6)	2	$n = 1,\dots,p+s-2,\ h = 1,\dots,s$
$a^h_{p,i}$	(3.1)	$p(h-1)$	$h = 2,\dots,s$
v^h_p	(3.7)	p	$h = 2,\dots,s$

Table 3. The complexity of algorithm A_3.

Coefficients to be computed	Used relation	Number of multiplications	Range of indices
$a^1_{n,n}$	(3.4)	n	$n = 1,\dots,p$
$a^1_{n,i}$	(3.5)	$n-1$	$n = 1,\dots,p$
v^1_n	(3.6)	2	$n = 1,\dots,p$
$a^h_{p,i}$	(3.8)	$2(p-1)$	$h = 2,\dots,s$
v^h_p	(3.9)	3	$h = 2,\dots,s$
$a^h_{p,p}$	(3.4)	p	$h = 2,\dots,s$

The numerical complexity of the next method (A_4) is summarized in Table 4. Its total complexity is

$$N_4 = (p+s-1)^2 + 2(p+s-1) + \sum_{h=2}^{s} \sum_{n=p}^{p+s-h} (n+2).$$

Since

$$\sum_{h=2}^{s} \sum_{n=p}^{p+s-h} (n+2) = \frac{1}{6}(s^3 + 3ps^2 + 3s^2 - 3ps - 4s),$$

we have

$$N_4 = p^2 + \frac{1}{2}ps(s+3) + \frac{1}{6}(s-1)(s^2+10s+6).$$

However, algorithm A_4 is still less effective than A_3, since for every $s \geq 2$,

$$N_4 - N_3 = (s-1)\left[\frac{1}{2}p(s-2) + \frac{1}{6}s\,(s+10)\right] > 0.$$

Table 4. The complexity of algorithm A_4.

Coefficients to be computed	Used relation	Number of multiplications	Range of indices
$a_{n,n}^1$	(3.4)	n	$n = 1, \dots, p+s-1$
$a_{n,i}^1$	(3.5)	$n-1$	$n = 1, \dots, p+s-1$
v_n^1	(3.6)	2	$n = 1, \dots, p+s-1$
$a_{n,i}^h$	(3.2)	n	$n = p, \dots, p+s-h,\ h = 2, \dots, s$
v_n^h	(3.3)	2	$n = p, \dots, p+s-h,\ h = 2, \dots, s$

Table 5. The complexity of algorithm A_5.

Coefficients to be computed	Used relation	Number of multiplications	Range of indices
$a_{n,n}^1$	(3.4)	n	$n = 1, \dots, p$
$a_{n,i}^1$	(3.5)	$n-1$	$n = 1, \dots, p$
v_n^1	(3.6)	2	$n = 1, \dots, p$
$P_{1,i-1}X_i$		$\sum_{i=1}^{p}(i-1)$	
c_i^h	(3.12)	$\sum_{i=1}^{p} i$	$h = 2, \dots, s$
v_p^h	(3.11)	$2p$	$h = 2, \dots, s$

The number of operations made when using A_5 is shown in Table 5.

The total complexity of this procedure is

$$N_5 = p^2\left(\frac{s}{2}+1\right) + p\left(\frac{5s}{2}-1\right).$$

Since for every $p \geq 2$

$$N_5 - N_3 = \frac{s}{2}(p-2)(p+1) + 1 > 0,$$

A_3 is still the most efficient method among those we mentioned.

The number of multiplications made during the application of the innovations algorithm is summarized in Table 6. It is important to realize that coefficients $\theta_{n,n-k}$ and errors v_n^h do not need to be computed for all values of their indices [see (2.7) and (2.10), the lower bound is h, not 1].

Table 6. The complexity of innovations algorithm.

Coefficients to be computed	Used relation	Number of multiplications	Range of indices
$\theta_{n,n-k}$	(2.8)	$2k+1$	$k=0,\dots,\min(p,n)-1,$ $n=1,\dots,p+s-1$
v_n	(2.9)	$2n$	$n=1,\dots,p-1$
$P_{1,i-1}X_i$		$\sum_{i=1}^{p}(i-1)$	
v_p^h	(2.10)	$2p$	$h=1,\dots,s$

The total complexity of the innovations algorithm is equal to

$$N_I = \sum_{n=1}^{p+s-1}\sum_{k=0}^{\min(p,n)-1}(2k+1)+\sum_{n=1}^{p-1}2n+\frac{p}{2}(p-1)+2ps$$
$$= \frac{1}{3}p^3+p^2(s+1)+p\left(2s-\frac{4}{3}\right).$$

When deducing the complexity of the Levinson algorithm, it is essential to realize that the constants C_k^M (for any k, M) have to be calculated only once, since their values are identical for any step h. It results from the fact that they depend only on the elements of the matrix $\mathbf{\Gamma}$ which does not change when h differs. The total complexity of the method is summarized in Table 7.

Table 7. The complexity of Levinson algorithm.

Coefficients to be computed	Used relation	Number of multiplications	Range of indices
C_0^0	(2.4)	1	
C_0^M	(2.5)	$2M+1$	$M=1,\dots,p-1$
C_k^M	(2.6)	1	$k=1,\dots,M,\ M=1,\dots,p-1$
a_0^0	(2.1)	1	$h=1,\dots,s$
a_M^M	(2.2)	$2M+1$	$M=1,\dots,p,\ h=1,\dots,s$
a_k^M	(2.3)	1	$k=0,\dots,M-1,\ M=1,\dots,p,$ $h=1,\dots,s$
v_p^h	(3.7)	p	$h=1,\dots,s$

The total complexity of the algorithm is equal to

$$N_L = \frac{3}{2}p^2(s+1) + \frac{1}{2}p(7s-1) + s.$$

Comparing innovations and Levinson algorithm with the so far best method A_3 we get

$$N_I - N_3 = \frac{1}{3}p^3 + p\left[s(p-1) - \frac{1}{3}\right] - s + 1 > 0$$

for every $p \geq 2$ and

$$N_L - N_3 = \frac{1}{2}p^2(3s-1) + \frac{1}{2}ps + 1 > 0,$$

respectively.

We come to a conclusion that the algorithm denoted A_3 is the most effective among the methods described. On the contrary, the highest number of operations has to be done when applying the innovations algorithm. It is partially caused by its generality. As we noted above, it can be used when predicting in non-stationary time series.

Table 8. The complexity of the algorithms for $p = 50$.

	Maximum step					
Algorithm	1	2	3	5	7	10
A_1	2600	5200	7800	13000	18200	26000
A_2	2598	2801	3056	3722	4596	6297
A_3	2600	2751	2902	3204	3506	3959
A_4	2600	2755	2965	3544	4375	6059
A_5	3825	5200	6575	9325	12075	16200
Innovations	46800	49603	52511	58650	65233	75975
Levinson	7651	11577	15503	23355	31207	42985

To make a better idea about theoretical complexity of described methods, we summarize the results for values $p = 50$ and $p = 200$ in Tables 8 and 9. It is obvious that the difference between the most efficient method A_3 and the classical algorithms is really large.

For the practical application of described methods we chose a series from [2] (p. 525, series A, 11th–60th observation), which was identified as ARMA(1,1) with parameters $\varphi_1 = 0,92$, $\vartheta_1 = -0,58$ and $\sigma^2 = 0,097$. Solving the Yule–Walker system, we get

$$\gamma(k) = \begin{cases} 0,17 & \text{for } k = 0, \\ 0,1 \times 0,92^{k-1} & \text{for } k \geq 1. \end{cases}$$

Numerical results are resumed in Table 10.

Table 9. The complexity of the algorithms for $p = 200$.

	Maximum step				
Algorithm	1	3	5	10	20
A_1	40 400	121 200	202 000	404 000	808 000
A_2	40 398	42 206	44 822	54 897	90 197
A_3	40 400	41 602	42 804	45 809	51 819
A_4	40 400	41 815	44 054	53 309	87 919
A_5	60 300	101 300	142 300	244 800	449 800
Innovations	2 747 200	2 830 011	2 914 450	3 132 775	3 601 250
Levinson	120 601	242 003	363 405	666 910	1 273 920

Looking at Tables 8 and 10, we find out that the real and theoretical complexities of the algorithms are almost directly proportional. Furthermore, the computer used for testing (processor Intel Pentium 4, 1.8 GHz, operational memory 256 MB, programmed in Borland Pascal) was able to make approximately 1.5 to 2 million operations per second. Hence, if we do not have to compute thousands or more predictions, all algorithms give the desired results in real time.

Table 10. Real time complexity of the algorithms for $p = 50$ in milliseconds.

	Maximum step					
Algorithm	1	2	3	5	7	10
A_1	1,60	3,24	4,83	8,07	11,26	16,09
A_2	1,65	1,76	1,92	2,37	2,92	3,96
A_3	1,59	1,65	1,76	1,87	1,98	2,19
A_4	1,64	1,70	1,86	2,25	2,86	3,96
A_5	2,03	2,91	3,79	5,49	7,14	9,45
Innovations	23,60	24,70	26,30	29,10	32,40	37,30
Levinson	4,61	7,03	9,39	14,28	19,17	26,42
Direct method	142,30	143,40	145,60	148,30	151,10	155,50

During computations another advantage of algorithm A_3 appeared, namely the numerical stability. While the results obtained by other methods showed some inaccuracies, results given by A_3 were quite precise. It could be caused by lower number of divisions and operations overall.

(Received February 17, 2004.)

REFERENCES

[1] P. Bondon: Recursive relations for multistep prediction of a stationary time series. J. Time Ser. Anal. *22* (2001), 399–410.
[2] G. E. P. Box and G. M. Jenkins: Time Series Analysis: Forecasting and Control. Holden–Day, San Francisco 1970.
[3] P. J. Brockwell and R. Dahlhaus: Generalized Levinson–Durbin and Burg algorithms. J. Econometrics *118* (2004), 129–149.
[4] P. J. Brockwell and R. A. Davis: Time Series: Theory and Methods. Second edition. Springer, New York 1991.
[5] N. Levinson: The Wiener RMS (root mean square) error criterion in filter design and prediction. J. Math. Phys. *25* (1946), 261–278.

Pavel Ranocha, Charles University, Faculty of Mathematics and Physics, Department of Probability and Mathematical Statistics, Sokolovská 83, 186 75 Praha 8. Czech Republic.
e-mail: ranocha@karlin.mff.cuni.cz