

The Whittle-Levinson-Durbin Recursion

R. J. Kinnear

June 12, 2019

Abstract

A brief overview of the Levinson-Durbin recursion for estimating autoregressive time series models is given. Whittle's generalized (multivariate) version is also expounded upon. We are essentially summarizing content from [1], [2], [3].

It will be seen that the algorithm itself provides deep insights into the structure of autoregressive time series models, and provides an indispensable algorithm in practice.

Contents

1	introduction	1
1.1	Yule-Walker Equations	2
1.2	Estimating Covariances	2
2	The Recursions	3
2.1	The Levinson-Durbin Recursion	3
2.1.1	Properties	3
2.2	Whittle's Generalization	3
3	Efficient Model Order Selectio	4

Todo list

Double check results for σ_k^2	3
Double check results for Σ_k	4
Improve step for Σ_{k+1}	4
BIC Criteria	4
Implementation via WLD	4

1 introduction

Suppose we have a process $x(t) \in \mathbb{R}^n$ generated by the all-pole model

$$\sum_{\tau=0}^p A(\tau)x(t-\tau) = v_t, \quad (1)$$

where $B(0) = I_n$ and v_t is a temporally uncorrelated driving sequence with $\mathbb{E}[v_t] = 0$. This is closely connected to fitting $VAR(p)$ time series models

$$\hat{x}(t) = \sum_{\tau=1}^p B(\tau)x(t-\tau), \quad (2)$$

where one can estimate A and then simply drop $A(0) = I$ and take $B = -A$.

If we observe only $x(t)$, how do we determine $A(\tau)$? An answer is provided by the Levinson-Durbin recursion. This algorithm is extremely efficient as it allows one to fit a sequence of $VAR(p)$ models for every $p = 1, \dots, p_{\max}$ all for the cost of inverting a single toeplitz (or block-toeplitz) matrix. This implies that there is effectively no additional cost for performing model selection (i.e. choosing p) over and above what it costs to fit a single $VAR(p_{\max})$ model.

1.1 Yule-Walker Equations

The Levinson-Durbin recursion is essentially an efficient procedure for solving the Yule-Walker equations in one dimension, and Whittle's generalization extends to the multivariate case. The Yule-Walker equations are simply

$$\sum_{\tau=0}^p A(\tau)R(s-\tau) = \delta_s \Sigma_v; s = 0, 1, \dots, p, \quad (3)$$

which describes the relationship between the coefficients A of Equation (2) and the covariance sequence $R(\tau) = \mathbb{E}[x(t)x(t-\tau)^\top]$. They can be derived easily by taking the model (2) and multiplying it on the right by $x(t-\tau)^\top$ and computing the expectation.

There is a close relationship between the Yule-Walker equations and Toeplitz matrices: If one is to write out equation 3 in a large matrix format, the resulting system is a toeplitz (or block-toeplitz in the multivariate case) system¹

$$\mathbf{a}^\top \mathbf{R} = e_1^\top \otimes \Sigma_v, \quad (4)$$

where $\mathbf{a} = [I \ A(1) \ \dots \ A(p)]$ and \mathbf{R} is a block-toeplitz matrix consisting of blocks $[\mathbf{R}]_{s\tau} = R(s-\tau)$, and $e_1^\top \otimes \Sigma_v = [\Sigma_v \ 0 \ \dots \ 0]$. It is critical to notice that the variables in this equation are $A(1), \dots, A(p)$ and Σ_v , so it is not written in the usual " $Ax = b$ " format, but is still a linear equation. In the unidimensional case we can write

$$Ra = \sigma_v e_1, \quad (5)$$

where R is a bona-fide toeplitz matrix.

¹The symbol \otimes indicates the Kronecker product

1.2 Estimating Covariances

Given a finite sample of data $\{x(t)\}_{t=1}^T$, we can treat this as an infinitely extended sequence $\tilde{x}(t)$ where $\tilde{x}(t) = 0$ for $t \leq 0$ or $t > T$ (i.e. a rectangularly windowed sequence) and then estimate the covariance via

$$\hat{R}(\tau) = \frac{1}{T} \sum_{t=\tau+1}^p x(t)x(t-\tau)^\top. \quad (6)$$

It is critical to use this particular covariance estimator in order to ensure that $R(\tau)$ is a positive (semi-)definite sequence, that is, the Toeplitz matrix \mathbf{R} satisfies $\mathbf{R} \succeq 0$.

2 The Recursions

2.1 The Levinson-Durbin Recursion

We will first write down the Levinson-Durbin recursion, which is the unidimensional method for solving equation 3. We will consider the “API” for this Algorithm as taking as input a sequence of $p+1$ covariance estimates $(r(0), r(1), \dots, r(p))$ for a unidimensional ($n=1$) time series $x(t)$, and returning $p+1$ variance estimates $\sigma_0^2, \sigma_1^2, \dots, \sigma_p^2$, as well as a sequence $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_p$ where $\mathbf{b}_k \in \mathbb{R}^k$ provides coefficients for an $AR(k)$ model of order k , where the estimated mean squared error of this model is given by σ_k^2

$$\begin{aligned} \hat{x}(t) &= \sum_{\tau=0}^k b_k(\tau)x(t-\tau), \\ \sigma_k^2 &= \frac{1}{T} \sum_{t=1}^T (x(t) - \hat{x}(t))^2 \\ &= \mathbb{E}(x(t) - \hat{x}(t))^2 + O\left(\frac{1}{\sqrt{T}}\right). \end{aligned} \quad (7)$$

Double
check
results
for σ_k^2

The algorithm is also applicable when $x(t) \in \mathbb{C}^n$, therefore in Algorithm 1 * indicates complex conjugate. It is important to keep in mind that $r(-\tau) = r(\tau)^*$

2.1.1 Properties

The algorithm runs in $O(p^2)$ time (whereas standard matrix inversion requires $O(p^3)$ time). As well there are a number of remarkable properties associated to Algorithm 1:

1. $|b_k(k)| \leq 1$ if and only if $r(0), \dots, r(k)$ is positive semi-definite for $k = 1, \dots, p$
2. $\sigma_k^2 \geq 0$ if and only if $r(0), \dots, r(k)$ is positive semi-definite for $k = 1, \dots, p$
3. The $AR(k)$ model obtained from \mathbf{b}_k is stable

Algorithm 1: Levinson-Durbin Recursion

input : Covariance Sequence $r(0), \dots, r(p)$
output : AR coefficients $\mathbf{b}_1, \dots, \mathbf{b}_p$ and error estimates $\sigma_0^2, \dots, \sigma_p^2$
initialize: $a_0(0) = 1$
 $\sigma_0^2 = r(0)$

```

1 for  $k = 0, \dots, p - 1$  do
2    $\gamma = \sum_{\tau=0}^k a_k(\tau)r(k - \tau + 1)$ 
3    $a_{k+1}(k + 1) = -\gamma/\sigma_k^2$  # Reflection Coefficient
4    $a_{k+1}(0) = 1$ 
5   for  $\tau = 1, \dots, k$  do
6      $a_{k+1}(\tau) = a_k(\tau) + a_{k+1}(k + 1)a_k^*(k - \tau + 1)$  # Copy to next array
7    $\sigma_{k+1}^2 = \sigma_k^2(1 - |a_{k+1}(k + 1)|^2)$ 
8    $\mathbf{b}_{k+1} = (-a_{k+1}(1), \dots, -a_{k+1}(k + 1))$  # Convert to VAR Coefficients
9   assert  $\sum_{\tau=0}^{k+1} a_{k+1}(\tau)r(s - \tau) = 0$ ; for  $s = 1, \dots, k + 1$  # Verify
10 return  $(\mathbf{b}_1, \dots, \mathbf{b}_p), (\sigma_0^2, \dots, \sigma_p^2)$ 

```

2.2 Whittle's Generalization

Whittle [3] generalized Algorithm 1 to the multivariate case. This generalization is non-trivial and requires both a *forwards* set of coefficients $A(\tau)$, but also a *backwards* set of coefficients $\bar{A}(\tau)$ corresponding to the anti-causal system

$$\sum_{\tau=0}^p \bar{A}(\tau)x(t + \tau) = \bar{v}_t. \quad (8)$$

The most direct reason that the Levinson-Durbin recursion does not immediately generalize is simply because matrix multiplication is not commutative.

The algorithm will consume a sequence of covariance matrices $R(0), \dots, R(p)$, and return a sequence $\mathbf{B}_1, \dots, \mathbf{B}_p$ of $VAR(k)$ model coefficients, where $\mathbf{B}_k = (B_k(1), \dots, B_k(k))$ as well as a sequence $\Sigma_0, \dots, \Sigma_p$ of error variance matrices where

$$\begin{aligned}
\hat{x}(t) &= \sum_{\tau=1}^p B(\tau)x(t - \tau), \\
\Sigma_v &= \frac{1}{T} \sum_{t=1}^T (x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^\top \\
&= \mathbb{E}[(x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^\top] + O\left(\frac{1}{\sqrt{T}}\right).
\end{aligned} \quad (9)$$

Keeping in mind that $R(-\tau) = R(\tau)^\mathsf{H}$, we have

Double
check
results
for Σ_k

Improve
step
for
 Σ_{k+1}

Algorithm 2: Whittle-Levinson-Durbin Recursion

input : Covariance Sequence $R(0), \dots, R(p)$
output : AR coefficients $\mathbf{B}_1, \dots, \mathbf{B}_p$ and error estimates $\Sigma_0, \dots, \Sigma_p$
initialize: $A_0(0) = I, \bar{A}_0(0) = I$
 $\Sigma_0 = R(0), \bar{\Sigma}_0 = R(0)$

- 1 **for** $k = 0, \dots, p - 1$ **do**
- 2 $\Gamma = \sum_{\tau=0}^k A_k(\tau) R(k - \tau + 1)$
- 3 $\bar{\Gamma} = \sum_{\tau=0}^k \bar{A}_k(\tau) R(\tau - k - 1)$
- 4 $A_{k+1}(k+1) = -\Gamma \bar{\Sigma}_k^{-1}$ # Use 'cholesky' and 'cho_solve' to invert
- 5 $\bar{A}_{k+1}(k+1) = -\bar{\Gamma} \Sigma_k^{-1}$
- 6 $A_{k+1}(0) = I, \bar{A}_{k+1}(0) = I$
- 7 **for** $\tau = 1, \dots, k$ **do**
- 8 $A_{k+1}(\tau) = A_k(\tau) + A_{k+1}(k+1) \bar{A}_k(k - \tau + 1)$ # Copy to next array
- 9 $\bar{A}_{k+1}(\tau) = \bar{A}_k(\tau) + \bar{A}_{k+1}(k+1) A_k(k - \tau + 1)$
- 10 $\Sigma_{k+1} = \sum_{\tau=0}^{k+1} A_{k+1}(\tau) R(-\tau)$
- 11 $\bar{\Sigma}_{k+1} = \sum_{\tau=0}^{k+1} \bar{A}_{k+1}(\tau) R(\tau)$
- 12 $\mathbf{B}_{k+1} = (-A_{k+1}(1), \dots, -A_{k+1}(k+1))$ # Convert to VAR coefficients
- 13 assert $\sum_{\tau=0}^{k+1} A_{k+1}(\tau) R(s - \tau) = 0$; for $s = 1, \dots, k+1$ # Verify
- 14 assert $\sum_{\tau=0}^{k+1} \bar{A}_{k+1}(\tau) R(\tau - s) = 0$; for $s = 1, \dots, k+1$
- 15 **return** $(\mathbf{B}_1, \dots, \mathbf{B}_p), (\Sigma_0, \dots, \Sigma_p)$

3 Efficient Model Order Selectio

BIC
Crite-
ria

Implementa
via
WLD