Multistep Forecasting Non-Stationary Time Series Using Wavelets and Kernel Smoothing

Mina AMINGHAFARI
Amirkabir University of Technology, Tehran, Iran

Jean-Michel POGGI
Université Paris Sud, Orsay, France

Aminaghafari@aut.ac.ir
Outline

- Introduction
- Preliminary material
- Stationary case: procedure and simulation
- Non-stationary case: procedure and simulation
- Application to real data
Wavelets in time series

- Interesting example of using wavelets for time series: Dahlhaus *et al.* 1999 studying a time-varying AR

- A lot of methods and problems using wavelets for time series are described in Percival and Walden 2000. However forecasting is not considered.

- A representation for non-stationary process using non-decimated wavelet transform (NDWT), called Locally Stationary Wavelet (LSW) process, is given in Nason *et al.* 2000.

Wavelets for forecasting

Idea: fit prediction equation directly

- To forecast $LSW$ process, Fryzlewicz et al. 2003 use wavelet spectrum to solve generalized Yule-Walker equations with time-dependent coefficients of the past data.

- Renaud et al. 2003 estimate prediction equation by direct regression of the process on the Haar past NDW-coefficients.

- Starting from previous work, Aminghafari and Poggi 2007 propose various extensions in different directions such as using more regular wavelets or extrapolating the low frequency component of a possibly non-stationary signal.
Proposed prediction method

Same idea: fit prediction equation directly

- In Aminghafari and Poggi 2007, low frequency component is extrapolated by local polynomial fitting

Here, we propose to examine two new topics:

- use kernel smoothing for the low-frequency component extrapolation and an extension to multistep prediction

- for high frequency components given by the details, we use reconstructed versions of the detail coefficients instead of the original signal
Let us assume that the observed time series is of the form

\[ Y_t = X_t + f(t) \]

- \( X_t \) is a stationary time series
- \( f(t) \) is a deterministic component

Using sufficiently regular wavelet, wavelet transform filters automatically non-stationary components
Why wavelets for forecasting?

Let us assume that the observed time series is of the form

$$Y_t = X_t + f(t)$$

$X_t$ is a stationary time series

$f(t)$ is a deterministic component

Using sufficiently regular wavelet, wavelet transform filters automatically non-stationary components

Complex multiscale structure of signal is simplified using wavelet decomposition
Why wavelets for forecasting?

- Let us assume that the observed time series is of the form

\[ Y_t = X_t + f(t) \]

- \( X_t \) is a stationary time series
- \( f(t) \) is a deterministic component

Using sufficiently regular wavelet, wavelet transform filters automatically non-stationary components

- Complex multiscale structure of signal is simplified using wavelet decomposition
- Coefficients of a long memory signal are of short memory
Non-decimated wavelet transform

The non-decimated detail and approximation coefficients of $X = (X_0, X_1, \ldots, X_{N-1})$ are defined for levels $j \geq 1$ by:

$$w_{j,t} = \sum_{k=0}^{L_j-1} g_{j,k} X_{(t-k) \text{mod} N}$$

$$c_{j,t} = \sum_{k=0}^{L_j-1} h_{j,k} X_{(t-k) \text{mod} N}$$

Reconstruction formula:

$$X = A_J + \sum_{j=1}^{J} D_j$$

$A_J$, $D_j$: suitably reconstructed version of NDW coefficients.
Kernel smoothing

The kernel estimator of $X_{n+s}$ based on $X_n, \ldots, X_2, X_1$:

$$\hat{X}_{n+s} = \sum_{t=r,\ldots,n-s} w_{n,t} X_{t+s}$$

where weight sequence: $w_{n,t} = \frac{K((X_{n,(r)}-X_{t,(r)}/h_n)}{\sum_{m=r,\ldots,n-s} K((X_{n,(r)}-X_{m,(r)}/h_n)}$

$X_{n,(r)} = (X_n, X_{n-1}, \ldots, X_{n-r+1})$: lagged variables

$K$: kernel $h_n$: bandwidth

⇒ Apply it on locally centered versions of the signal as in Poggi 1994
Multistep prediction equation can be written

\[ \hat{X}_{N+s} = \sum_{j=1}^{J} \sum_{k=1}^{r_j} a_{j,k} w_{j,N-k+1} + \sum_{k=1}^{r_{J+1}} a_{J+1,k} c_{J,N-k+1} \]

Define

\[ D_t = \begin{bmatrix} w_{1,t} , \ldots, w_{1,t-2r_1} , \ldots, w_{J,t} , \ldots, w_{J,t-2Jr_J} , c_{J,t} , \ldots, c_{J,t-2Jr_{J+1}} \end{bmatrix}^T \]

\[ \alpha = \begin{bmatrix} a_{1,1} , \ldots, a_{1,r_1} , \ldots, a_{J,1} , \ldots, a_{J,r_J} , \ldots, a_{J+1,1} , \ldots, a_{J+1,r_{J+1}} \end{bmatrix}^T \]

Then \( \hat{X}_{N+s} = D_N^T \alpha \) where \( \alpha \) is estimated by minimizing the empirical mean square prediction error
Stationary case: the procedure

- **Step 1**: Perform NDW decomposition at level $J$ of observed time series, $(X_1, \ldots, X_N)$
Stationary case: the procedure

- **Step 1:** Perform NDW decomposition at level $J$ of observed time series, $(X_1, \ldots, X_N)$

- **Step 2:** Set prediction equation
  Select $(r_1, \ldots, r_{J+1})$, the maximum numbers of explanatory variables at each level using AR process fitting. The prediction equation can be written for past data as

$$\hat{X}_{t+s} = X_{t+s} = D_t^T \alpha$$
Stationary case: the procedure

- **Step 1**: Perform NDW decomposition at level $J$ of observed time series, $(X_1, \ldots, X_N)$

- **Step 2**: Set prediction equation
  Select $(r_1, \ldots, r_{J+1})$, the maximum numbers of explanatory variables at each level using $AR$ process fitting. The prediction equation can be written for past data as $\hat{X}_{t+s} = X_{t+s} = D_t^T \alpha$

- **Step 3**: Estimate prediction equation
  Perform estimation using *ascending stepwise regression* between $X_{t+s}$ and $D_t$ and estimate $\alpha$ by $\hat{\alpha}_s$
Stationary case: the procedure

- **Step 1**: Perform NDW decomposition at level $J$ of observed time series, \((X_1, \ldots, X_N)\)

- **Step 2**: Set prediction equation
  Select \((r_1, \ldots, r_{J+1})\), the maximum numbers of explanatory variables at each level using AR process fitting. The prediction equation can be written for past data as \( \hat{X}_{t+s} = X_{t+s} = D_t^T \alpha \)

- **Step 3**: Estimate prediction equation
  Perform estimation using *ascending stepwise regression* between \(X_{t+s}\) and \(D_t\) and estimate \(\alpha\) by \(\hat{\alpha}_s\)

- **Step 4**: Compute prediction \( \hat{X}_{N+s} = D_N^T \hat{\alpha}_s \)
Consider two following examples:

- a high order autoregressive $AR(14)$: to experiment longer short-dependence

- a generalized thresholded autoregressive $GTAR$: a highly nonlinear model
For each model

- 50 realizations of size $N = 2000$, $(x^k_1, \ldots x^k_N)_{k=1,\ldots,50}$
- Each realization of size $N = 2000$ observations is divided in two groups of size $n = 1950$ (for the past) and $N - n = 50$ (for the future).
- Compute over 50 realization

$$R(x^k, \hat{x}^k) = \sqrt{\frac{1}{N-n} \sum_{t=n+1}^{N} (\hat{x}^k_t - x^k_t)^2}$$

$\hat{x}^k_t$ is a $s$-step prediction of $x^k_t$

- evaluate mean and standard error of $R(x^k, \hat{x}^k)$
- The closer $\bar{R}$ to 1, the better the prediction
Simulation results

\( \mathcal{M}_{AR} \) and \( \mathcal{M}_{GTAR} \): 10-Step Prediction Performance

<table>
<thead>
<tr>
<th>Wavelet</th>
<th>( \bar{R}_{\text{pred}} )</th>
<th>std(( R_{\text{pred}} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{M}_{AR} )</td>
<td>db2</td>
<td>1.06</td>
</tr>
<tr>
<td>( \mathcal{M}_{GTAR} )</td>
<td>db2</td>
<td>1.09</td>
</tr>
</tbody>
</table>
Non-stationary case

- Observed time series are of the form

\[ Y_t = X_t + f(t) \]

- \( X_t \) is a purely stochastic time series
- \( f(t) \) is a deterministic component

- From reconstruction equation for \( X \)

\[ Y_t = (A_J(X))_t + \left( \sum_{j=1}^{J} D_j(X) \right)_t + f(t), \quad 1 \leq t \leq N \]

\( A_J(X) \): reconstructed version of approximation of \( X \)
\( D_j(X) \): reconstructed versions of detail of \( X \)
Step 1: $D_t = \left( \sum_{j=1}^{J} D_j(X) \right)_t$ is predicted using the previous procedure by taking $r_{J+1} = 0$ since details are supposed to be free of $f$ for a convenient wavelet choice. Perform regression between $D_t$, and the past wavelet coefficients $\{w_{j,k}\}_{k \leq t-s}$.
Non-stationary case: the procedure

Step 1: \( D_t = \left( \sum_{j=1}^{J} D_j(X) \right)_t \) is predicted using the previous procedure by taking \( r_{J+1} = 0 \) since details are supposed to be free of \( f \) for a convenient wavelet choice. Perform regression between \( D_t \), and the past wavelet coefficients \( \{ w_{j,k} \}_{k \leq t-s} \).

Step 2: \( Z_t = Y_t - \hat{D}_t \) estimate \( (A_j(X))_t + f(t) \). \( Z_{N+s} \) extrapolated using kernel smoothing procedure.
Non-stationary case: the procedure

- **Step 1**: \( D_t = (\sum_{j=1}^{J} D_j(X))_t \) is predicted using the previous procedure by taking \( r_{J+1} = 0 \) since details are supposed to be free of \( f \) for a convenient wavelet choice.
  Perform regression between \( D_t \), and the past wavelet coefficients \( \{w_{j,k}\}_{k \leq t-s} \)

- **Step 2**: \( Z_t = Y_t - \hat{D}_t \) estimate \( (A_J(X))_t + f(t) \).
  \( Z_{N+s} \) extrapolated using kernel smoothing procedure

- **Step 3**: The prediction of \( Y \) is given by:
  \[
  \hat{Y}_{N+s} = \hat{D}_{N+s} + \hat{Z}_{N+s}.
  \]
Parameter selection

- The choice of kernel function does not strongly influence the asymptotic behavior of estimator
- The Gaussian kernel is used
- The choice of bandwidth is crucial
- We choose an initial value for $h$ say $h_0$

Select $r$, the number of lagged variable used in the extrapolation, by performing some kind of hold-out procedure on the past observations.

Once $r$ is selected, we select $h$ according to a similar procedure.
Prediction methods

- Method 1: Procedure especially designed to predict stationary process without any specific adaptation
- Method 2: Method proposed in Aminghafari and poggi 2007 i.e. using polynomial fitting to extrapolate low frequency components adapted to multistep prediction
- Method 3: Our proposed method here, using kernel smoothing to extrapolate low frequency components;
- Method 4: Direct kernel smoothing without preprocessing as proposed in Poggi 1994
Simulation results

\( M_{AR} + Sin.05t \): 1-Step Prediction Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>Wavelet</th>
<th>( \bar{R}_{pred} )</th>
<th>std( R_{pred} )</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>db2</td>
<td>1.02</td>
<td>.11</td>
<td></td>
</tr>
<tr>
<td>Method 2</td>
<td>db2</td>
<td>1.03</td>
<td>.10</td>
<td>15</td>
</tr>
<tr>
<td>Method 3</td>
<td>db2</td>
<td>1.03</td>
<td>.11</td>
<td>h=3 r=10</td>
</tr>
<tr>
<td>Method 4</td>
<td></td>
<td>1.06</td>
<td>.13</td>
<td>h=3 r=10</td>
</tr>
</tbody>
</table>
Simulation results: $\mathcal{M}_{AR} + \sin(0.05t)$
$$M_{AR} + Sin.05t: \text{ 10-Step Prediction Performance}$$

<table>
<thead>
<tr>
<th>Method</th>
<th>Wavelet</th>
<th>$\bar{R}_{pred}$</th>
<th>std $R_{pred}$</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>db2</td>
<td>1.28</td>
<td>0.17</td>
<td></td>
</tr>
<tr>
<td>Method 2</td>
<td>db2</td>
<td>1.16</td>
<td>0.14</td>
<td>15</td>
</tr>
<tr>
<td>Method 3</td>
<td>db2</td>
<td>1.09</td>
<td>0.14</td>
<td>h=3 r=100</td>
</tr>
<tr>
<td>Method 4</td>
<td></td>
<td>1.09</td>
<td>0.14</td>
<td>h=3 r=100</td>
</tr>
</tbody>
</table>
Simulation results: $\mathcal{M}_{AR} + Sin.05t$
## Nile data: prediction and performance

### Nile: 1-Step Prediction Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>Wavelet</th>
<th>$R_{pred}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>db2</td>
<td>50.09</td>
</tr>
<tr>
<td>Method 2</td>
<td>db2</td>
<td>48.50</td>
</tr>
<tr>
<td>Method 3</td>
<td>db2</td>
<td>49.4</td>
</tr>
<tr>
<td>Method 4</td>
<td>-</td>
<td>52.93</td>
</tr>
</tbody>
</table>

### Nile: 10-Step Prediction Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>Wavelet</th>
<th>$R_{pred}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>db2</td>
<td>1172.7</td>
</tr>
<tr>
<td>Method 2</td>
<td>db2</td>
<td>67.03</td>
</tr>
<tr>
<td>Method 3</td>
<td>db2</td>
<td>58.1</td>
</tr>
<tr>
<td>Method 4</td>
<td></td>
<td>55.01</td>
</tr>
</tbody>
</table>
Conclusion

- Proposed method gives very often better results than other methods
- Polynomial fitting gives the good performance only for $s = 1$ or 2
- First method designed for stationary signal does not give accurate performance for non-stationary signal as expected
- Direct kernel smoothing gives comparable results with our proposed method
THANK YOU!