Unit 12: Forecasting

Taylor R. Brown

Department of Statistics, University of Virginia

Fall 2020

Readings for Unit 12

Textbook chapter 3.4.

Last Unit

1. ACF for MA(q)
2. ACF for Causal ARMA(p,q)
3. Partial Autocorrelation Function

This Unit

1. Best linear predictor
2. ARMA forecasting

Motivation

In this unit, we explore forecasting: predicting future values of a time series based on observed data.

Forecasting

In forecasting, the goal is to predict future values of a time series, x_{n+m}, based on the observed data $\mathrm{x} = \{x_n, x_{n-1}, \cdots, x_1\}$. In this unit, we assume $\{x_t\}$ is stationary.

Conditional Expectations

Conditional expectations are almost always the way you want to use data to forecast/predict something else. It's no different in this situation:

$$x_{n+m}^{n} = \mathrm{E}(x_{n+m}|\mathrm{x}_{1:n}) \tag{1}$$

is the best way to forecast m steps into the future with the data you have in the sense that that point minimizes mean square error $\mathrm{E}\left[x_{n+m} - g(\mathrm{x}_{1:n})\right]^2$, where $g(\mathrm{x})$ is any function of the observations.

Forecasting

First, we restrict our attention to predictors that are linear functions of the observations, i.e.

$$x_{n+m}^{n} = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_j \tag{2}$$

where $\alpha_0, \alpha_1, \cdots, \alpha_n \in \mathbb{R}$. Linear predictors of the form (2) that minimize the mean square prediction error are called **best linear predictors (BLP)**.

Linear prediction depends on on the second-order moments of the process, which can be estimated from the data.

Property 3.3

We find the αs in $x_{n+m}^{n} = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_j$ by taking the derivatives of

$$E\left[\left(x_{n+m} - \left(\alpha_0 + \sum_{j=1}^{n} \alpha_j x_j\right)\right)^2\right]$$

and setting them equal to 0.

Property 3.3

We find the αs in $x_{n+m}^n = \alpha_0 + \sum_{j=1}^n \alpha_j x_j$ we solve the following system of equations:

$$E\left[\left(x_{n+m} - \left(\alpha_0 + \sum_{j=1}^{n} \alpha_j x_j\right)\right)\right] = 0$$

$$E\left[\left(x_{n+m} - \left(\alpha_0 + \sum_{j=1}^{n} \alpha_j x_j\right)\right) x_1\right] = 0$$

$$\vdots$$

$$E\left[\left(x_{n+m} - \left(\alpha_0 + \sum_{j=1}^{n} \alpha_j x_j\right)\right) x_n\right] = 0$$

Property 3.3

Find $x_{n+m}^n = \alpha_0 + \sum_{j=1}^n \alpha_j x_j$ by solving:

$$E\left[\left(x_{n+m} - \left(\alpha_0 + \sum_{j=1}^{n} \alpha_j x_j\right)\right) x_k\right] = 0$$

for $k = 0, 1, \ldots, n$ ($x_0 = 1$).
A few takeaways:

- Every time we get a new data point, we have to recalculate the prediction
- Every time we get a new data point, there is another equation in a new system of equations
- Prediction errors $x_{n+m} - x_{n+m}^n$ are orthogonal/uncorrelated with the prediction variables $(1, x_1, \cdots, x_n)$.

Property 3.3

Find $x_{n+m}^n = \alpha_0 + \sum_{j=1}^n \alpha_j x_j$ by solving:

$$E\left[\left(x_{n+m} - \left(\alpha_0 + \sum_{j=1}^{n} \alpha_j x_j\right)\right) x_k\right] = 0$$

for $k = 0, 1, \ldots, n$ ($x_0 = 1$).
More takeaways:

- We want recursive formulae: get x_{n+m}^n from x_{n-1+m}^{n-1}.
- We also want our recursive formulas to have bounded memory footprints.
- Different algorithms work for different models: innovations algorithm, Durbin-Levinson, Kalman filter, particle filters, etc.
- The book also describes algorithms that assume you have an infinite past of history.

Property 3.3

After simplifying the first equation

$$E\left[\left(x_{n+m} - \left(\alpha_0 + \sum_{j=1}^{n} \alpha_j x_j\right)\right)\right] = 0$$

into $\mu = \alpha_0 + \sum_{j=1}^{n} \alpha_j \mu$, we can solve for the intercept

$$\alpha_0 = \mu \left[1 - \sum_{j=1}^{n} \alpha_j\right].$$

So the general form of the forecast is

$$x_{n+m}^{n} = \mu + \sum_{j=1}^{n} (x_j - \mu)$$

From now on, we can assume without loss of generality that $\mu = 0$.

Property 3.3

Another step, focus on one-step-ahead, and change

$$x_{n+1}^n = \alpha_n x_n + \alpha_{n-1} x_{n-1} + \cdots + \alpha_1 x_1$$

into

$$x_{n+1}^n = \phi_{n1} x_n + \phi_{n2} x_{n-1} + \cdots + \phi_{nn} x_1.$$

Writing ϕ instead of α and reversing the subscripts is purely superficial, and is meant to suggest that the upcoming work is going to be more useful for pure AR(p) models. It turns

$$E\left[\left(x_{n+1} - \left(\sum_{j=1}^{n} \alpha_j x_j\right)\right) x_k\right] = 0$$

for $k = 1, \ldots, n$ into

$$E\left[\left(x_{n+1} - \left(\sum_{j=1}^{n} \phi_{nj} x_{n+1-j}\right)\right) x_{n+1-k}\right] = 0$$

Property 3.3

For $k = 1, \ldots, n$, write

$$E\left[\left(x_{n+1} - \left(\sum_{j=1}^{n} \phi_{nj} x_{n+1-j}\right)\right) x_{n+1-k}\right] = 0$$

as

$$\gamma_x(k) - \sum_{j=1}^{n} \phi_{nj} \gamma_x(k-j) = 0$$

or ...

Property 3.3

$\gamma_x(k) - \sum_{j=1}^{n} \phi_{nj}\gamma_x(k-j) = 0$ as

$$\begin{bmatrix} \gamma(0) & \gamma(1) & \cdots & \gamma(n-1) \\ \gamma(1) & \gamma(0) & \cdots & \gamma(n-2) \\ \vdots & \vdots & \ddots & \vdots \\ \gamma(n-1) & \gamma(n-2) & \cdots & \gamma(0) \end{bmatrix} \begin{bmatrix} \phi_{n1} \\ \phi_{n2} \\ \vdots \\ \phi_{nn} \end{bmatrix} = \begin{bmatrix} \gamma(1) \\ \gamma(2) \\ \vdots \\ \gamma(n) \end{bmatrix}$$

or in shorter form

$$\Gamma_n \phi_n = \gamma_n$$

Obtaining x_{n+1}^n

For each data set size n, we solve

$$\Gamma_n \phi_n = \gamma_n$$

for ϕ_n. The bigger the data, the more rows of that matrix. Theoretically, it's just

$$\phi_n = \Gamma_n^{-1} \gamma_n$$

and

$$x_{n+1}^n = \phi_n' x$$

where $x = (x_n, x_{n-1}, \ldots, x_1)'$. Practically, though, naively inverting a gigantic matrix can be a deal breaker.

Obtaining x_{n+1}^n

Once we have the forecast from the weight vector solution: $x_{n+1}^n = \phi_n' x$, we can calculate its variance as well:

$$\begin{aligned}
E[(x_{n+1} - x_{n+1}^n)^2] &= E[(x_{n+1} - \phi_n' x)^2] \\
&= E[x_{n+1}^2] - 2E[x_{n+1}\phi_n' x] + E[\phi_n' x \phi_n' x] \\
&= \gamma_x(0) - 2E[x_{n+1}x']\phi_n + \phi_n' E[xx']\phi_n \\
&= \gamma_x(0) - 2\gamma_n'\phi_n + \phi_n'\Gamma_n\phi_n \\
&= \gamma_x(0) - 2\gamma_n'\phi_n + \phi_n'\gamma_n \\
&= \gamma_x(0) - \gamma_n'\phi_n \\
&= \gamma_x(0) - \gamma_n'\Gamma_n^{-1}\gamma_n
\end{aligned}$$

$\gamma_n'\phi_n$ is the reduction in uncertainty and depends on how much autocorrelation you have.

Example: AR(2)

If we have an AR(2) $x_{n+1} = \phi_1 x_n + \phi_2 x_{n-1} + w_t$, let's find the predictions/forecasts for $n = 2$, $n = 3$, ... and see a pattern.

At $n = 2$, forecasting time 3 requires solving

$$\Gamma_n \phi_n = \gamma_n$$

which is just two equations:

$$\begin{bmatrix} \gamma_x(0) & \gamma_x(1) \\ \gamma_x(1) & \gamma_x(0) \end{bmatrix} \begin{bmatrix} \phi_{21} \\ \phi_{22} \end{bmatrix} = \begin{bmatrix} \gamma(1) \\ \gamma(2) \end{bmatrix}$$

Example: AR(2)

In this case, it helps to write

$$\begin{bmatrix} \gamma_x(0) & \gamma_x(1) \\ \gamma_x(1) & \gamma_x(0) \end{bmatrix} \begin{bmatrix} \phi_{21} \\ \phi_{22} \end{bmatrix} = \begin{bmatrix} \gamma(1) \\ \gamma(2) \end{bmatrix}$$

as

$$E\left[\left(x_{n+1} - \left(\sum_{j=1}^{n} \phi_{nj} x_{n+1-j}\right)\right) x_{n+1-k}\right] = 0$$

for $k = 1, 2$. Clearly $\phi_{21} = \phi_1$ and $\phi_{22} = \phi_2$.

Example: AR(2)

Assume now that $n > 2$. The prediction equations are now for $k = 1, \ldots, n$

$$E\left[\left(x_{n+1} - \left(\sum_{j=1}^{n} \phi_{nj} x_{n+1-j}\right)\right) x_{n+1-k}\right] = 0$$

Clearly

$$\begin{bmatrix} \phi_{n1} \\ \phi_{n2} \\ \phi_{n3} \\ \vdots \\ \phi_{nn} \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

is the solution to these.

Example: AR(p)

Assume at time $t \geq p$ we want to predict/forecast time $t+1$ with an AR(p) model for which we know the parameters $\phi_1, \ldots, \phi_p$. We would just use the previous p time points $t, t-1, \ldots, t-p+1$ and use $\phi_1 x_t + \cdots \phi_p x_{t-p+1}$ as a prediction.

We won't derive the **Durbin-Levinson** algorithm, but that's essentially what it's getting at. It's recursive, and the calculations don't get more difficult at each time point, but it's really only useful for pure AR(p) models.

The Innovations Algorithm

The book also mentions the **Innovations Algorithm**, which is more useful for pure MA(q) models, and can be extended to ARMA(p,q) models. The idea is to write predictions in terms of

$$\hat{x}_{n+1}^n = \sum_{j=1}^{n} \theta_{nj}(x_{n+1-j} - \hat{x}_{n+1-j})$$

instead of

$$\hat{x}_{n+1}^n = \sum_{j=1}^{n} \phi_{nj} x_{n+1-j}.$$

The $x_{n+1-j} - \hat{x}_{n+1-j}$ are called the innovations, and are kind of like w_{n+1-j}.