Forecasting for ARMA Models

February 12, 2019

The material in this set of notes is based on S&S Chapter 3, specifically 3.5. We're going to start by learning how to **forecast** future values of our time series, as if we knew the true autocorrelation functions and/or **ARMA**(p, q) parameters.

General Time Series Forecasting

First, we going to talk about how we might forecast future values of a time series in general using just knowledge of the autocovariance function. Without loss of generality, we're always going to assume a mean zero time series x_t. Any stationary time series z_t with nonzero mean μ_z can be transformed to a mean zero time series by just subtracting off the mean, $x_t = z_t - \mu_z$.

To define a "good" forecast, we need to define (a) what "good" means and (b) what kind of forecasts we're interested in considering. Supposing that we observed a length-n time series $\boldsymbol{x}$, we define a "good" a forecast $\hat{x}_{n+1}$ as one that minimizes the expected mean squared error,

$$v_{n+h-1} = \mathbb{E}\left[(x_{n+1} - \hat{x}_{n+1})^2\right], \tag{1}$$

and we consider forecasts that are linear functions of the observed values $x_1, \dots x_n$,

$$\hat{x}_{n+1} = \sum_{j=1}^{n} c_{nj} x_{n-j}, \tag{2}$$

for some $\boldsymbol{c}_n$. The values of $\boldsymbol{c}_n$ implicitly depend on h, and explicitly depend on the length of the time series used to construct the forecast.

Plugging (2) into (1), we get

$$\mathbb{E}\left[\left(x_{n+1} - \left(\sum_{j=1}^{n} c_{nj} x_{n-j},\right)\right)^2\right] = \mathbb{E}\left[x_{n+1}^2\right] + \boldsymbol{c}_n' \boldsymbol{A}_n \boldsymbol{c}_n - 2\boldsymbol{c}_n' \boldsymbol{b}_n, \tag{3}$$

where $a_{n,ij} = \gamma_x(i-j)$ and $b_{n,j} = \gamma_x(j)$.

The optimal $\boldsymbol{c}_n$ satisfy:

$$\boldsymbol{A}_n \boldsymbol{c}_n = \boldsymbol{b}_n, \tag{4}$$

so we can obtain $\boldsymbol{c}_n$ by inverting $\boldsymbol{A}_n$, i.e. $\boldsymbol{c}_n = \boldsymbol{A}_n^{-1} \boldsymbol{b}_n$.

When n is large, $\boldsymbol{A}_n$ is going to be very computationally burdensome to invert, making it computationally prohibitive to solve (4) for $\boldsymbol{c}_n$. Fortunately, there are two **recursive** algorithms that allow us to compute $\hat{x}_{n+1}$ without inverting $\boldsymbol{A}_n$.

- **Durbin-Levinson:** If x_t is mean-zero and stationary and the autocovariance function $\gamma_x(h)$ satisfies $\gamma_x(0) > 0$ and $\gamma_x(h) \to 0$ as $h \to \infty$, then the coefficients c_{nj} of (2) satisfy:

  - $c_{11} = \rho_x(1)$;
  - $v_0 = \gamma_x(0)$;
  - $c_{nn} = \left(\gamma_x(n) - \sum_{j=1}^{n-1} c_{(n-1)j} \gamma_x(n-j)\right) v_{n-1}^{-1}$;
  - $c_{nj} = c_{(n-1)j} - c_{nn} c_{(n-1)(n-j)}$;
  - $v_n = v_{n-1}(1 - c_{nn}^2)$.

  We're not going to prove why the Durbin-Levinson algorithm works in this class, it's a bit of a headache. The general idea is that assuming stationarity allows us to rewrite the prediction $\hat{x}_{n+1}$ in a convenient way. If you are interested in understanding this better, see Brockwell and Davis (1991) Section 5.2.

- **Innovations:** The innovations algorithm takes advantage of the fact that we are really only interested in computing the predictions $\hat{x}_{n+1}$, not the values of $\boldsymbol{c}_n$. Instead of finding $\boldsymbol{c}_n$, the innovations algorithm produces predictions $\hat{x}_{n+1}$ that are a linear function of prediction errors, $x_j - \hat{x}_j$ for $j = 1, \dots n$:

$$\hat{x}_{n+1} = \sum_{j=1}^{n} d_{nj} \left(x_j - \hat{x}_j \right). \tag{5}$$

So, instead of computing $\boldsymbol{c}_n$ we need to compute $\boldsymbol{d}_n$.

We note that, unlike the Durbin-Levinson algorithm, the innovations algorithm can be used **even if x_t is not stationary**. However, since we will just be dealing with stationary time series, we'll describe how the innovations algorithm works in the stationary case. The innovations algorithm works as long as the covariance matrix $\boldsymbol{A}_n$ with $a_{n,ij} = \gamma_x(i-j)$ is invertible. We define $\hat{x}_1 = 0$ and $\hat{x}_{k+1} = \sum_{j=1}^{k} d_{kj} \left(x_{k+1-j} - \hat{x}_{k+1-j} \right)$ if $k \geq 1$. The innovation algorithm sets:

  - $v_0 = \gamma_x(0)$;
  - $d_{n(n-k)} = v_k^{-1} \left(\gamma_x(n-k) - \sum_{j=0}^{k-1} d_{k(k-j)} d_{n(n-j)} v_j \right)$ for $k = 0, 1, \dots, n-1$;
  - $v_n = \gamma_x(0) - \sum_{j=0}^{n-1} d_{n(n-j)}^2 v_j$.

This lets us compute v_0, then d_{11}, then v_1, then d_{22} and d_{21}, and so on. Again, remember that the innovation algorithm does *not* give the coefficients of the lagged values $x_1, \dots, x_n$ in (2), but rather the coefficients of the lagged prediction errors $x_j - \hat{x}_j$ for $j = 1, \dots, n$. Again, we're not going to prove why the innovations algorithm works in this class. The general idea is that $x_j - \hat{x}_j$ is orthogonal to $x_k - \hat{x}_k$ for $k \neq j$, and that that allows us to simplify computations. If you are interested in understanding this better, this (like the Durbin-Levinson algorithm) is also covered in detail in Brockwell and Davis (1991) Section 5.2.

ARMA (p, q) Time Series Forecasting

Now that we have some methods for forecasting time series in general using the autocovariance function $\gamma_x(h)$, we can talk about how to forecast future values of **ARMA** (p, q) time series. This is easiest to understand in the context of the simplest possible **ARMA** (p, q) models, the **ARMA** $(1, 0)$ or **AR** (1) model and the **ARMA** $(0, 1)$ or **MA** (1) model. To make things even simpler, we'll focus on forecasting x_3.

First, let's just compute the autocovariance function for the general **ARMA** $(1, 1)$ model given by:

$$x_t = \phi_1 x_{t-1} + \theta_1 w_{t-1} + w_t. \tag{6}$$

Remeber - this **ARMA** $(1, 1)$ model is mean-zero. It has variance,

$$\begin{aligned}
\gamma_x(0) &= \mathbb{E}\left[x_t^2\right] \\
&= \phi_1^2 \mathbb{E}\left[x_{t-1}^2\right] + \theta_1^2 \mathbb{E}\left[w_{t-1}^2\right] + \mathbb{E}\left[w_t^2\right] + 2\phi_1\theta_1 \mathbb{E}\left[x_{t-1}w_{t-1}\right] \\
&= \phi_1^2 \gamma_x(0) + \left(\theta_1^2 + 2\phi_1\theta_1 + 1\right)\sigma_w^2 \\
\gamma_x(0) &= \frac{\left(\theta_1^2 + 2\phi_1\theta_1 + 1\right)\sigma_w^2}{1 - \phi_1^2},
\end{aligned} \tag{7}$$

and autocovariance function

$$\begin{aligned}
\gamma_x(h) &= \mathbb{E}\left[x_t x_{t-h}\right] \\
&= \phi_1 \mathbb{E}\left[x_{t-1}x_{t-h}\right] + \theta_1 \mathbb{E}\left[w_{t-1}x_{t-h}\right] \\
&= \begin{cases} \phi_1 \gamma_x(0) + \theta_1 \sigma_w^2 & h = 1 \\ \phi_1 \gamma_x(h-1) & h > 1 \end{cases}
\end{aligned}$$

We can simplify this further to

$$\begin{aligned}\gamma_x(h) &= \sigma_w^2 \phi_1^{h-1} \left(\frac{(1+\theta_1\phi_1)(\phi_1+\theta_1)}{1-\phi_1^2} \right) \\ \rho_x(h) &= \frac{(1+\theta_1\phi_1)(\phi_1+\theta_1)\phi^{h-1}}{1+2\theta_1\phi_1+\theta_1^2},\end{aligned} \tag{8}$$

both for for $h \geq 1$.

It's a bit of a pain to use (4) to predict $\hat{x}_3$, because we need to do some matrix computations. Instead, let's see what the Durbin-Levinson and innovation algorithms give us. For $\hat{x}_3$, we need to compute:

Durbin-Levinson:

1. $c_{11} = \rho_x(1)$;
2. $v_0 = \gamma_x(0)$;
3. $v_1 = \gamma_x(0)\left(1-\rho_x(1)^2\right)$;
4. $c_{22} = \left(\rho_x(2) - \rho_x(1)^2\right) / \left(1-\rho_x(1)^2\right)$;
5. $c_{21} = \rho_x(1)(1-\rho_x(2)) / \left(1-\rho_x(1)^2\right)$;
6. $\hat{x}_3 = c_{21}x_2 + c_{22}x_1$.

Note that I've simplified things quite a bit to make things easier.

For an $\mathbf{AR}(1)$ model where $\gamma_x(0) = \sigma_w^2/(1-\phi_1^2)$ and $\rho_x(h) = \phi_1^h$, this simplifies to:

1. $c_{11} = \phi_1$;
2. $v_0 = \sigma_w^2/(1-\phi_1^2)$;
3. $v_1 = \sigma_w^2$;
4. $c_{22} = 0$;
5. $c_{21} = \phi_1$;

6. $\hat{x}_3 = \phi_1 x_2$.

For an $\mathbf{MA}\,(1)$ model where $\gamma_x\,(0) = \sigma_w^2\,(\theta_1^2 + 1)$ and $\rho_x\,(h) = \theta_1/\,(\theta_1^2 + 1)$ if $h = 1$ and $\rho_x\,(h) = 0$ if $h > 1$, this simplifies to:

1. $c_{11} = \theta_1/\,(1 + \theta_1^2)$;
2. $v_0 = \sigma_w^2\,(1 + \theta_1^2)$;
3. $v_1 = \sigma_w^2\,(1 + \theta_1^4 + \theta_1^2)\,/\,(1 + \theta_1^2)$;
4. $c_{22} = -\rho_x\,(1)^2\,/\,\left(1 - \rho_x\,(1)^2\right)$;
5. $c_{21} = \rho_x\,(1)\,/\,\left(1 - \rho_x\,(1)^2\right)$;
6. $\hat{x}_3 = \left(\rho_x\,(1)\,/\,\left(1 - \rho_x\,(1)^2\right)\right)x_2 + \left(-\rho_x\,(1)^2\,/\,\left(1 - \rho_x\,(1)^2\right)\right)x_1$.

We can see that although things simplify nicely for the $\mathbf{AR}\,(1)$ model, i.e. we end up expressing $\hat{x}_3$ as just a function of the most immediate previous value x_2. Unfortunately things do not simplify as nicely for the $\mathbf{MA}\,(1)$ model - we have to hold onto $x_2 - \hat{x}_2$ *and* $x_1 - \hat{x}_1$, even though our $\mathbf{MA}\,(1)$ model lets us write x_3 just as a function of the previous noise w_2 and some additional independent noise.

We might ask if things work out better if we compute $\boldsymbol{d}_3$ under the innovation algorithm.

Innovation:

- $v_0 = \gamma_x\,(0)$;
- $d_{11} = \rho_x\,(1)$;
- $v_1 = \gamma_x\,(0)\left(1 - \rho_x\,(1)^2\right)$.
- $d_{22} = \rho_x\,(2)$;
- $d_{21} = \rho_x\,(1)\,(1 - \rho_x\,(2))\,/\,\left(1 - \rho_x\,(1)^2\right)$;

- $\hat{x}_3 = d_{21}(x_2 - \hat{x}_2) + d_{22}(x_1 - \hat{x}_1)$.

For the **AR** (1) model,

- $v_0 = \sigma_w^2 / (1 - \phi_1^2)$;
- $d_{11} = \phi_1$;
- $v_1 = \sigma_w^2$.
- $d_{22} = \phi_1^2$;
- $d_{21} = \phi_1$;
- $\hat{x}_3 = \phi_1(x_2 - \hat{x}_2) + \phi_1^2(x_1 - \hat{x}_1)$.

For the **MA** (1) model,

- $v_0 = \sigma_w^2(1 + \theta_w^2)$;
- $d_{11} = \theta_1 / (1 + \theta_1^2)$;
- $v_1 = \sigma_w^2(1 + \theta_w^2)$;
- $d_{22} = 0$;
- $d_{21} = \rho_x(1) / \left(1 - \rho_x(1)^2\right)$;
- $\hat{x}_3 = \left(\rho_x(1) / \left(1 - \rho_x(1)^2\right)\right)(x_2 - \hat{x}_2)$

This fixes our problem for the **MA** (1) model - we only need to hold onto the most recent lag x_2 and it's prediction $\hat{x}_2$. Unfortunately, the innovations algorithm doesn't simplify nicely for the **AR** (1) model - we have to hold onto x_2 *and* x_1, even though our **AR** (1) model lets us write x_3 just as a function of x_2 and independent noise.

This leaves us a bit stuck. For general **ARMA** (p, q) processes, applying the Durbin-Levinson or innovation algorithms to x_t directly won't give us our prediction for x_{n+1} as a function of only the $\max\{p,q\}$ most recent lagged values $x_{n+1-1},\dots,x_{n+1-p}$ and the $\max\{p,q\}$ most recent forecast errors $x_{n+1-1}-\hat{x}_{n+1-1},\dots,x_{n+1-p}-\hat{x}_{n+1-p}$.

I'm not going to go into this in detail, but it turns out that if we apply the innovations algorithm to a specific transformation of x_t denoted by $f\left(x_t\right)$ (I won't give the specific transformation here for the sake of our sanity!) and then transform back to get our predictions $\hat{x}_{n+1}$, we're able to get a recursive algorithm for predicting $\hat{x}_{n+1}$ under a **ARMA** (p,q) model that only requires us to hold on to:

- The p previous values of the time series $x_n,\dots,x_{n+1-p}$;
- The q previous values of the forecast errors $x_n-\hat{x}_n,\dots,x_{n+1-q}-\hat{x}_{n+1-q}$.

I don't expect you to know what the transformation or algorithm is. Rather I just hope we can all appreciate that forecasting a future value without having to hold onto the entire past is tricky, even if we assume a model like an **ARMA** (p,q) model that gives us a convenient way of expressing an observation at any given time as a function of only the $\max\{p,q\}$ most recent values. That said, if you really want to know how this works see Section 5.3 of Brockwell and Davis (1991).