Forecasting

April 20, 2020

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 just using knowledge of the autocovariance function. Without loss of generality, we're always going to assume a mean zero time series y_t. Any stationary time series z_t with nonzero and known mean μ can be transformed to a mean zero time series by just subtracting off the mean, $y_t = z_t - \mu$.

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 $\boldsymbol{y}$, we define a "good" a forecast of observation $\hat{y}_{m+1}$ of the $m+1$-th observation y_{m+1} given all of the past values $y_1, \ldots, y_m$ to be one that minimizes the expected mean squared error,

$$v_m = \mathbb{E}\left[(y_{m+1} - \hat{y}_{m+1})^2\right], \tag{1}$$

and we consider forecasts that are linear functions of the observed values $y_1, \dots y_m$,

$$\hat{y}_{m+1} = \sum_{j=1}^{m} c_{mj} y_{m+1-j}, \tag{2}$$

for some $\boldsymbol{c}_m$. The values depend on the length of the time series used to construct the forecast.

Plugging (7) into (6), we get

$$\mathbb{E}\left[\left(y_{m+1} - \left(\sum_{j=1}^{m} c_{mj} y_{m+1-j}\right)\right)^2\right] = \mathbb{E}\left[y_{m+1}^2\right] + \boldsymbol{c}_m' \boldsymbol{A}_m \boldsymbol{c}_m - 2\boldsymbol{c}_m' \boldsymbol{b}_m, \tag{3}$$

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

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

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

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

When m is large, $\boldsymbol{A}_m$ is going to be very computationally burdensome to invert, making it computationally prohibitive to solve (4) for $\boldsymbol{c}_m$. Furthermore, we will often want to solve (4) many times for $m = 1, \dots, n$ in order to get all n one-step-ahead forecasts given an observed $n\times$ times series $\boldsymbol{y}$ observed at n time points. Fortunately, there are two **recursive** algorithms that allow us to use the coefficients $\boldsymbol{c}_m$ and expected mean squared error v_m obtained by solving (4) to obtain the next set of coefficients $\boldsymbol{c}_{m+1}$ and expected mean squared error v_{m+1} for $m = 2, \dots, n$ and to compute $\hat{y}_{m+1}$ without inverting $\boldsymbol{A}_m$.

- **Durbin-Levinson:** If y_t is mean-zero and stationary and the autocovariance function $\gamma_y(h)$ satisfies $\gamma_y(0) > 0$ and $\gamma_y(h) \to 0$ as $h \to \infty$, then the coefficients c_{mj} and expected squared error losses v_m (7) satisfy:
  - $c_{11} = \rho_y(1)$;
  - $v_0 = \gamma_y(0)$;
  - $v_1 = v_0(1 - c_{11}^2)$;

- $c_{mm} = \left(\gamma_y(m) - \sum_{j=1}^{m-1} c_{(m-1)j}\gamma_y(m-j)\right) v_{m-1}^{-1}$;
- $c_{mj} = c_{(n-1)j} - c_{mm}c_{(m-1)(m-j)}$;
- $v_m = v_{m-1}(1 - c_{mm}^2)$.

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 each one-step-ahead prediction $\hat{y}_{m+1}$ 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 $\hat{y}_{m+1}$, not the values of $\boldsymbol{c}_m$. Instead of finding $\boldsymbol{c}_m$, the innovations algorithm produces predictions $\hat{y}_{m+1}$ that are a linear function of prediction errors, $y_j - \hat{y}_j$ for $j = 1, \ldots m$:

$$\hat{y}_{m+1} = \sum_{j=1}^{m} d_{mj}\left(y_{m+1-j} - \hat{y}_{m+1-j}\right). \tag{5}$$

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

We note that, unlike the Durbin-Levinson algorithm, the innovations algorithm can be used **even if y_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 $\boldsymbol{A}_m$ with $a_{m,ij} = \gamma_y(i-j)$ is invertible. We define $\hat{y}_1 = 0$ and $\hat{y}_{k+1} = \sum_{j=1}^{k} d_{kj}\left(y_{k+1-j} - \hat{y}_{k+1-j}\right)$ if $k \geq 1$. The innovation algorithm sets:

- $v_0 = \gamma_y(0)$;
- $d_{m(m-k)} = v_k^{-1}\left(\gamma_y(m-k) - \sum_{j=0}^{k-1} d_{k(k-j)}d_{m(m-j)}v_j\right)$ for $k = 0, 1, \ldots, m-1$;
- $v_m = \gamma_y(0) - \sum_{j=0}^{m-1} d_{m(m-j)}^2 v_j$.

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 $y_1, \ldots, y_m$ in (7), but rather the coefficients of the lagged prediction errors $y_j - \hat{y}_j$ for $j = 1, \ldots, m$. Again, we're not going to prove why the innovations algorithm works in this class. The general idea is that $y_j - \hat{y}_j$ is orthogonal to $y_k - \hat{y}_k$ 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.

Predicting More than One Step Ahead

Supposing that we observed a length-n time series $\boldsymbol{y}$, we define a "good" a forecast of an observation h units of time into the future, denoted by observation $\hat{y}_m^{(h)}$, given all of the past values $y_1, \ldots, y_m$ to be one that minimizes the expected mean squared error,

$$v_m^{(h)} = \mathbb{E}\left[\left(y_{m+h} - \hat{y}_m^{(h)}\right)^2\right]. \tag{6}$$

Note that this generalizes one-step-ahead forecasting, which corresponds to $h = 1$. Again, we consider forecasts that are linear functions of the observed values $y_1, \ldots y_m$:

$$\hat{y}_m^{(h} = \sum_{j=1}^{m} c_{m,j}^{(h)} y_{m+1-j}, \tag{7}$$

for some $\boldsymbol{c}_m^{(h)}$. The values depend on the length of the time series used to construct the forecast and how many steps ahead the forecast is from the last observed time series value y_m.

Plugging (7) into (6), we get

$$\mathbb{E}\left[\left(y_m^{(h)} - \left(\sum_{j=1}^{m} c_{mj}^{(h)} y_{m+1-j}\right)\right)^2\right] = \mathbb{E}\left[\left(y_m^{(h)}\right)^2\right] + \left(\boldsymbol{c}_m^{(h)}\right)' \boldsymbol{A}_m \left(\boldsymbol{c}_m^{(h)}\right) - 2\left(\boldsymbol{c}_m^{(h)}\right)' \boldsymbol{b}_m^{(h)}, \tag{8}$$

where $a_{m,ij} = \gamma_y(i-j)$ and $b_{m,j}^{(h)} = \gamma_y(j+h-1)$.

The optimal $\boldsymbol{c}_m^{(h)}$ satisfy:

As with one-step-ahead forecasting, this can be burdensome to compute when $\boldsymbol{A}_m$ is large. Fortunately, the innovations algorithm can be used to compute h-step ahead predictions. We can write:

$$\hat{y}_{m+h} = \sum_{j=h}^{m+h-1} d_{m+h-1,j} \left(y_{m+h-j} - \hat{y}_{m+h-j} \right).$$

Note that the observed time series values that appear in the sum are $y_1, \ldots, y_m$. The innovation coefficients $d_{m+h-1,j}$ are obtained using the same algorithm we discussed before.

ARMA (p, q) Time Series Forecasting

Now that we have some methods for forecasting time series in general using the autocovariance function $\gamma_y(h)$, 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 y_3.

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

$$y_t = \phi_1 y_{t-1} + \theta_1 w_{t-1} + w_t. \tag{9}$$

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

$$\begin{aligned}
\gamma_y(0) &= \mathbb{E}\left[y_t^2\right] \\
&= \phi_1^2 \mathbb{E}\left[y_{t-1}^2\right] + \theta_1^2 \mathbb{E}\left[w_{t-1}^2\right] + \mathbb{E}\left[w_t^2\right] + 2\phi_1\theta_1 \mathbb{E}\left[y_{t-1} w_{t-1}\right] \\
&= \phi_1^2 \gamma_y(0) + \left(\theta_1^2 + 2\phi_1\theta_1 + 1\right)\sigma_w^2 \\
\gamma_y(0) &= \frac{\left(\theta_1^2 + 2\phi_1\theta_1 + 1\right)\sigma_w^2}{1 - \phi_1^2},
\end{aligned} \tag{10}$$

and autocovariance function

$$\begin{aligned}\gamma_y(h) &= \mathbb{E}[y_t y_{t-h}] \\ &= \phi_1 \mathbb{E}[y_{t-1} y_{t-h}] + \theta_1 \mathbb{E}[w_{t-1} y_{t-h}] \\ &= \begin{cases} \phi_1 \gamma_y(0) + \theta_1 \sigma_w^2 & h = 1 \\ \phi_1 \gamma_y(h-1) & h > 1 \end{cases}\end{aligned}$$

We can simplify this further to

$$\begin{aligned}\gamma_y(h) &= \sigma_w^2 \phi_1^{h-1} \left(\frac{(1+\theta_1\phi_1)(\phi_1+\theta_1)}{1-\phi_1^2} \right) \\ \rho_y(h) &= \frac{(1+\theta_1\phi_1)(\phi_1+\theta_1)\phi^{h-1}}{1+2\theta_1\phi_1+\theta_1^2},\end{aligned} \tag{11}$$

both for for $h \geq 1$.

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

Durbin-Levinson:

1. $c_{11} = \rho_y(1)$;
2. $v_0 = \gamma_y(0)$;
3. $v_1 = \gamma_y(0)\left(1 - \rho_y(1)^2\right)$;
4. $c_{22} = \left(\rho_y(2) - \rho_y(1)^2\right) / \left(1 - \rho_y(1)^2\right)$;
5. $c_{21} = \rho_y(1)\left(1 - \rho_y(2)\right) / \left(1 - \rho_y(1)^2\right)$;
6. $\hat{y}_3 = c_{21} y_2 + c_{22} y_1$.

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

For an $\mathbf{AR}(1)$ model where $\gamma_y(0) = \sigma_w^2 / (1-\phi_1^2)$ and $\rho_y(h) = \phi_1^h$, 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. $\hat{y}_3 = \phi_1 y_2$.

For an $\mathbf{MA}(1)$ model where $\gamma_y(0) = \sigma_w^2(\theta_1^2 + 1)$ and $\rho_y(h) = \theta_1 / (\theta_1^2 + 1)$ if $h = 1$ and $\rho_y(h) = 0$ 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. $c_{22} = -\rho_y(1)^2 / \left(1 - \rho_y(1)^2\right)$;

5. $c_{21} = \rho_y(1) / \left(1 - \rho_y(1)^2\right)$;

6. $\hat{y}_3 = \left(\rho_y(1) / \left(1 - \rho_y(1)^2\right)\right) y_2 + \left(-\rho_y(1)^2 / \left(1 - \rho_y(1)^2\right)\right) y_1$.

We can see that although things simplify nicely for the $\mathbf{AR}(1)$ model, i.e. we end up expressing $\hat{y}_3$ as just a function of the most immediate previous value y_2. Unfortunately things do not simplify as nicely for the $\mathbf{MA}(1)$ model - we have to hold onto $y_2 - \hat{y}_2$ *and* $y_1 - \hat{y}_1$, even though our $\mathbf{MA}(1)$ model lets us write y_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:

- $v_0 = \gamma_y(0)$;
- $d_{11} = \rho_y(1)$;
- $v_1 = \gamma_y(0)\left(1 - \rho_y(1)^2\right)$.
- $d_{22} = \rho_y(2)$;
- $d_{21} = \rho_y(1)\left(1 - \rho_y(2)\right) / \left(1 - \rho_y(1)^2\right)$;
- $\hat{y}_3 = d_{21}(y_2 - \hat{y}_2) + d_{22}(y_1 - \hat{y}_1)$.

For the $\mathbf{AR}(1)$ model,

- $v_0 = \sigma_w^2 / (1 - \phi_1^2)$;
- $d_{11} = \phi_1$;
- $v_1 = \sigma_w^2$.
- $d_{22} = \phi_1^2$;
- $d_{21} = \phi_1$;
- $\hat{y}_3 = \phi_1(y_2 - \hat{y}_2) + \phi_1^2(y_1 - \hat{y}_1)$.

For the $\mathbf{MA}(1)$ model,

- $v_0 = \sigma_w^2(1 + \theta_w^2)$;
- $d_{11} = \theta_1 / (1 + \theta_1^2)$;
- $v_1 = \sigma_w^2(1 + \theta_w^2)$;
- $d_{22} = 0$;
- $d_{21} = \rho_y(1) / \left(1 - \rho_y(1)^2\right)$;

- $\hat{y}_3 = \left(\rho_y(1) / \left(1 - \rho_y(1)^2\right)\right)(y_2 - \hat{y}_2)$

This fixes our problem for the $\mathbf{MA}(1)$ model - we only need to hold onto the most recent lag y_2 and it's prediction $\hat{y}_2$. Unfortunately, the innovations algorithm doesn't simplify nicely for the $\mathbf{AR}(1)$ model - we have to hold onto y_2 *and* y_1, even though our $\mathbf{AR}(1)$ model lets us write y_3 just as a function of y_2 and independent noise.

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

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 y_t denoted by $f(y_t)$ (I won't give the specific transformation here for the sake of our sanity!) and then transform back to get our predictions $\hat{y}_{m+1}$, we're able to get a recursive algorithm for predicting $\hat{y}_{m+1}$ under a $\mathbf{ARMA}(p, q)$ model that only requires us to hold on to:

- The p previous values of the time series $y_m, \ldots, y_{m+1-p}$;
- The q previous values of the forecast errors $y_m - \hat{y}_m, \ldots, y_{m+1-q} - \hat{y}_{m+1-q}$.

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 $\mathbf{ARMA}(p, q)$ 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).