Week 7 - Estimating ARMA Models Using Maximum Likelihood & Computing Point and Interval Forecasts

Jonathan Thong

2023-04-12

Estimating AR and MA Models Using Maximum Likelihood

Let's begin by estimating an AR(1) model

$$y_t = c + \phi y_{t-1} + \epsilon_t$$

$$\epsilon_t \sim_{i.i.d} N(0,1)$$

We will use the full log-likelihood function that we presented in the lecture slides:

$$\log L(\boldsymbol{\theta}; y_1, \dots, y_T) = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\left(\frac{\sigma^2}{1-\phi^2}\right) - \frac{\left(y_1 - \frac{c}{1-\phi}\right)^2}{\frac{2\sigma^2}{1-\phi^2}} - \frac{T-1}{2}\log(2\pi) - \frac{T-1}{2}\log(\sigma^2) - \sum_{t=2}^{T}\frac{(y_t - c - \phi y_{t-1})^2}{2\sigma^2}$$

We will estimate the model using some simulated data so that we know the true value of the parameters:

```
rm(list = ls())

T = 1000
y <- arima.sim(n = T, list(ar = 0.2), mean = 10, sd = 1)
```

Note here that the mean that we specify in the **arima.sim** function corresponds to the parameter c. Once we have our simulated data, we can proceed to specify the log-likelihood function using **function**

```
fll.ar1 <-  function(c, phi, sigma){
  -dnorm(y[1], c/(1-phi), sqrt(sigma^2/(1-phi^2)), log = TRUE)
  -sum(dnorm(y[-1], c+phi*y[1:length(y)-1], sigma, log = TRUE))
}
```

Note above that when computing the log-likelihood we are actually computing the negative of the log-likelihood. This is because the **mle** function that we will be utilising actually computes a minimum, and so we have to minimise the negative log-likelihood in order to obtain the maximum log-likelihood. Alternatively, we could condition on the first observation and use the conditional likelihood function:

$$\log L(\boldsymbol{\theta}, y_2, \dots, y_T | y_1) = -\frac{T-1}{2}\log(2\pi) - \frac{T-1}{2}\log(\sigma^2) - \sum_{t=2}^{T}\frac{(y_t - c - \phi y_{t-1})^2}{2\sigma^2}$$

```
cll.ar1 <-  function(c, phi, sigma){
  -sum(dnorm(y[-1], c+phi*y[1:length(y)-1], sigma, log = TRUE))
}
```

Once we have correctly specified our likelihood function, we use the **mle** function from the **stats4** package to estimate our parameters. The **mle** function takes as it is inputs, the negative log-likelihood function that we have specified, a set of starting values and a numerical optimization method. Note that since our sample size $T = 1000$ is large, the first observation makes a very negligible contribution to the total likelihood, so the maximising the conditional likelihood will yield the same estimates as maximising the full likelihood.

```
library(stats4)

ar.mle.fll <- mle(fll.ar1, start = list(c = 2, phi = 0.01, sigma = 1), method = "L-BFGS-B")

ar.mle.fll@details$convergence # Check for convergence (0 if converged!)
```

```
## [1] 0
```

```
ar.mle.fll
```

```
##
## Call:
## mle(minuslogl = fll.ar1, start = list(c = 2, phi = 0.01, sigma = 1),
##     method = "L-BFGS-B")
##
## Coefficients:
##          c        phi      sigma
## 10.1526055  0.1873657  1.0015438
```

```
ar.mle.cll <-  mle(cll.ar1, start = list(c = 2, phi = 0.01, sigma = 1), method = "L-BFGS-B")

ar.mle.cll@details$convergence # Check for convergence (0 if converged!)
```

```
## [1] 0
```

```
ar.mle.cll
```

```
##
## Call:
## mle(minuslogl = cll.ar1, start = list(c = 2, phi = 0.01, sigma = 1),
##     method = "L-BFGS-B")
##
## Coefficients:
##          c        phi      sigma
## 10.1526055  0.1873657  1.0015438
```

We can also compare these estimates to the ones produced by the **Arima** function:

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
Arima(y, order = c(1,0,0), include.mean = TRUE, method = "ML")
```

```
## Series: y
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1     mean
##       0.1872  12.4942
## s.e.  0.0311   0.0389
##
## sigma^2 = 1.004:  log likelihood = -1420.13
## AIC=2846.27   AICc=2846.29   BIC=2860.99
```

Note here that the estimate of the mean that is reported in the above output represents an estimate of the unconditional mean of the data and thus corresponds to $\hat{E[y_t]} = \frac{\hat{c}}{1-\hat{\phi}} = \frac{9.4087775}{1-0.2456689} = 12.473$.

Now let's proceed to estimate an MA(1) model

$$z_t = \mu + \epsilon_t + \theta\epsilon_{t-1}$$

$$\epsilon_t \sim_{i.i.d} N(0, 1)$$

Let's again simulate some data so that we know what the true parameter values are:

```
epsilon = rnorm(T, mean = 0, sd = 1)

z = numeric(T)

mu = 0.6

theta = -0.8

z[1] = mu + theta*epsilon[1]

for (i in 2:T){
  z[i] = mu + epsilon[i] + theta*epsilon[i-1]
}
```

Note that unlike the AR(1) model, the errors ϵ_t are not observed directly (remember, when dealing with real (i.e., non simulated data), we only observe z_t). We will need to set an initial condition $\epsilon_0 = 0$, and then compute them recursively:

$$\epsilon_1 = z_1 - \mu$$

$$\epsilon_t = z_t - \mu - \theta\epsilon_{t-1}$$

Thus, the conditional log-likelihood is given by:

$$\log L(\boldsymbol{\theta}; z_1, \ldots, z_T | \epsilon_0 = 0) = -\frac{T}{2}\log(2\pi) - \frac{T}{2}\log(\sigma^2) - \sum_{t=1}^{T}\frac{\epsilon_t^2}{2\sigma^2}$$

```
cll.ma1 <-  function(mu, theta, sigma){

            res = numeric(T+1)

            res[1] = z[1] - mu

              for (i in 2:T){
                res[i] = z[i] - mu - theta*res[i-1]
                }
            return(-sum(dnorm(res, 0, sigma, log = TRUE)))
            }
```

Having defined the log-likelihood we can then proceed to estimate the parameters via the **mle** function:

```
ma.mle.cll <- mle(cll.ma1, start = list(mu = 1, theta = 1, sigma = 0.6), method = "L-BFGS-B")

ma.mle.cll@details$convergence # Check for convergence (0 if converged!)
```

```
## [1] 0
```

```
ma.mle.cll
```

```
##
## Call:
## mle(minuslogl = cll.ma1, start = list(mu = 1, theta = 1, sigma = 0.6),
##     method = "L-BFGS-B")
##
## Coefficients:
##         mu      theta      sigma
##  0.6007465 -0.7994471  1.0065253
```

We can also check our results against the estimates produced by the **Arima** function:

```
Arima(z, order = c(0,0,1), include.mean = TRUE, method = "ML")
```

```
## Series: z
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##           ma1    mean
##       -0.8121  0.5999
## s.e.   0.0197  0.0060
##
## sigma^2 = 1.008:  log likelihood = -1422.37
## AIC=2850.74   AICc=2850.76   BIC=2865.46
```

Computing Point and Interval Forecasts for AR and MA Processes

Let's first work with an MA(3) model.

$$w_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \theta_3 \epsilon_{t-3}$$
$$\epsilon_t \sim_{i.i.d} N(0,1)$$

Again, we will start by simulating some data:

```
w <- arima.sim(n = T, list(ma = c(0.2, 0.4, 0.8)), mean = 0, sd = 2)
```

Then, let's compute our estimates using the **Arima** function and store the MA coefficients in an object called **theta** and the estimate of σ^2 in an object called **sigma.sq.ma3**

```
ma3.mod <- Arima(w, order = c(0,0,3), include.mean = FALSE, method = "ML")

theta <- coef(ma3.mod)

sigma.sq.ma3 <- ma3.mod$sigma2
```

The, 1,2,3 and $h > 3$ step ahead point forecasts are given by:

$$\hat{w}_{T+1|T} = E[w_{T+1}|\Omega_T] = \theta_1 \epsilon_T + \theta_2 \epsilon_{T-1} + \theta_3 \epsilon_{T-2}$$

$$\hat{w}_{T+2|T} = E[w_{T+2}|\Omega_T] = \theta_2 \epsilon_T + \theta_3 \epsilon_{T-1}$$

$$\hat{w}_{T+3|T} = E[w_{T+3}|\Omega_T] = \theta_3 \epsilon_T$$
$$\hat{w}_{T+h|T} = E[w_{T+h}|\Omega_T] = 0 \qquad h > 3$$

It follows that the 1,2,3 and $h > 3$ step ahead forecast errors will be given by:

$$w_{T+1} - \hat{w}_{T+1|T} = \epsilon_{T+1}$$

$$w_{T+2} - \hat{w}_{T+2|T} = \epsilon_{T+2} + \theta_1 \epsilon_{T+1}$$

$$w_{T+3} - \hat{w}_{T+3|T} = \epsilon_{T+3} + \theta_1 \epsilon_{T+2} + \theta_2 \epsilon_{T+1}$$

$$w_{T+h} - \hat{w}_{T+h|T} = \epsilon_{T+h} + \theta_1 \epsilon_{T+h-1} + \theta_2 \epsilon_{T+h-2} + \theta_3 \epsilon_{T+h-3} \qquad h > 3$$

From these forecast errors, we will be able to compute the corresponding forecast error variances:

$$\sigma_1^2 = \sigma^2$$

$$\sigma_2^2 = \sigma^2(1 + \theta_1^2)$$

$$\sigma_3^2 = \sigma^2(1+\theta_1^2+\theta_2^2)$$

$$\sigma_h^2 = \sigma^2(1+\theta_1^2+\theta_2^2+\theta_3^2) \qquad h > 3$$

Then, the j step ahead $(1-\alpha\%)$ prediction intervals will be given by

$$\hat{w}_{T+j|T} \pm z_{\frac{\alpha}{2}}\sqrt{\sigma_j^2}$$

Using these formulas, let's compute point and interval forecasts for our MA(3) model for $h = 5$ steps ahead. First, we specify the vectors into which we will store the point forecasts and the forecast error variances:

```
h.ma3 = 5

j.ma3 <- seq(1:h.ma3)

what <- numeric(h.ma3)

sigmah.ma3 <- numeric(h.ma3)
```

Then, our point forecasts are computed as:

```
epsilon.ma3 <- ma3.mod$residuals

what[1] <- theta[1]*epsilon.ma3[T] + theta[2]*epsilon.ma3[T-1] + theta[3]*epsilon.ma3[T-2]
what[2] <- theta[2]*epsilon.ma3[T] + theta[3]*epsilon.ma3[T-1]
what[3] <- theta[3]*epsilon.ma3[T]
what[4:h.ma3] <- 0
```

Similarly, our forecast error variances are computed as:

```
sigmah.ma3[1] <- ma3.mod$sigma2
sigmah.ma3[2] <- ma3.mod$sigma2*(1+theta[1]^2)
sigmah.ma3[3] <- ma3.mod$sigma2*(1+theta[1]^2 + theta[2]^2)
sigmah.ma3[4:h.ma3] <- ma3.mod$sigma2*(1+theta[1]^2 + theta[2]^2 + theta[3]^2)
```

Then, we can compute our prediction intervals for $\alpha = 0.2$:

```
alpha = 0.20

lwrh.ma3 <- what - qnorm(alpha/2, lower.tail = FALSE)*sqrt(sigmah.ma3)
uprh.ma3 <- what + qnorm(alpha/2, lower.tail = FALSE)*sqrt(sigmah.ma3)

ma3.for <- data.frame(j.ma3,what, lwrh.ma3, uprh.ma3)
colnames(ma3.for) = c('hstep','what', 'lower', 'upper')

ma3.for
```

```
##   hstep       what     lower     upper
## 1     1 -0.2384707 -2.787266 2.3103243
## 2     2  1.2905533 -1.302326 3.8834321
## 3     3 -2.3300455 -5.139125 0.4790344
## 4     4  0.0000000 -3.457191 3.4571910
## 5     5  0.0000000 -3.457191 3.4571910
```

We can compare these values to the ones automatically computed by the **forecast** function to verify that we have indeed computed everything correctly:

```
forecast(ma3.mod, h = 5)
```

```
##      Point Forecast     Lo 80     Hi 80     Lo 95    Hi 95
## 1001     -0.2384707 -2.787266 2.3103243 -4.136516 3.659575
## 1002      1.2905533 -1.302326 3.8834321 -2.674913 5.256019
## 1003     -2.3300455 -5.139125 0.4790344 -6.626162 1.966071
## 1004      0.0000000 -3.457191 3.4571910 -5.287317 5.287317
## 1005      0.0000000 -3.457191 3.4571910 -5.287317 5.287317
```

Now let's compute point and interval forecasts for the AR(1) process that we estimated in the previous section. Using similar derivations to the ones covered in the lecture, we have that

$$\hat{y}_{T+1|T} = E[y_{T+1}|\Omega_T] = c + \phi y_T$$

$$\hat{y}_{T+2|T} = E[y_{T+2}|\Omega_T] = c + \phi E[y_{T+1}|\Omega_t] = c + \phi c + \phi^2 y_T$$
$$\hat{y}_{T+h|T} = E[y_{T+h}|\Omega_T] = c + \phi c + \phi^2 c + \cdots + \phi^{h-1} c + \phi^h y_T$$

Then, using recursive substitution, the forecast errors will be given by:

$$y_{T+1} - E[y_{T+1}|\Omega_T] = c + \phi y_T + \epsilon_{T+1} - c - \phi y_T = \epsilon_{T+1}$$
$$y_{T+2} - E[y_{T+2}|\Omega_T] = c + \phi y_{T+1} + \epsilon_{T+2} - c - \phi c - \phi^2 y_T = c + \phi c + \phi^2 y_T + \phi \epsilon_{T+1} + \epsilon_{T+2} - c - \phi c - \phi^2 y_T = \epsilon_{T+2} + \phi \epsilon_{T+1}$$
$$y_{T+h} - E[y_{T+h}|\Omega_T] = \epsilon_{T+h} + \phi \epsilon_{T+h-1} + \phi^2 \epsilon_{T+h-2} + \cdots + \phi^{h-1} \epsilon_{T+1}$$

It follows that the corresponding forecast error variances will be given by:

$$\sigma_1^2 = \sigma^2$$

$$\sigma_2^2 = \sigma^2(1 + \phi^2)$$

$$\sigma_h^2 = \sigma^2(1 + \phi^2 + \phi^4 + \cdots + \phi^{2h-2})$$

Then, the j step ahead $(1 - \alpha\%)$ prediction intervals will be given by

$$\hat{y}_{T+j|T} \pm z_{\frac{\alpha}{2}} \sqrt{\sigma_j^2}$$

Let's now compute point and interval forecasts for our AR(1) model for $h = 10$ steps ahead. First, we specify the vectors into which we will store the point forecasts and the forecast error variances:

```
h.ar1 = 10

j.ar1 = seq(1:h.ar1)

yhat <- numeric(h.ar1)

sigmah.ar1 <- numeric(h.ar1)
```

We can obtain our estimated parameter values using the **Arima** function:

```
ar1.mod <- Arima(y, order = c(1,0,0), include.mean = TRUE, method = "ML")

epsilon.ar1 <- ar1.mod$residuals

phi = ar1.mod$coef[1]

c = (1 - phi)*ar1.mod$coef[2]
```

Since the point forecasts and forecast error variances for an AR(1) have a recursive structure, we can use a loop to compute them:

```
yhat[1] = c + phi*y[T]

sigmah.ar1[1] = ar1.mod$sigma2

  for(i in 2:h.ar1){
    yhat[i] = c + phi*yhat[(i-1)]

    sigmah.ar1[i] = sigmah.ar1[i-1] + ar1.mod$sigma2*phi^(2*i -2)
  }
```

Then, we can compute our prediction intervals for $\alpha = 0.2$:

```
alpha = 0.20

lwrh.ar1 <- yhat - qnorm(alpha/2, lower.tail = FALSE)*sqrt(sigmah.ar1)
uprh.ar1 <- yhat + qnorm(alpha/2, lower.tail = FALSE)*sqrt(sigmah.ar1)

ar1.for <- data.frame(j.ar1,yhat, lwrh.ar1, uprh.ar1)
colnames(ar1.for) = c('hstep','yhat', 'lower', 'upper')

ar1.for
```

```
##    hstep     yhat    lower    upper
## 1      1 12.79416 11.50982 14.07851
## 2      2 12.55038 11.24372 13.85705
## 3      3 12.50474 11.19730 13.81218
## 4      4 12.49619 11.18873 13.80366
## 5      5 12.49459 11.18713 13.80206
## 6      6 12.49429 11.18683 13.80176
## 7      7 12.49424 11.18677 13.80171
## 8      8 12.49423 11.18676 13.80170
## 9      9 12.49423 11.18676 13.80169
## 10    10 12.49423 11.18676 13.80169
```

Again, we compare these values to the ones automatically computed by the **forecast** function to verify that we have indeed computed everything correctly:

```
forecast(ar1.mod, h = 10)
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 1001       12.79416 11.50982 14.07851 10.82993 14.75840
## 1002       12.55038 11.24372 13.85705 10.55201 14.54875
## 1003       12.50474 11.19730 13.81218 10.50518 14.50430
## 1004       12.49619 11.18873 13.80366 10.49660 14.49579
## 1005       12.49459 11.18713 13.80206 10.49500 14.49419
## 1006       12.49429 11.18683 13.80176 10.49470 14.49389
## 1007       12.49424 11.18677 13.80171 10.49464 14.49384
## 1008       12.49423 11.18676 13.80170 10.49463 14.49383
## 1009       12.49423 11.18676 13.80169 10.49463 14.49382
## 1010       12.49423 11.18676 13.80169 10.49463 14.49382
```