Time Series Analysis
-Home Exam

Nils Romanus

Jan 2023

Problem 1

The optimal predictor is designed by choosing the a and b that minimizes the mean-squared-error (MSE) objective function J defined in equation (1)

$$J := E[(\hat{x}_{t+k|t} - x_{t+k})^2] \tag{1}$$

By using the predictor $\hat{x}_{t+k|t} = ax_t + b$ the objective in (1) can be rewritten as (2).

$$J(a,b) = E[(ax_t + b - x_{t+k})^2] \tag{2}$$

expanding the square in (2) yields the following

$$J(a,b) = E[(ax_t + b)^2 - 2(ax_t + b)x_{t+k} + x_{t+k}^2] =$$
$$J(a,b) = E[a^2x_t^2 + 2abx_t + b^2 + 2ax_tx_{t+k} - 2bx_{t+k} + x_{t+k}^2]$$

Using the linearity of the expectation one finally arrives at expression (3).

$$J(a,b) = a^2E[x_t^2] + 2abE[x_t] + b^2 - 2aE[x_tx_{t+k}] - 2bE[x_{t+k}] + E[x_{t+k}^2] \tag{3}$$

In order to solve problem (4) one may differentiate w.r.t the parameters and set the derivatives to zero as shown in expression (5)

$$\underset{a,b}{\text{minimize}}\, J(a,b) \tag{4}$$

$$\begin{cases} \frac{\partial E}{\partial a} = 0 \\ \frac{\partial E}{\partial a} = 0 \end{cases} \iff \begin{cases} 0 = 2aE[x_t^2] + 2bE[x_t] - 2E[x_tx_{t+k}] \\ 0 = 2aE[x_t^2] + 2b - 2E[x_{t+k}] \end{cases} \tag{5}$$

The system in (5) is solvable by for example multiplying the second row in with $E[x_t]$ and subtracting it from the first row yielding the following:

$$0 = aE[x_t^2] - E[x_tx_{t+k}] - aE[x_t]^2 + E[x_t]E[x_{t+k}]$$

Solving for a yields the desired result as shown below, where the definition of the auto-correlation-function (ACF) is used in the last step.

$$a = \frac{E[x_t]E[x_{t+k}] - E[x_tx_{t+k}]}{E[x_t]^2 - E[x_t^2]} = \rho_x(k)$$

Using this a in the second row of (5) yields the following expression for b, where the fact that x's mean is constant stems from x being stationary.

$$b = E[x_{t+k}] - aE[x_t] = m_x - \rho_x(k)m_x$$

I.e by minimizing the MSE one arrives at the parameters shown in expression (6)

$$\begin{cases} a = \rho_x(k) \\ b = m_x - \rho_x(k)m_x \end{cases} \tag{6}$$

Problem 2

A process $\{y_t\}$ is weakly stationary if the conditions in (7) hold.

$$\begin{cases} E[y_t] = m_y(t) = m_0 \\ C[y_t, y_s] = r_y(t,s) = r_x(\tau) \\ E[|y_t|^2] < \infty \end{cases} \tag{7}$$

Where τ denotes the lag between the two time points t and s. The process y_t is bounded with constant (zero) mean for all values of the parameter c as shown in (8). I.e the first and last stationarity conditions hold for any c.

$$E[y_t] = \sin(ct)E[x_t] + \cos(ct)E[x_{t-1}] = 0 \tag{8}$$

The covariance function of y_t can be expressed by using the covariance definition as shown below.

$$r_y(s,t) = C[y_t, y_s] = C[\sin(ct)x_t + \cos(ct)x_{t-1}, \sin(cs)x_s + \cos(cs)x_{s-1}] \tag{9}$$

$$\begin{aligned} r_y(s,t) = \sin(ct)\cos(cs)C[x_t, x_s] + \sin(ct)\cos(cs)C[x_{t-1}, x_s] \\ + \cos(ct)\sin(cs)C[x_t, x_{s-1}] + \cos(ct)\cos(cs)C[x_{t-1}, x_{s-1}] \end{aligned} \tag{10}$$

The stationarity conditions imply that the covariance between time points t and s must only depend on the absolute difference between t and s, i.e $r_y(s,t) = r_y(|t-s|) := r_y(\tau)$. Since x_t are independent the covariance function evaluated at $t = s$ becomes the following.

$$r_y(t,t) = \sin^2(ct)V[x_t] + \cos^2(ct)V[x_t] = \sigma_x^2$$

Which is independent on both t and s and hence the stationarity condition holds for all values of c if $t = s$. By noting the fact that x_t are independant and the lags in equation (10) all terms will evaluate to zero if the absolute difference between t and s is greater than one. Yielding the expression that is only dependant on the difference between t and s below.

$$r_y(t,s) = 0 \quad if |t-s| > 1$$

This holds for all values of c and therefore gives us no constraint condition. Finally we examine the case where $|t-s| = 1$. If $t = s-1$ all terms terms in (10) will, as a consequence of x_t being independent, evaluate to zero except one. The same goes for $t = s+1$ and we are left with the system of equations in (11)

$$\begin{cases} r_y(t, t+1) = r_y(1) = \cos(ct)\sin(c(t+1))\sigma_x^2 \\ r_y(t, t-1) = r_y(-1) = \sin(ct)\cos(c(t-1))\sigma_x^2 \end{cases} \tag{11}$$

The covariance function of a stationary process must be even and we can therefore equate the two rows in (11) by using the symmetry condition $r_y(1) = r_y(-1)$ yielding the following expression.

$$\cos(ct)\sin(c(t+1))\sigma_x^2 = \sin(ct)\cos(c(t+1))\sigma_x^2 \tag{12}$$

Solving (12) for c yields the constraint on c, for y_t to be a stationary process, displayed in (13), where $\mathbb{Z}$ is the set of integers.

$$c = \pi n \quad n \in \mathbb{Z} \tag{13}$$

Problem 3

a)

The one-step predictor can be derived by using the definition of the process defined in (14) resulting in expression (15).

$$x_t + a_1 x_{t-1} + a_2 x_{t-2} = e_t \tag{14}$$

$$x_{t+1} = e_{t+1} - a_1 x_t - a_2 x_{t-1} \tag{15}$$

The best estimate of the noise at $t+1$ is its mean which is zero, resulting in expression (16).

$$\hat{x}_{t+1|t} = -a_1 x_t - a_2 x_{t-1} \tag{16}$$

The same goes for the two-step prediction as shown below.

$$x_{t+2} = e_{t+2} - a_1 x_{t+1} - a_2 x_t \tag{17}$$

Once again we may replace the unknown noise with its mean. Equation (17) contains an x_{t+1} term which is also unknown. The best estimate of this term is given by our one-step predictor outlined in equation (16). Using these estimates results in expression (18) for the 2 step prediction.

$$\hat{x}_{t+2|t} = -a_1 \hat{x}_{t+1} - a_2 x_t = -a_1(-a_1 x_t - a_2 x_{t-1}) - a_2 x_t = (a_1^2 - a_2)x_t + a_1 a_2 x_{t-1} \tag{18}$$

The prediction error variances are computed using by comparing the predictors to the actual value at time step $t+k$. The prediction error variance for the one-step prediction $V[\hat{\epsilon}_{t+1}]$ hence becomes the following.

$$\begin{aligned} V[\hat{\epsilon}_{t+1}] &= V[\hat{x}_{t+1|t} - x_{t+1}] \\ &= V[-a_1 x_t - a_2 x_{t-1} - (e_{t+1} - a_1 x_t - a_2 x_{t-1})] \\ &= V[-e_{t+1}] \\ &= \sigma_e^2 \end{aligned} \tag{19}$$

The prediction error variance for the two-step prediction $V[\hat{\epsilon}_{t+2}]$ is computed using the same approach as shown below.

$$\begin{aligned} V[\hat{\epsilon}_{t+2}] &= V[\hat{x}_{t+2|t} - x_{t+2}] \\ &= V[-a_1 \hat{x}_{t+1} - a_2 x_t - (e_{t+2} - a_1 x_{t+1} - a_2 x_t)] \\ &= V[-e_{t+2} - a_1(\hat{x}_{t+1} - x_{t+1})] \\ &= V[e_{t+2}] + a_1^2 V[\hat{x}_{t+1} - x_{t+1}] \\ &= \sigma_e^2 + a_1^2 V[\hat{\epsilon}_{t+1}] \\ &= (1 + a_1^2)\sigma_e^2 \end{aligned} \tag{20}$$

In summary, we get the following predictors and prediction error variances.

$$\begin{cases} \hat{x}_{t+1|t} = -a_1 x_t - a_2 x_{t-1} \quad ; \quad V[\hat{\epsilon}_{t+1}] = \sigma_e^2 \\ \hat{x}_{t+2|t} = (a_1^2 - a_2)x_t + a_1 a_2 x_{t-1} \quad ; \quad V[\hat{\epsilon}_{t+2}] = (1 + a_1^2)\sigma_e^2 \end{cases} \tag{21}$$

b)

The linear extrapolation predictor is derived by computing the slope between x_t and x_{t-1} and using it to get from x_t to x_{t+k} k time steps further. The linear extrapolation predictor is displayed in equation (22).

$$\hat{x}_{t+k|t} = x_t + k(x_t - x_{t-1}) \tag{22}$$

The one- and two-step predictors hence become the following.

$$\begin{cases} \hat{x}_{t+1|t} = x_t + 1(x_t - x_{t-1}) = 2x_t - x_{t-1} \\ \hat{x}_{t+2|t} = x_t + 2(x_t - x_{t-1}) = 3x_t - 2x_{t-1} \end{cases} \tag{23}$$

The prediction error variances can be computed using the same methodology as in section a), resulting in the k-step prediction error variance expression displayed in (24).

$$\begin{aligned} V[\hat{\epsilon}_{t+k}] &= V[\hat{x}_{t+k|t} - x_{t+k}] \\ &= V[(1+k)x_t - kx_{t-1} - x_{t+k}] \\ &= C[(1+k)x_t - kx_{t-1} - x_{t+k}, (1+k)x_t - kx_{t-1} - x_{t+k}] \\ &= ((1+k)^2 + k^2 + 1)r_x(0) - 2(1+k)kr_x(1) - 2(1+k)r_x(k) + 2kr_x(k-1) \end{aligned} \tag{24}$$

Evaluating (24) at $k = 1$ or $k = 2$ yields expressions 25 for the one- and two-step prediction error variances.

$$\begin{cases} V[\hat{\epsilon}_{t+1}] = 8(r_x(0) - r_x(1)) \\ V[\hat{\epsilon}_{t+2}] = 14r_x(0) - 8r_x(1) - 6r_x(2) \end{cases} \tag{25}$$

The covariance function $r_x(k)$ can be derived using the Yule-Walker equations as shown below.

$$r_x(k) + a_1 r_x(k-1) + a_2 r_x(k-2) = \begin{cases} \sigma_e^2 \quad ; \quad k = 0 \\ 0 \quad ; \quad k \neq 0 \end{cases} \tag{26}$$

$$\begin{bmatrix} 1 & a_1 & a_2 \\ a_1 & (1+a_2) & 0 \\ a_2 & a_1 & 1 \end{bmatrix} \begin{bmatrix} r_x(0) \\ r_x(1) \\ r_x(2) \end{bmatrix} = \begin{bmatrix} \sigma_e^2 \\ 0 \\ 0 \end{bmatrix} \tag{27}$$

Using (26) one may express the covariances at lags 1 and 2 in terms of $r_x(0)$ yielding the expressions (28).

$$\begin{cases} r_x(1) = \frac{-a_1}{1+a_2} r_x(0) \\ r_x(2) = \frac{a_1^2 - a_1(1+a_2)}{1+a_2} r_x(0) \end{cases} \tag{28}$$

Solving the matrix-vector equation in (27) yields expression (29) for the covariance function at lag 0, i.e the variance of the process.

$$r_x(0) = \frac{(1-a_2)\sigma_e^2}{(1+a_2)(1-a_1-a_2)(1+a_1-a_2)} \tag{29}$$

Using (28) in (25) yields the one- and two-step prediction error variances expressed in $r_x(0)$ as shown in (30).

$$\begin{cases} V[\hat{\epsilon}_{t+1}] = 8\frac{1+a_1+a_2}{(1+a_2)} r_x(0) \\ V[\hat{\epsilon}_{t+2}] = (14 + \frac{8a_1}{(1+a_2)} + 6\frac{a_1(1+a_2)-a_1^2}{1+a_2}) r_x(0) \end{cases} \tag{30}$$

Using (29) in (30) yields the final expression (31) for the prediction error variances.

$$\begin{cases} V[\hat{\epsilon}_{t+1}] = 8\sigma_e^2 \frac{(1-a_2)(1+a_1+a_2)}{(1+a_2)^2(1-a_1-a_2)(1+a_1-a_2)} \\ V[\hat{\epsilon}_{t+2}] = \sigma_e^2 \frac{14(1+a_1+a_2)+6a_1(a_2-a_1)}{(1+a_2)} \frac{1-a_2}{(1+a_2)(1-a_1-a_2)(1+a_1-a_2)} \end{cases} \tag{31}$$

c)

The optimal predictor can be constructed by solving the minimization problem (32).

$$\begin{gathered} J_k(a_1, a_2) := V[\hat{\epsilon}_{t+k}](a_1, a_2) \\ \underset{a_1, a_2}{\text{minimize}}\, J(a_1, a_2) \end{gathered} \tag{32}$$

One may solve (32) using any standard optimization method. However, for the one- and two-step predictor it is easier to simply look at the expressions for the objective function if $k = 1$ or $k = 2$. As shown in expression (31) both $J_1(a_1, a_2)$ and $J_2(a_1, a_2)$ include the factor $(1 - a_2)$ in the expression. I.e both functions have a zero at $a_2 = 1$. As long as J does not have a pole that cancels this zero the value of the objective will be zero. The objective will not have a pole in $a_2 = 1$ as long as any of the terms in the denominators in (31) does not evaluate to zero. This is true for all non-zero a_1. I.e

$$\begin{cases} (a_1, a_2) = (\gamma, 1) \Rightarrow J_1(a_1, a_2) = 0 \quad ; \quad \gamma \neq 0 \\ (a_1, a_2) = (\gamma, 1) \Rightarrow J_2(a_1, a_2) = 0 \quad ; \quad \gamma \neq 0 \end{cases} \tag{33}$$

Where γ is any non-zero value. Since $J_k(a_1, a_2)$ is a variance it is minorized by zero (variances can't be negative). Hence $J(a_1, a_2) = 0$ is the optimal value in the minimization problem (32), i.e

$$\begin{cases} 0 \leq J_1(a_1, a_2) \quad , \quad J_1(\gamma, 1) = 0 \iff (a_1^*, a_2^*) = (\gamma, 1) \\ 0 \leq J_2(a_1, a_2) \quad , \quad J_2(\gamma, 1) = 0 \iff (a_1^*, a_2^*) = (\gamma, 1) \end{cases}$$

Where γ is a non-zero value and (a_1^*, a_2^*) is the solution to (32)

Problem 4

A recursive algorithm for minimizing the Huber criteria can be derived by recursively minimizing the quadratic approximation of the Huber loss shown in (34).

$$J(t,\theta) := \tilde{f}(t,\theta) = f(t,\hat{\theta}_{t-1}) + (\theta - \hat{\theta}_{t-1})^T f'(t,\hat{\theta}_{t-1}) + \frac{1}{2}(\theta - \hat{\theta}_{t-1})^T f''(t,\hat{\theta}_{t-1})(\theta - \hat{\theta}_{t-1}) \tag{34}$$

Where $\tilde{f}(t,\theta)$ denotes the quadratic approximation given in the assignement instructions. The optimization problem in each step hence becomes (35)

$$\underset{\theta}{\text{minimize}}\, J(t,\theta) \tag{35}$$

Problem (35) can be solved by equating the derivative of the objective to zero as shown in (36), which turns out to be the expression for the Newton optimization method.

$$\frac{\partial J}{\partial \theta} = f'(t,\hat{\theta}_{t-1}) + (\theta - \hat{\theta}_{t-1}) f''(t,\hat{\theta}_{t-1}) = 0 \iff \theta^* = \hat{\theta}_{t-1} - [f''(t,\hat{\theta}_{t-1})]^{-1} f'(t,\hat{\theta}_{t-1}) \tag{36}$$

Where θ^* denotes the solution to (35). We can express the update more explicitly by computing the gradient and hessian of f defined in (37).

$$f(t,\theta) = \sum_{\ell=1} \rho\left(\sigma_e^{-1}\left(y_\ell - \mathbf{x}_\ell^T\theta\right)\right)\sigma_e^2 \tag{37}$$

where ρ is given by the Huber function shown in (38).

$$\rho(u) = \begin{cases} \frac{u^2}{2} & \text{if } |u| \le c \\ c|u| - \frac{c^2}{2} & \text{if } |u| > c \end{cases} \tag{38}$$

The first derivative is given by expression 39

$$\begin{aligned} \frac{\partial f(t,\theta)}{\partial \theta} &= \sum_{l=1} \frac{\partial \rho\left(\sigma_e^{-1}\left(y_\ell - \mathbf{x}_\ell^T\theta\right)\right)}{\partial \theta}\sigma_e^2 \\ &= \sum_{l=1} \sigma_e^2 \rho'\left(\sigma_e^{-1}\left(y_\ell - \mathbf{x}_\ell^T\theta\right)\right)\frac{\partial\left(\sigma_e^{-1}\left(y_\ell - \mathbf{x}_\ell^T\theta\right)\right)}{\partial\theta} \\ &= \sum_{l=1} \sigma_e^2 \rho'\left(\sigma_e^{-1}\left(y_\ell - \mathbf{x}_\ell^T\theta\right)\right)\left(-\sigma_e^{-1}\mathbf{x}_\ell^T\right) \\ &= \sum_{l=1} -\sigma_e \rho'\left(\sigma_e^{-1}\left(y_\ell - \mathbf{x}_\ell^T\theta\right)\right)\mathbf{x}_\ell \\ &= \sum_{\ell=1} -\sigma_e \rho'\left(\sigma_e^{-1}\left(y_\ell - \mathbf{x}_\ell^T\theta\right)\right)\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \end{aligned} \tag{39}$$

Where $\rho'(.)$ is given by expression (40).

$$\rho'(u) = \begin{cases} u & \text{if } |u| \leq c \\ -c & \text{if } u < -c \\ c & \text{if } u > c \end{cases} \tag{40}$$

The hessian is computed by following the same steps as in (39) resulting in expression (41).

$$\frac{\partial^2 f(t,\theta)}{\partial \theta^2} = \sum_{\ell=1} \rho''\left(\sigma_e^{-1}\left(y_\ell - \mathbf{x}_\ell^T \theta\right)\right) \begin{bmatrix} x_1 x_1 & \dots & x_1 x_n \\ \vdots & \ddots & \vdots \\ x_n x_1 & \dots & x_n x_n \end{bmatrix} \tag{41}$$

Where $\rho''(.)$ is given by expression (42).

$$\rho''(u) = \begin{cases} 1 & \text{if } |u| \leq c \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

In summary. The algorithm becomes the following. Initialize θ_0 as any parameter vector. Compute the gradient $f'(t,\theta_0)$ and the inverse of the hessian $[f''(t,\theta_0)]^-1$ according to (39) and (41) respectively, using (40) and (42). Use these in the update scheme (36) to get an estimate of the next parameter vector $\hat{\theta}_1$. Repeat for every time step in the sequence. This algorithm is more "robust" than the standard RLS algorithm as it is less sensitive to outliers. In the standard RLS algorithm, the mean-squared-error loss function is used for parameter estimation. As suggested by the name, the loss grows quadratically with respect to the size of the error terms. An outlier that creates a large error term will thus give a large contribution to the weight update. The Huber loss combines MSE and mean-absolute-error(MAE), it is quadratic for small errors and linear for large errors. This means that the loss function, for large errors, that are likely to stem from outliers, will not grow quadratically as a function of the size of these errors. Hence these larger errors will contribute less to the weight update. Thus the Huber loss is more robust to outliers than the MSE-based RLS.