Ural Federal University
named after the first President of Russia B.N.Yeltsin
Institute of radioelectronics and information technologies

THE ARIMA MODELS OF TIME SERIES

Laboratory Task no. 4

Yekaterinburg
2019

Ural Federal University
Institute of radioelectronics and information technologies

Contents

1. Introduction

The general model for time series of ARIMA = AutoRegressive Integrated Moving Average of (p, d, q) order is theoretically formulated as follows:

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + \ldots + \phi_p \tilde{z}_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q}.$$

This model is really useful on practice for the creation of the forecast for any time series data. Hence, the ARIMA technique is always available in every possible framework for time series data analysis and forecast.

2. Laboratory Task

The final result of laboratory task is presented as the report in form of *Jupyter*-notebook (or something similar), with all the needed commands for a complete Run of cells. Now, let's begin with time series data analysis:

1) Import into your code the following libraries:

```
import numpy as np
import numpy.random as rand
import matplotlib.pyplot as plt
import h5py
from statsmodels.tsa import api as tsa
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.arima_model import ARIMA
%matplotlib inline
```

2) First of all, let's start with the creation of our own simulated time series for different ARIMA models, with the estimation of their characteristics.

3) Let's create two AR(1) processes = Autoregressive data of first order:

$$z_t = 0.8z_{t-1} + a_t \quad \text{and} \quad z_t = -0.8z_{t-1} + a_t$$

where a_t – normal distributed random value with small amplitude (somewhere around 0.2), $z_0 = 1$.

```
z1 = np.zeros(100)
z2 = np.zeros(100)
z1[0] = 1
z2[0] = 1
for i in range(1,100):
    z1[i] = 0.8 * z1[i - 1] + 0.2 * np.random.randn()  # positive AR
    z2[i] = -0.8 * z2[i - 1] + 0.2 * np.random.randn() # negative AR
plt.figure(figsize = (10, 5))
plt.plot(z1, 'b')
plt.plot(z2, 'r')
plt.show()
```

4) Plot the Autocorrelation Functions for those time series with **plot_acf**:

```
plt.figure(figsize = (10, 5))
plot_acf(z1, lags=50)
plot_acf(z2, lags=50)
plt.show()
```

5) Compare the ACF plots against each other: find how they are similar and how they are different. Try to formulate, how to distinguish those AR models based only on their ACF plots.

6) Estimate the weight coefficient for your AR(1) model with formula $\phi = \rho_1$, meaning that the coefficient is equal the first lag point of ACF. Also, make sure that ACF points for AR(1) are changing in accordance with the polynomial law of $\rho(l) = \phi^l$.

7) Similarly, create two MA(1) processes = Moving Average of first order:

$$z_t = a_t - 0.8a_{t-1} \quad \text{and} \quad z_t = a_t - (-0.8)a_{t-1}$$

where a_t – normal distributed random value

```
z3 = np.zeros(100)
z4 = np.zeros(100)
ar = 0.2 * np.random.randn(100)
for i in range(1, 100):
   z3[i] = ar[i] - 0.8 * ar[i - 1]
   z4[i] = ar[i] + 0.8 * ar[i - 1]
plt.figure(figsize = (10, 5))
plt.plot(z3, 'b')
plt.plot(z4, 'r')
plt.show()
```

8) Similarly, plot the ACF for those time series with **lags = 25**.

9) Compare the ACF plots against each other: find how they are similar and how they are different. Try to formulate, how to distinguish those MA models based only on their ACF plots.

10) Estimate the weight coefficient for your MA(1) model from formula:

$$\theta_1^2 + \theta_1 / \rho_1 + 1 = 0,\ |\theta_1| < 1$$

11) Also, take notice that ACF points for MA(1) are changing in accordance with the rule – non-zero at first lag, zero everywhere else:

$$\rho_k = \begin{cases} \dfrac{-\theta_1}{1+\theta_1^2}, & k = 1 \\ 0, & k \geq 2 \end{cases}$$

12) And now, create the time series for ARMA(1, 1) model:

$z_t = 0.8 z_{t-1} + a_t - 0.3 a_{t-1}$ and $z_t = -0.8 z_{t-1} + a_t - 0.3 a_{t-1}$

where a_t – normal distributed random value, $z_0 = 1$.

Write your own *Python* code for these models, based on the combination from the previous examples.

13) Plot the figures of estimated data, also below plot the ACF for them.

14) There is a better way to generate the ARMA models in *Python*. For example, this code will generate time series data of ARMA (2, 2) model:

```
from statsmodels.tsa.arima_process import arma_generate_sample
ar = np.array([0.75, -0.25])   # AR autoregressive coefficients
ma = np.array([0.65, 0.35])   # MA moving-average coefficients
y = arma_generate_sample(np.r_[1, -ar], np.r_[1, ma], 100)
# estimate the time series with 100 points from ARMA(2, 2) model
```

15) Now let's try to analyze and estimate the unknown ARIMA model for the test/example time series data. Next, the same technique should be used by students in order to analyze and estimate the ARIMA model for their own variant task (based on last 2 digits of student ID card number).

16) Create the initial time series data with 24 points:

TEST = [0.00, 9.99, 12.89, 10.70, 5.12, -1.21, -6.50, -7.96, -4.30, 0.42, 3.41, 4.50, 3.57, 2.24, 1.78, 0.89, -1.20, -3.43, -2.35, -0.85, -0.21, -0.08, 0.95, 0.45]

17) Plot the data ***TEST*** and also plot its ACF estimation with **plot_acf()**.

18) Based on ACF, the data is clearly **quasi-stationary**, and **autoregressive** with **one coefficient being negative**. What's the best order of AR for this data? Let's find out.

19) Let's create three test models of ARIMA for this time series data: AR(1) = ARIMA(1, 0, 0); AR(2) and AR(3); no trends (trend = 'nc')

```
arima1 = ARIMA(TEST, order = (1, 0, 0))          # create model
model_fit1 = arima1.fit(disp = False, trend='nc') # fit it to your data
print(model_fit1.summary())        # print out the table of results
arima2 = ARIMA(TEST, order = (2, 0, 0))
model_fit2 = arima2.fit(disp = False, trend='nc')
print(model_fit2.summary())
arima3 = ARIMA(TEST, order = (3, 0, 0))
model_fit3 = arima3.fit(disp = False, trend='nc')
print(model_fit3.summary())
```

20) If everything is okay, you will see three tables, similar to this:

ARMA Model Results

Dep. Variable:	y	No. Observations:	24
Model:	ARMA(2, 0)	Log Likelihood	-41.543
Method:	css-mle	S.D. of innovations	1.201
Date:	Sun, 10 Mar 2019	AIC	89.086
Time:	23:50:38	BIC	92.620
Sample:	0	HQIC	90.024

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.y	1.5108	0.056	27.117	0.000	1.402	1.620
ar.L2.y	-0.9641	0.035	-27.509	0.000	-1.033	-0.895

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	0.7836	-0.6506j	1.0185	-0.1103
AR.2	0.7836	+0.6506j	1.0185	0.1103

21) The required coefficients of model are under the column **coef**. The error of its estimation is the next column **std err**.

22) **How to choose the best model based on these summary tables?** First of all, you should look at **AIC** – Akaike information criterion, which is an estimator of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models, also based on the number of their parameters. Thus, AIC provides a means for model selection. In simple terms, **the lower AIC means the better model**.

23) Also, there is a similar **BIC** – Bayesian information criterion, modification of AIC. It is based, in part, on the likelihood function and it is closely related to the Akaike information criterion, but the penalty term for extra parameters is larger in BIC than in AIC. **The model with the lowest BIC is preferred**.

24) And there is also a **HQIC** – Hannan-Quinn information criterion, very similar to the BIC. Accordingly, **the lower HQIC means the better model**.

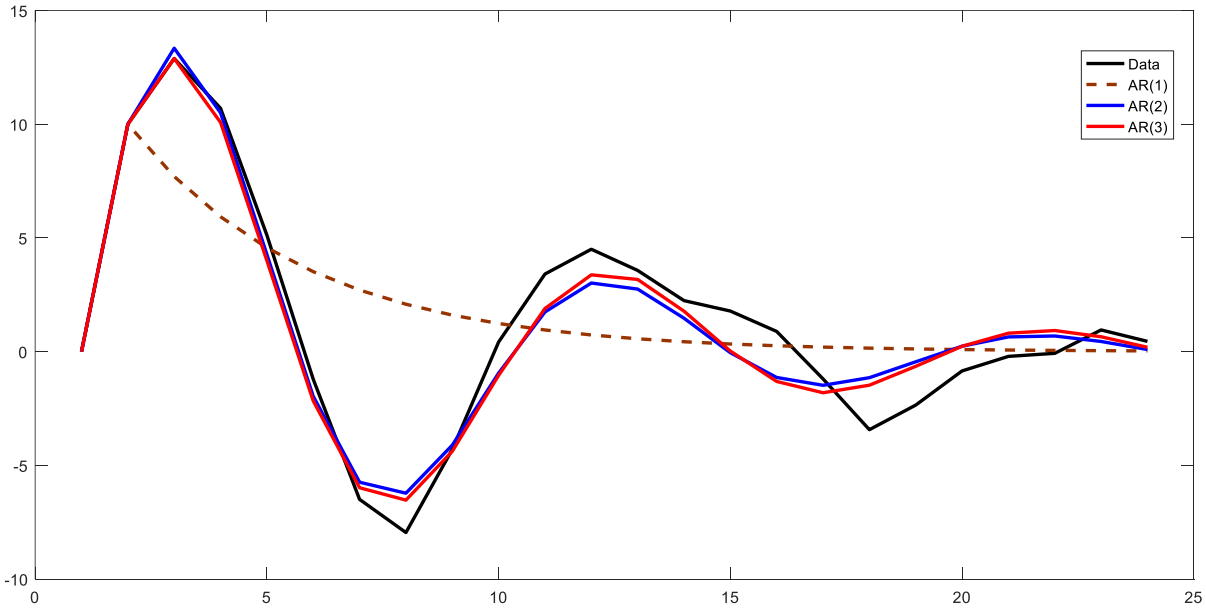
25) In any case, looks like the best model will have all of those information criterions at minimum against every other model. On practice, it is highly recommended to choose model based on the **BIC** value, because the penalty term for extra parameters is larger than AIC or HQIC, and for the ARIMA models it is essential to minimize the order of model.

26) For our test time series data the **AR(2) should be the best model**, based on comparing AIC, BIC and HQIC with other models.

27) There is also another simple way to choose the best fit model of ARIMA for data, just by comparing them visually. You can plot the fitted points of your ARIMA model on the same figure as data:

plt.plot(model_fit.fittedvalues)

28) For example, for the ***TEST*** data, AR(1), AR(2), AR(3):



AR(1) is a bad fit for the test data, AR(2) and AR(3) are really close. AR(3) have more parameters/order than AR(2), making it informationally bloated, thus, we should stop on the second order of autoregressive and choose **AR(2)** as the best fit. The same result as by choosing with AIC/BIC/HQIC.

29) Now try to estimate the weight coefficients for AR(1) and AR(2) models of ***TEST*** data with your own calculations, based on ACF function. Compare your results against those from tables.

30) To estimate the AR(1) one weight coefficient:

$$\phi = \rho_1$$

where ρ_1 – ACF at lag point = 1.

31) To estimate the AR(2) two weight coefficients:

$$\phi_1 = \frac{\rho_1(1-\rho_2)}{1-\rho_1^2}, \quad \phi_2 = \frac{\rho_2 - \rho_1^2}{1-\rho_1^2}$$

where you will need ACF at lag points 1 and 2

32) Compare your results against those from tables of ARIMA results. They should be really close, but not exactly the same.

33) Now, estimate the best ARIMA model with the same technique for your time series data variant (based on your last 2 digits of student ID card number). To load the data into time series ***Z*** use this:

```
file = h5py.File('12.mat', 'r')
data = file.get('z12')
Z = np.array(data)
```

34) Plot the data on figure. Plot the ACF for your time series.

35) Estimate the best ARIMA model for your data and its best order. For simplification, the data, presented to students, correspond only to the pure AR and MA models, meaning that you should use for estimation only such ARIMA classes as `ARIMA` with `order = (p, 0, 0)` or `order = (0, 0, q)`.

36) Choose the best fit of ARIMA based on information criterions, write down the coefficient for this model. Provide these coefficient and type of model (AR or MA) to your teacher for check.

37) Next, try to find the best fit of ARIMA model to your data without any constraints, meaning, use the ARIMA class with absolutely different sets of parameters for `order = (p, d, q)`. Find the best fit based on AIC/BIC/HQIC for ARIMA model among every other possible model.

38) Show your final results and the *Jupyter*-notebook to your teacher for check.

3. Report requirements

Report in Jupyter-notebook should include: number of laboratory task, name of student, student ID or variant number, student group number and the complete task (code, plots and figures) with student comments in the report.