VT2023: IL2233 Lab 2
ARIMA Model and Prediction

Zhonghai Lu

April 6, 2023

1 Introduction

In statistical time-series analysis, AR, MA, and ARMA models or processes are common models or processes which can be used to describe a time series based on its inherent time-dependent structure. To fit a stationary time series, the model itself has to be stationary. However, not all AR models are stationary. Thus understanding the stationarity property of AR models is important. For MA models, invertibility is important because different MA models may have the same autocorrelation relation, resulting in the same ACF graph. This creates the problem that one ACF graph can be mapped to multiple MA models. To avoid such ambiguity, the invertibility of MA models should be ensured. ARMA is a combination of AR and MA models. Thus we need to check both stationarity and invertibility of ARMA models. Invertibility is the counterpart to stationarity for the MA part of an ARMA process.

ARIMA is a commonly used prediction model for time series data. ARIMA assumes the time-series data stationary, or if not stationary, a difference operator can be applied to make the series stationary. A stationary (weakly stationary) series implies constant mean (no trend) and constant variance. An ARIMA(p, d, q) model consists of three parts: the AR(p) part, the differencing part (d), and the MA part (q), where p, d and q are three order parameters corresponding to the three parts, respectively. All p, d, and q are natural numbers. The AR part is to regress a predicted future value onto its previous values with a lag up to p. The differencing part is to remove the non-stationarity of time series, making it stationary. The MA part is to predict future values using its previous moving average prediction errors up to lag q.

To facilitate the understanding and application of the ARIMA model, we describe in detail the Box-Jenkins ARIMA modeling and prediction methodology.

2 The Box-Jenkins Methodology

When constructing an ARIMA(p, d, q) model for time-series data prediction, there are four main steps following the Box-Jenkins methodology. We assume that the series is already checked that it is not a random series.

1. Check stationarity and detect seasonality. The first step in developing a Box-Jenkins model is to determine if the series is stationary and if there is any significant seasonality that needs to be modeled.

- We first check if the time series is stationary. If yes, then go to Step (2). If not, the time series has likely a trend component. In this case, remove any trend component embedded in the data. A typical way to remove trend is to differencing two neighboring values (lag-1 differencing) in the time series. The order of differencing is parameterized by a number d. With the trend component be removed, an ARIMA process becomes ARMA process (the differencing "d" part is now handled and its consequence will be considered in prediction).

  Although Box and Jenkins recommend the differencing approach to achieve stationarity, fitting a curve and subtracting the fitted values from the original data can also be used in the context of Box-Jenkins models.

  Besides visual inspection of relevant plots, we can use statistical hypothesis test methods to check stationary. One common method is the Augmented Dickey Fuller (ADF) test and the other is the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test. Both check for stationarity of a series around a deterministic trend. Note that both tests assume *non-seasonal* series. To test stationarity with seasonal series. there exist other methods to test seasonal unit roots such as HEGY seasonal unit-root test and CH seasonal unit root test, which are beyond the scope of the course.

  The ADF test is a *unit-root* test method for stationarity. The null hypothesis for this test is that there is a unit root, thus the series is not stationary [1].

  After running the test, we can interpret the results according to p-value. In general, a p-value of less than 5% (0.05) means we can reject the null hypothesis that there is a unit root. We can also compare the calculated test statistic with a tabulated critical value. If the test statistic is more negative than the table value, reject the null hypothesis of a unit root. Note: The more negative the test statistic, the stronger the evidence for rejecting the null hypothesis of a unit root.

- Consider seasonality.

  At this stage, our goal is to detect seasonality, and if it exists, to identify the order for the seasonal autoregressive and seasonal moving average terms.

  How to consider if there is any seasonality in the time-series data? Seasonality means periodic behavior, indicating a periodic pattern in the data. Many time-series data might not have a strict seasonality (fixed periods in days, months, years) like in financial time-series data. Our interest is to study if the data periodicity in the scale of seasonality has significant impact on the ARIMA prediction model. If yes, a Seasonal ARIMA (SARIMA) model shall be considered. Otherwise, we can use the ARIMA model without seasonality. For many series, the period is known and a single seasonality term is sufficient. For example, for monthly data we would typically include either a seasonal auto-regression 12 term or a seasonal moving average 12 term.

  In time-series analysis, season and cycle are close but different concepts. A time series may contain seasonal variation or seasonality, which means regular repetitive cycles over time. For example, a repeating pattern within each year or month is known as a seasonal variation. In fact, seasonality is a more general term representing a repeating pattern within any fixed period. A cycle structure in a time series is common, but may or may not be seasonal. If it consistently repeats at the same frequency, it is seasonal, otherwise it is not seasonal and is still cyclic.

[1] To understand this, you need to understand unit roots. You can safely skip this detail.

2. Order identification: After removing trend and addressing seasonality, the next step is to identify the order (i.e., the p and q) of the autoregressive and moving average terms.

   The primary tools are the autocorrelation plot and the partial autocorrelation plot. The sample autocorrelation plot and the sample partial autocorrelation plot are compared to the theoretical behavior of these plots when the order is known.

   In short, the auto-correlation functions, ACF and PACF, are used to determine p and q. Specifically, the ACF can be used to determine q for MA(q) and the PACF to determine p for AR(p) of the ARMA(p, q) model.

   Let $\{X_t, t \in Z\}$ be a stationary time series. ACF measures the correlation between itself $\{X_t\}$ and a shifted or lagged version of itself $\{X_{t-h}\}$, where value h is referred to as "lag".

   Partial autocorrelation function (PACF), by its name, gives the partial correlation of a time series with its own lagged values but with the values of the time series at all intermediate lags removed. Specifically, PACF measures the correlation between itself $\{X_t\}$ and a lag-h shifted version of itself $\{X_{t-h}\}$, but with all intermediate lag values $\{X_{t-1}\}$, $\{X_{t-2}\}$, $\cdots$, $\{X_{t-(h+1)}\}$, removed. By contrast, the ACF does not control for other lags.

   Autocorrelation analysis can help us to understand how much impact a variable's current values would have on its future values. An autocorrelation of +1 means a fully positive correlation, indicating that an increase in one time-series leads to a proportional increase in the other. By contrast, an autocorrelation of -1 is the opposite, i.e., fully negative correlation, meaning an increase in one time-series leading to a proportional decrease in the other. A value of 0 indicates no correlation.

3. Model selection: Fine-tune the (p, q) values for the ARIMA model and model selection based on the principle of *parsimony*.

   In step (2), p and q values are determined separately, but ARIMA model is a combination of AR and MA models. In Step (2), the p and q values actually give the maximum good values for possible p and q. In this step, their combinations, i.e., [1, p] and [1, q] can be tested in combination to find a best fitting ARIMA model. Usually a grid-based brute-force search can be used but is very time consuming.

   To choose the right order for both the AR(p) and MA(q) processes, we may try different combinations of orders, fit an ARIMA model with these orders, and use a criterion for model selection. A common criterion is known as Akaike Information Criterion (AIC) defined as follows.

$$AIC = -2Log(L) + 2k, \tag{1}$$

   where L is the likelihood of the data and k the number of parameters in the model. This criterion is useful for selecting the order (p, d, q) of an ARIMA model.

   - The first term is the indicator of accuracy. If the model is accurate, the variance of the error would be small and so does the first term.
   - The second term is the indicator of the model size. If the model size is large, the AIC becomes large. For an ARIMA model, the higher the order is, the more accurately the model fits the training data set. However, if the order is too high (the model has too many parameters), it is likely to overfit.

AIC intends to balance these two factors: model accuracy and model complexity. In general, smaller AIC leads to a better model. Hence, we select the model with the lowest AIC when comparing models. This is so called principle of parsimony, as it prefers models with fewer parameters; more parameters will increase the AIC score and thus penalize the model. Note that the AIC cannot be used to select the order of differencing (d), because differencing the data changes the likelihood (L) of the data. The AICs of models with different orders of data differencing are thus not comparable.

The auto_arima() function (`https://pypi.org/project/pmdarima/`) conducts a grid search, scanning a range of (p, q) value combinations. The best model with the least AIC will be chosen as the output. But this grid search is usually very time consuming. If not using the auto search function, one can try different combinations by experience.

4. Model validation and prediction. After the ARIMA model is well trained, it is almost ready to be used for prediction. We start with in-sample and then out-of-sample prediction.

   - In-sample prediction means that the model is applied to the data points in the training set, to check how good fit the estimated values against the actual values.
   - Out-of-sample prediction means that the model is used to forecast future data points, which do not exist in the training data set.

   With in-sample prediction, we can obtain the remainder sequence of the original data sequence by differencing the original value and the predicted value. Then we can test the randomness of the remainder. If the remainder contains pure random data, this means that the trained ARIMA model is effective. If not, the model needs improvement, and the step can be iterated to Step 3.

   After model validation with in-sample prediction, the model is ready for forecasting future values.

3 Purpose

The laboration intends to deepen the students' understanding of the ARIMA model and to use it to solve practical modeling and prediction problems. It contains both theoretical and practical parts. To this end, the lab has the following objectives:

- Deepen understanding of AR(p), MA(q) and ARMA(p, q) models and be able to analyze their statistical properties such as stationarity, ACF/PACF characteristics etc.
- Use the ARIMA process to model time-series data
- Use the ARIMA model for in-sample and out-of-sample prediction

This lab is a group work, with a size of two students per group.

4 Preparation and Tasks

In this lab, we will use Python statsmodels library (`https://www.statsmodels.org/stable/index.html`) to study AR, MA, and ARIMA models, and use ARIMA for time-series prediction.

4.1 Installation

If you don't have statsmodels installed, then do the following.

1. Download and install Anaconda (Python will be installed as part of the installation) `https://www.anaconda.com/products/individual`
2. Open a command window with Anaconda
3. Run the following command to install statsmodels

   conda install -c conda-forge statsmodels

4.2 Task 1: Stationarity of AR models

Consider the following AR(p) models, where p is the order, and $\{\epsilon_t\}$ is a standard Gaussian noise sequence.

1. AR(1): $x_t = 0.8x_{t-1} + \epsilon_t$
2. AR(1): $x_t = -1.1x_{t-1} + \epsilon_t$
3. AR(2): $x_t = x_{t-1} - 0.5x_{t-2} + \epsilon_t$
4. AR(2): $x_t = x_{t-1} + 0.5x_{t-2} + \epsilon_t$

Complete the following tasks:

- Generate four time series according to the above AR models, one for each. Draw a line plot for each time series.
- Judge whether each time series is stationary or not, by visual inspection.
- Judge whether each time series is stationary or not, by its model's auto-regressive coefficient values.
- Call the following statsmodel function to judge if each AR process is stationary.

```
arma_process = sm.tsa.ArmaProcess(arparams, maparams)
print(arma_process.isstationary)
```

- Use the unit-root based Augmented Dickey-Fuller (ADF) test to check if each time series generated by the AR models is stationary or not. Do the results match your visual inspection?

Answer the following questions:

- With visual inspection, how do you identify if a time series is stationary or not?
- How do you judge the stationarity of time series using the unit-root method? Does it always give correct results?
- What is the role of component ϵ_t in the model? Why is it important?
- To have an AR(p) model be stationary, is there any requirement on the auto-regressive coefficients? List the constraints for AR(1) and AR(2) models.

4.3 Task 2: ACF, PACF of AR models

Given the following four AR(p) models, where p is the order and $\{\epsilon_t\}$ is a standard Gaussian noise sequence.

1. AR(1): $x_t = 0.8x_{t-1} + \epsilon_t$
2. AR(1): $x_t = -0.8x_{t-1} + \epsilon_t$
3. AR(2): $x_t = x_{t-1} - 0.5x_{t-2} + \epsilon_t$
4. AR(2): $x_t = -x_{t-1} - 0.5x_{t-2} + \epsilon_t$

Complete the following tasks:

- Generate four time series according to the above AR models, one for each. Are all of them stationary? Draw a line plot for each time series.
- Draw histogram, density plot, and box plot for each time series with 1000 data points. Are there any outliers? Why?
- Draw lag-1 and lag-2 plots for each time series. Do you observe any auto-correlation from the lag plots?
- Draw ACF graph for each time series.
- Draw PACF graph for each time series.

Answer the following questions:

- What characteristics can you observe from the ACF graphs of the AR(p) models?
- What characteristics can you observe from the PACF graphs of the AR(p) models?

4.4 Task 3: Invertibility, ACF, PACF of MA models

Given the following four MA(q) models, where q is the order, and $\{\epsilon_t\}$ is a standard Gaussian noise sequence.

1. MA(1): $x_t = \epsilon_t - 2\epsilon_{t-1}$
2. MA(1): $x_t = \epsilon_t - 0.5\epsilon_{t-1}$
3. MA(2): $x_t = \epsilon_t - \frac{4}{5}\epsilon_{t-1} + \frac{16}{25}\epsilon_{t-2}$
4. MA(2): $x_t = \epsilon_t - \frac{5}{4}\epsilon_{t-1} + \frac{25}{16}\epsilon_{t-2}$

Complete the following tasks:

- Generate four time series according to the above MA(q) models, one for each. Draw a line plot for each time series.
- Judge whether each time series is invertible or not, by visual inspection.

- Judge whether each time series is invertible or not, by its models's auto-regressive coefficient values.
- Call the following statsmodel function to judge if each MA process is invertible.

```
arma_process = sm.tsa.ArmaProcess(arparams, maparams)
print(arma_process.isinvertible)
```

- Draw histogram, density plot, and box plot for each time series with 1000 data points. Are there any outliers? Why?
- Draw lag-1 and lag-2 plots for each time series. Do you observe any auto-correlation from the lag plots?
- Draw ACF graph for each time series.
- Draw PACF graph for each time series.

Answer the following questions:

- Are all the MA models invertible? If not, which ones are invertible and which ones are not invertible?
- What characteristics can you observe from the ACF graphs of the MA(q) models?
- What characteristics can you observe from the PACF graphs of the MA(q) models?
- To have an MA(q) model be invertible, is there any requirement on the auto-regressive coefficients? List the constraints for MA(1) and MA(2) models.

4.5 Task 4: Stationarity, ACF and PACF of ARMA models

Consider the following three ARMA(p, q) models.

- AR(1) [ARMA(1, 0)]: $x_t = 0.8x_{t-1} + \epsilon_t$
- MA(1) [ARMA(0, 1)]: $y_t = \epsilon_t + 0.7\epsilon_{t-1}$
- ARMA(1, 1): $z_t = 0.8z_{t-1} + \epsilon_t + 0.7\epsilon_{t-1}$
- ARMA(1, 1): $z_t = -0.8z_{t-1} + \epsilon_t - 0.7\epsilon_{t-1}$

Complete the following tasks:

- Generate three time series according to the above ARMA(p, q) models, one for each. Draw a line plot for each time series.
- Judge whether each time series is stationary or not, by visual inspection.
- Use the ADF test method to judge whether each time series is stationary or not. Do the results match your visual inspection?

- Call the corresponding statsmodels functions to judge if each ARMA process is stationary and invertible.
- Draw histogram, density plot, and box plot for each time series with 1000 data points. Are there any outliers? Why?
- Draw lag-1 and lag-2 plots for each time series. Do you observe any auto-correlation from the lag plots?
- Draw ACF graph for each time series.
- Draw PACF graph for each time series.

Answer the following questions:

- What characteristics can you observe from the ACF, PACF graphs of the AR(p) model?
- What characteristics can you observe from the ACF, PACF graphs of the MA(q) model?
- What characteristics can you observe from the ACF, PACF graphs of the ARMA(p, q) model?

By answering the questions, you are able to fill in the following table.

Table 1: The ACF and PACF characteristics of ARMA models

Model	**ACF**	**PACF**
AR(p)		
MA(q)		
ARMA(p, q)		

4.6 Task 5: ARIMA modeling and prediction

Now we use the ARIMA(p, d, q) model to fit a time series sequence and prediction. The time series is a sequence of the average temperature change of global weather recorded per year from 1880 to 1985. It has in total 106 data points.

The data are given in the Appendix. You first need to record the data in a csv file, which you can then access using a function in pandas. Each data entry stays on one line. The first four lines and the last line of your temperature.csv file looks like the following:

```
year, temperature change
1880,-0.4
1881,-0.37
1882,-0.43
...
1985,0.05
```

Based on the Box-Jenkins methodology, conduct the modeling procedure as follows to fit an ARIMA(p, d, q) model to the series.

- Step 1. Randomness test.

  Check if the sequence is random using line plot, lag-1 plot and Lung-Box test. If it is, you are done. If it is not, continue to the next steps.

- Step 2. Stationarity test and differencing.

  Test the stationarity of the series. If it is not stationary, use the difference operation to generate a differenced sequence. In this step, you determine the value of d. You may need to do second difference for some time series, if necessary.

- Step 3. Model identification.

  Draw ACF and PACF plots of the sequence. Based on the ACF, PACF graphs, you determine the values for p and q.

- Step 4. Parameter estimation and model optimization.

  Since there can be multiple models which give good results, you can choose one optimal one based on, e.g., the AIC criterion.

- Step 5. Model validation.

  This is to determine if the fitted model is acceptable. If not, go back to Step 4. This is done through in-sample prediction. With in-sample prediction, you can generate a remainder sequence so as to test if the fitted model is valid based on the randomness of the remainder sequence. The residual series of a well-fitted model contains only random noise.

  You can also calculate some accuracy metrics, e.g. mean-squared error (MSE).

- Step 6. Model forecasting.

  Use the validated model for future-value prediction. This is the out-of-sample prediction.

We start with drawing visual graphs to inspect any possible structure or pattern or characteristics in the data. Then we use the Box-Jenkins methodology to fit an ARIMA(p, d, q) model to the data.

Perform relevant activities and answer the following questions:

- Draw a line plot for the time series data. Do you observe any trend, season in the data?
- Draw histogram, density plot, heat map, and box plot for the time series data. Are there any outliers? Why?
- Draw lag-1 and lag-2 plots for the time series data. Do you observe any auto-correlation from the lag plots?
- Is the series random? How do you check it? Are the three methods (line plot, lag-1 plot, and Ljung-Box test) give the same results?
- Is the series stationary? Try with visual inspection and ADF test. Do they give the same results?

- If the series is not stationary, how do you make it stationary? If you use the differencing operation, how do you decide a proper order of differencing without under-differencing/over-differencing?
- What (p, d, q) values do you use? How do you determine them?
- After model fitting, is the reminder series (in-sample prediction) considered to be white noise?
- What is the MSE of the fitted model for the data?
- For out-of-sample prediction, do the predicted values (10 steps) reflect the trend and fluctuation of the series?

4.7 Task 6: Series transformation

There are a few transformations which can be used to stabilize trend, co-variance, and make the series stationary. In particular, differencing is the transformation recommended by the Box-Jenkins methodology.

Answer the following general questions:

- What are the common transformation techniques applicable to turn a non-stationary series into a stationary series?
- What is the Box-Cox transform? Give its definition and explain its generality.
- Can a differencing operation remove a *linear* trend? Give an example by generating a synthetic series, and draws the series before differencing and after differencing.
- Can a differencing operation remove an *exponential* trend? If not, which additional transformation needs to be taken? Give an example by generating a synthetic series, and plots the series before transformation and after transformation, before differencing and after differencing.
- Can a differencing operation remove a *seasonal* (periodic) trend? If yes, under what condition? Give an example by generating a synthetic series, draw its ACF, and draws the series before differencing and after differencing with different step length.

5 Task 7. Documentation and Deliverables

Your final task is documentation and deliverables.

- Write a short, concise technical report (1) about the tasks you have done, (2) to record the results you have obtained, and (3) to answer all the questions in the lab.
- Prepare a deliverable that zips all your source code and the result figures. Write a short Readme.txt file to describe the zip folder structure and purpose of each file.

After your lab results have been approved by the lab assistant, submit your report together with your source code zip file via the Canvas course website:

`https://canvas.kth.se/courses/39013`

6 Appendix

6.1 ARIMA for time series forecasting

Time series forecasting is a process of predicting future values of a time series based on past values using a statistical or a machine learning model. You can find some tutorials about time-series forecasting with ARIMA on the web. Two examples are given below.

Online tutorial: Python — ARIMA Model for Time Series Forecasting. `https://www.geeksforgeeks.org/python-arima-model-for-time-series-forecasting/`

On the Kaggle website, you can read "Time Series For beginners with ARIMA", `https://www.kaggle.com/freespirit08/time-series-for-beginners-with-arima`

Be aware that you might not be able to run some code or reproduce exactly the same results, due to library updates or slight differences in dataset or stochastic behavior. Sometimes, you need to adjust the code. To do that, you need to understand the code in detail.

6.2 Statsmodels for time series modeling and forecasting

Since we are using statsmodels, it is always useful to visit its website to find exact information. Remember that Statsmodels has very rich comprehensive documentation. Most of them are far beyond the scope of the course.

Two links are provided, if you are interested in digging further. General examples. `https://www.statsmodels.org/stable/examples/`. Time series analysis examples. `https://www.statsmodels.org/stable/examples/index.html#time-series-analysis`

6.3 pmdarima for ARIMA modeling

To efficiently search the model parameters for ARIMA, you may use auto_arima() from the pmdarima library, which implements a step-wise search or grid-search of the model parameters to minimize AIC (Akaike Information Criteria). The article "Efficient Time-Series Analysis Using Python's Pmdarima Library" may give you a hint.

`https://towardsdatascience.com/efficient-time-series-using-pythons-pmdarima-library-f6825407b7f0`

Installation command: pip install pmdarima

6.4 Temperature change data from year 1880 to 1985

Year, Temperature change 1880,-0.4 1881,-0.37 1882,-0.43 1883,-0.47 1884,-0.72 1885,-0.54 1886,-0.47 1887,-0.54 1888,-0.39 1889,-0.19 1890,-0.4 1891,-0.44 1892,-0.44 1893,-0.49 1894,-0.38 1895,-0.41 1896,-0.27 1897,-0.18 1898,-0.38 1899,-0.22 1900,-0.03 1901,-0.09 1902,-0.28 1903,-0.36 1904,-0.49 1905,-0.25 1906,-0.17 1907,-0.45 1908,-0.32 1909,-0.33 1910,-0.32 1911,-0.29 1912,-0.32 1913,-0.25 1914,-0.05 1915,-0.01 1916,-0.26 1917,-0.48 1918,-0.37 1919,-0.2 1920,-0.15 1921,-0.08 1922,-0.14 1923,-0.13 1924,-0.12 1925,-0.1 1926,0.13 1927,-0.01 1928,0.06 1929,-0.17 1930,-0.01 1931,0.09 1932,0.05 1933,-0.16 1934,0.05 1935,-0.02 1936,0.04 1937,0.17 1938,0.19 1939,0.05 1940,0.15 1941,0.13 1942,0.09 1943,0.04 1944,0.11 1945,-0.03 1946,0.03 1947,0.15 1948,0.04 1949,-0.02 1950,-0.13 1951,0.02 1952,0.07 1953,0.2 1954,-0.03 1955,-0.07 1956,-0.19 1957,0.09 1958,0.11 1959,0.06 1960,0.01 1961,0.08 1962,0.02 1963,0.02 1964,-0.27 1965,-0.18 1966,-0.09 1967,-0.02 1968,-0.13 1969,0.02 1970,0.03 1971,-0.12 1972,-0.08 1973,0.17 1974,-0.09 1975,-0.04 1976,-0.24 1977,-0.16 1978,-0.09 1979,0.12 1980,0.27 1981,0.42 1982,0.02 1983,0.3 1984,0.09 1985,0.05