Pronóstico de la demanda en empresas retail

Técnica basada en Business Intelligence y Machine Learning

Raúl Benítez - Alberto Garcete

Tutores: PhD. Diego P. Pinto Roa - Ing. Aditardo Vázquez

Universidad Nacional de Asunción - Facultad Politécnica

Agosto 2018

ARIMA

Auto Regressive Integrated Moving Average (ARIMA): Es una generalización de Auto Regressive Moving Average (ARMA), que combina procesos autorregresivos (Autoregressive, AR) y procesos de media móvil (Moving Average, MA) creando así un modelo compuesto de series de tiempo. ARIMA (p, d, q) captura los siguientes elementos:

- AR: Autoregression. Un modelo de regresión que utiliza las dependencias entre una observación y un número de observaciones pasadas (p).
- I: Integrated. Para hacer la serie temporal estacionaria midiendo las diferencias de observaciones en diferentes momentos (d).
- MA: Moving Average. Un enfoque que tiene en cuenta la dependencia entre las observaciones y los términos de error residual cuando se usa un modelo de promedio móvil para las observaciones pasadas (q).

En econometría, los pronósticos de series de tiempo se aplican tradicionalmente utilizando modelos ARIMA, generalizado por Box y Jenkins [?]. ARIMA es un método estándar para el pronóstico de series de tiempo y habituales en el modelado de series temporales económicas y financieras.

ARIMA

A simple form of an AR model of order p, i.e., AR (p), can be written as a linear process given by:

$$x_t = c + \sum_{i=1}^{p} \emptyset_i \, x_{t-i} + \varepsilon_t$$

Where x_t is the stationary variable, c is constant, the terms in $\emptyset_i$ are autocorrelation coefficients at lags $1, 2, \ldots, p$ and ε_t , the residuals, are the Gaussian white noise series with mean zero and variance σ_ε^2. An MA model of order q, i.e., MA (q), can be written in the form:

ARIMA

$$x_t = \mu + \sum_{i=0}^{q} \theta_i \, \varepsilon_{t-i}$$

Where μ is the expectation of x_t (usually assumed equal to zero), the θ_i terms are the weights applied to the current and prior values of a stochastic term in the time series, and $\theta_0 = 1$. We assume that ε_t is a Gaussian white noise series with mean zero and variance σ_ε^2. We can combine these two models by adding them together and form an ARMA model of order (p, q):

ARIMA

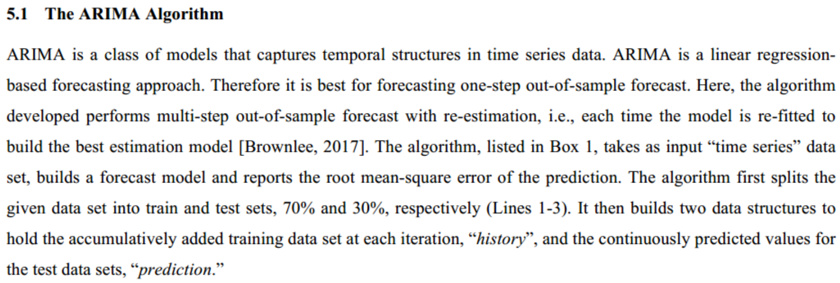
$$x_t = c + \sum_{i=1}^{p} \emptyset_i \, x_{t-i} + \varepsilon_t + \sum_{i=0}^{q} \theta_i \, \varepsilon_{t-i}$$

Where $\emptyset_i \neq 0, \theta_i \neq 0$, and $\sigma_\varepsilon^2 > 0$. The parameters p and q are called the AR and MA orders, respectively. ARIMA forecasting, also known as Box and Jenkins forecasting, is capable of dealing with non-stationary time series data because of its "*integrate*" step. In fact, the "integrate" component involves differencing the time series to convert a non-stationary time series into a stationary. The general form of a ARIMA model is denoted as $ARIMA\ (p, d, q)$.

ARIMA

With seasonal time series data, it is likely that short run non-seasonal components contribute to the model. Therefore, we need to estimate seasonal $ARIMA$ model, which incorporates both non-seasonal and seasonal factors in a multiplicative model. The general form of a seasonal ARIMA model is denoted as $ARIMA\ (p,d,q)\times(P,D,Q)S$, where p is the non-seasonal AR order, d is the non-seasonal differencing, q is the non-seasonal MA order, P is the seasonal AR order, D is the seasonal differencing, Q is the seasonal MA order, and S is the time span of repeating seasonal pattern, respectively. The most important step in estimating seasonal ARIMA model is to identify the values of (p,d,q) and (P,D,Q). Based on the time plot of the data, if for instance, the variance grows with time, we should use variance-stabilizing transformations and differencing. Then, using autocorrelation function (ACF) to measure the amount of linear dependence between observations in a time series that are separated by a lag p, and the partial autocorrelation function (PACF) to determine how many autoregressive terms q are necessary and inverse autocorrelation function (IACF) for detecting over differencing, we can identify the preliminary values of autoregressive order p, the order of differencing d, the moving average order q and their corresponding seasonal parameters P, D and Q. The parameter d is the order of difference frequency changing from non-stationary time series to stationary time series.

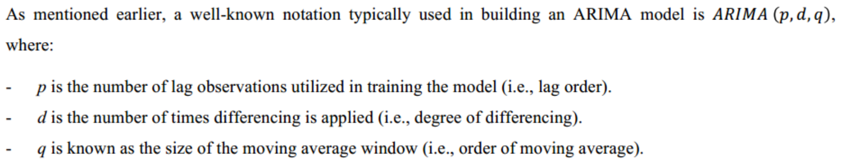
ARIMA



5.1 The ARIMA Algorithm

ARIMA is a class of models that captures temporal structures in time series data. ARIMA is a linear regression-based forecasting approach. Therefore it is best for forecasting one-step out-of-sample forecast. Here, the algorithm developed performs multi-step out-of-sample forecast with re-estimation, i.e., each time the model is re-fitted to build the best estimation model [Brownlee, 2017]. The algorithm, listed in Box 1, takes as input "time series" data set, builds a forecast model and reports the root mean-square error of the prediction. The algorithm first splits the given data set into train and test sets, 70% and 30%, respectively (Lines 1-3). It then builds two data structures to hold the accumulatively added training data set at each iteration, "*history*", and the continuously predicted values for the test data sets, "*prediction*."

ARIMA

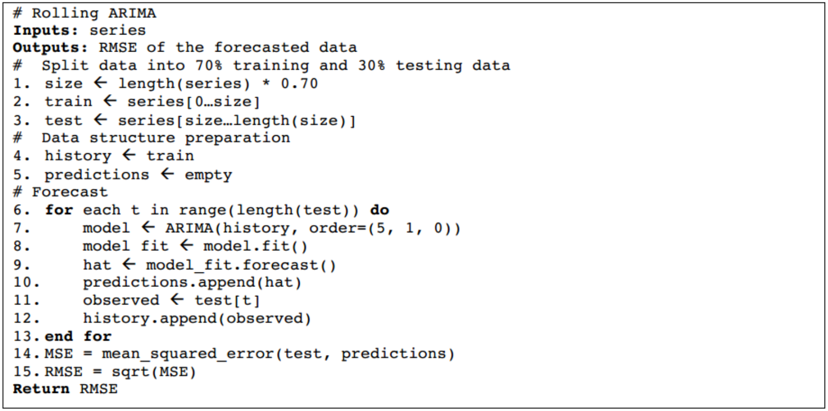


As mentioned earlier, a well-known notation typically used in building an ARIMA model is $ARIMA\,(p, d, q)$, where:

- p is the number of lag observations utilized in training the model (i.e., lag order).
- d is the number of times differencing is applied (i.e., degree of differencing).
- q is known as the size of the moving average window (i.e., order of moving average).

ARIMA

Box 1. The developed rolling ARIMA algorithm.

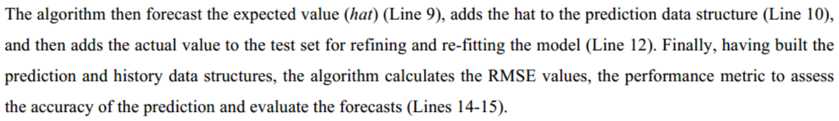


```
# Rolling ARIMA
Inputs: series
Outputs: RMSE of the forecasted data
#  Split data into 70% training and 30% testing data
1.  size ← length(series) * 0.70
2.  train ← series[0…size]
3.  test ← series[size…length(size)]
#  Data structure preparation
4.  history ← train
5.  predictions ← empty
# Forecast
6.  for each t in range(length(test)) do
7.      model ← ARIMA(history, order=(5, 1, 0))
8.      model fit ← model.fit()
9.      hat ← model_fit.forecast()
10.     predictions.append(hat)
11.     observed ← test[t]
12.     history.append(observed)
13. end for
14. MSE = mean_squared_error(test, predictions)
15. RMSE = sqrt(MSE)
Return RMSE
```

ARIMA

Through Lines 6-12, first the algorithm fits an *ARIMA* (5,1,0) model to the test data (Lines 7-8). A value of 0 indicates that the element is not used when fitting the model. More specifically, an *ARIMA* (5,1,0) indicates that the lag value is set to 5 for autoregression. It uses a difference order of 1 to make the time series stationary, and finally does not consider any moving average window (i.e., a window with zero size). An *ARIMA* (5,1,0) forecast model is used as the baseline to model the forecast. This may not be the optimal model, but it is generally a good baseline to build a model, as our explanatory experiments indicated.

ARIMA


The algorithm then forecast the expected value (*hat*) (Line 9), adds the hat to the prediction data structure (Line 10), and then adds the actual value to the test set for refining and re-fitting the model (Line 12). Finally, having built the prediction and history data structures, the algorithm calculates the RMSE values, the performance metric to assess the accuracy of the prediction and evaluate the forecasts (Lines 14-15).