CSCI-GA.3033-016. Big Data and ML Systems - Fall 2017

Assignment 2a

November 7, 2017

In this (sub-)assignment, you will be working with time series data that we have provided to you, and learn how to implement some of the techniques taught in class (AR, ARMA and ARIMA). **The recommended language for this assignment is Python, because of the `statsmodels` package.**

Description

This is the bus trace data from the New York City MTA. Each bus in the city has a GPS tracking unit that relays the location of the bus, along with several other pieces of information such as the time stamp accurate to the second, route ID, direction of motion, previous bus stop, next bus stop and vehicle ID. The *route ID* is the bus number (e.g. M15, B44, etc.) that represents a route, whereas the *vehicle ID* refers to a physical vehicle. There will be several buses with the same route ID, having different vehicle IDs.

The data given to you is in two parts:

- *Time series data*: This is the series of average speeds on each segment. Each value represents the average speed of all vehicles that were moving on that segment between the previous time stamp and current time stamp. The gap between two time stamps is 10 minutes.
- *Segment graph*: This is a graph representation of all the road segments – the list of segments and their corresponding neighboring segments.

You will be following the Box-Jenkins methodology to find the best model for the time series in this dataset, i.e. basically building a model to predict the average speed on a segment given historical values. The methodology is a stochastic-model building process and is an iterative approach that consists of the three steps:

- **Identification.** Use the data and all related information to help select a sub-class of model that may best summarize the data.
- **Estimation.** Use the data to train the parameters of the model (i.e. the coefficients).
- **Diagnostic Checking.** Evaluate the fitted model in the context of the available data and check for areas where the model may be improved.

It is an iterative process, so that as new information is gained during diagnostics, you can circle back to step 1 and incorporate that into new model classes.

Identification

Test that the time series are *stationary*, before fitting any kind of AR or MA model. If not stationary, then the time series would have to be differenced repeatedly until we obtain a stationary process. Then, plot the ACF (Autocorrelation Function) and view the plots. Below is the table that provides guidelines on model selection based on the ACF plot (available on Wiki too). See here for further guidelines, on picking p, d and q wherever needed.

Shape	Indicated Model
Exponential, decaying to zero	AR model. Use the PACF plot to identify the order.
Alternating positive and negative, decaying to zero	AR model. Use the PACF plot to identify the order.
One of more spikes, rest are essentially zero	MA model, order identified by where plot becomes zero.
Decay, starting after a few lags	Mixed AR and MA model
All zero or close to zero	Data is essentially random
High values at fixed intervals	Include seasonal AR term
No decay to zero	Series is not stationary

In the table below, PACF is the Partial Autocorrelation Function. Read about the two functions on Wikipedia if you are not familiar with them.

This is usually the most time-consuming step, as there is trial and error involved. For instance, you might have to try multiple value(s) of p, d and q. The same model may or may not work on all road segments as well. Functions in the either `statsmodels.tsa.stattools` or `statsmodels.graphics.tsaplots` may be useful.

Estimation

In the next step, once you have chosen a model, you estimate the parameters of the model, such as p, d and q in a $ARIMA(p, d, q)$ process. For this, refer to the models available in `statsmodels.tsa`. Do not use the entire time series for estimation. Use roughly about two thirds (2/3rd) of the length for training (i.e. fitting the model) and remaining one third (1/3rd) for testing (i.e. forecasting). The MTA data is available for three months, so you could use the first two months for training and last month for forecasting/testing. Or you could do it another way. Also, compute, if possible, the same model parameters using `mllib`'s linear regression library. There are three different linear regression methods you can try – ordinary least squares, ridge regression and lasso regression.

Diagnostic Checking

Finally, the computed models need to be evaluated for goodness. This is called model validation. Refer here for details on model validation. Generate *4-plots* of the residuals from the model estimation, for each model, and also the models estimated using the linear regression libraries in Spark `mllib`, if you used them. See here for details on the 4-plots. Run the *Chi-Squared* goodness-of-fit tests and include the results in your submission.

Points on Implementation

- You are expected to use Spark to achieve parallelism in this assignment. Ideas for usage:
  - Treat a time series as a single object. Create an RDD of several time series objects and use `map` and similar functions to enable parallel executions (such as during model estimation).
  - Compute Goodness-of-Fit tests in the model validation phase using the functions in `mllib.stat.Statistics`.
- You are expected to pick **any five** segments to work on, for the time series analysis. You can do more, if you like.
- You might have to repeat the steps of the Box-Jenkins methodology a few times before converging on the best model for the time series in focus.
- Other possibly useful packages in Python – `numpy`, `pandas`, `matplotlib`.

Deliverables

- For the Box-Jenkins methodology, list the number of segments you picked to work with, as well as the models you attempted to fit the data on. Mention that model that worked best for each segment that you picked.
- Plots: Five different sets of plots (acf, pacf, 4-plot) for the five different chosen time series. For the same five time series, also show run-sequence plots of the forecasted values compared with the actual values (only for testing period). Run-sequence plot is a simple plot of the time series with time on the x-axis and value on the y-axis.
- Numbers from the model validation, including the Goodness-of-Fit tests.
- Your code and write up of your submission. Include details about your code, your workflow and other references you used.

Useful Resources

- A full run-through of creating an ARIMA model for time series forecasting in Python using `statsmodels`, with an illustrated example – `https://machinelearningmastery.com/arima-for-time-series-fo`
- `statsmodels` package documentation – `http://www.statsmodels.org/stable/index.html`
- Box-Jenkins model steps – `http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc445.htm`