MTA Time Series Analysis

Preparing the Data

Finding the 5 Best Segments

My metric for ‘best’ segments is simple: Segments with the least nan’s. What that means is segments that for almost all time windows, they have a bus traveling on that route. Picking segments with the most traffic would provide for the best time series as there would be little missing data. To save time, I found the best segments for the first week. Logically, a bus stop with high traffic for an entire week will not suddenly stop having buses travel through it for the next few months. If you look at the plots of each time series, you will in fact see a good population of data points.

The code for finding the best segments is in the file: findbestsegments.py

Extracting the Time Series Data for each segment into a CSV

After finding the top 20 segments, I wanted to extract the time series of the best 5 into a csv file. So, for every csv in the top 5, I ran a small script to extract the dataframe and save the timestamps and average speeds into a csv.

The code for creating the csvs is in the file: createcsv.py

Handling NaN’s and outliers

One issue with the data is when a bus is not running during a time interval, the time series simply displays ‘nan.’ This breaks the ARIMA model. So, I had the choice to either zero out nan’s or replace them with the mean of the entire series. I chose the later. Zeroing out nan’s is not quite logical as nan’s are not always just from buses not running. They can occur from messing up data collection. Putting in the mean is safer.

There were also many outliers in the data that hinted at data corruption. Sometimes the average speed was over 90mph. I simply picked a cut off point (35mph) and any value over that was also set to the mean.

The code for finding the average of a time series is in the file: findaverage.py

There were two approaches to clean up nan’s. I could find the average and manually clean up the file or just add code to any file that loaded in the time series. I started with the former and then added a piece of code to identification.py and estimation.py to achieve the later:

Identification

For every segment, I captured 4 different plots of the segments with 0, 1, and 2 differences to aid in deciding which model to use. The plots are: The time series itself, the ACF of the time series (both full and only to lag50), and the PACF of the series to lag50. I also used the Dickey Fuller test to determine if plots were stationary.

After another recitation, I learned to only plot the ACF and PACF's for the first day of plots so the plots made more sense. So, I plotted those for each segment for differencing of 0 and 1.

The plots can all be found in the plots folder.

The code for identification can be found in identification.py

Segment 500004_500005

I determined this most resembled the nature of the ACF: **Decay, starting after a few lags Mixed AR and MA model**

Also, I ran the Dickey-Fuller test and determined no differencing was needed. For p and q, I needed to use the ACF and PACF plots. Using the 0-Diff plots, p would need to be around 28 and q would be 12. These parameters were impossible to run with ARIMA and I had to resort to testing smaller values in estimation using grid search.

The FULL ACF plot of this segment showed oscillation in and out of the confidence interval and also in and out of negative values. The ACF was in fact decaying after a few lags but it was repeating this CYCLE over and over for thousands of lags. This worried me as this did not fit any model in the box jenkins choices well. I then tried to seasonal difference. I used a seasonal order of 144 and ran SARIMAX on the first 1200 data points in this interval to observe what happened. I used grid search on many variations of pdq and PDQ. In results/500004_500005/seasonal_results.txt you can find all AIC values for all parameters.

As you can see, [ARIMA(0, 0, 0)x(0, 1, 1, 144)144 - AIC:4226.854773062325] was the best parameters. However, these scores were still not much better than just running ARIMA so I abandoned this approach.

Going back to ARIMA, I then ran a grid search on a prediction train/test set for about 16 hours on a smaller sample (3074 of 3 weeks) of this dataset choosing these values for p,d,q:

```
p_values = [0, 1, 2, 4, 6, 8, 10, 12]
d_values = [0,1,2]
q_values = [0,1,2]
```

I calculated the lowest MSE of the predicted results. These were the results for these sample sizes:

```
#Best for 500004_500005:288 - (2,1,1)
#Best for 500004_500005:576 - (0,0,2)
#Best for 500004_500005:3074 - (8,0,1)
```

As you can see, there wasn't a very clear answer for best ARIMA so I had to go back to the drawing board for best p,d,q. However, identification phase was relatively over. I learned that no differencing was needed (for larger samples) and an ARIMA model was a good choice as the best orders included a p value and a q value, meaning both AR and MA were needed. In the estimation page you can see how I honed in on the best values for p,d,q.

NOTE: FULL plots - Alternating positive and negative, decaying to zero: AR model. Use the PACF plot to identify the order. Is a possible interpretation of this ACF, HOWEVER, grid search runs ARIMA tests with only values for p and this is the SAME as a simple AR function. So, grid search will also cover this possibility.

Segment 500008_500009

In the plots folder, you can see the ACF, PACF, and actual plots for this segment with differencing of 0,1, and 2.

Using Dickey Fuller test, no difference was needed. So, I used the 0-Diff plots to determine the best model to use. Looking at ACF-Lag50, **Decay, starting after a few lags Mixed AR and MA model** seemed to be correct. I would now need to determine p and q values from the plots (no d since no differencing). P would need to be around 30 and Q would need to be around 13. These parameters were impossible to run with ARIMA and I had to resort to testing smaller values in estimation using grid search.

NOTE: FULL plots - Alternating positive and negative, decaying to zero: AR model. Use the PACF plot to identify the order. Is a possible interpretation of this ACF, HOWEVER, grid search runs ARIMA tests with only values for p and this is the SAME as a simple AR function. So, grid search will also cover this possibility.

Segment 500081_505233

In the plots folder, you can see the ACF, PACF, and actual plots for this segment with differencing of 0,1, and 2.

Using Dickey Fuller test, **ONE round** of differencing was recommended. So, I used the 1-Diff plots to determine the best model to use. Looking at 1-Diff-ACF-Lag50, I determined p to be 2 (first non significant value) and q would be in the realm of 30 (from the PACF). These parameters were impossible to run with ARIMA and I had to resort to testing smaller values in estimation using grid search.

NOTE: FULL plots - Alternating positive and negative, decaying to zero: AR model. Use the PACF plot to identify the order. Is a possible interpretation of this ACF, HOWEVER, grid search runs ARIMA tests with only values for p and this is the SAME as a simple AR function. So, grid search will also cover this possibility.

Segment 505233_501414

In the plots folder, you can see the ACF, PACF, and actual plots for this segment with differencing of 0,1, and 2.

Using Dickey Fuller test, no difference was needed. So, I used the 0-Diff plots to determine the best model to use. Looking at ACF-Lag50, **Decay, starting after a few lags Mixed AR and MA model** seemed to be correct. I would now need to determine p and q values from the plots (no d since no differencing). P would need to be around 30 and Q would need to be around 13. These parameters were impossible to run with ARIMA and I had to resort to testing smaller values in estimation using grid search.

NOTE: FULL plots - Alternating positive and negative, decaying to zero: AR model. Use the PACF plot to identify the order. Is a possible interpretation of this ACF, HOWEVER, grid search runs ARIMA tests with only values for p and this is the SAME as a simple AR function. So, grid search will also cover this possibility.

Segment 505238_503996

In the plots folder, you can see the ACF, PACF, and actual plots for this segment with differencing of 0,1, and 2.

Using Dickey Fuller test, no difference was needed. So, I used the 0-Diff plots to determine the best model to use. Looking at ACF-Lag50, **All zero or close to zero Data is essentially random** seemed to be correct. Looks like no model can be used on this, but checking ARIMA wouldn't be harmful. Again, ARIMA grid search using 0 for d and 0 for q is the same as using AR.

I can't determine any p,d,q's from a random model so I would use grid search in the estimation phase.

Estimation

Code - estimation.py

Segment 500004_500005

From my work above, I learned that no differencing was needed as grid search returned the worst values for any values of d that were not 0. So, I was ready to grid search once again. I planned on using mode_fit.summary() to find indicators of best performance for a massive

permutation of p,d,q values. This is similar to how I grid searched for SARIMAX. I used the following ranges for p,d, and q:

```
p_values = [0,1,2,4,6,8,10]
d_values = [0,1]
q_values = [0,1,2]
```

I checked for d=1 just as a sanity check. I was already confident no differencing was needed but I wanted to at least try with grid search with d=1 for every combination of p and q. The results for grid search are in results/500004_500005/arima_gridsearch.txt The following was the best order with the lowest AIC:
Best ARIMA(6, 0, 2) AIC=47570.535

I took the top 2 best models (from AIC) and then did a train/test prediction split where test was the final 144 data points (the 90th day). The model with the best MSE was the best ARIMA model for this dataset:

1. ARIMA(6, 0, 2) - AIC=47570.535
2. ARIMA(4, 0, 2) - AIC:47570.83052508612

Results of train/test:

1. ARIMA(6, 0, 2) - Test MSE: 2.462
2. ARIMA(4, 0, 2) - FAILED - raise LinAlgError("SVD did not converge"), numpy.linalg.linalg.LinAlgError: SVD did not converge
3. (TEST) ARIMA(4,1,0) - Test MSE: 2.677

The prediction plot is in plots/50004_500005/evaluation

Segment 500008_500009

The results for grid search are in results/500008_500009/arima_gridsearch.txt

```
p_values = [0,1,2,4,6,8,10,12]
d_values = [0,1]
q_values = [0,1,2]
```

Best ARIMA(8, 0, 2) AIC=49185.252

I took the top 2 best models (from AIC) and then did a train/test prediction split where test was the final 144 data points (the 90th day). The model with the best MSE was the best ARIMA model for this dataset:

1. ARIMA(8, 0, 2) AIC=49185.252
2. ARIMA(12, 0, 0) - AIC:49192.73081303116

Results of train/test:

1. ARIMA(12, 0, 0) - Test MSE: 5.781
2. ARIMA(12, 0, 1) - Test MSE: 5.782

3. ARIMA(8, 0, 2) - FAILED - raise LinAlgError("SVD did not converge"), numpy.linalg.linalg.LinAlgError: SVD did not converge

The prediction plot is in plots/50008_500009/evaluation

Segment 500081_505233

The results for grid search are in results/500081_505233/arima_gridsearch.txt

```
p_values = [0,1,2,4,6,8,10,12]
d_values = [0,1]
q_values = [0,1,2]
```

Best ARIMA(12, 0, 1) AIC=44595.888

I took the top 2 best models (from AIC) and then did a train/test prediction split where test was the final 144 data points (the 90th day). The model with the best MSE was the best ARIMA model for this dataset:

1. ARIMA(12, 0, 1) AIC=44595.888
2. ARIMA(12, 0, 2) - AIC:44596.71636712272

Results of train/test:

1. ARIMA(12, 0, 2) - Test MSE: 2.905
2. ARIMA(12, 0, 1) - Test MSE: 2.906

The prediction plot is in plots/50081_505233/evaluation

Segment 505233_501414

The results for grid search are in results/505233_501414/arima_gridsearch.txt

```
p_values = [0,1,2,4,6,8,10,12]
d_values = [0,1]
q_values = [0,1,2]
```

Best ARIMA(2, 0, 2) AIC=55441.996

I took the top 2 best models (from AIC) and then did a train/test prediction split where test was the final 144 data points (the 90th day). The model with the best MSE was the best ARIMA model for this dataset:

1. ARIMA(2, 0, 2) AIC=55441.996
2. ARIMA(10, 0, 1) - AIC:56080.18535078779

Results of train/test:

1. ARIMA(10, 0, 1) - Test MSE: 3.496
2. ARIMA(12, 0, 1) - Test MSE: 3.496
3. ARIMA(2, 0, 2) - FAILED - raise LinAlgError("SVD did not converge") numpy.linalg.linalg.LinAlgError: SVD did not converge

The prediction plot is in plots/505233_501414/evaluation

Segment 505238_503996

The results for grid search are in results/505238_503996/arima_gridsearch.txt

```
p_values = [0,1,2,4,6,8,10,12]
d_values = [0,1]
q_values = [0,1,2]
```

Best ARIMA(10, 0, 2) AIC=71365.697

I took the top 2 best models (from AIC) and then did a train/test prediction split where test was the final 144 data points (the 90th day). The model with the best MSE was the best ARIMA model for this dataset:

1. ARIMA(10, 0, 2) AIC=71365.697
2. ARIMA(12, 0, 0) - AIC:71367.37087796666

Results of train/test:

1. ARIMA(12, 0, 0) - Test MSE: 22.033
2. ARIMA(10, 0, 2) - Test MSE: 22.047

The prediction plot is in plots/505238_503996/evaluation

Diagnostic Checking

I tested the residuals of both the model fitted on the entire time series of the residuals of the test range in the estimation phase.

Code - model_validation.py
Code - test_validation.py

Segment 500004_500005

4-Plots for the entire model residuals found in plots/500004_500005/validation_model
4-Plots for the test results found in plots/500004_500005/validation_test

Segment 500008_500009

4-Plots for the entire model residuals found in plots/500008_500009/validation_model
4-Plots for the test results found in plots/500008_500009/validation_test

Segment 500081_505233

4-Plots for the entire model residuals found in plots/500081_505233/validation_model
4-Plots for the test results found in plots/500081_505233/validation_test

Segment 505233_501414

4-Plots for the entire model residuals found in plots/505233_501414/validation_model
4-Plots for the test results found in plots/505233_501414/validation_test

Segment 505238_503996

4-Plots for the entire model residuals found in plots/505238_503996/validation_model
4-Plots for the test results found in plots/505238_503996/validation_test

Spark Implementation

LinearRegression Spark Code - lr.py
Chi-Squared Spark Code - chi.py

In spark, I used LinearRegression and calculated MSE and compared to results of ARIMA.
In spark, I ran Chi-Squared Goodness of Fit on my residuals from ARIMA

The run commands are in the files. Simply place them in your folder in Dumbo and run using the command and correct input/output parameters.

My files are in /user/ntg251/input/___insert segment name here___.csv

Linear Regression Results:

500004_500005:
Mean Squared Error = 5.44319720161

500008_500009:
Mean Squared Error = 8.16361244575

500081_505233:
Mean Squared Error = 9.0658443869

505233_501414:
Mean Squared Error = 9.34491725266

505238_503996:
Mean Squared Error = 116.256630205

Chi-Squared Goodness of Fit Test

500004_500005:
Chi squared test summary:
method: pearson
degrees of freedom = 143
statistic = 180.51826630642293

pValue = 0.018385885065593577
Strong presumption against null hypothesis: observed follows the same distribution as expected.

500008_500009:
Chi squared test summary:
method: pearson
degrees of freedom = 143
statistic = 449.6731809594941
pValue = 0.0
Very strong presumption against null hypothesis: observed follows the same distribution as expected..

500081_505233:
Chi squared test summary:
method: pearson
degrees of freedom = 143
statistic = 213.3646035826261
pValue = 1.2453348979013512E-4
Very strong presumption against null hypothesis: observed follows the same distribution as expected..

505233_501414:
Chi squared test summary:
method: pearson
degrees of freedom = 143
statistic = 352.28232968207465
pValue = 0.0
Very strong presumption against null hypothesis: observed follows the same distribution as expected..

505238_503996:
Chi squared test summary:
method: pearson
degrees of freedom = 143
statistic = 830.8820866561031
pValue = 0.0
Very strong presumption against null hypothesis: observed follows the same distribution as expected..

Project Results

Basic Workflow:

1. Find the best 5 segments
2. Read all values for that segment
3. Replace nan values with the time series mean
4. Use Dickey Fuller test to test for stationality
5. Plot the ACF and PACF for each segment for differencing of 0,1,2
    a. Plot the ACF and PACF for each segment for just first day
6. Find the ballpark of p,d,q from these plots and then grid search for combinations of ranges of values
    a. Find the best orders by comparing AIC values
7. Create a train/test estimation pipeline to test the ARIMA models and calculate MSE
    a. Due to speed constraints, only predict the last day (144 values)
    b. Capture residuals from these predicted values
8. Plot the 4-Plots using residuals and look for how good the model performed
    a. Compare these results to 4-plots of residuals of the model fit to entire time series
9. Use Spark and LinearRegression to create a train/test model
    a. Compare the MSE scores of linear regression vs ARIMA
10. Use Spark to perform ChiSquared Goodness of Fit test on the residuals

Results

I believe the results of my ARIMA models on these time series was poor. The main reason is that the datasets are seasonal but my attempts at using SARIMAX were just as poor. I would need to use a seasonal value of 144*7 (one week ago from each value) to use in the ARIMA model. This was impossible for speed reasons and also using HPC was not viable as there are no reliable ARIMA or SARIMAX functions. I had to resort to using poor p,d,q values and use ARIMA locally for my Box Jenkins methodology phase. If you look at the predicted plot superimposed over the actual values, you can see the model not ‘try very hard’ to predict the values. I believe this is because the data is so noisy and the best model ‘played it safe’ and did not deviate very far from the mean. If I had more time and better computing power, I would try and use SARIMAX with a s-value of 144*7. Another improvement would be to lower the granularity of the predictions and just average the speed for a single day and then use a SARIMAX model with a s-value of 7.

ARIMA outperformed LinearRegression as expected. The MSE’s were much much lower and ARIMA also had less ‘wiggle room’ with predictions (only 144). This shows that a more sophisticated forecasting tool is needed to predict speeds in a crowded city.

The last segment (505238_503996) was the noisiest and this showed in the results of both ARIMA and Linear Regression. Running Box-Jenkins methodology pointed to ‘random data.’

References

For AIC GridSearch -
https://www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting-with-arima-in-python-3

For MSE GridSearch -
https://machinelearningmastery.com/grid-search-arima-hyperparameters-with-python/

For Basic ARIMA setup -
https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

For BoxJenkins methodoloy -
http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc445.htm

For DickeyFuller Stationality Test -
https://machinelearningmastery.com/time-series-data-stationary-python/