Progress 1

Noah Anderson

2023-11-27

Analysis

The StemGNN model, a product of Microsoft, was applied in forecasting the shortage of bicycles at numerous Divvy Stations in Chicago. The study utilized data from the Chicago Data Portal (City of Chicago, 2023), specifically covering the period from March to September 2022. This dataset provided a detailed historical account of the statuses of Divvy stations, including the availability and total number of docks at each station, recorded at 10-minute intervals. To streamline the model tuning and exploratory analysis, the research focused on a selection of neighborhoods: Lake View, Lincoln Park, Uptown, and North Center. These neighborhoods comprise 49 bike stations, although data for 8 stations were missing due to a download error. Despite this, the data from the remaining 41 stations was deemed sufficient for preliminary analysis. The data was reformatted into a T*N matrix, with 'T' representing time stamps and 'N' representing nodes, each entry reflecting the bicycle deficit at a particular node and time.

The StemGNN model integrates Discrete Fourier Transform (DFT) and Graph Fourier Transform (GFT) methodologies (Cao et al., 2020). DFT is utilized to model temporal dependencies, identifying patterns such as seasonality and autocorrelation. GFT, on the other hand, is employed to analyze interseries correlations by examining spatial interactions between nodes.

This research focused on fine-tuning three main parameters of the StemGNN model: learning rate, window size, and forecast horizon, while other settings were kept default. The learning rate influences the model's learning speed, the forecast horizon determines the prediction range, and the window size specifies the quantity of past data points used for forecasting. The study tested horizons of 3 and 6 (equating to 30 and 60 minutes into the future), learning rates of 0.01, 0.001, and 0.0001, and window sizes of 6, 12, 24, and 30, culminating in 24 unique model configurations.

Evaluation metrics included MAPE (Mean Absolute Percentage Error) and RMSE (Root Mean Square Error). For a 3-horizon forecast, the optimal model had a learning rate of 0.001 and a window size of 12, yielding a MAPE of 0.47 and an RMSE of 1.12. For the 6-horizon, the best model also had a learning rate of 0.001 but with a window size of 6. Notably, higher window sizes generally led to less accurate predictions, suggesting that analyzing more distant past data did not enhance forecast accuracy. This trend, however, was not as pronounced for the 6-horizon with a learning rate of 0.001. The RMSE scores were closely clustered, with a marginal difference of 0.04 bikes between the least and most accurate models. The slight increase in RMSE for the 6-window size (1.14) compared to the 12-window size (1.12) is offset by the doubled training time for the larger window. Consequently, a window size of 6 is preferred for its efficiency and relatively high accuracy.

Nevertheless, the best MAPE achieved was 0.45, indicating modest forecasting ability, albeit better than random guessing. This limitation might stem from the scope of the data used. Expanding the temporal window or incorporating more nodes, given the model's emphasis on spatial correlations, might enhance accuracy. However, the preference for smaller window sizes in this study suggests limited benefit from a broader temporal scope.

This research aims to establish a foundational understanding of optimal StemGNN parameters for future Divvy bike station models. This groundwork is vital, especially when considering larger datasets, which will inevitably increase model run times. Understanding effective parameter settings at the outset can significantly improve efficiency.

Note: Efforts to address the missing data issue are ongoing, though data retrieval has been challenging. Time constraints may limit the ability to re-run multiple models. If feasible, re-evaluations may focus on the learning rate of 0.001 across various window sizes. However, a comprehensive re-run of all models is not planned. # References > Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., … & Zhang, Q. (2020).
Spectral temporal graph neural network for multivariate time-series forecasting.
Advances in neural information processing systems, 33, 17766-17778.

> City of Chicago. “Divvy Trips.” Data published by City of Chicago. Accessed on November 27, 2023. Available at: https://data.cityofchicago.org/Transportation/Divvy-Trips/fg6s-gzvg.

Code

I have not yet integerated the figures into my anlaysis, but this is the basic idea of what I want to incorporate.

```r
library(tidyverse)
```

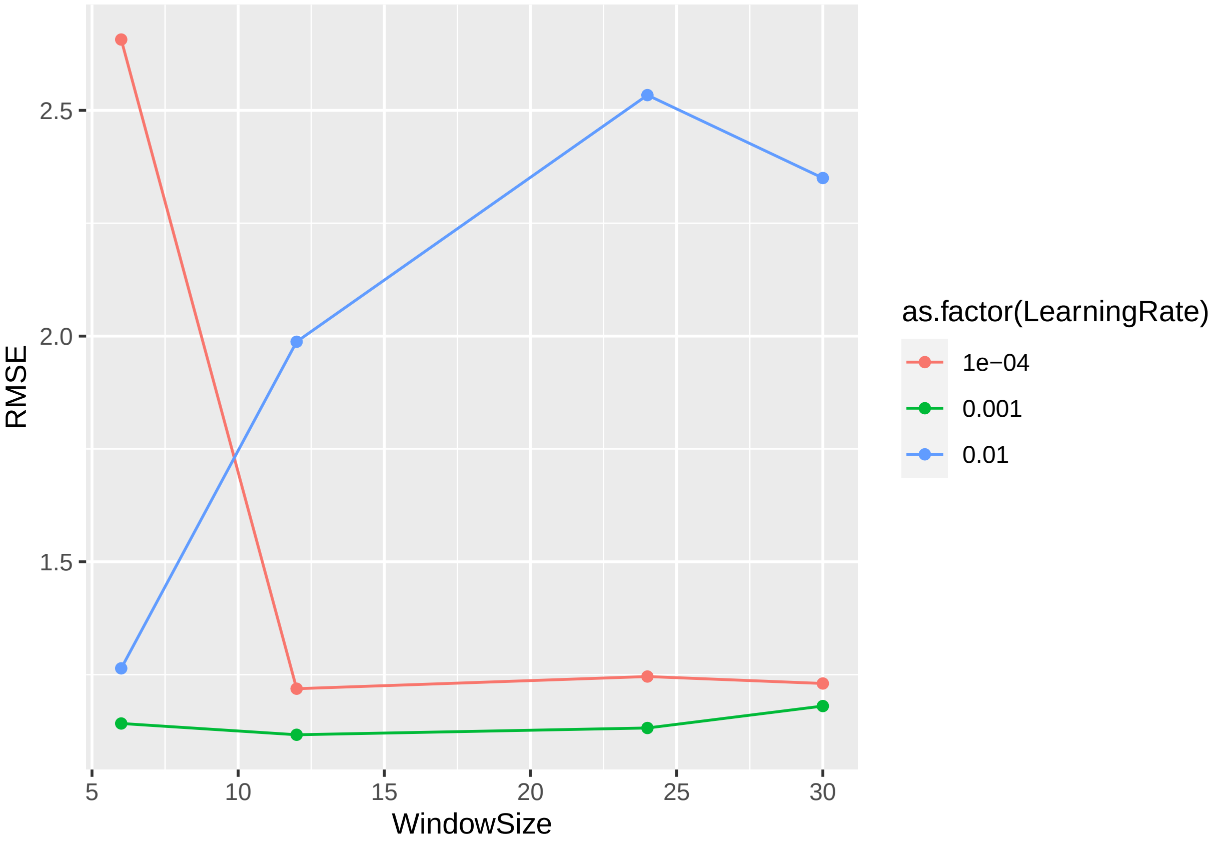
```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
metrics <- read_csv("/Users/noahanderson/Documents/GitHub/cap-stone-PDAT/src/eval/metrics/metrics_df.csv
```

```
## Rows: 24 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (6): Horizon, LearningRate, WindowSize, MAPE, MAE, RMSE
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

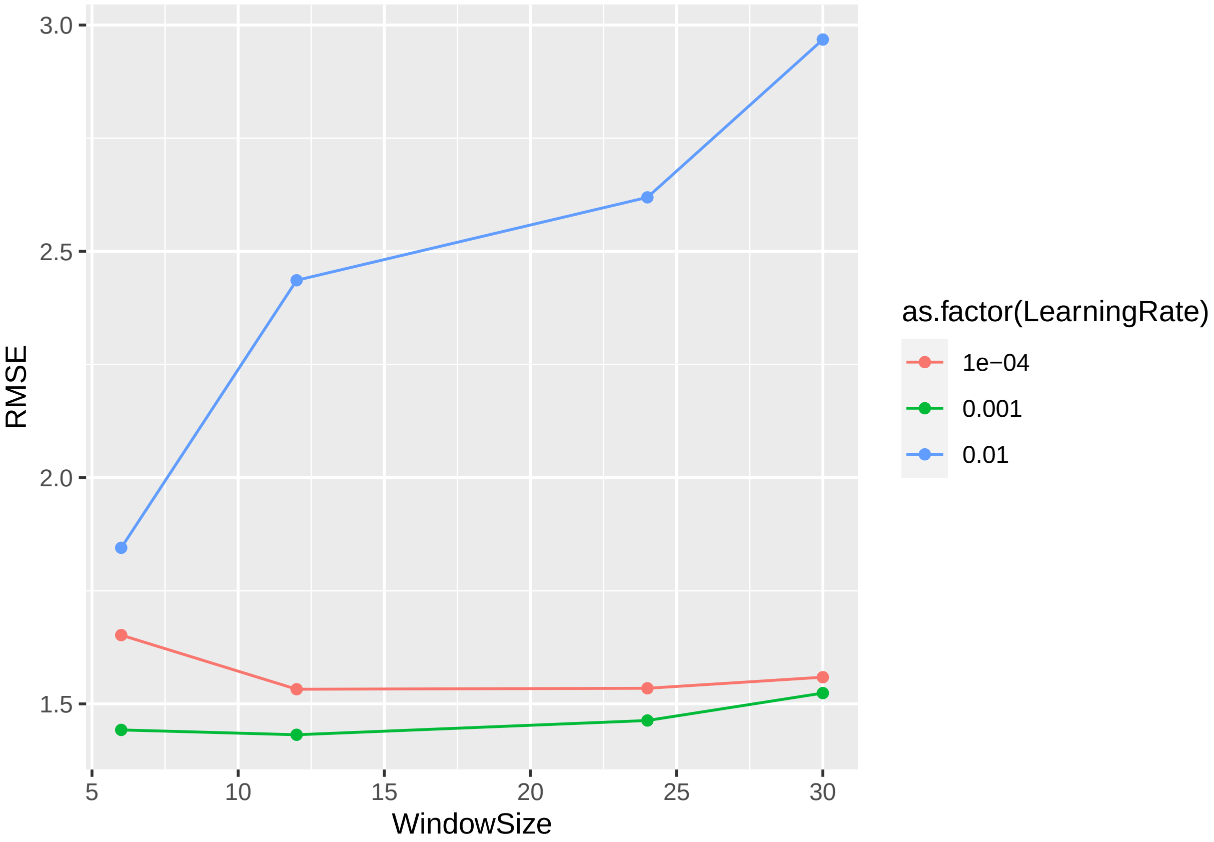
```r
horizon_3 <- metrics %>%
  filter(Horizon == 3)

horizon_3 %>%
  ggplot() +
  geom_line(aes(x = WindowSize, y = RMSE, color = as.factor(LearningRate))) +
  geom_point(aes(x = WindowSize, y = RMSE, color = as.factor(LearningRate)))
```



```r
horizon_6 <- metrics %>%
  filter(Horizon == 6)

horizon_6 %>%
  ggplot() +
  geom_line(aes(x = WindowSize, y = RMSE, color = as.factor(LearningRate))) +
  geom_point(aes(x = WindowSize, y = RMSE, color = as.factor(LearningRate)))
```


3.0
2.5
2.0
1.5
RMSE
5
10
15
20
25
30
WindowSize
as.factor(LearningRate)
1e−04
0.001
0.01