SHORT-TERM TRAFFIC FLOW PREDICTION USING kNN AND LSTM

Lydia King
University of Rochester
lking19@ur.rochester.edu

Yifei Jin
University of Rochester
yjin43@ur.rochester.edu

Oswald Garcia
University of Rochester
ogarcia3@ur.rochester.edu

Abstract

Short-term traffic flow prediction based on current traffic flow is useful for warning drivers of an ongoing or impending traffic jam in time to change routes to avoid the jam, possibly even preventing the jam from occurring. In this paper we investigate two different approaches to short-term traffic flow prediction, using kNN and LSTM models. In each case the current traffic flow, speed, time of day, and day of the week are used to predict the traffic flow 5 minutes in the future. The final test set performance achieved by the two models as measured by the mean absolute percent error (MAPE) is 8.66% for the kNN model and 11.88% for the LSTM model.

1. INTRODUCTION

In real life, with the acceleration of urbanization and the rapid increase in population, traffic problem is gradually becoming a very headache-inducing issue for citizens. However, short-term traffic flow prediction can provide real-time and accurate traffic information and road conditions so that travelers can make better traffic decisions and travel plans. In this way, it can improve the traffic transportation efficiency including buses, taxis, private cars, trucks and so on. Therefore, it is useful to city planners and designers to optimize the city traffic and road design as well.

kNN is a widely used classification algorithm, however it can also be used to solve the regression problem. Instead of classifying the unclassified point to the majority class of the k nearest data points, it calculates the mean value of the k nearest data points as its prediction value. kNN has multiple benefits compared to other algorithms. First it is easy to implement, sklearn provides fully functional and comprehensive APIs for us to implement; Second, it doesn't need an explicit training process, it is flexible and convenient when dealing with datasets. Besides, according to Zhang, Lun et al. [1], the kNN prediction MAPE can reach less than 9% on the short-term traffic flow prediction task, which means the difference between the predicted value and the actual value is within 9%. For example, there are 100 cars in the next 5 minutes, the prediction values are around 91-109. The result is acceptable and useful to predict actual value. Therefore, we seek to use kNN to predict traffic flow in California.

Although a kNN may give good traffic flow prediction results, a drawback to this approach is that kNN is a lazy algorithm, requiring all data points to be stored and used at prediction time, potentially slowing the prediction. An alternative approach is to use an LSTM model, which may also give better MAPE performance due to modeling results from more of the previous points instead of only the current points. Several groups have successfully implemented traffic flow prediction using LSTM models, obtaining (respectively) MAPE of 6.49% over 15-minute prediction time [2] and 26.4% when fitting the traffic flow curve for 24 hrs [3]. Therefore, we also implement this approach.

2. METHODS

We constructed the dataset used in our implementation by using the California Department of Transportation website. In doing so, we looked at different interstate highways within different districts of California. For our implementation, we chose interstate highway I10 going West in District 7. Furthermore, instead of gathering data from the entire highway we decided to look at vehicle detection sensor (VDS) 717129 which would give us data from only a section of the highway. This data contained vehicle speed (mph), flow (veh/5 mins), number of lanes (4 lanes), and the observation rate of each data point. The observation rate became critical in choosing how big we wanted the dataset to be. If the observation rate was less than 80% then it was most likely that the datapoint was artificial; however, anything over 80% meant that the datapoint was observed and true. Therefore, we looked for data points which had a 100% observation rate. We then chose data from December 1, 2022, to February 28, 2023, which gave us 25921 data points for our dataset.

For preprocessing our dataset which contains 25921 data points and features such as the date, speed, flow, # of lanes, and observation rate we added two more features and the output. The first feature added is to determine the day of the week (0 = Thursday and 6 = Wednesday). Second feature added is to determine the time of day for every 5 min interval (0-287 for every 5 minutes within a day). Being that we are predicting traffic flow, the flow column is used to create our output column. In order words, the output for the first row will be the flow of the second row, and so forth. Since we are predicting 5 minutes into the future, we simply drop the first datapoint in the flow column and set what remains in our flow column as our output. Moreover, we also drop the entire last row (datapoint 25919) since it does not have an output. In adding the three columns we decided to drop unnecessary columns such as the data, # of lanes, and observation rate since these columns will not help us for prediction.

After finalizing the dataset of 25920 data points we used "train_test_split" from sklearn.model_selection to split and shuffle the dataset into an 8:1:1 ratio meaning 80% training, 10% test, and 10% validation. Furthermore, we normalized only the features by using "StandardScaler" from sklearn.preprocessing while leaving the original output as it is.

Since sklearn provides the kNN Regressor API, therefore, there is no need to construct the kNN model on our own. We choose MAPE (Mean Absolute Prediction Error) as our evaluation metric, because it provides more intuitive results compared to metrics such as MSE, RMSE. Because whether the result of these metrics is good or bad varies depending on the dataset. The KNN implementation uses knn.neighbors.KNeighborsRegressor.

To form the dataset for the LSTM model, the original dataset (preprocessed only by defining the output for each sample as the traffic flow from the next time step) was split into segments, with the segment length L_in and overlap between segments being left as a hyperparameter. The output of each segment was defined to be the output of the last sample in the segment (the traffic flow during the next sample following the segment). The dataset was then randomly split into training, validation, and test sets with the ratio 8:1:1. Finally the features were normalized based on the training set: the mean and standard deviation of every point from every segment in the training set were computed, the mean was subtracted from all points in each of the data sets, and the results were divided by the standard deviation. Note that some of the training points will be counted multiple times during computation of mean and standard deviation if overlapping segments are assigned to the training set during the random partitioning, potentially resulting in an incorrect mean and standard deviation used to normalize all the data sets. However, this is not a major concern because the goal of this normalization was not so much to ensure a specific distribution (zero mean and standard deviation of 1) as to ensure that all the features have similar mean and standard deviation, allowing the LSTM to consider all features with roughly equal weight.

The LSTM network was implemented using the Pytorch framework and consists of 2 LSTM layers with hidden dimension of 512 neurons, followed by 1 fully connected layer with 512 neurons and ReLU activation, then a dropout layer with dropout probability p=0.1, and finally a fully connected layer with 1 neuron for the output.

3. EXPERIMENTS

The kNN has several hyperparameters, in traffic prediction problem, there are only 2 hyperparameters matters (others do not change the final prediction MAPE) the number of nearest training samples (k value) and the Lp distance (this article only tried Euclidean distance and Manhattan distance). As the validation result shown in Figure 1.and Figure 2., the best k = 21 and the best distance is Manhattan distance. The test set also uses this set of parameters (k=21, p=1) to evaluate performance. Surprisingly, the test set performs even better than the validation set, the MAPE value decreases to 8.6%

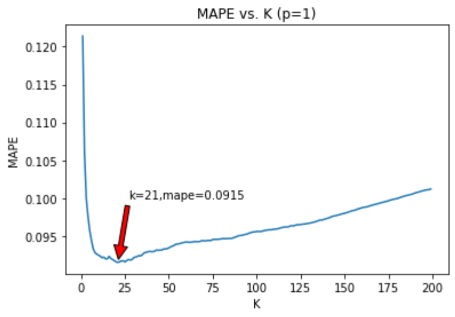

Figure 1. kNN validation set MAPE value with Manhattan distance.

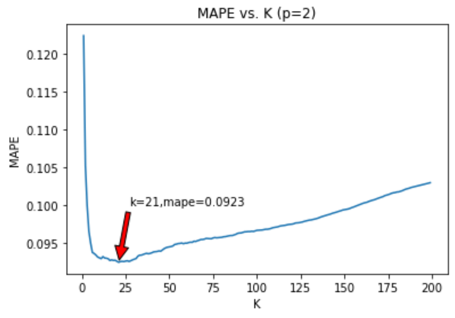

Figure 2. kNN validation set MAPE value with Euclidean distance.

The LSTM was trained using a GPU on the Google Colaboratory platform. A batch size of 32 was selected, the Adam optimizer with learning rate = 0.01 and mean squared error loss criterion was used, and the network was trained for 11000 epochs on the training set and evaluated on the validation set. After tuning the hyperparameters on the validation set, the final selection was segment length L_in =

36 with a hop size of 1 (overlap of 35) for the maximum amount of training data. Using the final network trained for 11000 epochs with these parameters, the final test set performance was MAPE = 11.88%. The MSE measured on the training and validation sets during training are plotted in Figure 3 below, showing that the training loss decreases throughout training and nearly saturates by the end of 11000 epochs, while the validation loss decreases more slowly and appears to saturate earlier, but does not begin increasing.

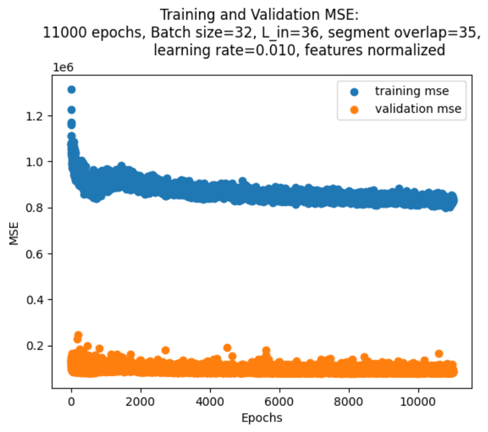

Figure 3. LSTM Training and validation set MSE during training.

4. CONCLUSIONS

In general, both LSTM and KNN perform well in predicting short-term traffic flows. LSTM has the advantage of working with sequential data and nonlinear relationships. However, the training process of LSTM is more complex and requires more computing resources and time with a little bit worse performance compared to kNN. kNN, by contrast, is simpler and easier to implement and adjust parameters. At the same time, kNN can also make use of the similarity of historical data to make the prediction result more reliable.

5. FUTURE DIRECTIONS

Comparing the test set performance of the two models, the kNN model (8.66% MAPE) performs slightly better than the LSTM (11.88% MAPE). Therefore, future improvements include adjusting the architecture of the LSTM network to make it more flexible (since overfitting was not observed during training). It may also be beneficial to increase the L_in hyperparameter to model longer dependencies, since the used value of 36 only accounts for the last 3 hours. Another idea to investigate is to keep both the kNN and LSTM models and average their outputs to obtain the final output. Finally, future directions include predicting farther into the future than 5 minutes; 15-minute or 30-minute prediction times would be more useful to drivers to reroute in time to avoid a traffic jam.

Besides, traffic flow prediction is a very complicated problem affected by other factor such as weather, major events (such as pandemic or other big social events) and so on, in the future, these factors(features) should be taken into consideration, maybe a more complex model is needed for example the combination of kNN and LSTM.

6. REFERENCES

[1] Zhang, Lun & Liu, Qiuchen & Yang, Wenchen & Nai, Wei & Dong, Decun. (2013). An Improved K-nearest Neighbor Model for Short-term Traffic Flow Prediction. Procedia - Social and Behavioral Sciences.96.653-662.10.1016/j.sbspro.2013.08.076

[2] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), 2015.

[3] Q. Chu, G. Li, R. Zhou, and Z. Ping, "Traffic flow prediction model based on LSTM with Finnish dataset," 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), 2021.

[4] "Caltrans, performance measurement system (pems)," 2023. [Online]. Available: http://pems.dot.ca.gov.

[5] "Traffic Prediction: How Machine Learning Helps Forecast Congestions and Plan Optimal Routes." AltexSoft, 27 Jan. 2022, www.altexsoft.com/blog/traffic-prediction/.