Crowd Flow Prediction on Grid Partition using Machine Learning Techniques

Yin Xiaohe
Data Science
United International College
n830026144@mail.uic.edu.cn

Wu Enyan
Data Science
United International College
n830026118@mail.uic.edu.cn

***Abstract* – With the rapid development of technology, crowd flow prediction is regarded as an critical issue. The government can control the crowd flow in a certain region if they have a good crowd flow prediction. Therefore, in the project, we decide to forecast the crowd flow of the next 15 minutes based on grid partition, different regressions have been used and compared. First, we observe the whole dataset and do preprocessing on both features and data. Then, after choosing the prediction models, we set up three models, which are KNN regression, random forest regression and gradient boost regression and compare their performance based on some evaluations. Eventually, we build up the final model to do the prediction. In our case, we have found out that the Gradient Boost regression can interpret the provided data well and do the best prediction among the three models we selected.**

***Keywords**: Crowd Flow Prediction, Machine Learning, KNN Regression, Random Forest Regression, Gradient Boost Regression*

I. Motivation and Background

Nowadays, a common issue is increasingly being considered and widely applied, which is the issue about the prediction of crowd flow in the city. With the rapid development of technology, it is available to have access to obtain data like mobile phone data and metro/bus swiping data, which is much easier to extract information to do further analysis. A good forecasting crowd distribution is essential as it can improve urban planning and traffic management. Currently, many researchers focus on spatiotemporal data by considering several main factors such as events and holidays. Besides, they also do prediction using both grid partition and flexibly complex partition.

However, in reality, especially in the rural regions (refer to those areas with relatively low popular density), due to the totally different patterns of human activities and events, it is not suitable and practical to utilize the approach used in urban to sense the current situation in such regions due to the deficiency of valuable data. Therefore, in this paper, we are more concerned about the issue of crowd flow prediction based with grid partition in rural region using baseline machine learning techniques. By using several techniques, we do comparison and evaluation among them to make prediction be effectively exploited and to figure out relevant features for the issue.

II. Related Works

We review some previous works on crowd flow prediction. Many researchers have proposed model to tackle the issues in terms of short-term and long-term. From the perspective of short-term, an improved KNN model based on spatiotemporal correlations is proposed. (Cai et al., 2016) Such statistical model can perform well in time-varying situation. Similarly, by using real-world taxi data, the authors (Xia et al., 2016) generate a Spatial-Temporal Weighted K-Nearest Neighbor (STW-KNN) model, which consider both time and space domain. For the perspective of long-term prediction, many researchers prefer to apply neural network into the model. Deep Spatio-Temporal Residual Networks (Zhang et al., 2018) is based on three CNN models in terms of period, closeness and trend. Based on this model, the researchers propose DeepSTN+ (Lin et al., 2019) by considering the effect of location attributes.

Apart from the statistical models and neural network models mentioned above, traditional machine learning techniques (Xie et al., 2020) are also considered such as

support vector regression (SVR) and Bayesian network (BN) model, which are suitable for some specific conditions.

Besides, no matter what kind of approach the researchers used, it is necessary to select the effective features. Generally, there are some main factors affecting the flow prediction, which refer to daily flow activity patterns, anomalies of flow, weather and holidays. (Xie et al., 2020)

III. Data Observation

The dataset is obtained from Zhuhai Open Data Innovation Apps Contest. Totally, there are 225,1329 tuples from raw data, we have access to the attribute of observation time, grid ID, latitude, longitude and current number of people in grid.

We first observe the exact location of grids using Gaode API. As **Figure 1** suggests that location distributes referring to red areas concentrate on the area of Meixian, Meizhou in Guangdong province and Longyan in Fujian province. There are 86 locations are being observed. By calculating the difference between latitude and longitude, the size of each grid is approximately 800 to 900 meters.

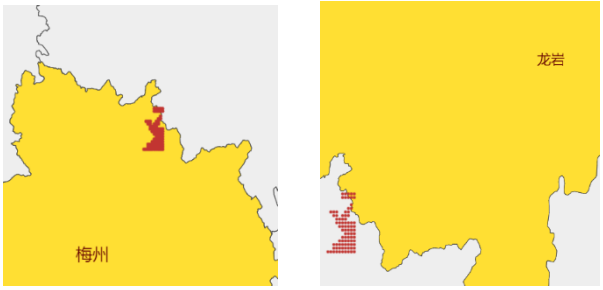

Figure 1: Location distributions

On previous research, a large number of studies focus on spatiotemporal data, which means it is essential to understand the time features. The raw data we used record the flow observation from 1st October 2019 to 30th December 2019. For every location, we can extract more than 20,000 tuples. Every 5 minutes, the number of current flows will be updated.

Generally, time feature is a periodic and cyclical feature data. Therefore, **Figure 2, 3** visualize timeseries for the data of two whole weeks (the first week: 1st October to 7th October, the second week: 8th October to 14th October). We figure out that unlike urban crowd flow, except few abnormal flows, the similar flow pattern is followed by every day in the same location no matter weekend or weekday it is.

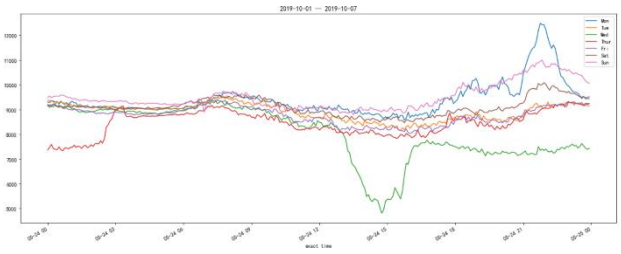

Figure 2: Crowd Observation (1st October to 7th October)

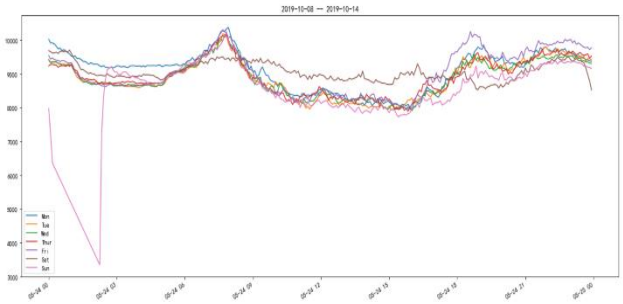

Figure 3: Crowd Observation (8th October to 14th October)

IV. Data Preprocessing

Because the dataset is big enough and has some periodical pattern based on our observation, therefore, we can get rid of the missing value. Then, we also find out that the data contain the crowd flow in the different grids and at different times. To simplify our model, we first deal with one particular area at different times. In the report, we pick up the first grid id (5004) to obtain the data we need as an example.

A. Feature Preprocessing

And now, we only have the data time, grid id, longitude, latitude, and the current crowd flow information. So, we want to find out more useful information from the data time which can help to set up a reasonable model. Here, we generate 6 categorical features derived from the data time.

The first feature is the day of the week, which tells us what day of the week it is. The attribute has 7 categories, indicating from Monday to Sunday.

And the second one is the x, y value. In the dataset, the exact time is being recorded. Such time data is cyclical as one hour has 60 minutes and one day has 24 hours. In this case, one-hot encoding might be a wise approach to handle time since high dimensionality will be generated without any connection. Therefore, we first generate an exact time in one day to combine hours with minutes.

$$hourfloat = \frac{hour}{24} + \frac{minute}{60}$$

Since we know sine and cosine graph is also periodic, then we use since and cosine transformation to store the value of exact time (hourfloat), which allows us to represent time data in a more meaningful way.

In order to make a more intuitive on time features, **Figure 4** plots 50 arbitrary samples with transformation of sine and cosine features. It makes sense to represent cyclical data with cycle.

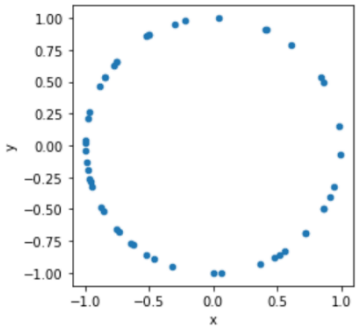

Figure 4: Random samples with sine and cosine transform

The weekend and holiday attributes are the binary attribute, where the data is True value if it is weekend or holiday, and vice versa.

The time attribute is categorical contains 4 categories. Early morning from 0 a.m. to 6 a.m., morning from 6 a.m. to 12 p.m., afternoon from 12 p.m. to 6 p.m., and evening from 6 p.m. to 12 a.m.

Besides, we also need to find out the crown flow after 15 of each data time, which is our label of the model. Due to the granularity of the data time is 5 minutes, therefore, we can directly derive the crowd flow next 15 minutes from the dataset. there are some records without the record of its later 15 minutes, we remove this kind of data since our dataset is big enough. **Figure 5** show an overview of the dataset after doing the feature preprocessing.

	grid_id	datatim	day_of_week	x	y	weekend	holiday	time	cnt	next15_cnt
0	5004	2019-10-01 00:30:00	Mon	0.130526	0.991445	False	True	early morning	9217	9203
1	5004	2019-10-03 23:00:00	Wed	-0.258819	0.965926	False	True	evening	7489	7532
2	5004	2019-10-12 12:25:00	Fri	-0.108867	-0.994056	True	False	afternoon	8399	8392
3	5004	2019-10-15 07:00:00	Mon	0.965926	-0.258819	False	False	morning	9408	9782
4	5004	2019-10-15 16:55:00	Mon	-0.960050	-0.279829	False	False	afternoon	8069	8081
...	...	...	...	...	...	...	...	...	...	...
26175	5004	2019-12-21 16:05:00	Fri	-0.876727	-0.480989	True	False	afternoon	7744	7707
26176	5004	2019-12-25 09:25:00	Tue	0.625923	-0.779884	False	False	morning	8178	8151
26177	5004	2019-12-25 15:00:00	Tue	-0.707107	-0.707107	False	False	afternoon	7501	7550
26178	5004	2019-12-28 23:00:00	Fri	-0.258819	0.965926	True	False	evening	9322	9308
26179	5004	2019-12-29 12:15:00	Sat	-0.065403	-0.997859	True	False	afternoon	8400	8351

Figure 5: Sample of the dataset after feature preprocessing

B. Data Preprocessing

In addition to the features, we also need to preprocess the data. Firstly, all the categorical variables are converted into dummy variables. Then, because the scale and value range of each attribute is quite different, we do the normalization and standardization. In order to have comparability among different features, we apply the standardization formula on the attribute, after transformation, the feature distribution does not change, and the new mean of the data is 0, and the standard deviation is 1.

$$Z = \frac{X - \mu}{\sigma}$$

And the purpose of data normalization is to make each feature have the same influence on the target variable, and after Min-Max transformation, the feature data will be scaled and changed, all data will fall in 0 to 1 interval.

$$Z = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Now, we have totally three types of data, which are original, standardized data, and normalized data, and we will do the comparison later to find out which data can improve our model prediction accuracy. And we also split our dataset into two datasets, 75% for training and 25% for testing.

V. Methodology

1) *K-nearest Neighbors Regressor*: Similar with KNN Classification, the idea of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. In the preprocessing, we obtain both continuous variables and categorical variables. For such two kinds of variables, we have to apply different distance measurements to calculate respectively.

For continuous variables, we apply Euclidean distance.

$$D_E = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

In the case of categorical variables, we utilize the Hamming distance, which is to detect the number of instances that are equivalent.

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

Therefore, we choose the optimal K value by first inspecting the data. When having K-nearest neighbors, we iterate all tuples in training dataset by computing distance based on the above equations, then we figure out the top K neighbors. Eventually, we take the average of labels value for those K items to obtain average value as predicted value.

2) *Random Forest Regression*: Random Forest is a bagging technique. Compared with individual decision trees, decision trees might be different while training

data is adjusted, which turns out different predictions. Besides, decision tress is easily having risk of overfitting. Therefore, the idea of random forest is to construct a number of decision trees while training and output the mean prediction of the individual trees. All trees in random forest are also allowed to run in parallel.

Figure 6 shows a general procedure of the algorithm. We first split training dataset into N subsets using bootstrap sampling. Then we train different decision trees for various sub-samples. We obtain prediction of each decision tree and take average of all predictions as predicted values.

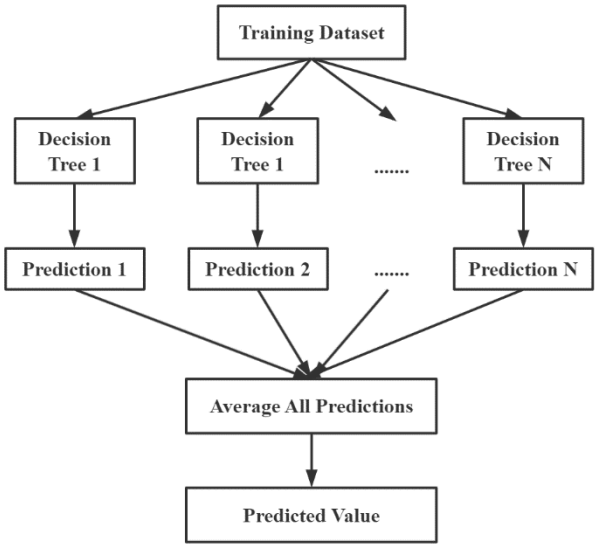

Figure 6: The Procedure of Random Forest Regression

For many datasets, random forest algorithm can generate a highly accurate classifier and run efficiently on large datasets.

3) *Gradient Boosting Regression*: Gradient boosting is a forward stage wise additive modeling. Normally, decision tree is required as a weak learner. The idea is to generate a prediction model based on an ensemble of weak prediction models, which allows for the optimization of differentiable loss functions. In this state-of-art technique, the basic thought of boosting is being followed, which aims to focus on the incorrect predicted samples in the previous iteration. In order to adjust those samples, we use the strategy of gradient descent to offer a compensation. It follows the general procedure as following.

We first have training dataset $\{x_i, y_i\}_{i=1}^{n}$, and a differentiable loss function, which can also refer to squared residue.

$$L\big(y_i, F(x_i)\big) = \frac{1}{2}(y_i - F(x_i))^2$$

$$\frac{\partial L}{\partial F(x_i)} = -(y_i - F(x_i))$$

Then, we initialize model with a constant value γ as a predicted value that minimizes the summarization function. Basically, the value for γ should be average of labels.

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$$

Suppose we make M trees, then we need to iterate M times to do adjustment. For every iteration, we compute the residue sample to fit a regression tree. After that, we take the previous prediction value into account and find value for γ that minimizes the equation above. And we do linear combination of every prediction value with learning rate. Finally, we output $F_M(x)$.

Scaling the model by the learning rate results in a small step in the right direction since it can provide better predictions with a testing dataset with lower variance.

VI. Experimental Study and Result Analysis

After model selection, we need to choose the useful feature data as the input of the model. In order to find out which feature contribute to our prediction model, firstly, draw a heat map graph (**Figure 7**) to have an overview of the correlation between every numeric feature pair.

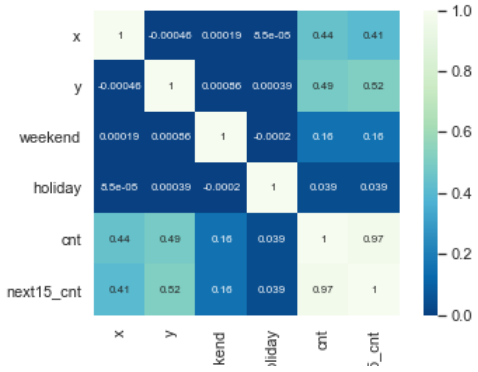

Figure 7: Heat map of every feature pair.

It is clearly seen from **Figure 7** that the 'cnt' attribute has a very strong relationship with our label 'next15_cnt', however, 'weekend' and 'holiday' show a very little relation with the crowd flow of the next 15 minutes. Therefore, we try to get rid of these two attributes and set up our model by making use of the packages in ***sklearn*** modules in python to see whether they are necessary or not. Following **Figure 7** show the comparison of 4 cases, which are setting up the model using all features, using all features except 'weekend',

using all features except holiday, and using all features except 'weekend' and 'holiday'.

Table 1

Comparison of the score of different feature selections

	All Features	Without Weekend	Without Holiday	Without Both
KNN	0.939	0.940	0.940	0.940
Random Forest	0.959	0.959	0.959	0.958
Gradient Boosting	0.960	0.961	0.961	0.961

According to **Table 1**, we can know that using all features without 'weekend' to set up our models may perform better in all three models, therefore, we decide to choose all feature data except 'weekend' attribute as input of our models. Besides, as we mentioned before, we want to compare the original data, standardized data, and normalized data. And we do the comparison of these three types of data based on three evaluations.

The score, which is similar to the R square, can reflect the relative degree of regression contribution. A larger R square value indicates that the independent variable has a higher degree of explanation for the dependent variable. The formula of calculating the R square of the regression model is following.

$$R^2 = \frac{\sum_i (\hat{y}_i - \bar{y})^2}{\sum_i (y_i - \bar{y})^2}$$

The Mean Square Error (MSE) is the expectation of the square of the difference between true and predicted values, and a large MSE value means the error of our model's prediction is bigger. The MSE value can calculate using the following formula

$$\text{MSE} = \frac{1}{n}\sum_i (\hat{y}_i - y_i)^2$$

Similar to the Mean Square Error, there is another evaluation to measure the error of the prediction, which is the Mean Absolute Error (MAE). The MAE value can calculate using the following formula

$$\text{MAE} = \frac{1}{n}|\hat{y}_i - y_i|$$

After training the models, we can obtain three regression models, which are KNN Regressor, Random Forest, and the Gradient Boost. Then, by making use of the three evaluations on the **test dataset**, we can do a comparison of the data type of the input data.

Table 2 is the evaluation value of three model results of the original data, the standardized data, and the normalized data. In terms of the score and R square evaluation, there is no big difference between these three data types in the Random Forest model and Gradient Boost, all are about 0.959 and 0.961 respectively, which means these two models are not sensitive to different types of data. In the KNN Regressor model, the score or R square value is **0.94** when using the original data, **0.956** when using the standardized data, and **0.961** in normalized data. Although the score or R square does not change a lot in these three models, it is not hard to find out that the MSE and MAE have decreased a lot from original data to standardized data and reach the minimum in normalized data. And the low MSE and MAE in normalized data might cause by the scaling of the normalization, however, to obtain a model that can do prediction with higher accuracy, we decide to use the normalized data as the input data of our models.

Table 2

Model Result Comparison of Original, Standardized and Normalized Data

	Score	MSE	MAE	R^2
Original Data				
KNN	0.940	27369.6	103.14	0.937
Random Forest	0.959	18810.19	78.55	0.957
Gradient Boosting	0.961	17596.80	82.33	0.959
Standardized Data				
KNN	0.956	27369.6	103.14	0.937
Random Forest	0.959	18810.19	78.55	0.957
Gradient Boosting	0.961	17596.80	82.33	0.959
Normalized Data				
KNN	0.959	0.00015	0.007	0.955
Random Forest	0.959	0.00014	0.0068	0.957
Gradient Boosting	0.961	0.00013	0.0072	0.959

After choosing features and data types, we can get our final model to do the prediction. The evaluation of our **final models** is shown in **Table 3.** According to the value of these evaluation, our models considered as good models to interpret the data in the training dataset and do the prediction of the crowd of next 15 on the test dataset, because the score or R square value of the three models are all about **0.96** and the Mean Square Error are about **0.0001**. We can see that the Gradient Boost performs best in the three models we selected with a **0.961** score, and **0.000135** Mean Square Error.

Therefore, we will use Gradient Boost as our model to do the prediction later.

Table 3
Evaluation of final models

	Score	MSE	MAE	R^2
KNN	0.959	0.00015	0.007	0.955
Random Forest	0.959	0.00014	0.0068	0.957
Gradient Boosting	0.961	0.00013	0.0072	0.959

In addition to numerical evaluation, we can do the visualization as well. **Figure 8** shows the relation between the true y value of our testing dataset and the predicted value of Gradient Boost. We can see that most of the points lie in the Y_pred = Y_test, which means most points have been correctly predicted.

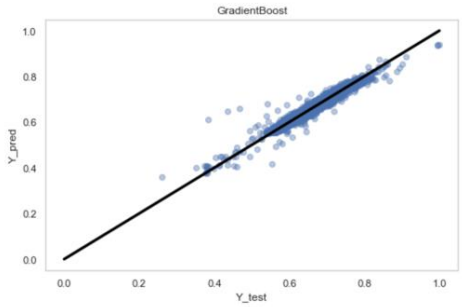

Figure 8: Result visualization of Gradient Boost

VII. CONCLUSION AND FUTURE WORK

In conclusion, in our project, data observation and preprocessing are very necessary, for instance, we need to find out the periodical pattern by data observation and data normalization or standardization by data preprocessing. Although we cannot directly conclude that data normalization can improve the performance of the models, to some extent, we still can use normalization to decrease the prediction error of the models. After comparing three regression models, we choose the Gradient Boost model as our final prediction model which gives us the best performance among the three models we selected, and it is good enough to do the prediction in general. Our prediction model can apply to many scenarios, for example, if there are some big events, our model can be used to predict the next 15 minutes of crowd flow based on the current crowd flow, therefore, the government can do some preparation in advance, to in case of some accidents caused by crowd.

Due to the time limit, we did not try the time series to build up our model, however, we will try to time series to approach the prediction and try to find out a model that can predict the crowd flow at different time and different areas.

REFERENCES

A. (2016, December 8). *amirkhango/DeepST*. GitHub. https://github.com/amirkhango/DeepST

Cai, P., Wang, Y., Lu, G., Chen, P., Ding, C., & Sun, J. (2016). A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting. *Transportation Research Part C: Emerging Technologies*, *62*, 21–34. https://doi.org/10.1016/j.trc.2015.11.002

Lin, Z., Feng, J., Lu, Z., Li, Y., & Jin, D. (2019). DeepSTN+: Context-Aware Spatial-Temporal Neural Network for Crowd Flow Prediction in Metropolis. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 1020–1027. https://doi.org/10.1609/aaai.v33i01.33011020

Xia, D., Wang, B., Li, H., Li, Y., & Zhang, Z. (2016). A distributed spatial–temporal weighted model on MapReduce for short-term traffic flow forecasting. *Neurocomputing*, *179*, 246–263. https://doi.org/10.1016/j.neucom.2015.12.013

Xie, P., Li, T., Liu, J., Du, S., Yang, X., & Zhang, J. (2020). Urban flow prediction from spatiotemporal data using machine learning: A survey. *Information Fusion*, *59*, 1–12. https://doi.org/10.1016/j.inffus.2020.01.002

Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., & Li, T. (2018). Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence*, *259*, 147–166. https://doi.org/10.1016/j.artint.2018.03.002