MDS 5210 Final Project Report

Weixiang Wang[a], Youcong Lu[a], Runkai Zheng[a], Hang Yi[a], Ziteng Weng[a]

[a]*School of Data Science, the Chinese University of Hong Kong, Shenzhen*

May, 2022

Abstract

We deployed Machine Learning algorithms to predict the final exits based on their previous trajectories. To accommodate the requirement of structured data in Machine Learning, we conducted Exploratory Data Analysis and Feature Engineering. Our Feature Engineering is motivated by Kinematics. As for the results, the F1-scores of LR, SVM, and RNN on the test set are 0.9293, 0.9301 and 0.9196, respectively.

1. Introduction

Mobile phone data are a popular source of positioning information in many recent studies that have largely improved our understanding of human mobility. These data consist of timestamps and geolocations recorded by mobile devices, and they allow for unprecedented tracking of populations of millions of individuals over long periods. Machine Learning offers a framework to analyze and identify the pattern of these data. However, the unstructured data is not applicable for Machine Learning algorithms.

In this project, we conducted Exploratory Data Analysis, Feature Engineering, and then applied Logistic Regression, Support Vector Machine, and Recurrent Neural Network to predict whether the last position of each device is in the city center.
[1]

2. Exploratory Data Analysis

In the EDA section, we used five visualization methods to explore the features in the data, including trajectory exploration, distribution exploration, clustering, mobility heat-map and correlation between distance and position.

2.1. Trajectory Exploration

In this part, we explored different trajectories among data samples. For the data that ended in the city center, we classified them into two types: starting from the city center or starting from outside. From Figure 1, those trajectories starting from the outside usually show a clear track. Therefore, we can extract their features with Kinematics. On the other hand, as for the other kind, there is no clear track. However, the destination is fairly close to the last entry and the second last entry, so these two entries are also extracted as our features.

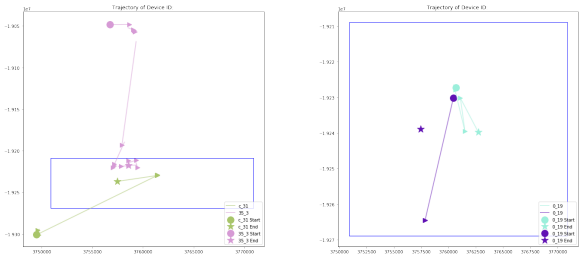


Figure 1: Trajectories examples

2.2. Trajectory Distribution

Based on Figure 2, the trajectories in the data set is not time-balanced. The number of trajectories grows as time passes and reach the peak at 15:00.

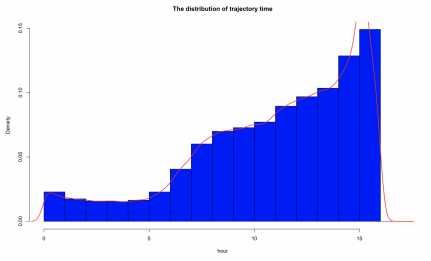


Figure 2: Time Distribution

2.3. Trajectory Distribution

As for the distribution of location, the highest density part locates at the city center as expected.

[1] Weixiang Wang, Youcong Lu and Hang Yi are responsible for Exploratory Data Analysis and Feature Engineering. Ziteng Weng is responsible for Logistic Regression and Support Vector Machine. Runkai Zheng is responsible for Recurrent Neural Network.

What is noticeable is that there is an colored cluster around the edges which may lead to miss classification eventually.

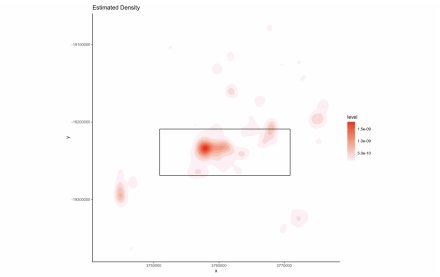


Figure 3: Density

2.4. Trajectory Clustering

To achieve high efficiency, we used shuffle techniques to randomly choose 10000 data for clustering. After deriving the chosen data, we do min-max normalization on such data. Then, we used K-means for clustering and K-means++ for randomly picking the initial center points. Based on that, we used GridSearch method to pick the best number of clusters, and the result was 9. The outcome is as follows.

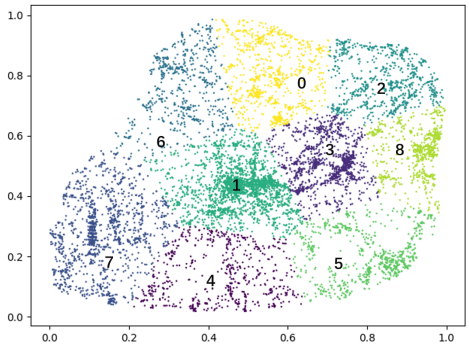


Figure 4: 9 Cluster Outcome

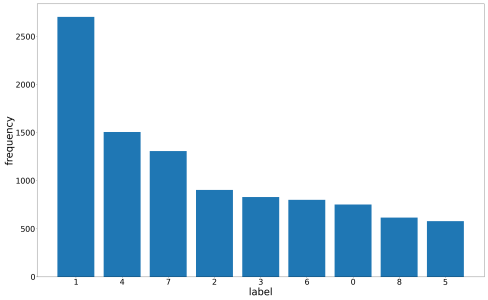
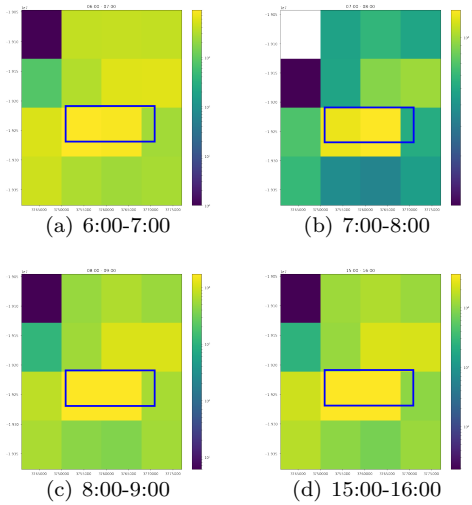


Figure 5: 9 Cluster Outcome

Then we can see that Cluster 1 is visited the most. After inverse normalization, we see its center is [3748053.625569816, -19236427.86188999].

2.5. Mobility Heat-map

Based on the heat-map, it is obvious that, the mobility outside the city center changes seriously from 6:00 a.m. to 9:00 a.m. Then the mobility stays almost unchanged after 9:00 a.m..



(a) 6:00-7:00 (b) 7:00-8:00

(c) 8:00-9:00 (d) 15:00-16:00

Figure 6: Mobility

2.6. Correlation

With figure 7, it shows that there's a strong positive correlation between the variable "if the entry in the city centre" and "if the exit in the city centre". Also, there is a negative correlation between the distance to the center and whether it is in city centre.

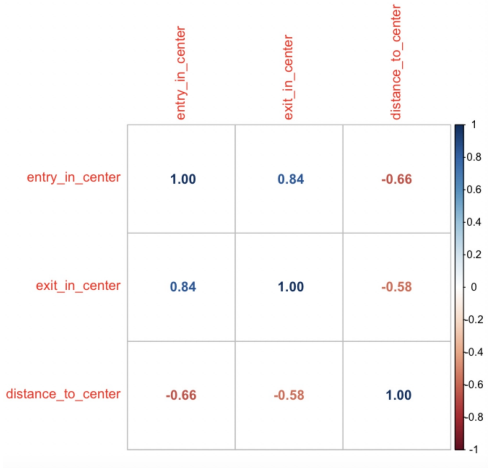


Figure 7: Correlation plot

3. Feature Engineering

Motivated by Kinematics, we consider that the exit position is determined by the entry position, the velocity, the time and the direction. Therefore, we utilized the motion information in the data.

3.1. Velocity

Due to the large number of missing values in velocity data, we didn't use these three features. Instead, we estimated the average velocity by

$$\bar{v} = \frac{\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} t_i}$$

where s_i and t_i are the length and time of the i-th trajectory.

3.2. direction

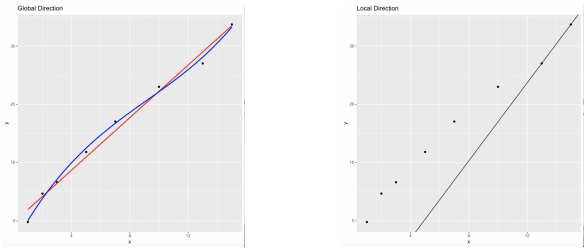


Figure 8: Global and Local Direction

The global direction was calculated by fitting a linear or polynomial regression; the local direction was the line connecting the last two available points.

3.3. Final features

After many trials on different feature combinations, we finally determined the following 6 features.

1. Last_Entry_In_Center(LEIC): For each citizen, this feature illustrates whether the last entry is in the city center or not, and this is a dummy variable.
2. Second_Last_Entry_In(SLEI): For each citizen, this feature illustrates whether the second last entry is in the city center or not, and this is a dummy variable.
3. Last_Traj_Time(LTT): For each citizen, this feature illustrates the duration of the last trajectory.

$$LTT = Time_{[last\_exit]} - Time_{[last\_entry]}$$

4. Closer_Rate(CR): For each citizen, this feature illustrates the proportion of the number of trajectories of a single citizen moving closer to the center except for the last trajectory. When this rate is comparatively high, we tend to consider that the destination of this citizen is more likely to be in the center.

$$Closer = I_{[d(center,exit) \leq d(center,entry)]}$$

$$CR = \frac{1}{n-1} \sum_{i=1}^{n-1} Closer_i$$

5. Local_Prediction(LP) & Global_Prediction(GP): With the direction, the velocity and the time, we can tentatively predict the position of the final exit. The prediction is given by,

$$\hat{x}_{exit,n} = x_{entry,n} + v_n t_n \cos\theta_n$$
$$\hat{y}_{exit,n} = y_{entry,n} + v_n t_n \sin\theta_n$$

where θ_n is the estimated direction, which could be either the global estimate or the local estimate.

Then, we used dummy variables to indicate whether these tentative predictions are in the city center or not.

4. Logistic Regression (LR) and Support Vector Machine (SVM)

4.1. Kernel Methods

In this project, the kernel methods we used are the radial basis function kernel (RBF), the polynomial kernel with degree 1 and the polynomial kernel with degree 2.

Because the number of devices is quite large, using the full kernel feature map is inefficient and impractical. Therefore, we used the kernel approximation methods to project the data into some approximated low dimensional space, and this will reduce the memories we used for the feature maps. For the RBF kernel and the polynomial kernel, we use *RBFSampler* function (via random Fourier) and *PolynomialCountSketch* function (via Tensor Sketch) respectively provided by *sklearn*.

4.2. Regularization Techniques

We used ℓ-1 regularization and ℓ-2 regularization for both LR and SVM.

4.3. Learning Algorithm

To speed up the training process, we called *GDClassifier* (also provided by *sklearn* to use stochastic gradient descent for learning.

4.4. Variable Selection

In this project, we used three feature sets.

1. LEIC, LTI and CR.
2. LEIC, SLEI, LTI, CR, LP and GP.
3. Principal component analysis on the second feature sets with number of components = 4.

4.5. Cross Validation with $K = 5$

The first purpose of the cross validation is to use the validation error to approximate the test error, which is used to evaluate models' performances.

Moreover, it can also be used for hyperparameter tuning. The most important hyperparameters in these models are the kernel coefficient γ and the regularization strength λ. For γ, we chose the scale mode which equals to $\frac{1}{\#features \times X.var()}$ where X is the training data. As for λ, the candidates are [0.000001, 0.000018, 0.00032, 0.0056, 0.1] and use validation errors to select the best λ for each combination of the feature sets, the kernel methods and the regularization techniques. The example of hyperparameter tuning is in Figure 9.

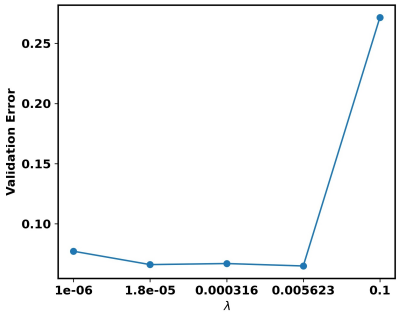


Figure 9: The example of hyperparameter tuning

We used LR and the first feature set in the above example. Moreover, polynomial kernel with degree 2 and ℓ-2 regularization were selected. According to the above figure, 0.005623 is the best λ for this combination. After applying this logic to each combination, we had the best λ for these combinations.

4.6. Result of LR

We implemented LR on each combination, and the validation errors of all combinations with corresponding best λ are shown in Figure 10.

According to this figure, the RBF kernel has the best performance among the proposed kernel methods. As for the regularization technique, ℓ-2 regularization is better. Moreover, PCA leads to worse performance.

4.7. Result of SVM

The result of SVM is in Figure 11.

According to Figure 11, the RBF kernel also has the best performance among the proposed kernel methods. As for the regularization technique, ℓ-1 regularization is better.

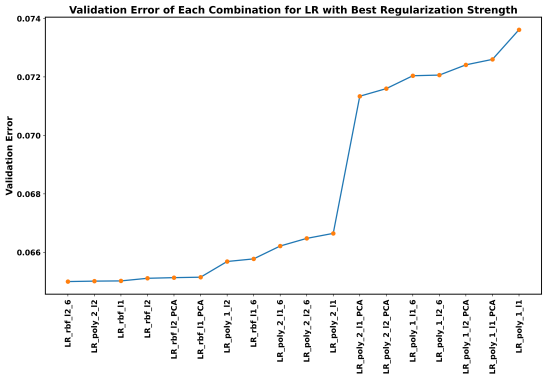


Figure 10: The result of LR

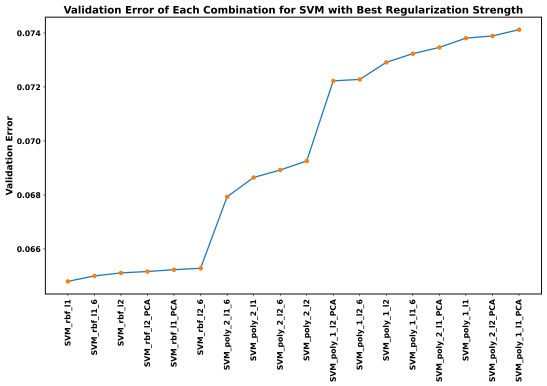


Figure 11: The result of SVM

4.8. Comparison between LR and SVM

Based on Figure 10 and Figure 11, although LR has a better averaged performance, the best performance of SVM is better than the best performance of LR.

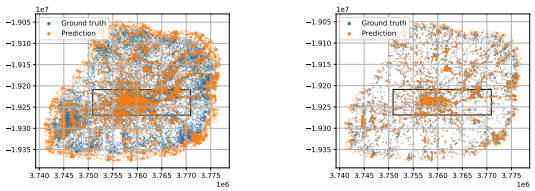
5. RNN

5.1. Overall Structure

In this part, we study the model performance of recurrent neural network (RNN) on the raw features. The overview of RNN structure is shown in figure 13. The input of RNN is a sample citizen, which contains multiple trajectories. For each trajectory, there is an entry point and an exit point. We use the raw features, i.e., the time, velocity and location information of the entry point in the current step as the input, and the output will be the current hidden state. The hidden state of each step is then feeded into a two layers fully-connected network to do the prediction for the exit point.

5.2. Classification or Regression

Note that the problem can be either formulated as a classification task or a regression task. For example, we can output a binary prediction indicating



(a) MSE (b) ℓ_1

Figure 12: This figure visualize the ground truth location and the predicted location of the training data using different loss function.

that the given sample is in the city center or not, as the formulation below:

$$\hat{\boldsymbol{z}}^{(t)} = f(\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}), \quad \hat{y}^{(t)} = \text{argmax}_i \, \hat{\boldsymbol{z}}_i^{(t)},$$

where f denotes a single step RNN function, $\boldsymbol{h}^{(t-1)}$ is the last step hidden state and $\boldsymbol{x}^{(t)}, \boldsymbol{z}^{(t)}, \hat{y}^{(t)}$ are the input, output and prediction at time step t respectively. Alternatively, we can output a predictions of the coordinate, and do the judgement based on the range of the city center:

$$\hat{\boldsymbol{z}}^{(t)} = f(\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}), \quad \hat{y}^{(t)} = \mathbb{1}_{\mathcal{C}}(\hat{\boldsymbol{z}}^{(t)}),$$
$$\mathcal{C} = \{\boldsymbol{z} : \boldsymbol{z} \text{ is in the city center}\}.$$

We conduct experiments on both models. In the classification task, we use cross entropy as the loss function, while in the regression task, we mininmize the mean squared error (MSE). The validation F1 score for classification is 73.36% and that of regression is 90.72%, which performs much better than the former. We argue that this is because the coordinate information is a sufficient and necessary condition for judging whether it is in the city center or not. Using regression plus the condition is a better way to combine with the prior information we have. If we do just classification, we do not provide any prior information about the city center to the model.

5.3. Loss Normalization

In the above mentioned model, we calculate one loss value for each time step. This can lead to a severe imbalanced loss problem among different citizen samples. Because in practice, different citizens may have different number of trajectories, citizens with more trajectories will have larger loss value, hence some sample citizens with less trajectories will lack supervision. To eliminate this problem, we normalize the loss value using the length of each sample before reduction. The normalization raise the F1 score from 90.72% to 91.38%.

	Batch size	#Hidden dim	#Epochs	Val F1
Before	256	64	50	92.31%
After	1024	256	300	**92.46%**

Table 1: This table list the hyperparameters used for RNN and the corresponding validation F1 score.

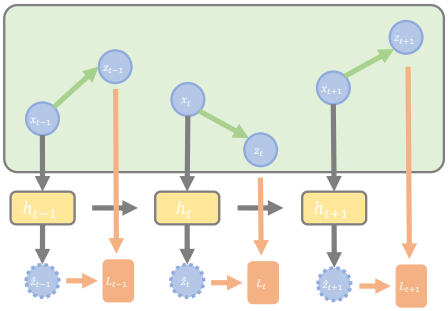


Figure 13: This figure shows the overall structure of the RNN used for trajectory prediction.

5.4. Choice of Loss Function

To study further on the model behavior, we visualize the model predictions compared with ground truth on the map. As shown in the figure 12, we found the model hardly capture the precise position of most of the points. We argue that this is because the MSE loss is too hard for the model to minimize, due to the unavoidable existence of the outliers. In order to handle this, we switch the loss function to ℓ_1-based loss, e.g., Huber loss and ℓ_1 loss with sub-gradient. Their F1 scores are 91.77% and 92.31%, improved by 0.41%, 0.93% over MSE respectively. As can be seen in figure 12, ℓ_1 loss help the model focus more on the in-distribution points and the predictions on positions become much more accurate.

5.5. Solving Underfitting

We observed that the training score is close to the validation score, which indicates that the model still can not fit the training data well. Hence, the next step is to improve the model capacity and train it longer. We list the hyperparameter used and the validation score before and after adjustment in table 1, the results show that optimizing the network gains considerable improvement.

6. Conclusion

In this project, we completed the data analysis process. We then trained three Machine Learning models: Logistic Regression, SVM and RNN.

As for the result, the F1-scores of LR, SVM, and RNN on the test set are 0.9293, 0.9301 and 0.9196, respectively.