AML Project Final Report

Panda Xu px48 **Scarlett Huang** sh2557 **Zihan Zhang** zz698

Abstract

Traffic condition prediction plays a vital role in travel decision-making and urban congestion alleviating in today's mobile internet era. This project evaluates the traffic prediction accuracy generated from a variety of classification algorithms to help drivers to optimize route plans as well as city planning. In our experiments, Deep Neural Networks (DNN) algorithm, which yields the highest accuracy score among other chosen algorithms, is proven to be the most suitable classification algorithm for fitting high-dimensional discrete traffic dataset provided by Didi Technology Co., and predicting traffic conditions.

1 Introduction

With the advent of the mobile internet era, all mobile device holders can become portrayers of traffic conditions and road capacity. If it is possible to have a more accurate prediction of traffic conditions based on real-time and historical traffic information, it will undoubtedly play a vital role in travel decision-making and alleviating urban congestion.

However, it is very difficult to predict future road conditions, which are affected by many factors such as time period, road capacity, the downstream topology of road networks, navigation traffic, and sudden road conditions.

The goal of our project is to accurately predict the traffic conditions (ie, unblocked, eased, and congested) in a certain period of time in the future based on real-time and historical road condition characteristics of road segments, basic road attributes, and road network topology relationship diagrams to help drivers to optimize route plans as well as city planning.

The project is an application of machine learning to real-world data and may be used to improve the algorithm of Didi company.

2 Background

In view of the complex road conditions in today's cities, the traditional prediction methods for road conditions are not so accurate, and the optimization algorithm for the logistics distribution path is not sensitive to changes in the road conditions so that its application in an actual logistics distribution system is not effective[1].

Neural networks are widely used in road condition prediction problems. Gyanendra Singh (2020) compared gene expression programming (GEP) and random effect negative binomial (RENB) models with DNN and indicated the DNN had the best performance[2]. Gao (2020) came up with a method that can resolve the queue length estimation problem in a mixed traffic scenario[3]. Liu, LJ (2017) proposed an MRT passenger flow prediction model with a deep neural network (DNN)[4]. Wu (2018), inspired by recent work in machine learning, introduced an attention-based model that automatically learns to determine the importance of past traffic flow. The convolutional neural network was also used to mine the spatial features and the recurrent neural network to mine the temporal features of traffic flow[5].

3 Method

This project mainly applies machine learning and deep learning techniques, including DNN, CNN, KNN and Logistic Regression, to solve this road condition prediction problem.

For Deep Neural Networks (DNN), we use Keras, an open-source software library that provides a Python interface for artificial neural networks, and chose TensorFlow as the back-end tool for Keras to build the DNN model to solve this multi-classification problem. The DNN model we build is composed of Input Layer, Hidden Layer, Output Layer, and softmax function. The input layer is composed of 4 neurons, corresponding to 40 features in the dataset as the input vectors. The hidden layer has two layers, each with 5 and 6 neurons, respectively. The output layer consists of three neurons, corresponding to the number of classes of the target variable in the dataset. Finally, the softmax function was created to solve multi-classification problems. In this model, the neuron activation function we chose is the ReLU function, the loss function we chose is cross-entropy, and the optimizer is Adam (adaptive learning rate). The weights and biases of each layer are randomly generated at the beginning. Corresponding to the above DNN structure, we built it with Keras.

For the Convolutional Neural Network (CNN), we adopt CNN to train and evaluate the accuracy. Different layers, batch sizes, and epochs were compared in order to find out the most suitable combination of layers and parameters.

For the K-nearest neighbors classifier (KNN), to obtain the most accurate result, we use the brute-force search to iterate through the values passed to fit method. The parameter n neighbors, the number of neighbors to use for KNN queries, was set to n neighbors = 4 due to the fact that there are only four different types of traffic conditions. We used the Minkowski distance metric with the power parameter p=2, which is equivalent to the standard Euclidean metric.

For Logistic Regression, the multiclass option is set to one-vs-rest (OvR) scheme for the training algorithm so that each classifier is trained on traffic conditions with that label as positive data and all conditions with other labels as negative data.

4 Experimental Analysis

4.1 Data Preprocessing

We signed up for the road condition Spatio-temporal prediction competition organized by China Computer Society and Didi Travel on DataFountain. By submitting an application for data acquisition, the original data set was downloaded on the Gaia Data Open Project website. This competition provides real-time and historical road condition information of Xi'an, as well as road attributes and road network topology information on the Didi platform from July 1, 2019, to July 31, 2019.

We extracted the data from the original TXT documents and stored them in CSV format. Specifically, we used pandas to extract the data into a data frame table with delimiters, and extracted some important time features: we extracted the time slice speed of each time cycle, the maximum, minimum, average, difference value of ETA speed, etc. The total number of features that we used is 40.

4.2 Deep Neural Networks (DNN)

4.2.1 Experimentation

After defining the DNN model, we need to train the model and evaluate the model performance on the test set. In this process, we ran four experiments to optimize model performance. We used *Accuracy* as the metrics and used the learning curve to see how the model fits.

(a) Experiment 1: 1 hidden layer, epochs=20, batch size=130, 5,000 training samples

First, we used 1 hidden layer (units=5) for the DNN model, and set the number of epochs to be 20 and the batch size to be 130. Then, we compacted the dataset to improve the training speed of the model, so that more parameters could be tried in a limited time. We took 5,000 samples from the full training set as a small training set and experimented on them. The experiment result was: $accuracy = 0.7980$, $loss = 0.7601$. From Figure 1, we found the loss was still very high despite 20 epochs.

(b) Experiment 2: Add one more hidden layer

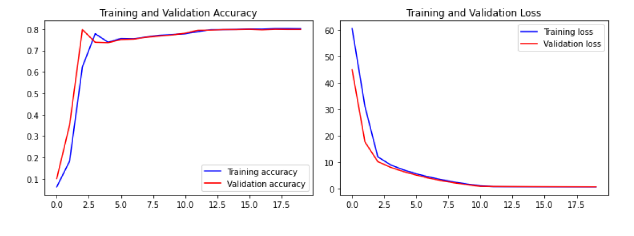


Figure 1: 1 hidden layer, epochs=20, batch size=130, 5,000 training samples

Based on experiment 1, we increased the number of hidden layers to 2 layers (units=5,6) and we still experimented on the 5,000 samples. Other model configurations remained the same as experiment 1.

The experiment result was: $accuracy = 0.8093$, $loss = 0.4763$. Compared to experiment 1, the *loss* was greatly reduced and the *accuracy* was also slightly improved. From Figure 2, we can see that the model fits well. This experiment shows that by increasing the number of hidden layers, the model performance can be significantly improved.

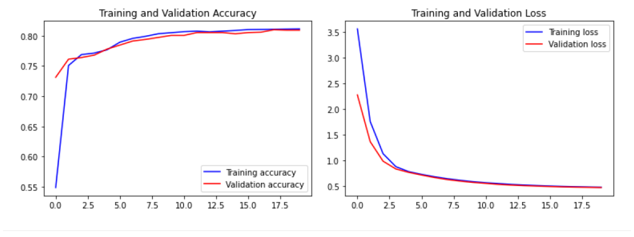


Figure 2: Add one more hidden layer

(c) Experiment 3: Increase epochs to 100

Based on experiment 2, we increased the number of epochs of the model from 20 to 100 and we still experimented on the 5,000 samples. Other model configurations remained the same as experiment 2.

The experiment result was: $accuracy = 0.8247$, $loss = 0.4345$. Compared to experiment 2, the *accuracy* was significantly improved and the *loss* was also reduced. From Figure 3, we can see that the model still fits well. This experiment shows that by increasing the number of epochs, the model performance can acquire a significant improvement.

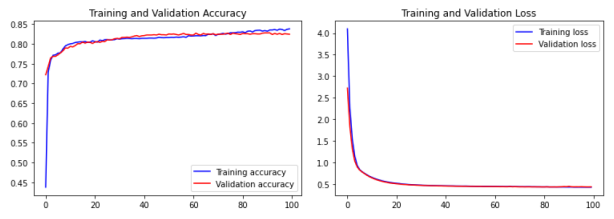


Figure 3: Increase epochs to 100

In the training process, we found that the data in the training set was seriously unbalanced. Among the 5,000 training samples, 81.36% are label 1, while the number of label 2 and label 3 is very few. label=1 contains 4068 samples, label=2 has 719 samples and label=3 has 213 samples. Therefore, the accuracy was not very high. So we decided to use the method of adding more data to solve this class imbalance problem. Thus we conducted experiment 4.

(d) Experiment 4: Expand the size of the training set to 4,094,303 samples

Based on experiment 3, we increased the size of the training set from 5,000 samples to 4,094,303 samples and increased the DNN model's batch size from 130 to 1000. Other model configurations remained the same as experiment 3. The data distribution of the new training set is: label=1, 3322308; label=2, 604008; label=3, 167988.

The experiment result was: *accuracy* = 0.8518, *loss* = 0.3815. Compared to experiment 3, the *accuracy* was greatly improved and the loss was greatly reduced. From Figure 4, we can see that the model still fits well. (Note: The Y-axis has a very small unit interval, so the learning curves seem to fluctuate wildly.) This experiment shows that adding more data and balancing data distribution are good ways of improving model performance.

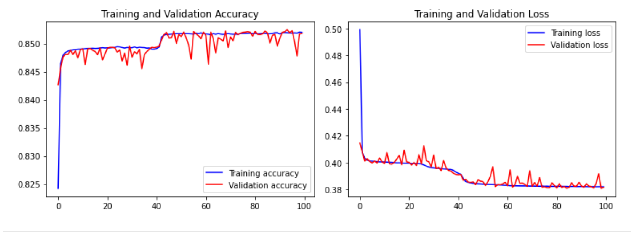
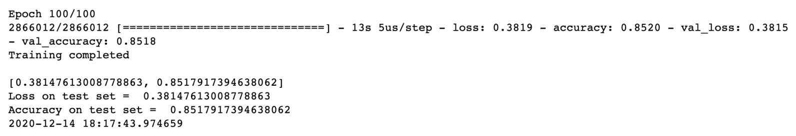


Figure 4: Expand the size of the training set to 4,094,303 samples



```
Epoch 100/100
2866012/2866012 [==============================] - 13s 5us/step - loss: 0.3819 - accuracy: 0.8520 - val_loss: 0.3815
- val_accuracy: 0.8518
Training completed

[0.38147613008778863, 0.8517917394638062]
Loss on test set =  0.38147613008778863
Accuracy on test set =  0.8517917394638062
2020-12-14 18:17:43.974659
```

Figure 5: Result

4.2.2 Model Finalization

Based on the experiment results above, we chose the DNN model in experiment 4 as our finalized model, because it yielded the best performance among all models we experimented with. Its training set consists of 4,094,303 samples and its *epochs* = 100, *batchsize* = 1000. The model *accuracy* = 0.8518, *loss* = 0.3815.

4.3 Comparison With Accuracy of Other Models

4.3.1 Convolutional Neural Network (CNN)

(a) Create a normal CNN model with Conv1D(32,7,'relu'), *batchsize* = 32, *epochs* = 10

Result: [*loss*, *accuracy*] = [0.43771999811212103, 0.8364492654800415]

(b) Model 1: The model in a and *epochs* = 50

Result: [*loss*, *accuracy*] = [0.4425459194309546, 0.8283397555351257]

(c) Model 2: Add *dropout* = 0.5

Result: [*loss*, *accuracy*] = [0.44653557607106215, 0.833827018737793]

(d) Model 3: Add another (Conv1D(64,7,activation = 'relu')) and (MaxPooling1D (5))

Result: [*loss*, *accuracy*] = [0.5268136289855133, 0.8255232572555542]

4.3.2 K-nearest neighbors classifier (KNN)

The highest accuracy score for KNN, 0.8156, is obtained when using the brute-force search approach, n neighbors = 4, and Minkowski distance metric with the power parameter p=2.

4.3.3 Logistic Regression

The highest accuracy score for Logistic Regression, 0.8356, is obtained when the multiclass option is set to one-vs-rest (OvR) scheme for the training algorithm.

5 Discussion and Prior Work

Previously, the dataset was parsed into a train set and a development set; the prediction model was trained using a support vector machine with only 10,000 rows of data with 129 traffic features, which consists 1% of the given traffic dataset. The highest f1 score, 0.75 was obtained when the regularization parameter c = 0.01.

Building upon previous work, the dataset is expanded to around 4,000,000 rows of data and we carefully selected the 40 most representative features such as the number of passing vehicles, average speed, etc. To find the most suitable machine learning model, a variety of algorithms and methods were evaluated such as CNN, DNN, KNN, Logistic Regression, etc. We found that DNN yielded the highest accuracy score among other chosen algorithms and is proven to be the most suitable classification algorithm for the provided dataset.

Key takeaways from the experiments:

(a) Tuning parameters on a small subset before testing on the whole dataset is vital to increase computing speed. Tuning parameters is only to find the appropriate parameters, not to produce the final model. Generally, the parameters that are appropriate in a small dataset will not be too bad in a large dataset. So we don't need to run every experiment on the full large training set, which was what we always do in the previous stage of this project. Instead, we can sample the training data in a small proportion and run experiments on that, so that we can try more parameters in a limited time.

(b) Expanding data size is an effective method for improving the performance of deep learning models.

(c) Plotting learning curves. During the experiments, we found that plotting learning curves is a very good method, which could help us know how the model fits and guide our directions of subsequent parameter adjustments.

(d) Selecting the number of epochs is important. With the increase of the number of epochs, the number of weight updating iterations in the neural network increases, and the curve gradually enters the optimized fitting state from the initial unfitting state to the final overfitting state.

(e) DNN is suitable for unstructured discrete data with high-dimensional features given that DNN can model complex non-linear relationships with multiple layers between the input and output layers.

6 Conclusion

In our project, we evaluated and compared the prediction accuracy of a variety of machine learning classification algorithms such as CNN, KNN, DNN, and logistic regression models for traffic condition prediction. Based on the experimental results, we draw the conclusion that DNN is more suitable for unstructured discrete data with high-dimensional features.

However, predicting traffic condition in real-time using DNN model is challenging due to the fact that the DNN model needs to iterate through multiple layers between the input and output layers in each epoch, which relies on computer performance.

In the follow-up work, we plan to combine the attributes of high importance, and then predict the overall data. Since the data set has a total of 128 features, we plan to try more feature selection combinations and find the feature with the highest weight for prediction. At the same time, we plan to combine CNN and RNN to learn and predict data sets using LSTM.

Ensemble learning is to combine multiple weakly supervised models in order to obtain a better and more comprehensive strong supervised model. The underlying idea of ensemble learning is that even if a certain weak classifier gets a wrong prediction, other weak classifiers can also correct the error back. We plan to use ensemble learning to learn and predict the model in the future, and compare the results with the work that has been done.

References

[1] Li, W., et al. (2019). Road condition prediction and logistics distribution path optimization algorithm based on traffic big data. Journal of Algorithms & Computational Technology, 13, 174830261987419-.

[2] Singh, G., et al. (2020). Deep neural network-based predictive modeling of road accidents. Neural Computing and Applications, 32(2006).

[3] Gao, K., et al. (2020). An integrated algorithm for intersection queue length estimation based on IoT in a mixed traffic scenario. Applied RNCES, 10(6), 2078.

[4] Liu, L., et al. (2017). An MRT daily passenger flow prediction model with different combinations of influential factors. 601-605.

[5] Wu, Y., et al. (2018). A hybrid deep learning-based traffic flow prediction method and its understanding. Transportation Research, 90(MAY), 166-180.