LSTM Model for Short-term Traffic Flow Prediction

Yubing Bai[1] **Zhonghui Hu**[2]

Abstract

Short-term traffic flow prediction is an essential function of the intelligent transportation system. In this project, we propose an RNN-based method to predict short-term traffic flow with historical records of different lengths. To better address the problem of gradient vanishing and exploding in time-series prediction, we apply the LSTM method, which keeps a reasonable balance between memorizing and forgetting, to the model with Pytorch. To get better performance, different sets of conditions are trained and eventually we got a well-fitted model with the MSE of 74.1 in the training dataset and 80.68 in the testing dataset, which indicates a great performance on the prediction by our model.

1. Introduction

Traffic flow prediction is defined as the problem of estimating future traffic from the previous and achieved traffic flow data. With the exploding number of vehicles and increasing level of traffic congestion, it has already become a critical issue to provide precise and timely traffic flow prediction to the drivers and passengers. What's more, short-term traffic flow prediction plays an important role in the intelligent transportation system as well. It can be used for real-time traffic planning and navigation.

Methods for predicting traffic flow can be divided into two categories. On the one hand, traditional statistical methods are utilized to predict the short-time traffic flow. Among these methods, autoregressive integrated moving average (ARIMA) model(Fernandes et al., 2019) was commonly used to predict short-term traffic data, which is simple to implement while cannot capture the stochastic property of the traffic system.

On the other hand, machine learning methods are increasingly used in predicting traffic flow. K-nearest neighbors regression and support vector regression is utilized to predict the traffic flow. During recent years, with the development of deep learning, methods(Lv et al., 2015; Huang et al., 2014) base on deep neural networks have become more and more popular in the prediction of traffic flow data. Huang(Huang et al., 2014) predict the traffic flow with deep belief networks with multitask learning. Wang(Wang et al., 2016) combined chaos with RBF neural network to do the prediction. Experiments show that these methods achieve promising results.

In deep neural networks, recurrent neural networks are capable of capturing the properties of time series data. Long Term Short Memory network (LSTM)(Hochreiter & Schmidhuber, 1997) as a special type of RNN, has memory cells to store history data, allowing the model to tackle vanishing and exploding gradients – which happens when you back-propagate through time too many time steps thus causing the gradients accumulating to either infinite or zero.

In this paper, we introduce a short-term traffic flow prediction method base on LSTM. As LSTM has the ability to memorize the historical data, we use traffic flow series of different lengths for the experiments and compare the prediction results respectively to find the most appropriate length to predict the short time traffic flow under this model.

In the following sections we will concisely introduce the background knowledge of RNN and LSTM in section 2, depict our complete method architecture in section 3, display the experimental results and make the corresponding analysis based on them in section 4, and finally, give a brief conclusion in section 5.

2. Background

2.1. Recurrent Neural Network

Recurrent neural networks (RNN) is a type of feed-forward neural networks. For traditional feed-forward neural networks, the information is passed forward through the network, and thus do not memorize the historical information. There are some problems of the application of such networks whose states depend only on current inputs. For example, the preceding words must be concerned to predict

[1]New York Univeristy, New York City, NY, The USA [2]New York Univeristy, New York City, NY, The USA. Correspondence to: Anonymous <>.

the next word of an incomplete sentence, and a simple full-connected network is hard to do that. RNNs are competent to deal with sequential problems for its ability to memorize the historical information by saving the previous output of the network as part of the current input. Technically, there are no limitations on the length of sequences needed to be processed by RNNs, but it is hypothesized that current state depends only on a few previous states in order to reduce the complexity of models in practice(Schuster & Paliwal, 1997).

RNNs consist of input units, output units, and hidden units. Hidden units, as distinct from full-connected neural networks, are capable of catching previous information of hidden units by a chain of repeating modules of neural work which has a simple structure, as equation 2.1.

$$\mathrm{y}_t = \sigma_y(W_y s_t + b_y)$$
$$\mathrm{s}_t = \sigma_h(W_h x_t + b_h + U_h s_{t-1})$$

2.2. Long Short Term Memory

Gradient vanishing and gradient exploding that occurs in standard RNNs results in that RNNs are hard to catching long-term previous information, while LSTMs(Hochreiter & Schmidhuber, 1997) are designed to avoid such problem and are proven to be very efficient.

LSTMs have a more complex structure than RNNs. There are two states cell state c_t and hidden unit state h_t in one cell of LSTMs, and three gate-controlled signals $z^o, z^f, andz^i$. The relationships between these states and signals are shown in equation 2.2.

$$\mathrm{z} = \tanh(\mathrm{W}\ (\mathrm{x}_t, h_{t-1}))$$
$$\mathrm{z}^i = \sigma(W^i(x_t, h_{t-1}))$$
$$\mathrm{z}^f = \sigma(W^f(x_t, h_{t-1}))$$
$$\mathrm{z}^o = \sigma(W^o(x_t, h_{t-1}))$$
$$\mathrm{c}_t = z^f \odot c_{t-1} + z^i \odot z$$
$$\mathrm{h}_t = z^o \odot tanh(c_t)$$
$$\mathrm{y}_t = \sigma(W h_t)$$

where $\odot$ is the element-wise product. $a \odot b; a = 0or1$ represents that a is a control signal that it will get b if a is 1 or 0. z^f, z^i, z^o are vectors whose values are between 0 and 1 as signals to control the gate closed or open, and z is between -1 and 1 as an input. Gate-controlled signals help LSTMs choose specific and important information to memorize and forget others, which explicitly improves the performance of LSTMs at long-term dependencies.

3. Architecture

In this section, we will illustrate our method and implementation in detail. After introducing the overview diagram of the model, we specifically describe the data processing method, parameters setting of the LSTM model, loss function selection and optimizer selection.

3.1. Architecture Diagram

diagram.pdf

Figure 1. The diagram of overall model

The diagram of our experiment is shown in Figure 1. Firstly data is preprocessed in python. The preprocessing includes data screening, data cleaning, and data format processing. Then we load our well-processed data into the training model, which comprises of the input interface, the two-layer LSTM model and the output interface. In the model block, there exists an LSTM core controlling the forgetting and memorizing. Traffic data is fed sequentially and each turn the model will output the current prediction as well as the output gate, which is used to generate the gate in the next round. Finally, we input our test data into the trained-model to evaluate the model performance.

3.2. LSTM

The LSTM model is implemented with the base class Module in Pytorch. Several critical parts are the input layer, the hidden layer, and the output layer respectively.

Input vector dimension is dependent on how long the history of traffic flow we need, and it will be adjusted according to the performance. As to batch size, we input all of our training data at once without dividing into several batches

to get better precision.

There are two hidden layers in the model, in which each hidden node is a 32-dimension vector. Hidden node dimension ranging from 20 to 40 does not make a great difference, so this is just an empirical parameter.

Between the layers is the activation function. In our project, we utilize the linear activation function, which takes the input multiplied by the weight, and creates an output proportional to the input.

Finally, the output is designed as a one dimension vector since we need to predict the traffic flow at a single point in the future.

3.3. Loss Function

Loss function evaluates the difference between the output of the model and the real word result at each turn so that the model can find its right direction to optimize the performance as fast as possible.

Here we choose mean square error(MSE) loss function, which measure the difference by the sum of square of difference between each pair of (Y^i_{output}, Y^i_{real}). The formulation can be written as:

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} \left(Y^i_{output} - Y^i_{real}\right)^2$$

We select this form because compared to smooth L1 loss, MSE is more suitable for problems that are not very high dimensional and has low numerical features, right as our data shows. Besides, this method is basic and easy to derive the gradient.

3.4. Optimizer

The optimizer is a strategy we use to find a way to decrease the loss function and then update the parameter.

In this project, we involve Adam (Kingma & Ba, 2014) as our optimizer. This method combines RMSprop and Stochastic Gradient Descent(SGD) with momentum, as with the former one it scales the learning rate and as with the latter one it moves an average of the gradient instead of solely gradient.

The first and second moment estimates can be written as:

$$m_t = \frac{\beta_1 m_{t-1} + (1-\beta_1)\, g_t}{1-\beta_1^t}$$

$$v_t = \frac{\beta_2 v_{t-1} + (1-\beta_2)\, g_t^2}{1-\beta_2^t}$$

where g_t denotes the gradient at time t. And the final update rule of optimization is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

Algorithm 1 LSTM

Input: data $\vec{x}$, size n
model parameter settings
repeat
  Forward pass with LSTM model.
  Loss evaluation with MSE.
  Backward pass for gradient.
  Parameter updating with Adam.
until Achieve Convergence

3.5. Pseudo-code of the Algorithm

The algorithm can be described by Algorithm 1.

4. Experiments

4.1. Data

Traffic flow data is generated from the public dataset Caltrans Performance Measurement System(Pems). California Department of Transportation is a freeway performance measurement system which collects real-time and historical traffic information. We choose road I5-S and obtain the total flow data of lane 3 and lane 4. The time interval of the traffic flow is 5 minutes. As the traffic flow data has different patterns during weekdays and weekends, and also varies in the peak hours and other time periods, we put the emphasis on the prediction of traffic flow during peak hours on weekdays, which is from 7:00 to 10:00 during 2018.

To compare the experimental results of different sequence length, traffic flow sequences of 3, 6, 9 and 12 data points are generated, which are corresponding to 15, 30, 45, 60 minutes. During the experiment, 80 percent of data is used for training and others for testing. Based on the traffic flow sequence, we predict the traffic flow in the next 5 minutes using over 2000 traffic flow sequences.

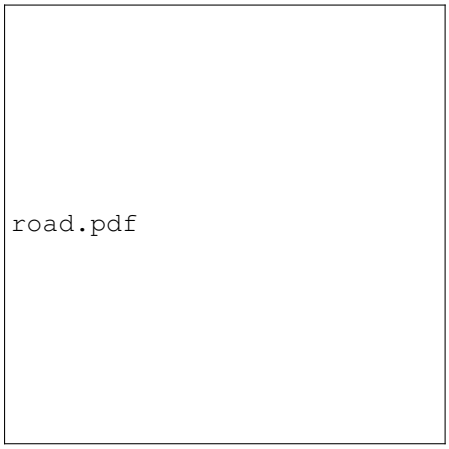

Figure 2. Location of road

Table 1. Result comparison with training set and testing set

SEQ LENGTH	TRAININGSET MSE	TESTINGSET MSE
3	112.19	124.15
6	119.52	126.84
9	74.10	80.68
12	83.84	88.31

4.2. Results And Analysis

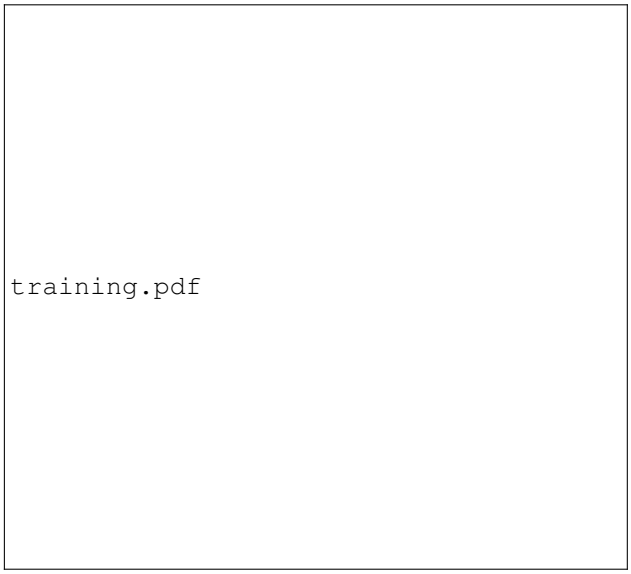

Figure 3. Result of training set

Figure 3 shows the comparison of output of training set on different conditions. The left column shows the comparison between the true data and predicted data over 2000 data records, the middle column shows a partial detail for 100 data points in the left column, and the right column draws the process of convergence of the MSE. The right column shows that the MSE drops sharply during the first several epochs, and then slowly converges to the stable value. Figure 4 shows the comparison of output of testing set on different conditions. Images on the left shows the overview of the MSE mapping and the right ones draws the details.

We can see a strong relationship between prediction and real label, which means the model generates a proper imitation of short-term traffic flow trending, and it also has a good performance for predicting the short-term traffic flow in the testing dataset.

The corresponding quantitative results are shown in table 1, traffic flow sequence of 9 data points have the best performance under this LSTM network. The MSE for the training set is 74.1. As the MSE for the sequence of 3 data points and 6 data points in the training and testing sets has relatively high values and we can infer that when the length of the sequence increases from 9 data points, the performance will get worse, and when the length of the sequence decreases, it also gets worse.

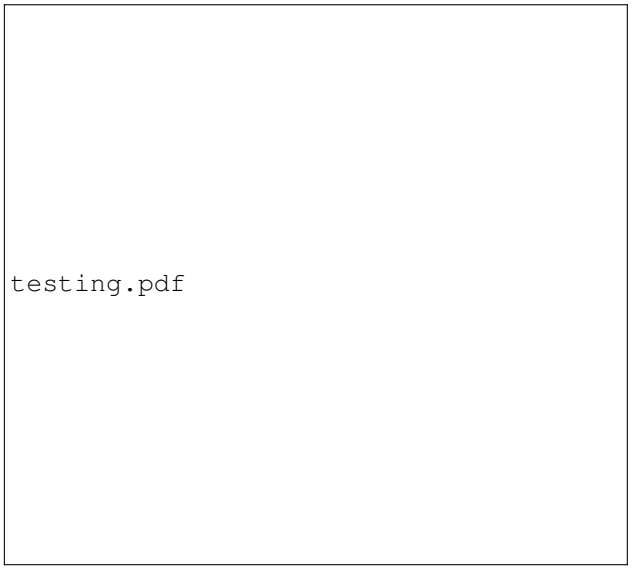

Figure 4. Result of testing set

5. Conclusion

An LSTM neural network is proposed in this paper for single point short-term traffic flow prediction. The contributions mainly include two aspects: 1) we design a well-fitted LSTM network for predicting traffic flow data; 2) traffic flow sequences of different length are generated to make comparisons between the performance of each length, and 45-minute traffic flow sequence under this model is proved to be the optimal length for prediction.

Acknowledgements

References

Fernandes, B., Silva, F., Alaiz-Moretón, H., Novais, P., Analide, C., and Neves, J. Traffic flow forecasting on data-scarce environments using arima and lstm networks. In *World Conference on Information Systems and Technologies*, pp. 273–282. Springer, 2019.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Huang, W., Song, G., Hong, H., and Xie, K. Deep architecture for traffic flow prediction: deep belief networks

with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2191–2201, 2014.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lv, Y., Duan, Y., Kang, W., Li, Z., and Wang, F.-Y. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

Schuster, M. and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Wang, C., Zhang, H., Fan, W., and Fan, X. A new wind power prediction method based on chaotic theory and bernstein neural network. *Energy*, 117:259–271, 2016.