The Application of Deep Learning Method in Traffic Flow Prediction

LIU Xingwei[1], Kuniaki SASAKI[2]

[1]Student not member of JSCE, Postgraduate Student, Graduate School of Engineering University of Yamanashi
(4-4-37, Takeda, Kofu, Yamanashi, 400-8510 Japan)
E-mail:emma.liu5218@gmail.com
[2]Member of JSCE, Professor, Dept. of Civil and Environmental Engineering Waseda University,
(3-4-1, Okubo, Shinjuku, Tokyo, 169-8555, Japan)
E-mail: sasaki.k@waseda.jp

Traffic flow prediction is in a very important position in the research of transportation. There are many kinds of traffic flow prediction methods which can handle different situations. However, with the development of the city, transportation is becoming more complex going with sudden evens. Deep learning methods have high adaptability to meet the requirements and deep learning methods are gradually applied on traffic flow prediction by more researchers.

This thesis introduces the definition of neural network, recurrent neural network and long short term memory model. The thesis also summarizes the principle, calculation method and application scope of these three models. Through the analysis, it is necessary to adopt the deep learning method when facing the big data traffic flow and it provides the reference to choose the deep learning method.

Key Words : *traffic flow prediction, deep learning, neural network, recurrent neural network*

1. INTRODUCTION

The research of traffic flow prediction methods started on the 70s of the last century. Through the continuous efforts of researchers, traffic flow prediction method has undergone many improvements and updates. The current mainstream methods can be roughly divided into three categories[1)], they are models based on mathematical statistics theory, such as autoregressive (AR) model, moving average (MA) model, autoregressive moving average (ARMA) model, and integrated autoregressive moving average (ARIMA) model. Models based on nonlinear theory includes wavelet analysis, chaos theory and so on. Models based on intelligent prediction theory consists of deep learning, SVM model and so on.

Some reference pointed out that the time series model is widely used and most application in real case of traffic flow prediction is based on the ARIMA algorithm. With the deepening of research, the complexity and nonlinear characteristics of traffic flow are gradually being focused by researchers. Only replying on time series model can't meet the prediction of the complex transportation system.

With the continuous development of computer technology, it appears many concepts such as machine learning, deep learning and so on. Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world[2)]. Shallow learning is the first development of machine learning and now deep learning appeared as emerging information. Figure

1 shows the relationship among these concepts.

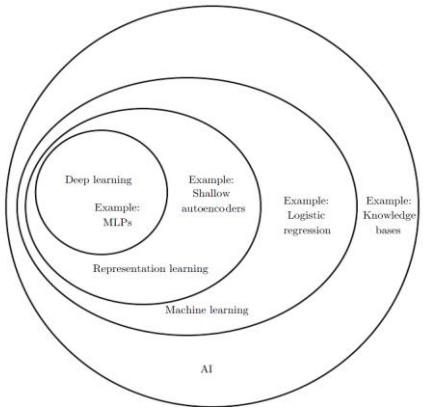

Fig.1 Concepts relationship

Deep learning is gradually developed along with the neural networks. Hinton first proposed the concept of deep learning in 2006 and deep learning is a kind of algorithm used a multi-layered structure to model the complex relationships among data. At present, deep learning is mainly applied in the fields of image recognition, speech recognition, automatic translation and has achieved remarkable results. In 2013, deep learning is named one of the top 10 technological breakthroughs by MIT Therefore, scholars have gradually turned to the use of neural network algorithms to solve problems in traffic flow prediction.

This thesis first introduces the definition of deep learning and give several methods of deep learning. Besides, this thesis also introduces the model principles, usage conditions, advantages and disadvantages of different methods.

2. NEURAL NETWORK

Neural network refers to imitating the neuron activity of human or animal, learning its behavioral characteristics, and constantly adjusting the relationship among the nodes in parallel to achieve the goals. The traditional neural network consists of three network structures, including the input layer, the hidden layer, and output layer[3].

In figure 2, $x_1, x_2, x_3 \cdots$ represents input values, $\omega_1, \omega_2, \omega_3 \cdots$represents weight values. y is the output value. A neural consists of three parts: weights, sum function and activation function. The activation function can limit the output range of neuron. The common activation functions are S function, tanh function, rectifier function and so on.

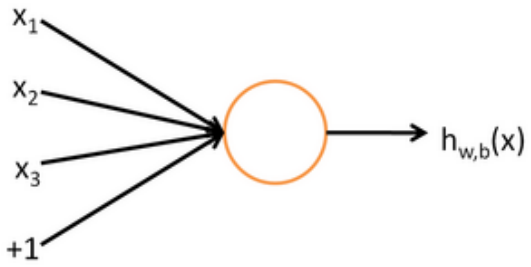

Fig.2 The structure of neural network

The function can be written as:

$$y = f_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{t} W_i x_i + b)$$

The training of neural network is the process that adjusts the weights in network. The network weight is a vital factor to influence neuron performance. Therefore, adjusting the weights can build a suitable neural network. The purpose of neural network is to reduce the errors happened from parameters includes the weights, threshold and bias. In the training of neural network, it always adopts the back propagation algorithm.

Back propagation algorithm is one of the most popular algorithms. The gradient descent method is used to adjust the network weight to minimize the error function of expected output and network output where the weight moves in a negative direction and each step can make the error smaller until the minimum error reached, or threshold value reached. The back propagation calculation has two periods: forward propagation and back propagation.

In forward propagation weights keep invariant. It can be written as:

$$y_j = \sum_{i=1}^{k} \omega_{ij} x_i \, , j = 1,2, \cdots m$$

$$y_q = (y_1, y_2, y_3 \cdots y_m)$$

The activation function here is the sigmoid function:

$$f(x) = \frac{1}{1+e^{-x}}$$

In back propagation, the local gradient is written as:

$$\delta_{jk}^{d} = -\frac{\partial y_k}{\partial \omega_{jk}^{d}}$$

d means training times. η is the learning rate of neural network weight.

The adjustment value of weight is written as:

$$\Delta\omega_{jk}^{d+1} = \eta\delta_{jk}^{d}$$

The weight after adjusting can be written as:

$$\omega_{jk}^{d+1} = \omega_{jk}^{d} + \Delta\omega_{jk}^{d+1}$$

If the error $y_k - \widehat{y_k}$ is less than the set value or research the training times, then the weights won't update. Otherwise, the value will follow the process as above.

3. RECURRENT NEURAL NETWORK

The information in the traffic flow data is not independent, and there have time connections between the data more or less, so the traffic flow data can be regarded as time series. In the traffic flow prediction, if we plan to predict the traffic flow at time $t+1$, only traffic flow data at time t is not enough, data from time 0 to time $t-1$ are also required. In the general neural network model, no connections among hidden layers. When the traffic flow data is inputted, the time series information will be lost during that period[4) 5)]. But recurrent neural network can retain timing information with the help of its special structure.

The structure of recurrent neural network consists of input layer, hidden layer and output layer and connections between neurons in the hidden layer.

A big characteristic of RNN is that there have connections among neurons in the hidden layer which make the sequence information can be preserved.

To facilitate mathematical analysis, mathematical symbols are included in Figure 3.

Where:

x_t represents the vector of the input layer value;

s_t represents the vector of hidden layer value, it depends on the input value x_t of current time and value s_{t-1} of the previous time;

O_t represents the vector of the output layer value;

U is the weight matrix from the input layer to the hidden layer;

V is the weight matrix from the hidden layer to the output layer;

W is the weight matrix. The weight matrix W at the current time is obtained from the result at a previous time.

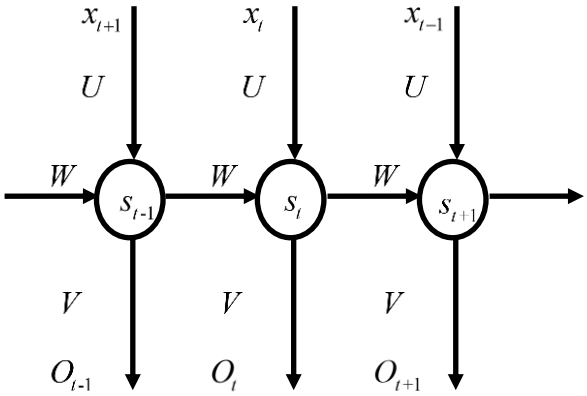

Fig.3 RNN structure flowchart after assignment

From Figure 3, it can be known that:

$$O_t = g(Vs_t)$$

$$s_t = f(Ux_t + Ws_{t-1})$$

V_{st} is the weight matrix of the output layer,

g() and f() are activation function.

Substituting $s_t$into O_t, we can obtain equation:

$$O_t = g(Vs_t)$$

$$= Vf(Ux_t + Ws_{t-1})$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + Ws_{t-2}))$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Ws_{t-3})))$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \cdots))))$$

We used back propagation through time algorithm to train the recurrent neural network which is to do partial derivative for each parameter.

The activation function here is:

$$\widehat{y_t} = softmax(Vs_t)$$

The cost function is:

$$E_t(y_t, \widehat{y_t}) = -y_t log\widehat{y_t}$$

Here net_t represents the weighed input at time t, it wrote as:

$$net_t = Ux_t + \mathrm{W}s_{t-1}$$

$$s_{t-1} = \mathrm{f}(net_{t-1})$$

The calculation of error can write as:

$$\delta_h^t = \frac{\partial E}{\partial net_h} = \frac{\partial E}{\partial net_t}\frac{\partial net_t}{\partial net_k}$$

$$= \frac{\partial E}{\partial net_t}\frac{\partial net_t}{\partial net_{t-1}}\frac{\partial net_{t-1}}{\partial net_{t-2}}\cdots\frac{\partial net_{k+1}}{\partial net_k}$$

Similarly, it can be known:

$$\delta_h^{l-1} = \frac{\partial E}{\partial net_t^{l-1}}$$

From this equitation, the error can transmit to the upper layer.

The last step of BPTT is gradient weight calculation.

The partial derivative can be written as:

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial z_y^t}\frac{\partial z_y^t}{\partial W} = (h^t)^T\delta_y^t$$

$$\frac{\partial E}{\partial V} = \frac{\partial E}{\partial z_h^t}\frac{\partial z_h^t}{\partial V} = a^T\delta_h^t$$

$$\frac{\partial E}{\partial U} = \frac{\partial E}{\partial z_h^t}\frac{\partial z_h^t}{\partial U} = (h^{t-1})^T\delta_h^t$$

$$\frac{\partial E}{\partial b_h} = \delta_h^t$$

The gradient calculation equation in hidden layer can be summarized as:

$$\delta_h^t = \frac{\partial E}{\partial a_h^t} = \theta_h'(a_h^t)\left(\sum_{h'=1}^{H}\delta_{h'}^{t+1}\omega_{hh'}\right)$$

$$= \sum_{h_{t+1}=1}^{H}\cdots\sum_{h_T=1}^{H}\sum_{k=1}^{K}(y_k^T - z_k^T)\,\omega_{hk}\prod_{m=t}^{T-1}\theta_{h_m}'(a_{h_m}^m)\,\omega_{h_m h_{m+1}}$$

It can be seen that the right side of equation has $\prod_{m=t}^{T-1}\theta_{h_m}'(a_{h_m}^m)$, when the number of T is increasing, we can get:

$$\begin{cases}\delta_h^t \to \infty, & if|\theta'\omega| > 1.0\\ \delta_h^t \to 0, & if|\theta'\omega| < 1.0\end{cases}$$

When $\delta_h^t \to \infty$, gradient blow up will happen and $\delta_h^t \to 0$ will cause gradient vanish. That means when complex situations occur, there will be a long interval among important features and predicted information. The RNN model will gradually reduce memory ability of important features which can affect the prediction accuracy. In order to reduce the possibility of gradient blow up and gradient vanish, it can insert into a constrain into algorithm to keep $|\theta'\omega| = 1.0$. That is the model called LSTM model.

4. LSTM MODEL

Long short term memory (LSTM) was proposed by Hochreiter and Schmidhuber in 1997 and was improved and promoted by Alex Graves in 2005.

In many application scenarios, LSTM has made breakthroughs and is widely used and researched. The key difference in LSTM model is in hidden layer. The hidden layer in recurrent neural network only has one calculation module and the hidden layer in LSTM model adds a state module called cell state to describe the incremental changes in the network. Calculation module and cell state constitute the memory blocks together in a hidden layer of LSTM model[6)7)].

Memory blocks in the hidden layer are to deal with a long time series. Memory blocks contain cell states to store the temporal states of the network, calculation modules store the long term states and gates to control activations of state. Figure 4 shows the flow chart of the LSTM. How the memory blocks works will explain as following and it has four steps.

(1) Determining the discarded information

In the first step, we need to determine the information that is discarded from the cell state. This process is mainly done by the forget gate which represents in Figure 4 through the red line. The mathematical formulation is:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

W_f is the weight matrix of forget gate. $[h_{t-1}, x_t]$ is a long vector composited by two vectors. b_f is the bias term in forget gate, σ is a sigmoid function. The weight matrix W_f is obtained from two matrices W_{fh} and W_{fx}. The input items corresponding to W_{fh} and W_{fx} are h_{t-1} and x_t. It can be written as:

$$[W_f]\begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} = [W_{fh} \;\; W_{fx}]\begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$
$$= W_{fh}h_{t-1} + W_{fx}x_t$$

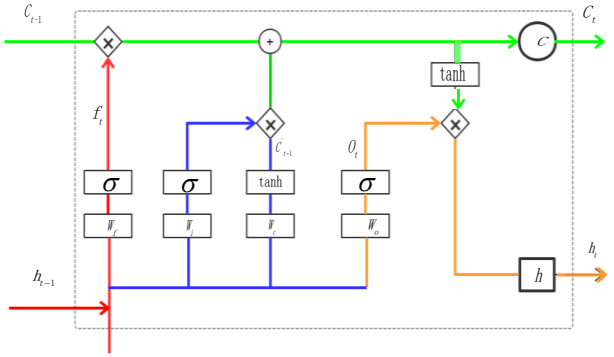

Fig.4 The core algorithm of LSTM model

(2) Determining the update

The blue line in Figure 4 showed this process. It consists of two parts. The sigmoid function determines the part that needs to be updated. The tanh function creates a new candidate vector and moves to the next operation. The mathematical formulation is:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$C'_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

(3) Updating the calculation module

In the previous step, we just confirmed what information needs to be updated. In this step, we need to update the cells state in the hidden layer from C_{t-1} to C_t. The mathematical formulation is:

$$C_t = f_t \times C_{t-1} + i_t \times C'_t$$

As shown through the green lines in Figure 4, multiplying previous cell state C_{t-1} by forget gate f_t, the model discarded unnecessary information. Multiplying current cell state C_t by input gate i_t, we get the new candidate value. Finally, we also get new cell state C_t. This process essentially is the combination with current (short) memory and long term memory.

(4) Outputting the information

After updating the cell state, the model needs to output the value learned through this layer. The process is shown by the orange line in Figure 4 and the mathematical formulation is:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \times \tanh(C_t)$$

5. CONCLUSION

Here we used highway traffic flow data collected from the Hachioji IC to the Kawaguchiko IC totally 1430 sets to verify these models. This dataset records the number of vehicles pass through the highway section every day.

The evaluation index of model error is root mean square error (RMSE). RMSE can measure the unbiasedness of time series and it has the same units as the original data. The smaller RMSE, the smaller the dispersion of the error distribution which means a better prediction.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}$$

$\hat{y}_t$ is predicted value, y_t is real value.

The prediction results are shown as below.

Table 1 RMSE value of different models

Model type	NN	RNN	LSTM
RMSE	768.6	712.4	572.5

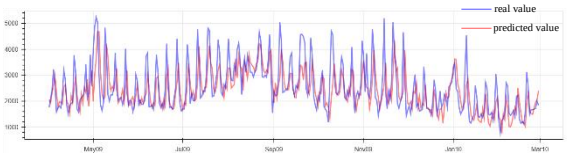

Fig.5 The prediction results of neural network

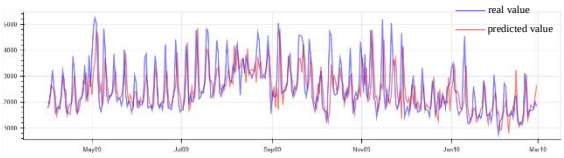

Fig.6 The prediction results of recurrent neural network

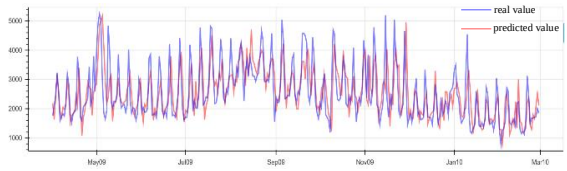

Fig.7 The prediction results of LSTM model

From the analysis as above, we can get the conclusion. Comparing with other models, the neural network has a simple network structure and easy to understand which is a good choice for beginners. But neural network can't store the time sequence information. The recurrent neural network has connections between neurons which could solve that problem. However, RNN will appear the gradient blow up or gradient vanish when the time series is long. The memory blocks in LSTM model can catch features in a long series where is the main advantage among all kinds of deep learning methods. It also provides the basis of dealing with big data. In the real traffic flow prediction projects, when the data we got is big with strong temporal correlation, LSTM model will be a good choice.

REFERENCES

1) Rong Y, Zhang X, Feng X, et al. Comparative analysis for traffic flow forecasting models with real-life data in Beijing[J]. Advances in mechanical engineering, 2015, 7(12): 1687814015620324.
2) Bengio Y. Learning deep architectures for AI[J]. Foundations and trends® in Machine Learning, 2009, 2(1): 1-127.
3) Jordan M I, Mitchell T M. Machine learning: Trends, perspectives, and prospects[J]. Science, 2015, 349(6245): 255-260.
4) Zhang J, Zheng Y, Qi D, et al. DNN-based prediction model for spatio-temporal data[C]//Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2016: 92.
5) Kuremoto T, Kimura S, Kobayashi K, et al. Time series forecasting using a deep belief network with restricted Boltzmann machines[J]. Neurocomputing, 2014, 137: 47-56.
6) Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
7) Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J]. Neural Networks, 2005, 18(5-6): 602-610.
8) Pascanu R, Mikolov T, Bengio Y. Understanding the exploding gradient problem[J]. CoRR, abs/1211.5063, 2012.