Comparison between elementary network structure and Prophet model for time-series forecasting: Merchandise sales

Xu Zhang 1, \*, Lixian Zhang1, Ziang Yang1

1School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, Guangdong, China

2School of aaaa, bbbb University, Changsha, China

**Keywords:** N

**Abstract:** In order to maximize the profits of supermarkets, this article discusses different models in relation to time series prediction, based on previous time points, to help with setting the merchandise price. In detailed, we compared 2 deep learning methods(BiLSTM and Transformer) and Prophet model for predicting the future sales information. The overall dataset comes from certain information in the cost unit price of each category in a certain market within continuous 61 days. We evaluated these models based on the gaps bewteen their predictions and actual values, as well as the model’s Interpretability. Our experiments show that … has performed better than … and ….

1. Introduction

?

1. Method

**2.1 Bi-directional LSTM**

1. **Overview**

BiLSTM[1] is an expansion of LSTM[2], which records not only backward contents structure but forward contents as well. It helps with obtaining contexts from both directions. Bseides, it could learn information from farther size of a sequence of time-series information and reduce the potential vanishing gradient problem as in the RNNs.Here, the implementation uses the pytorch’s inner package which contains the torch.nn.LSTM, which is largely used in many other articles.

1. **Model**
   1. **LSTM-unit**

BiLSTM could be divided into single LSTM unit. And the LSTM unit is comprised of 4 sub-unit: input gate, memory-cell, forget-gate and output-gate. And we have one hidden layer accompanied by too. At any given time, the inputs to it are of input vector , previously hidden layer previously memory cell , and outputs are current hidden state , and current memory cell .

Detailed introduction of inner components of a single LSTM unit:

Here represents the corrresponding weight in gate or cell unit, denotes the bias, and signals the activation function as sigmoid and etc.

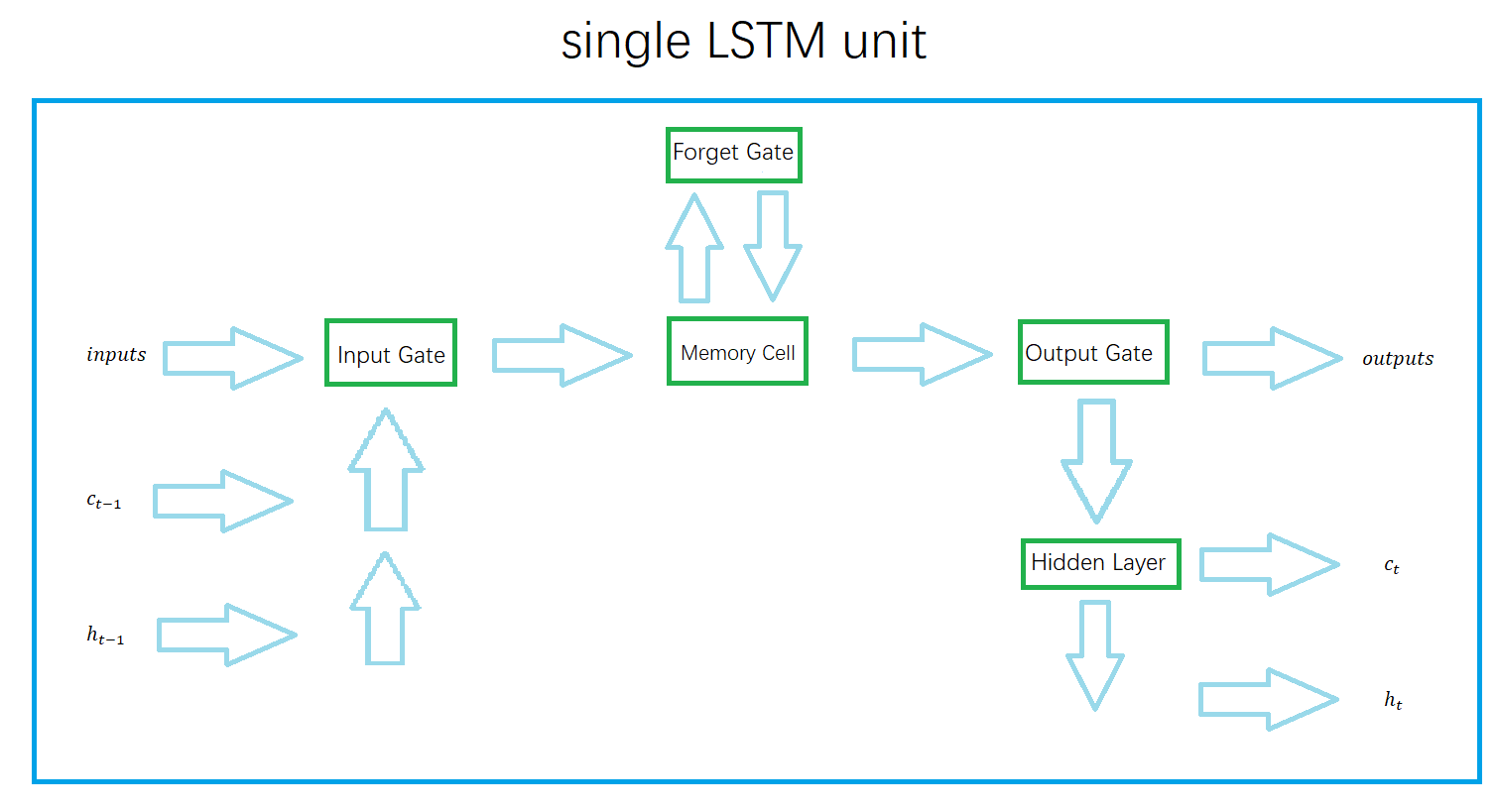
Input Gate:

Forget Gate:

Memory Cell:

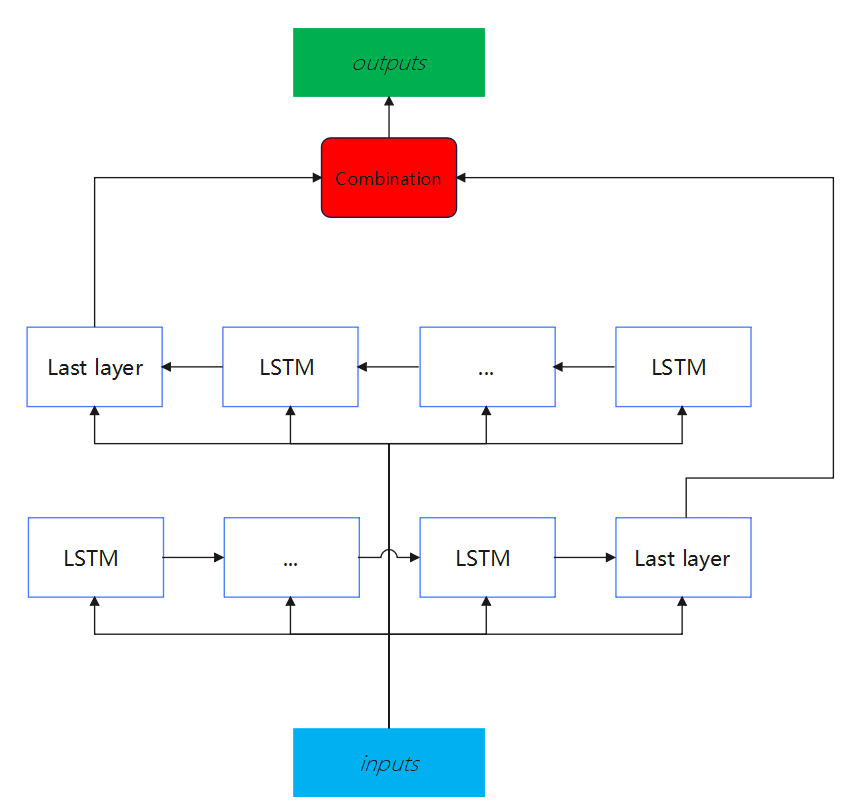
Output Gate:

Hidden layer:



* 1. **BiLSTM structure**

The bidirectional nature of BiLSTM employs two separate LSTM layers – one processing the input sequence in the forward direction, and the other in the backward direction.



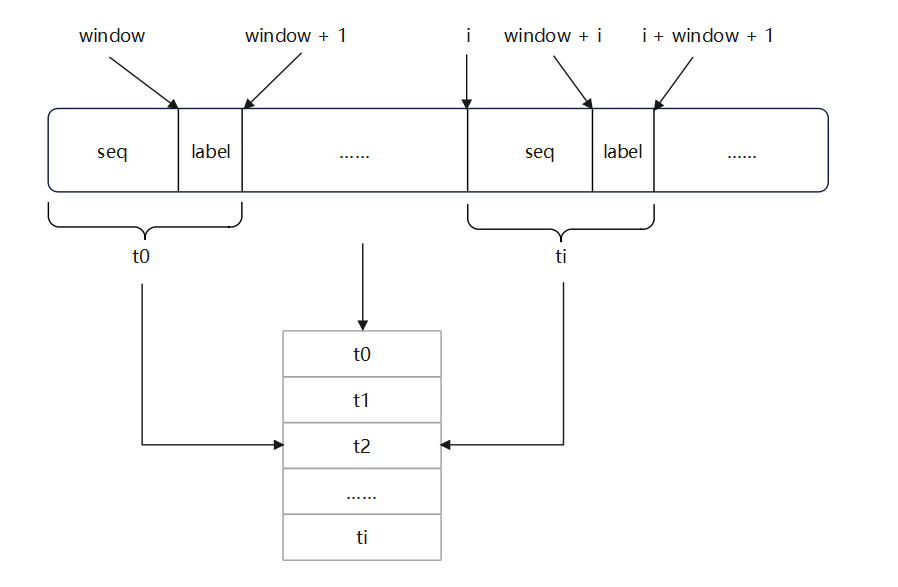
* 1. **Meaning**

The Bi-directional LSTM could reach a higher accuracy and lower loss compared with LSTM in prediction[3]. Apart from time-series prediction[4], it is even helpful for text classification[5] and recognition[6].

**2.2 Transformer**

1. **Overview**
2. **Model**
   1. **Data preprocessing**

For a given data set, the time series T in the data set is divided into equal time steps according to the historical window, which contains seq items and label items, and is spliced into a time series matrix. The specific diagram is as follows:



* 1. **Embedding**

When we apply the Transformer model to time series prediction, there are two key embeddings to consider: feature embedding and location embedding.

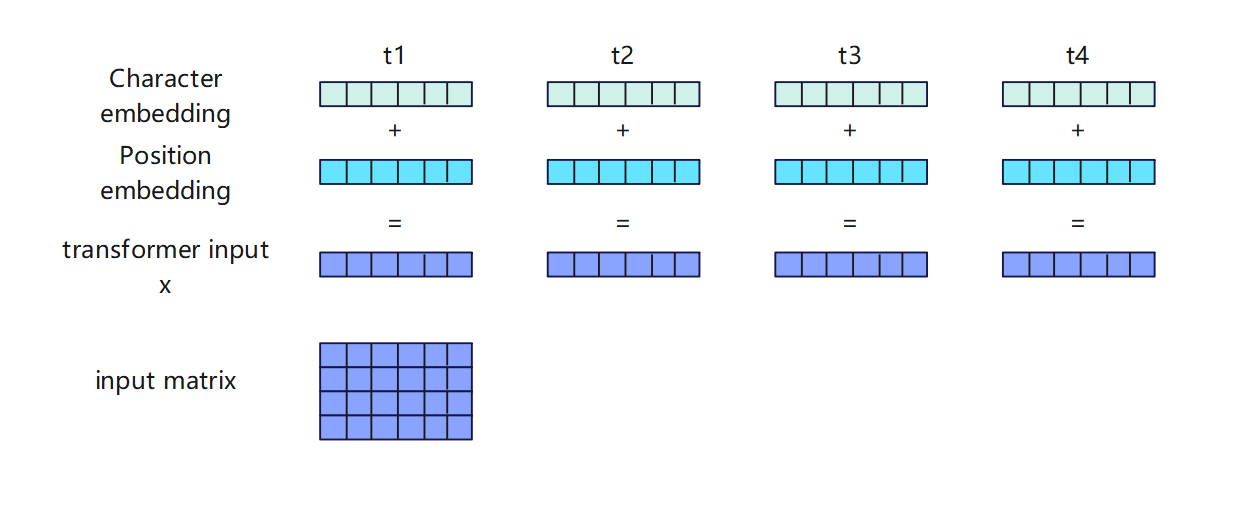
* + 1. **Feature embedding**

The purpose of feature embedding is to map each feature into a continuous vector representation. For each feature, we use an embedding matrix, where each row represents the embedding vector of a feature. Given the index of a feature (usually an integer), its embedding vector can be represented by the following mathematical formula:

* + 1. **Location embedding**

In order to introduce the position information of the sequence in the Transformer, we use Positional Encoding. The purpose of positional encoding is to provide the model with information about the position of elements in the input sequence. A common method of position encoding is to use trigonometric functions (usually sine and cosine functions) to generate position embeddings. The mathematical formula for positional encoding is usually expressed as:

Here, is the position of the element in the input sequence, and is the index of the embedding dimension. By adding feature embedding and position embedding, we provide the model with an input embedding that contains both feature and position information.



* 1. **Q , K, V matrix calculation**

In the Transformer model, the QKV matrix (Query-Key-Value matrix) is obtained by linearly transforming the input matrix X with three weight matrices. These three weight matrices are used to calculate linear mappings of queries, keys, and values respectively.

As the input obtained above, assume that the dimensions of the input matrix X are (batch size, sequence length, feature dimension), where the sequence length represents the length of the time series, and the feature dimension represents the number of features at each time step.

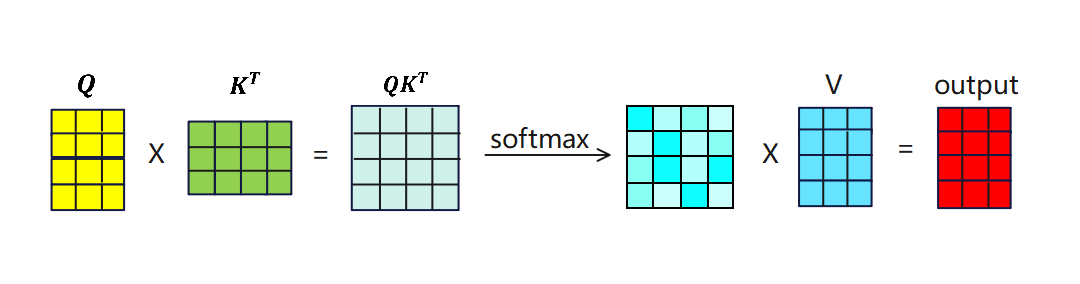
First, we obtain the Query (Q), Key (K), and Value (V) matrices by multiplying the input matrix X and the three weight matrices respectively, where represents matrix multiplication:

* 1. **Calculate the output of Self-Attention**

In the self-attention mechanism, we need to calculate the attention score and use it for weighted summation to get the output. The attention score is calculated as follows:

Among them, is the dimension of each Query and Key (usually equal to the feature dimension), which represents the multiplication of the transpose matrix of the Q matrix and the K matrix. Then, the attention score is normalized by the Softmax function to obtain the attention weight.

Finally, use the attention weight to perform a weighted summation of the Value matrix to obtain the output of Self Attention:



Among them, the Softmax color block matrix represents the correlation between each time step and other time steps, and each row of the output matrix represents the sum of the proportion coefficients of all time steps multiplied by the V matrix after Softmax normalization, thereby extracting Key information and features.

* 1. **Multi-Head-Attention**

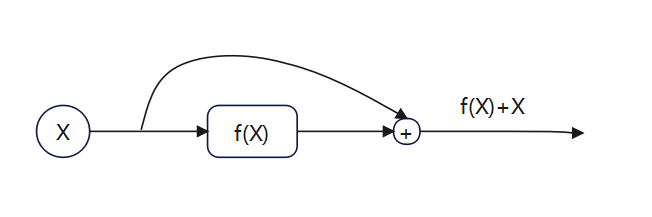
Multi-Head-Attention is connected based on the original single Self-Attention, which contains multiple Self-Attention layers. After obtaining the Q, K, and V matrices, the matrix needs to be split into multiple heads to obtain :

For each header, calculate the Attention-Score value:

For the obtained , the calculated output is:

Finally, concatenate all outputs:

* 1. **Add & Norm, Feed Forward**

****

The function of Add is equivalent to the residual connection to prevent network degradation. Layer Normalization is a normalization technology used in neural networks. It normalizes each feature dimension of each sample so that the mean value of each feature Close to 0, variance close to 1.

The Feed Forward layer corresponds to two fully connected layers. The first layer uses the Relu function for activation, and the second layer does not use the activation function.

* 1. **Prophet**

1. **Overview**

Prophet is a Facebook's open source time series prediction algorithm, which can effectively process holiday information and fit the changing trends of time series data by week, month, and year. According to the official website, Prophet has a good fitting effect on historical data with strong periodic characteristics. It can not only handle the situation where there are some outliers in the time series, but also handle the situation of some missing values. The algorithm provides two implementations based on Python and R.

Judging from the description in the paper, this prophet algorithm is based on time series decomposition and machine learning fitting. The open source tool pyStan is used when fitting the model, so it can get the required results in a faster time. Predicted results.

1. **Model**

The Prophet model uses a decomposable time series model, which is mainly composed of trend term, seasonal term (seasonality) and holiday factors (holidays).

is a trend function, representing non-periodic changing values, represents cyclical changes (such as weekly and yearly seasonality), indicates the impact of holidays that occur on a potentially irregular schedule.

* 1. **trend term**

The trend term is the core component of Prophet, which is used to analyze and fit non-periodic changes in time series. We chose the saturated growth model among them.The saturated growth model does not have an infinite upward trend. When the trend reaches a certain level, it will become saturated, and the saturation value changes dynamically with time.

is the carrying capacity, is the growth rate, is the offset parameter.

* 1. **seasonal term**

represents the periodic change of the time series, which can be used to simulate various periodic change trends such as weekly, monthly, and yearly. It is expressed by Fourier series.

P represents the period, and the parameters can be expressed as.The adjustment of N acts as a low-pass filter. We found that for the annual cycle and weekly cycle, N is better when selected as 10 and 3 respectively.

* 1. **holiday term**

Holidays and important events will have a greater impact on time series forecasting, and these effects are usually predictable. Incorporating these influencing factors into the model as prior knowledge is of major significance to improving the accuracy of the model. represents non-periodic irregular holiday effects. The model implements predictions under holiday or emergency scenarios by customizing the holiday list.

Note that

* 1. **errors term**

represents the part of noise that is not reflected in the model and assumes that the noise factor follows a normal distribution.

* 1. **Advantages**

The Prophet model has good scalability, can be compatible with many aspects of influence, and is closer to life scenes. The obvious nature of the periodic function can provide more effective guidance for the market's replenishment strategy. In addition, this model can also complete missing values, reduce the requirements for data integrity, and is more versatile, making it a relatively good analysis method.

* 1. **Model accuracy criterion**

The experiment adopted the the root mean square error(RMSE) and mean absolute percentage error(MAPE) to evaluate the model results, which are shown below:

Where is a true value, is an estimated value, is the number of index variables.

m categories are listed, so we calculate the average of MAPE given 3 models.

1. Experiments
   1. **Dataset proprocessing and synthesis**

**1）Cost Calculation**

**2）Prophet**

First, load the csv file contained in the initial data set into dataframe format for storage, which is counted as data\_df. Then the data is preliminarily classified according to the classification name, and the two elements of date and purchase quantity are extracted as specified attribute columns. In order to match the data format of the prophet, you need to rename the date column to ds and the purchase quantity column to y, which represent the timestamp and target value respectively.

**3）BiLSTM**

First load a CSV file of raw data , then select specific vegetable categories and attributes to filter the corresponding data. Next, extract the date and specified attribute columns, reset the index, and normalize the values of the attribute columns. The data is organized into a form suitable for LSTM model training by creating a sliding window data set, which includes the input (X\_train, X\_test) and output (Y\_train, Y\_test) of the training and test sets. This ensures that the data format meets the input requirements of the model and provides preparation for subsequent training.

**4）Transformer**

The above code first reloads the CSV file of the original data, then selects specific vegetable categories and attributes, extracts the value of the 'Incoming quantity (kg)' column, and stores it in data\_values .

Next, input and output sequences of training and test data were created based on the size of the history window. The training data uses the data of the first 54 days, and the test data uses the data of the last seven days .

Finally, the training and test data are provided in batches , and the training data is shuffled to ensure that the model sees a different order of samples at each time step .

Therefore, the entire data set is organized into sequential samples based on sliding windows, where each sample includes the sales data of the historical window size (7 days) as input, and the sales volume of the next day as the output label. This organization helps organize the data into a form suitable for time series forecasting models.

* 1. **Analysis of experimental results**

**3.3 Analysis of experimental results**

1. Conclusions

The trend of mass data in power system provides a basis for load characteristic analysis and prediction model establishment, but the classical load forecasting method can not afford such a huge time and computing resource consumption. The problem of over fitting in large sample set will affect the prediction accuracy. In this paper, a power load forecasting model is built by using the BP neural network model, making full use of the powerful data processing function of Clementine and preventing the over fitting function. The experimental results show that the BP neural network model has good predictability and robustness, and has a certain practical application value.

References

1. Schuster M, Paliwal K K. Bidirectional recurrent neural networks[J]. IEEE transactions on Signal Processing, 1997, 45(11): 2673-2681.
2. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735\_1780, 1997.
3. Siami-Namini S, Tavakoli N, Namin A S. The performance of LSTM and BiLSTM in forecasting time series[C]//2019 IEEE International conference on big data (Big Data). IEEE, 2019: 3285-3292.
4. Zeroual A, Harrou F, Dairi A, et al. Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study[J]. Chaos, solitons & fractals, 2020, 140: 110121.
5. Liu Z-x , Zhang D-g , Luo G-z , Lian M , Liu B . A new method of emotional analysis based on CNN–BiLSTM hybrid neural network. Cluster Comput 2020:1–13 .
6. Graves A , Jaitly N , Mohamed A-r . Hybrid speech recognition with deep bidi- rectional LSTM. In: 2013 IEEE workshop on automatic speech recognition and understanding. IEEE; 2013. p. 273–8 .