E-commerce Sales Forecast Based on Neural Network LSTM

Sizhe Zhou

EMAIL: zsz470469575@gmail.com

Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Russia

Abstract: With the advancement of the Internet and the popularization of e-commerce, sales forecasting has become an essential marketing strategy in the online market. This article uses the historical sales data of a Russian e-commerce company to build an LSTM model under the TensorFlow framework.

Keywords: Long Short Term Memory Networks, Sales Forecasting, Convolutional Neural Networks.

1. Introduction to LSTMs

1.1. RNN(Recurrent Neural Networks)

RNNs are a class of extended artificial neural networks that are generated for modeling sequential data. For instance, text is a sequence of letters and words; voice as a sequence of syllables; video as a sequence of images; meteorological observation data and stock trading data, etc., are also serial data. [1] The core idea of RNN: There is a sequential relationship between the samples, with each sample being related to the previous one. Through the expansion of the neural network in time series, the serial correlation between samples can be discovered. Figure 1 shows the structure of the RNN loop body expanded in time. In this figure, A represents the hidden layer, x represents the input of each time period, and h represents the output obtained by each input. At each moment there is an input Xl, and then the current state Al of the RNN provides an output. The current state Al is jointly determined according to the previous state Al-1 and the current input Xl. [2]

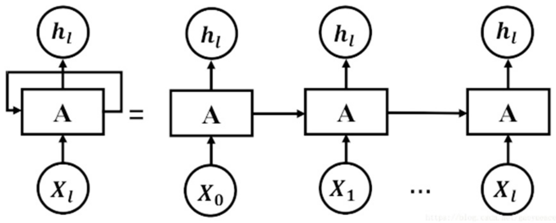


Figure 1: An unrolled recurrent neural network

MSEA 2023, May 26-28, Nanjing, People's Republic of China
Copyright © 2023 EAI
DOI 10.4108/eai.26-5-2023.2334251

1.2. LSTM

LSTM - is a special category of RNN capable of learning long-term dependencies. The full name of LSTM is Long Short Term Memory Networks. As shown in Figure 3, compared with the traditional recurrent neural network, the LSTM has three additional gates: the input gate, the forgetting gate and the output gate, and an internal storage unit in its internal structure. [3]

A simple demonstration of the LSTM process is shown in Figure 4.

In the first step, it is determined which information must be dropped from the Cellular State and is processed by the sigmoid unit of the oblivion gate, which outputs a vector between 0 and 1 by viewing h_{t-1} and x_t information, and the 0-1 values in vectors determine which messages in the cell state c_{t-1} reserved or discarded.[4]

In the second step, decide which new information to add to the cell state. ① Use h_{t-1} and x_t to decide which information to update through the operation of the input gate. ② Use h_{t-1} and x_t to obtain new information on candidate cells C_t through a tanh layer that possibly to be updated with cells information.[4]

In the third step, the original cell c_{t-1} is renewed to the new cell message C_t. The update regulations are: a part of the old cell message is selected by the forgetting gate, and then a part of the candidate cell message $\widetilde{C_t}$ is added by the output gate to get the new cell message C_t.

In the fourth step, after updating the cell state, it's important to determine which state characteristics of the output unit are based on the input h_{t-1} and x_t. Here, it is necessary to transfer the input through a sigmod layer called the output gate to get the determination requirements, then the cell state is obtained as a vector between [-1, 1] by tanh, which is multiplied with the determination requirements derived from the input gate to get the output of the last RNN cell.[4]

The four state functions and outputs are shown in Figure 2 below:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$it = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma(W_0[h_{t-1}, x_t] + b_0)$$
$$h_t = o_t * \tanh(C_t)$$

Figure 2: State functions and outputs

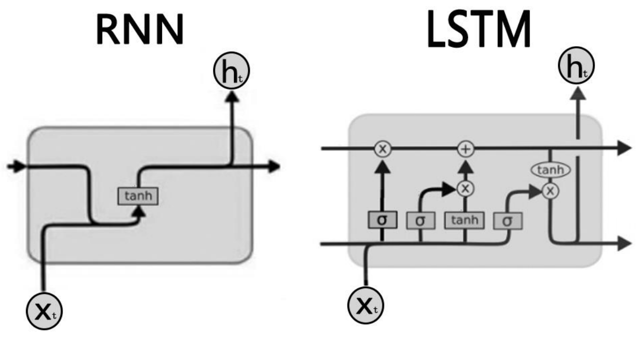


Figure 3: RNN and LSTM

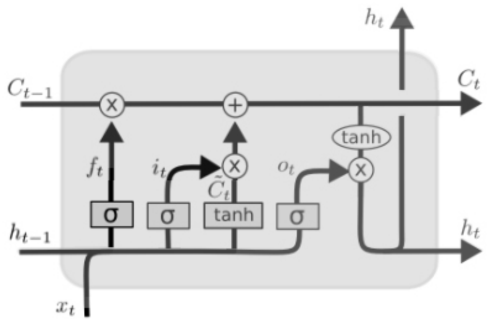


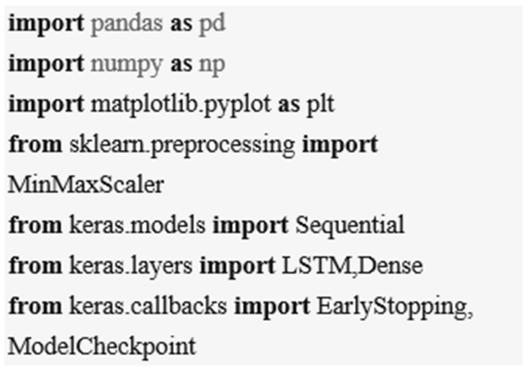
Figure 4: LSTM module

2. Experiment

2.1. Data preprocessing

2.1.1 Import python module, read the information of the file

Import the required modules, as shown in Figure 5:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import
MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM,Dense
from keras.callbacks import EarlyStopping,
ModelCheckpoint
```

Figure 5: Libraries in python

Make a list of documents

```
# read data
sales_train_file = 'data/sales_train.csv'
final_test_file = 'data/test.csv'
category_file = 'data/item_categories.csv'
item_file = 'data/items.csv'
shop_file = 'data/shops.csv'

raw_df = pd.read_csv(sales_train_file)
final_test_df = pd.read_csv(final_test_file)
category_df = pd.read_csv(category_file)
item_df = pd.read_csv(item_file)
shop_df = pd.read_csv(shop_file)
```

The data in raw_df has 6 contents: date-time, date_block_num-increasing by one per month, shop_id-store code, item_id-item code, item_prince-item price, item_cnt_day-item daily sales volume.

Final_test_file has 3 contents, ID-used to submit the final answer, shop_id-store code, item_id-item code.

Category_df has 3 contents, item_category_id and item_category_name, which provide a description of each category ID.

There are 2 contents in shop_df, shop_name and shop_id, which also provide descriptions for each category.

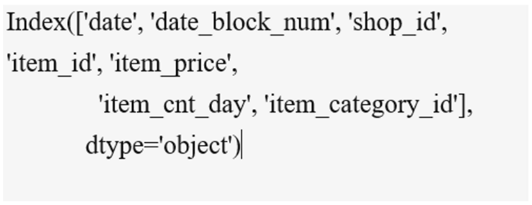
2.1.2 Organizing data

Since the features of the training and testing sets are different, the training set has more features and the sales in the training set are daily data, which have different time dimensions if they need to predict the next month's data. In order to avoid cumulative errors, we choose to extend the features of the test set to make them consistent with those of the training set, first sum the data monthly, the model is then trained to predict monthly sales.

Merge training files .Use the pandas merge function to merge item_df and raw_df, and then remove the item_name column after merging. Convert the date column string to timestamp format to facilitate the processing of time.

```
item_all_df = pd.merge(item_df, category_df, on=['item_category_id'])
print(item_all_df.info())
# handle time data
dt_format = '%d.%m.%Y'
raw_df['date'] = pd.to_datetime(raw_df['date'], format=dt_format)
print(raw_df.head())
print(raw_df.info())
```

The results are shown in Figure 6:



```
Index(['date', 'date_block_num', 'shop_id',
'item_id', 'item_price',
       'item_cnt_day', 'item_category_id'],
      dtype='object')
```

Figure 6: Results

Organize daily sales data into monthly sales data, and expand the training set into a full matrix of shops*items. Monthly sales are merged by date_block_num, since this number represents a particular month. The year and month information is already extracted in the time series feature, and the data is merged by these two data: summing all sales data for a given year and month, but only the average value is required for the price data.

2.1.3 Establishing an all-distance array

Compare the data shape of both

```
print(sales_per_month.shape)
print(final_test_df.shape)
```

The results are as follows

```
(424124, 71)
(214200, 3)
```

Since each row of the current test set and training set represents a combination of shop_id and item_id with different numbers, the data sets need to be merged to create the full distance array.

```
full_shop_item_matrix = pd.DataFrame([])
all_items = item_df[['item_id']]
for shop_id in shop_df['shop_id'].values:
    all_items_per_shop = all_items.copy()
    all_items_per_shop['shop_id'] = shop_id
    if full_shop_item_matrix.shape[0] ==0:
        full_shop_item_matrix = all_items_per_shop
    else:
        full_shop_item_matrix = pd.concat([full_shop_item_matrix, all_items_per_shop],
axis=0)

print(full_shop_item_matrix.shape)

sales_per_month = pd.merge(left=sales_per_month,right=
full_shop_item_matrix,on=['shop_id','item_id'],how='right')

print(sales_per_month.info())
sales_per_month.drop([('item_category_id', '', '')], axis=1, inplace=True)

sales_per_month = pd.merge(left=sales_per_month,
                                    right=item_df[['item_id',
                                                   'item_category_id']],
                                    left_on=[('item_id',
                                              '',
                                              '')],
                                    right_on=['item_id'],
                                    how='right')
sales_per_month[('item_category_id', '', '')
                ] = sales_per_month['item_category_id']
sales_per_month.drop(['item_category_id', 'item_id'], inplace=True, axis=1)
print(sales_per_month.info())

sales_per_month.fillna(value=0, inplace=True)

print(sales_per_month.info())
```

2.2. Feature Engineering

Reasonable feature engineering can substantially improve the effectiveness of the model.

Extract 18 months of sales data and price data as a set of samples.

Roll iteratively through 33 months of data, creating one set of samples at a time.

Combine all samples into one large sample. 4.

Reorder price and sales.

Slice and dice the training and validation groups and the test group.

Normalize the training and validation groups.

Convert the training, validation and test groups into 3D arrays.

The data structure of the training group is as follows.

(19953000, 18, 2)

2.3. Build LSTM model

The LSTM layer is the core layer in keras, and its usage is basically similar to the general Dense layer. The main difference lies in the following points.

input_shape: LSTM needs to input 3D array, usually input (n_steps, n_features), which is apparently 2-dimensional, but actually means that the first dimension (the number of samples) is None, that is, no restriction.

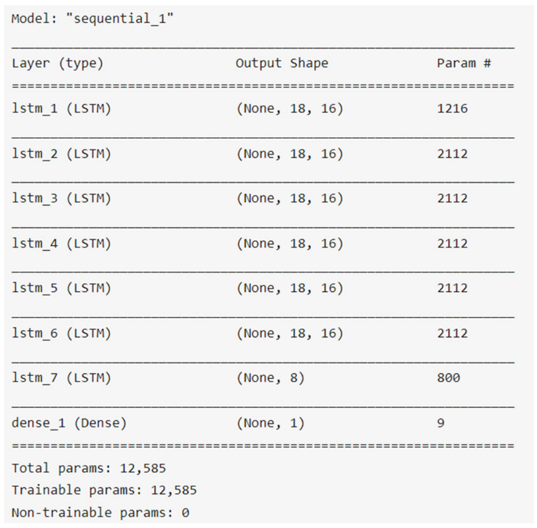
return_sequences: This parameter indicates whether to return the hidden features of each time step (generally denoted by h). As the first input layer, and the intermediate layer, it is chosen to return True. the last LSTM layer is chosen to return False, indicating that only the hidden features of the last step are returned.

The rest of the parameters can be referred to the general Dense layer.

To prevent overfitting, I have activated the dropout scale here. [5]

```
n_hidenlayers = 5
dp_ratio = 0.4

n_steps = 18
n_features = 2
model = Sequential()
model.add(LSTM(16, dropout = dp_ratio,return_sequences=True, input_shape=(n_steps,
n_features)))

#model.add(LSTM(n_hidenfreature, dropout = dp_ratio,return_sequences=False,
input_shape=(n_hours, n_features)))

for _ in range(n_hidenlayers):
    model.add(LSTM(16,dropout = dp_ratio,return_sequences=True))

model.add(LSTM(8,dropout = dp_ratio))
model.add(Dense(1))
model.compile(loss='mse', optimizer='adam')
model.summary()

callbacks = [EarlyStopping(monitor='val_loss', patience=20),
             ModelCheckpoint(filepath='best_model.h5', monitor='val_loss',
save_best_only=True)]

# fit network
history = model.fit(melt_train_X, melt_train_y, epochs=150, batch_size=13302*5, callbacks =
callbacks,validation_data=(melt_validate_X, melt_validate_y), verbose=1, shuffle=False)
# plot history
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()
```

The model is built and the structure of the model is previewed in Figure 7. Notice that the second dimension of the first 5 layers in the output shape are all 18, indicating that the sequence is returned. the last LSTM does not return the sequence, but only the hidden state of the last time step.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 18, 16)            1216
_________________________________________________________________
lstm_2 (LSTM)                (None, 18, 16)            2112
_________________________________________________________________
lstm_3 (LSTM)                (None, 18, 16)            2112
_________________________________________________________________
lstm_4 (LSTM)                (None, 18, 16)            2112
_________________________________________________________________
lstm_5 (LSTM)                (None, 18, 16)            2112
_________________________________________________________________
lstm_6 (LSTM)                (None, 18, 16)            2112
_________________________________________________________________
lstm_7 (LSTM)                (None, 8)                 800
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 9
=================================================================
Total params: 12,585
Trainable params: 12,585
Non-trainable params: 0
```

Figure 7: Structure of the model

2.4. Test Model

Prediction training set, as shown in Figure 8:

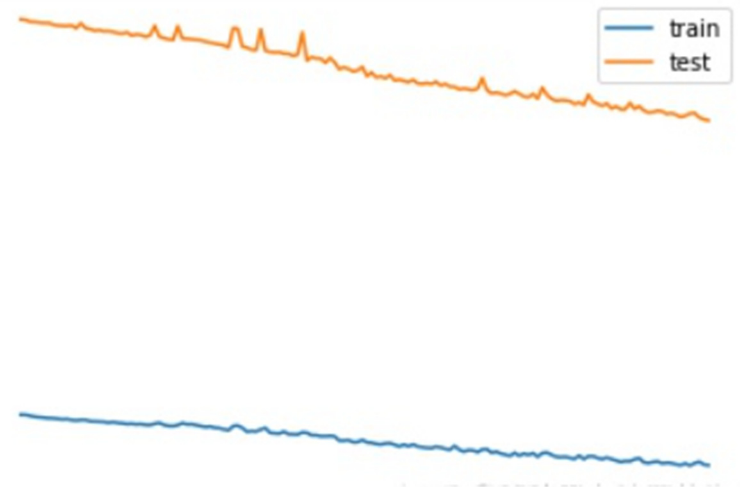


Figure 8: Prediction training set

2.5. Predictive test set

Predict the next month's sales for the full matrix.

Filter out the combinations that appear in the test set.

Output the results.

```
result = shop_item_id.copy()
yhat = model.predict(melta_test_X)
result['item_cnt_month'] = yhat
# melt_test_df.reset_index(inplace=True)
result['item_cnt_month'].plot()
print(result.info())
result = pd.merge(
    final_test_df,
    result,
    left_on=[
        'shop_id',
        'item_id'],
    right_on=[
        'shop_id',
        'item_id'],
    how='left')
print(result.isna().any())
print(result.info())
# result.fillna(0, inplace=True)
# result['ID'] = result.index
result['item_cnt_month'].clip(0,20,inplace=True)

result[['ID', 'item_cnt_month']].to_csv('submission.csv', index=False)
```

3. Conclusion

For e-commerce companies, demand forecasting is the most common and important application problem for most companies in their daily operations. In this paper, we predict the sales data of e-commerce companies by building a neural network LSTM model. However, the LSTM model is more time consuming to train and the error is below 1.

References

[1] Wang X, Liao T, Zhang Shunxiang. Online multi-task sales prediction model based on CNN-LSTM network[J]. Journal of Fuyang Normal University (Natural Science Edition),2021,38(02):85-91.DOI:10.14096/j.cnki.cn34-1069/n/2096-9341(2021)02-0085-07.

[2] Gaoyueace, (2018) A simple framework implementation of recurrent neural network RNN.https://blog.csdn.net/gaoyueace/article/details/80484234

[3] Hu Yuyang,Zhang, Cheng,Zheng, Ming,Xia Dingchun. Design and implementation of LSTM-based sales forecasting system[J]. Computers and Networks,2020,46(23):65-67.

[4] Colah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[5] Hu, Bo-Wen, Li, Jun, (2021) Multi-layer LSTM-based e-commerce merchandising prediction.