Predict Future Sales using LSTM

Zhao Jin, Jiaxu Shao, Kangjing Fan, Tong Xiang

***Abstract*—In this project we propose different neural methods to solve the sales prediction problem. We transfer the original problem into a time series problem where data are organized as sequences and propose different neural network structures with long short term memories. We also provide a data augmentation method for this specific problem, which can further enhance the performance of general models. Our approaches make use of very light-weight structure, thus save huge amount of running time, and perform relatively well compared to other state-of-the-art neural methods.**

***Keywords*—LSTM**

I. INTRODUCTION

Predicting future based on temporal sequences is one of the most significant problem in science and economics. Different examples exist from weather forecasting to predicting the future price of stocks on markets. The desire to know the future is often the driving force behind all these behaviors of predicting and forecasting[1].

Sales prediction is one of the most important problem among predicting future problems. Generally speaking, sales prediction is the process of estimating future sales. Precise sales forecasting enable companies to make better decisions and help company managers to evaluate the overall performance properly. Given data from the past few years, we can treat the sales prediction problem as a time series problem where a series of data point is indexed in time order. It is reasonable to treat the prediction problem as a time series problem as the timing is one of the most important influencing factors in markets.

In recent years different complicate statistical and neural methods have been proposed and applied to forecasting problems. For statistical methods, they have drawbacks, and the most important one is that for each problem the model has to choose a specific assumption on data. To address this kind of problems, some people proposed different neural methods where the models could generate and engineer features by themselves. But traditional neural methods often make use of convolutional neural networks to address this kind of prediction problems, thus lead to missing lots of vital information lies in data. Therefore we propose a series of neural methods where we try to use long short term memories as well as series network structures to capture dependency between data from different years, and make use the extracted information to predict future sales.

II. RELATED WORK

As we dive deeper into this problem, we found that there are various methods used to predict future sales besides LSTM. In this Kaggle competition, many competitors used XGboost to predict future sales, which is an implementation of gradient boosted decision trees. XGboost takes different features as input and doesn't rely much on time series data. And those people who used XGboost tend to achieve a good result in this problem with a good competition ranking. Also, people used CNN, MLP in this problem. Based on their work, we implemented all these methods to compare with LSTM.

III. DATASET

The dataset is given by Kaggle for the competition Predict Future Sales[2].

Training set contains 1048576 entries of data including daily historical sales data from January 2013 to October 2015. Each entry of data also contains information about price of that product, the id of that product etc. Test set contains 214201 entries of data, each entry only contains the corresponding shop id and item id. Overall there are 22171 unique products; they belong to 84 categories and sold by 60 corresponding shops. The training data is unbalanced; roughly half of the products belong to 2 categories.

The rudimentary data fields are listed below.

data	description
ID	an Id that represents a (Shop, Item) tuple within the test set
shop_id	unique identifier of a shop
item_id	unique identifier of a product
item_category_id	unique identifier of item category
item_cnt_day	number of products sold. You are predicting a monthly amount of this measure
item_price	current price of an item
date	date in format dd/mm/yyyy
date_block_num	a consecutive month number, used for convenience. January 2013 is 0, February 2013 is 1,..., October 2015 is 33
item_name	name of item
shop_name	name of shop
item_category_name	name of item category

To get the time series data we need for LSTM, we calculate the monthly sales for every item from daily sales to get 33 month's sales.

In later work, we also calculated the monthly prices of each item together with monthly sales as time series data.

IV. METHOD

Since this is a time series problem, we need to use the sales data of the first 33 months to predict the sales data of the 34th month, then RNN is undoubtedly our first choice. In addition to RNN, we also tried using CNN, MLP and XGBOOST for comparison.

A. *RNN*

We tried different RNN architectures, including three LSTM and one GRU. The output of every method is the same, that is the prediction of sale of 34th month, they are different in two areas: input and model architecture. For each method, we do not have a particularly complex network structure, the summary is shown in the table below.

Model	**Input**	**Architecture**
Basic-LSTM	Series of 33 monthly sale	64 units LSTM + 0.3 Dropout
Multi-feature LSTM	Series of 33 monthly sale and price	64 units LSTM + 0.3 Dropout
LSTM with data division	Series of 10 monthly sale	64 units LSTM + 0.4 Dropout
GRU	Series of 33 monthly sale and price	64 units GRU + 0.4 Dropout

B. *CNN*

For the CNN model we used on convolutional hidden layer followed by a max pooling layer. The filter maps are then flattened before being interpreted by a Dense layer and outputting a prediction. The input are series of 33 monthly sale, and output is the prediction of sale of 34th month.

Also we tried a mix of CNN and LSTM, by using this method, we were hoping that CNN can put the sequences into different subsequences and get some kinds of feature, then use a LSTM to get the final result from these block of time that CNN derived. The input are series of 33 monthly sale, and then use a 1-dimension convolutional layer whose kernel size are 2 to merge sub sequences. Then flatten the output there and input it into LSTM layer with 50 units.

C. *MLP*

This multilayer perceptron model takes in the input of 33 monthly sale and does not take it as sequence. There just a dense layer with 100 units between the input and output.

D. *XGBOOST*

Cause this is not a neural nets so we will not talk much about this. Like MLP, XGBOOST model also does not take the input as sequence. It takes in the every features that we derived from the data as a whole and get a result. Key to this method is how we deal with the features, how to define and how to get them. We learnt this feature-extracting method from the Kaggle open-source notebook. The architecture of this model is rather simple, just take in a 1-dimension feature vector as input and output the result.

V. RESULT

The result is required to clip into [0,20] range and is evaluated by root mean squared error(RMSE). And the results for all methods we tried are shown in the table below.

Method	**RMSE**
Basic-LSTM	1.022
Multi-feature LSTM	1.044
LSTM with data division	1.018
GRU	1.026
CNN	1.123
CNN-LSTM	1.087
MLP	1.099
XGBOOST	0.912

VI. DISCUSSION OF RESULTS

Basic-LSTM and GRU are two simple RNN models and give bland results. We use the results as criteria to evaluate other methods.

A. *Multi-feature LSTM*

In multi-feature LSTM, we add prices as a second feature since sales volume is highly dependent on the price. However, the result is very disappointing. One possible reason is that the time series data is too short, with only 33 months of data, and most of the goods are missing lots of monthly sale data. Even use the 33 monthly prices to predict 34th monthly sale in a lstm model cannot get a convincing result. After realizing this missing data situation, we used three kinds of values to interpolate, which are float('inf'), -1 and the highest price in the past * 1.1. Because the data of missing price will also be 0, according to conventional logic, low- priced sales will high, so for those months with zero sales, we chose to use a larger price to interpolate. The final accuracy rate is still horrible, even worse than the average effect in the last few months. MSE of our result can be over 1000, whereas MSE for the average effect in the last three months is under a hundred. The incompleteness of the price data makes this feature only a burden when we predict sales.

B. *LSTM with data division*

In this method, we make some variance to the original data and the result is a little bit better. By spliting the data, we have more data for training. It's just like rotation augmentation for image data. Even though the length of the each sample is much smaller, it doesn't matter because the result suggests that the patterns of the data in previous years may not have much relationship to the pattern we are looking for. Besides, we are able to focus on the most recent data and prevent the model to learn something from previous years which may not apply to the current year. Although the result is better, the improvement is quite tiny. On the one hand, the amount of data is still small. On the other hand, sales data doesn't only depend on the previous sales data. But we are unable to use features other than sales data efficiently in RNN.

C. MLP

The multilayer perceptron model does not take the sales data as sequences, so what it learns is the pattern of the sales volume as parallel data. It won't focus on the change of data according to time.

D. CNN

Convolutional layer treats the data as many subsequences and will tend to find the patterns of subsequences. Dense layers take those patterns as input and what they do is similar to MLP. Besides, CNN model has much more parameters than other models, which makes it easier for CNN to run into an over-fitting problem.

E. XGBOOST

Tree model has greater performance in this problem. First of all, it is still necessary to say that because the time series is too short and data is missing, the data does not reflect the integrity of the time series and it is difficult to find a fixed pattern. But tree model does not rely much on the integrity of the time series. For example, if when there are only 1,5 and 9 months of data, lstm treats the missing months almost equally as they are really there instead of treat them as missing data, but the tree model can better judge these discrete data, it only learns something from those months which are really exist. Secondly, tree model is not only a vertical comparison on a single product, but can also learn correlation between peer item where lstm is weak in that aspect. Tree model is closer to the shopping link in life and judge the different characteristics of different products, just like people's selection of products, so it can better extract horizontal correlation between peer items, while lstm pays more attention to the chronological change of a single product.

VII. CONCLUSION

According to the analysis, it seems that previous sales data doesn't have too much relationship to the results. What counts is the recent sales data and some other features. Due to the loss of data, it's hard for us to add other features into RNN models. That's why tree model performs much more better than RNN models. For the next steps, the most important thing, in our opinoion, is to make full use of the features. For example, we can classify the items according to other features and then use RNN to make predictions separately. But this requires large amount of data.

REFERENCES

[1] Thiesing, F. M., & Vornberger, O. (1997, June). Sales forecasting using neural networks. In *Proceedings of International Conference on Neural Networks (ICNN'97)* (Vol. 4, pp. 2125-2128). IEEE.

[2] https://www.kaggle.com/dimitreoliveira/deep-learning-for-time-series-forecasting

[3] Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." (1999): 850-855.

[4] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016.

[5] Zheng, Huiting, Jiabin Yuan, and Long Chen. "Short-term load forecasting using EMD-LSTM neural networks with a Xgboost algorithm for feature importance evaluation." *Energies* 10.8 (2017): 1168.

[6] Wang, Jin, et al. "Dimensional sentiment analysis using a regional CNN-LSTM model." *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2016.

[7] Hamilton, James Douglas. *Time series analysis*. Vol. 2. Princeton, NJ: Princeton university press, 1994.