Predict Future Sales

Haoran Zhu

I. Definition

Project Overview

The central problem that this project is designed to accomplish is forecasting future sales. Forecasting sales are extremely essential for retailers to control cost. Depending on the type of retailer, the smaller the profit margin for a given market, the more critical on cost control. Optimizing stock, as the quantity of merchandise available on the premise of the store or warehouse, also means better utilization of shelf space and a greater variety of products to be sold at retail or online for maximizing sales.

Problem Statement

The sales forecasting for the project is based on one of the largest software firms in Russia, 1C Company. With the forecasting report, the company can make the precise decision on procurement to reduce cost and prevent overstocking. The data used for the forecasting is contributed by the Coursera course organizer who created the Kaggle competition "Predict Future Sales". The training data is the historical daily sales record. The sales record is individual entries that contain the date, price, count, product ID, category ID, and shop ID. The approach of solving the problem is to explore the sales data and find a correlation between sales to region and categories of items, then to determine how sale changes in terms of time. The data that contains strings, such as product name, category name, and shop name are all in the original language, Russian, and translation of the string to English may be required for better interpretation.

1. Explore the dataset
2. Clear outliers
3. Reshape the dataset into training data
4. Build LSTM model and train
5. Evaluate the performance
6. Try different architectures

Metrics

Predict the number of products sold in next month and upload the submission file to Kaggle. Kaggle will evaluate it with real sale number by a root mean squared error (RMSE) and return a score (the smaller the better).

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2}$$

Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE should be more useful when large errors are particularly undesirable. Consequently, RMSD is sensitive to outliers.

II. Analysis

Data Exploration

The time-series dataset consisting of daily sales data, kindly provided by one of the largest Russian software firms - 1C Company

File descriptions

- **sales_train.csv** - the training set. Daily historical data from January 2013 to October 2015.
- **test.csv** - the test set. You need to forecast the sales for these shops and products for November 2015.
- **sample_submission.csv** - a sample submission file in the correct format.
- **items.csv** - supplemental information about the items/products.
- **item_categories.csv** - supplemental information about the items categories.
- **shops.csv**- supplemental information about the shops.

Data fields

- **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

sales_train.csv

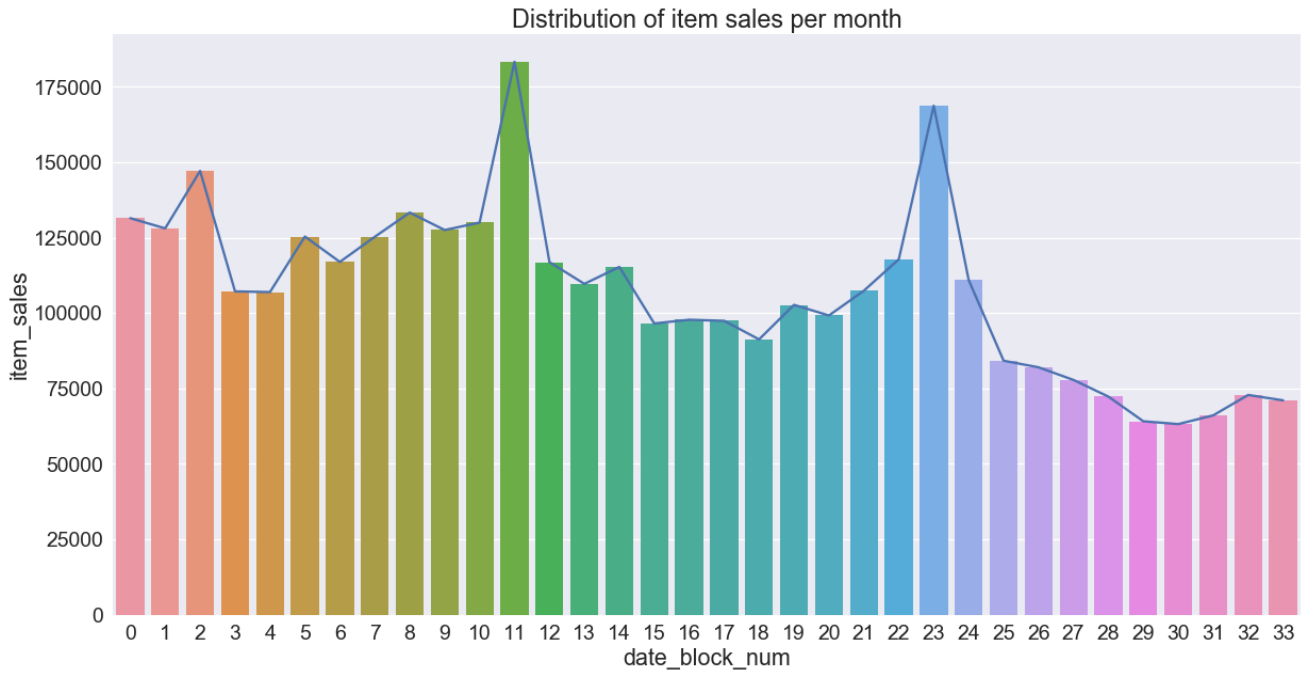
	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	2013-01-02	0	59	22154	999	1
1	2013-01-03	0	25	2552	899	1
2	2013-01-05	0	25	2552	899	-1

	item_price	item_cnt_day
count	2935845	2935845
mean	890.73	1.24
std	1720.15	2.29
min	0	-22
25%	249	1
50%	399	1
75%	999	1
max	50999	100

Exploratory Visualization

The time-series data from 1C Company is composed of 5 datasets with all the items data, item categories data, sales training data, test data, and shops data. Initial exploratory data analysis provide a few understandings on how to find our solution of predicting future sales of the next month for each store and item combinations. Some of these findings include that the data shows us there are 84 unique item categories, 60 stores, and 22,170 items. Our test set has 214,200 entries, and our sales training data has 2,935,849 entries.

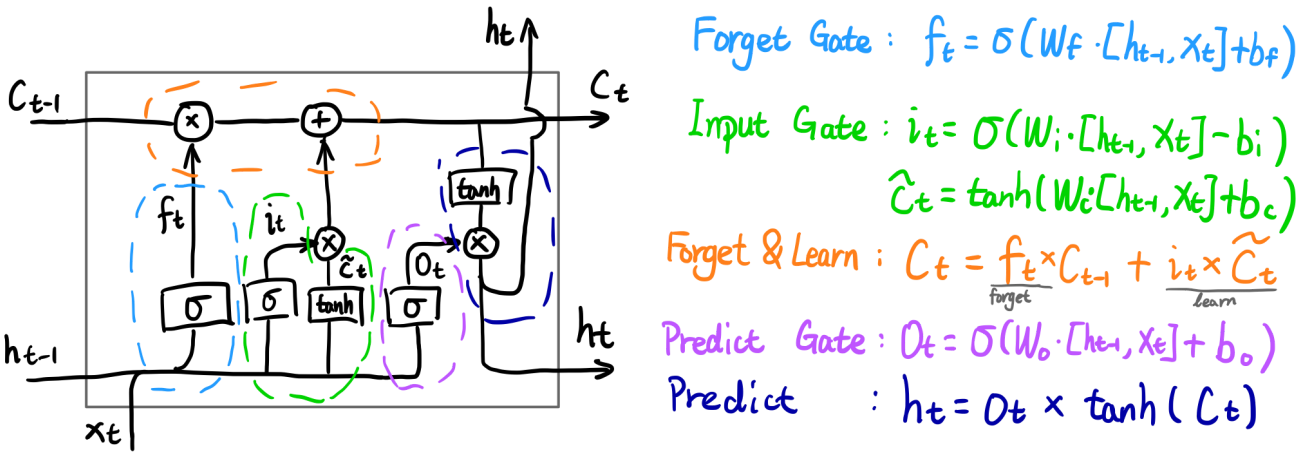
Since our goal is to predict futures sales for the next month using a store-item combination, sales over time of each combination is in a time-series, so we can gain some insights by first computing and plot the total sales per month for the entire company as a function of time before diving into details.



We observe that within the 34 months of our collected sales data, total sales have an overall decreasing trend and that the peaks and valleys indicate a seasonal sales cycle with higher sales during the holiday season every November and December, which indicates that our future sales prediction should increase as it approaches month 35 and 36. Therefore, we can reasonably hypothesize that our prediction for the total sales for every product and store in the next month should follow this pattern and be higher than its preceding months but its peak should be lower than the peaks in month 11 (highest in 2013) and month 22 (second highest).

Algorithms and Techniques

For this problem, the sales contain sequential information of item price and the number of products sold and the target is the prediction of a monthly number of products sold. RNN could use sequential information, and the output being depended on the previous computations. Thus I choose long short-term memory (LSTM), a special kind of recurrent neural network (RNN), to solve this problem. Each LSTM network unit will be responsible for 'remembering' values over arbitrary time intervals to help us with our forecast. We hope that the model is robust enough to tackle situations for our nuanced datasets that have changes for shops and items from month to month.



Benchmark

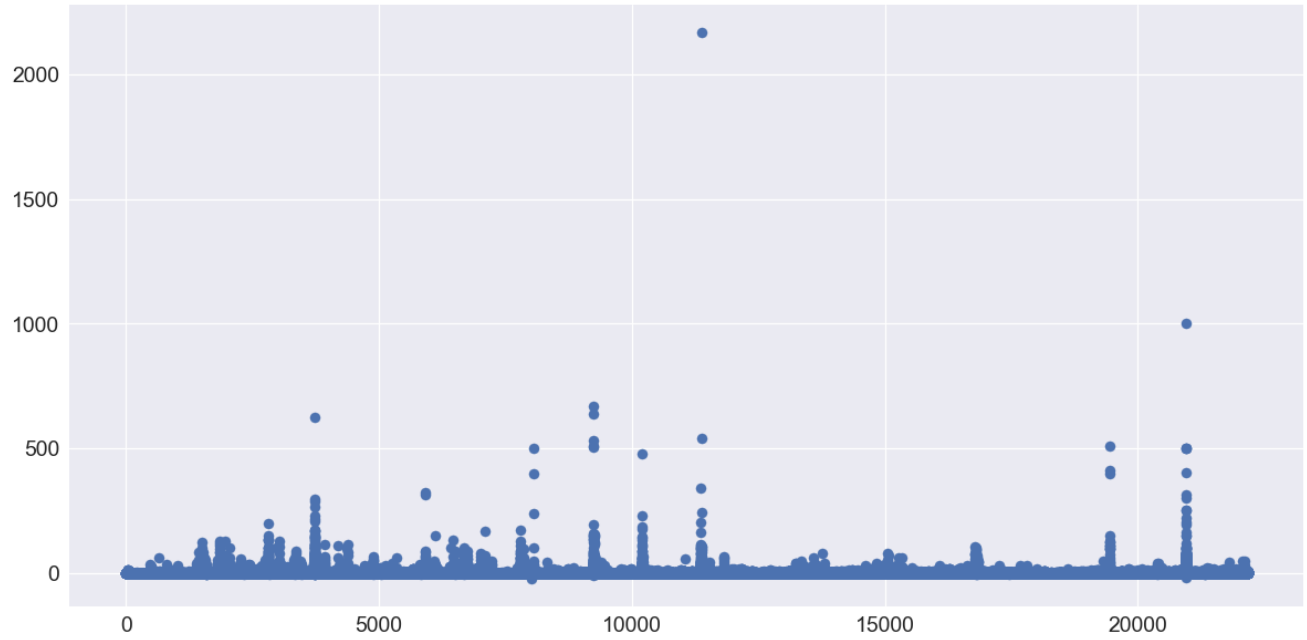
In discussion board, a kaggler shared a solution "Playing in the Sandbox" which is using XGBoost. Its prediction score is 1.19015.

III. Methodology

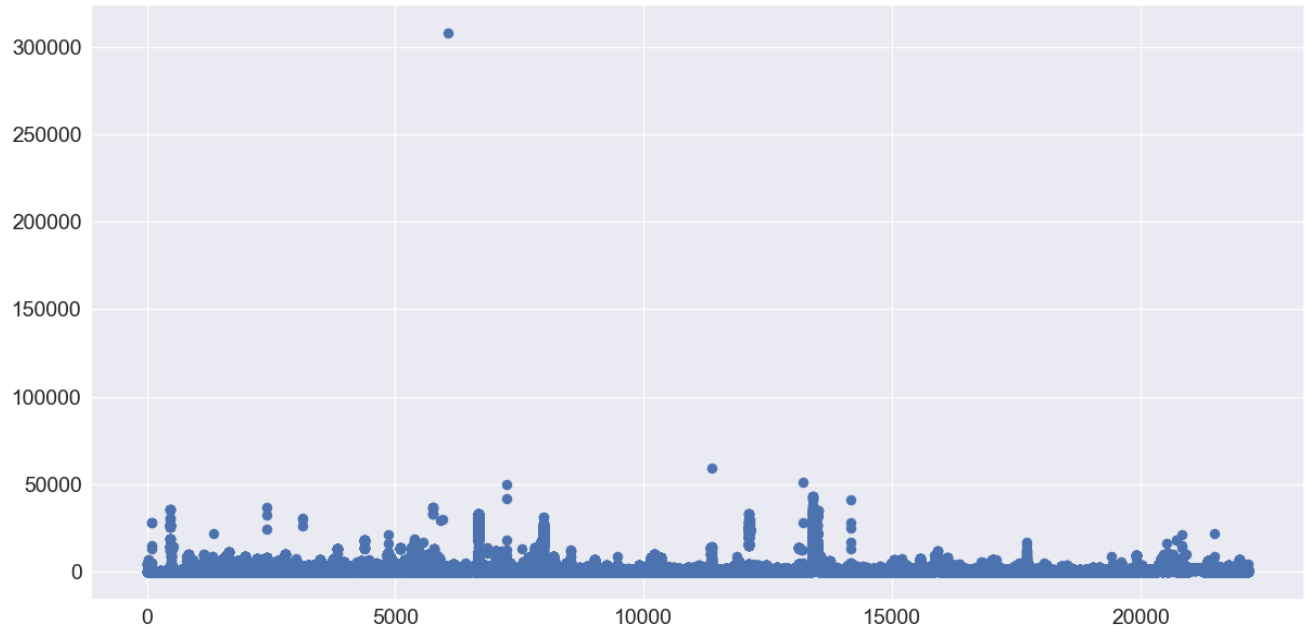
Data Preprocessing

Before we rearrange our data into a new dataframe, monthly sales, we should detect and clear outliers, which have strange sales or prices. Thus, we plot all items' sale and price distribution to check.

Sale Distribution



Price Distribution



Then, we found two strange entries in charts. One has a daily sale larger than 2000, and another one has a price larger than 300000. What is more, there is a third outlier which has a negative price. Thus, we will drop all of these three outliers.

Then we rearrange our data into a new dataframe, monthly sales of shop's items. Since we are using a narrower sequence for our machine learning, we will fill in the resulting empty values with 0.

Implementation

We then build our recurrent neural network units for the training and prediction tasks. Our long short-term memory (LSTM) networks are composed of units that each contain a cell, an input gate, an output gate, and forget gate. Responsible for remembering values over arbitrary time intervals, which should help us with our time-series data that contain changes in shop and item categories from month to month.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 33, 64)            16896
_________________________________________________________________
lstm_2 (LSTM)                (None, 33, 64)            33024
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                33024
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 83,009
Trainable params: 83,009
Non-trainable params: 0
_________________________________________________________________
```

We use early stop technology to prevent overfitting and save running time. We randomly choose 20% most recent training data for cross-validation data to keep our testing set and validation set as similar as possible. The early stop technology will focus on cross-validation loss and find the smallest one. To avoid stopping at a local minimum, we will keep training three more epochs when cross-validation loss stops decreasing.

Refinement

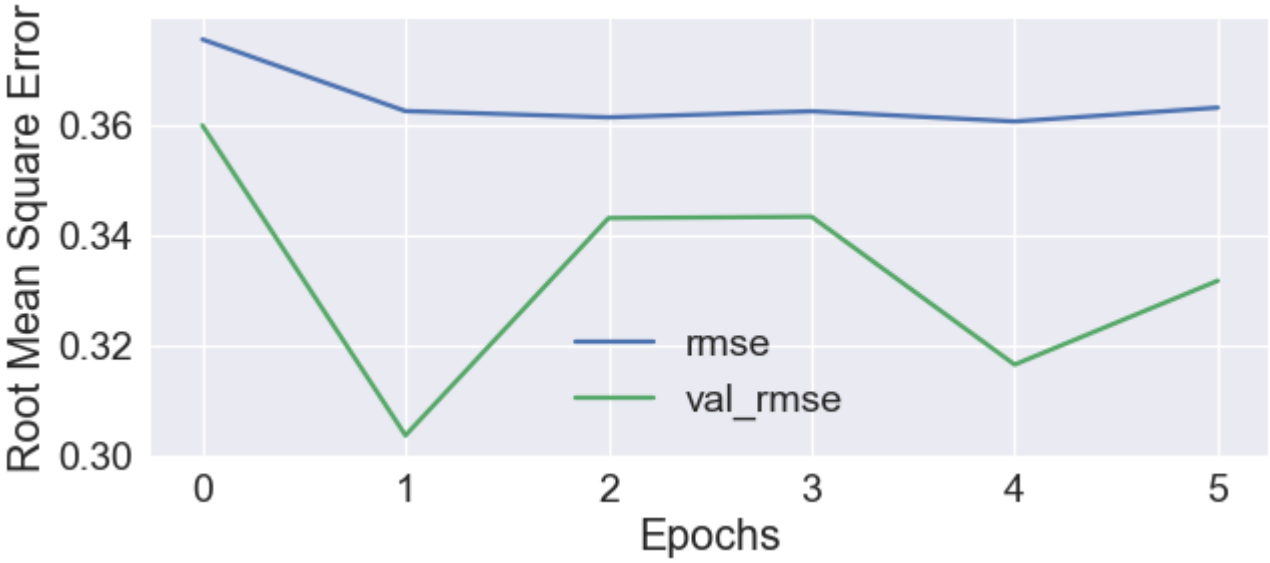
Score	Node	Dropout	Epoch	Validation	Seq Length
1.04441	100		5		33
1.02193	100	0.5	5		33
1.03346	128	0.5	5		33
1.02235	32/32/32	0.5	5		33
1.02311	32/64/32	0.5	5		33
1.02194	64/64	0.5	5		33
1.02143	64/64	0.5		0.1	33
1.01903	64/64	0.5		0.1	30
1.01885	64/64	0.5		0.1	28

We tried to apply normalization and add price feature on the model, but both of them decreased the performance.

IV. Results

Model Evaluation and Validation

After building our LSTM model, we will now finally able to evaluate all of our operations and analysis so far by computing the root mean squared error for our model above using 5 epochs. Our resulting scores are:



- Train RMSE: 0.3627
- Test RMSE: 0.3036
- Kaggle Score: 1.01885 (Rank = 179 / 817)

On epoch 1, we have the best model. Before epoch 1, the test score and training score is decreasing, which means the model is underfitting. After epoch 1, the test score stays at 0.36 and test score is increasing, which means the model is overfitting.

Justification

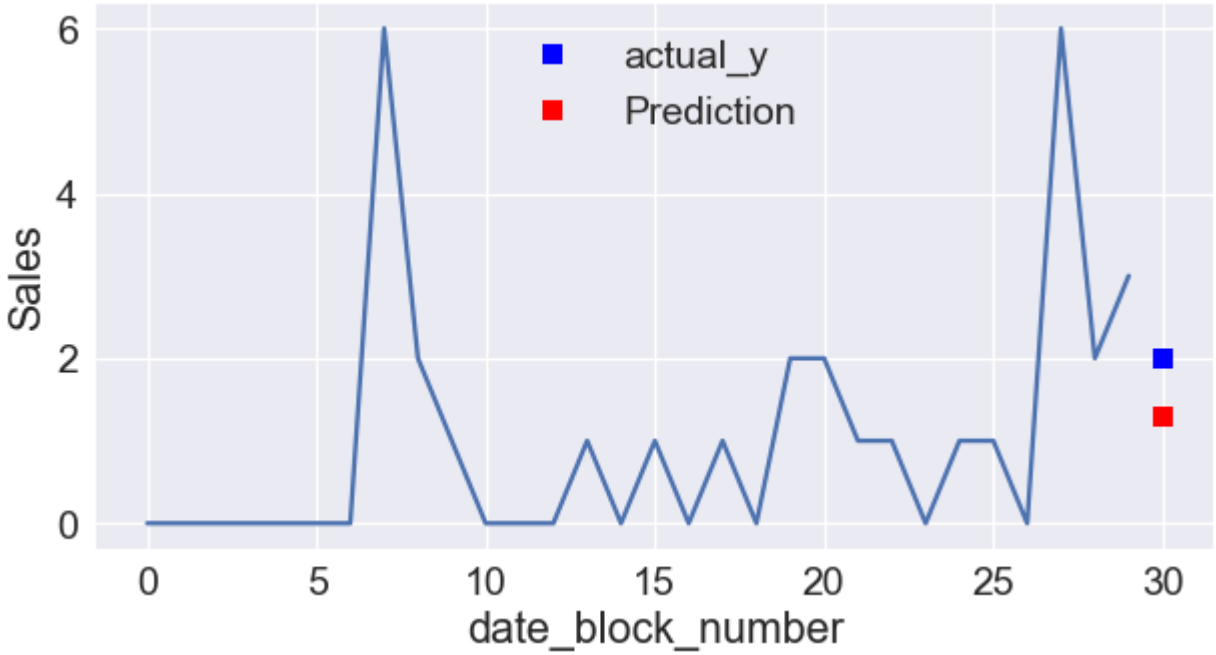
Kaggle	Benchmark XGBoost	My LSTM
Score	1.19015	1.01885
Rank	439 / 817	179 / 817

Our LSTM has much better performance than the benchmark's XGBoost.

Conclusion

Free-Form Visualization

A Cross-Validation Sample of Sales History and Prediction



Even though the sales increases on block 29, the model correctly predicts than it will decrease on block 30.

Reflection

We have trained an LSTM model using past sales to predict future sales. Because we got to rank 179 of 817 on Kaggle, our model is pretty good at predicting future sales.

Difficult parts

- Design the LSTM model. The nodes in LSTM is not the more the better. Too complex LSTM model not only requires longer training time but also has worse performance.
- Choose appropriate feature. We tried to add item price or item category as features, but neither has better performance than previous.

Improvement

Even though the results are promising, one of the next steps of this project is to add more feature to the training set. This step can be improved by deeply exploring the data set.

Could look into using Time Series KFold CV and show the different folds scores.