Capstone Project Lucas Kuttner Amin

Machine Learning Engineer Nanodegree

July, 10, 2018

Sales Forecast prediction

I. Definition

Companies often finds itself allocating many resources on unnecessary segments, executing bad investments, enduring declining sales and lost opportunities for promotions and product releases because they don't have the proper data and future predictability.

Sales forecasting allows companies to execute all the tasks that need future data and actions that need predictability. It is the process of predicting what a certain team, salesperson or company will sell in any given future time period. They are used mainly by company managers and directors to anticipate decisions like hiring, resource management, budgeting and available working capital.

Companies usually build their sales forecast based on manually on excel or any other financial tool, through many hypotheses, analysis and even competition sales information, choosing between bottom-up or top-down approaches, based on their business plan.

Project Overview

The objective of this project is to learn, test and analyze two sales forecasting solutions, based on machine learning. These are: Random forests and LSTM (long short term memory) networks. Including complexity, train and classifying time and general results.

Using machine learning to execute sales forecasting makes the process faster and more reliable, even if it is used only as a support for the sales forecast team. This problem was proposed at Kaggle, at the competition "Predict Future Sales". The problem may be divided into several relevant features extracted from past data, making analysis of the achieved results based on success rate and mean squared errors.

This project has been retrieved from a Kaggle's website competition, here. Kaggle disposes introductive materials and knowledge rich kernels to learn from. Plus the necessity of the main market to reliable sales forecast, I've chosen this problem to solve as capstone project.

The dataset is built from daily historical sales data, with item categories, shops and sales archives, each with its supportive file (Test data or supplemental information). The general information provided is helpful for predicting future sales, using item prices, quantity sold, date, categories and ids.

Problem Statement

The objective of this capstone project is to test and compare two solutions to the sales forecasting problem through machine learning techniques of predictive analysis. The predicted feature of this dataset will be the quantity of items sold in any given month.

The execution consists of analysis of two supervisioned learning models and comparing their results of train time, test time, mean square error and complexity involved. The analyzed models are heavily based in Playing in the sandbox and Sales Forecast LSTM kernel implementations.

It has been analyzed a model that uses the algorithm random forests and compared its results to a LSTM algorithm, given analysis of its requirements, metrics and performance, searching for a conclusion for the best spot to use each model.

Metrics

The results will be analyzed on the 33rd month of data at both random forest and LSTM models, retrieving its accuracy using mean squared error.

The accuracy evaluation metric will be the same as used at Kaggle's competition; root mean squared error at the test set. This is the used evaluation because it measures precisely the average of the squares of the errors, which is the difference between the estimator and the result.

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

The other metrics used for comparison are train time, test time and a general result, comparing general uses and use cases using this project and available information of the models.

II. Analysis

Project Design

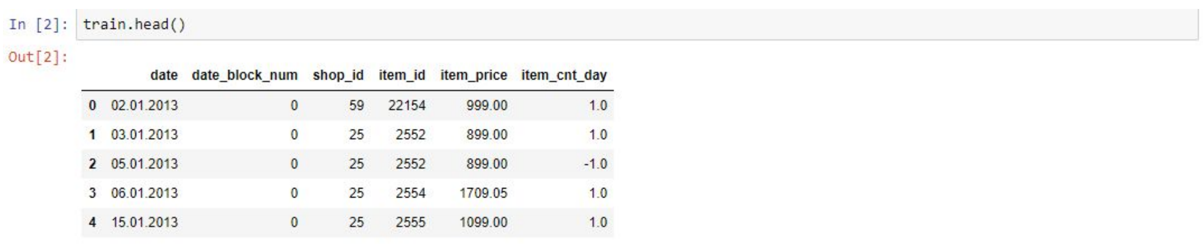
The approach begins analyzing the data and extracting insights from it, using cumulative variance, mean values, std, analysis of the time-series behavior of all graphics. Second, investigating the results of a random forest model approach, setting up graphics and score information. Lastly, I explore and document a long term short memory neural network approach.

The training file has 2935849 rows, separated futurely in validation and test subsets for fitting and RMSE scoring for both models.

Data Exploration

The data set used for this project is present at this link. It is composed by 6 files: Train, test, submission, items (relation between items and categories), item_cats (item categories) and all items shops, representing a total of 9 features (date, date_block_num, shop_id, item_id, item_price, item_name, item_category, item_category_id and shop_name). However, the used files excludes submission and test, as the test data will be provided by the train file, splitting it on train, validation and test data.

The main learning data is the item_id, shop_id and date_block_num in relation to the number of items sold in that day (item_cnt_day). This way some of the labels from files, such shop, item and category names and price can be ignored, due to their absence of relevant information for the models to learn. The train data format is shown at figure 1.



```
In [2]: train.head()
Out[2]:
```

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.00	1.0
1	03.01.2013	0	25	2552	899.00	1.0
2	05.01.2013	0	25	2552	899.00	-1.0
3	06.01.2013	0	25	2554	1709.05	1.0
4	15.01.2013	0	25	2555	1099.00	1.0

Figure 1 - Train dataset initial data

Exploratory Visualization

The data analysis consists of getting insights from graphical representations of data. The figure 2 shows the graph that contains the mean quantity of sales among all shops which have enough data points, showing a clear descent of sales combined into a seasonality of items sold.

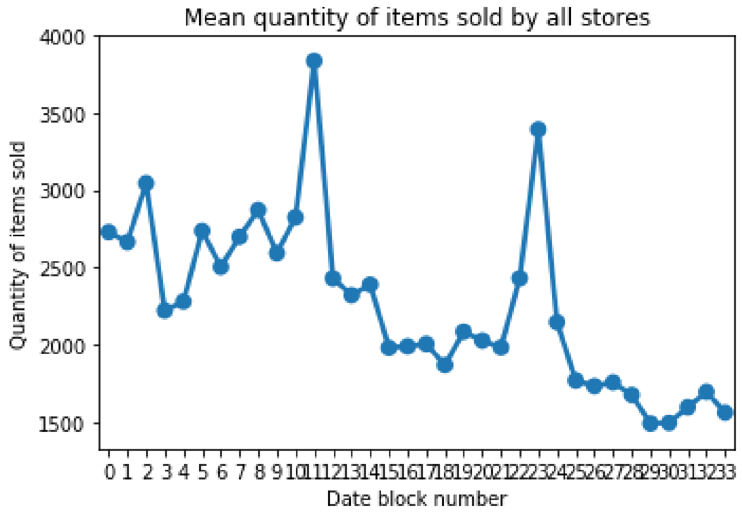


Figure 2 - Graph showing mean quantity of all items sold

This graphic shows a necessity of predicting data with patterns of seasonality, which may not be possible for all machine learning algorithms. This project will analyze this feature in relation to a LSTM and a random forest machine learning model.

Algorithms and Techniques

This project has the intent to analyze and conclude about the use of LSTM network compared to random forest on time-series data particular to the sales forecasting environment.

Because the problem analyzed has data that can be treated as time series and it shows seasonality, different kinds of sources and items, there is space for good results towards algorithms that can learn from all these features.

LSTM

Simple implementations like simple neural networks (low number of neurons and layers) or a simple support vector machine should not solve this problem successfully. LSTM network should perform much better than these, as it has short term memory between the training data, so it is capable of analyzing and predicting data considering seasonality and repeating patterns.

A LSTM network is a particular type of recurrent neural network. A RNN is a neural network that is capable to learn sequence of data, memorizing patterns, seasonality and repetitions. The key difference between a LSTM network and a normal Recurrent neural network is that the first is designed to store information for a long time, avoiding a problem called “The vanishing gradient problem”. This is done by the inclusion of a memory cell in addition of the normal RNN neurons, which are capable of controlling information retrieval.

Random Forest

Random forest is implemented through multiple decision trees models, connected by a common number of classifications and questions through all the used tree models. It will be used due to its simple algorithm and easy to tune and refine model

Benchmark

RSME

The accuracy analysis benchmark has been made using sklearn`s mean_squared_error function to determine the root main squared errors between the predicted results and the provided test data using the provided data.

The comparison results between random forest and LSTM algorithms have been far from the predicted initially, at this project proposal. Even though LSTM has less data for testing predicting (it uses a single shop and item for training), it showed a far worse result

(1.29) on RMSE when compared to the random forest result, which achieved an RMSE value of 0.023.

Duration

LSTM model is expected to be far shorter than random forest, due to its data specification of only a shop and a specific item, resulting in fewer data for training.

	LSTM	Random forest
Train time	4.75 seconds	27.3 seconds
Predict time	0.18 seconds	0.10 seconds

Table 1: displays the fitting and predicting duration for both models:

III. Methodology

Data Preprocessing

Preprocessing was needed for both the algorithms due to the unnecessary number of features on the dataset, and possibility for improvement on the computational power (scaling and features removal).

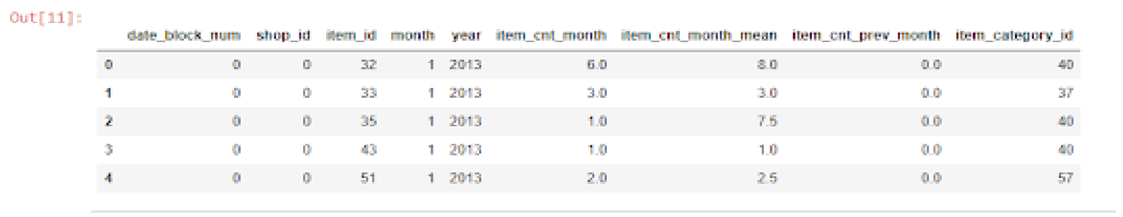
LSTM

The dynamic temporal behavior shown in the LSTM nodes helps to catch memory and seasonality needed onto training the timeseries type of data this problem has. The LSTM implementation I've chosen for comparison is capable to train only a defined shop_id and item_id, making a more reliable training, when compared to the full shop_id and item_id prediction model used in random forest. After removing all unnecessary data for the chosen model, only shop_id, item_id and the date block number were used.

After the features were extracted, the code has a process of scaling the data for improved computational speed then reshaping for adequacy with the used model.

Random forest

The random forest algorithm extracts its training data from the date_block_num information (which month was used, from 0 to 33), turning it into month and years features. Then, I add the mean of a given item count for a given month for each (shop_id, category_id) combination to each row of the dataframe, resulting in the figure 3.



Out[11]:

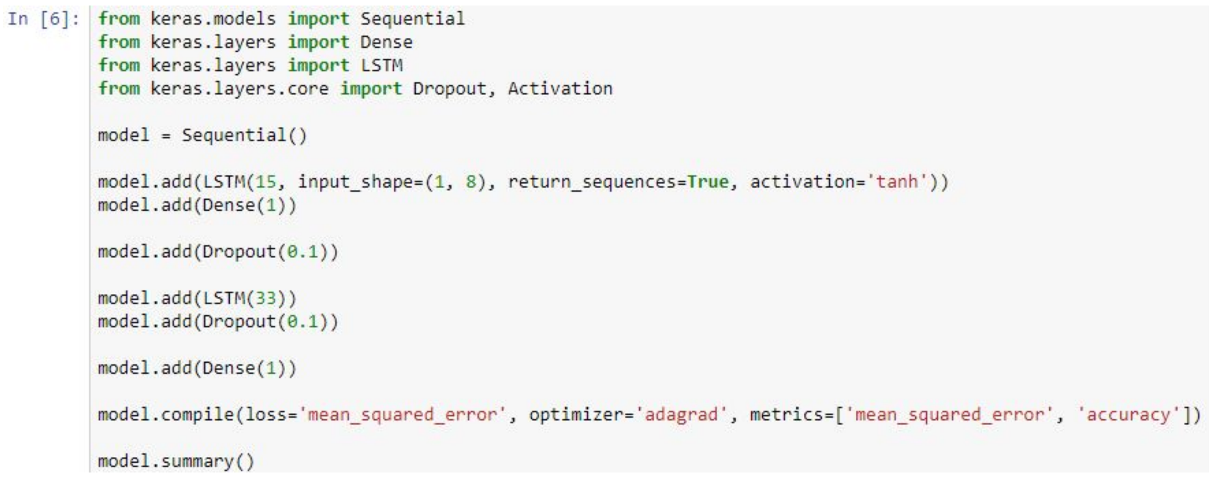
	date_block_num	shop_id	item_id	month	year	item_cnt_month	item_cnt_month_mean	item_cnt_prev_month	item_category_id
0	0	0	32	1	2013	6.0	8.0	0.0	40
1	0	0	33	1	2013	3.0	3.0	0.0	37
2	0	0	35	1	2013	1.0	7.5	0.0	40
3	0	0	43	1	2013	1.0	1.0	0.0	40
4	0	0	51	1	2013	2.0	2.5	0.0	57

Figure 3 - Used train data structure for the created random forest model.

Implementation

LSTM

The LSTM algorithm is implemented using the Keras library and dependencies, using 2 dense keras layers and a LSTM layer into the network. The exact network implementation is shown at figure 4.



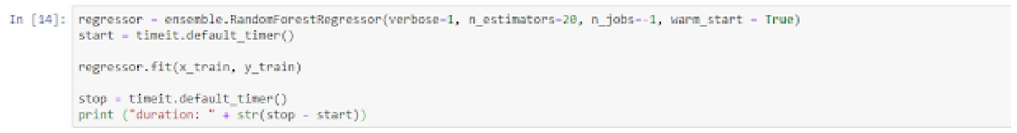
```
In [6]: from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import LSTM
        from keras.layers.core import Dropout, Activation

        model = Sequential()

        model.add(LSTM(15, input_shape=(1, 8), return_sequences=True, activation='tanh'))
        model.add(Dense(1))

        model.add(Dropout(0.1))

        model.add(LSTM(33))
        model.add(Dropout(0.1))

        model.add(Dense(1))

        model.compile(loss='mean_squared_error', optimizer='adagrad', metrics=['mean_squared_error', 'accuracy'])

        model.summary()
```

Figure 4 - LSTM network instantiation with python code

This specific layer format was achieved empirically, comparing train time and root main squared error achieved. I`ve experimented with several number of layers (It caused heavy overfitting) and only a single layer of LSTM network, which caused underfitting. The 'adagrad' optimizer was used to diminish the influence of learning rate parameter, as it adjusts this value while the model is trained.

Random Forest

The random forest model was built using sklearn library, using a RandomTreesRegressor ensemble. The parameters used were first obtained following the used ensemble defaults and then tweaked, incrementing or decrementing the number of estimators and max tree depths. The random forest implementation is at figure 5.



```
In [14]: regressor = ensemble.RandomForestRegressor(verbose=1, n_estimators=20, n_jobs=-1, warm_start = True)
         start = timeit.default_timer()

         regressor.fit(x_train, y_train)

         stop = timeit.default_timer()
         print ("duration: " + str(stop - start))
```

Figure 5 - Random forest implementation using python code

Refinement

LSTM

The LSTM implementation achieved, shown at figure 4, was based on this kernel and optimized for the complex data structure of sales forecasting by inserting more layers to the network. Then, setting the optimizer for adagrad and activation function for tanh after multiple tries with several configurations.

It has been tried to adapt the LSTM base code to work with multiple shop and item IDS. The achieved result was weak, all the predicted values were a mean value of the input data. The result of refinement caused a change of a RMSE value of 1.7 to 1.29.

Random Forest

The random forest algorithm implemented was far easier to refine, by punctual changes like: Removing maximum tree depth, after testing without success max_depth values of 5, 10 and then 15, as this dataset seems to have complex date patterns, so the depth should not have a maximum. Plus raising the number of simultaneous estimators for speed.

IV. Results

Model Evaluation and Validation

Long short-term memory network

The achieved LSTM model is not ideal due to its limitation on a specific shop and item to be trained. And even with this limitation, it performs worse at root main squared error of 1.29 when compared to random forest (0.02). This model is not robust enough to solve the problem as expected during the project proposal.

Random Forest

The random forest algorithm however showed itself a proper solution for the prediction of sales, showing a low RMSE value. This random forest implementation might be a good solution to treat the sales forecast problem. A final time consumption graph between both models is shown at figure 6.

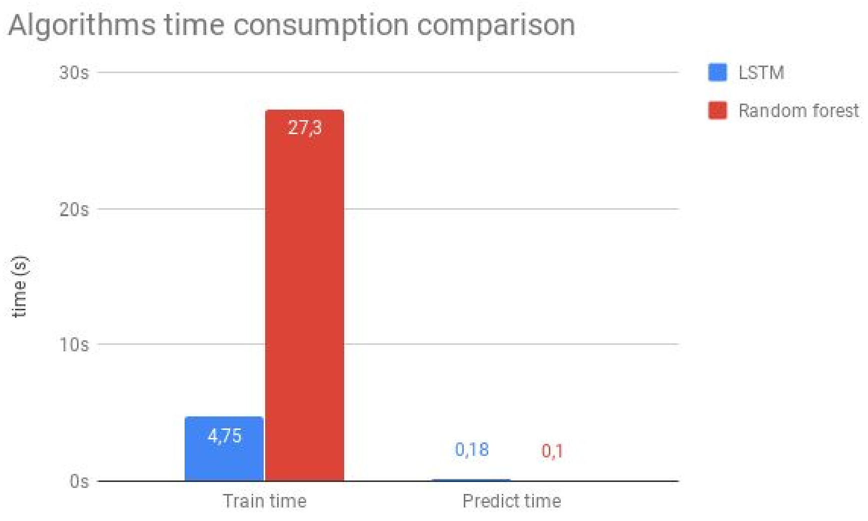


Figure 6 - Time consumption between random forest and LSTM implementations

Justification

The comparison between the two implementations is easy analyzing the results, random forest is much more general (it can analyze multiple shop and items simultaneously), faster and more accurate than the LSTM network, given the RMSE and time results.

The random forest implementation, with a given dataset, is capable of forecasting future sales successfully. Meanwhile, LSTM needs to be studied in other datasets or different implementations before it can be used in a reliable way for this problem.

V. Conclusion

Free-Form Visualization

An interesting aspect of this project can be seen as the comparison between long short-term memory networks and random forest, both applied to a time series problem. With some research and along with this project used models, the random forests algorithm is simpler. It should be used to problems that need to be solved quickly and don't need much work of tuning multiple and complex hyperparameters. On the contrary, LSTM networks show themselves in a complex and work intensive manner. From the data preprocessing to the number of network layers, optimizers and activation functions, LSTM shows a bigger room for improvement and tuning in a specific problem, but it demands attention and knowledge.

Reflection

This project main difficulty was the lack of knowledge of LSTM and time series data, from its preprocessing to ways which it can change results. Luckily this project had a lot of inspiration and content to learn from at the Kaggle website, where I learned how LSTM works, how to implement, data preprocessing and used libraries.

Preprocessing was fairly simple, mainly removing seemingly unnecessary features and scaling the used features for computational purposes.

Implementation was the worst part, mainly for LSTM, where I tried my best implement a network that can decently fit and predict the used dataset and only achieved underfitting models. This model didn't fit my final expectations for the problem, as it is more complex than random forest, took much more work and study and it showed a somehow dim result.

Refinement showed itself a clear section of the work, where it resumed on tweaking hyperparameters, ratio of test and validation datasets and number of layers at the LSTM network.

Improvement

The major improvement would be on the LSTM network, working towards it to train and predict using multiple shop and item ids. When achieved, there should be plenty space for tuning for hyperparameters and data preprocessing towards this same approach.

Another improvement would be testing the influence of each feature on both Random Forest and LSTM network, with the purpose of feature removal and scaling data preprocessing.

References

- https://blog.hubspot.com/sales/sales-forecasting
- https://trackmaven.com/marketing-dictionary/sales-forecasting/
- https://www.thebusinessplanshop.com/blog/en/entry/how_to_forecast_sales

- https://machinelearningmastery.com/challenging-machine-learning-time-series-forecasting-problems/
- https://www.kdnuggets.com/2017/05/springml-sales-forecasting-using-machine-learning.html
- https://rstudio-pubs-static.s3.amazonaws.com/105869_f6e7f8d4e0434c40bd939a3d1e792af9.html
- http://www.insightsquared.com/2013/05/why-data-driven-sales-forecasts-are-important/
- https://deeplearning4j.org/lstm.html
- https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Support Kaggle kernels:

- https://www.kaggle.com/the1owl/playing-in-the-sandbox
- https://www.kaggle.com/carmnejsu/sales-forecast-lstm-67-beginner-friendly
- https://www.kaggle.com/jagangupta/time-series-basics-exploring-traditional-ts
- https://www.kaggle.com/anqitu/feature-engineer-and-model-ensemble-top-10
- https://www.kaggle.com/minhtriet/a-beginner-guide-for-sale-data-prediction