Using Feedforward neural network to solve a Sales prediction problem

Marios Koletsis
May 27, 2017

I. Definition

Project Overview

Time series prediction problems are a difficult type of predictive modeling problem. Unlike regression predictive modeling, time series also adds the complexity of sequence dependence among the input variables. The biggest difference with a regular regression problem is:

- It is time dependent.
- Seasonality is important.

Time series are found in many areas including communication, health, finance and general business for revenue prediction, as is the present work.

The present, is a time series problem related project based on a competition from kaggle, and the main challenge is to predict the daily sales of 1115 stores for the German drugstore chain Rossmann for a 6 weeks period (2015/06/14 to 2015/07/31) using past data provided by the store from the period 2013/01/01 to 2015/06/13.

Problem Statement

The problem we are going to deal in this project is revenue prediction for the second largest drugstore chain in Germany with over 3,000 drug stores in 7 European countries. Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. Specifically the target is to predict 6 weeks of daily sales for 1,115 stores located across Germany.

In order to tackle the problem, the original data provided as part of the competition (store info and sales data) is going to be used and there are consideration that 3rd party data like from google trends and Facebook could also add on the accuracy of the prediction model since is heavily suggested as part of the competition. Since the objective is to predict the daily sales for the next 6 weeks we have a time series problem the initial consideration was to use RNN (recurrent neural networks) and LSTMs (long sort term memory) in specific through Keras backed by Tensorflow on the technology side.
After some testing and some research, the vast majority of LSTM related models I have discovered, was based on predicting specific classes like words in a sentence and there were very few literature suggesting success in regression problems. Furthermore in terms of practical implementations, there were mostly introductory examples with very little success and predictive power for long term cases like in ours (6 weeks). For this reasons I decided to

try and tackle the problem with traditional FFNN (feed forward neural networks).

Metrics

Having a time series regression problem, evaluation is based on the Root Mean Square Percentage Error RMSPE as it is suggested by the literature (Biddy and Toutenburg, 1977) and because it is also suggested by the competition it self. The last part also allowed an accurate benchmark for the overall performance of the model created.
The RMSPE is calculated as below:

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2},$$

II. Analysis

Data Exploration

The data as provided is described as below:

Historical sales data for 1,115 Rossmann stores with which the objective is to forecast the "Sales" column. A further challenge is that within the time period the data was collected, some stores were temporarily closed for refurbishment.

It mainly consists of 4 files:

- train.csv - historical data including Sales
- test.csv - historical data excluding Sales
- sample_submission.csv - a sample submission file in the correct format
- store.csv - supplemental information about the stores

Following are the descriptions for the fields included in all of the files:

1. Id - an Id that represents a (Store, Date) duple within the test set
2. Store - a unique Id for each store
3. Sales - the turnover for any given day (this is what you are predicting)
4. Customers - the number of customers on a given day
5. Open - an indicator for whether the store was open: 0 = closed, 1 = open
6. StateHoliday - indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = None
7. SchoolHoliday - indicates if the (Store, Date) was affected by the closure of public schools
8. StoreType - differentiates between 4 different store models: a, b, c, d

9. Assortment - describes an assortment level: a = basic, b = extra, c = extended
10. Competition Distance - distance in meters to the nearest competitor store
11. Competition Open Since [Month/Year] - gives the approximate year and month of the time the nearest competitor was opened
12. Promo - indicates whether a store is running a promo on that day
13. Promo2 - Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not participating, 1 = store is participating
14. Promo Since [Year/Week] - describes the year and calendar week when the store started participating in Promo2
15. Promo Interval - describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store.

Data Overview:

The first 5 lines of the train.csv

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	15/07/31	5263	555	1	1	0	1
1	2	5	15/07/31	6064	625	1	1	0	1
2	3	5	15/07/31	8314	821	1	1	0	1
3	4	5	15/07/31	13995	1498	1	1	0	1
4	5	5	15/07/31	4822	559	1	1	0	1

The first 5 lines of the test.csv

	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
0	1	1	4	15/09/17	1	1	0	0
1	2	3	4	15/09/17	1	1	0	0
2	3	7	4	15/09/17	1	1	0	0
3	4	8	4	15/09/17	1	1	0	0
4	5	9	4	15/09/17	1	1	0	0

What worth mentioning regarding the test.csv file is that not only Sales as a field to predict is missing but also the customers field as well which makes sense since the correlation among the feature is expected to be too strong plus on a real life settings, there is no way the store would know the number of customers for the next 6 weeks.

Details for the main and some engineered features that are expected to play a crucial part follows in the Exploratory Visualization section.

The first 5 lines of the store.csv

Store	Store Type	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
1	c	a	1270	9	2008	0			
2	a	a	570	11	2007	1	13	2010	Jan,Apr,Jul,Oct
3	a	a	14130	12	2006	1	14	2011	Jan,Apr,Jul,Oct
4	c	c	620	9	2009	0			
5	a	a	29910	4	2015	0			

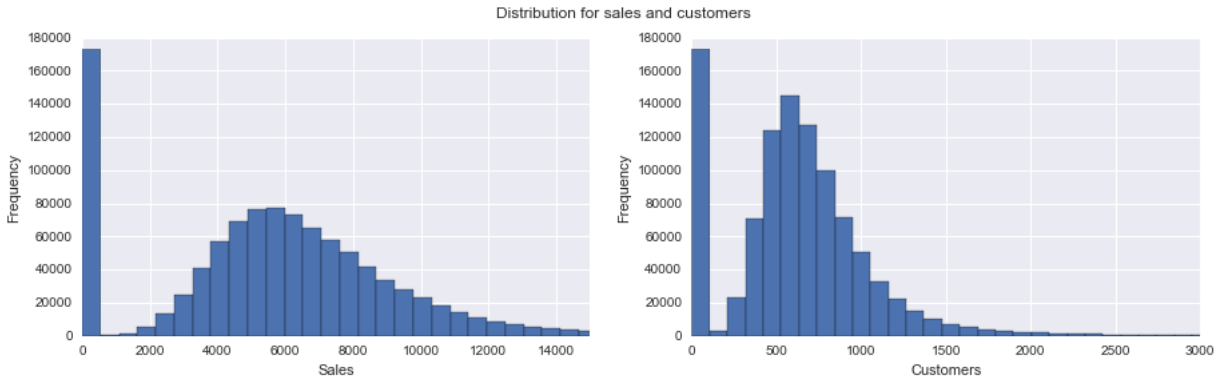
Outlier detection

For the outlier detection I have applied Tukey's method (Tukey, 1977) resulting in 10,864 outliers on the total of 1,017,155 data points. After testing on both visualizing the effect of excluding these data points, some practical tests on the model with and without the outliers followed but at the end, the results suggested that leaving the sales related outliers are they were could help the overall predictive power of the model.

Exploratory Visualization

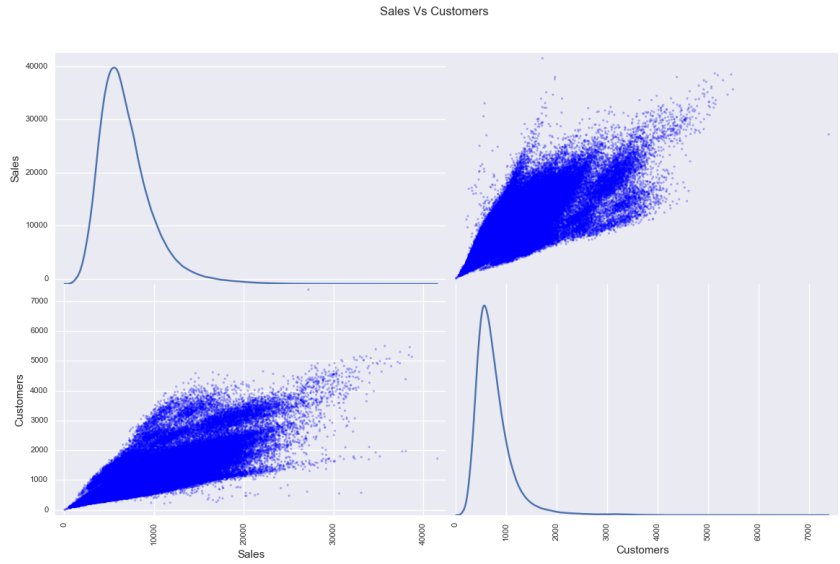
As sales is the was the Y value to be predicted and the most important feature in the data set, exploratory analysis will start from Sales and customers, a feature expected to be the most correlated with the sales.

Figure 1: Sales and customers distributions



We can see that the effect of many stores being closed on specific dates leads to 172817 cases with no sales or customers in the respective stores (analysis will follow). For this reason and since we interested in predicting sales for the days the stores are open, I will analyze the sales data without those days.

Figure 2: Sales Vs Customers



As can be seen the curves are similar with a high correlation of 0.82355 and the distribution are heavily skewed on the right and close to being a lognormal distribution or gamma distribution. On the scatterplot there seem to be some point that seem like outliers but would have to come back on this again later.

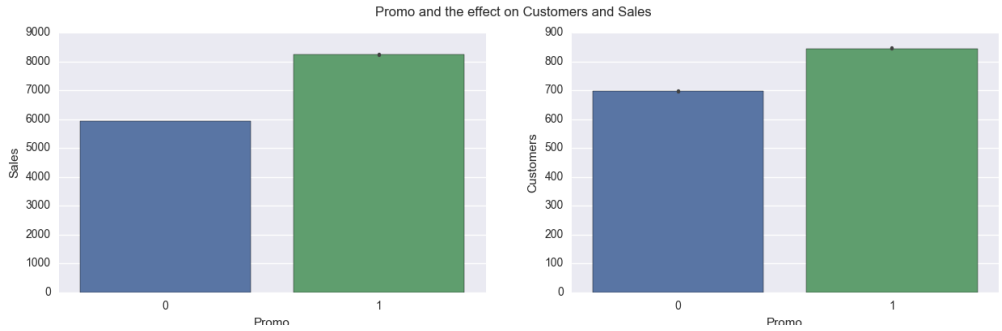
	Sales	Customers
count	844338	844338
mean	6955.959134	762.777166
std	3103.815515	401.194153
min	46	8
25%	4859	519
50%	6369	676
75%	8360	893
max	41551	7388

As for the statistics around sales and customers, after removing the 0 values we have minimum of 46 and 8 respectively and we can see that the mean for sales is 6955 and 762 for customers. The SD is also large for both features (3103 and 401 respectively) suggesting large deviation among stores and/or days especially on the right side of the mean and the that we have to be careful when suggesting the existence of outliers. In order to minimize this effect on the prediction, I have takes the logarithmic values for both features and the visual is as below.

As the next step we will analyze the closed stores data.

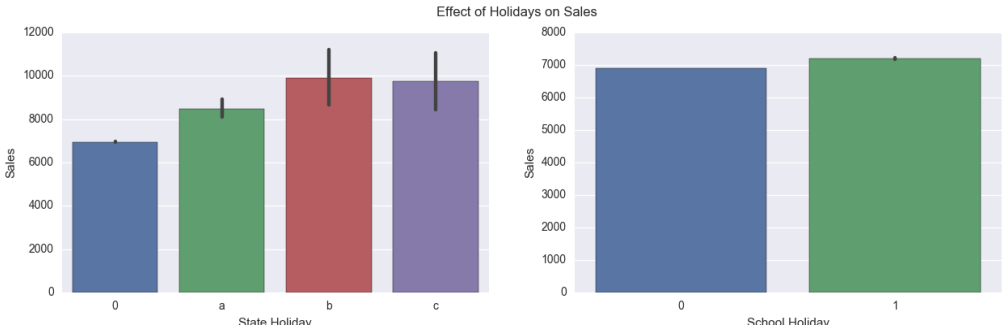
As can be seen the day that stores are consistency closed is day '7' (Sunday) so when dealing with missing values, will not take day '7' into count. There are 3593 cases in which stores where open while in Sunday, specifically 33 unique cases from the 1115 stores in total and from which 134 times it was the same store (682). This means that most stores opened on Sundays very few times if any. On the other hand trying to see the days the stores where closed on weekdays can see that all of the stores at some point were closed even on a weekday but it was mainly just due to state holidays.

We will then see the effects of promotions, and holidays on sales and customers.



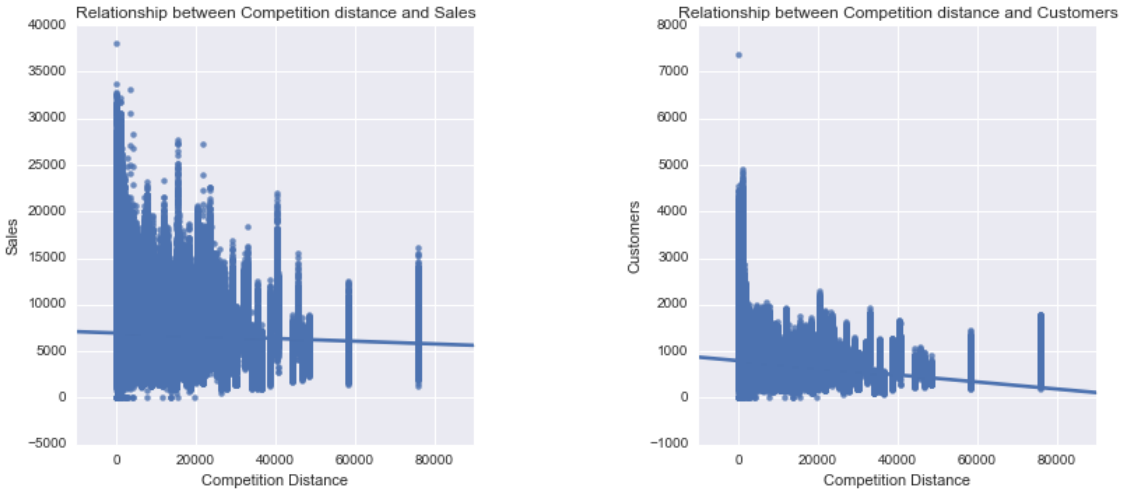
We can see that sales while in Promo was increased by more than 2000, we can also see that the average sales per customer is also increased from 8.50

for the non promo days to 9.74 for the promo days, an insight that indicates strong effect of the promo feature for sales.

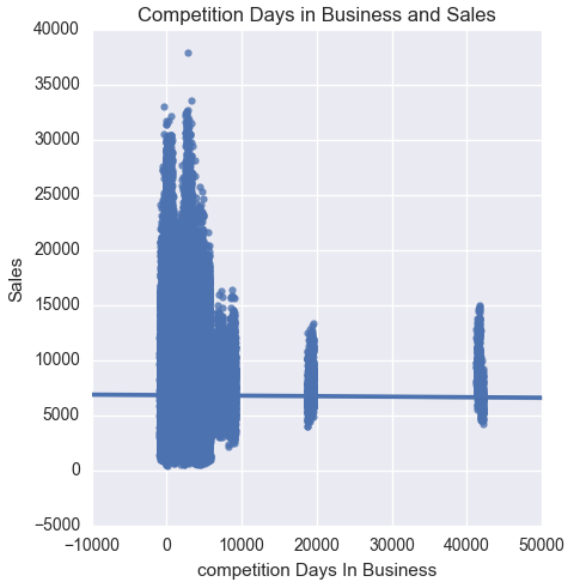


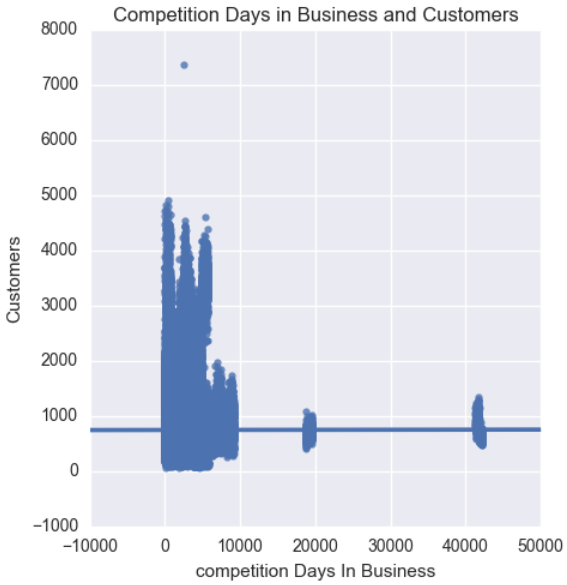
As for the holidays, school holidays have a not so strong effect on overall sales while for state holidays the effect seem to be stronger especially for the Easter holiday and Christmas and less for public holidays but still close to 2000.

As there is also data about competitors, it would be interested to see how distance and days in business for a competitor affect sales.



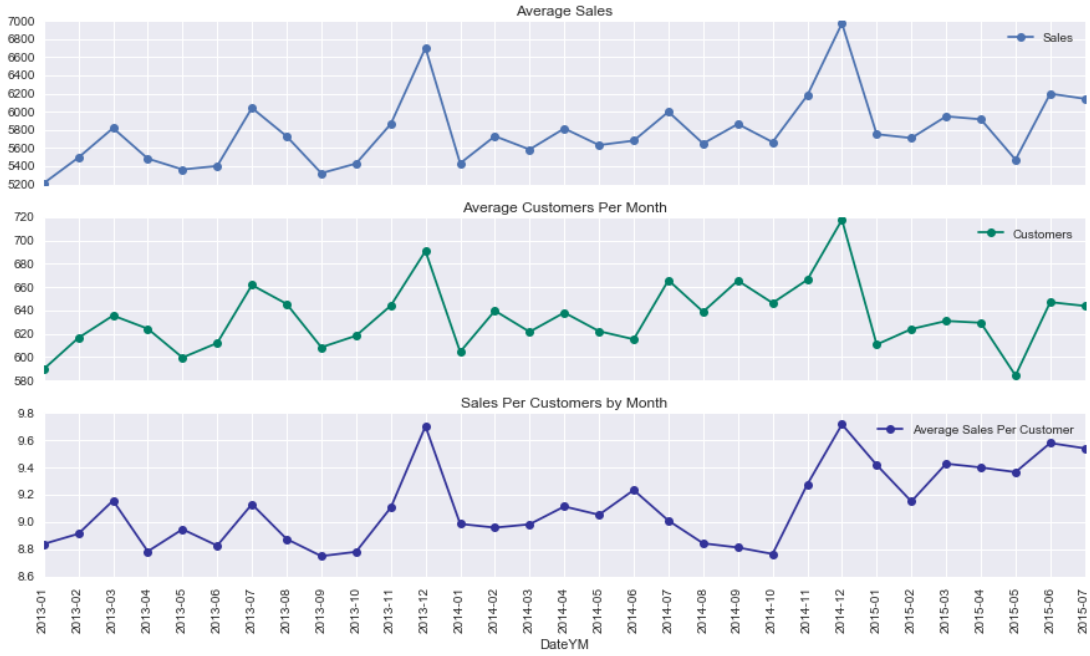
As can be seen, the seems to be a slight downward trend (some negative correlation) but I dout there is causation behind this trend, a guess is that it basically is due to that having lot of competitors beeing close, means beeing in a central area that usually comes with larger sales while not having many competitors close means the area is not as competitive, not so many customers and thus decreased overall sales.





For the days in business for the competition again there is very little if at all correlation. All of the competition related features seem not to provide valuable information at least on a summary level (as displayed above) but will try how the model performs with and without them.

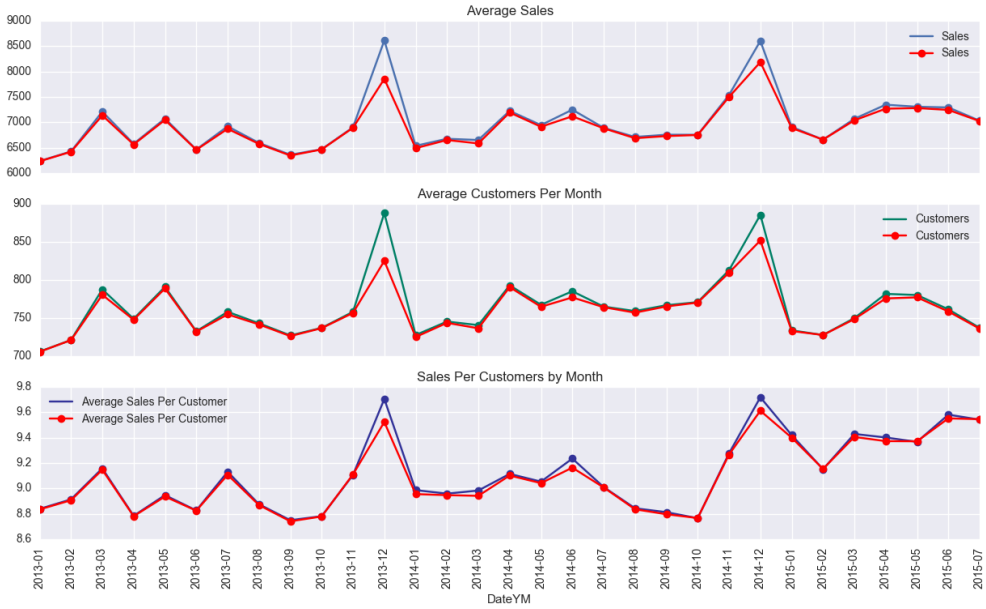
Lastly, we will see how sales and the number of customers are effected by seasonality.



We can see some huge picks on January for average sales, customers and sales per customer ratio per month. For 2013 and 2014 June was the second better month while after 2015, the stores seem to perform better on June seems to be stronger. Other than these strong trends, other months performance seem to be lower but the comparative performance seem to

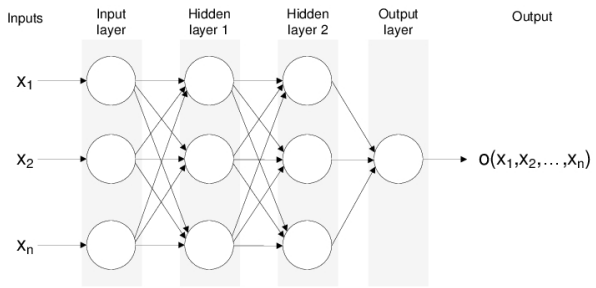
change every year, for instance May September was the worst month for 2013 but the second better for 2014.

Seeing the results after applying the outlier detection discussed earlier, we can see that for each of the features, the trend does not change, but months with high performance like January seem to be compressed (on red is the results with out the outliers). In order to make sure how the model behaves in the presence/absence of the data points considered as outliers through the Tukey's method, I tried the results with both cases and in fact, the results were better without removing any outliers from the data set (other than some points where there was no sales data regardless of the stores being open).

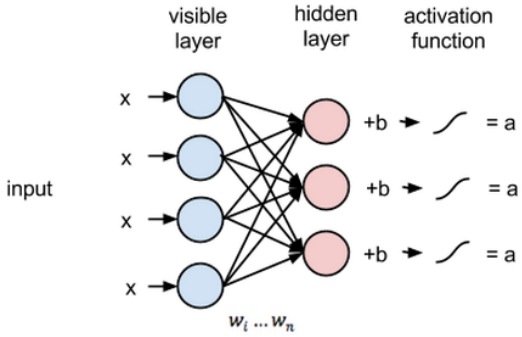


Algorithms and Techniques

As explained above, FFNN have been chosen to solve the problem due to the sparsity of the data available and the ability of FFNN to deal with such problems. The way an FFNN works is that it consists of nodes in different layers. The input layer, intermediate usually called hidden layer(s) and the output layer. Each of these layers further consists of nodes as cab be seen the representation below.

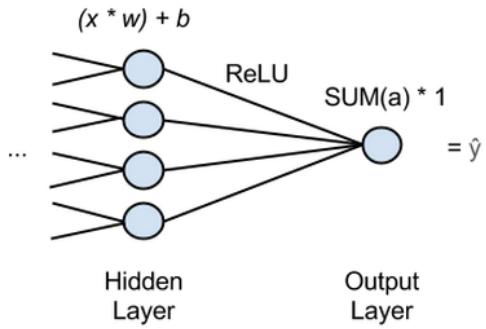


The connections between nodes of adjacent layers have "weights" ($W_i…W_n$) and “biases” (b) associated with them.



The goal of the FFNN (and ANN in general) is to “learn” meaning assign the correct weights to these edges. So given our input data ($X_i…X_n$) these weights and biases with the help of the activation function will determine what the output (and input to the next layer) is.

The way this learning often (and in this project as well) happens is through Backward Prorogation of Errors. Backward Prorogation is a supervised training scheme, which means, it learns from labeled training data (there is a supervisor, to guide its learning). In simple terms, it is like "learning from mistakes". The supervisor corrects the FFNN whenever it makes mistakes correcting the weights and biases in each connection until the loss (level of these mistakes) are minimized.



In our case the objective is to use a variety of inputs to predict a continuous output (a regression problem) as can be seen in the image above.

For each hidden node, the activation function (ReLU in the case above) outputs an activation, a, and the activations are summed going into the output node, which simply passes the activations’ sum through. Since as mentioned in our case we have a regression problem, we will have one output node, and that node will just multiply the sum of the previous layer’s activations by 1. The result will be ŷ, “y hat”, the network’s estimate for sales in our case.

As can be understand on its deep format (more than hidden layers) the network can learn through correction of a number of weights and biases and thus adjust in the needs of complicated and sparse data set like the one in this project, and that is why I have chosen to use this approach for tackling this problem.

The following are some of the parameters can be tuned to optimize the classifier:

Training parameters
    - Activation function: Activation function to use (relu, tanh, sigmoid etc.)
    - Starting values for biases
    - Starting values for weights
    - Number of epochs
    - Learning rate
    - The optimizer (AdagradOptimizer, gradient decent etc)
- Neural network architecture
    - Number of layers
    - Number of nodes in each layer
- Preprocessing parameters (see the Data Preprocessing section)

On the technology side I have used Tensorflow.

Benchmark

For the benchmark of the model created, the results from the competition Leaderboard (measured in RMSPE) will be used and a good score will be defined as one being among the top 15% of the scores ranked. In practice this means an RMSPE of 0.125 to 0.100.

The reason that I perceive a score within the top 15% of the scores could be considered an adequately good score is because instead of using the whole test set provided, as mentioned above a percentage of the training set will be used meaning that there are chances that there is not going to be enough data to help creating a model equally good as by using the whole test set data.

III. Methodology (approx. 3-5 pages)

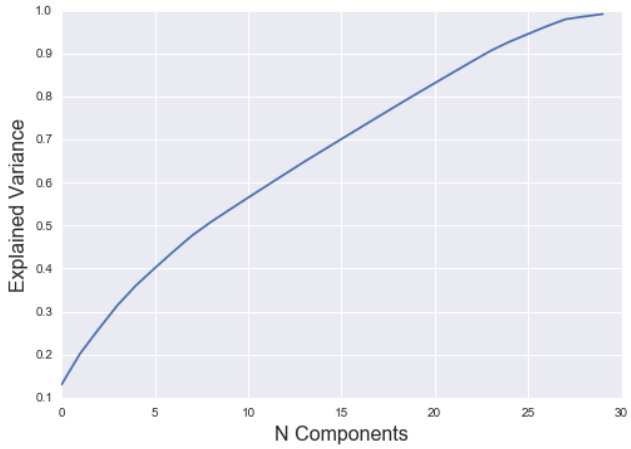
Data Preprocessing

The preprocessing is done in the "preprocessing.py" file and consists of the following steps:

- Merging of the of "train.csv" with historical data including sales and "store.csv" with supplemental information about the stores

- Formatting: In terms of formatting, there were some feature engineering involved in regards to dates so initially they feature was transformed to 'datetime' while fields like Store, Day of Week, Open and similarly form the store.csv the mutual fields with train.csv and some additional like the store type, assortment and Promo2 were transformed to categorical variables.
- Cleaning:
    - There were a few data point where the store was open but still there was no sales data which seems rather unlikely so these points where removed
    - For the outliers I have implemented the Tukey's method (Tukey, 1977) but after comparing the results with and without the supposing outliers, the results were better with the dataset untouched so decided to keep them in.
- Feature transformation:
    - One hot encoding for categorical features (StoreType, Assortment, promotion, Month etc)
    - Standardization: Through StandardScaler(), standardization has been operated prior to feeding the data into the NN.
- New feature engineering:
    - Significant effort has been given into new features engineering, specifically the following fields were created.
        - DayOfWeek, Month, Year created from the Date field
        - Combining the store to the training data
            - From the competition related data, CompetitionOpenSince field was created and with its comparison to the actual data of each data point, the Days in business of the competitor was created
            - Similarly, the days a promo was active was calculated by comparing the data of start to the actual date of each data point.
            - Also since Sales and customer data was not available in the test data (a notion that I have also followed), have created two fields for the average sales and average customers per day for each store in order to add an indication on the potentials for each store to the dataset.
    - At the end 39 features were created so considered using dimensionality reduction technique like PCA but after some tests, the results were superior without it plus in order to keep 99% of the variance explained by the model, 30 features were required so the benefit in terms of the increased computing speed were not huge.

The Variance explained Vs the n_components



Implementation

As explained previously, originally LSTMs were suggested instead of FFNN but for the reasons also explained in the first paragraphs, FFNN were chosen. In terms to the specific implementation of the algorithm, a fair amount of tests were operated in terms of the architecture of the NN, the number of hidden layers and the respective number of nodes.

In terms of the way input were connected to the NN, since there were 1115 stores whose data sales had to be predicted, I have decided to train the NN individually in each of the stores data and the way did it was be creating two large dictionaries with the store id and the respective data and iterated through it applying the NN in each of the stores combining the output on a single file and calculating the rmspe as below:

```
def rmspe(y_hat, y):
    rmspe = np.sqrt(np.mean((((y-y_hat))/y)**2))
    return rmspe
```

Refinement

As mentioned in the Benchmark section, the objective was to achieve a RMSPE of 0.100 to 0.125. The model created with its initial parameters achieved initially a RMSPE of 0.242 but with hyper parameter tuning a RMSPE of 0.1297 was achieved.

The reason that I consider the higher than the originally projected 0.100 to 0.125 score as adequate is that most competitors were using 3rd part data like from Google, Facebook etc that was widely available in the forums of the competition. Since this data is in reality known data about the term of the test set (to some degree like cheating), I have decided not to follow the same approach as in this project my objective is to create a model that can generalize and be able to predict the future sales with no previous knowledge of the to be predicted term. Considering this fact along with the previously

shorter training term, I consider the 0.1297 a decent score that can be considered as it solution to the problem at hand.

Below is the graphical representation of the improvement achieved.

The approach was to

- start with identifying the best performing architecture hidden layers and their nodes.
- Optimize around the activation functions.
- Optimize on the starting values for inputs and weights
- Optimize on activation functions
- Optimize on activation functions
- Optimize on the learning rate
- Finalize the model
-

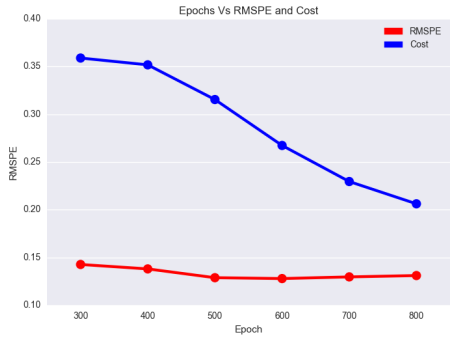
Worth mentioning that since the results were heavily varied based on the epochs, all of the above combinations were tested on a range of 100 to 800 epochs (100,200,300,400,500,600,700,800) at the same time to make sure that the results were not limited by the epoch factor. At the beginning more extreme values were used but tests showed that the best performance comes from within these values.

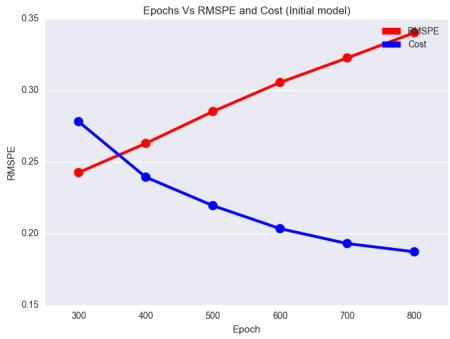
Below are the plots of the epochs and the corresponding model cost (training set) and calculated RMSPE (testing set) outputted while testing the NN architecture. In the initial tests following the suggestions of (Boger and Guterman, 1997) suggesting that hidden nodes should be as dimensions [principal components] needed to capture 70-90% of the variance of the input data which in our case meant 24 to 30 nodes. But as the results were not as good I tried increasing the hidden nodes as suggested by (Berry and Linoff, 1997, p. 323) on a level not larger than twice the number of inputs. The models below have the following parameters:

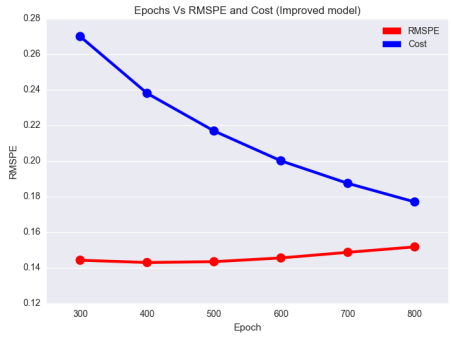
Initial model:
Architecture: 3 hidden layers (8,8,8), initial weights = initial biases = 0.01, activation functions = (sigmoid, tanh, sigmoid), optimizer = GradientDescentOptimizer, learning rate = 0.5
RMSPE = 0.242 at 300 epochs

Improved model model:
Architecture: 3 hidden layers (12, 8,8), initial weights = initial biases = 0.1, activation functions = (sigmoid, sigmoid, sigmoid), optimizer = AdagradOptimizer, learning rate = 0.5
RMSPE = 0.14297 at 400 epochs

The best performing model will be discussed in details in the next section.







IV. Results

Model Evaluation and Validation

The final architecture and hyper parameters chosen are described as below:

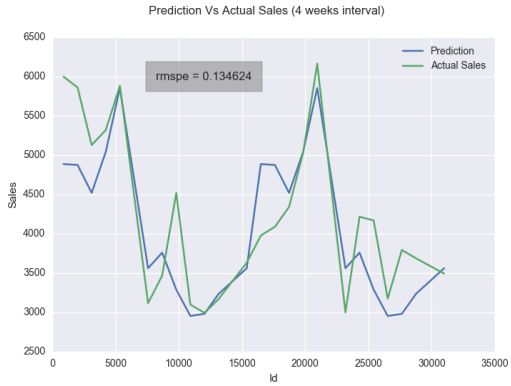
Training parameters

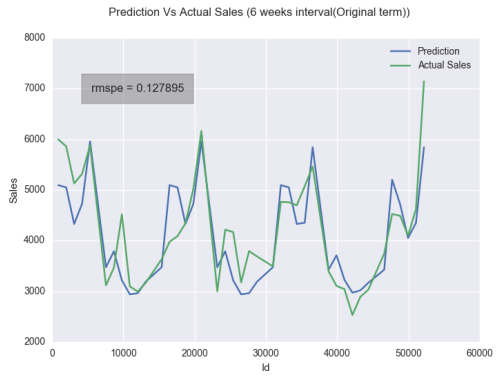
- Activation function: sigmoid in all layers
- Starting values for biases: 0.1
- Starting values for weights: 0.1
- Number of epochs: 600
- Learning rate: 0.7
- Optimizer (AdagradOptimizer)

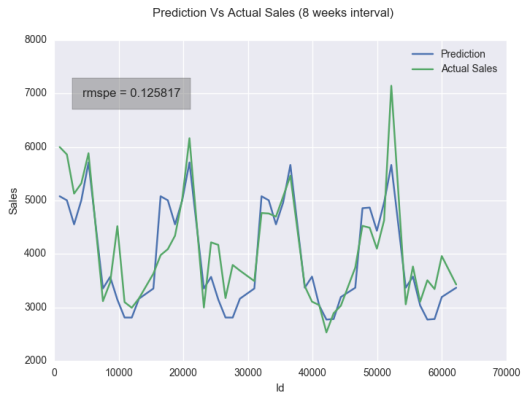
- Neural network architecture
  - Number of layers: 3 hidden layers
  - Number of nodes in each layer: the final structure was 35 inputs > H1 (20 nodes) > H2 (22 nodes) > H3 (8 nodes) > 1 Output
- Preprocessing parameters (see the Data Preprocessing section)

The parameters as described were chosen systematically and are those that maximized the overall performance of the model.

Trying to validate the performance of the model I have tried the performance for different intervals on 1 random store (store 879) in the case different time intervals were used so other than the original 6 weeks, have tried for 2, 4 and 8 weeks.







As expected the best performance comes form the shortest term with surprisingly the worst performance coming from the 4 weeks interval but the main point here is that the performance is comparable regardless of the term and around the overall models 0.129 so we can conclude that it provides a robust solution to the problem

Justification

As described above, the final RMSPE for the model was 0.129 without using any external data. This is considered a good result especially considering the fact that the data used was limited compared to the one on the original competition. In the present project the target was to predict 6 weeks period (2015/06/14 to 2015/07/31) of sales using past data provided by the store from the period 2013/01/01 to 2015/06/13 while the in the original competition, the data available for training was from 2013/01/01 to 2015/07/31and the testing sample 6 weeks after that. This fact by itself can imply some limitation on the accuracy the model will be able to achieve so the RMSPE achieved can be considered a good result.

Furthermore considering that the model was tested in both shorter and longer prediction intervals and still could have similarly good results we can conclude that the model can generalizes adequately well and can be considered a solution to the problem.

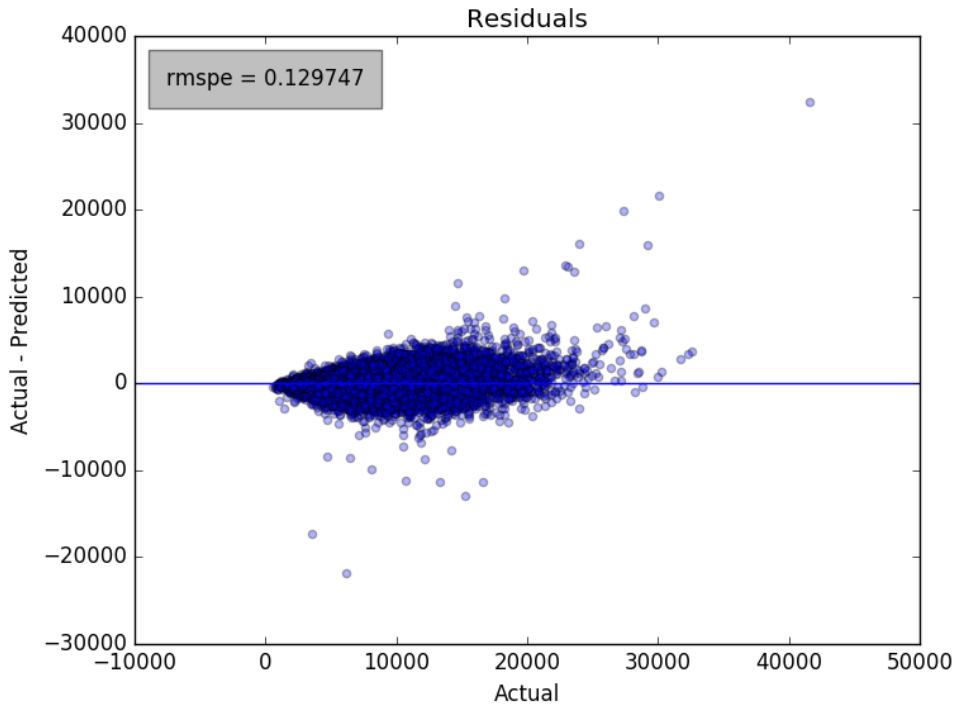
V. Conclusion

Free-Form Visualization

In order to understand better the performance of the model we can take a closer look to the residuals. Below is the visualization of the difference actual – predicted Vs the actual values. Generally the model seems prone to underestimating the actual sales since we can see that the majority of the data points lay on the positive side especially as the value of the actial values increases.

Residuals are as expected lower for lower values but interestingly they are not proportionally increasing with the size of the actual values (sales). They are increasing until the area around 12,000 and start decreasing again even until the relatively extreme values of more than 25,000 with limited exceptions.

Furthermore considering the total number of the 45,884 data points we can see that there very few (around 10 ~ 20) cases that the model fails completely to predict the actual values which is a very good indication for its robust.



Reflection

The process followed in this project can be summarized as follows:

1. Getting the data from Kaggle
2. Getting the 3rd party data used to supplement the first
3. Setting the benchmarks for an acceptable performance

4. Preprocessing the data
5. Creating the structure of the NN in TF
6. Tuning the hyper parameters of the model so as to achieve a performance close to the benchmark
7. Test the models robustness

The most challenging parts were step 4 and 6.
About step 4, the challenge was that the data was coming from 1115 different stores of different types and location characteristics and with different operating structures (some were closed on Sundays some were not for instance). This made the whole preprocessing step challenging and for this reason I have decided to train the weights and biases of the NN for each store individually.

The challenge on Step 6 was that I run the whole training on a local machine and it took 5-6 hours for a full run. This made the model optimization and NN tuning extremely difficult.

On the positive side I believe that the fact that the model performed similarly well on different time intervals regardless of having been tuned on all of the stores at the same time, shows good signs about its robustness and I believe it can be consider an adequate solution to the problem.

Improvement

As mentioned in the previous section the biggest challenge was the time needed to train the NN. On my tests, smaller networks did not performed equally good and that was the reason I have used 3 hidden layers but considering the limitation this fact have created on the ability to try as many combination as possible I am confident that in the case of using cloud computing the model could have been improved even more.

Similarly and again in connection to the training time challenge, I was not able to use a grid search technique in order to automate the optimization process, this again would have been a huge addition on maximizing accuracy.

Finally, my initial though of using LSTMs instead of RNN at least theoryally seems to be a promising approach but due to my luck of understanding and real life examples I was not able to try it and compare the results.

References:

1. Armando Bernal, Sam Fok, Rohit Pidaparthi (2012), Financial Market Time Series Prediction with Recurrent Neural Networks.
2. Rossmann Store Sales (2016). Retrieved from https://www.kaggle.com/c/rossmann-store-sales

3. Karpathy Andrej, Johnson Justin, Fei-Fei Li. (2016) VISUALIZING AND UNDERSTANDING RECURRENT NETWORKS, Department of Computer Science, Stanford University
4. S. Hochreiter. (1991) Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut f. Informatik, Technische Univ. Munich, . Advisor: J. Schmidhuber.
5. Biddy J, Toutenburg H. (1977). Prediction and Improved Estimation in Linear Models.
6. Tukey, JW. (1977). Exploratory data analysis. Addison-Wesely
7. Boger, Z. & Guterman, H. (1997). Knowledge extraction from artificial neural networks models. Proceedings of the IEEE International Conference on Systems Man and Cybernetics, SMC'97, 3030-3035, Orlando, Florida.