Combining different forecasts using ANN in the M5 competition

Abstract

In this paper, I describe my method with which I participated in the M5 competition for time series forecasting which ranked in a 4% top score. I used a combination of 4 forecasting methods. The optimal combination was produced by a Deep Artificial Neural Network with 4 hidden layers. The simplification of the loss function for evaluating the model and the dismissal of using multipliers were the two key factors of this method's success. This method created a simplistic way to capture the dataset's future shape giving an easy benchmark for post processing the results for different aggregation levels.

1. Introduction

The task in the M5 time series forecasting competition was to provide forecasts for 30,490 numeric time series regarding product sales of Walmart products. In addition, each product sales would group in another 12,350 aggregation levels regarding total sales, sales for each State, sales for each store etc. So, the final evaluation is being done in 42,840 using the Root Mean Squared Scaled Error (RMSSE). Also, explanatory variables such as product prices and calendar events were provided.

The competition was structured in two phases, 1. Validation phase and 2. Evaluation phase. In the validation phase, 56 time steps where hidden and the validation was being calculated for the 1 … 28 steps ahead using the Kaggle Submission. In the Evaluation phase, the ground truth for the 1…28 time steps was made public and the final submission was calculated using the RMSSE in the 29…56 time steps. Each team participating in the competition was free of choosing any method they wanted. Also, a lot of participants created public kernels for other data scientists to follow along. In my approach, I used 4 of these public kernels to create a new combination as the final forecast.

I participated in the competition as a solo undergraduate Statistics student so I had to create a flexible, bottom-up, time consuming method that would minimize the RMSSE without too much post processing although that is highly recommended.

An approach a lot of participants followed was manually multiplying their forecasts with a constant in order to minimize the RMSSE in the validation phase. Practitioners of such methods are proven to often overfit their models and fail in the hidden sets, thus where dismissed.

2. Method

The forecasts of each time series was obtained using the combination of 4 different methods proposed from other researchers. These 4 different methods were:

- LSTM model (with a 14 day timestep)
- Prophet model (Top down method)
- Deep Neural Network (with categorical embeddings)

- LGBM model (Department by Department)

All of these models achieved great scores in the validation period. A few of them made use of constant multipliers from their respective creators in order to minimize validation score, but the multiplier was removed and the kernel was re-run in my approach.

The reason for picking these 4 approaches was not only that they scored well. The key idea is that they are four completely different approaches so there is a sense of mapping patterns from their predictions. I did not use, for example 10 different LGBM models (which were the top used models in the M5 competition) with different hyperparameters as I believe it would create noise in the training set. Let's assume the predictions of 10, top performing in the public leaderboard, bottom up LGBM models were used as input variables. Since the approach and the model is identical except of the hyperparameters, the small difference in the validation score is due to a vector of small random errors ε. Thus:

Predictions 1: OPT + ε_1

Predictions 2: OPT + ε_2

…

Predictions 10: OPT + ε_{10}

Where OPT is the matrix with the optimal predictions that could be produced in an identical LGBM bottom up structure with a specific set of Hyperparameters. There are two extreme scenarios using these predictions as input for an ANN. A) The weights of the ANN, until a certain layer, manage to discriminate the OPT matrix from the random errors ε_i and the ANN feeds the next layers with one input variable which is this OPT matrix, so the final forecast is a neural network using one input variable, the OPT matrix. B) The random error terms create noise in the training phase and the final weights of the ANN are useless or/and there are convergence problems.

Obviously, the worse case scenario B was needed to be avoided. The best case scenario A is still a weak ANN since it is basically a one variable input model that can not have much power. The OPT matrix would be a matrix very close to any of the, already fine tuned, Predictions 1, …, 10 so feeding an ANN with one of any of the 10 Predictions instead of matrix OPT, would not make much difference.

2.2 Creating the Artificial Neural Network

In order to train the ANN, the predictions of the 4 other methods were stacked in a data frame. For the 28th day, the predictions where held out as a test set (where the mean squared error was calculated to). So, the full training dataset is a 823,230 x 5 data frame. During the training, a 10% validation was specified, so the train occurred in 740,907 samples and the validation in 82,323 samples. Such big data frames are ideal for Deep Learning models. Basically, the Artificial Neural Network is feeded with 27x30,490 = 823,230 samples, which are a fraction of the original 1913*30,940 samples but, still 823,230 samples is a huge training set.

Different approaches were tested, such as feature engineering in the input variables using first differences. Another approach was adding a fifth input variable which was the 28 next-days after the past most correlated 15 days of the last 15 days of each product. The algorithm for creating the fifth input variable goes as follows:

For each series:

A) Calculate the MSE(1899 to 1913 , all consecutive 15 days in 1 to 1872)
B) Find the minimum(MSE_1 , MSE_2, … , MSE_{1857})
C) Let MSE_n be the minimum, store the next 28 days (n+15 to n+15+28)

Both approached did not improve the mean squared error in the test set.

The hyperparameter optimization as well as the scalers and the activation functions, where tuned manually by looking at the test set performance. The final form of the model is a Deep Neural Network with hidden layers such as: 4(input) x 9(relu) x 112(relu) x 43(relu) x 48(relu) x 1(relu / output).

The MinMax scaler was used to transform the data and the optimizer was of ADAM type with a learning rate = 0.00001 (bigger learning rates often resulted to convergence problems).

The model's minimum MSE was 5.41 on the test set. The rate with which the test MSE was being minimized differed significantly in each re-configuration with identical parameters. Because the MSE had significant changes from epoch to epoch, I decided in a slightly different approach from the standard methodology in deep learning modeling. I created an algorithm that runs epochs until the model gets a test score close to the observed minimum. If that does not occur in a lot of steps (50 epochs), the model is re-configured and the loop re-runs until the test loss is close to the observed. The key here is that, when this score is achieved, the model is not trained for another epoch as it could result in an increasing spike.

2.3 Forecasts using the Artificial Neural Network

The forecasting with the ANN model is very simple. The only time consuming part is reproducing the predictions of the four models for the hidden 29,…,56 days ahead, because the models have to be re-trained by adding the 1, …. ,28 ground truth sales in the training data. After getting these predictions, the ANN model makes the final forecasts which were used raw for the final submission. The advantage of using this type of training sets instead of the original, ground truth 1,900x30,490 sales is that, we can produce the input variables for the hidden 29,…,56 days easily.

It is clear this is a bottom up approach which means that, no aggregation levels were taken into consideration. The aggregation levels were taken into consideration only in some of the input predictions. The idea of this method is combining different sophisticated methodologies into an optimal "black-box" mechanism.

3. Results

The results which were used in the final submission are the raw forecasts for the sales of each product. With no post processing at all, we can check the visualizations for different aggregation levels. Surprisingly, the graphs are very sophisticated, considering that in it's core, this is a bottom up without categorical embeddings or top down methods. We can check the total sales for the validation set and the ANN predictions in the graphs bellow just to get an idea of how the model captured the general shape.

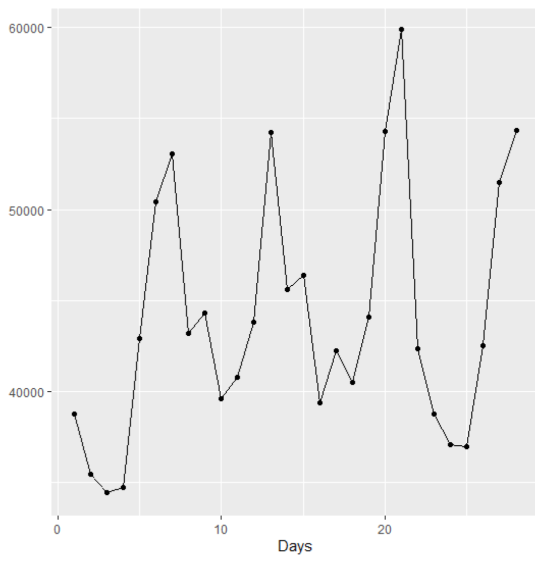

Figure 1 Real values of Validation Set (Days 1 to 28)

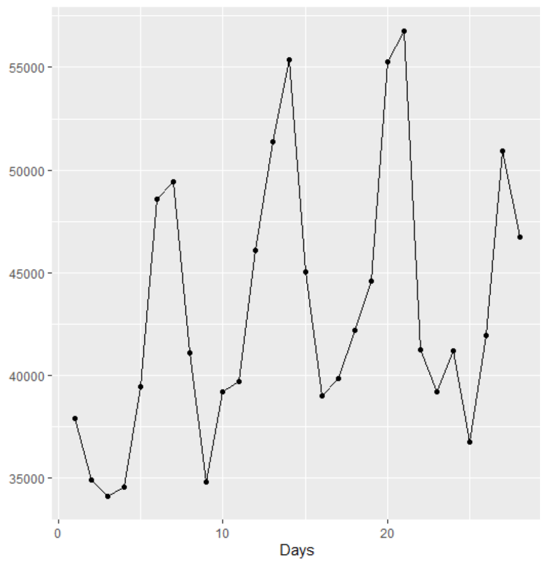

Figure 2 Forecasts of Evaluation Period (Days 29 to 58)

The validation (Public) WRMSSE was never calculated using the ANN since the only predictions made using this model were for the evaluation phase. As stated previously, the test measure for the ANN was the MSE of the 28th day. The Table below shows the Public, Private and Test scores for the 4 methods of the other researchers used as input and the ANN. The results are also compared with the top performing statistical benchmark provided by the organizers (Bottom-Up Exponential Smoothing). If this method was used, it would rank at 416th place (top 8%). To make a comparison, in the M4 competition, only one participant beat the benchmark.

Model	Public	Test (MSE)	Private
LSTM	0.69720	6.2638	0.79636
DNN	0.62082	6.2344	0.67628
Prophet	0.63419	5.9213	0.71013
LGBM	0.62602	5.7673	0.68907
ANN (Combining the Above)	NA	5.41	0.62643
Exponential Smoothing	0.75698	NA	0.67098

It is worth mentioning that the “fifth input variable” discussed in section 2.2 managed to get a Private score of 1.028, outperforming the Naive and Random Forest benchmarks.

The relationship between the Test loss and the Private loss is not linear. For example, DNN's Test loss is poor compared to LGBM's but the Private score for DNN is better.

Public Score appears to be a better approximation of Private Score since the ranking of the models is the same for these two losses (Except for the ES model).

The highlighted green row is the ANN's performance. We can see that it out-performed all other models. Specifically, it outperformed the best scoring model (DNN) by 7.9 %.

I could go on and calculate each of the 4 models 28 days prior of the validation set and use the ANN to calculate its public score. As the table suggest, it would have been a better plan compared to calculating only the Test loss.

4. Discussion

As can be seen from Table 1, the ANN model can create a great general case for combining different predictions. One thing to note is that, when dealing with big data sets it is rather difficult to create a robust combination of the predictions without the use of machine learning. Using simple combinations such as the arithmetic mean is common practice in smaller datasets and the early M-competition made such cases but when dealing with bigger datasets it appears that the use of complex combinational models is necessary. The variety of different underlying distributions, the variety of trends and heteroskedastic time spans, the information derived by categorical variables etc, makes the predictions impossible to combine with simplistic rules.   An indication can be seen in the private leaderboard, where the combination of the statistical benchmarks using a simple arithmetic mean, did not outperform the Exponential Smoothing and Exponential Smoothing with Explanatory variables (ESX) models.

Further improvements of the model could be:

- Using an LSTM model instead of the non-recurrent ANN to make the combinations
- Using more prediction models as input variables
- Using the categorical variables in the final model to better capture aggregation levels and trends
- Using WRMSSE as the evaluation metric instead of plain MSE
- Random sampling the model for higher quality Cross Validations

All the necessary code will be available here, I encourage other researchers to implement their own ideas and further improvements.

5. Conclusion

In this paper, a description of the methodology to obtain point forecasts for the M5 competition was provided. I used ANNs in a big data forecasting competition where I combined four popular prediction methods in the validation phase to output better forecasts in the evaluation phase. The dataset's main difficulty is the fact that it contains a lot of zero values which are very unpredictable and can greatly affect the performance of the final submission. This method is transferable to any other point forecast or even regression problem. Given the results, the improvement is significant in this particular dataset were the distribution of each product is a discrete, positive value. It is important to note that this method can provide state of the art results only when the input variables make high quality predictions in the validation set.