Automated Hyperparameter Tuning based on Bayesian Optimization for Time-Series Prediction using NAR and LSTM Networks

ArunKumar Nachimuthu Palanichamy, Shoban Venugopal Palani, Viswanathan Babu Chidambaram Ayyappan
Department of Electrical and Computer Engineering, New York University
{an3333, sv2244, vc2173}@nyu.edu

Abstract

A Non-linear Autoregressive Neural Network (NARNN) [1] is designed for time series prediction of Walmart sales data [9,10] with an objective of maximizing the test accuracy. Bayesian Optimization [5-7] is utilized in this network for automated hyperparameter tuning to determine the optimum parameters that do not overfit, and this method is analyzed in comparison with grid search and manual search in order to realize to what extent it improves the overall hyperparameter tuning process. The NAR network will be compared with the Long Short-Term Memory (LSTM) Network [3,4] in order to obtain the most accurate model.

1 Introduction

Hyperparameter tuning is often time consuming and tedious especially if the dataset used is enormous in size and at times, tuning methods like grid search or manual search does not give the best set of parameters due to the presence of computational and time limit concerns. Automated hyperparameter tuning algorithms formulated based on Bayesian Optimization [5-7] eases the exhausting process of choosing the right set of parameters suitable for the given dataset. In the Bayesian Optimization method, an initial group of parameters are defined and then the hyperparameter values are optimized in relatively few iterations by forming a model based on conditional probabilities. This model navigates the parameter search based on insights gained from the previous iterations of the search whereas grid search is purely deterministic i.e., each search is not based on any information from previous iterations.

Since Non-linear Autoregressive Neural Networks (NARNN) [1] predicts the output based on weighted combinations of the output in previous time steps. These networks are suitable for mapping the pattern in a time dependent dataset. Long Short Term Memory (LSTM) networks are also proven to perform well in predicting time series data [3,4]. Hence, if Bayesian Optimization is combined with these networks, the resulting network would be sturdy enough to predict time series with very minimal error.

In this paper, Walmart Sales data of various products across three states in the United States is used and this dataset was used in the M5 2020 Forecasting-Accuracy competition [9] and it is readily available in the Kaggle Website [10].

2 Dataset and Models

This section briefly explains about the dataset used and about the process of different models.

2.1 Dataset

The M5 dataset, made available by Walmart, is used and this dataset was utilized in Kaggle's M5 2020 time series prediction competition as well [9]. It contains information about the unit sales of different products sold across 10 stores of Walmart, located in three states in the USA: California, Texas and Wisconsin. The dataset has time series data of the products over a period of more than 5 years from 01/29/2011 to 06/19/2016 (1941 data points) given in Figure 2. The dataset is divided into three different categories namely hobbies, household and foods and the unit sales heat map of the three categories is given in Figure 1a, 1b and 1c. The dataset also contains information about other variables like special events (holidays, sporting events, promotion events etc.,) and sale price of each product. Sample data entries for a single product for 10 days is given in Table 1.

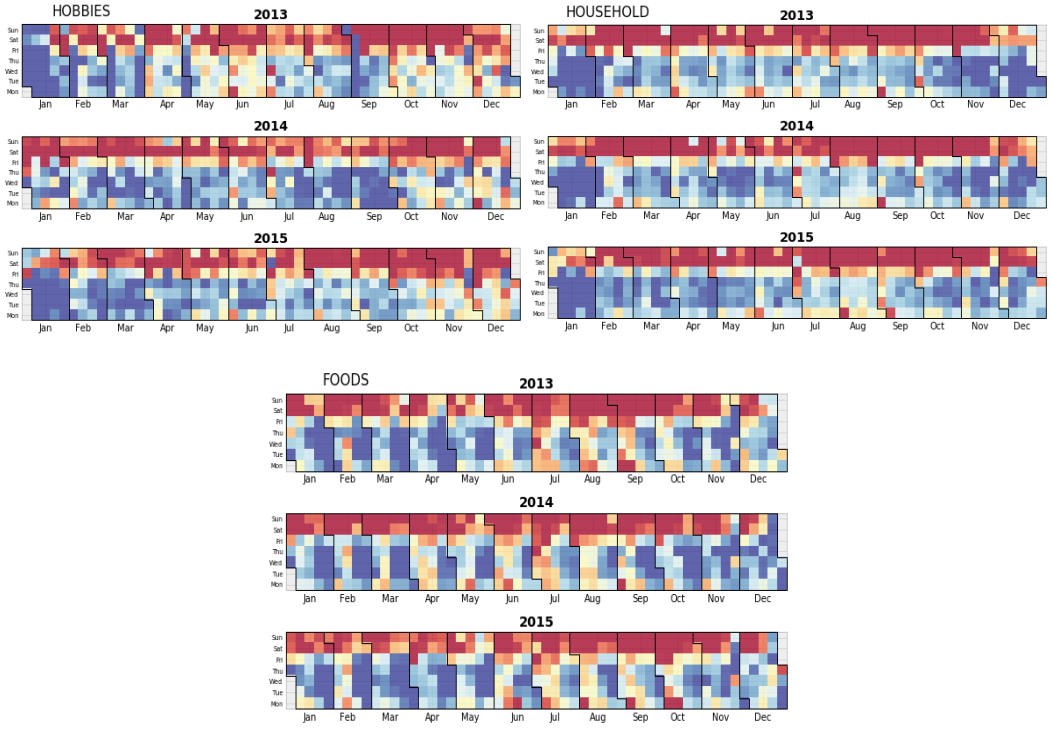

Figure 1(a), 1(b), 1(c): Unit sales heat map of three different categories

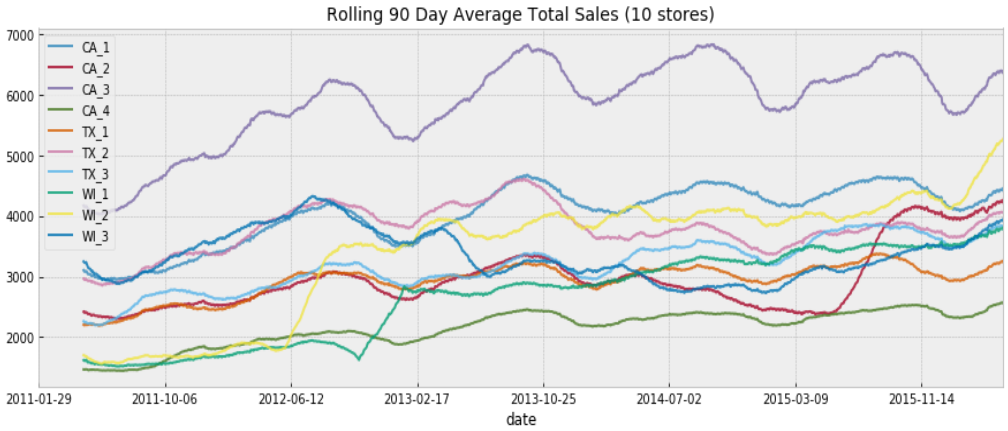

Figure 2: Unit Sales Data over 5 years

Table 1: Data Entries of a single product of Walmart

DAY	UNIT SALES
1	12
2	15
3	0
4	0
5	0
6	4
7	6
8	5
9	7
10	0

2.2 LSTM Model

Long Short-Term Memory (LSTM) Network [3,4] is a type of recurrent neural network which are designed to understand patterns in sequences. Hence, it is proven to be suitable for time series forecasting. The LSTM architecture is given in figure 3 and the governing equations of the LSTM network is given in figure 4.

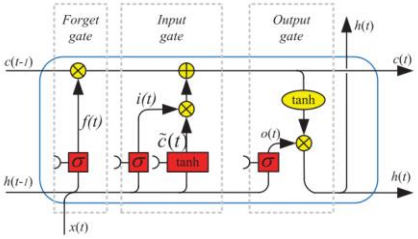

Figure 3: LSTM Architecture [4]

$$
\begin{aligned}
f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f),\\
i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i),\\
\tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}),\\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t,\\
o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o),\\
h_t &= o_t \cdot \tanh(c_t).
\end{aligned}
$$

Figure 4: LSTM Governing Equations [4]

An LSTM model consist of three gates and a cell state which gives the ability to learn selectively, unlearn or information can be retained for our need.

Cell State: It helps the information flow through the LSTM units without altering them, by allowing only some linear interactions between them.

Every unit has an input, output and a forget gate which allows us to add/remove information to the cell state. The work of forget gate is to decide which information needs to be forgotten from the precious cell state which is done using the sigmoid function. The input gate allows the information to flow to the

current state with a point-wise multiplication operation of tanh and sigmoid. At last, the output gate controls which information to pass on to the next hidden state.

Cell State Updates: The previous cell state gets (c_t-1) is multiplied with the result of the forget state. Then the new information is added to get the latest cell state (c_t). To update the hidden state, the current cell state (c_t) is passed through a activation function (tanh) and finally multiplied with the output gate's results.

2.3 NAR Model

The Non-linear Autoregressive Neural Network (NARNN) model [1] is specifically designed for predicting time series data. The NAR network operates based on the principle that the data in each time step has a weighted dependence on previous time steps. The structure of the NAR network is similar to a basic artificial neural network except that the output is dependent only on the input in a basic neural network. In a NAR network, the output is given by the following equation,

$$y(t) = f(y(t-1), y(t-2), \dots y(t-d))$$

Where y is the predicted output and d is time delay which dictates the number of previous time steps that the current time step data is dependent on. The training phase of NAR network determines the mapping function f that includes the weights of previous time steps.

3 Methodology

This section explains about the techniques utilized and the hyperparameters chosen for the initial model.

3.1 Bayesian Optimization

Most of the conventional hyperparameter search algorithms like grid search and random search are computationally expensive. Grid search just evaluates the model in loop based on the given hyperparamter range by varying them with fixed or variable steps whereas random search algorithm does vary the hyperparameters in steps and it evaluates the model by picking hyperparameters from the search space on a random basis. Both of these methods are not intelligent as nothing is learned in each search and there are chances that these methods do not find the optimum values.

Algorithm:

1. Build a gaussian probability model of the objective function
2. Choose hyperparameters that performs best on the gaussian process
3. Apply these chosen hyper-parameters to the true objective function
4. Update the gaussian model

Repeat 2-4 until maximum iteration is reached

Bayesian Optimization based hyperparameter tuning [5-7] follows a probabilistic approach where each step of search influences the further steps in the sample space of parameters such that valuable information is gained in every step which reduces the total time of the search and it is done with almost no manual effort except the initial network setup. This method is mainly based on the Bayes theorem given by,

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

In the above formula, the event A can be thought of as the accuracy metric and the event B is analogous to the set of hyperparameters. Using this approach, the probability of maximizing the accuracy with the given set of hyperparameters is calculated in each step and consecutively, this approach is used to optimize a function such as the Gaussian Process regression to find the area in the hyperparameter sample space that maximizes the accuracy with increasing precision in each step

of the search.

In hyperparameter optimization, the objective function is the validation error of a model using the set of hyperparameters. We need to find the hyperparameters which gives the least error on the validation set believing that it will generalize to the testing set.

As said, Bayesian optimization though good in tuning the hyperparameters, has its disadvantages. The algorithm will switch from trying new values to exploiting them as the search progresses. I.e., if the algorithm finds a local minima and might concentrate on the hyperparameters values around the local minimum instead of trying different values far from the domain space. If we want a more informed grid search, we need to use boosting algorithms which helps us define a smaller grid concentrating around the best hyperparameters values. If the algorithm is performing well as it progresses, it's better to stop and start a new search. At the same time, if we found the optimal hyperparameter values, the algorithm should focus on the same values in the subsequent searches as well.

3.2 Initial Parameters of the NAR Network

Bayesian Optimization requires an initial model to begin its search. In this paper, for the given dataset, the objective is to predict unit sales data for a single product for the next 28 days and a delay of 28 is chosen for the NAR network with two hidden layers of size 10 and 20 with dropout regularisation and hence, the architecture of the NAR network is (28, 10, 20, 28). The given data is transformed such that the next 28 steps (outputs) depend on the previous 28 steps (inputs). The Adam Optimizer [11] is used to train the network and a batch size of 32 is chosen. The MSE function is used as the loss function. Hyperparameter sample space for Bayesian optimizer search: learning rate range – 1e-5 to 0.1and number of epochs range – 10 to 50.

4 Results

Using the NAR network for sales prediction of one product on one time series example by creating a sequence of 28 samples (for forecasting 28 days). This is done using the sliding windows concept with a delay value of 28. The performance of the NAR network is not satisfactory with hyperparameters chosen manually since the model is unable to learn anything with a basic fully connected layer. The train loss and test loss on the dataset is very poor for the chosen small dataset. Hence, Bayesian Optimization is implemented to automate hyperparameter search to improve performance [13]. The test loss curve of the NAR network with Bayesian optimized hyperparameters is given in figure 5.

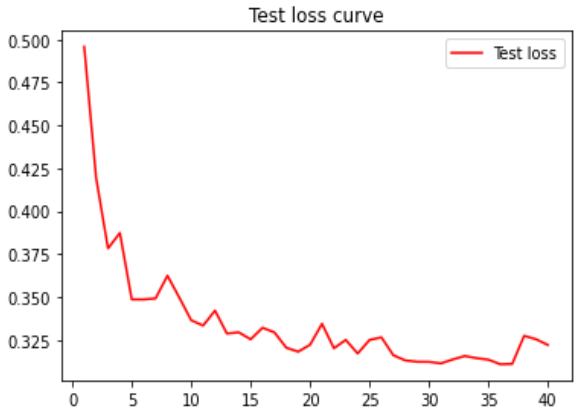

Figure 5: Test loss curve of NAR network with Bayesian Optimization

The Bayes Optimization method is compared with the Grid search optimization method where the hyperparameters are changed in each loop in a step wise manner with uniform increase or decrease. The test loss curve of the grid search method is given in Figure 6.

Table 2: Test Performance of Grid search vs Bayesian Optimization

METHOD	TEST LOSS
Grid Search	0.335
Bayesian Optimization	0.325

After search optimization, the NAR network is trained again with the best parameters: learning rate and number of epochs. The learning rate chosen by the Bayes optimizer is 0.0015 and it is 0.1 for grid search. The number of epochs chosen by the Bayes optimizer is 40 and it is 24 for grid search. Note that the best set of parameters vary for the search methods as they follow different approaches. The test loss at the end of training for Bayesian optimization method and the grid search method is given in table 2.

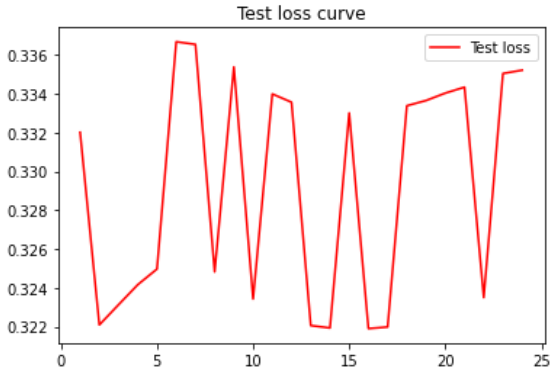

Figure 6: Test loss curve of NAR network with Grid search Optimization

Though the Bayesian optimization performs better, the performance of the NAR network is not satisfactory. So, an LSTM network is implemented whose results are given in the section 4.1.

4.1 Results of the LSTM Network

An LSTM network is implemented to predict the data with better performance. Before that, the data has to preprocessed. After multiple preprocessing techniques, the final dataset looks like this

- 1262 sets of 28 samples each as the features (X) and 1262 labels as our target(y) in the training set
- 622 sets of 28 samples with 622 labels in our tests set

With a basic model with multiple LSTM layers [11] and one dense input layer evaluated with MSE loss for 500 epochs and lr=1e-3, an RMSE value of 2.02 was obtained, which is better than the NAR network, but the model is further improved. The number of LSTM layers are increased further which gave an RMSE of 1.79. Then with the same model, more features are added other than sales units to the data set and lagged the features, for the same hyperparameters, the resulting RMSE is 1.687. The predicted and the actual time series graph is given in figure 5.

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28
0	HOBBIES_1_001_CA_1_validation	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	HOBBIES_1_002_CA_1_validation	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	HOBBIES_1_003_CA_1_validation	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	HOBBIES_1_004_CA_1_validation	1.0	2.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
4	HOBBIES_1_005_CA_1_validation	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Figure 6: Predicted Unit Sales for a few products over 28 days

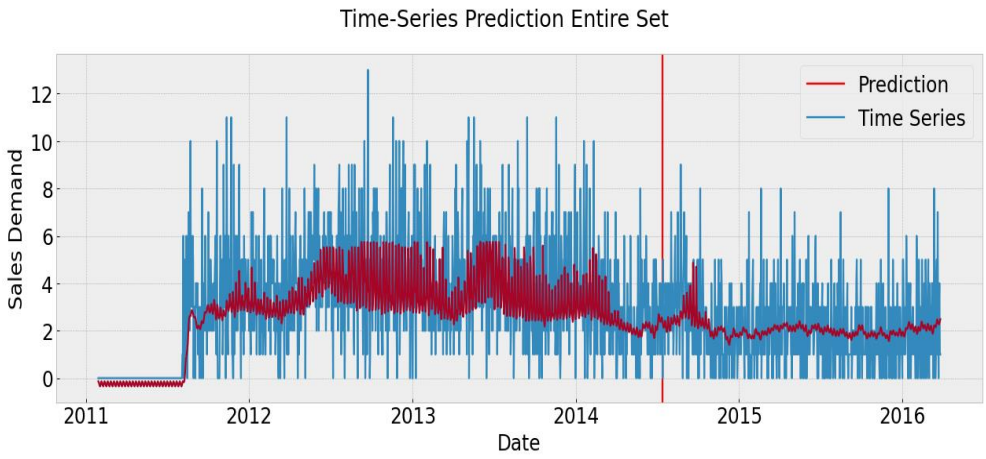

Figure 5: Actual time series and predicted data

5 Conclusion

Time series prediction was tried out with NAR network with manual hyperparameter optimization but the results were not good. Next, the NAR network parameters were optimized with Bayesian Optimization and Grid search. While these two search methods, the Bayesian Optimization turned out to be superior to the grid search method. Also, the Bayesian Optimization can save a lot of time compared to the grid search method when dealing with large datasets. To improve the performance further by using the whole dataset of all products, an LSTM model was implemented which gave better results and the performance of the model was further improved with various preprocessing techniques and added more features to the dataset which gave a train loss of 0.0984 and a test loss of 0.08139. Hence, it is concluded that the LSTM model performs better than the NAR model for the considered dataset.

References:

[1] Ruiz, Luis G.B., Manuel P. Cuéllar, Miguel D. Calvo-Flores, and María D.C.P. Jiménez. 2016. "An Application of Non-Linear Autoregressive Neural Networks to Predict Energy Consumption in Public Buildings" *Energies* 9, no. 9: 684. https://doi.org/10.3390/en9090684

[2] H. T. Siegelmann, B. G. Horne and C. L. Giles, "Computational capabilities of recurrent NARX neural networks," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, no. 2, pp. 208-215, April 1997, doi: 10.1109/3477.558801.

[3] Chimmula, Vinay Kumar Reddy, and Lei Zhang. "Time series forecasting of COVID-19 transmission in Canada using LSTM networks." *Chaos, Solitons & Fractals* 135 (2020): 109864.

[4] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). *A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Computation, 1–36.* doi:10.1162/neco_a_01199

[5] Wu, Jia, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. "Hyperparameter optimization for machine learning models based on Bayesian optimization." *Journal of Electronic Science and Technology* 17, no. 1 (2019): 26-40.

[6] Pelikán, Martin, David E. Goldberg and Erick Cantú-Paz. "BOA: the Bayesian optimization algorithm." (1999).

[7] J. Snoek, H. Larochelle, and R.P. Adams. Practical bayesian optimization of machine learning algorithms. In NIPS, 2012.

[8] https://towardsdatascience.com/quick-tutorial-using-bayesian-optimization-to-tune-your-hyperparameters-in-pytorch-e9f74fc133c2

[9] Makridakis, Spyros & Spiliotis, Evangelos & Assimakopoulos, Vassilis. (2020). The M5 Accuracy competition: Results, findings and conclusions.

[10] https://www.kaggle.com/competitions/m5-forecasting-accuracy/data

[11] https://www.kaggle.com/code/omershect/learning-pytorch-lstm-deep-learning-with-m5-data

[12] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR), 2015.

[13] https://towardsdatascience.com/hyperparameter-tuning-of-neural-networks-with-optuna-and-pytorch-22e179efc837