TIME SERIES FORECASTING OF MONTHLY ACTIVE USERS WITH LONG SHORT-TERM MEMORY NETWORKS

TECHNICAL CONTRIBUTIONS [DRAFT]

Matthew Louis Rosendin
University of California, Berkeley
Department of Industrial Engineering and Operations Research

April 13, 2018

Contents

List of Figures

1 Introduction

Our project, in partnership with Adobe Systems Inc., is an interactive dashboard that forecasts monthly active users for any Adobe software product. We've accomplished forecasts with excellent accuracy as far as 5 months into the future. The growth model's forecast can be used for a number of reasons, including as an input to forecasting revenue or as a measure of product health. While forecasting monthly active users is nothing new, our application of artificial neural networks (ANNs) seems to be a novel approach that has yielded promising results.

In this chapter of the paper I analyze and discuss the implications of our model's performance and describe my work in deploying our model. I begin by explaining the concrete goals of this work, then outlining the motivation, before finally diving into a technical introduction to our model. Following the basic conceptual discussion around our model, I start the discussion about how I refined the "raw" model code into a flexible, scalable, and configurable object. The translation of the model into a manageable machine learning software system led to additional improvements to the model, such as hyperparameter optimization. A byproduct of the optimization was an exhaustive experimentation of model performance metrics. The underpinning of the preceding work is the user interface. I discuss my work in front-end and systems engineering that resulted in the final deliverable: a forecasting dashboard powered by machine learning.

1.1 Goal

Our team's foremost goal is to create a practical and interactive product growth model for Adobe senior management. My individual goal was to design and create a user interface to interact with the model. Throughout my contributions, I've added additional value around testing the performance of our model and gathering relevant metrics.

1.2 Motivation

The motivation for creating a dashboard is straightforward. The dashboard enables the user to train the machine learning model resulting in performance metrics and the monthly active user forecast. The motivation for the section on hyperparameter optimization was simply to improve the results of our model. From the resulting tests, we've found that our model's forecasting capability had improved. Part of the discussion in this paper is reserved for analyzing the source of model improvement.

1.3 Time Series Forecasting

In order to explain why we chose our particular model (a long short-term memory network) I will explain the characteristics of the problem we are solving through a brief taxonomy. The problem is best framed as, "how do we forecast monthly active usage multiple weeks into the future?". The most salient characteristic of this problem is that the data is a long sequence (i.e., data is chronologically ordered) of values, known as a *time series*. What we are trying to achieve is a forecast of time series data, so the solution to the problem is known as *time series forecasting*.

1.4 Autoregressive Models

The simplest model might be an *autoregressive* (AR) model in which the forecast values are regressed on prior data:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \tag{1}$$

where c is a constant, ϵ_t is an error term, and $\phi_1, ..., \phi_p$ are the parameters to be estimated by linear regression. To forecast a value for the present time t, we specify a parameter p, known as the "order" or "lag". With $p = 3$ the corresponding model would look like this:

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \phi_3 X_{t-3} + \epsilon_t \tag{2}$$

We would call this an autoregressive model with a lag of 3 time steps where *time steps* are defined as the number of sequential data points. The autoregressive model can be notated as $AR(3)$ to indicate its order with $p = 3$.

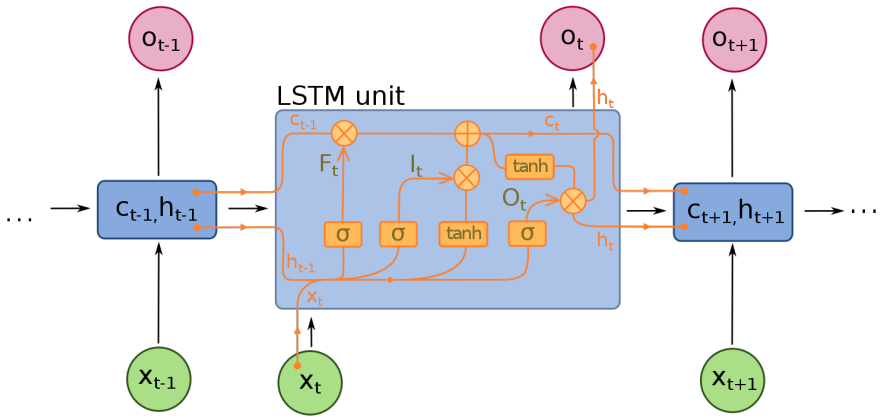
In our work we use the concept of "lag" from the autoregressive model in our supervised learning model. Our dataset is multivariate, meaning there are more than two observations for each time step. Since we would like to forecast multiple weeks into the future, our model is multi-stepped.

1.5 Recurrent Neural Networks

There are many good reasons to justify using an artificial neural network. Much of the recent literature on time series forecasting is focusing on the advantage of ANNs versus other methods, including autoregressive interactive moving average (ARIMA) models. For instance, Ahmed et al. and Zhang show that ANNs are shown to be superior for generalized time series problems [2].

Although complex models such as neural networks can hard to interpret, our project team values performance over interpretability. Furthermore, we have designed features similar to those in financial time series models. For financial stock data, Adebiye et al. show that ANNs are superior to ARIMA models [1]. Lastly, ANNs are capable of modeling non-linearities which can improve the forecast.

Figure 1: Diagram of a one-unit Long Short-Term Memory (LSTM) network



Recurrent neural networks (RNNs) are a class of artificial neural network where connections between units form a directed graph along a sequence. In other words, RNNs retain state at one time to the next, using the previous state's output for the current estimate. This characteristic enables multi-stepped time series forecasting. A particular architecture of RNNs known as Long Short-Term Memory (LSTM) allows the model to recognize and retain short-term patterns for long periods of time. This architecture is used best where data from an earlier state needs to be recalled at a later state. Examples of LSTM networks in industry include Google Voice [3].

2 Object-Oriented Abstraction

- Working model in a jupyter notebook
- Needed to automate runs
- Needed to organize the code
- Needed to pass in arbitrary data

2.1 Decomposition

- Break complex system into parts that are easier to conceive, understand, program, and maintain.

2.2 Encapsulation

- Restricting components of model
- Allowing specification of options

3 Hyperparameter Optimization

- Learning paramters (hyperparameters) vs. model parameters (learned parameters or the weights for features in regression)
- Runtime and compute resources

3.1 Cross-validation

Train/test split time series data samples observed at fixed time intervals, in train/test sets. No shuffling. Successive splits must have higher test indices. In the kth split, it returns first k folds as train set and the (k+1)th fold as test set. Note that unlike standard cross-validation methods, successive training sets are supersets of those that come before them.

3.2 Grid Search

This search exhaustively generates candidates from a grid of parameter values specified with the `param_grid` parameter

3.3 Results

3.4 Discussion

4 Systems Engineering

4.1 Server Infrastructure

All software written for this project was developed and tested on local machines. System specifications of the host machine producing the results in this report are shown in Figure 2. The system runs Python 3.6.4 and we've tested on local machines (macOS) and a Docker virtual host container running Ubuntu 16.04. The Docker virtual host container can scale to run multiple instances on a cloud computing platform without any modification to the source code.

Figure 2: Host machine system information

4.2 Distributed Task Queue

The application can be deployed onto HTTP-accessible hosts where multiple clients can

5 User Interface

5.1 Forecast Visualization

Figure 3: User Interface: Dashboard

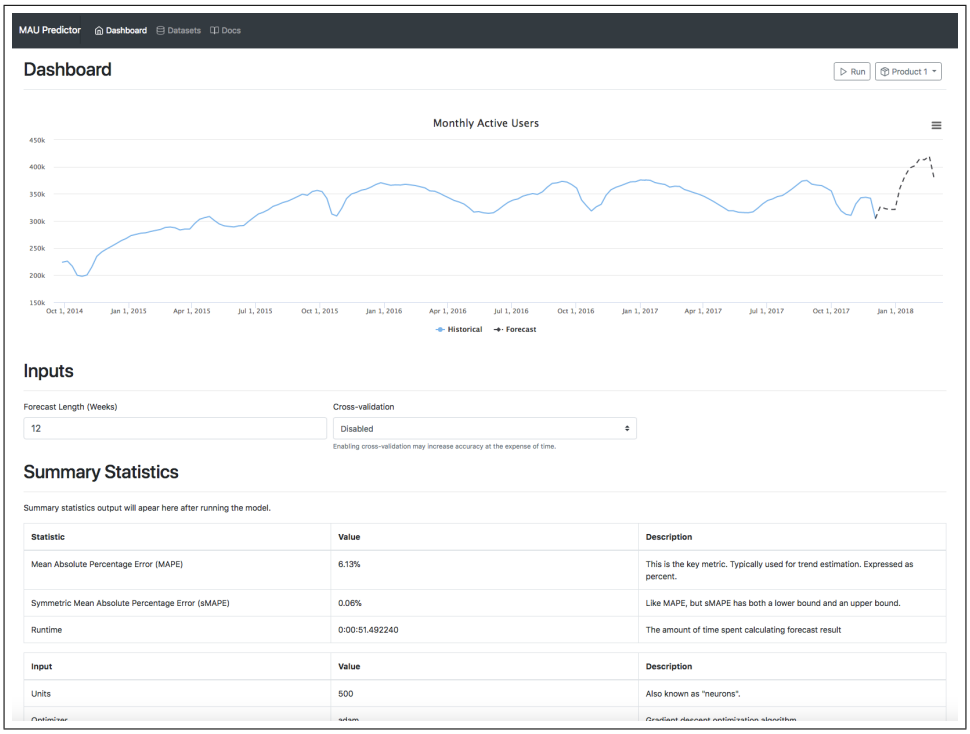


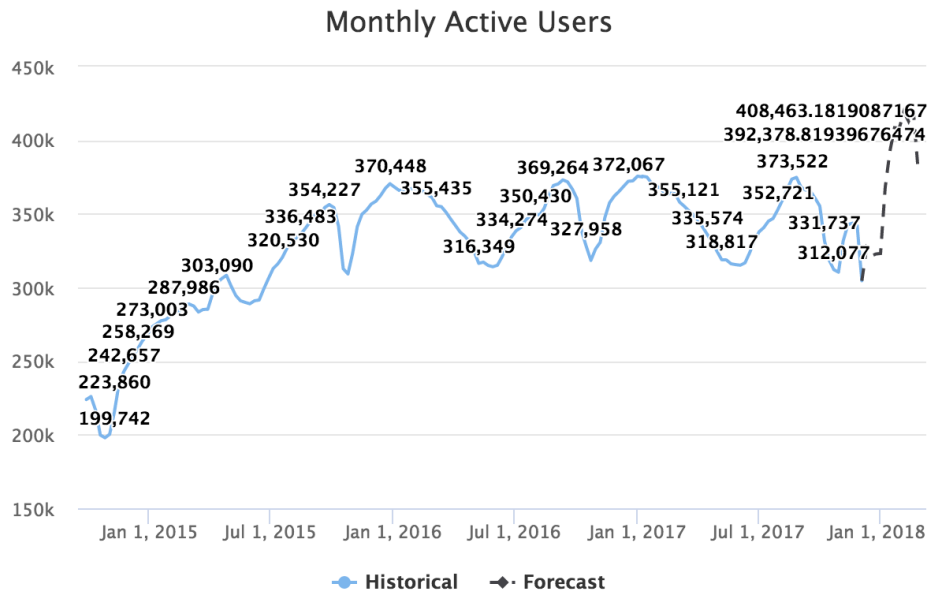
Figure 4 shows the historical monthly active user trend of Product 1 with the 12 week forecast in the dashed line.

Figure 5 shows the forecast for Product 1 over a 12 week period (December 3rd - February 25th 2018).

5.2 Documentation

Wrote documentation (akin to a usage guide)

Figure 4: Product 1 Trend: 6.24% MAPE



6 Next Steps

6.1 Websockets

6.2 Data Pipeline Automation (Apache Hadoop)

6.3 Dimensionality Reduction by Feature Selection

6.4 Improving Hyperparameter Optimization Efficiency

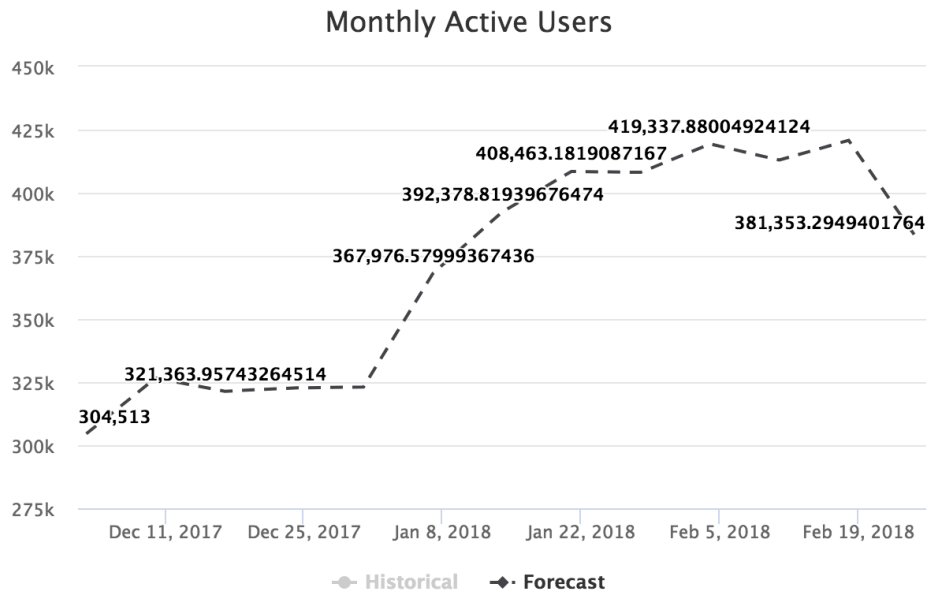
6.5 Confidence Interval Visualization

A major upgrade would be a visualization of forecast confidence intervals on the current graph.

6.6 Distributed Machine Learning Clusters

As discussed previously, the Docker container can be scaled to run on a distributed system.

Figure 5: Product 1 Forecast: 6.24% MAPE



7 Conclusion

In conclusion, ...

References

[1] Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and Artificial Neural Network Models for Stock Price Prediction. *Journal of Applied Mathematics, 2014*.

[2] Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews, 29*(5-6).

[3] Beaufays. (2015, August 11). The neural networks behind Google Voice transcription [Blog post]. Retrieved from https://research.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html.