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

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

April 13, 2018

Contents

List of Figures

1 Introduction

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. First, a quick word on the motivation. I believe that our model is a novel application of machine learning to model active user growth. The results of our forecasts on the testing set are remarkable and therefore I would like to explain our choice of model and explore the forecast results in more detail in the following sections.

In order to explain why we chose a long short-term memory network to model active user growth 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., it 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*.

1.1 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:

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

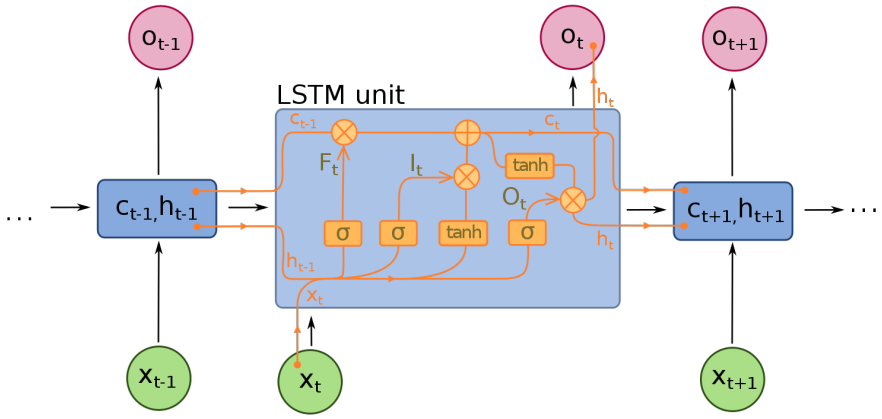
We would call this an autoregressive model with a lag of 3 time steps where *time steps* are defined as the number of sequential data points. The autoregressive model can be notated as $AR(3)$ to indicate its order with $p = 3$.

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.

1.2 Recurrent Neural Networks

There are many good reasons to justify using an artificial neural network (ANN). 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. 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. 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 [1].

1.3 Motivation

Senior management needs to interact with this tool.

1.4 Goal

The goal is to make the model interactive while tuning and testing.

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

4.2 Distributed Task Queue

Job scheduling via Celery

4.3 Forecast Visualization

4.4 Documentation

5 Next Steps

5.1 Data Pipeline Automation (Apache Hadoop)

5.2 Dimensionality Reduction by Feature Selection

5.3 Improving Hyperparameter Optimization Efficiency

5.4 Statistical Significance Boundary Visualization

5.5 Distributed Machine Learning Clusters

6 Conclusion

References

[1] 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.