TECHNICAL CONTRIBUTIONS [DRAFT]

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

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.*

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 (just ignore it for now), and $\phi_1, \ldots, \phi_p$ are the parameters to be estimated by 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 the number of sequential data points. The autoregressive model can be notated as $AR(3)$ to indicate its order with $p = 3$. To illustrate the next component of our model you can imagine we are predicting the fourth time step, where $t = 4$:

$$X_4 = c + \phi_1 X_3 + \phi_2 X_2 + \phi_3 X_1 + \epsilon_4 \tag{3}$$

Supposing we only have only two data, $X_1 = 1$ and $X_2 = 2$, how would we forecast the next 2 time steps? We could solve for all prior unknown data points in chronological order by choosing an error term and adjusting the parameters by regressing.

$$\begin{aligned} X_4 &= c + \phi_1 X_3 + \phi_2 X_2 + \phi_3 X_1 + \epsilon_4 \\ X_3 &= c + \phi_1 X_2 + \phi_2 X_1 + \epsilon_3 \end{aligned} \tag{4}$$

$\sum_{i=1}^{q} \theta_i \epsilon_{t-i}$

- ARMA model
- ARIMA model
- Supervised learning: using lags, differencing, and moving averages as features / one-stepped vs. multi-stepped / univariate vs. multivariate. ANNs can model non-linearity and have been shown to perform better than ARIMA models on similar time series datasets.
- Recurrent neural networks
- Long short-term memory neural networks

- Sliding window strategy, also known as a lag method: The use of prior time steps to predict the next time step - Univariate vs. multivariate: Univariate Time Series: These are datasets where only a single variable is observed at each time, such as temperature each hour. The example in the previous section is a univariate time series dataset. Multivariate Time Series: These are datasets where two or more variables are observed at each time. - Onestep vs. multistep: One-Step Forecast: This is where the next time step (t+1) is predicted. Multi-Step Forecast: This is where two or more future time steps are to be predicted. - This time series is multi-stepped

- "The encoder-decoder recurrent neural network is an architecture where one set of LSTMs learn to encode input sequences into a fixed-length internal representation, and second set of LSTMs read the internal representation and decode it into an output sequence. This architecture has shown state-of-the-art results on difficult sequence prediction problems. The encoder-decoder architecture still achieves excellent results on a wide range of problems. Nevertheless, it suffers from the constraint that all input sequences are forced to be encoded to a fixed length internal vector. This is believed to limit the performance of these networks, especially when considering long input sequences, such as very long sentences in text translation problems." - Machine Learning Mastery (https://machinelearningmastery.com/attention-long-short-term-memory-recurrent-neural-networks/) - "A potential issue with this encoderâĂŞdecoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus." - Dzmitry Bahdanau, et al. ([Neural machine translation by jointly learning to align and translate](https://arxiv.org/abs/1409.0473))

1.1 Motivation

1.2 Goal

2 Object-Oriented Abstraction

2.1 Decomposition

2.2 Encapsulation

3 Hyperparameter Optimization

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

3.1 Cross-validation

3.2 Grid Search

3.3 Results

3.4 Discussion

4 Systems Engineering

4.1 Server Infrastructure

4.2 Distributed Task Queue

Job scheduling via

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

7 Reference