Machine Learning Engineer Nanodegree
Capstone Proposal

Clara Zang
September 12th, 2017

Domain Background

From stock market prices, weather forecasting to speech and image recognition, real world data is usually wildly collected by taking account of time. This kind of data is known as time series. Different fields encapsulate different problems depending on the purpose of data, for example finance market is interested in the stock prices for the future days while a robot system may need to recognize the next visual signal. So the time series analysis can range from classification, segmentation, to anomaly detection and prediction.

Time series analysis is attracting me by its challenge in feature extraction and its openness to technique exploration. Traditional statistical method like ARIMA and machine learning algorithms like deep neural networks can find a place in this field.

Problem Statement

Given approximately 145K Wikipedia articles with their historical daily views, starting from July 1st, 2015 to December 31st, 2016, the problem is to forecast future daily web traffic, from January 1st, 2017 up until March 1st, 2017 for each article. For example, we can simply use the today's views to predict tomorrow's views for all these Wikipedia articles, and we can measure our solution by comparing the difference between prediction and true values. The challenge here is that we have approximately 145,000 time series instead of one, and the fact that daily views follow a random walk, the future daily views are hard to be predicted.

Datasets and Inputs

The datasets used are from data in Stage 1 of a Kaggle competition – Web Traffic Time Series Forecasting. The training dataset consists of approximately 145K time series. Each time series contains a Page name and a sequence of daily views for the corresponding Page, starting from 2015-07-01 to 2016-12-31. The Page name is made of article name, the Wikipedia project like 'en.wikipedia.org', type of access such as desktop and type of agent like spider, which is in the format 'name_project_access_agent'. For values in each time series of daily views, there is no difference between zero values and missing values, so a missing value may mean the traffic was zero or just that the data is not available for that day. Now that the stage 1 of the competition is over, I will also use another dataset containing the daily views for January 1st, 2017 up until March 1st, 2017 as the answer key to evaluate the model prediction performance.

Solution Statement

Since we have about 145K time series, it is not feasible to get different model for each time series, and traditional art of state models may not good at handling such huge different time series. Therefore, I would like to apply deep learning specifically Long Short Term Memory which has a nature of chain-like structure that is a good fit to sequential analysis. The range of daily views are huge, so necessary data transformation like log-scale, max-min scale or standardization is taken before modeling. We also need to appropriately define training, validating and testing sets. In this case, we use sequence of 60 values as target since we need to predict values for the next 60 days. The problem now becomes predicting a sequence by using a given sequence, i.e. sequence-to-sequence problem.

Benchmark Model

Since the prediction range is 60 days, the benchmark model I choose is to use the median of values of the previous 60 days as prediction of web views for the next 60 days for each Page. Choosing median is because it is not affected by outliers in the time series. With more than 145K sequences, median turned to be an easy and effective measurement in the predictions.

Evaluation Metrics

Kaggle uses **SMAPE** between predicted values and true values as the evaluation metric. SMAPE is short for Symmetric Mean Absolute Percentage Error, which is an accuracy measure based on percentage errors. The formula of SMAPE is

$$\text{SMAPE} = \frac{200\%}{n}\sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|}$$

where n is the number of observations, y_t is the true value and $\hat{y}_t$ is the predicted value where t = 1, …, n. SMAPE has a lower bound 0 and an upper bound 200%. One problem of SMAPE is that if the true value and predicted value are both 0, the formula returns error. So in practice, we define that if $|y_t| + |\hat{y}_t| = 0$, then SMAPE = 0; otherwise SMAPE is calculated by the above formula.

In the training process, I also use another metric Mean Absolute Error (**MAE**) which is a quality to measure how close predictions are to the actual values, and defined as

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$$

Project Design

First of all, I will prepare the datasets. Preprocess it by transforming the training dataset using log-scale and max-min scale. In order to validate the model, I take the last 60 days as y_validate

and the previous days except the first 60 days as X_validate. The choice of X_validate is to keep the same length as X_train which will be all days except the last 120 days, and y_train will be the 60 days before y_validate. In this case, there is no overlap between y_train and y_validate. So finally, X_test is all days except the first 120 days, which has the same length as X_train and X_validate, while y_test is unknow and represent views for the so-called future 60 days.

Second, get prediction results from benchmark models and so some explorative analysis. For each y_train/y_validate, calculate the median of views of its previous 60 days and the corresponding SMAPE value as the benchmark for later modeling. For example, I will try to analyze some of the given time series to find some patterns and possible seasonality. To get a general idea, aggregate on Wikepedia project countries, types of access and agents and plot the trends to explore any possible trends.

Lastly, build and tune Long Short Term Memory models. With X_train/X_validate/X_test and y_train/y_validate transformed and normalized, we are ready to build model and train on these datasets. In this step, we need to tune several (hyper)parameters like number of nuerons in LSTM, number pf epochs, batch size and so on. We also need to avoid overfitting by using validation set and add dropout layer, etc..

Reference

1. Gamboa, John Cristian Borges. "Deep Learning for Time-Series Analysis." *arXiv preprint arXiv:1701.01887* (2017).
2. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
3. Chong, Eunsuk, Chulwoo Han, and Frank C. Park. "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies." *Expert Systems with Applications* 83 (2017): 187-205.
4. Blog https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-networks-python/