Machine Learning Engineer Nanodegree

Udacity

Capstone Project

Proposal

Multivariate Time Series Forecasting

Ognian Dantchev

July 2nd, 2021

Table of Contents

1. Domain Background

Time series forecasting has many applications in various industries like: Medical (EEG reading), Healthcare (birthrate, hospital cases), Retail (unit sales per product per shop), Agriculture (crops yield), Finance (stock price), Transportation (number of passengers), IT (server utilization), etc.

Time series forecasting is applied when scientific predictions are needed based on historical data with time component.

In [1] they describe the terms and the use of ML in the field:

“Making predictions about the future is called extrapolation in the classical statistical handling of time series data. More modern fields focus on the topic and refer to it as time series forecasting.

Forecasting involves taking models fit on historical data and using them to predict future observations."

Time series forecasting derives from the much larger field of Time series analysis, https://en.wikipedia.org/wiki/Time_series Methods for time series analysis may be divided into two classes: frequency-domain methods and time-domain methods. They may also be divided into linear and non-linear, and univariate and multivariate.

2. Problem Statement

An attempt to predict the temperature for a given future period will be presented. It will be based on data series for the temperature and the atmospheric pressure, for the target and three other cities in proximity.

3. Datasets and Inputs

Weather data for the period 1931-2018 will be used for four cities in Bulgaria – Ruse, Sofia, Varna and Veliko Tarnovo. After struggling with the format of some of the local sources, I found that the National Centers for Environmental Information of the National Oceanic and Atmospheric Administration https://www.ncei.noaa.gov/ provide an access to a global source of climatic data. It was also used in [2], and one can pick almost any location in the world.

Here's the sequence to acquire, clean and reshape the data:

* generate the raw data set at https://www.ncdc.noaa.gov/ (https://www7.ncdc.noaa.gov/CDO/cdoselect.cmd – the old data request form)

* review the original dataset files

* resample data and helper functions

* remove data (where gaps are too big)

* add data (linearly interpolated from the neighboring values)

* save the clean, resampled data to a binary file.

This will be done in a separate Jupyter notebook, and then the clean data will be saved in NumPy .npy standard binary file format.

The raw dataset is uploaded to GitHub for review:
https://github.com/ogniandantchev/ML_engineer_nd009_capstone

4. Solution Statement

Recurrent neural networks (RNN) are a class of neural networks that is powerful for modeling sequence data such as time series or natural language. The 1st model will be based RNN: Gated Recurrent Unit (GRU). The 2nd one, I used in [3] will be based on Dilated Causal CNN, often used for audio processing.

One of the models will then be deployed on AWS.

5. Benchmark Model

An attempt will be made to compare both custom models – Gated Recurrent Unit (GRU) RNN and the Dilated Causal CNN to Amazon's own Gluon Time Series (GluonTS) model toolkit by Amazon Science.

6. Evaluation Metrics

The re-sampled data will be used to create "future" period by shifting the target-data.

Mean Squared Error (MSE) will be used as the loss-function that will be minimized and used as a metric. This measures how closely the model's output matches the true output signals.

At the beginning of a sequence, the model has only seen input-signals for a few time-steps, so its generated output may be very inaccurate. Therefore the model will be given a "warmup-period" of 50 time-steps where we don't use its accuracy in the loss-function, in hope of improving the accuracy for later time-steps, see [2]

7. Project Design

The implementation will be split into 5 or 6 smaller Jupyter Notebooks:

a. Data import and exploration; Data cleaning and pre-processing; Feature engineering and data transformation; Save the clean data in compact new format for use in other notebooks and upload to AWS bucket,

b. Split the data into training, validation, and testing sets; Define and train a Gated Recurrent Unit (GRU) RNN Model,

c. Split the data into training, validation, and testing sets; Define and train a Dilated Causal CNN Model,

d. Split the data into training, validation, and testing sets; Define and train a model from Gluon Time Series (GluonTS) toolkit by Amazon Science,

e. Evaluate and compare the models,

f. Deploy one of the models to AWS

8. References

1. Jason Brownlee, https://machinelearningmastery.com/time-series-forecasting/

2. Magnus Erik Hvass Pedersen, TensorFlow-Tutorials / GitHub / Videos on YouTube

3. Ognian Dantchev, Multivariate Time Series Forecasting with Keras and TensorFlow, GitHub repo