Time Series Forecasting with LSTMs

Description of data

We are presented with irregularly spaced out, once a day recordings of temperature and precipitation levels at 28 sports venues spread across North America. There are 2132 data points in total with each venue averaging about 70 data points. There are two exceptions to this: American Airlines Center and the Staples Center have an unusually high number of recordings of 148 and 146 respectively. The range of dates of recordings are as follows:

2017-07-01 - 2017-07-17
2017-10-17 - 2018-04-11
2018-07-02 - 2018-07-17
2018-10-16 - 2019-01-23

Precipitation levels are missing in 26 rows and are recorded as zero in about 1700 cases.

Problem description

Predict the temperature of the next day in a particular venue, Capital Center.

Formulation of solution

There were many factors to consider in the formulation of a solution:

Choice of data to be used:
- Train with data of only Capital Center
- Train with data of all venues
- Train with data of all date ranges
- Train with data from selected date ranges
- Train with only temperature data
- Train with temperature, precipitation and other generated features
- Create rolling windows of chosen lag size over the dataset to artificially generate more training examples

Choice of model architecture:
- Conventional LSTM
- Stacked LSTM
- Bidirectional LSTM
- CNN LSTM

Choice of data processing techniques:
- Whether to scale the data
- Whether to normalize the data
- Whether to de-seasonlize the data

Choice of hyper-parameters:

- Lag size of features (how many prior time steps of features to consider for training)
- Optimization function for back propagation
- Loss function

Data processing

American Airlines Center and Staples Center data were dropped as they are outliers in terms of the amount of data points recorded in those venues.
Null values in precipitation were filled with '-1'.

Dataset design

I experimented with all the data options detailed in the section 'Formulation of solution'. In the end, I found using data from all venues only from two date ranges of October to January, along with multiple features worked the best. The two date ranges were processed and treated separately so as to create features independently for each date range.

The features used were:
- Temperature
- Precipitation
- Interval in days between current and previous date
- Week of observation

Since the problem required a prediction in the latter half of January and since October - January is a large chunk of continuous time data, I thought it logical to train a model to predict a value for the latter half of January, using past data of an appropriate lag size. Using only Capital Center data in various ways proved inadequate due to extremely small dataset. Creating more training examples with the rolling window also failed.

Since I chose to use data only from one season, winter, there is no question of seasonality.

I tried both scaling and normalizing the input data but this was found to not add much value in terms of either optimization or loss minimization.

I tried various lag sizes of 3, 5, 10, 15, 18 and 20. 18 gave me the best results.

Experiments

There are two Jupyter notebooks in my submission. The code is not modular since it is presented in notebook format. One notebook is a model built in PyTorch that takes as input a sequence of a particular lag size, one data point at a time, predicts the next data point in the sequence after which the gradient of the loss function is back propagated through the model. The model is not fed its own prediction but is always given the ground truth as input in the next time step. This is a commonly used methodology called Teacher forcing that is used to assist easier convergence and increased stability of Recurrent networks. This goes on for the whole sequence of input and for all input sequences. This is one variety of time series or sequence prediction using LSTMs.

The other submission is a model built in Keras, a high level deep learning framework built on top of Tensorflow, and takes as input the entire sequence of a particular lag size at once and predicts only the next time step. I had wanted to build this model in PyTorch as well but was not able to finish it to my satisfaction by the deadline.

Of the two models, the latter worked best for this particular problem. I suspect this is the case because of the smallness of the dataset, and given more datapoints, the former can be expected to work much better as it is the textbook case of the usage of LSTMs.

Model details

Of all the LSTM model architectures tried, CNN LSTM was the best performing one. A CNN LSTM architecture is one that has a convolutional layer before the LSTM layer. The convolutional layer divides the input into subsequences and passes a convolution filter over the subsequences to generate feature maps. The output from this is then max-pooled and flattened. This output is then passed to the LSTM layer as a single input. The LSTM then predicts the next time step in the input sequence.

CNN filter size: 1 * 1
CNN number of feature maps: 32
Max-pool filter size: 2 * 2
LSTM hidden layer dimension: 100

I used Mean Squared Error loss which is the standard cost metric for time series predictions.

I tried SGD, RMSprop and Adam optimizers and settled on Adam since it performed the best.

I trained for 200 epochs in a single batch with early stopping.

Results

I divided the dataset into 90:10 train:validation split. Since the dataset is so small, I did not want to use too many samples for validation. Training and validation loss, prediction for Capital Center and plots of loss and predicted vs. actual values can be found in the Keras notebook.

References

https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
https://github.com/osm3000/sequence_generation_pytorch/blob/master/experiment.py
https://machinelearningmastery.com/how-to-scale-data-for-long-short-term-memory-networks-in-python/
https://machinelearningmastery.com/use-dropout-lstm-networks-time-series-forecasting/
https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/
https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html
https://www.jessicayung.com/lstms-for-time-series-in-pytorch/

Note: Every run of model fitting will result in different results due to inherent stochasticity of neural networks. Model finally picked was best performing among the runs.