Forex Trend Prediction using Deep Learning Networks

Pradyumna Kaushik, Abhishek Jain, Sreedhar Kumar Binghamton University

December 5th

Abstract

Most machine learning algorithms are designed for independent and identically distributed data but many interesting data, like time-series data, are sequential, which require the learning algorithm to use contextual information across data points.

Recurrent Neural Networks are specialised Neural Networks which can hold "memory" of past events. In particular a type of RNN architecture, Long Short Term Memory, has been found to be suited to learning long-term temporal dependencies without running into problems of exploding or vanishing gradients.

For this project we are working on applying the LSTM architecture to forex trend prediction. The main challenge will be in searching the optimal hyperparameters for the LSTM architecture and addressing overfitting by various strategies like addition of dropout in layers.

Readme

The purpose of this project is to zero in on the exact LSTM network structure and hyperparameters that will enable one to accurately predict future data values derived from the same data distribution.

Dataset

The Trade Weighted Index(TWI) is a weighted average of a basket of currencies that reflects the importance of the sum of USA's exports and imports of goods by country. This dataset consists of 304 time points, each being the month ending value of the index, from May 1970 to August 1995. The interpretation of the effective exchange rate is that if the index rises, other things being equal, the purchasing power of that currency also rises.

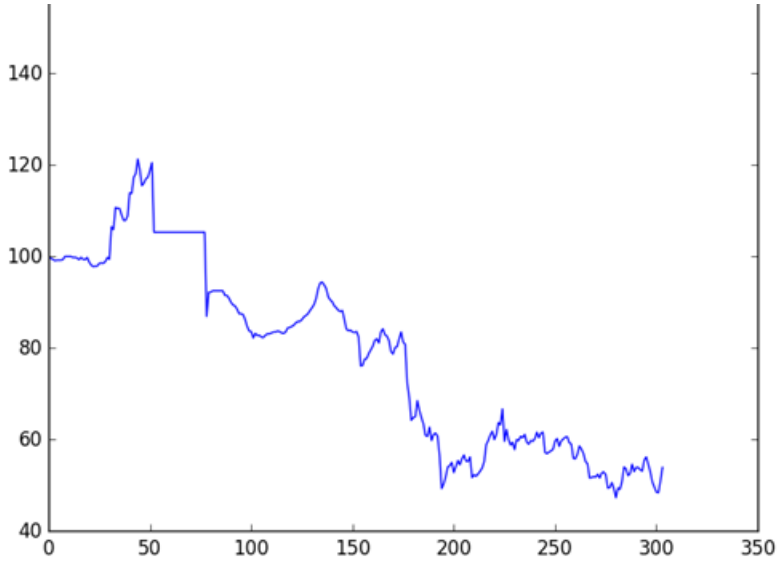

Figure1. Dataset

Methodology

In order to implement the LSTM deep learning model we used the Keras Deep Learning library. While there is sufficient documentation online to enable one to create the required models, we found that iterating through the parameter space manually was difficult. So we decided to automate the process by developing a GUI that provides an easy to use interface, with knobs for tuning the hyperparameters of the network.

We limited our experimentation to adjusting the following important hyperparameters while keeping the choice of Optimisers limited to Stochastic Gradient Descent(SGD), RMSprop and Adam. (Please note that of the three chosen Optimisers, SGD has a non-adaptive learning rate while RMSprop and Adam have an adaptive learning rate.) For each of the so obtained neural network models we trained them on 50% of the data and evaluated its performance on the rest of the data. The evaluation metric used was test-RMSE(Root Mean Squared Error).

1. **Network Architecture:** This is determined by the number of layers and the number of computational LSTM units within each layer. In order to experiment with the above we adopted the default parameter settings for each of the three Optimisers as given in the Keras documentation and began the experimentation by varying the number of layers from 1-6 and the number of units from 10-60 for each of the hidden layers. (The range was randomly chosen)
2. **Optimisers:** For the best architecture arrived at from the previous experimentation, we went about selecting the best weight learning algorithm from among SGD, RMSprop and Adam using their default settings.
3. **Learning rates:** For the best selected architecture we then experimented with the learning rates for each of the three Optimisers and went about looking to obtain the best combination of Optimiser and Learning rate.
4. **Momentum:** For the three Optimisers we then went about selecting the optimal momentum values corresponding to the best architecture and optimal learning rates obtained from the previous experiments.
5. **Dropout:** Dropout is a technique that is used to reduce overfitting in Deep Neural Networks. In order to further refine the accuracy of the optimal model obtained until now, we varied the dropout percentage looking for a setting that would further reduce the RMSE.

Results

Please refer to the ppt or its pdf version for a full listing of the results. For the given dataset, assuming that future data points will be from the same data distribution, the optimal hyperparameters for the LSTM Deep Learning Neural Network are

1. No: of layers = 4, Layer dimensions = [1,60,60,1], Optimser = Adam, Dropout-recurrent units = 0.4, Learning rate = 0.001, Momentum = 0, Epochs = 100, Test-RMSE = 2.071.
2. No: of layers = 4, Layer dimensions = [1,60,60,1], Optimser = SGD, Dropout-recurrent units = 0, Learning rate = 0.1, Momentum = 0, Epochs = 700, Test-RMSE = 1.980.
3. No: of layers = 4, Layer dimensions = [1,60,60,1], Optimser = SGD, Dropout-recurrent units = 0, Learning rate = 0.01, Momentum = 0.9, Epochs = 100, Test-RMSE = 1.983.

Increasing the number of epochs, while computationally costly, will provide the algorithm more iterations over the data, enabling it to converge to the right weights even for a plain vanilla optimizer.

Future directions

The current implementation of the GUI can only handle univariate time series data. It can be extended to multivariate time series data with a few changes and can be scaled up to handle more number of hyperparameters. We will be uploading the developed GUI on Github as a public repository so that anyone who wants to experiment or model time-series data with LSTM Deep learning neural networks, can use the LSTM GUI tool to build, run and visualise the results.

References

1. Greff, Klaus, Srivastava, Rupesh Kumar, Koutnk, Jan, Steunebrink, Bas R, and Schmidhuber, Jurgen. Lstm: A search space odyssey. arXiv preprint arXiv:1503.04069, 2015.

2. G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, CoRR, vol. abs/1207.0580, 2012

3. `https://datamarket.com/data/set/22tb/exchange-rate-twi-may-1970-aug-1995#!ds=22tb&display=line`

4. Greff, Klaus, Srivastava, Rupesh Kumar, Koutnk, Jan, Steunebrink, Bas R, and Schmidhuber, Jurgen. Lstm: A search space odyssey. arXiv preprint arXiv:1503.04069, 2015.

5. `http://cs229.stanford.edu/proj2015/054_report.pdf`

6. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

7. Schaul, Tom, Sixin Zhang, and Yann LeCun. "No more pesky learning rates." ICML (3) 28 (2013): 343-351.