Time-series analysis using ANNs, RNNs, LSTMs, and LSTMs with attention

Anna Epishova

July 29, 2018

Abstract

Recently Deep Learning became an important topic in the Machine Learning community and produced outstanding results in many scientific directions. In this project, I focus on Recurrent neural networks (RNNs) which can be used to analyze time series data such as economic and financial indicators, currency exchange rates, temperature or air pollution measurements etc. RNN can be used to forecast a value of a target variable based on its past values.

1 Introduction

Many different techniques were developed in order to forecasting time series. General statistical methods for forecasting time series have ranged from moving average and exponential smoothing to linear and nonlinear regression.

The Universal approximation theorem [Csá01] states that a neural network can represent a wide variety of interesting functions when given appropriate parameters. A lot of algorithmic techniques and network architectures were developed in order to achieve a good approximation.

In this project, I build and compare four types of ANN models: fully connected ANN, RNN, LSTM, LSTM with Attention. The main challenge is to develop a good solution for forecasting an outcome of a process of interest based on historical data. To achieve that I combine the existing techniques of function minimization and regularization with proper ANN architecture, investigate if techniques like LSTM [HS97] and Attention mechanism [BCB14] can improve the accuracy of the network.

I compared the accuracy of forecasts produced by ANNs on two data sets. First one relates to the measurements of a physical process, and the second one contains financial data.

2 Problem Description

We have two datasets which contain time series. The goal is to build deep neural networks which can learn the temporal patterns in data and predict a value of future observation. We create and compare ANNs, RNNs, LSTMs, and LSTMs with attention. For those models, we compare the accuracy of predictions and the speed of the training process.

3 Our Idea

First, we preprocess the input data. We split it into the training and testing datasets. In every case, we use last year of observations as the testing data and the rest ones as training data.

We begin by creating and training simple fully connected ANNs in order to choose the loss function and optimization algorithm.

Next, we train a big fully connected ANN with the chosen hyperparameters. We compare its performance with and without regularisation. For this model, we chose the number of layers and units so that the model would correspond in its complexity to our recurrent networks.

Next, we build several RNN models. We vary the number of observation which RNNs used to learn temporal patterns. After that, we create LSTM models which get the same inputs as corresponding RNNs.

```
Mean absolute error of a naive predictor 13.236967827769714
Mean squared error of a naive predictor 630.882408079342
```

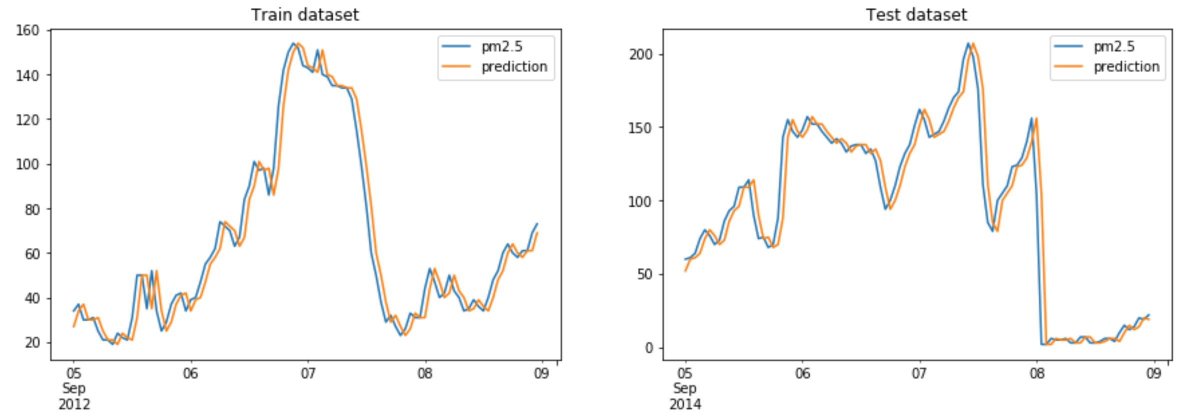

Figure 1: Baseline predictions.

Finally, we implement Attention Mechanism and use it to train two final models. Additionally, we train those two models on our second dataset.

For each model, we provide a brief description of the training process, assess the accuracy, and plot the model's predictions.

All networks were trained on two laptops. The combined CPU time to train all developed networks is approximately one week.

4 Details

4.1 Data Exploration and Hyperparameter Tuning

First dataset contains observations of air pollution from 01-01-2010 to 12-31-2014. The measurements were gathered each hour, so to avoid messy visualizations we plot the mean value of air pollution measurement for 3 days. We can observe that the air pollution increases during the winter months.

In order to make comparisons, first, we establish a baseline based on a common-sense approach. We suppose that without using any analysis a common-sense approach is to always predict that the target value 1 time-step from now will be equal to the target value right now. So, we can use this baseline in order to demonstrate the usefulness of advanced machine learning models.Figure 1 depicts the baseline predictions. We plot forecasts for several days in 2012 and 2014. The predicted values are plotted in orange and the true observations are plotted in blue.

First, we compare two loss functions: Mean Absolute Error and Mean Square Error. We begin by training the networks on time series consisting of one-day observations. So, as an input, the networks receive 24 air pollution measurements. The lowest value of the MAE loss function on the validation set is 12.2514 which is a bit lower than the baseline MAE. Network trained with MSE loss function produced similar results. So, if there is no much difference between two loss functions, we choose MSE.

Next, we compared RMSProp optimization algorithm with Adam. Again, on our dataset, we did not observe any significant change in models' performance. So, we choose Adam for our future models.

4.2 Fully connected Neural Network

After deciding on the hyperparameter values we train ANN with two fully connected layers. This model gets 186 input values which are the observations for 7 days. For this model, the total number of trainable parameters is about 2800. We chose this type of architecture in order to match the model's complexity with our future RNNs. Complex RNNs are notorious for being difficult to train. So, we will not have much flexibility in changing the number of units for those models.

First, we trained the above model without any regularisation. It converged during 1000 iterations and produced predictions shown in Figure 2. After that, we added a Dropout layer after each fully connected layer. However, the resulting model did not converge to a good local minimum. So we

```
Train Score: 496.37 MSE (22.28 RMSE)
Test Score: 522.69 MSE (22.86 RMSE)
```

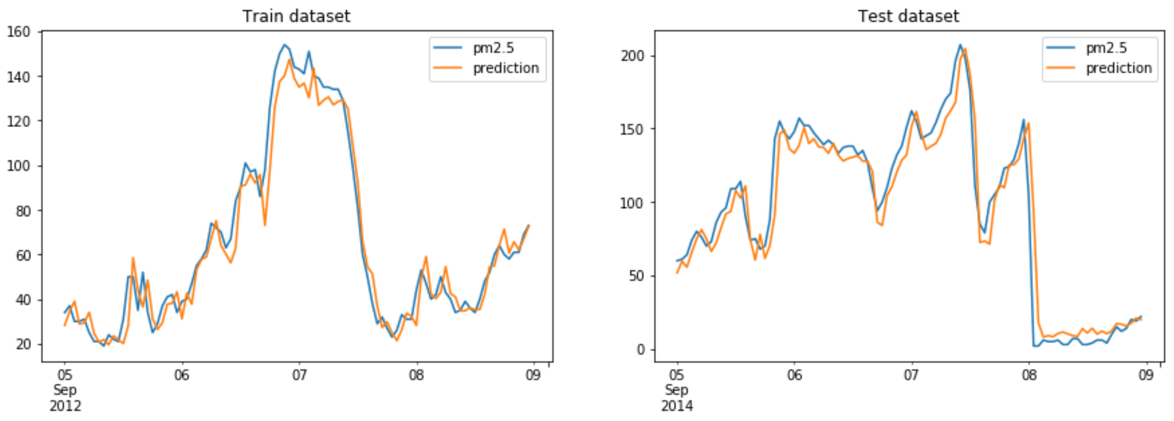

Figure 2: ANN predictions trained on 7-day time series.

removed one Dropout layer. New model covered almost as fast as unregularised model and produced a bit worse predictions on the training data and similar predictions on the validation data.

4.3 RNN Models

We note here that for our next models we do not use regularisation. We base our desicion on several research papers which showed that widely used approaches to regularise fully connected ANNs do not work with RNNs [ZSV14, MKS17]. In fact, there are regularisation techniques which can be applied to RNNs, however, they are non-standard and not implemented in Tensorflow or Keras.

We begin by creating a simple RNN which gets as an input only one past observation. This model has 109 trainable parameters. This simple RNN just echoes its input. It is expected because the model gets only one past value.

Next, we try to train RNN on one-day observations. So, as an input, the networks receive 24 air pollution measurements. We should note here that as the length of input time series increases the number of trainable parameters does not change. This RNN which was trained on one-day observations does not perform very well. We assume that one-day time series is too short to learn any patterns from the data. So, RNN learns that the best strategy is to echo its input. This model was as easy to train as the fully connected ANNs.

So, we try to train an RNN on 3-day and 7-day time series. Figure 3 shows the forecasts produced by RNN trained on 3-day tie series. This model can capture more patterns in the input data, and it less echoes its input. However, it is far from a desirable model. Additionally, this model required more iterations during the training and each iteration took more time in comparison to the above simple RNN. It is interesting, that with only 109 trainable parameters this RNN could capture some patters in data.

Finally, RNN trained on 7-day time series did not improve much after 200 epochs. In order to improve predictions we increased number on hidden units. This model has 1405 trainable parameters. It was the most difficult to train, and each iteration took considerable time. Complex RNNs are notorious for being difficult to train.

So, a simple RNN model is not better then a complex fully connected ANN. Additionally, more complex RNN required a lot of time to train.

Next, we check if LSTM can improve training and accuracy of forecasting.

4.4 LSTM Models

LSTM models are considered to be better than RNN in capturing time series data. So, we do not train any model on 1-day observations and begin by training my first LSTM model on 3-day time series.

Even though our first LSTM has 2521 trainable parameter it was problematic to train. Training process took about 1200 epochs but the model converged to a good local minimum. It outperformed

```
Train Score: 666.54 MSE (25.82 RMSE)
Test Score: 530.11 MSE (23.02 RMSE)
```

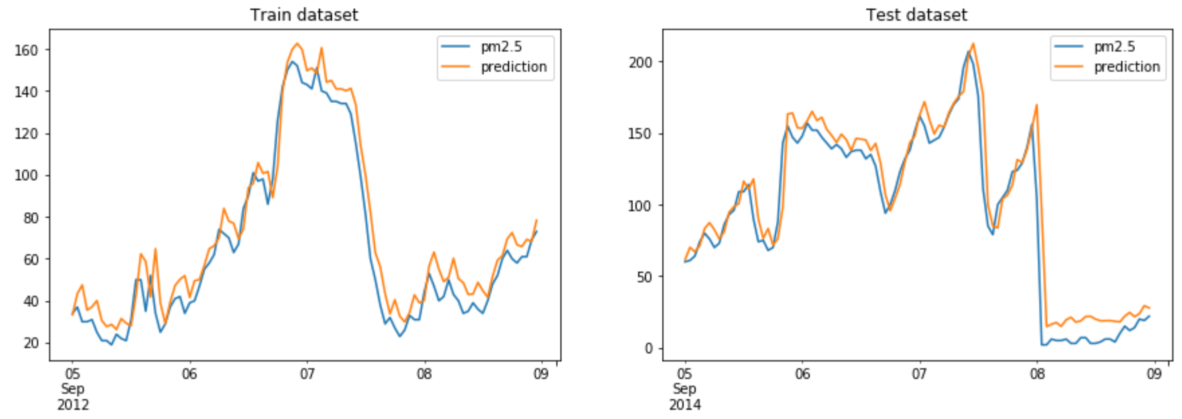

Figure 3: RNN predictions trained on 3-day time series.

```
Train Score: 528.63 MSE (22.99 RMSE)
Test Score: 466.22 MSE (21.59 RMSE)
```

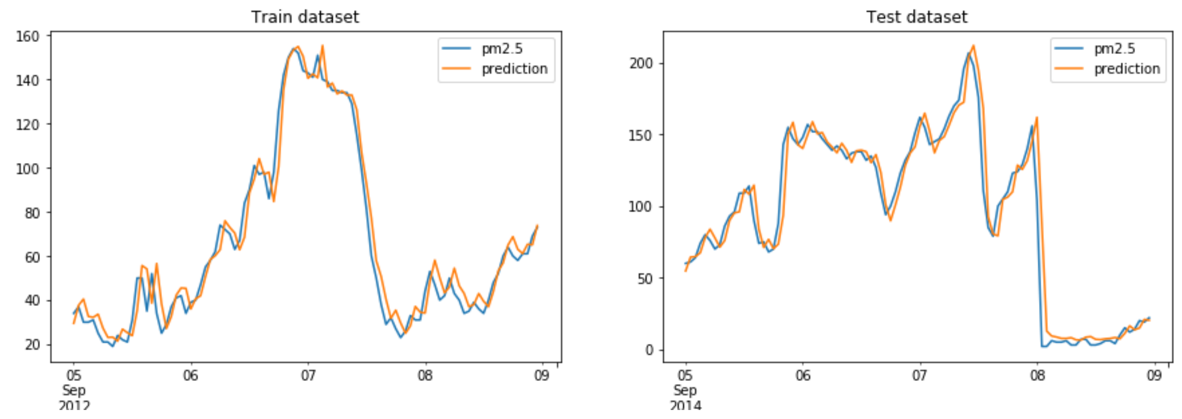

Figure 4: LSTM predictions trained on 7-day time series.

the baseline. In comparison, RNN model on comparable complexity did not converge.

The second LSTM has 9649 trainable parameters. It also converged to a good local minimum and bit the baseline. I created this big model in order to check if it is possible to train and if increased complexity can allow the model to learn better. Figure 4 depicts the model's forecast.

We can conclude that LSTM is a good improvement over RNN. We could train a big LSTM model, however it did not learn more patterns in data in comparison to simple LSTM.

4.5 LSTM Models with Attention Mechanism

We implement Attention Mechanism (module attention.py) and train networks on 3-day and 7-day time series.

Models with Attention performed slightly better than pure LSTM models. Both models have about 2700 trainable parameters. Inn both cases, models with Attention have smaller MSE than the corresponding LSTM models. However, plots changes only slightly. They are shown on Figure 5 We would say that the models more accurately forecast smooth changes. On the other hand, sharp rises or drops could be outliers in the dataset. For example, some measurements can be missing. So, there could be no pattern that the neural network can learn.

```
Train Score: 510.93 MSE (22.60 RMSE)
Test Score: 482.00 MSE (21.95 RMSE)
```

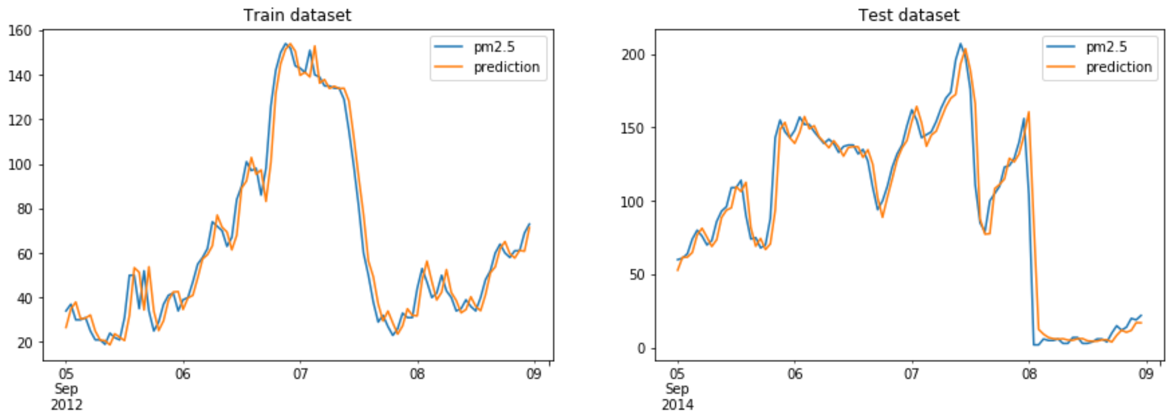

Figure 5: LSTM with Attention predictions trained on 7-day time series.

```
Train Score: 2473.34 MSE (49.73 RMSE)
Test Score: 4444.06 MSE (66.66 RMSE)
```

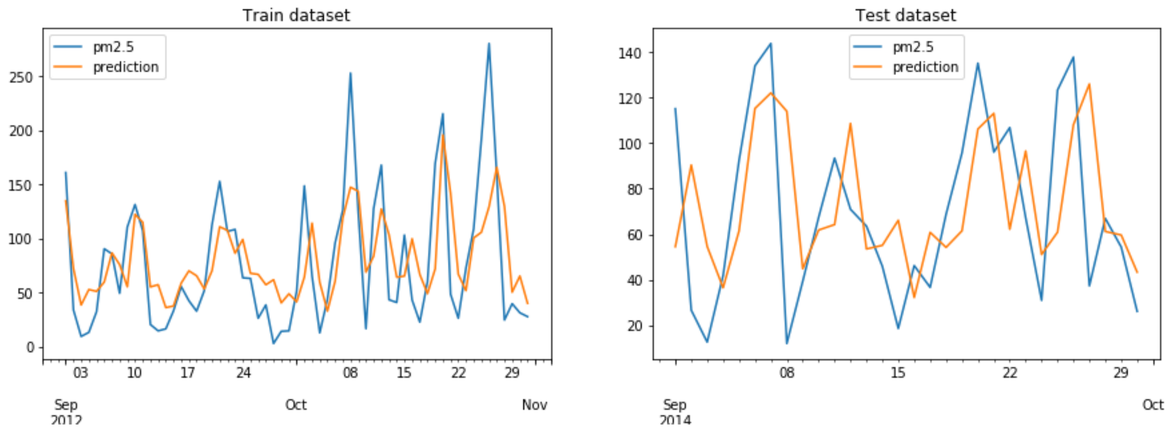

Figure 6: LSTM with Attention predictions trained on 72-day smooth time series.

4.6 China currency exchange rates dataset

Finally, we train LSTM with Attention on China currency exchange rates. We use the same approach to compute the baseline predictions.

We train LSTM with Attention on 72-day time series. This model does not echo its input. However, the MSE became a bit higher in comparison to the baseline. In a literature, the authors report much higher accuracy then our exchange rates model. We attribute it to bigger networks and longer training time.

4.7 Average Air Pollution Data

Finally, to check our hypothesis about outliers in air pollution dataset we transformed data by averaging observations for a day. Additionally, we filled in missing observations by copping values from the corresponding previous days. Thus, we obtained a dataset with a single observation for each day during 4 years. The baseline MSE $= 5136$

We trained LSTM with and without attention on this dataset. The resulting models outperformed a baseline with a considerable margin. The loss reduced more than in two times. The results are shown in Figure 6. On the other hand, two models performed eqully well on this dataset. So, the attention mechanism did not have a signifficant influence on the forecast accuracy.

5 Related Work

Researchers have used ANN methodology to forecast many nonlinear time series events. It was shown that ANN models with one or more hidden layers are able to identify the patterns from noisy data and hence give better forecasts [Zha01].

Furthermore, number of studies compared fully connected ANN models with RNN models. For example, Ho et al. [HXG02] showed that RNN models can outperform the feed-forward models in terms of lower predictive errors. Unlike traditional RNNs, LSTM model is able to learn the time series with long time spans and automatically determine the optimal time lags for prediction. These type of networks were also successfully used to predict time series [MTW+15, GES02].

Finally, a number of studies investigate the usefulness of Attention Mechanism [BCB14] applied to forecast time series [RVC+16, CBS+16, RE15].

6 Conclusions

In this project, we built and compared four types of ANN models: fully connected ANN, RNN, LSTM, LSTM with Attention. We compared predictions produced by those models on two datasets. Additionally, we checked if data smoothing could help to learn more patterns in the data set.

To summaries, on the first data set when the output variable varied significantly it was easy to outperform a baseline model. Complex fully connected ANNs could bit the baseline model. Next, RNNs were more difficult to train but they were a bit more accurate than simple fully connected ANNs. On the other hand complex fully connected model showed very good results and was as good as LSTMs with or without Attention. LSTM models were more powerful than RNNs and easier to train. For example, we could train LSTM with 7-day time series input which was not possible with an RNN. Finally, Attention Mechanism further improved LSTM models. The resulting networks were faster to train and produced good results.

In contrast, on China exchange rates dataset the model with Attention Mechanism did not show an improvement over the baseline model. We assume that for such a slowly changing time series more sophisticated models are required. Additionally, it is clear that predicting exchange rates are more difficult task than predicting air pollution.

References

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[CBS+16] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016.

[Csá01] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[GES02] Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*, pages 193–200. Springer, 2002.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HXG02] SL Ho, M Xie, and TN Goh. A comparative study of neural network and box-jenkins arima modeling in time series prediction. *Computers & Industrial Engineering*, 42(2-4):371–375, 2002.

[MKS17] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

[MTW$^+$15] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197, 2015.

[RE15] Colin Raffel and Daniel PW Ellis. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*, 2015.

[RVC$^+$16] Matthew Riemer, Aditya Vempaty, Flavio Calmon, Fenno Heath, Richard Hull, and Elham Khabiri. Correcting forecasts with multifactor neural attention. In *International Conference on Machine Learning*, pages 3010–3019, 2016.

[Zha01] Guoqiang Peter Zhang. An investigation of neural networks for linear time-series forecasting. *Computers & Operations Research*, 28(12):1183–1202, 2001.

[ZSV14] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.