Financial forecasting using LSTM

Michael Allemann
michael.allemann@epfl.ch
EPFL

Roman Bachmann
roman.bachmann@epfl.ch
EPFL

***Abstract*— Recent advances in Deep Learning have seen great success in applications dealing with time series. The question naturally poses itself, if those techniques can also be applied in financial forecasting. We propose a forecasting method using Long short-term memory (LSTM) units, taking only the sign of the logarithmic returns as input. The proposed method is evaluated in a rolling window fashion over four years between 2004 and 2016. We analyze the effectiveness of using LSTMs for financial forecasting.**

I. INTRODUCTION

Deep Learning techniques, applied to time series data, have been tremendously successful, revolutionizing for example the field of automatic speech processing. The question, if these techniques can also be applied to financial data leading to meaningful outcome, arises naturally. Following the lead of Jakob Aungiers' blog on time series prediction using LSTM deep neural networks[1], we investigate to what extent LSTMs can be used for financial market forecasting. The data we use is tic-by-tic data of ABB (ABBN), Credit Suisse (CSGN), Nestlé (NESN) and Novartis (NOVN) between 2003 and 2017. Due to large noise inherent in financial data, we perform preprocessing, compressing the data by only keeping 5 data points per asset per day in the form of a sign. The goal of this project is to forecast, if the the price of the stock will rise in the next week, or if it will fall. We verify our results by comparing to the historical prices listed on yahoo finance. Due to the heavy computations involved, we weren't able to create forecasts for all years, but chose the evenly spaced leap years (2004, 2008, 2012, 2016), in order to be using data from before and after the rise of deep learning. Additionally this data comes from before, during and after the financial crisis.

II. METHOD

In this section we specify the details of our data pipeline.

A. Preprocessing

The raw data consists of a compressed CSV file for each day and for each stock. The file contains 8 columns per stock, whereby the first two encode the date and time. We compute the average best offer (midprice), format the date and time as a Pandas datetime object and set it as the index and drop the rest of the columns. We then resample the data hourly keeping the mean of the midprice, and keep only 5 hourly data points (11:00, 12:00, 13:00, 14:00, 15:00). The choice of resampling hourly is arbitrary and could be optimized. We then keep only the sign of the log returns, drastically shrinking the size of the data. This preprocessing takes about 1 hour on a normal personal computer.

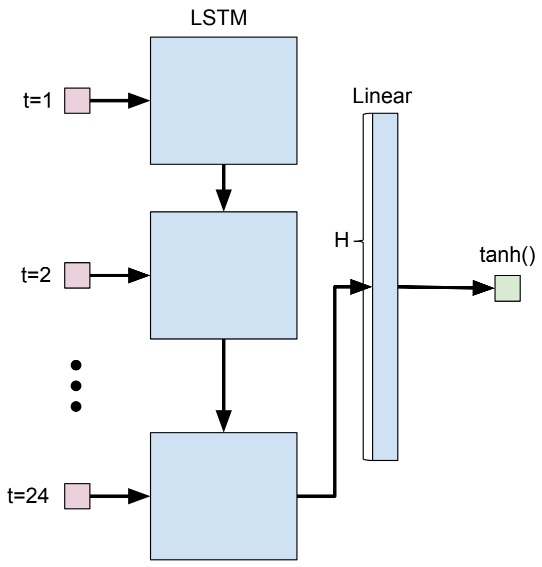


Fig. 1: Proposed network architecture with one LSTM layer and one fully connected layer. The output activation function $\tanh$ squeezes logits between -1 and 1.

B. Data pipeline

We want to forecast the sign of the cumulative sum of 25 future data points (5 days). For this we train an LSTM on 500 past data points, These two parameters are again an arbitrary choice and could be optimized. Within this training set we take a sliding window of 25 data points. We take the first 24 as an input and try to predict the 25th data point. When training is over, we take the next 24 so far unseen data points and try to predict the 25th one. We then take this predicted point, add it to the end of our data and feed this into the network again, predicting a further data point. We do this 24 times. When this is done, we shift the training and test data forward by 25 days and repeat the procedure until we have reached the end of the year.

C. Model architecture

For our predictions we have opted to use a neural network architecture using LSTM [2] units, shown in Figure 1. Specifically, the 24 input data points get transformed by the LSTM unit to a hidden representation $H \in \mathbb{R}^{64}$. We take this last output of the LSTM and feed it to a fully connected layer, giving us one output logit. The function $\tanh$ is applied on it to keep the results between -1 and 1. For prediction, we take the sign of this output.

The model is trained using all available training samples for 300 epochs, using a batch size of 32 and a learning rate of 0.005. We computed the Mean Squared Error loss between the predictions and the ground truth and adjusted the model's weights using the Adam optimizer with weight decay 0.001. All named parameters were chosen rather arbitrarily by trial and error, taking the ones that made training faster while still preventing it from diverging. Training the proposed model on a NVIDIA GTX 970 graphics card takes roughly 3 minutes per sliding window. For all sliding windows of all four years, the total training time was roughly 10 hours.

III. RESULTS

To visually assess our model on all selected assets over the four years, we decided to plot the cumulative sum of the testing window and overlay it with our predictions. To this end, we took the cumulative sum of each prediction window and superimposed it on the true data, as shown in Figure 2. Displaying the network predictions and ground truth data in this way allows us to very nicely compare the true direction of the stock over one week with the network's estimate.

The process of feeding the network it's own predictions (as is very common in synthesis tasks using recurrent networks) results in some self-reinforcing behaviour. If for example the network predicts the price going up, taking this information as a fact in the next evaluation round will result in the network predicting the same direction with high certainty. There are however some examples where this is not the case, as for example in the case of the NESN predictions in July 2016.

To evaluate our method numerically we fetch the real historical prices for the four stocks on Yahoo finance. For every forecast period of 5 days we calculate the cumulative sum and take the sign. We now have a sign for each stock and for each forecast period. This sign encodes the forecast our model gives for the price of the stock in 5 days. If the sign is 1, our model predicts that the price will rise. If the sign is 0, our model predicts, that the price will be the same and if the sign is -1, then our model predicts that the market will sink. To evaluate our forecasts we create a data frame containing the sign of our forecast per forecast period and the difference of the adjusted close price for each asset. We then calculate the point wise multiplication for each asset between the sign of our prediction and the difference of the actual price, then we sum everything up. This gives us one number for each of the four years and all stocks, that we selected as examples. By doing these operations we simulate investing in a stock, if the sign is 1, doing nothing if the sign is 0 and shorting a stock, if the sign is -1. 5 days later we put aside, what we earned or what we lost and again invest or short according to the next prediction. By summing what we put aside after each forecast period we can get an idea if our strategy would have yielded an overall gain or loss. The results are displayed in Table I. In the years 2004 and 2008 we would have suffered loss. In the years 2012 and 2016 we would have gained.

Stock/Year	**2004**	**2008**	**2012**	**2016**
ABBN	-0.50	5.10	2.21	-1.63
CSGN	-1.25	-26.13	4.38	-0.43
NESN	0.10	-0.71	2.45	6.49
NOVN	-6.21	19.79	-0.61	9.58
Total	-7.87	-1.95	8.43	14.01

TABLE I: Gains and losses of stocks in each tested year.

IV. CONCLUSION

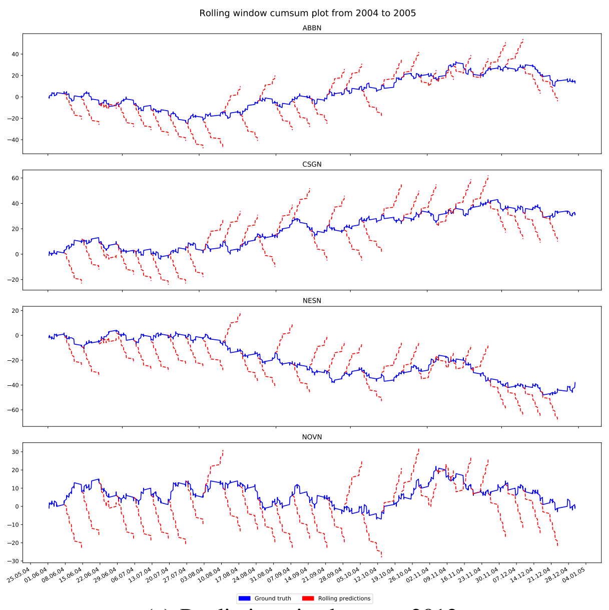
We have built a pipeline, which given tic-by-tic data of stocks performs preprocessing and keeps 5 data points per day in the form of a sign. We then train an LSTM on 500 data points, then we take the 24 most recent data points and predict the next 25 data points. We accumulate the values of the predicted points and take the sign, which encodes our forecast for the stock. We have made arbitrary choices for the resampling period, the number of data points we keep per day, the number of datapoints used for training, the size of the rolling window and the number of datapoints we forecast. All these parameters would need to be optimized before any definite conclusion can be made. Additionally we would have to test our model on many more years and many more assets in order to estimate the mean and the variance of the gains our method yields. From the four numbers we have it is not possible to come to a conclusion, however the results for the years 2012 and 2016 are rather promising.

REFERENCES

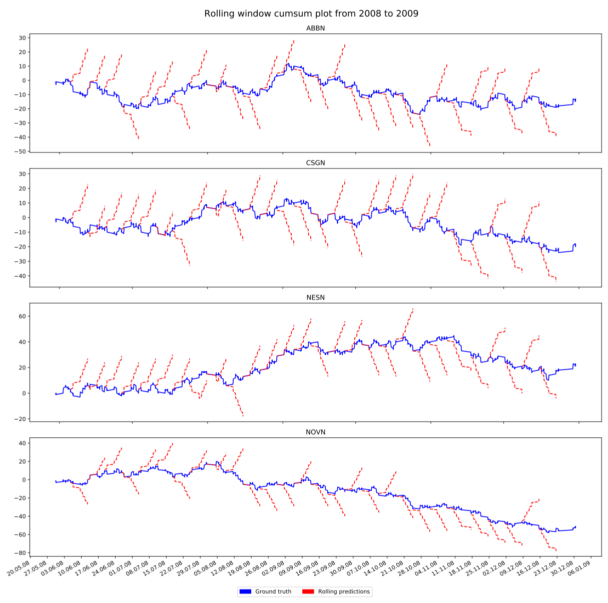
[1] Jakob Aungiers. Time Series Prediction Using LSTM Deep Neural Networks. `https://www.altumintelligence.com/articles/a/Time-Series-Prediction-Using-LSTM-Deep-Neural-Networks`. Accessed: 15.01.2019.

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

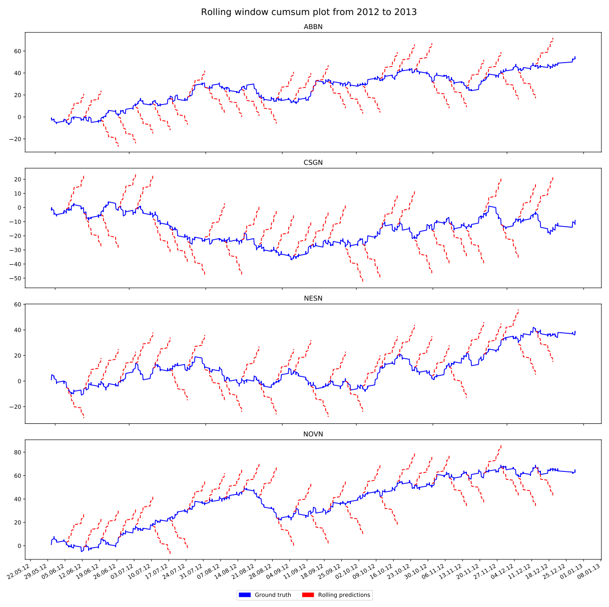
(a) Predictions in the year 2004.



(b) Predictions in the year 2008.



(c) Predictions in the year 2012.



(d) Predictions in the year 2016.

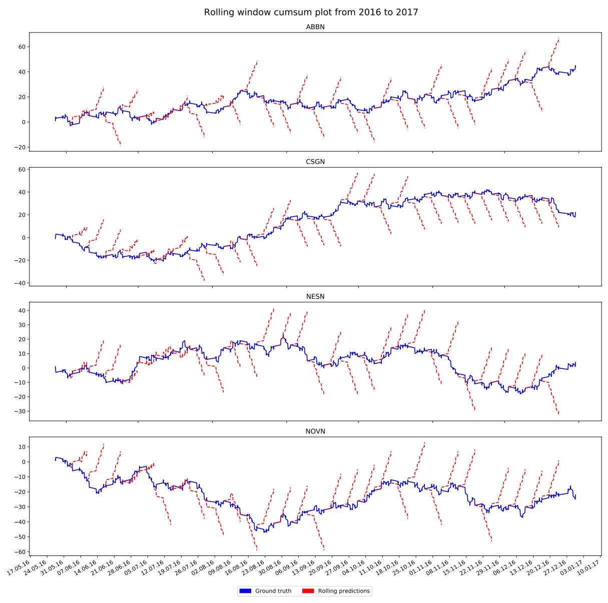


Fig. 2: Predictions on a rolling window of cumulative signed returns.