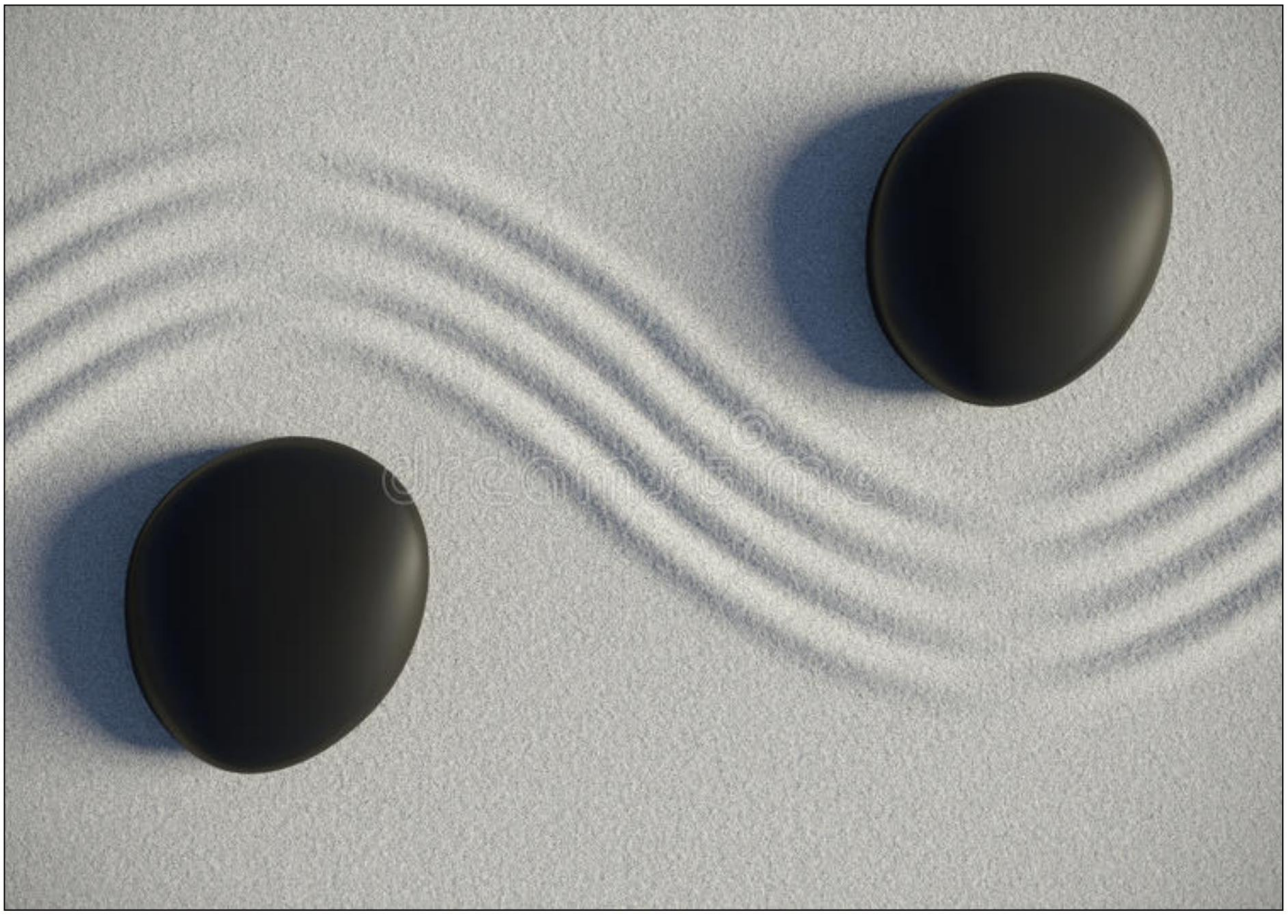
Stock forecasting using Deep Learning for CS230

Author: Francis Tran, fon.tran@gmail.com

20 May 2018



Introduction

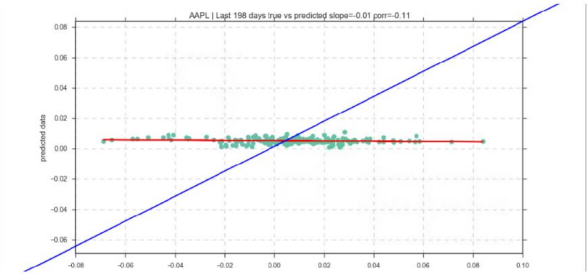
The problem statement is simple. Does deep learning have predictive power for financial stocks and if it does how is it compared to the traditional alpha predictive model like Multi Factors model (MFM). This paper has goal to 1) show how the predictive power of Deep learning combining different model sequence model like LSTM and combined them with filtering model like Wavelet and PCA and 2) it will compare the result with multi-factor model in terms of their predictive power. We will use spearman ranking correlation between true and predicted value to compare different models.

Baseline

We started the model based on available github repository [1] and modify the model to fit our requirement. Namely we modified the model to:

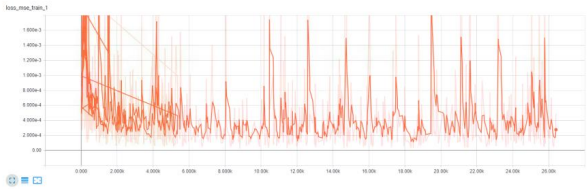
1. Enable more than one factors instead of just close price. In fact it can now add different factors
2. Different way to measure the accuracy
3. Different forecast horizon (20days, 5 days etc)
4. Changed to model to work on with python 3+ instead of 2.7

The data set at the stage is still using what the github has. We changed to enable more than one factor and to see the predictive level with a very simple model. As expected the predictive power is very low.



5 days slop (red) is 0.01 and corr=-0.11

True (green) vs predicted (orange)



The above 2 graphs are based on 5 days forward returns using close, high, low, open price to predict. The model was a simple 1 layer LSTM model with 128 neutrons and 50 time steps. We can see that that the model has no predictive power as the slop of (true vs predicted) is near zero and have negative correlation with each other. Not only that, the lost is very bouncy and not decaying gradually.

Next Steps

The next steps are to focus on improving the predictive model using Deep Learning by focusing on 2 areas: Data quality and Model complexity:

Data: we will use more data from QuandL [3]. There are about 80+ factors covering 3 areas: Fundamental factors [4] (e.g: Price/Earning), Economic [5] (e.g: Interest rate) and Technical [6] Factors (e.g: MACD). We want to predict 20 days market excess returns because anything less will be too volatile and not tradable due to high transaction cost. We want excess return because that shows the true alpha of a stock and not because of the markets. Data is from 2006 all the way to 2018. We will use mainly 50 stocks to test different scenarios. We sometime test it on 200 names to see the impact of big data. 50 names will give us about 150K training data and 200 names about 450K. 5% will use to do dev and test data.

Model & Related work (part I data filter): We will try to replicate the model propose by Bao et al [2] that combines wavelet and LSTM model. We will also try different model that combines different filtering method like PCA as suggested by Chan et all [7]. Wavelet is used to separate the high / low frequency of a time series and to reduce noise as suggested by Mikko [8]. Similar to Bao, two levels wavelet transformation is applied to our data to reduce the risk of overfitting. Similarly we want to try the usage of PCA for dimension reduction whether it will help in the prediction power of our model. Fig 1 shows how we smooth out one of our factors. The final part of data cleansing/smoothing is to apply standard normalisation so all factors are entered around zero and the standard deviation is one.

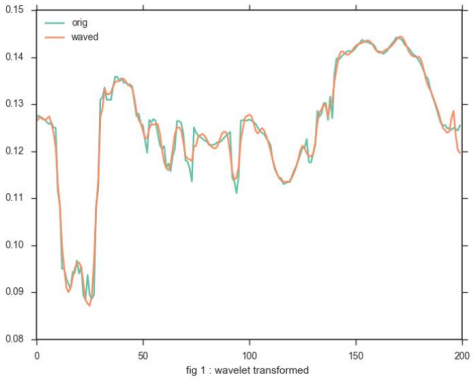


fig 1 : wavelet transformed

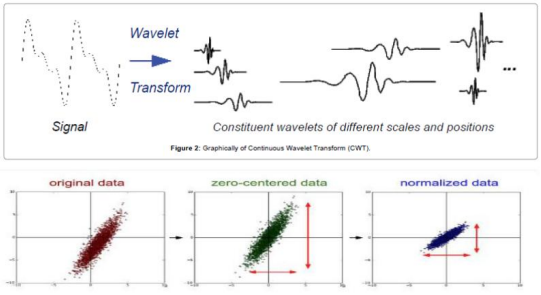


Figure 2: Graphically of Continuous Wavelet Transform (CWT).

Model & Related work (part II Deep Learning model): Our goal for this project is to build a model inspired by the Bao et al model and using with (or without) different encoding and decoding mechanism. The main part of the model is to use a sequential deep learning model like LSTM [9]. One of the main reason of using LSTM is its ability to remember long memory using its three gates (input, forget and output). So we capture the effect of changes over last days, week or even months to see what type of changes will affect future stock performance. The main question is how many steps back should should we have. This is part of analysis we will conduct. Before feeding the time series into LSTM, we pass the filtered data into three layers of Stacked AutoEncoder (SAE) to generate better approximation to non-linear [10] aspect of the financial data. We also use Relu activation (function f()) with SAE. Together, SAE and two layers of LSTM, will be combined to act as an encoding component. We will use the hidden

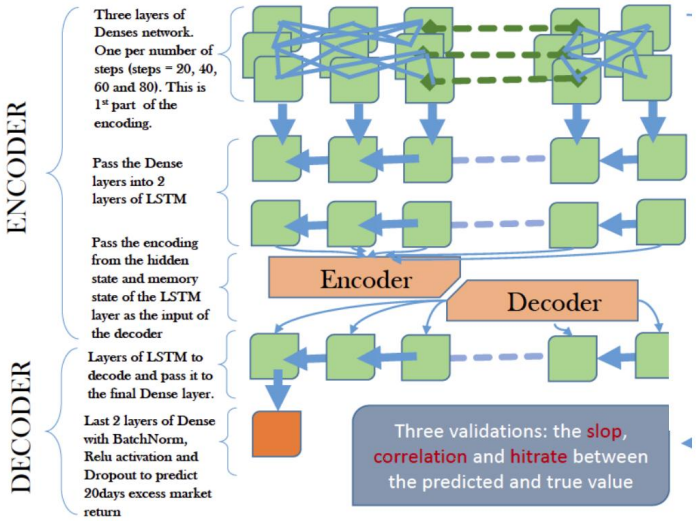
$$
\begin{aligned}
i_t &= \sigma\left(x_t U^i + h_{t-1} W^i\right) \\
f_t &= \sigma\left(x_t U^f + h_{t-1} W^f\right) \\
o_t &= \sigma\left(x_t U^o + h_{t-1} W^o\right) \\
\tilde{C}_t &= \tanh\left(x_t U^g + h_{t-1} W^g\right) \\
C_t &= \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right) \\
h_t &= \tanh(C_t) * o_t
\end{aligned}
$$

$$a(x) = f(\mathbf{W}_1 x + b_1)$$

$$x' = f(\mathbf{W}_2 a(x) + b_2)$$

and output states of last LSTM layer and feed them to the Decoder.

The Decoder component uses one layers of LSTM and output the last steps into two layers of Dense layers followed by Batch Normalisation, LeakyRelu activation and Dropout layer and finally, the prediction: the 20 days excess market return.



$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$

We choose to use standard RME loss and Adaptive Moment Estimation (Adam) Optimiser for it's adaptive learning rate. Adam combines both RMSProp and Momentum and it moves faster in the dimension that has high gradient variance. Notice we added Encoder/Decoder logic on top of Bao et al model.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Experiments/Results/Discussion

We use three matrix to measure the model accuracy:

- HitRate: calculate how many positive are actually positive. For stock trading this is particular important because this could mean loss money versus winning less.
- Correlation: this is important as we want to be able to pick the top winner and the bottom one.
- Slope: This is important because we want the slope to be as near as one as possible. if it's near to zero, although we might have high nitrate, there will be high variable in our prediction.

Multi-factor model (MFM) we tested based on the same data set and time frame has the following value: slope=0.005, Corr=0.021, HitRate:55%. Notice MFM has a good Hit-rate percentage but the slope is very flat and showing high bias.

Notice how we differ from the Bao paper in how we measure the quality of the model and it's predicting power in 20 days.

Batch size has been tested from 64 to 512. The difference is minor and we stick to use 128 so that the run for epoch isn't too long and we still get good level of randomness.

Choice of model combination: we want to experiment different model combinations and here is the list of names and the model they refer to:

- sae + lstm: Stacked Autoencoder + 2xlstm
- wave + sae + lstm (Bao et al model): wavelet + Stacked Autoencoder + 2xlstm
- sae + lstm + enc : sae +2xlstm + encoding/decoding + 1x lstm
- wave + sae + lstm + enc: wave filtering + sae +2xlstm + enco/deco + 1x lstm

Each of them will end with 2 layers of Dense + BatchNorm + LeakyRelu + Dropout. The models have roughly about 300K parameters.

Parameters tuning: Beside the choice of the model combination, we also try to vary different hyper parameters:

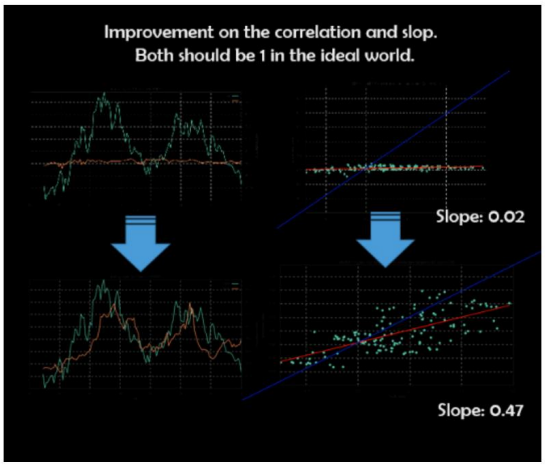
- Steps: how many steps we look back (20,40,60,80)
- Dropout: from 50% to 0%
- Regularisation: 0.01, 0.001, 0.0001 and 0.00001

Data set: Most of the experiment are conducted with 50 names (150K data set) but few experiments are conducted on 200 names (450K data set) to see the effect of big data. We would use more data but the process is just too slow with the AWS machines that was given to us and free credit got run out fairly quickly.

	mse	slope	corr	hitrate
MFM		**0.5%**	**2.1%**	**55.0%**
sae + lstm	0.82%	0.7%	4.3%	55.1%
sae + lstm + enc	0.90%	1.4%	3.0%	53.5%
Bao	0.83%	0.2%	1.8%	53.2%
Bao + enc	0.85%	0.4%	3.0%	52.0%
pca + sae + lstm	0.85%	-0.4%	0.7%	47.7%
Bao + enc + reg 0.01	0.83%	0.0%	n/a	56.9%
Bao + enc + reg 0.001	0.83%	0.0%	4.3%	49.8%
Bao + enc + reg 0.0001	1.07%	0.8%	2.5%	47.3%
Bao + enc + reg 0.00001	0.83%	0.1%	-1.0%	48.5%
Bao + enc + Keep=0.5	0.85%	1.0%	3.5%	52.0%
Bao + enc + Keep=0.6	0.83%	0.4%	2.4%	56.6%
Bao + enc + Keep=0.7	1.01%	1.0%	2.2%	50.8%
Bao + enc + Keep=0.8	1.11%	0.8%	3.1%	54.7%
Bao + enc + Keep=0.9	0.91%	0.8%	1.2%	49.7%
Bao + enc + Keep=1	0.89%	-0.3%	-0.4%	51.0%
Bao + enc + Steps=20	1.07%	1.8%	1.9%	48.9%
Bao + enc + Steps=40	0.90%	0.5%	1.2%	50.5%
Bao + enc + Steps=60	0.89%	1.2%	3.9%	56.1%
Bao + enc + Steps=80	0.85%	0.4%	3.0%	52.0%
Bao + enc (200 names)	1.11%	0.8%	3.1%	54.7%

Results analysis: The excel table on the left shows all the different combination we had. Unless mentioned, they are all based on 50 names. Firstly, the multi factor model is very hard to beat from this project but it's not conclusive that it's better than Deep learning. This will be further discussed in the next steps sections.

We noticed that regularisation is not doing anything but harming the performance. The slop is nearly zero and sometime we can't even get any correlation number. this wasn't too surprising given we haven't really overfit the model yet. The keep rate has some effects on the performance but it's not sure if it's purely random or 60% keep rate is the best range or not. further study must be conducted to verify. Number of steps does seem to have an effect on the results and 60 works days seems to be having highest positive impact with highest slope, correlation and hit ratio. On the right side the graph is one of the out example using step 60 and keep rate of 0.6. When we increase the number of stocks, we can see steady performance and good correlation but it's taking too long to have multiple runs.



Conclusion and next steps: Even though the result of this paper seems to be encouraging. we can't really conclude anything at this stage. because 1) most of the test were only based on 50 names and 2) we don't have enough test data to have a high confidence level. However, I believe we are on the right track and should perform further analysis in the following:

a. Have a bigger / faster server to test with higher number of stocks

b. Change the loss function to include different sign issue. if y, y_hat are (3,1) and (1,-1). but difference is 2 but the later one will cost us losing money where the former one will only cause us to earn less. we can also see this problem in our result.

Contribution: [1] github repository had a base LSTM Tensorflow [11] framework. I added SAE, wavelet, PCA, encoder/decoder and also rewrote the code w Keras.

References:

[1] https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html

[2] A deep learning framework for financial time series using stacked auto-encoders and long short term memory Wei Bao , Jun Yue, Yulei Rao

[3] quandl.com bundle data on China data set https://www.quandl.com/bundles/chinese-stocks-and-macroeconomics

[4] Fundamental factors https://www.businesstoday.in/moneytoday/investment/key-financial-ratios-analyze-company-stock-investment/story/209789.html

[5] Economic factors https://www.moneycrashers.com/leading-lagging-economic-indicators/

[6] Technical signal https://github.com/bukosabino/ta

[7] Forecasting East Asian Indices Futures via a Novel Hybrid of Wavelet-PCA Denoising and Artificial Neural Network Models http://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0156338&type=printable

[8] Mikko Ranta Wavelet Multiresolution Analysis of Financial Time Series https://www.univaasa.fi/materiaali/pdf/isbn_978-952-476-303-5.pdf

[9] LSTM explained http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[10] Deep, Narrow Sigmoid Belief Networks Are Universal Approximators https://www.mitpressjournals.org/doi/10.1162/neco.2008.12-07-661

[11] Tensorflow https://www.tensorflow.org/about/bib