Quantitative Trading Strategies using Deep Learning: Pairs Trading

Simerjot Kaur
sk3391@stanford.edu

Abstract

Pairs trading is a commonly used quantitative trading strategy used in hedge funds and investment banks. A key step in pair trading strategy is the prediction of spread amongst a pair of selected financial instruments. However, given the complex dynamics of the markets, forecasting spread typically requires establishing predictive models based on deep financial knowledge. Current state of the art methods use statistical methods to perform the spread prediction. In this paper we develop an LSTM based RNN model for predicting the spread and use this predicted spread in an end-to-end trading pipeline. In particular, given the value of spread up to time instant 't', we predict the value of spread 1-time step (S_{t+1}), 5-time step (S_{t+5}), and 15-time steps (S_{t+15}) in future and compare the performance of our deep learning model with current statistical models such as ARIMA, VAR, and Kalman Filter. Using the proposed deep learning model we are able to significantly reduce the mean squared error (MSE) between the predicted and true future time series as compared to the MSE achieved using statistical techniques. Further, we achieve near optimal performance in predicting one time step ahead, and a substantially lower MSE when predicting 15-time steps in the future.

1. Introduction

Quantitative trading is making trade decisions based on large amounts of data. Over the last decade, there has been a parabolic rise in quantitative trading, owing to the advent of growth in computing power as well as advances in statistical methods.

One commonly used quantitative trading strategy in hedge funds and investment banks is Pairs trading. It exploits financial markets that are out of equilibrium. The idea behind pairs trading is as follows. First, a pair of assets is selected, that are known to historically move together and have some sort of long-run relationship. Subsequently, under the assumption that the spread, defined as the difference in price between the paired assets, is mean-reverting the deviations from the mean are exploited to make trading decisions.

To implement the above strategy effectively two keys aspects need to be optimized :

1. Identify the best pairs of financial instruments
2. **Forecast the spread** between the paired assets based on historical data, and use the forecasted spread to make trading decisions.

However, due to the complex dynamics of the markets, forecasting spread between the paired assets requires establishing predictive model with professional financial knowledge, and adopting various statistical methods, which makes it a non-trivial exercise. ***In this project, we develop a framework for pairs trading strategy, by forecasting the spread between paired assets using LSTM based RNN and compare it with the current state of the art statistical techniques such as ARIMA, VAR and Kalman Filter.*** Formally, given a time series of spread (S_1, … , S_t), we predict the value of spread 1-time step (S_{t+1}), 5-time step (S_{t+5}), and 15-time steps (S_{t+15}) in the future.

2. Related Work

Pairs trading is one of the most commonly studied algorithmic trading strategies in literature [1]. In [1] the spread prediction problem is formulated as a simple distance measurement problem, but this was not used much in practice owing to simplistic assumptions made in the model. Subsequently, in [2], [3] authors proposed a kalman filtering based approach to predict spread which is currently one of the most commonly used techniques in pair trading. Recently, there has been an interest in applying deep learning methods for stock price prediction problem such as in [4]. In [4] authors use wavelet de-noising and LSTM network to predict stock price. Since spread prediction for pair trading for ETFs is still an unsolved problem, in this paper we explore the possibility of using wavelet de-noising with stacked LSTM augmented with additional features as a means to predict spread for pair trading strategy.

3. Dataset and Features

The proposed pairs trading strategy is implemented on historical data of Exchange traded funds (ETFs). Given the increasing popularity and easier accessibility of ETFs, it is interesting to investigate the performance of pairs trading on these instruments rather than vanilla stocks. To create the dataset we extracted the daily 'close', 'open', 'high', 'low' price and 'volume traded' for various exchange traded funds (ETFs) from 10.01.2008 to 10.01.2018 from Yahoo! Finance Live using the Auquan Toolbox[5]. After extracting the raw data from Yahoo Finance, we performed few data-processing steps, described as follows:

3.1 Data Cleaning

We modified the dataset to ensure that we only use those ETFs which have been active throughout the ten year period. Any dead ETFs were discarded from the dataset. We also adjusted the prices to take into account dividends and splits.

3.2 Feature Extraction

In order to facilitate learning we calculated and extracted momentum, volatility, trend, volume and other technical indicators (see table below) from raw time series data which were then used as additional inputs for our deep learning model. The technical indicators were extracted in python using **TA** [6] library.

Technical Indicator	Type of Indicator	Description
Relative Strength Index	Momentum (t=14)	$RSI = 100 - \frac{100}{1 + RS}$ $whr\ RS = \frac{Average\ of\ t-day's\ up\ closes}{Average\ of\ t-day's\ down\ closes}$
Money Flow Index	Momentum (t=14)	$MFI = 100 - \frac{100}{MFR}$ $whr\ MFR = \frac{t-period\ Positive\ Money\ Flow}{t-period\ Negative\ Money\ Flow}$
Accumulation / Distribution Index	Volume	$ADL = ADL_{prev} + (MFM * Volume)$ $whr\ MFM = \frac{[(Close-Low)-(High-Close)]}{High-Low}$
Volume – Price Trend	Volume	$VPT = VPT_{prev} + (Volume * \frac{Close_{today} - Close_{prev}}{Close_{prev}})$
Average True Range	Volatility (n=14)	$ATR_t = \frac{ATR_{t-1}*(n-1)+TR_t}{n}$ $where\ TR_t = \max(High - Low, abs(High - Close_{prev}), abs(Low - Close_{prev}))$
Bollinger Bands	Volatility (n=20)	$BB = n-day\ Simple\ Moving\ Average$
Average Directional Movement Index	Trend (n=14)	$ADMI = Average\ of + ve\ Directional\ Movement\ and - ve\ Directional\ Movement$
Exponential Moving Average	Trend (n=14)	$EMA = [Close - EMA_{prev}] * \frac{2}{n+1} + EMA_{prev}$
Moving Average Convergence Divergence	Trend (n_fast = 14, n_slow = 30)	$MACD = EMA_{n\_fast} - EMA_{n\_slow}$
Daily Log Return	Other	$Log\ Return = ln\frac{Close_i}{Close_{i-1}}$

3.3 Wavelet Denoising

Wavelet transforms are commonly used for filtering financial time series given their ability to handle non-stationary data. Unlike Fourier transform which only give information about the frequency content of a signal, wavelet transform allow us to get information about frequency content of a signal and also when that content was present. This makes them uniquely suited to handle non-stationary data.

For de-noising the input time series, we use the haar wavelet basis owing to its good performance in filtering and producing stock price data. Formally, we approx. the signal x(t) as a combination of orthogonal wavelet series as follows:

$$x(t) = S_j(t) + D_j(t) + D_{j-1}(t) + \cdots + D_1(t)$$

where $S_j(t)$ is the coarsest approximation of the input time series x(t). The multi-resolution decomposition of x(t) is the sequence of $\{S_j(t), D_j(t), D_{j-1}(t), \ldots, D_1(t)\}$. To eliminate noise, we zero out any coefficients which are extremely small, and below a particular threshold. The wavelet transform was implemented in python using **PyWavelets** [7] library.

4. System Pipeline and Methods

Figure 1 below depicts our end-to-end system pipeline. As a first step, we identify the pair of ETFs on which we perform pairs trading. Subsequently, we forecast the spread amongst the selected ETF pairs. The forecasted spread is then used to make a decision if we should buy or the sell the particular ETF. Each of the above steps are described in more details in the sub-sections below.

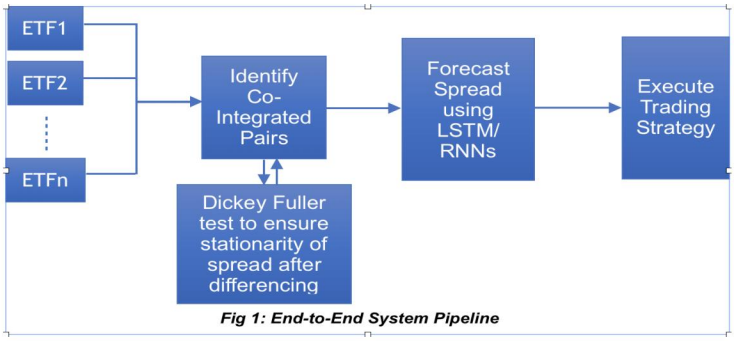

Fig 1: End-to-End System Pipeline

4.1 Selecting Co-integrated Pair of ETFs

The selection of a pair of ETFs from amongst the entire list of possible ETFs is done using the co-integration method [8] by testing which pairs, if any, are co-integrated with each other. The co-integrated ETFs are expected to have same long-run trend, and a mean-reverting behavior which is important for pairs trading methodology. To ensure that the co-integration value is statistically significant, we also evaluate the p-value of co-integration score and make sure the selected pairs have an extremely small p-value.

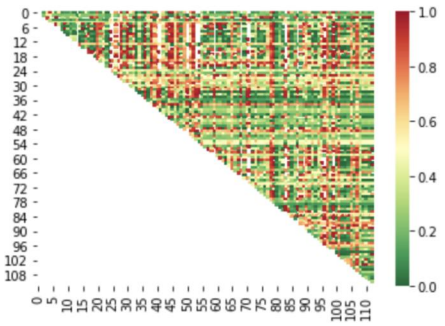

Fig 2 : Heatmap showing p-value among all pairs of 108 chosen ETFs.

ETF1	ETF2	P-Value
SPY	IVV	0
IWV	VTI	1.66E-21
IVE	IJJ	1.74E-05
PRF	IVE	5.64E-05
VYM	DVY	6.76E-05
VFH	KBE	9.76E-05
VT	ACWI	1.42E-04
DON	DVY	1.58E-04
ITOT	VV	1.68E-04
EWT	VPL	2.57E-04

Fig 3 : Ten ETFs with lowest p-value

Figure 2 above shows the heat-map of p-values generated to identify the best pairs of ETFs. Figure 3 lists the top 10 ETFs with lowest p-value. Based on the heat map, we selected the two most co-integrated ETF pairs, 'IVE' and 'IJJ' with a p-value of 1.74e-05 and 'VYM' and 'DVY' with a p-value of 6.76e-05 for pairs trading. The selected pairs of ETFs are a good choice for pairs trading, as they contain similar industry based stocks, and hence can be expected to be co-integrated. It may be noted that pairs 'SPY' and 'IVV' as well as 'IWV' and 'VTI' were discarded although they have a p-values very close to zero since these pairs of ETFs contain the same underlying stocks and are different only because of the transaction costs, fees, etc.

It maybe noted here that since we are forecasting the spread, defined as the difference in price between the paired assets, we naturally remove the first order non-stationarity in the data via differencing method. Further, we used the Dickey-Fuller test to quantify the non-stationarity in our time series.

4.2 Spread Forecasting Methods

Since the spread between the selected co-integrated pairs of ETFs is essentially a time series, we formulate the problem of forecasting the spread as a time series prediction problem. Formally, given the value of spread up-to time instant 't', we want to predict the value of spread at 1-time step (S_{t+1}), 5-time step (S_{t+5}), and 15-time steps (S_{t+15}) in the future. In this work we compare and contrast between spread forecasted using current state-of-the art statistical methods v/s using deep learning techniques, in particular using LSTM based RNNs.

4.2.1 Spread Forecasting using Statistical Methods: Baseline Models

To handle the increasing variety and complexity of financial forecasting problems, many statistical techniques have been developed in recent years. In particular, some of the commonly used statistical techniques used for predicting spread in pairs trading strategy are Kalman Filtering, VAR and ARIMA models. The section below describes the various statistical techniques we used for predicting the spread. These act as our baseline models, and we compare the performance of our deep learning models against these statistical models.

- **Kalman Filter:** Kalman filters are very commonly used for predicting time series in the presence of noisy observations. Use of Kalman filters for predicting spread amongst co-integrated stocks has been very well studied in literature [2] and is one of the most commonly used methods to predict spread. Kalman filter was implemented in python using **pyKalman** [9] library. The transition and observation matrices were learned using Expectation-Maximization (EM) algorithm.
- **ARIMA:** ARIMA (Auto-Regressive Integrated Moving Average) models are, in theory, the most general class of models for forecasting a time series. Lags of the stationarized series in the forecasting equation are called "autoregressive" terms, lags of the forecast errors are called "moving average" terms, and a time series which needs to be differenced to be made stationary is said to be an "integrated" version of a stationary series. ARIMA was implemented in python using **statsmodels** [10] library.
- **VAR:** Vector Autoregression (VAR) is a stochastic process model used to capture the linear interdependencies among multiple time series. VAR models generalize the univariate autoregressive model (AR model) by allowing for more than one evolving variable. All variables in a VAR enter the model in the same way: each variable has an equation explaining its evolution based on its own lagged values, the lagged values of the other model variables, and an error term. VAR was implemented in python using **statsmodels** [10] library.
- **Facebook Prophet:** Facebook Prophet is a modular regression model with interpretable parameters that can be intuitively adjusted by analysts with domain knowledge about the time series. It is a decomposable time series model with three main model components: trend, seasonality, and holidays. The forecasting problem has been framed as a curve-fitting exercise, which is inherently different from time series models that explicitly account for the temporal dependence structure in the data. Facebook Prophet was implemented in python using **fbprophet** [11] library.

4.2.2 Spread Forecasting using LSTM based RNN Method: Deep Learning Model

To forecast the spread using deep learning, we used stacked LSTM based recurrent neural network. Long Short Term Memory networks, usually just called "LSTMs", are a special kind of RNN, capable of learning long term

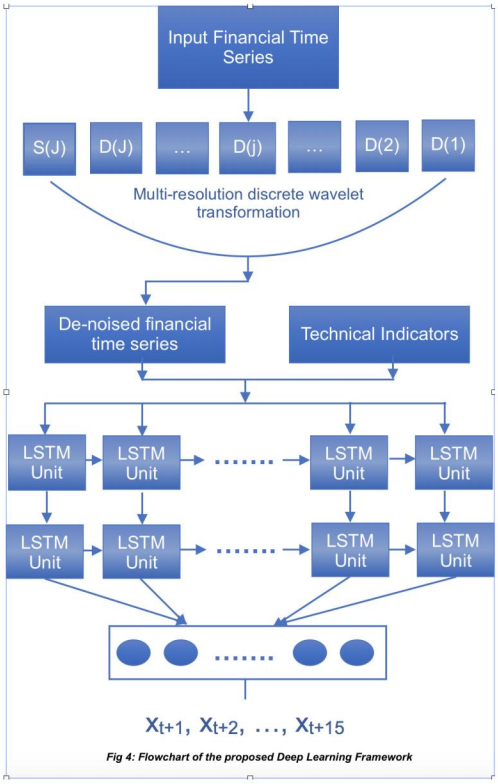

Fig 4: Flowchart of the proposed Deep Learning Framework

dependencies. The architecture is composed of a memory cell, an input gate, an output gate and a forget gate. An LSTM cell takes an input and stores it for some period of time. The activation function of the LSTM gates is often the logistic function. Intuitively, the input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

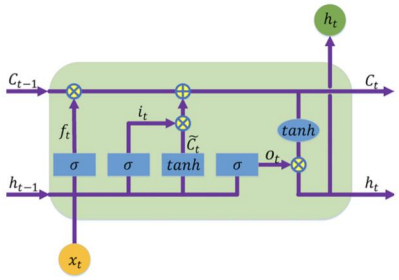

Fig 4 on the left shows the flowchart of the proposed deep learning framework to predict spread. As can be observed, the financial time series is first de-noised using multi-resolution discrete wavelet transformation (as explained in 3.3) and is then fed into stacked LSTM (2 layered) model followed by a fully connected layer along with the extracted technical indicators (as explained in 3.2). The model then predicts the value of spread 1-time step (S_{t+1}), 5-time step (S_{t+5}), or 15-time steps (S_{t+15}) in the future.The LSTM based stacked RNN was implemented using Keras[12].

4.3 Executing Trading Strategy

We used the spread information to create a simple trading strategy to determine when to buy or sell the ETF. In particular, we look at the value of the normalized spread and when the spread significantly deviates from the mean of the spread, we take a short position in the overvalued asset and a long position in the undervalued asset. As soon as the spread converges back to its mean, we unwind both the positions and estimate the profit. Fig 5 below shows an example output from our trading strategy depicting when to buy, and when to sell.

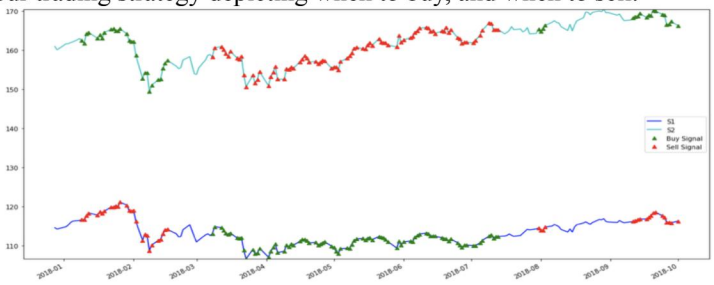

Fig 5: Sample output from our trading strategy

5. Results and Discussions

In this project, we split the 10 years data into 90:5:5 ratio to form the training, dev and test dataset respectively. Given the large historical data as well as large number of hyperparameters, the training and testing was done on AWS machines. The error metric used for training deep learning model and statistical models is **Mean Squared Error(MSE)** between predicted and true time series. Fig 6 below shows the MSE (on dev set) as a function of various model hyper-parameters such as number of neurons in LSTM layers, number of epochs, batch size and dropout. Based on the results, we chose 256 neurons for both the LSTM layers, dropout rate of 0.2 and 200 epochs with a batch size of 20.

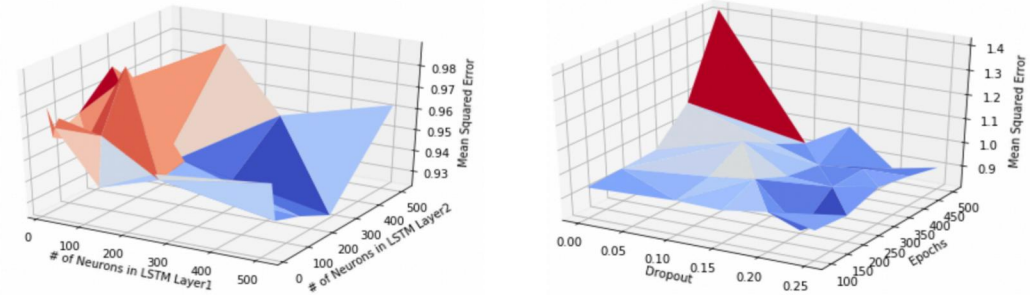

Fig 6. Hyperparameter Tuning with hyperparameters as # of neurons in both layers, # of epochs, batch_size and probability of dropout

The deep learning model was then used to predict the value of spread 1-time step (S_{t+1}), 5-time step (S_{t+5}), and 15-time steps (S_{t+15}) in future. Fig 7(a) below shows the actual vs predicted spread value using Kalman filter, and our deep learning approach. As is clearly evident, our deep learning model significantly outperforms the state-of-art Kalman filtering based approach. Also, it may be noted that our model performs significantly better than the statistical models as well as deep learning model used in [4] and achieves near optimal performance in predicting spread S_{t+1}. Further, it may be noted that unlike existing literature, we are able to extend and achieve reasonably good results on long term spread prediction, up to 15-time steps ahead, fig 7(b). Clearly, for long-term prediction while we are not able to achieve near optimal value, we are able to capture trend, volatility and momentum of actual spread very well. Also, performance of our deep learning model far succeeds state of the art statistical techniques.

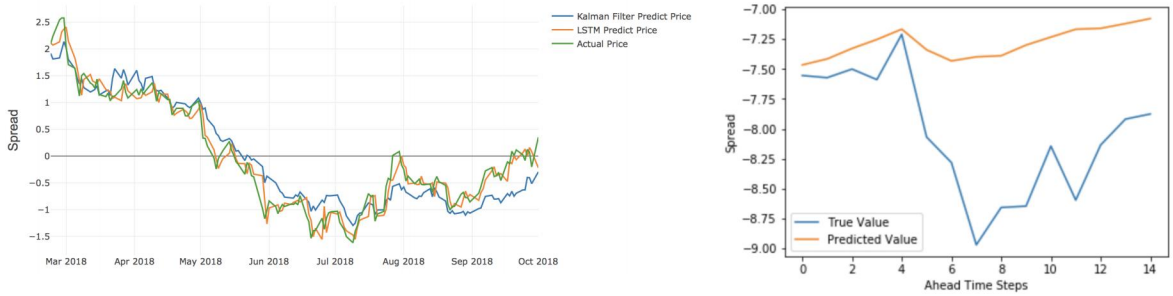

Fig 7. Comparison between (a) S_{t+1} true spread and predicted spread using Kalman Filter and LSTM on left (b) S_{t+15} true and predicted spread using LSTM on right

Further, Fig 8 below shows a numerical comparison amongst the various models used for predicting spread. The table of the left compares the performance of various models on ETF pair 'IVE' and 'IJJ', while the table on the right shows the performance for ETF pair 'VYM' and 'DVY'. As is clearly evident from MSE, deep learning based methods clearly outperform statistical methods in predicting spread value. Further, adding more technical indicators and filtering the input time series using wavelet transform further helps improve performance significantly.

Pair = 'IVE' & 'IVJ'	1-Time Step Ahead Prediction		5-Time Step Ahead Prediction		15-Time Step Ahead Prediction	
*TI = Technical Indicators	Training Error (MSE)	Test Error (MSE)	Training Error (MSE)	Test Error (MSE)	Training Error (MSE)	Test Error (MSE)
Statistical Methods						
Kalman Filter	N/A	1.9347	N/A	N/A	N/A	N/A
VAR	N/A	0.5792	N/A	0.8156	N/A	1.1932
ARIMA	N/A	0.5797	N/A	0.8182	N/A	1.1936
Facebook Prophet	N/A	2.1973	N/A	2.2625	N/A	2.3586
Deep Learning Techniques						
LSTM	0.1599	0.4342	0.2586	0.7112	0.4293	1.009
LSTM + TI	0.1499	0.4183	0.2213	0.6682	0.2451	0.9439
Stacked LSTM + TI	0.1449	0.4055	0.2197	0.6552	0.2439	0.9342
Stacked LSTM + Wavelet Denoising + TI	0.1394	0.4024	0.2174	0.6436	0.2427	0.8945
Stacked LSTM + Wavelet Denoising + TI + Optimized Regularization	0.1392	0.4002	0.2137	0.5997	0.2305	0.8091

Pair = 'VYM' & 'DVY'	1-Time Step Ahead Prediction		5-Time Step Ahead Prediction		15-Time Step Ahead Prediction	
*TI = Technical Indicators	Training Error (MSE)	Test Error (MSE)	Training Error (MSE)	Test Error (MSE)	Training Error (MSE)	Test Error (MSE)
Statistical Methods						
Kalman Filter	N/A	1.6048	N/A	N/A	N/A	N/A
VAR	N/A	0.3762	N/A	0.5597	N/A	0.7803
ARIMA	N/A	0.3748	N/A	0.5569	N/A	0.7705
Facebook Prophet	N/A	1.0099	N/A	1.0209	N/A	1.0277
Deep Learning Techniques						
LSTM	0.1116	0.2917	0.1733	0.4961	0.2694	0.7618
LSTM + TI	0.0998	0.2714	0.1485	0.4874	0.2128	0.7491
Stacked LSTM + TI	0.0967	0.2688	0.1482	0.4738	0.1701	0.7364
Stacked LSTM + Wavelet Denoising + TI	0.0943	0.2608	0.1473	0.4391	0.1619	0.7164
Stacked LSTM + Wavelet Denoising + TI + Optimized Regularization	0.0923	0.2408	0.1453	0.4118	0.1581	0.7084

Fig 8. Comparison between baseline models and deep learning models for 1-time step (S_{t+1}), 5-time step (S_{t+5}), and 15-time steps (S_{t+15}) in future for pair 'IVE' and 'IJJ' on the left and pair 'VYM' and 'DVY' on the right

Fig 9 below also compares that the profits achieved for various spread prediction methods. Clearly, deep learning methods are able to achieve profits which are very close to the maximum possible profit that could be achieved using the implemented pairs trading strategy.

Pair = 'IVE' & 'IVJ'	Ground Truth	Kalman Filter	VAR	ARIMA	Facebook Prophet	Deep Learning Method
Profit Made	$91,965	$29,960	$64,448	$64,580	$49,317	$75,689

Pair = 'VYM' & 'DVY'	Ground Truth	Kalman Filter	VAR	ARIMA	Facebook Prophet	Deep Learning Method
Profit Made	$6,299	$3,158	$4,456	$3,941	$3,345	$5,384

Fig 9. Comparison of profits between deep learning methods and the baseline statistical techniques for pair 'IVE' and 'IJJ' on the left and pair 'VYM' and 'DVY' on the right

6. Conclusion and Future Work

In this paper, we developed a framework for pairs trading strategy, by forecasting spread between paired assets using LSTM based RNN and compared it with current state of the art statistical techniques. We were able to predict the value of spread 1-time step (S_{t+1}), 5-time step (S_{t+5}), and 15-time steps (S_{t+15}) in future. We observed that the prediction of S_{t+1} is performing well and achieved near optimal spread value. Additionally, we found that while predicting n-time steps ahead, although it does not always predict the near optimal spread value, but it does capture the trend, volatility and momentum of actual spread very well. We tried various versions of deep learning methods such as vanilla LSTM, Stacked LSTM, Stacked LSTM with additional features, wavelet denoising and optimized regularization. It was observed that MSE obtained as well as profits generated from pairs trading strategy using deep learning methods were significantly better as compared to baseline statistical models ARIMA, VAR, Kalman Filter and Facebook Prophet. It was also observed that using additional features, like technical indicators, as well as performing Wavelet de-noising that eliminates market data noise, improves performance of our model significantly.

Finally, our current work can be extended in various ways. The model can be further improved by providing it additional information from other sources, such as current news, twitter feeds etc. via NLP to help improve long term predictive power of model. Also we could explore deep learning methods for identifying the co-integrated pairs, as an alternative to current statistical method of stock selection.

7. Code

The code can be downloaded from:
https://github.com/sk3391/CS230

8. References

[1] Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (1999). Pairs trading: Performance of a relative-value arbitrage rule. Working paper, Yale School of Management's International Center for Finance.
[2] Elliott, R. J., van der Hoek, John, & Malcolm, W. P. (2005). Pairs trading. Quantitative Finance, 5(3), 271–276.
[3]Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (2006). Pairs trading: Performance of a relative-value arbitrage rule. Review of Financial Studies, 19(3), 797–827.
[4] Wei Bao, Jun Yue2, Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long short term memory.
[5] Auquan Toolbox: https://bitbucket.org/auquan/auquantoolbox/wiki/Home
[6] Technical Analysis Library: https://technical-analysis-library-in-python.readthedocs.io/en/latest/
[7] PyWavelets Library: https://pywavelets.readthedocs.io/en/latest/
[8] Vidyamurthy, G. (2004). Pairs Trading: Quantitative Methods and Analysis. John Wiley & Sons.
[9] PyKalman Library: https://pykalman.github.io/
[10] StatsModels Library: https://www.statsmodels.org/stable/index.html
[11] Facebook Prophet Library: https://facebook.github.io/prophet/docs/quick_start.html#python-api
[12] Keras Library: https://keras.io/