Robert Chang & Michel Kamel

Prof. Peter Henstock

CSCI E-82

Fall 2017

Machine Learning for Stock Trading Predictions

CNN, DTW & Kalman Filter

*Abstract*

Predicting future stock return was modelled using traditional econometric models such as ARIMA, ARCH, etc. This paper uses the CNN architecture which is designed in a way to use images as inputs and predict any output. Daily stock movement on S&P data (each minute) is used to predict future return for different time horizon including quarter of the day, 1 hour, 30 minutes and 5 minutes. We also use machine learning techniques to improve upon strategies for stock-pair trading. Pairs trading is an important research area of computational finance that typically relies on time series data of stock price for investment. Using the S&P dataset, we create a novel strategy for stock pair selection, and determining optimal entry and exit points for pairs trades, using dynamic time warping (DTW), K-means clustering, and the Kalman filter.

**CNN for Predicting Stock Returns**

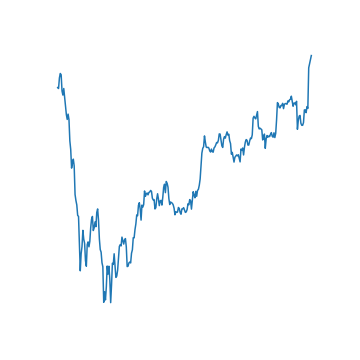
Predicting stock returns is a challenging problem requiring a lot of inputs from different sources like historical data, news, structure of the company issuing stocks, economy, etc. All these together affect the supply and demand on a stock in the future so should a model capture them to predict accurately. Currently used models for predictions are statistically based that try to fit a function for historical movements and assume that this function will hold in the future. The first part of the paper will approach the problem in the same logic but instead of looking at historical movement numerically, we will plot the chart of price and then do the image-transformation using a convolution and pooling to create features that will be the input to a fully connected neural network to predict future return. The difference with standard CNNs is that no colors difference is there across the picture, only one color (will remove axis) and created feature will differ from a region of the image to another by the fact if there is colored pixel in that region or not.

One important assumption to build traditional models is the autocorrelation between instants of tracking the stock, i.e. each minutes and the minute before if the data is on minute basis. Here we don’t look at it and let the feature creation process and network weights find it in another way. Currently we cannot interpret weights of that but will track the result of the model.

This prediction will help us make money in the future when CNN says that the stock price will goes up so we can buy it and sell it when CNN says that the stock will go down.

**CNN Data Preparation**

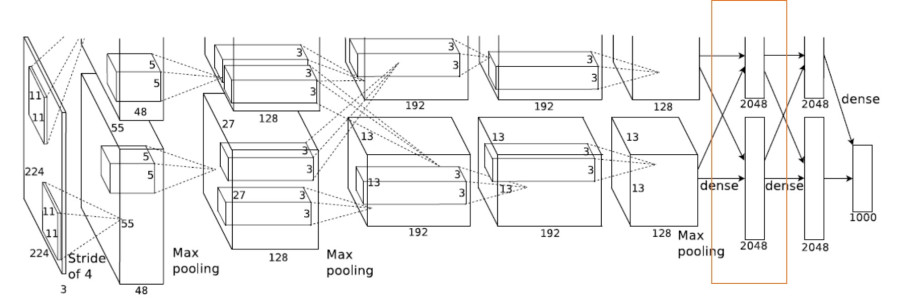
High frequency data would be better in our case to see all fluctuation in one image while it is not available publically easily. We found a minutes (each 60 seconds) data on a website mentioned in our references which contains 41266 rows and 500 stocks as well as the S&P 500. The data is ranging from April to August 2017. A transformation of the data is needed whereby we have chosen to create one image from each day data representing what happened during the first part of the day (almost 75% of the day, i.e. around 290 minutes) and then calculate the return for 5 minutes, 30 minutes, and 100 minutes (end of day) which will be the dependent variable corresponding to each image in the inputs. Below is an example of an image.



**CNN Architecture**

Two types of architectures have been used, LeNet and AlexNet on different forecasting time horizon (5 minutes, 30,…) using a Batch mode of 256 rows at a time. Since colors is not important in this exercise, we used only one level of channels.

The loss function used is the standard L2 for the difference between actual and predicted returns. This is aligned with what we need from the network which is accurate predictions of the level of return so the lower the above loss the better the model.



A batch mode with hundreds of epochs used to estimate weights of the network for two architectures: Le-Net 5 and AlexNet. For each architecture, several time horizon are considered like 30 minutes and 3 hours. Due to resources constraints we were not able to use more than around 4,000 images but will try in the future with more powerful PC or on the cloud.

**DTW & Kalman filter for Pairs Trading**

Pairs trading is an important research area of computational finance that typically relies on time series data of stock price for investment, in which stocks are bought and sold in pairs for arbitrage opportunities. Some stocks move in tandem because the same market events affect their prices. However, idiosyncratic noise might make them temporarily deviate from the usual pattern and a trader could take advantage of this apparent deviation with the expectation that the stocks will eventually return to their long-term relationship. Two stocks with such a relationship form a “pair”.

A pairs trading strategy consists of two components:

* Choose a pair which will give you good statistical arbitrage opportunities over time.
* Choose the entry/exit points.

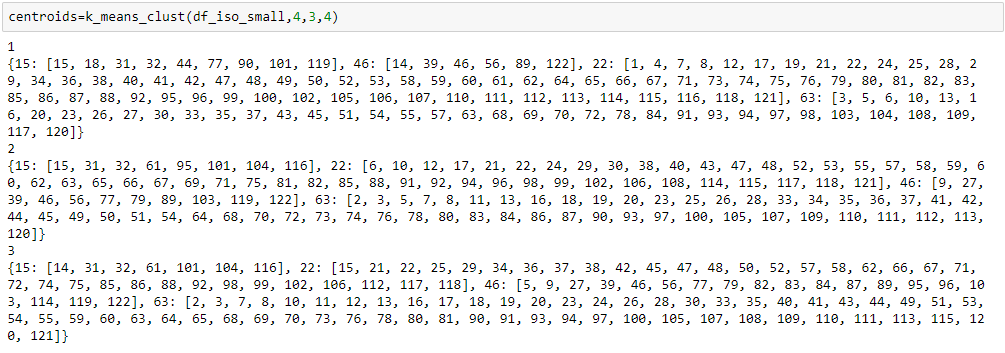
**Choosing Good Stock Pairs**

For the problem of choosing good pairs, there are a number of academic papers that discuss ways to select stock pairs to trade. This is not a trivial problem, due to the sheer number of pairs available for selection, for example, pair trading in the S&P 500 would provide 125,000 (500^2 / 2) pairs to select from. Most of the current research falls into one of these categories.

* Distance approach - Choose pairs based on their squared distance of the normalized price series. The smaller the distance, the greater the co-movement of prices.
* Cointegration approach - Perform a cointegration test between every possible pair of stocks, pairs are ranked according to the t-statistic obtained, and the ones with the lowest t-statistic are picked.
* Stock Clustering - Cluster the stocks according to pre-defined risk factors. Usually this is combined with a dimensionality reduction technique such as PCA. TSfresh can also be used to extract characteristics from time series.

We use a distance-based approach to select stock pairs, and use a Dynamic Time Warping (DTW) distance measure. DTW will stretch or compress segments of temporal data, to determine an optimal match for any pair of time series. In this way, time series exhibiting similar patterns occurring at different time periods, are considered as being similar. Euclidean distance and its variants present several drawbacks, that make them an inappropriate choice for selecting stock pairs. DTW has several advantages over Euclidean distance, it doesn't need to compare only time series of the same length, can appropriately handle outliers or noise, and is not overly sensitive to shifting, uniform amplitude scaling, and time warping. Although DTW is a suitable choice calculating the average of a collection of time series (which is required for clustering) is challenging. Once we have a DTW distance measure, we use K-Means, where the cluster centers are calculated through averaging the data, to find similar stocks that will be suitable for pairs trading.

Cluster results for 4 clusters, 3 iterations, and a Keogh LB window of 4 are shown below.



**Choosing Entry/Exit Points**

To determine our entry and exit points for trades we will use state space models, the primary benefit of which is that their parameters can adapt over time. The general premise of a state space model is that we have a set of states that evolve in time, specifically in our case, the hedge ratio between two cointegrated pairs of equities. However, our observations of these states contain statistical noise, and we are unable to ever directly observe the "true" states.

In its simplest form, we can model the relationship between a pair of securities in the following way:

*beta(t) = beta(t-1) + w* beta(t), the unobserved state variable, that follows a random walk

*Y(t) = beta(t)X(t) + v* The observed processes of stock prices Y(t) and X(t)

where:

*w ~ N(0, Q)* meaning w is gaussian noise with zero mean and variance Q

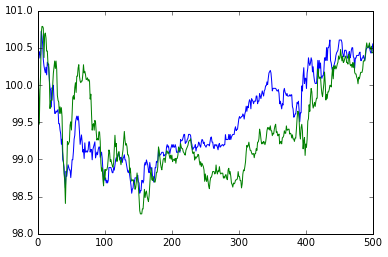
*v ~ N(0, R)* meaning v is gaussian noise with variance R

This is a standard stock pairs relationship Y = beta \* X + v, where the typical approach is to estimate beta using least squares regression. In this standard framework, beta is static, or periodically updated at regular intervals. In the Kalman framework, beta is itself a random process that evolves continuously over time, as a random walk. Because it is random and contaminated by noise we cannot observe beta directly, but must infer its (changing) value from the observable stock prices X and Y.

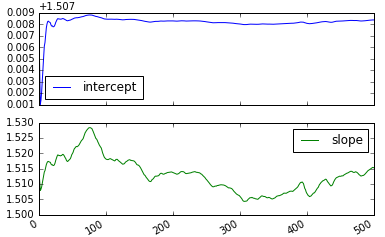
The goal of the state space model is to infer information about the states, given the observations, as new information arrives. We will use the Kalman Filter to carry out this procedure. The Kalman Filter is ubiquitous in engineering control problems, including guidance & navigation, spacecraft trajectory analysis and manufacturing, but it is also widely used in quantitative finance.

Beta, being a random process, obviously contains some noise: but the hope is that it is less noisy than the price process. The idea is that the relationship between two stocks is more stable – less volatile – than the stock processes themselves. If the variance in the beta process is low relative to the price process, we can determine beta quite accurately over time and so obtain accurate estimates of the true price Y(t), based on X(t). Then, if we observe a big enough departure in the quoted price Y(t) from the true price at time t, we have a potential trade. The success of such a strategy depends on the accuracy of our estimates of beta, which depends on the noisiness of the beta process, i.e. its variance, Q. If the beta process is very noisy, i.e. if Q is large, our "departures" are going to be too noisy to be useful as the basis for a reversion strategy.

**An Example**



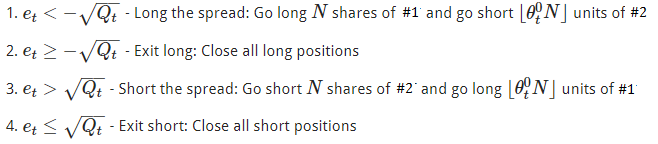
We can see from our clusters that stock #32 (NASDAQ: DISCA) and stock #104 (NASDAQ: STX) are clustered together. We will form a pair with these stocks to test the pairs trading strategy. We use the S&P data from April 2017 to August 2017. A plot of their prices (normalized to 100) for the first 500 minutes of trading is shown above.



We then use the Kalman Filter to plot the time-varying slope and intercept for the first 500 minutes of data, shown above. As mentioned previously, the slope and the intercept need to be constantly adjusted according to our model. Clearly the time-varying slope changes dramatically over the first 500 minutes of trading. It is not difficult to see that utilizing a fixed hedge ratio in a pairs trading strategy would be far too rigid. In addition, the estimate of the slope is relatively noisy. When we come to develop a trading strategy it will be necessary to optimize this parameter delta across baskets of pairs.

Now that we’ve “trained” our Kalman Filter on the first 500 minutes of data, we want to now execute our trading strategy. The synthetic "spread" between DISCA and STX is the time series that we are interested in longing or shorting. The Kalman Filter is used to dynamically track the hedging ratio between the two to keep the spread stationary (and hence mean reverting). To create the trading rules, it is necessary to determine when the spread has moved too far from its expected value. We can create a rule that uses the standard deviation of the spread as a boundary. We can go "long the spread" if the forecast error drops below the negative standard deviation of the spread. Respectively we can go "short the spread" if the forecast error exceeds the positive standard deviation of the spread. The exit rules are simply the opposite of the entry rules.

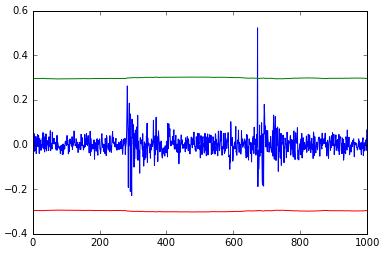
The trading rules are as follows:



The implementation of the strategy involves the following steps:

1. Receive daily market prices for both DISCA and STX
2. Use the recursive "online" Kalman filter to estimate the price of DISCA today based on yesterday’s observations of STX
3. Take the difference between the Kalman estimate of DISCA and the actual value, often called the forecast error or residual error, which is a measure of how much the spread of DISCA and STX moves away from its expected value
4. Long the spread when the movement is negatively far from the expected value and correspondingly short the spread when the movement is positively far from the expected value
5. Exit the long and short positions when the series reverts to its expected value

We use the first 500 minutes of data to “train” the Kalman Filter and the next 1000 minutes of data to execute the trading strategy. As we receive new prices for DISCA and STX we can recursively estimate the price of DISCA today based on yesterday’s observation of STX. We then calculate our forecast error. When the forecast error exceeds the expected value by a standard deviation we execute our trading rules accordingly.



The plot above shows our forecast error (blue) and our positive and negative standard deviation “boundaries” (green and red). We can see that around minute 700, the spread deviates from expectation by greater than a standard deviation and we should execute our trading strategy. In the future, I plan to use the QS Trader package in Python to fully back-test the strategy. Back-testing will have to account for trading frictions – the costs incurred in entering, adjusting and exiting positions across multiple symbols in the portfolio. In practice, trading frictions eliminate a large portion of the “theoretical” profit.

**References**

* <https://medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877>
* <http://cs231n.stanford.edu/reports/2015/pdfs/ashwin_final_paper.pdf>
* <http://alexminnaar.com/time-series-classification-and-clustering-with-python.html>
* <https://www.quantstart.com/articles/State-Space-Models-and-the-Kalman-Filter>
* <http://jonathankinlay.com/2015/02/statistical-arbitrage-using-kalman-filter/>
* Dynamic Time Warping Averaging of Time Series allows Faster and more Accurate Classification

<http://www.cs.ucr.edu/~eamonn/ICDM_2014_DTW_average.pdf>

* Machine Learning in Pairs Trading Strategies

<http://cs229.stanford.edu/proj2012/ChenRenLuMachineLearningInPairsTradingStrategies.pdf>