Using Machine Learning Models to Predict S&P500 Price Level and Spread Direction

Alex Fuster (*akfuster@stanford.edu*), Zhichao Zou (*zzou@stanford.edu*)

1 Introduction

Pairs Trading is an investment strategy applied by many buy-side financial institutions. In pairs trading, instead of making investment decision based on future stock price level, it is based on prediction of the spread between two stocks in the future. Consider two stocks moving in the similar trend, following the same market dynamic and trading at a spread that is mean-reverting. If the spread widens between the two stocks, then we can short the overpriced stock and buy the underpriced stock. Later on as the spread between the stocks mean-reverts to the equilibrium level, the profit will be realized. Traditionally, the methodology applied in pairs trading typically lies in the linear regression, Time Series Analysis such as ARIMA, SARIMA as well as GARCH model what captures the stochasticity of the volatility. These classical methods have proved to be quite effective in the old regime of the financial world. However, as the world of finance shifted into a new regime, particularly after the dramatic improvement in the computing speed, faster diffusion of the news and more transparent market, the quantitative trading industry has started employing more innovative strategies, such as Machine Learning, Deep Learning and Reinforcement Learning algorithms. In this project, we seek to explore how to best apply learning algorithms in the pairs trading by implementing various machine learning models and compared against each other.

We are inspired by the paper from (van der Have [2017]) and (Wu [2015]) and the models implemented by Alex Dai (Dai), where they modeled the spread between two stocks assuming Ornstein-Uhlenbeck (OU) process. Following their ideas, we selected co-integrated S&P 500 stock timeseries using cointegration test, and created input features using OU process, this process will be detailed in section 5.1. To make our model more realistic, instead of using the absolute spread level as the label, we convert the problem into a classification problem that predicts the direction of the future spread move, as it is much more feasible and easier to predict the general trend of the spread rather than the absolute level. Therefore, the label is 1 when the spread between the two stocks will tighten in the future and 0 otherwise, so if we can correctly predict a label then it is guaranteed to realize the profit.

Our baseline model is the traditional time series model, then logistic regression, Gaussian Discriminant Analysis, Support Vector Machine (SVM), and Neural Network will be applied. Eventually the accuracy across different models are compared and analyzed.

2 Related Work

In the early era of pairs trading, it mainly replis on empirical study and the number of model used is limited. (Evan Gatev [1998]) introduced distance method, a non-parametric trading strategy which executes a trade when the spread between two stocks deviates two standard deviations from the mean, this strategy resulted large excess return up to 11%. (Vidyamurthy [2004]) created one of the first parametric methods in pairs trading, detailed the co-integration method and created a framework for pairs selections used for pair trading. Even though (Vidyamurthy [2004]) did not provide any empirical results, his co-integration method is still widely used now in the industry. (Robert J. Elloitt and Malcolm [2005]) built up an analytical framework for pairs trading, proposed a mean-reverting Gaussian Markov chain model for the spread which is observed in Gaussian noise. Predictions from the calibrated model are then compared with subsequent observations of the spread to determine appropriate investment decisions. (Robert J. Elloitt and Malcolm [2005]) makes it feasible to model the mean-reversion property of the spread between two stocks, and it is the essential part of pairs trading. Based on this analytical framework, (Marco Avellaneda [2010]) modeled the idiosyncratic returns as mean-reverting processes, and generated trading signals using Principal Component Analysis (PCA). Nowadays, the prediction of financial timeseries focuses more on the neural network techniques, (Danko Brezak [2012]) forecasted financial time series performances of feed-forward and recurrent neural networks (NN) trained with different learning algorithms, concluded that both methods worked well on the stationary time series. There will definitely more to come in the use of deep learning for pairs trading.

3 Feature Engineering and Label Creation for Pairs Trading

3.1 Dataset

The datasets used cover the S&P daily closing price from 1986 to 2018, and historical stock prices from the last 5 years (including the daily open, high, close, and volume) for all companies found on the S&P 500 index as of February 2018.

3.2 Feature Engineering

The input features of our pairs trading model are modeled as the normalized residuals:

$$T_{feature} = |\frac{\xi_t^{feature} - \mu_{feature}}{\sigma^{feature}}| \tag{1}$$

Therefore we need to model $\mu_{feature}$ and $\sigma^{feature}$

The classical pairs trading spread model is:

$$\frac{dA_t}{A_t} = \alpha dt + \beta \frac{dB_t}{B_t} + d\xi_t \tag{2}$$

where A_t and B_t are the two securities for pairs trading, and $d\xi_t$ is the spread we want to model. After predicting the direction of $d\xi_t$, we will go short the spread if it is too wide or long the spread if it is too tight.

To model $d\xi_t$, we use the classical method proposed by Elloitt et al. (2005). who assumes that the mean-reverting spread is driven by a latent state variable ξ_t following an Ornstein-Uhlenbeck process, which is defined by:

$$d\xi_t = \kappa(\mu - \xi_t)dt + \sigma dW_t \tag{3}$$

Where μ and σ are the parameters we need to estimate, and dW_t is the white noise. Making use of the fact that the solution to (3) is Markovian, the equation can be re-written to a discrete time transition equation, which a a simple AR(1) process, resulting in the following:

$$\xi_{t+1} = A + B\xi_t + \epsilon_{t+1} \tag{4}$$

where $A = (1 - e^{-\theta\tau})\mu$, $B = e^{-\theta\tau}$ where τ is the time interval between two data points and $\tau = 1$ in our case as our time interval between two observation is one day. Notice that A and B can be simply estimated from running an AR(1) model on the residuals of the time series. Re-writing the definitions for A and B gives the closed form formula for μ and σ as shown below:

$$\mu = \frac{A}{1-B}, \quad \sigma = \sqrt{\frac{Var(\epsilon_t)}{1-B^2}} \tag{5}$$

where $Var(\epsilon_t)$ is simply the sample variance of residual from (4).

Finally, our input features for the models are the normalized version of the residuals from (1).

In our model, the normalized input features are created from: 1. 30 day moving average price 2. 5 day moving average price 3. 30 day standard deviation of the price 4. z-score, which is $\frac{MA_5 - MA_{30}}{std_{30}}$ 5. daily price returns.

3.3 Label Creation

Instead of predicting the absolute level of price returns, it is much easier and feasible to predict the direction of the future spread move, hence the label can be simply created as:

if current spread - minimum spread in next 10 days > threshold:

then label = 1

else: label = 0

Here threshold is a hyperparameter we need to set. This labeling implies our holding period, it means that if our model predicts the current spread is too wide and will tighten in the future, then we should buy the underpriced stock, short the overpriced stock. We will hold this portfolio for at most 10 days, as long as our prediction is correct, meaning the spread will indeed tighten in the future, then we will definitely make money.

4 Methods

4.1 Time-Series Analysis

The baseline is an ARIMA model using a simple average and exponential smoothing on the log differences between closing price each day. By taking the log we convert an exponential growth pattern into a linear trend and stabilize the variance. Let p be the number of autoregressive terms and q be the number of lagged forecast errors in the prediction equation. Then the general focasting equation for ARIMA(p, q) is:

$$\hat{\mathrm{y}}_\mathrm{t} = \mu + \phi_1 \mathrm{y}_{\mathrm{t}-1} + \ldots + \phi_\mathrm{p} \mathrm{y}_{\mathrm{t}-\mathrm{p}} - \theta_1 \mathrm{e}_{\mathrm{t}-1} - \ldots - \theta_\mathrm{q} \mathrm{e}_{\mathrm{t}-\mathrm{q}},$$

with auto regressive parameters ϕ and moving average parameters θ indexed by p and q respectively.

4.2 Co-integration test for Stock picking

The first task in the pair trading is to choose the appropriate pairs of stocks that are highly co-integrated, mathematically:

$$Y = \alpha X + \epsilon \tag{6}$$

where X and Y are the two stocks, α is the constant ratio and ϵ is the white noise. For pairs trading to work between two time series, the expected value of the ratio over time must converge to the mean, i.e. they should be co-integrated. We use the MacKinnon's approximate of the p-value to determine the co-integration of the two series, and the criteria is that the smaller the p-value, the higher the co-integration of the two series, and the more suitable these two stocks should be used in pairs trading.

We run the co-integration test on the top 50 largest shares in S&P 500 in terms of market capitalization, since these are the most liquid and stable shares, then choose the top 10 co-integrated pairs.

4.3 Logistic Regression

The logistic regression is the classical model in classification problem, here we assume $y|x;\theta \sim$ Bernoulli(ϕ), subsequently our model is:

$$p(y = 1|x;\theta) = h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{7}$$

where θ are the parameters we estimate and x are the input features. The output from the logistic function is always between 0 and 1, hence to classify the input, the prediction is 1 when $h_\theta(x) > 0.5$ and 0 otherwise.

4.4 Gaussian Discriminant Analysis

The Gaussian Discriminant Analysis (GDA) is another algorithm for classification problem, the assumptions of the GDA are

$$y \sim Bernoulli(\phi) \tag{8}$$

$$x|y = 0 \sim N(\mu_0, \Sigma) \quad and \quad x|y = 1 \sim N(\mu_1, \Sigma) \tag{9}$$

We can derive the Maximum Likelihood Estimate for parameters ϕ, μ_j and Σ as

$$\phi = \frac{1}{n}\sum_{i=1}^{n} 1_{y^(i)=1} \quad \mu_j = \frac{\sum_{i=1}^{n} 1_{\{y^(i)=j\}} x^(i)}{\sum_{i=1}^{n} 1_{\{y^(i)=j\}}} \quad \Sigma = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \mu_{y^(i)})(x^{(i)} - \mu_{y^(i)})^T \in \mathbb{R}^{d\times d} \tag{10}$$

Once we have the estimation of parameters, the prediction is simply:

$$p(y = 1|x;\phi,\Sigma,\mu_j) = \frac{1}{1 + exp(-\theta^T x)} \tag{11}$$

and the prediction is 1 when $p(y = 1|x;\mu_j,\Sigma) > 0.5$ and 0 otherwise.

4.5 Support Vector Machine

The support vector machine aims to find the line that maximize the minimum distance to the line separating the two classes of data, the goal is to find the w from the optimization problem:

$$min_{w,b} \quad \frac{1}{2}\|w\|^2 \tag{12}$$

$$s.t. \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 \tag{13}$$

Note that to improve the predictive power of the SVM, we apply RBF Kernel to the input features in order to capture higher dimension from the data, it is commonly used in SVM. The RBF kernel on two samples x and $x^{'}$, represented as feature vectors in some input space, is defined as:

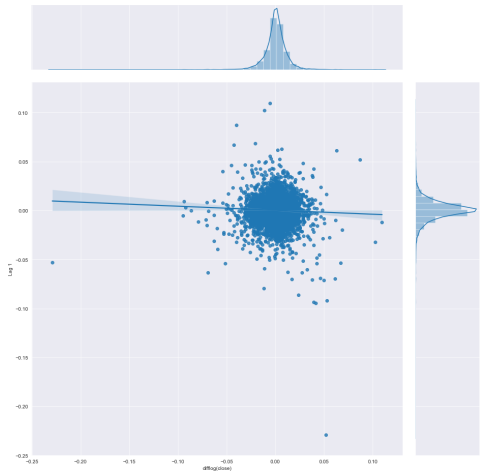
$$K(x, x^{'}) = exp(-\frac{\|x - x^{'}\|^2}{2\sigma^2}) \tag{14}$$

4.6 Neural Network

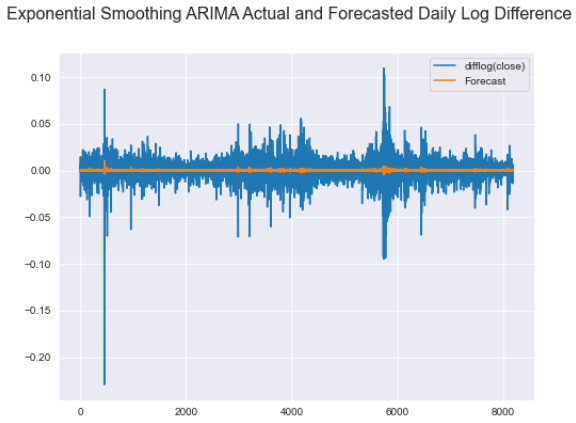
The neural networks are a class of models that are built with layers, it has the ability to learn by themselves and produce the output that is not limited to the input provided to them. It is able to capture high degree of non-linearity in the data. Even if a neuron is not responding or a piece of information is missing, the network can detect the fault and still produce the output. In this project, we developed a neural network with three layers, using ReLU activation for the hidden layers, and sigmoid activation function for the output layer.

5 Experiments/Results/Discussion

The results for our baseline ARIMA model showed little correlation between the change in value from one day to the next (5a). As a result, the model predicts tiny variations relative to what is actually happening day-to-day (5b). This motivates our pairs trading strategy.



(a) Original variables vs. Lagging Variables



(b) ARIMA: Moving Average with Exponential Smoothing

Figure 1: Baseline Results

For our classification problem, the classical evaluation metrics are Accuracy, Precision, Recall and F1 Score, they are defined as:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{15}$$

$$Accuracy = \frac{TruePredictions}{AllPrediction} \quad F1 = 2 * \frac{Precision * Recall}{TruePositive + FalseNegative} \tag{16}$$

We ran the co-integration test as described above, and pick the top 10 co-integrated pairs of stocks as our pairs trading sample, then we ran the four models of each pair, the results from our model are shown in Figure 3 and 4.

Accuracies of Predictions across Different Methods					
ticker1	ticker2	logistic	GDA	SVM	NN
ABT	COST	57%	57%	57%	58%
ACN	NEE	50%	52%	49%	47%
AMZN	NFLX	63%	63%	65%	65%
COST	ACN	51%	51%	49%	52%
GOOG	NEE	48%	48%	45%	46%
INTC	MRK	55%	54%	54%	56%
INTC	PFE	56%	56%	57%	57%
PFE	MRK	54%	54%	51%	53%
T	MRK	56%	56%	56%	53%
VZ	KO	53%	53%	53%	54%

(a) Accuracy

F1 Score of Predictions across Different Methods					
ticker1	ticker2	logistic	GDA	SVM	NN
ABT	COST	72%	72%	71%	72%
ACN	NEE	54%	53%	56%	58%
AMZN	NFLX	76%	76%	79%	78%
COST	ACN	58%	57%	63%	63%
GOOG	NEE	54%	53%	59%	55%
INTC	MRK	68%	67%	68%	70%
INTC	PFE	69%	69%	70%	69%
PFE	MRK	63%	61%	64%	64%
T	MRK	70%	70%	69%	66%
VZ	KO	51%	50%	61%	59%

(b) F1 Score

Figure 2: Modelling Results

In terms of accuracy, the performance has shown that the prediction from Neural Network is slightly higher than other models, proving that it is able to capture higher degree of non-linearity of the stock spread changes. Moreover, the performance difference between logistic regression and GDA are negligible. GDA shows reasonable predictive power, meaning that the stronger assumption in GDA is realized, and the spread is indeed distributed as a normal distribution. Another observation from 3(a) is that the SVM's performance is not superior of the other three. It might be due to our insufficient hyper-parameter tuning or incorrect use of the Kernel, this is definitely something we should explore further.

On top of the accuracy, the F1 score might be a better measure to use if we need to seek a balance between precision and recall, especially under the situation that there is an uneven class distributions. The F1 score confirms that the NN has stronger predictive power than the rest of models. Interestingly, the F1 score of SVM is higher than that in logistic regression and GDA, meaning that SVM has greater predictive power in predicting the positive examples.

ticker1	ticker2	model	Accuracy	Precision	Recall	F1
ABT	COST	logistic	57%	57%	96%	72%
ABT	COST	GDA	57%	58%	95%	72%
ABT	COST	SVM	57%	57%	96%	71%
ABT	COST	NN	58%	59%	92%	72%

Figure 3: Pairs trading (AMZN, NFLX)

We also analyzed the result from another angle, we analyzed the performance of each model from each evaluation metric only for AMZN and NFLX pair, the interesting finding is that the pairs trading model tend to perform well for predicting the positive examples, but on the other hand, it tends to classify lots of negative examples into positive examples, hence driving down the precision ratio. This is the direction we should be working on to improve the accuracy, i.e. reduce the false positive predictions.

6 Conclusion

In this application project, our objective is to predict future directions of spread between two stocks, a technique that has been widely used in the pairs trading strategy. Our dataset is S&P500 timeseries all the way from 1980s. We ran the co-integration test to the top 50 largest stocks, selecting pairs that are co-integrated and hence suitable for pairs trading. We ran four classification models, namely logistic regression, GDA, SVM and Neural Network, to predict the future direction of spread move. The result has shown that the Neural Network has the highest predictive power over other models, mainly due to its greater flexibility and ability of capturing non-linearity in the data. Moreover, models are more inclined to predict correctly the positive examples, and in the downside, create a lot false positive examples.

Future steps include improving our existing model by tuning the hyper-parameters (increase the training size, etc.), and exploring more modern models used in pairs trading, such as employment of Long Short Term Memory (LSTM). Literature has proved that Neural Network with LSTM has greater predictive power for financial time series.

7 Contributions

Alex focused on implementing time series analysis models. Owen focused on further pairs-trading models.
Our code is available at: `https://github.com/akfuster/CS229`

References

A. Dai. High frequency trading svms. *https://github.com/alexdai186/HighFrequencyTradingSVMs*.

D. M. J. K. Danko Brezak, Tomislav Bacek. A comparison of feed-forward and recurrent neural networks in time series forecasting. In *Conference: Computational Intelligence for Financial Engineering Economics (CIFEr), 2012 IEEE Conference on*, 3 2012.

K. G. R. Evan Gatev, William N. Goetzmann. Pairs trading: Performance of a relative value arbitrage rule. In *Yale ICF Working Paper No. 08-03*, 12 1998.

J.-H. L. Marco Avellaneda. Statistical arbitrage in the us equities market. *Quantitative Finance*, 10:761–782, 2010.

J. V. D. H. Robert J. Elloitt and W. P. Malcolm. Pairs tradings. *Quantitative Finance*, 5:271–276, 2005.

R. van der Have. Pairs trading using machine learning: An empirical study. *Erasmus School of Economics*, 12 2017.

G. Vidyamurthy. *Pairs Trading: Quantitative Methods and Analysis*. Wiley, 2004.

J. Wu. A pairs trading strategy for goog/googl using machine learning. *Stanford University CS229 Final Project*, 12 2015.