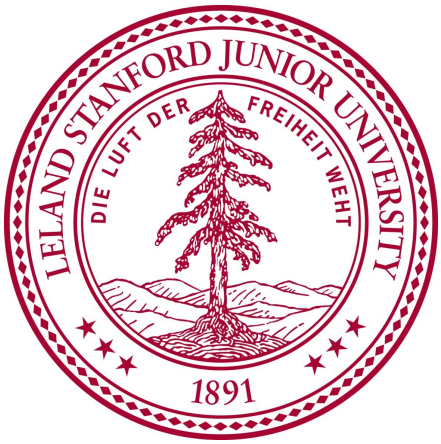


Stanford University

MS&E 448

Big Financial Data and Algorithmic Trading

# Trend Following: A Machine Learning Approach

*Authors:*
Art Paspanthong, Divya Saini, Joe Taglic, Raghav Tibrewala, Will Vithayapalert

June 10, 2019

# Contents

# List of Figures

# Introduction and Strategy

Trend following is one of the most classic investment styles used by investors for over decades. The concept of trend following is relatively simple: When there is a trend, follow it; when things move against you or when the trend isnt really there, cut your losses.

However, due to its simplicity, our team believes that trend following strategy itself might not be able to capture the nuance and the complexity of the financial market. Consequently, with increased availability of data, we believe machine learning techniques could play an important role in constructing a better trend following portfolio. That's why our task for this project is to replicate and improve on the basic ideas of trend following.

# Data

## Investment Universe Selection

As per the project proposal, we narrowed down our universe of assets to futures markets. Using data sets from Quandl, we have access to multiple different futures contracts. However, we first select 9 different commodities to start off with, including Crude Oil, Natural Gas, Gasoline, Gold, Silver, Copper, Agriculture, Corn, Wheat, and Soybean. We consider 6 different contracts for each commodity (1 to 6 months expiration). The primary reason for looking into a diverse set of assets is to diversify the portfolio. In addition to that, the volume of futures contracts for specific commodities could be a lot smaller than equity markets. Large buy or sell orders could potentially move the market. That's why we want to invest in many different contracts.

After inspecting and considering each data set, we ended up selecting 7 different commodities, dropping Natural Gas and Gasoline from our study due to incompleteness of the data set. In addition to that, we also filter out commodities futures with low volume out as well. In the end, we have in total of 36 different contracts from 7 commodities.

## Data Exploration

Since the data set we selected are relatively complete, we did not encounter any challenging problems. However, the original features in the dataset is somewhat limited, so we decided to add approximately 50 new "trend-following" features into the data set. Details of these features will be discussed in the next section.

In addition to that, we also explore the correlation between different assets. The correlation plot is shown in the figure below.

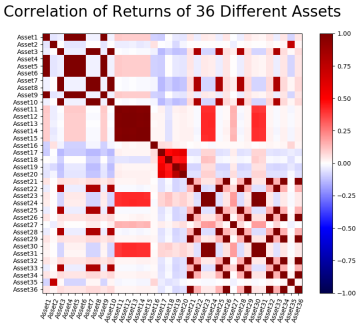


**Figure 1:** Correlation of Returns of 36 Different Assets

In the plot above, there are quite a few noticeable clusters of assets with high positive correlation. Such clusters are the same commodity with different expiration period. It's also notable that among all assets we selected, there is no pair of futures contracts that have high negative correlation.

# Feature Generation

Features selected for the modeling were based on traditional trend following indicators. These were used in the prediction of the final response variable, next day return, or $(P_{t+1} - P_t)/P_t$.

### Continuous Variables

1. Simple Moving Average (SMA)
2. Exponential Moving Average (EMA)
3. Moving Average Convergence Divergence (MACD)
4. Momentum Indicator
5. Day Since Cross
6. Number of days up - down

The simple moving average, momentum indicator, and number of days of price upward movement minus number of days of price downward movement were calculated over several lookback windows. Specifically over the time-frames of 5, 10, 15, 20, 50, and 100 days back. EMA variables were included over lookback windows of 10, 12, 20, 26, 50, and 100 days. And, MACD was calculated as 12-day EMA - 26-day EMA. Days since cross indicates the number of days since the last crossover between an asset price and its EMA.

### Categorical Variables

1. SMA Crossover indicator variables
2. EMA Crossover indicator variables
3. MACD Crossover indicator variables

The categorical variables were labeled at each timestep as +1 to indicate a crossover with buy signal, 0 to indicate no crossover, and -1 to indicate a crossover with a sell signal. They were calculated as asset price crossovers with all the SMA, EMA, and MACD indicator variables mentioned in the continuous variables section. In traditional trend following strategies, these crossover variables are important indicators of detecting upward or downward trends that can be ridden for profit. Our reasoning for feeding all of them into our models was to allow the algorithm to determine which ones are more accurate predictors of next day returns.

## Models

### Linear Model

First, a linear regression model was trained on 2014-2017 data and tested on 2017-2018 data. The technique provided fairly stable predictable patterns and in the unregularized version, all parameters mentioned in the feature generation section of this paper were used. A separate regression was run on each asset available in the training data in order to allow the models more expressiveness in their understanding. The advantages of using a linear model on this problem are that it is simple and easy to understand, and it fits decently well to the data. Second, a regularized lasso regression model was trained on the same training data and tested on the same test data. Finally, a linear regression model was trained to predict returns over a longer time frame. Specifically, on 5-day returns. We attempted this model because in a non-ideal trading system there are frictions. Namely, that one-day returns are small and may be erased by transaction costs and we might not enter the position until the next day. So, the question became whether we could reliably predict 5-day returns and whether that would improve the efficacy of our trading algorithm.

### Results

The figures below showcase the plots of the predicted versus actual values as well as a histogram of the linear regression errors.

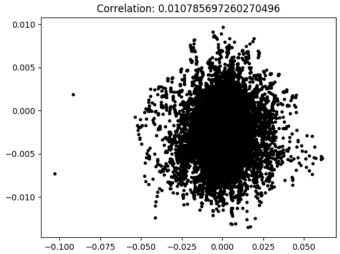


**Figure 2:** Predicted versus actual values of unregularized linear regression model.

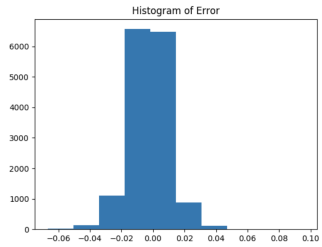


**Figure 3:** Histogram of error values of unregularized linear regression model

The overall train mse was 2.187 E-04. The test mse was 1.47 E-04. In analyzing the beta values of the linear regression, we noticed that exponential Moving Averages are generally better predictors than simple moving averages in terms of higher absolute values of betas. One of a 5 day, 10 day, 12 day, and 100 day indicators were statistically significant at the five percent level. Thus we also noticed that recent trends are most significant, though longer term trends are not irrelevant. Finally, we noticed that because of the change of sign between EMA 10, 12, 20 indicator variable beta values, there is an importance to recent crosses, which validates the inclusion of categorical crossover variables in our feature selection. These beta values are summarized with their p-values in the chart below.

| Covariate Name | Beta Value | P Value |
|---|---|---|
| **EMA 10** | **4.84** | .004 |
| EMA 20 | 2.45 | .219 |
| **SMA 100** | **0.06** | .004 |
| EMA 100 | 0.03 | .17 |
| SMA 15 | 0.01 | .817 |
| SMA 10 | -0.002 | .981 |
| SMA 20 | -0.008 | .866 |
| SMA 50 | -0.04 | .149 |
| **SMA 5** | **-0.07** | 0.044 |
| EMA 50 | -0.29 | .235 |
| EMA 26 | -0.45 | .749 |
| **EMA 12** | **-6.38** | .016 |

**Figure 4:** Beta values of unregularized linear model and their significance values.

The overall trading strategy based on the linear regression model price predictions performed quite well. Below is a chart of the portfolio growth based on the linear regression model compared to a naive strategy. Over the course of 2017-2018, the portfolio grew to 1.3x using the linear regression model return predictions.

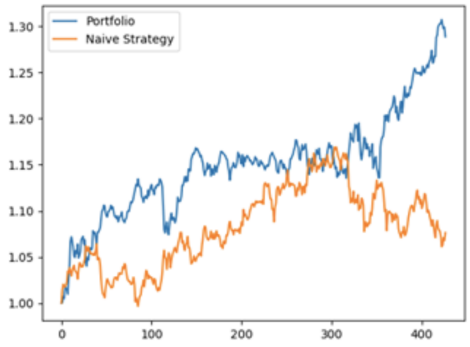


**Figure 5:** Portfolio over 2017-2018 using unregularized linear model predictions.

Next, for the lasso model, we decided that it may be interesting to train in order to get rid of some of the overfitting of a linear regression. This would be accomplished by automatically selecting only more important features. The advantages of this model would be that it is less likely to overfit and is less prone to noise, which we believe there is a lot of in the pricing data. The disadvantages are that it does not solve the complexity issue and can reduce the expressiveness that we may need in explaining returns. The lasso model predicted versus actual distribution as well as error histogram are displayed below.

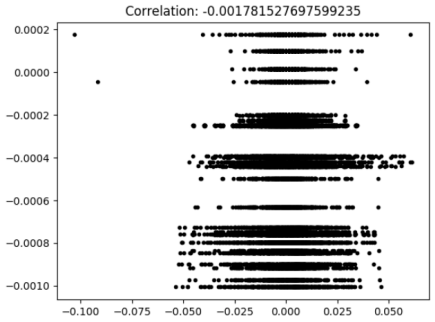


**Figure 6:** Predicted vs actual values of the lasso regression model.

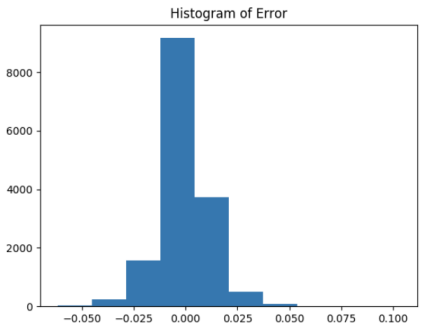


**Figure 7:** Lasso regression model histogram of errors.

It turns out that though the mse were relatively similar to the unregularized linear model, with a train MSE of 2.281 E-04 and a test MSE: 1.353 E-04, the overall strategy based on the return predictions performed worse over the course of our test period. The portfolio growth compared to the naive strategy are displayed below.

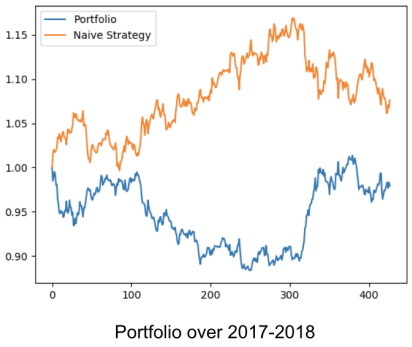


**Figure 8:** Portfolio over 2017-2018 using lasso model predictions.

Finally, for the 5-day return predictions we noticed 5-day returns are generally about 2-3x larger than 1 day returns, and, thus, a roughly 6.5x increase in mean squared error (MSE: 9.47E-04) indicates that the predictions are about equivalent to 1-day predictions. The portfolio performed as shown in the figure below. The 5-day return portfolio did not perform as well as our 1-day return portfolio, with merely a 1.2x growth factor as compared to the earlier 1.3x growth factor over this test set period.

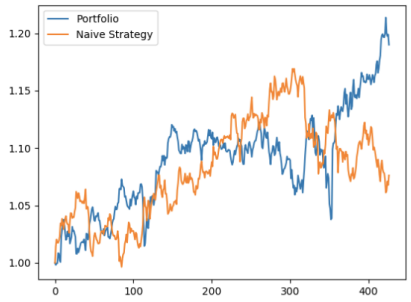


**Figure 9:** Portfolio over 2017-2018 using 5-day linear regression return predictions.

Interestingly, the daily returns of this portfolio vs. the naive portfolio are fairly comparable (0.04% vs. 0.02%) but the 5-day returns are notably better (0.22% vs. 0.07%).

## RNN Model

Recurrent Neural Network (RNN) model is considered to be one of the most powerful models that can make accurate prediction on future stock prices. Especially Long Short Term Memory (LSTM) model has its configuration that incorporates historical information to capture the data pattern. Furthermore, most of research concluded that Neural Network structure has outperformed simple linear regression in substantial margins, although they didn't explicitly explain how specific hyper-parameters were selected. We also choose to build LSTM architecture to investigate whether it can drive up the profitability of our trend-following strategy.

In this project, our RNN architecture consists of 3 layers of LSTM, and one fully-connected layer at the end. Each layer has 128 hidden units with the linear activation in the last step, as the prediction is a regression problem. The input features include six exponential moving averages (10, 15, 20, 50, 100 days lookback window), six simple moving averages (10, 15, 20, 50, 100 days lookback window) as well as the MACD. To fasten the covergence of optimization algorithm, we also normalize each input feature by transforming them to be a standardized Z-score. The de-

tails of LSTM architecture are illustrated below.

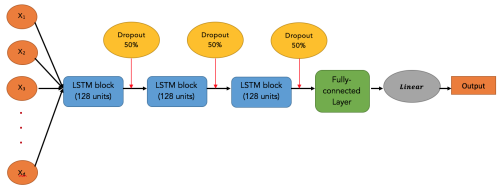


**Figure 10:** The architecture of 3-layer LSTM

In the modeling process, we trained the model by using all available data prior to 2016 and used the validation set to perform regularization. As illustrated in the figure above, one of our regularization techniques is dropping out 50% of parameters between hidden layers. In addition, we also used early stopping when the training loss increases and doesn't seem to converge to lower loss.

The last essential step is tuning paramaters and hyperparameters to improve the model performance. We used grid search method to construct multiple sets of variables and chose the most optimal set. The grid contains different values of 4 hyperparameters (learning rate, number of epochs, number of hidden units, and batch size) and 1 parameter (lookback window over the past 1, 5, 10, 15, and 25 days). Using this approach, we obtained the optimal lookback window and hyperparameters as following:
Learning rate: 0.0001
Number of epochs: 50
Number of hidden units: 128 units
Batch size: 32
Lookback window: 10 days

### Results

We visualized the results of LSTM model including the correlation between actual returns and predicted returns, the histogram of errors, and the plot of portfolio value over time. First, the plot of correlation shows that the predicted returns are not centered at a certain point but rather more spread out, in contrast to those from linear regression. This suggests that LSTM's prediction doesn't follow a particular pattern and tends to be more randomly made, as illustrated in the plots below.

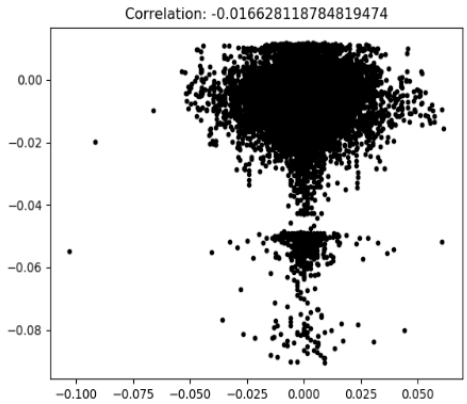


**Figure 11:** Correlation of actual next day's returns and predicted next day's returns

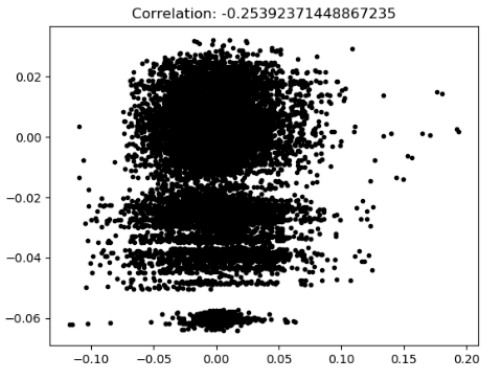


**Figure 12:** Correlation of actual next 5-day's returns and predicted next 5-day's returns

We also observed that the prediction on next 5-day's returns is more random than the one of next day's return. We suspected that the prediction on further period might be less accurate. After looking at the histogram of errors, we can conclude that the further prediction is indeed less accurate. The histogram of errors for the next 5-day's return appears to be more variant.

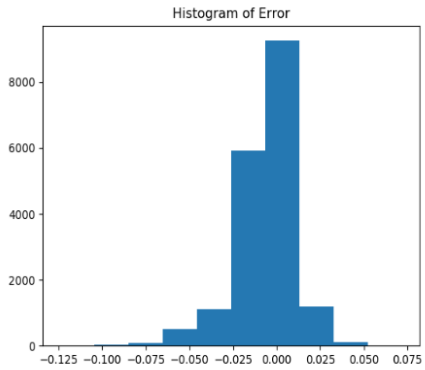


**Figure 13:** Histogram of errors for prediction on next day's returns

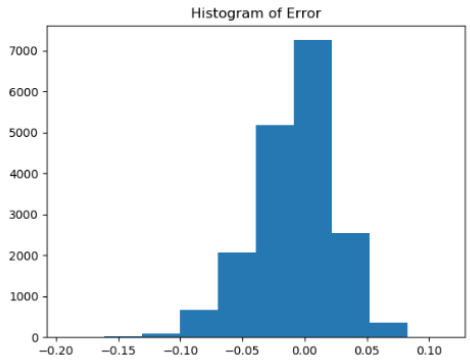


**Figure 14:** Histogram of errors for prediction on next 5-day's returns

Despite less accurate prediction, the portfolio constructed by trading based on next 5-day's returns turns out to be more profitable than the one of next day's returns. However, as illustrated in the plots below, over 426 trading days on the test set, leveraging LSTM models in our trading strategy doesn't produce significant alpha.

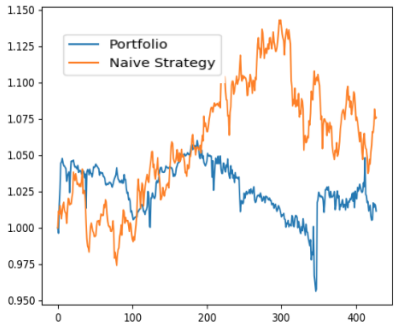


**Figure 15:** Portfolio value over 2017-18 using LSTM model prediction on next day's returns

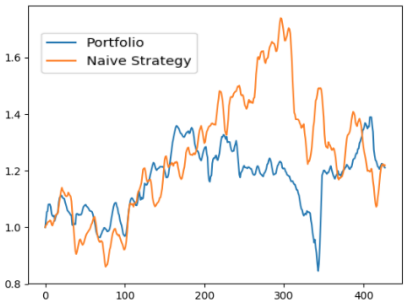


**Figure 16:** Portfolio value over 2017-18 using LSTM model prediction on next 5-day's returns

Although the results are not compelling, this is not surprising to use. There is in fact a common conclusion from many deep learning research that complex models as LSTM or other types of RNN would outperform simple regression only if the size of training data is very large (approximately in $10^5$ scale). However, the size of our dataset is only in $10^3 - 10^4$. Hence, it doesn't allow LSTM model to capture the trend of data and perform well as it is supposed to be.

## Neural Net Model

We also employ a supervised learning algorithm: the fully connected neural network. The goal is to perform regression: the output variable is the daily return which is a numerical continuous variable.

We wish to better understand the general relation between the return and the 26 input data features fed to the neural network. The input features consist of six exponential moving averages (with 5, 10, 15, 20, 50, 100 days lookback window respectively), six simple moving averages (with 5, 10, 15, 20, 50, 100 days lookback window respectively), the MACD (moving average convergence divergence) and the respective crossover data (categorical variables) for each of these parameters. We normalize the continuous features by dividing the numerical value of the feature by the respective settle price on that day and subtracting by 1.

The architecture of the neural network is one

input layer with 26 input units as described above, two hidden layers with the ReLU (rectified linear units) activation functions and one output layer. The number of hidden units we have used in our model are 50 and 20 respectively.These were selected arbitrarily. Since, the neural network is fully connected, the number of parameters which the network must learn are 2320 ($26 \times 50 + 50 \times 20 + 20 \times 1$ ignoring bias units). The hyper-parameters in the neural network model are the learning rate and the convergence error.

In the neural network model, we have used all the data for all the assets before 2016 as training data. Since, the model trains simultaneously for all 42 commodities in the dataset that we have considered, it implicitly assumes that the relation is similar for all the different commodities. This might not necessarily be true. Data from 2016 to 2017 is taken as the validation data. This validation dataset is used for stopping our training. The stopping criterion is when the loss difference over the validation set decreases below the convergence error, we stop training the neural network. All data after 2017 is used as test data.

The first difficulty in the implementation of this model was the implicit assumption that all commodities have a similar input - output relation as discussed above. The second difficulty comes from the fact that the neural net model uses random initialization of the parameters and uses an iterative process (gradient descent) to find the minimum of the cost function. The stochastic nature of the model gives us different results on running the model. In the figures below, the same neural network was run thrice on the same data, but the neural network returned different values. The blue line shows the value of the portfolio is the neural network was used and it has been compared to the orange line which is the naive strategy where the all the commodities have been bought equally.

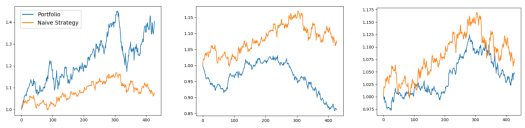


**Figure 17:** Different Results given by Neural Net Model due to Stochastic Nature of Neural Nets

To ensure that the model was training and running correctly, we have plotted the loss value as a function of the epochs. This is shown in the figure below. Since is the loss is monotonously decreasing as a function of the epochs, the neural net is indeed training correctly.

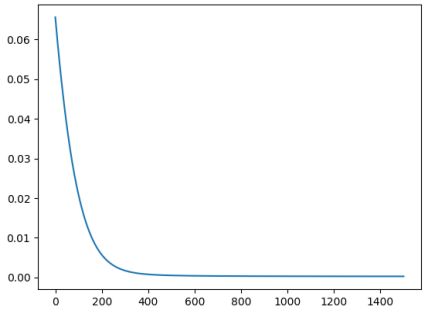


**Figure 18:** Loss as a function of epochs

### Comparison with Linear Regression

We removed the ReLU non-linear activation layer and worked only with linear layers in the neural network. A fully connected neural network without any non-linear layers reduces to a simple linear regression. However, the linear regression results above used a package which solves the normal equation whereas the neural network uses an iterative simple gradient descent to do the same. Solving the normal equation gives the same result every time however, simple gradient descent being an iterative procedure, the neural network without any anon-linear activation still does not yield the same result. The figure below on the left shows the final linear regression output whereas the figure on the right shows the final neural network output without any non-linear activation. As we see, both the linear regression and neural network do better

than the naive strategy and both their outputs are close.

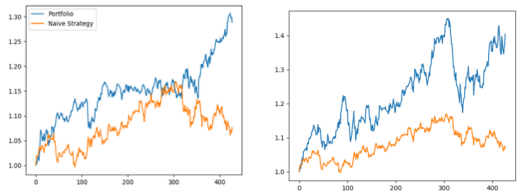


**Figure 19:** Comparison of Linear Regression and Neural Network without Activation

We also observe the following when we compare a neural network to the linear regression:

- The neural network has been given validation data and the model stops training based on the validation loss, however, the linear regression can theoretically overfit the past data.
- Sometimes the neural network (without non-linear activation) performs better than linear regression but sometimes worse. The mean squared error for linear regression and neural network (without non-linear activation) is almost equal.
- However, with ReLU activation, it performs worse because of the nature of the data and the nature of ReLU. almost equal.

Further, we also note that the linear regression has been trained for different commodities individually whereas the neural network has been trained at once for all the commodities. Hence, the model obtained from neural network can be easily generalized to other asset classes and other commodities very easily.

### Results

To show the results of the neural network, we have used one of neural network models that we trained for 5000 epochs. The correlation of the actual returns versus the predicted results for the test set (after 2017) has been shown below.

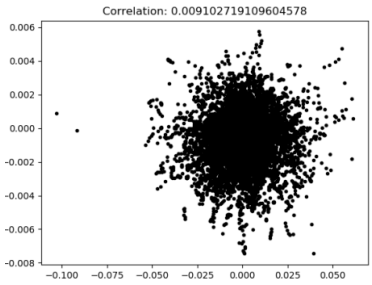


**Figure 20:** Correlation: predicted and actual returns

On observing the correlation plot, we see that there is a cluster of points around zero. This cluster of points is where the actual returns and the predicted returns were both close to zero. However, we also observe outliers which are away from these cluster of points. The outliers can be classified as "good" or "bad". The good outliers are where both the actual and predicted returns are high. These "good" outliers enhance the performance of the model as they generate the **correct** long or short trend signal during portfolio optimization. However, the "bad" outliers are when the predicted returns are high but the actual returns are not. These "bad" outliers are responsible for making bad and wrong bets due to incorrect trend signals they generate. They decrease the overall performance of the model. In the model, if the number of "good" outliers is more than the number of "bad" outliers, the model generates alpha. In this neural net model, we report the correlation to be positive. The histogram of errors is yet another important visualization of the model's performance. It is shown in the figure below.

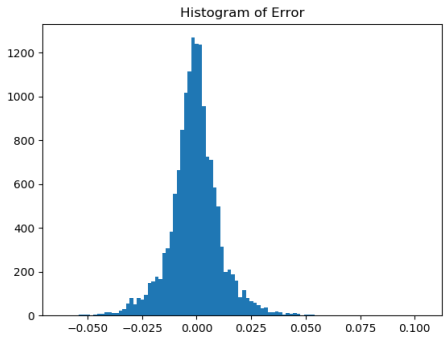


**Figure 21:** Histogram of Errors from Neural Net Model

Using the histogram of errors, we see that the errors are slightly less than zero, which means that the model slightly underestimates the values of the returns. Further, since the histogram is very narrow, the overall variance in the errors of the neural network model is very low.

We report the mean squared error of the neural network model on the training set to be 2.329E-04 and the MSE on the test set to be 1.358E-04. The $R^2$ for this model turns out to be -0.01.

The final portfolio for the neural network that we trained in comparison to the naive strategy is shown in the figure below. It can clearly be seen that the neural network does almost as well as the naive strategy.

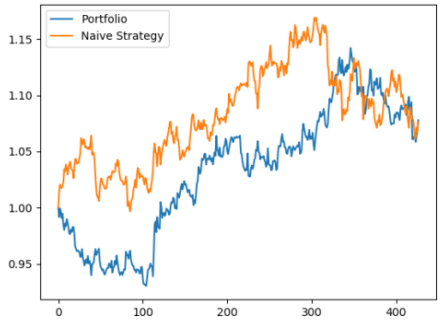


**Figure 22:** Final Saved Portfolio from the Neural Net Model compared to the Naive Strategy

### Summary and Comparison

In most machine learning models, the mean squared error is a metric of performance. The table comparing the mean squared error of all the models, we worked on, is given below:

| Models | MSE train set | MSE test set |
|---|---|---|
| Linear | 2.187 E-04 | 1.47 RE-04 |
| Lasso | 2.281 E-04 | 1.353 E-04 |
| Neural Net | 2.329 E-04 | 1.358 E-04 |
| LSTM | 1.34 E-03 | 3.05 E-03 |

However, when machine learning is applied to trend following, mean squared error can no longer be a metric. The reason for this is trend following looks for signals given by the model to long or short. When the model predicts the values close to the mean, the mean squared error is essentially very low but the portfolio optimizer does not generate any signal to long or short. The predicted outliers which increase the mean squared error significantly also generate trend signals which then trade. AS described in the previous section, if the outliers are "good", then the model does well. However, if the outliers are "bad", the model does not do well.

## Portfolio Construction

### Portfolio Optimizer

Each alpha model discussed in the previous discussion takes in a certain feature vector, which varies slightly from model to model, and outputs expected returns for different assets. Regardless of whether the output returns are daily or weekly, a portfolio optimizer is necessary in order to convert those expected returns to a tradeable portfolio.

We perform a standard portfolio optimization at each time step during simulation. It takes in a vector of current holdings $h$, a vector of expected returns $r$, a desired variance $v$, a transaction cost ratio $b$, and a covariance matrix $C$ to find the optimal set of trades $x$:

$$\begin{aligned} \text{maximize} \quad & w^T(x+h) - b\left(\mathbf{1}^T|x|\right) \\ \text{subject to} \quad & \mathbf{1}^T|x| \leq 1.0 \\ \text{and} \quad & (x+h)^T C(x+h) \leq v \end{aligned}$$

The optimization consists of an objective and two constraints. The objective function is simply a return maximization after transaction costs. The first constraint is a restriction on how heavily the portfolio can be leveraged. In this case, we do not allow leveraging of the portfolio. The portfolio can be long or short, or any combination thereof, such that the total position does not sum to more than the portfolio value. Lastly

the expected variance, based on the covariance matrix, is constrained to be below the desired variance. There is no constraint on position size, as any meaningful constraint would also severely limit the expressiveness of the portfolio due to the small size of the universe.

The inputs to the optimizer were mostly straightforward. During simulation, the current holdings were tracked, and the expected returns came from the alpha model. The desired variance can be set to any value, but was usually set to $4\times10^{-5}$ for daily returns or $2\times10^{-4}$ for weekly returns, roughly equivalent to the "market" variance. In most simulations, the transaction cost ratio was set to 0.0 and so it did not factor into the trade calculation. The covariance matrix was calculated based on returns from the validation dataset (data from the year immediately prior to the test data).

The covariance matrix first used a simple covariance calculation over the entire time period. However, after initial testing showed that it fairly poorly predicted the resulting variance, a shrunk version of the matrix was tested. Generally, covariance matrix shrinkage pulls the most extreme coefficients towards more central values, systematically reducing estimation error. Ledoit and Wolf of the University of Zurich developed a namesake algorithm for shrinking matrices that "can result in sizeable improvements over the sample covariance matrix and also over linear shrinkage." [1] Using the scikit-learn implementation of the Ledoit-Wolf shrinkage, the optimizer was able to more closely match the desired variance, and so this shrunk matrix was used for calculations.

## Stop Loss

One element of many trend following strategies is an exit after experiencing a loss, mirroring the adage to cut short one's losses. While we generally expect the models to be able to infer further loss based on recent performance, we also experimented with including a stop-loss step after the optimization step.

The stop-loss was implemented by forcing closure of positions of any assets that moved in the opposite direction of the position more than a certain value since that asset was first purchased. Ultimately the stop-loss was ineffective in improving the performance of the portfolio. At high loss tolerance, the stop-loss almost doesn't change the performance at all. At low loss tolerance, the stop-loss forces the closure of too many positions and significantly reduces portfolio performance. The plots of portfolio over time in the figure below demonstrate this behavior on the test data.

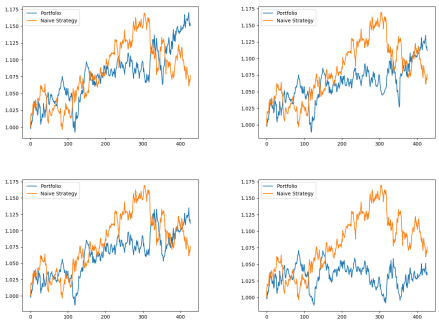

**Figure 23:** Plots of Portfolio Value over Time for Linear Regression Portfolio with Stop Loss (No SL, 15%, 10%, 5%)

While a stop-loss can be effective in cutting losses, there are a few reasons it doesn't in this case. As previously stated, the positions the portfolio takes already account for expected motion. It also seems that most of the losses incurred in the portfolio were fairly large bets that resulted in a large single-day loss, rather than accumulating loss over time. The funds removed by the stop-loss were not otherwise invested, so that could also limit performance. This is also sure to cause more value to be lost to trading costs, but that is a lesser concern.

[1] Ledoit, Olivier, and Michael Wolf. Honey, I Shrunk the Sample Covariance Matrix. The Journal of Portfolio Management, Institutional Investor Journals Umbrella, 31 July 2004, jpm.iijournals.com/content/30/4/110.

## Risk Management Philosophy

The main risk management tool in this strategy is the use of the covariance matrix in the optimizer to restrict excessive fluctuation of the portfolio value in expectation. We measure risk with variance, and show an ability to reasonably limit the variance of the portfolio. Should we prefer a less risky portfolio, we can tune this parameter to effectively reduce the risk of the portfolio.

We attempt to control for big losses by using a stop loss but it had too great an effect on the portfolio's performance to be truly useful in constructing the portfolio. Big losses generally occur when the optimizer makes a big bet on one asset and is wrong. Some measures that can be taken to prevent this are forcing position limits. However, as previously mentioned, the small universe makes it difficult to both limit the size of the portfolio positions and have a portfolio expressive enough to capture the detected alpha. It would be our hope that with many assets, the portfolio might be less reliant on large bets and less sensitive to individual asset movements.

Since the portfolio is not consistently net short or net long, this also eliminates a considerable amount of systemic risk. The main risk would be that in a sharp downturn, the portfolio would attempt to go short, which could be disastrous depending on the execution strategy as most market participants would also be looking to go short or close long positions. This is a fairly considerable risk in the strategy, that we were largely unable to model due to data and simulation limitations.

## Portfolio Results

### Baseline Strategy

Since our goal is to improve upon traditional trend following method, we construct a benchmark portfolio, calling it our baseline strategy, by using traditional trend following's crossover strategy. The baseline strategy is traded base on crossover between EMA 50 (50-day exponential moving average) and the daily price. As per convention, we also set stop losses to limit the amount of risk we are willing to take as well. Specifically, we set the stop loss at 4% and limit sell and 6%.

### Comparison of Results from Different Models

With the baseline strategy defined above, we compare the portfolio results from different models shown in the figure below.

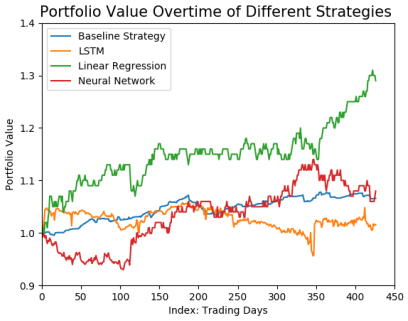


**Figure 24:** Comparison of the portfolio over time for different models

It is noteworthy that linear regression performs the best in terms of the annualized profit and portfolio result, even better than the more complicated models like Neural Network and LSTM. Even though the financial market is extremely complicated and doesn't seem to be easily explained by a linear regression, the result is not surprising to us. Given that we have access to free data on Quandl, the quantity of data points we have is limited. In our best data set, we would have approximately 14,000 data points of daily price of the futures contracts. However, there are some commodities where we only have approximately 1,300 data points. Lack of data to train complicated model could be the main reason why they under perform in this setting.

In addition to portfolio profit, we also con-

sider other metrics as well. Specifically, we look at the Sharpe ratio and maximum drawdown of each model. Comparison of our models are shown in the table below.

| Strategies | Sharpe Ratio | Annual Return | Max Drawdown |
|---|---|---|---|
| Baseline Strategy | 1.494 | 3.815% | -0.4134% |
| Linear Regression | 1.436 | 15.66% | 0% |
| Neural Network | 0.4777 | 5.129% | -7.000% |
| LSTM | 0.1434 | 1.351% | -4.360% |

We can see that the baseline strategy was able to achieve the highest Sharpe ratio, despite posting average annual return of only 3.815%. In the aspect of risk-adjusted return, our baseline strategy perform the best. However, the ability to achieve higher profit of the baseline strategy is still questionable. This is simply because high leverage in market with less volume like futures markets means higher price impact with all the trades we make. That's why we might not be able to achieve 15.66% return while retaining the same Sharpe ratio by just leveraging our baseline strategy 5 times.

In addition, it is quite interesting to see results where maximum drawdown of all models are very low. This could potentially be because our test-set is a very typical year in terms of price movement. The year 2017-2018 don't have any tail events that could be a challenge for our models.

## Execution Discussion

If we were to trade this strategy, we would be likely to use the baseline model as opposed to any of the regressive models. In general, it did better at identifying alpha and had a lower Sharpe ratio, but also at significantly lower returns. During execution, however, we would not be able to observe and trade at the same price, which was assumed in the model. The additional complications that come with real-world trading are a spread and, if investing a large amount, potentially moving the market with trades.

To meet these challenges would require additional analysis on some more extensive data. The first would be intraday prices, so that the effectiveness of the model could be tested when entry is not immediate. Should that succeed, more data on spread information and transaction costs could give more insight on whether entry is still justified, or at what point entry is justified if different from the original model.

Further down the line, it could be necessary to model the order book in order to truly model the impact of our execution strategy. But at small sums, since trend following at its core suggests entering as soon as a trend is detected, execution would probably be a much less pressing question answered by market orders at the time of trend detection.

## Retrospective Discussion

After having read Trend Following with Managed Futures written by Greyserman and Kaminski, our group decided to create a new trend-following strategy that leverages a data-driven approach to provide us actionable insights. However, the project presented a wide variety of challenges that prevented the strategies from being as successful as we would have liked. The project was definitely abound with learning opportunities, and our work suggests a few different directions we could take moving forward.

Model returns based on past prices, the meat of our project, unsurprisingly provided the most challenges, but also taught us the most. The challenge largely lied in the fickle nature of financial data. A machine learning model is only as good as its training data, and frequently the

training data did not paint a clear enough picture of what observations suggest large upcoming returns. However, using multiple different ML algorithms was a great learning experience. As different models become more prevalent in the field of quantitative finance, knowledge of how they work becomes ever more important, and this was a very solid first step.

If we were to continue with the project, we would hope to do two things: find a more expressive dataset, and do a deeper dive of why the baseline was better at detecting alpha. Our dataset was useful for providing daily prices but that is not the only information useful in trend following; moreover, in a fairly small universe it is hard to sufficiently diversify. This might change the scope of the project, but might give us more interesting and more successful results.

A deeper dive into why the baseline worked while our models did not would probably start with a look at our models. Overall, they were not very good at finding a correlation between past and future performance, as most of the figures of predicted vs. actual returns would bear out. Success generally was largely due to correct prediction on a few outlying data points, largely by chance.

One avenue that was not explored was considering movement of the same asset but with a different contract expiry in the feature vector. Given the correlation between contracts of the same underlying commodity, this could be useful for predicting movements. This might then start to resemble a pairs trading approach, which could also be a valid approach to trading futures but deviates from the trend following background.

If we were to move forward, we might think of the problem in a different way: can we predict when the asset prices will move up, down, or stay constant? Rather than trying to predict the returns in a regression-style algorithm, we could instead classify data points with any of a number of labels based on returns. As the number of labels approaches infinity it would resemble a regression, but at a small number of labels (likely 3-5) it might be possible to create a machine learning strategy that, like the baseline, does not attempt to predict numeric returns but instead attempts to predict sign or size of return more abstractly. This would then require its own, more novel optimizer, or a simple strategy of "trade the most extreme labels equally" could also be used. The hope would be that using classification could reveal the same insights as the baseline crosses-only model.

Overall, the project provided plenty of challenges from both a theoretical and technological perspective. Our models only achieved moderate success, but the experience was valuable and their shortcomings gave us insight into what strategies are effective in quantitative finance. While a successful approach might look different from the work we have done, it was an interesting experience that served as a great introduction to the field.