Algorithmic Trading of Futures via Machine Learning

David Montague, *davmont@stanford.edu*

Algorithmic trading of securities has become a staple of modern approaches to financial investment. In this project, I attempt to obtain an effective strategy for trading a collection of 27 financial futures based solely on their past trading data. All of the strategies that I consider are based on predictions of the future price and volatility of the various securities under consideration, and so the majority of the effort in this project has been directed toward using machine learning techniques to obtain predictions for future price and volatility. This project was inspired by the Quantiacs futures competition, to which I submitted a number of the trading strategies I obtained during my work on this project.

Introduction

The Contest

The goal of this project was to obtain a high-performing trading strategy for the Quantiacs futures contest [1]. Quantiacs is a quantitative trading platform that invests in crowdsourced trading systems, and connects users trading systems with institutional investors, [sharing] a performance fee with the developer. The Quantiacs futures contest was a competition to obtain a trading strategy which invests in a collection of futures securities and attempts to maximize the Sharpe ratio (a measure of risk-adjusted return) over both a training period (Jan. 1, 2001 Nov. 30, 2014) and a live trading period (Dec. 1, 2014 Jan. 31, 2015). Trading systems are ranked using the minimum of the performance over these two intervals, and the top three performing strategies (determined January 31, 2015) will receive guaranteed minimum investments ($250,000 for third place, $500,000 for second, and $1,000,000 for first).

The Trading Model

The trading model employed by Quantiacs is somewhat simplified relative to a real-world trading scenario. In particular, the daily closing price is what is used to determine the buying/selling price when orders are placed on a given day, and orders may only be placed once per day. However, the model does include small fees meant to simulate the effects of slippage and trading commissions, and so may offer a relatively realistic trading simulation for relatively long term investment strategies (where trades are being performed on a daily basis, as opposed to high frequency approaches with many trades being performed each second).

The Data

The backtesting data for the contest consisted of approximately 3800 days of trading price and volume data (from Jan. 1, 2001 to the present day) provided by Quantiacs for 27 different futures contracts (including various currencies, precious metals, agricultural products, etc.). More specifically, the data contained the daily high, low, opening, and closing prices, and daily trading volumes over this time period. In addition, the rules of the competition specified that on any given day, the only legal input into the trading strategy was the previous 504 trading days worth of this data for each of the 27 securities under consideration.

The Objective

As mentioned above, the goal in the Quantiacs futures competition is to obtain a trading strategy with opti-

mal performance as measured by the strategy's Sharpe ratio. The Sharpe ratio is determined by the ratio of average return to average volatility. More specifically, the Sharpe ratio is computed as follows: Let $(e_j)_{j=0}^n$ be the set of equity values of our portfolio over a period of n trading days (starting with value e_0). Define a vector of daily (percent) returns $(d_j)_{j=1}^n$ by

$$d_j = \frac{e_j - e_{j-1}}{e_{j-1}},$$

and compute the average daily and yearly return (using the fact that there are 252 trading days in a calendar year) by

$$r_{\text{daily}} = \left(\prod_{j=1}^{n} 1 + d_j\right)^{1/n} - 1 = \left(\frac{e_n}{e_0}\right)^{1/n} - 1,$$

$$r_{\text{yearly}} = (1 + r_{\text{daily}})^{252} - 1.$$

Next, compute the volatility by the formula

$$\text{vola}_{\text{daily}} = \text{StDev}(\{d_1, ..., d_n\}),$$

and estimate yearly volatility from this by assuming that daily volatility is independent, obtaining

$$\text{vola}_{\text{yearly}} = \sqrt{252}\text{vola}_{\text{daily}}.$$

Finally, define the Sharpe ratio by

$$\text{Sharpe}(e) = \frac{r_{\text{yearly}}}{\text{vola}_{\text{yearly}}}.$$

The various entries in the Quantiacs futures competition are ranked in ascending order according to the minimum of their Sharpe ratio over the evaluation period (Jan. 1, 2001 to Nov. 30, 2014) and their Sharpe ratio over the live trading period (Dec. 1, 2014 to Jan. 31, 2015). Thus, it is important to obtain a strategy which both performs well on the backtesting data from the past, and which also generalizes well to live trading data.

Supervised Learning Problems

One of the guiding principles behind my approach to the contest was that accurate oredictions of the return and volatility for each of the individual futures were sufficient inputs to obtain an effective trading algorithm (where effectiveness is measured by the Sharpe ratio obtained). I also made the implicit assumption that the various securities could be analyzed independently, so that predictions for a given security's future return and volatility need only depend on that particular security's past performance. In reality, it is certainly not the case that the returns of the various securities behave independently, but I still believe this is a reasonable simplifying assumption for a prediction model.

Based on the above, I decided to attempt to predict the future return and volatility of each of the futures independently based on its past performance.

Training Examples

On each trading day, the trading strategy is allowed to use as input the past 504 days worth of trading data for each future. In order to reduce the redundancy of my set of training examples, I downsample the trading days, generating a new training example only once every five trading days for each security. Because of my assumption that each of the futures behaves independently, I obtained one training example for each security and each downsampled trading day for which data was available sufficiently far into the future (so that future return and volatility could be computed). The raw training feature vector is the 2,520-dimensional vector of the past 504 days of daily high, low, closing, and opening prices and trading volumes, but significant dimensionality reduction was performed in order to make the problem more tractable.

Learning Objectives

While I have defined the Sharpe ratio for the portfolio as a whole, the definition can also be readily applied to individual securities, and provides some guidance on how to choose return and volatility prediction objectives. In particular, inspired by the definition of the Sharpe ratio, I chose to use the following two supervised learning objectives for the return and volatility prediction respectively:

- For return prediction, I consider a one-parameter family of objective functions. As shown above, the quantity r_{daily} can be computed directly from the quantity e_n/e_0, the performance of a given future over n trading days (here e represents the price of the specific futures contract). Therefore, I use $R_n := e_n/e_0$, the percent increase in price n days in the future, as the prediction objective. Here n is a freely chosen parameter, and after some experimentation I settled on the choice $n = 20$, although given more time a more principled approach to the selection of this parameter may be merited.

- For volatility, I take an approach similar to return, using the quantity used to measure volatility when computing the Sharpe ratio as the volatility prediction. More specifically, I use $V_n := \text{vola}_{\text{daily}} := \text{StDev}(\{d_1, ..., d_n\})$ as the one-parameter family of objective functions (where d_j is defined the same as it was for the Sharpe ratio). Also, I again made the (unprincipled) parameter choice of $n = 20$.

Formulation of the problems

Let x_{ik} represent the feature vector on day i for futures contract k, and let $R_n(i,k)$ be the value of R_n (the percent change in price n days in the future) for security k starting on day i, and define $V_n(i,k)$ similarly. To obtain return and volatility predictions, I applied a variety of machine learning algorithms to obtain prediction functions R^* and V^* which attempt to minimize the mean squared error, i.e., minimize the quantities

$$\sum_i \sum_k (R^*(x_{ik}) - R_n(i,k))^2, \text{ and}$$

$$\sum_i \sum_k (V^*(x_{ik}) - V_n(i,k))^2$$

respectively.

Features

As noted above, the raw feature vector is 2,520-dimensional, which is far too large for effective use of most machine learning algorithms (especially considering that I only have approximately 12,000 training examples). As a result, it is necessary to use a modified, lower-dimensional feature vector. I consider two distinct approaches to feature selection:

1. I applied PCA to a normalization of a 250-dimensional time-downsampled subvector of the original feature vector, obtained by taking the values of each of the five types of variable every 3 days for the past 150, and dividing it by the current day's value. I decided on the specific number of principal components to use by cross-validation (I go into more detail in the results section).

2. I also generated a vector of 18 standard technical indicators used by professional financial analysts (normalizing them appropriately for comparison between securities). The technical indicators I used consisted of the Average True Range (ATR) computed over 14 and 45 trading days; the ratio of the Exponential Moving Average over 12, 26, and 50 trading days to the Exponential Moving Average over 100 trading days; the On Balance Volume indicator over 5, 15, and 60 days; the Percentage Price Oscillator (PPO); the Percentage Volume Oscillator (PVO); the Rate of Change (ROC) over 5, 21, and 125 days; the ratio of the Simple Moving Average (SMA) over 100 days to the SMA over 200 days, the Relative Strength Index (RSI) over 14 days, and William %R over 14, 45, and 125 days. Definitions of these indicators may be found in [2].

Results

For both volatility and return prediction, I considered four regression algorithms: linear and regularized (ridge) linear regression (implemented myself); Neural Networks (using the MATLAB Neural Network toolbox); Random Forests (using the MATLAB TreeBagger function); and gradient-boosted decision trees (using the R package GBM).

As I show below, the volatility prediction problem was significantly more tractable than the problem of return prediction, so I began by performing feature selection on the volatility prediction problem, and used the results of that for the return prediction problem.

For both problems, I used a training set consisting of 9424 examples (80% of the data), and a test set consisting of 2356 examples (20% of the data). The training vs. test division was chronological, so that the training examples consisted of the initial 80% of the data, and the test examples consisted of the final 20% (it is important to segregate the data by time period since the performance of the various futures is not independent over the same time period). In addition, wherever I refer to cross-validation, I performed 10-fold cross-validation on the initial 80% of the data (the training data set), where each of the cross-validation folds was chosen to be a contiguous time block.

Volatility Prediction

Before comparing the performance of the various machine learning algorithms, I performed feature selection using linear regression. The results are contained in the following table:

Features	Training r^2	CV r^2	Test r^2
PCA (82)	0.631	0.490	0.473
TI (18)	0.710	0.645	0.637
TI (7)	0.713	0.647	0.639

Here PCA stands for principal components analysis, and TI for technical indicators. The number in

parentheses is the dimension of the feature vector.

After performing principal components analysis on the 250-dimensional subsampled and normalized feature vector, I obtained through cross-validation an optimal number of principal components of 82, but even after choosing this optimal number of principal components, linear regression using the 18-dimensional feature vector of all technical indicators performed much better, and greedy backward-elimination allowed me to obtain a subset of seven of the technical indicators which also gave some small improvements.

I interpreted the results of this feature selection procedure as signifying that the 18-dimensional feature vector of technical indicators had much greater explanatory power of the futures' behavior, and so decided to use these features when comparing the various machine learning algorithms' performance when predicting both volatility and return.

The following table contains the performance results for the various algorithms for volatility prediction, using only the seven features remaining after backwards-elimination during the variable selection process.

Algorithm	Training r^2	CV r^2	Test r^2
LR	0.713	0.647	0.639
NN	0.734	0.660	0.632
RF	0.731	0.664	0.649
GBM	0.701	0.666	0.638

Here LR refers to linear regression, NN to neural networks, RF to random forests, and GBM to the gradient boosted decision trees.

Note that although I only report the r^2 values, rather than the MSE, the two are related in such a way that a higher r^2 corresponds precisely to a smaller MSE, so that considering the r^2 value is sufficient to analyze algorithm performance. Considering both the performance on the cross-validation and the test data set, the Random Forest model seemed to offer the best performance, though this performance was only marginally better than the linear regression model.

Return Prediction

The following table contains the results of using the various algorithms to predict return performance using the 18 technical indicators as the feature vector:

Algorithm	Training r	CV r	Test r
RR	0.169	0.142	0.138
NN	0.088	0.013	-0.088
RF	0.236	0.069	-0.102
GBM	0.328	0.154	0.028

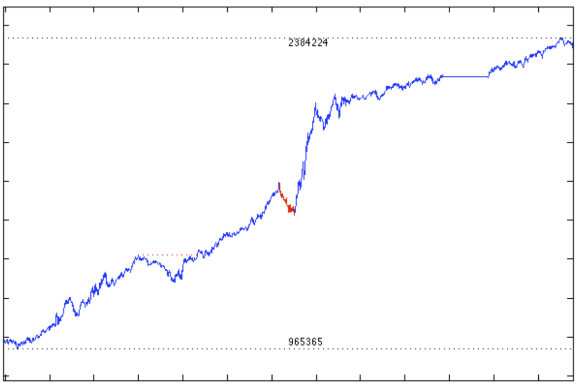

Figure 1: *Performance of grid-search optimized strategy.*

Here the algorithm acronyms represent the same things, except that RR here represents ridge regression, which offered some small improvements here over ordinary least squares linear regression.

Unsurprisingly, return prediction is much more challenging than volatility prediction – note that we have reported values of r, rather than r^2 in the table above. However, as I attempt to demonstrate later in the section on theoretical algorithm performance, even predictions with a relatively low value of r can generate surprisingly effective trading algorithms as long as there is some correlation.

Unfortunately, only gradient-boosted decision trees performed comparably to ridge regression in cross-validation, and none of the algorithms performed close to as well as ridge regression on the test set, with Random Forests and Neural Networks in fact performing worse than using the mean for prediction. As a result, I used the ridge regression predictions in my trading algorithms, but I still believe there is significant potential for improvements in these predictions, and I think such improvements are one of the primary ways that I could improve my trading algorithm performance.

Trading Algorithms

Description of algorithms

To make use of the volatility and return predictions, I created a 3-parameter trading algorithm which weights the predicted returns by a power of the predicted volatility, and uses this as the desired portfolio distribution. The chart below plots the performance of this strategy for one specific choice of these three parameters, chosen by grid search to obtain the optimal Sharpe ratio.

Figure 1 shows the performance of this optimal

strategy, which obtained a yearly return of 6.63% and a yearly volatility of 5.58%, for a Sharpe ratio of 1.18. To demonstrate the effectiveness of this approach to strategy selection, I performed a similar selection procedure on only the first half of the data, and was able to obtain a strategy obtaining a Sharpe ratio of 1.33 on the first half of the data, and this strategy obtained a Sharpe ratio of 0.73 over the second half. This shows that we should not expect the strategy from Figure 1 to perform equally well in the future, but that it should still work to some extent.

Theoretical algorithm performance

I now consider the performance of an extremely basic trading strategy using as input only corrupted versions of the predicted returns, with noise added to obtain a specified correlation coefficient r. Specifically, the strategy just takes the predicted returns and uses an appropriate scaling as the desired equity distribution (noting that negative numbers correspond to "selling short"). For each choice of r, thirty separate instances of the random predictions were generated, and the mean and standard deviations of the Sharpe ratio of the resulting strategies are included in the table.

r	0	0.1	0.15	0.2	0.3	0.5
Mean	-0.36	0.09	0.41	0.97	1.90	3.62
SD	0.26	0.48	0.44	0.33	0.29	0.31

This table demonstrates that even a small value of r^2 is enough to obtain a good trading strategy, with $r^2 = 0.1$ ($r \approx 0.3$) being good enough for this very basic strategy to be very effective. As a result, it seems that an appropriately chosen strategy (which also takes volatility into account) could reasonably reliably achieve a Sharpe ratio of 1 or greater with return predictions only slightly more accurate than those I obtained from ridge regression (which achieved $r \approx 0.14$ in cross-validation and on the test data).

Discussion and Future Work

According to [3], a Sharpe ratio of "1 or better is considered good, 2 or better is very good, and 3 or better is considered excellent." As a result, the strategies I obtained through my work on this project leave a bit to be desired since they have achieved a Sharpe ratio of at best about 1.2 on the backtesting data, and appear to perform less well on future data (though still obtaining some positive returns). While it seems that these strategies will still generate positive returns on future data, I would not expect them to obtain a Sharpe ratio greater than 1 with very high probability.

This is a somewhat disappointing conclusion, but there are several ways that I think the work could be improved with additional work that could lead to the generation of more effective strategies:

- More effort could be put into the choice of prediction objective – in particular, I have only investigated predictions for $n = 20$ days in advance. Other choices may lead to more reliable predictions.
- In both of the supervised learning problems, I have only considered models which predict future return and volatility for each security independently. However, the various securities seem to exhibit dependency over a fixed period of time. I think a model which incorporates such dependence might be able to generate more effective predictions, and is worth investigating.
- Recall that the feature vector of technical indicators already led to much improved predictions of volatility over the PCA approach to the raw data. Similarly, I suspect that another set of features could also lead to improved predictions of return, and I believe that further investigation into feature generation and selection could lead to improved return predictions (and therefore improved Sharpe ratios).
- For the sake of just obtaining a high-performing strategy (which may not be a legal submission to the Quantiacs competition), I believe that it would be very beneficial to include outside data that does not consist solely of past trading performance (for example, sentiment analysis of news articles related to the securities in question). In addition, using data from more than just the 27 futures under consideration in this competition to train the model could also lead to performance improvements.

References

[1] Quantiacs – Futures Contest Rules `https://quantiacs.com/Contest/FuturesContestRules.aspx`

[2] Technical Indicators and Overlays `http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators`

[3] Understanding The Sharpe Ratio `http://www.investopedia.com/articles/07/sharpe_ratio.asp`