Portfolio Optimization of Algorithmic Trading Strategies
Columbia University - Machine Learning in Finance
COMS W4995

Pierre Andurand
pa2570@columbia.edu

Ping-Feng Lin
pl2730@columbia.edu

Jack Yao
yy2857@columbia.edu

Abstract

A double optimization on a portfolio of Algorithmic Trading Strategies (ATS) is explored on a portfolio of futures contracts representing different assets. The ATS are chosen to be a mix of momentum based and oscillator based methods. The first optimization finds the optimal weights between ATS for each contract, which maximize the Sharpe ratio on training data, and subject to risk management constraints per contract. The second optimization then finds the optimal weights between contracts which maximize the Sharpe ratio on training data subject to portfolio risk management constraints. Two different methods of ATS parameters selection are also investigated through a third optimization procedure which should be run before the first. The results on testing data infer that this types of ATS portfolio optimization show good predictive power and robustness. Traditional Machine Learning methods such as AdaBoost and Random Forests are also explored and show promising results as well. A portfolio of the three methods output the best results. All methods studied are in the supervised training category. Code can be found in https://www.github.com/PierreAndurand2/MLFinance

1 Introduction

Algorithmic Trading has shown exponential growth over the last two decades, and there are many different categories in that space: High Frequency Trading, Mean Reversion, Trend Following, Statistical Arbitrage, Risk Parity to name a few. Algorithmic Trading is estimated to generate over eighty percent of trade orders in the Foreign Exchange Market according to [1].
Machine Learning is increasingly being used in algorithmic trading, and the processes are increasingly complex. For example, satellite images are now being analysed by machines to predict economic changes or the change in crude oil inventories. This allows the use of alternative data which was impossible before. Machine Learning can also use unstructured data such as text and images in order to deduce emerging patterns. Fields of Computer Science such as Deep Learning, Reinforcement Learning, and Natural Language Processing are increasingly used in Algorithmic Trading, with varying degrees of success. Quantitative Hedge Funds such as Renaissance Technologies or D.E. Shaw have shown that it is possible to generate very high risk-adjusted returns over long periods of time with the exclusive use of Algorithmic Trading. It is not an easy endeavour though, and those funds employ teams of hundreds of PhD researchers building factories of ATS.
The most basic Machine Learning methods in algorithmic trading would represent a given daily state of a contract as a vector of features containing past returns and technical indicators. This series of features vectors is then split into training and testing data. Supervised Machine Learning programs are then trained on training data in order to predict trends on testing data. Generally the horizon is fixed, for example 1 or 5 days, and the algorithm is built to predict a trend (classification task: up or down) rather than the actual returns (regression task). Reinforcement Learning does not have a fixed

horizon and can help decide when to get in or out of a position.
In this report, we focus on optimizing the weights of ATS per contract and between contracts, and then compare the results with typical Machine Learning methods such as AdaBoost and Random Forest on similar data. We aim to find the weights maximizing the Sharpe ratio subject to risk management constraints of the following return series in two steps:

$$R_t = \sum_{j=1}^{N} W_j \times \sum_{i=1}^{n} w_{j,i} R(ATS_{j,i})_t \textit{ s.t. risk management constraints} \quad (1)$$

where $R(ATS_{j,i})_t$ is the return of Algorithmic Trading Strategy number i for asset j at time t, and N the number of assets, and n the number of trading strategies. W_j is the weight of portfolio of asset j. $w_{j,i}$ is the weight of trading strategy i for asset j.

2 Data Preparation

Daily "last" prices of futures contracts are downloaded from Bloomberg. For past data, the last price will be the same as the settlement price. For today's data, it would be the last live daily price.
The futures contracts selected for this study are the next delivery contracts of: Brent Crude Oil (CO1), Copper (HG1), SP500 (SP1), Nasdaq (NQ1), 10-year Treasuries (TY1), and Gold (GC1). They represent very different assets with separate dynamics. We assume that the program is run a few minutes before the close to place the trades at the close (TAS: Trades at Settlement). We assume that this difference of a few minutes over time would not change the results significantly over time.
The data selected here is from start of 2000 to today, and we split it equally between training data and testing data. This choice is made so that the training period incorporates all types of markets: range bound, trending, and crisis. We select a long testing period in order to quickly test the robustness and longevity of the predictions.

3 Algorithm

3.1 ATS selection

The fixed-parameters ATS selected are a mix of momentum and oscillator based signals:

- Simple Moving Average with 20-day window (SMA(20)): Buy if last price is above SAM(20), otherwise Sell
- Simple Moving Average with 50-day window (SMA(50)): Buy if last price is above SAM(50), otherwise Sell
- Simple Moving Average with 200-day window (SMA(200)): Buy if last price is above SAM(200), otherwise Sell
- Moving Average Convergence Divergence, with SMA(8) and SMA(20): Buy if SMA(8) is above SAM(20), otherwise Sell
- Moving Average Convergence Divergence, with SMA(50) and SMA(200): Buy if SMA(50) is above SAM(200), otherwise Sell
- Bollinger Bands with 20-day lookback window and 2 standard deviations: Sell if last price above the upper band, Buy if below lower band
- RSI index with 14-day lookback window and thresholds 30 and 70: Sell if RSI(14) greater than 70, Buy if RSI(14) below 30

3.2 First Optimization

The ATS signals are generated from the daily last prices, are volatility weighted, and then transformed into a return series per ATS per contract (7x6=42 series). We choose to volatility weigh the signals pre-optimization in order to reduce leverage size automatically in times of crisis, and to keep a relatively stable level of risk over time. The signals series are therefore: {-1,+1} $\times \frac{\sigma_{target}}{\sigma_{20d-t}}$, where σ_{20d-t} is the

20-day historical volatility at step t, and σ_{target} the targeted volatility for that particular asset. Here we choose σ_{target}={10%, 20%, 30%} respectively for Treasuries, Equities, and Commodities.
The return series is generated by multiplying the signals vector by the contract return vector shifted by 1 day forward. For each contract, the return series for each ATS is fed into the first optimizer, and the optimal weights between ATS maximize the Sharpe ratio on the training data for this given contract, subject to risk management constraints. We choose to set the limit to the absolute sum of the signals per contract to 2, except for TY1 where the threshold is chosen to be 4. For each contract, we therefore get a weight vector of dimension 7, representing the weights for each ATS for a given contract. The optimizer used is optimize.minimize (of minus Sharpe) from Scipy, and the method used is SLSQP. It is an iterative method for constrained nonlinear optimization.

3.3 Second Optimization

The $w_{j,i}$ weights now being fixed, each contract now has a single return series, and those return series are input in a second optimizer, which maximize the Sharpe ratio over the training data, and subject to the sum of weights being equal to 1. The second optimizer used is the same as in the first optimization and outputs the weights W_j.
As we have now have the weights for each ATS per contract, and the weights between contracts, we can therefore run the predictions and output return series on test data, as well and cumulative pnl and return, annualized return, Sharpe ratio, volatility and maximum drawdown. The weights will be constant on test data, but there needs to be daily trading rebalancing to keep the weights constant every day.

3.4 ATS parameters optimization

We investigate if the results would improve if we optimize the ATS parameters before running the first optimization.

Grid search per contract and per strategy
For each ATS and contract, we run a grid search on the ATS parameters. For example, finding the optimal look-back period for the Long Term Simple Moving Average ATS for a given contract to maximize the Sharpe ratio. We get the following results:

	CO1	SP1	TY1	HG1	NQ1	GC1
LTMA	160	230	210	170	230	230
MTMA	63	100	42	80	63	45
STMA	25	18	25	21	19	15
SMACD	(5,22)	(9,25)	(7,16)	(9,29)	(4,29)	(11,28)
LMACD	(54,240)	(54,210)	(54,150)	(56,150)	(44,150)	(58,240)
RSI	(19,30)	(15,30)	(10,20)	(6,35)	(20,30)	(15,30)
BB	(11,1.9)	(10,1.1)	(31,1.9)	(10,1.9)	(28,1.9)	(15,1.9)

Table 1: ATS optimal parameters with grid search

Coordinate Gradient Descent This method helps optimize the ATS parameters as a portfolio, instead of for individual ATS. It was developed for its ability to solve K-means problems. The basic concept is to iteratively hold one set of parameters fixed, and optimize the other set. Every update of the parameter decreases the objective function compared to the previous value. Therefore, the objective function is monotonically decreasing.

The problem is formulated as follows:

- Select the types of technical indicators as the features to be learned. (In the experiment we picked: Simple Moving Average, MACD, Relative Strength Index, Bollinger Bands)
- Select a machine learning model to do regression/classification with buy-and-sell signals (In the experiment we choose Linear Regression)

- Optimize both the parameters of technical indicators and the parameters of the machine learning model

In our experiment, the machine learning model is Linear Regression, because of its relatively simple objective function:

$$L = \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

Its optimal *w* for a dataset is

$$w_{LS} = \left(X^T X\right)^{-1} X^T y$$

And by taking derivative with respect to variables which belongs to each feature generation function, we can obtain the gradient as the direction to find the optimal parameters for technical indicators. For example, assume the *i*-th data pair is in the form (x_i, y_i),
where

$$x_i = \left[x_{i,1}, x_{i,2}, ..., x_{i,d}\right],$$

$$x_{i,1} = SMA\left(\text{window\_size} = p, day = t_i\right) = \frac{1}{p}\sum_{t=1}^{p} c_{t_i - t + 1},$$

c_t is the close price on day *t*

The partial derivative w.r.t *p* gives us the gradient

$$g = \nabla_p L = \sum_{i=1}^{n} -2w_1 y_i \frac{dx_{i,1}}{dp} + 2w_0 w_1 \frac{dx_{i,1}}{dp} + 2w_1 \frac{dx_{i,1}}{dp} \sum_{l=1}^{d} w_l x_{i,l}$$

where

$w_{LS} = [w_0, w_1, w_2, ...w_d]$, w_0 is the weight for the constant.

Thus we can find the direction that minimizes the objective function and do the optimization by changing $x_{i,1}$ in the whole series with the new SMA parameter:

$$p' = p - \eta g$$

where η is the learning rate.

The performance and optimal parameters under this framework can be seen in the following table:

	CO1	SP1	TY1	HG1	NQ1	GC1
SMA	75	129	130	161	544	538
MACD	(732,44)	(348,437)	(264, 641)	(505,460)	(141,100)	(681,625)
RSI	1251	531	33	1043	636	114
BB	(2,2)	(2,487)	(134, 2)	(2, 243)	(2,1168)	(2, 85)

Table 2: ATS optimal parameters with LR coordinate descent

There is no guarantee that this would help our problem as we are not optimizing a Linear Regression, but we explore if there is some kind of information in such a process.

4 Random Forest and AdaBoost

The following features are used to define the daily vector of a given contract:

- (price-SMA(8))/SMA(8): How much above or below the SAM(8) the last price is in percentage
- (price-SMA(20))/SMA(20): How much above or below the SAM(20) the last price is in percentage

- (price-SMA(50))/SMA(50): How much above or below the SAM(50) the last price is in percentage
- (price-SMA(100))/SMA(100): How much above or below the SAM(100) the last price is in percentage
- (price-SMA(200))/SMA(200): How much above or below the SAM(200) the last price is in percentage
- RSI(14,30)/100
- 20-day historical volatility
- change in volatility between current 20-day volatility and the 20-day moving average of 20-day historical volatility

4.1 AdaBoost

AdaBoostClassifier (Adaptive Boosting) is an ML ensemble method which combines multiple weak classifiers to build one strong classifier. The base model is a decision tree. The weights of incorrectly classified data points are increased such that subsequent classifiers focus more on difficult cases. We choose AdaBoost with 100 estimators and default parameters from sklearn. We run cross validation with gridsearchCV on training data to choose the optimal depth of the tree (between 1 and 5) for the base model, and run the predictions with those parameters.
After fitting the model on training data (X=features matrix of rows representing days, and columns representing features; and Y= vector of next day's return sign), we forecast the next day's return sign on testing data, and output the return series being volatility weighted and cumulative pnl alongside key metrics such as total return, annualized return, volatility and Sharpe ratio on training and testing data.
Here we use the volatility weight defined by: $w_\sigma = 0.25/\sigma_{20d-t}$ in order to target a portfolio volatility between 15% and 22% depending on correlations between contracts.

We assume that we put the same amount of money in each asset at the beginning of the testing period and do not rebalance.

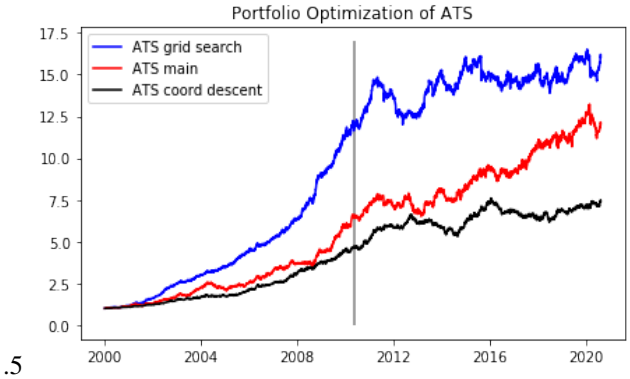
4.2 Random Forest

A random forest fits a number of decision tree classifiers on various sub-samples of the training dataset. It uses majority vote to classify, and generally improves the predictive accuracy and controls over-fitting. We repeat what we did in the previous subsection by replacing AdaBoost by RandomForest with 50 estimators, and run a cross validation on the depth of tree (between 1 and 6) on training data with gridsearchCV, and select the best model parameters for each contract to predict on test data.

5 Results

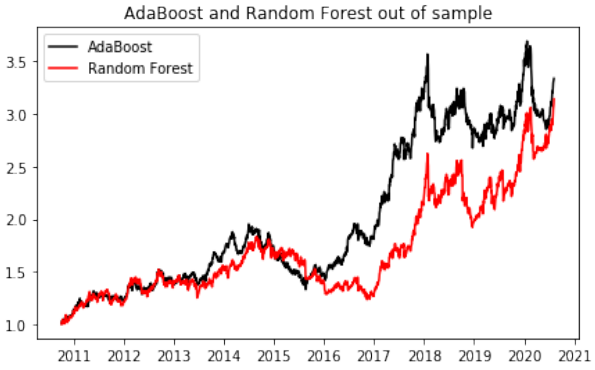
Methods	ATS P.O.	ATS P.O. +g.s.param opt	ATS P.O. +coord. des. opt	AdaBoost	Random Forest	model	benchmark
Sharpe OS	0.57	0.32	0.57	0.82	0.7	0.95	0.25
Sharpe IS	1.84	3.22	2.13	64.08	2.92		
Ann. ret OS	6.04%	2.9%	4.46%	15.87%	12.34%	9.6%	3.1%
Ann. ret IS	19.6%	26.6%	15.9%	1304%	49.78%		
Volatility OS	10.5%	8.9%	7.9%	15.87%	17.59%	10.12%	12.5%
Volatility IS	10.7%	8.3%	7.4%	20.35%	17.02%		

Table 3: Risk and return metrics in and out of sample 50% test data



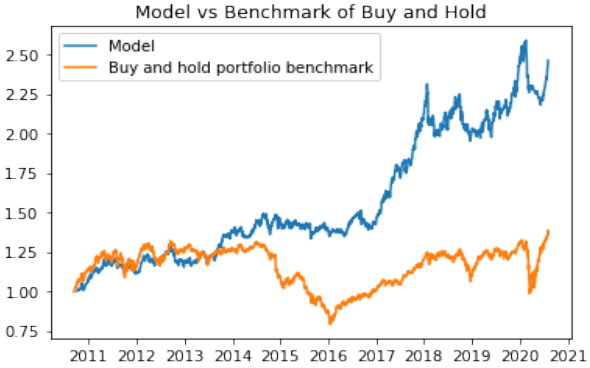
.5

Figure 1: ATS portfolio optimization



.5

Figure 2: AdaBoost and RandomForest



[h]

Figure 3: Average 25% RF, 25% AB, 50% ATS P.O. vs Benchmark

The model is a portfolio of methods composed of 50% in ATS P.O., 25% in AdaBoost , and 25% in Random Forest. The benchmark is defined as a Buy and Hold strategy on a portfolio of equally weighted futures, and rebalanced daily.

The ATS portfolio optimization without ATS parameter optimization works better than with out of sample, as parameter optimization seems to over fit to training data. The coordinate descent optimization is mixed with ATS parameter grid search for the technical indicators we cannot get with the coordinate descent. For example, in the way we coded coordinate descent, we can only get the best overall SMA, but not the best short term, medium term and long term. The coordinate descent optimization seems to have good results, but we are worried that it might not always work and could lose some logic.

The grid search parameter optimization over fits to training data. We therefore decide to select the ATS portfolio optimization without parameter optimization out of the three, as it does the best on

testing data, as can be seen visually and looking at the Sharpe ratio. It is also the one where the Sharpe goes down by the least between training data and testing data.
In the poster, we tested Random Forests and AdaBoost with the sklearn default parameters, and concluded then that the ATS portfolio optimization was bringing better results. Once we do a crossvalidation gridsearchCV to find the optimal depth of trees on training data, and also use a volatility weighting on the trading signals, the results are more encouraging for AdaBoost and Random Forest. AdaBoost gets a 0.82 Sharpe ratio out of sample, and Random Forest a 0.7.
As the three methods do not have a correlation equal to 1, averaging them brings even better results. For example 25% in Random Forests, combined with 25% in AdaBoost, and 50% in the ATS portfolio optimization brings a Sharpe of 0.95 over 10.5 years of testing data. This is very highly statistically significant. The volatility of this average is lower than the lowest volatility of the methods. It is also interesting to notice that there is very little difference between volatilities between training and testing period for every method. The model of the portfolio composed of 25% in AdaBoost, and 50% in the ATS portfolio optimization performs much better than the benchmark of a buy and hold portfolio equally weighted and rebalanced daily.

Below are the summary metrics if we use 60%, 75%, and 85% training data.

Methods	ATS P.O.	AdaBoost	Random Forest	model	benchmark
Sharpe OS	0.71	0.34	0.7	0.9	0.09
Sharpe IS	1.77	3.99	5.46		
Ann. ret OS	5.81%	5.55%	12.23%	8.11%	1.09%
Ann. ret IS	14.33%	75.13%	120%		
Volatility OS	8.15%	16.24%	17.43%	9.04%	12.23%
Volatility IS	8.08%	18.83%	22%		

Table 4: Risk and return metrics in and out of sample 40% test data

Methods	ATS P.O.	AdaBoost	Random Forest	model	benchmark
Sharpe OS	0.82	1.3	1.32	1.64	0.52
Sharpe IS	1.63	8.45	4.13		
Ann. ret OS	7.6%	18.78%	20.3%	14.02%	6.97%
Ann. ret IS	14.6%	208%	88.62%		
Volatility OS	9.3%	14.5%	15.38%	8.57%	13.39%
Volatility IS	8.97%	24.62%	21.47%		

Table 5: Risk and return metrics in and out of sample 25% test data

Methods	ATS P.O.	AdaBoost	Random Forest	model	benchmark
Sharpe OS	0.74	0.85	1.7	1.48	0.52
Sharpe IS	1.64	9.82	1.43		
Ann. ret OS	7.39%	11.41%	25.44%	12.66%	7.53%
Ann. ret IS	15.71%	277.8%	23.78%		
Volatility OS	9.94%	13.37%	14.93%	8.56%	14.57%
Volatility IS	9.56%	25.22%	16.64%		

Table 6: Risk and return metrics in and out of sample 15% test data

The results of the model portfolio are very good on all the testing periods, and much better than the benchmark.

6 System

As well as running backtests, the system can output historical leverage per contract, and historical trades per contracts, as well as trades to execute by the close on a given day. It asks the input of a selection of futures contracts to be downloaded from Bloomberg, the start of training data, and end of testing data, and the percentage of training data. It also asks for lot size and total amount of money to be managed in order to calculate historical trades and trades to be executed.

7 Possible next steps

Transactions costs are around 4$/lot for futures, so they represent a cost of around 0.1% to the portfolio every year, which we can neglect. We suggest to run the backtests on an expanding training window to be more accurate. This is something that remains to be developed. We suggest getting the trade orders keeping a 15% testing period in order to be able to check that the backtests have good results. We should run tests for other portfolios of futures, and add more Technical Indicators for ATS and features.

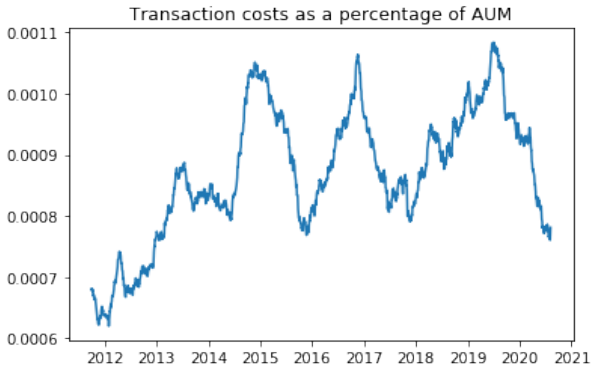


Figure 4: Transaction costs as a percentage of AUM

8 Conclusion

In this report, we compared ATS portfolio optimization methods with traditional Machine Learning methods such as AdaBoost and Random Forests on a selection of futures contracts and over long training and testing periods. Combining the three methods brought the best results with a Sharpe ratio of 0.95 over 10.5 years out of sample. Spending more time selecting ATS and features more carefully, and potentially training on an expanding window could potentially even better the results. As the strategies do not have a high leverage and the contracts are only traded once daily maximum, the transaction costs can be neglected. This method has much better results than a buy and hold benchmark.

9 Note

We hope to have a grade in excess of 20/22 for this project, so a total of 23 to 25 on 25. We managed to download data dynamically from Bloomberg, try different ML techniques and portfolio optimisation of ATS, and get exciting results which could generate large profits. The team members contributed in the following proportions: Pierre: 50%, Ping-Feng 30%, and Jack 20%.

10 Bibliography

[1] https://link.springer.com/chapter/10.1007