Building an Investment Strategy for the Financial Markets

Jinqiu Liu (LaGuardia Researcher)
Mentor: Dr. Omar Ait Hellal
JQL1988@Gmail.com

Introduction

Abstract

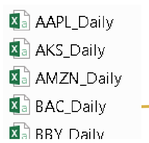
This research project is a Computer Science project focused on the Stock Market, which heavily uses the knowledge from Java programming, statistics and Data Structures. The goal of the project is to implement an algorithm that filters out "noise" in the stock market and determines key inflexion points. We create a logical model that uses these points to detect certain patterns in the stock market. We refine the model using various parameters to obtain higher winning percentage and profit, hence, better results. We simulate a hypothetical account invested using our approach and show that our trading strategy outperforms the S&P500 benchmark every year. In the real world, such model can give suggestions for people to enter a likely winning stock trade when a specific pattern is detected in real time.

Objective

We want to design an investment strategy that is based on a specific trading pattern, which would outperform the S&P500 benchmark while keeping the drawdown to the minimum (smooth).

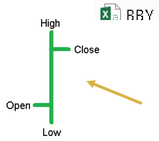
Data

The Data that we run our simulation on is a combination of popular Stocks, Commodities and Indices. (76 total)



These files contains the historical bar prices from the year 2000 to the end of 2014.

Date	Open	High	Low	Close
1/3/2000	50.25	50.25	48.00	48.44
1/4/2000	47.75	47.94	44.94	45.56
1/5/2000	45.06	46.44	44.50	46.06
1/6/2000	46.94	50.00	46.75	50.00
1/7/2000	49.62	49.62	48.00	48.69



A typical "bar" contains the key information of one day's trade. The open price, closing price, day's highest and lowest price.

Trading Long and Short

In order to take full advantage of the market, we will perform both Long and Short trades.

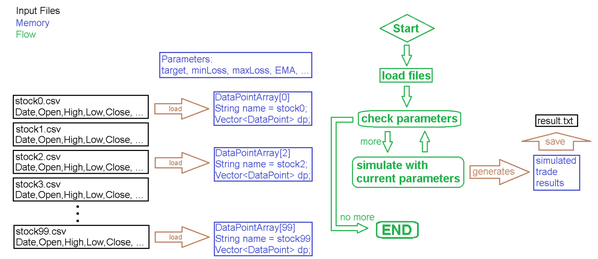
A long trade is buying at a lower price, expecting the price to go higher to sell the stock. The difference of the two prices becomes the profit.

A short trade is first "borrowing" stocks from others, selling them immediately at a higher price, expecting the price to fall. When the price is lower, we buy them back to return the stocks to the original owner, making a profit in between.

Methodology

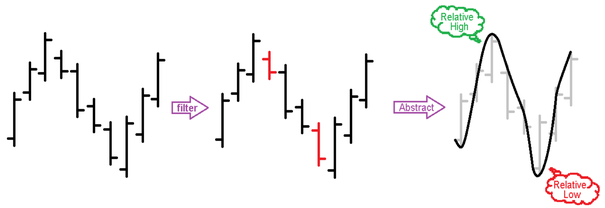
General Program Flow

Our program loops though key parameters (risk percent, target, minimum/maximum loss, etc.), and for each parameter it loops though all the different models (patterns) we have designed, to find the best model and the best parameters for it.



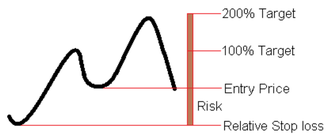
Key algorithm: filter

The filter method is meant to reducing the "noises" in the market, hence, getting rid of the less useful information. As illustrated below, the algorithm would delete bars with less information and then mark some key points with certain flags. The purpose of this is to detect the patterns much easier.



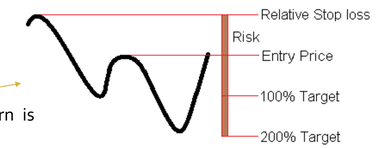
Key algorithm: detect pattern

The detect pattern method varies between the different models, yet its purpose is the same: to find a specific pattern inside the market; when it occurs, the method will trigger a trade. Below is an example of a pattern that we can look for.



The long trade pattern

Whenever this specific pattern is detected, a trade will be entered at open price the next day, waiting to reach either the Target (which would result in a profit) or Stop loss (which would prevent further losses).



The short trade pattern is the exact opposite.

The idea of the pattern is to consider the bigger trend over the current price move. In the example of the long trade model, we can see that the relative highs are getting higher, and the relative lows are also getting higher. However, the current price is dropping, lower than the previous relative low. We would like to believe that this is a sign of a temporary pullback, which is a good time to enter the market for a long trade, expecting the price to go higher.

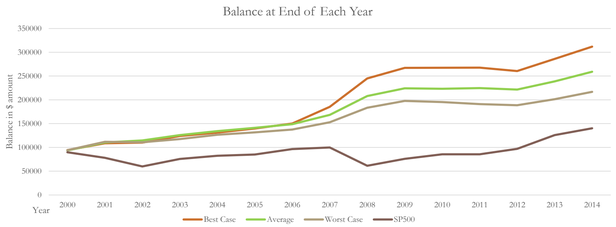
Conclusions

Benchmarking

The final step of this project is to simulate the trading strategy: using the data over 14 years, with $100,000 to start with. We decided create a finite number of "trading slots" available, which limits the amount of money invested per trade to be *total balance* / *# of slots available*. Doing so also limits the maximum loss per trade to a fraction of the account balance; in general, it is preferable not risk more than 2% of the total balance in a single trade. In the scenario that trading slots are not enough to cover all possible trades, the trades will be selected randomly. We run the simulation 10,000 times to cover this random factor, which would include a best case, worse case and average case in the conclusion.

Best Results

The best results we have obtained is by having a maximum risk of 20%, 20 trade slots and target as 200%. It outperforms the S&P500 benchmark from every prospective we have accounted for.



Total Balance	Our model	S&P500
Average return:	**6.79**	**4.07**
Standard Deviation:	**7.53**	**18.77**
Max drawdown:	**11.08**	**55.31**
Sharpe Ratio:	**0.61**	**0.1**

We have developed a flexible software that allows us to test multiple patterns with varying parameters. This helped us create a successful trading strategy. We showed that our model yields not only a higher profit return but also a lower risk compared to the S&P500 benchmark. Furthermore, the Sharpe ratio for our system against 90-day T-Bills is 0.61, six times that of the S&P500.

Acknowledgements

Dr. Jorge Gonzalez, ME Professor, PI of CILES grant, CCNY
Dr. Yasser Hassebo, EE Professor, LAGCC

Sponsors

LaGuardia Community College (LAGCC)