Evaluation of Fitness Functions for Evolved Stock Market Forecasting

J. F. Nicholls, Prof. A. P. Engelbrecht*, and K Malan

Department of Computer Science,
School of Information Technology,
University of Pretoria,
Pretoria, 0002, South Africa
**E-mail: engel@cs.up.ac.za*
cirg.cs.up.ac.za

J. F. Nicholls

Suite 402, Private Bag x 10, Elarduspark
Pretoria, 0047, South Africa
E-mail: nicholls.jason@gmail.com

This article investigates the impact of three different fitness functions used when evolving a simple genetic algorithm to forecast the market. The genetic algorithm evolves the selection of technical analysis functions and parameters through the Darwinian process of natural selection. Those programs with the highest fitness value are evolved. Results showed a significant difference in performance when using an acumulative return of investment fitness function compared to a final return on investment fitness function or a fitness function that penalizes complexity.

Keywords: Evolutionary Computing; Fitness Function; Market; Forecast; Technical.

1. Introduction

2. System Design

A simple genetic algorithm evolves an individual that has a selection of technical analysis functions and parameters. This individual uses the evolved functions and paramerers, as well as the current stock data to generate BUY, SELL and HOLD signals. These signals are executed by a virtual banker. The result over time is net cash value. The objective is to maximize return on investment.

The process of evolution requires natural selection. Selection is done by one of three fitness functions (discused later). Those individuals in a given generation that obtain the highest value from the fitness function are used to create a new generation. This process continues for a fixed number of epochs. An elite individual is selected from the final generation by vitue of the fitness function. This individual is then tested against data it has never seen to determine the effectiveness of the individual in comparision to other individuals that implemented different fitness functions. The following sections outline the system design.

2.1. *Technical Functions*

Node	Name	Function
0	BB	$+BB(n) = SMA(n) + MD(n)$ $-BB(n) = SMA(n) - MD(n)$
1	SMA	$SMA(n) = \frac{P_t + P_{t-1} + \cdots + P_{t-n+1}}{n}$
2	EMA	$\alpha = 1 - \frac{2}{n+1}$ $EMA(n) = \frac{\alpha^n P_t + \alpha^{n-1} P_{t-1} + \cdots + \alpha^2 P_{t-n+2} + \alpha P_{t-n+1}}{\alpha^n + \alpha^{n-1} + \cdots + \alpha^2 + \alpha}$
3	WMA	$WMA(n) = \frac{\omega P_t + (\omega - 1) P_{t-1} + \cdots + 2P_{t-n+2} + P_{t-n+1}}{\omega + (\omega - 1) + \cdots + 2 + 1}$
4	RSI	$TG = \sum_{t=1}^{n} (P_{t+1} - P_t)$ for all t where $P_{t+1} - P_t > 0$ $TL = \sum_{t=1}^{n} (P_t - P_{t+1})$ for all t where $P_t - P_{t+1} > 0$ $RSI(n) = 100 - \frac{100}{1 + \frac{TG/n}{TL/n}}$
5	MFI	$MF = P_t * V_t$ $MFI(t) = 100 * \frac{+MF}{+MF + -MF}$
6	K-Line	$KLine(t) = \frac{P_t - lowest}{heighest - lowest} * 100$
7	D-Line	$DLine(n) = SMA$ of $KLine$ $DLine(n) = \frac{KLine_t + KLine_{t-1} + \cdots + KLine_{t-n+1}}{n}$
8	OBV	$OBV_t = OBV_{t-1} + \begin{cases} volume & \text{if } Price_t > Price_{t-1} \\ 0 & \text{if } Price_t = Price_{t-1} \\ -volume & \text{if } Price_t < Price_{t-1} \end{cases}$
9	ADI	$CLV(t) = \frac{(closePrice - lowPrice) - (highPrice - closePrice)}{(highPrice - lowPrice)}$ $ADI(t) = ADI_{t-1} + V_t x CLV_t$
10	ROC	$ROC(n) = \frac{P_t - P_{t-n}}{P_{t-n}} * 100$
11	CCI	$\theta = \frac{\sum_{t=1}^{n} \lvert SMA_t - typicalPrice_t \rvert}{n}$ $CCI(t) = \frac{1}{0.015} \frac{typicalPrice_t - SMA(n)}{\theta}$
12	MACD	$MACD(t) = EMA(12) - EMA(26)$

Where t is the day of trade, P is the closing price of the day, n is the number of days back. t is the current period. V is the Volume. MD is the mean deviation, BB, SMA, EMA, WMA, RSI, MFI, K-Line, D-Line, OBV, ADI, ROC, CCI and MACD are technical functions known as Bolinger Bands, Simple Moving Average, Exponential Moving Average, Weighted Moving Average, Relative Strength Index, Money FLow Index, Fast Stochastic Oscillator, Slow Stochastic Oscillator, On Balance Volume, Accumulative Distribution Index, Rate of Change, Commodity Channel Index and Moving Average Convergence or Divergence respectivly.

BB produce two bounds if the closing price breaks the upper bound a sell signal is generated if it breaks the lower bound a buy signal is generated.

SMA, WMA and EMA use the *limit* variable to determine agressiveness. If the value is greater than 50 an OR operator is used else an AND operator. The *longValue* and *shortValue* is used as n to get a long and short SMA, WMA or EMA. If the short result is greater than the closing price a SELL signal is generated, if it is less than a BUY signal is generated. If the short result is greater than the long result a BUY signal is generated, if it is less than a SELL signal is generated. Depending on the agressiveness of the function the two signals or ORed or ANDed together resulting in the final BUY or SELL for the given function.

RSI, MFI, K-Line, D-Line, ROC and CCI use the *value* variable as n. If the result is greater than the *upper* limit then a SELL signal is generated if it is less a BUY signal is generated.

OBV and ADI us *value* variable as n and are compared to 0. If the result is less than 0 a BUY is signaled if it is greater than 0 a SELL is signaled.

2.2. *Encoding*

As with nature, so too much the system have a form of DNA. The DNA in this case is the encoding of the technical analysis functions, weighting and its parameters. This encoding is known at the genotypic information. Each individual is made up of an array of nodes. Each node is linked to a specific technical analysis function. A node has additional variables used as input to the function that the node is set as. The variables are *longValue*, *shortValue*, *upperLimit*, *lowerLimit*, *limit*, *value*, *buyWeight*, *sellWeight* and *enabled.* The variables are all integer based with the exception of enabled. All functions or nodes generate a BUY, SELL or HOLD result. The results are doubles and are denoted as -1.0 (SELL), 0.0 (HOLD) and 1.0 (BUY). The signals are multiplied by the weight of the node. BUY and SELL each has its own weight determined through natural selection. If a node is disabled the node function will result in a 0.0 or HOLD. The final action of a given individual is the sum of the actions of the nodes. If the sum is greater than 1 a BUY signal is generated if the result is less than -1 a SELL signal is generated else a HOLD signal is generated.

The limit variables (*upperLimit*, *lowerLimit* and *limit*) are integers between 0 and 99 inclusive. Limits are used to determine BUY, SELL and

HOLD signals for a given function. For example if the RSI value is greater than the *upperLimit* than sell else if it is is less than the *lowerLimit* buy.

The value variables (*lowerValue*, *upperValue* and *value*) may be any multiple of 10 between 10 and 200 inclusive or the alue of 5. The value variables denote time. For example in the calculation of SMA the long SMA is calculated using the *longValue* and the short SMA the *shortValue*

The weights (*buyWeight* and *sellWeight*) may be an integer between 0 and 4 (inclusive).

2.3. *Genetic Operations*

Simulated evolution with not be evolution if it did not have genetic operations. The genetic algorithm in question implemented four genetic operations: Elitism, Creationism, Crossover and Mutation. Elitism is the selection of the best individual within a generation and transfering that individual un-altered to the next generation. Creationism introduced a specific number of new individuals into the next generation. Crossover selects a set of parents. These parents are the top x best performing individuals. CrossOver then selects two parents from this set. One parent is designated male and the other female. Each of the nodes in the array is addressed and a probability factor is selected. If this factor determines that crossover should occur for a given function or node a random number is selected. The number equates to a parameter within that specific node. The male and female then swap the values allocated to that parameter. The male and female then become new individuals in the next generation. Mutation is the final operator and acts on all the new individuals created through crossover. A mutation probability is selected to determine if an individual must undergo mutation, if selected a random number will select which of the functions to mutate. The process is repeated for a specific parameter. The parameter of a function is the altered based on its type. Integers become new integers and booleans are switched.

2.4. *Evolutionary Process*

Initialize the first generation, rate each individual against the fitness function, find and save the elite. Select the parents. Create new children using creation. Create new children using crossover, mutate the new children using mutation. Store all the children and the elite as new individuals of the next generation. Test each of the individuals against the fitness function. Repeat the process untill a set amount of generations has passed.

2.5. *Fitness Functions*

The purpose of this paper is to test the outcome of three different aproaches to a fitness function when implementing simulated evolution of stock market traders. Ghandar er al implemented return on investment (RIO) as a fitness function. Their fitness function then penalized the agent for certain errors. Their implementation used two such penalties, Ockham's razor and portfolio loss. For the pupose of this paper the ROI will be calculated in the same way using the following formula:

$$ROI = \frac{ln(V_{t_1}) - ln(V_{t_0}}{t_1 - t_0} \tag{1}$$

A penalty for complexity known as Ockham's razor will be subtracted from the ROI. Each node that is enabled will reduce the fitness value by 0.001. Schoreels et al used the area under the individual's total asset value graph. To implement this, the value of the shares and cash will be added up for each iteration of trading. The final fitness test is a standard ROI. It will be calculated as the final asset value at the end of trade.

For each experiment a fitness function is selected and all individuals are rated against the fitness function. The higher the value the better the individual has faired.

3. Historical Data

The last few months saw high volatility on the South African stock market. For this reason a number of shares were selected that experienced a different market conditions. During the first half of 2008 banking shares experienced negative trades, a highly bear market fueled by higher interest rates and the subprime crisis of 2007. As traders moved money out of banking shares and into resources, resource companies such as BHP Billiton and Anglo experienced record highs. At the oil price increased throughout the first half of 2008 it fueled a resource super cycle a very boyant bull market. While banks fell and resources climbed, the interest rate took effect other parts of the market such as construction and retail saw sidewards movement. For these reasons the following JSE top 40 listed companies were selected: Anglo (AGL), ABSA (ASA), BHP Billiton (BIL), Gold Fields (GFI), MTN (MTN), Murray and Roberts (MUR), Remgro (REM), Standard Bank (SBK) and SASOL (SOL). In additional to the blue chip companies selected the Satrix 40 Index (STX40) was also selected.

Each Genetic Algorithm was trained on data from the 3rd of April 2007 to the 22nd of January 2008 (200 trading days). Testing was done on data from the 22nd of January 2008 to the 18th of June 2008 (100 trading days).

4. Results and Analysis

For comparision 50 individual populations were evolved using 50 unique random seeds. This was done for each fitness function. Each elite individual was run against an out of sample test set. The final cash value was used as a benchmark for testing. The cash value was calculated using an initial cash value, given the current market data, the individual is asked for an action, if the action is buy, the maximum number of shares are purchased, if the action is sell all the shares are sold.Nothing is done if the action is hold. At the end of trade all shares are automatically sold. The cash value is the amount remaining. Each purchase and sale incures the following charges: ¡insert charges¿ The process was repeated for all stocks within the historical data set. For reference the table below includes the net cash value should the individual have bought the stock on the first day and held it until the end of the trading period.

Results

Stock	Function 1 (Mean)	Function 2 (Mean)	Function 3 (Mean)	Buy & Hold	Kruskal-Wallis (p-value)
AGL	119,500	130,000	120,500	136,373.48	0.002446
ASA	99,160	92,220	99,420	82,654.63	$1.625\text{x}10^{-7}$
BIL	142,100	148,700	146,700	160,094.48	$2.729\text{x}10^{-7}$
GFI	94,930	92,020	95,370	77,787.74	0.1673
MTN	121,200	122,100	119,300	120,294.12	0.621
MUR	99,770	103,800	100,900	100,449.12	0.5897
REM	107,700	107,500	107,400	113,929.46	0.004269
RCH	116,400	119,900	118,600	121,958.40	0.9674
SBK	88,640	87,400	87,750	82,539.28	0.04025
SOL	128,000	136,200	129,700	146,443.29	0.4503
STX40	111,500	117,500	112,600	126,337.72	0.005445

Statistical Analysis using Kruskal-Wallis Compared all three fitness functions to one another to find a significant difference. The fitness functions were then individually compared to one another using the Mann and Whitney U test.

As a market may move sideways and then spike over a period, it is understandable that the area under the market value makes a better fitness function that the total return on investment. A total return on investment may lead to an individual that is designed to take advantage of any spikes in the training data, this will lead to function fitting and not the generality required for continous succesfull prediction within a market. What is surprising is that implementing Ockham's Razor yields a partially successful result. This could be that the penalty for a given stock is too low thus not making an impact as in the case of, or the penalty is too great as in the case of. But appears to work in

5. Conclusion