Evolutionary Signal Processing

Senior Project Report

Financial Time Series Predictions

Author: David D. Brown
Adviser: Dr. Shane Cotter

March 30, 2010
Union College Senior Project Report

Evolutionary Signal Processing of Financial Time Series Predictions

Dave Brown

Advisor: Dr. Shane Cotter

1 The Problem

2 Background
   2.1 Stock Market Assumptions
   2.2 Signal Processing
   2.3 Evolutionary Algorithms

3 The Method
   3.1 Design Considerations
      3.1.1 Alternatives
      3.1.2 External Constraints
   3.2 Program Flow
   3.3 Evolutionary Considerations
      3.3.1 Genotype
      3.3.2 Population
      3.3.3 Breeding
   3.4 Signal
   3.5 The Inputs
      3.5.1 Fundamental Inputs
      3.5.2 Technical Inputs
      3.5.3 Other Inputs
   3.6 Filters: A Function Approximation Engine
      3.6.1 Pre-processing
      3.6.2 Filters
Chapter 1

The Problem

Assume there exists information in past data that relates to future trends. This is the case with our physical world. It allows physics models to predict exactly how fast an apple will fall. With historical data in the physical world, such as the apple has been $8m$ above the surface for a long time and that the apple’s stem breaks at $t = 0$, it is possible to predict where the apple will be in the future, at $t > 0$.

If this assumption is made for financial data, the resulting trend data would be extremely valuable. However there is a great volume of past data and much more complex and transient rules; in short decoding the limited information is difficult. A software signal processing system is proposed to attempt to decode trend information from historical financial data.
Chapter 2

Background

2.1 Stock Market Assumptions

Efficient Market Hypothesis

Is the assumption reasonable that there exists information in past financial data that relates to future financial trends? Stocks, though they are actually an ownership in a company, can be thought of as a commodity; thus the principles of supply and demand can be applied to their pricing. When stocks are traded in large volume on a major exchange it is reasonable to assume they operate in an efficient market. This led to the formation of the efficient market hypothesis in the 1960s [1]. This hypothesis states that financial markets are informationally efficient and prices on assets already reflect all known information and change instantly to new information.

Efficient Market Implications

If you accept the efficient market hypothesis then you can only consistently outperform the market through dumb luck. Stocks merely take random walks around their intrinsic value as various agents in the market adjust their valuations [2]. Of course there are some exceptions. A private equity firm, like Cerberus Capital Management who took Chrysler private, or majority shareholders should be able to create additional value that smaller shareholders cannot as they must accept the current management. Additionally, someone may have inside knowledge on the market. While it would be illegal for someone with inside knowledge of a particular company or even a future analyst report on a company to use that information for profit, someone could have inside information on their own trades. Imagine if Warren Buffet leaked information that he intended to buy a certain company. The market would
react instantly according to the efficient market hypothesis, but if Warren Buffet already held some of the company and then sold it as it rose on this news, then it would be called a pump and dump. It is unfortunately not a rare act on Wall Street [3].

Trading Strategies

The efficient market hypothesis proposes that all information within past data has already been acted on. Fortunately there is evidence that the efficient market hypothesis is not a universal law. Throughout the history of the stock exchange there have been two traditional trading strategies, fundamental analysis and technical analysis [4]. Fundamental analysis is concerned with predicting future stock price based on the past performance of the business: its management, and its credit risk. Technical analysis is concerned only with the past price and volume of the stock, observing that history tends to repeat itself and there are patterns within this information. Both these techniques require that this information is not fully priced into the stock and both these techniques are widely practiced within the industry. So does the efficient market hypothesis mean that in the long run and on average we are doomed to trade in mediocrity? Of course not. It is, after all, still only a hypothesis. There is increasing data, especially in the area of behavioral economics, that the stock market is not perfectly efficient [1]. Thus it seems reasonable to assume that there is information in past data for financial markets.

2.2 Signal Processing

A signal is information. Typically we think of this as a voltage signal or current signal, but in its most basic form a signal is a value whose modulation encodes information. We use signals everyday. Your TV remote sends a digital signal over the IR channel that codes your commands to the TV. That TV is connected to speakers which convert a voltage signal sent over a copper channel into pressure waves sent through the air which are converted back into electrical signals that travel along a channel of nerves to the brain. The processing of signals is really the conversion of the signal from one form to another, you cannot add additional information but you can remove extraneous information and bring forward the most pertinent information. Signal related to stocks like price and volume, fundamentals like earnings and forecasted earnings, or news articles can be processed. One signal processing application would be to condense the information of multiple signals
into a prediction of stock price. If the historical financial data is considered
as signals, then the prediction could use the techniques of signal processing.
The signal processing field involves both analogue and digital methods.

2.3 Evolutionary Algorithms

Evolutionary algorithms, or EAs, are a subset of derivative-free optimization
algorithms. The term derivative-free means that they do not require inform-
ation on dependencies, which is helpful in complex real-life systems where
dependencies are not always clear. EAs have other advantages as well. They
lend themselves naturally to parallelization to speed up their runtime, can
solve continuous as well as discrete optimization problems, and are less likely
than other derivative free approaches to get stuck in local minima [5].

The premise of EAs is to mimic nature. In the very general case a large
population of candidate solutions are created by seeding or random selection
and each member of the population is evaluated for its fitness for the final
application. Then a new generation is created using mating between high
fitness members. The mating algorithm differs between implementations, but
it combines the elements of the individual in some way. Often a small amount
of mutation of the elements is used during the mating process, which helps
avoid local minima. This emulates the evolution of life. In life the fitness is
a function of the ability to survive and the ability to attract a mate.

Genetic algorithms are a subset of EAs and the most familiar. A genetic
algorithm establishes an encoding of parameters into a binary word. This
word is called a chromosome. A large population of parameters is created by
seeding or random selection and each member of the population is evaluated
for fitness. Then a new generation is created using mating between high
fitness members. The mating algorithm differs between implementations, but
it combines the parameters in some way. Often a small amount of mutation
of the parameters is included during mating.

Evolutionary algorithms have been used with success in digital signal pro-
cessing applications. For example, recently Saranyan Vigraham showed that
“controllers evolved using evolvable hardware techniques were robust and
superior in quality than the traditional controllers” for correcting thermo-
aoustic instability in jet engines using lean fuel mixtures [6]. The evolved
controllers were able to limit oscillations to a greater degree than previous
controller designs.
Chapter 3
The Method

3.1 Design Considerations

The intent of the system is to predict the future values of stocks, more specifically the goal is produce accurate time series trend predictions of stock price. To achieve this the system processes financial signals such as price and volume using a complex filter created using an evolutionary algorithm. In effect, the problem is reduced to a search for the optimal filter topology and filter coefficients. According to the No Free Lunch Theorem over all search problems no one algorithm is superior, and there are cases where a random search performs better than an evolutionary method [7]. However, there are cases where a evolutionary method is superior to a random one; a genetic programming method provides an elegant way to search through the vast array of possible topologies.

3.1.1 Alternatives

Traditional Signal Processing

In order to design a traditional signal processing application the relationship of the inputs to the outputs must be known. When that correlation is unclear, the guess and check method is often used. This might work for a single parameter, such as the cutoff frequency of a low pass filter, but with a bunch of parameters, the sheer number of possible combinations will quickly overwhelm the method. Quantative analysis of financial data is often takes a long time and is very specific to a certain stock and certain time periods.
Genetic Algorithm

The evolutionary algorithm used to evolve the solution is a combination of a neural network and genetic programming. A much simpler evolutionary method is a genetic algorithm. There are a finite number of parameters that are encoded into a chromosome. This reduces the solution space and also simplifies mating. However, this simplified mating does not align as well with the idea of a cascaded filter. Mating discontiguous parts of the cascaded filter would offer little advantage over random selection.

Code Language

There were alternatives to using the Java language. Matlab is a mature technical computing environment, it has powerful signal processing abilities. However its use in conjunction with evolutionary methods was not as clear and it was discounted because of this. Python is another high level language with many established libraries. The author was not as familiar with Python as Java, and seeing no specific benefit over Java, Python was also discounted. The system requirements demanded a language that was able to achieve strong floating point performance. The Java Just-In-Time compiler has brought Java from a slow byte code language into a dynamically compiled native one and very close to the performance of C [8].

3.1.2 External Constraints

The design of the system was done in consideration of its external constraints and its impact. It is always important to take into account the economic, environmental, and ethical considerations while practicing engineering. These must be given certain implicit weight in the design of the system.

Economic

A requirement of the system is to run on available computer and network hardware. The computing resources available are limited to a few local computers with ethernet links. It will not create undue burden on limited college resources. If this method was combined with automatic trading it would be important to consider the stability of the whole financial market. However, all trades are matched between willing parties by the exchange and the use of computers in this manner is entirely legal.
Environmental

The system will have extremely limited impact on the environment. It takes advantage of hardware that is already in place for other reasons and likely idle if not used for the system. It occupies a virtual world.

Ethical

The system takes advantage of Java which is available from Sun Microsystems under a GPL license. Java includes a classpath exception so code created to run on java can be licensed under a license other than the GPL. The JFreeCharts library used in the GUI is under a LGP license. The remainder of the code was written by the author. Again the premise to use historical trends to predict future ones is an ethical undertaking.

The data the system processes is freely available from the Google Finance API for purely “informational purposes.” This is ethical for research, but for trading purposes a different source of information would have to be found. If the system were distributed as a product it would be important to convey a sense of uncertainty about the future, even if trends are discovered, it is still no guarantee of future results.

Political

The political impact of the system is extremely limited. It follows all US rules and regulations, particularly those set by the Securities and Exchange Commission. If the software system were used to produce recommendations the firm selling those recommendations would be required to register and show proper certification to the SEC. It would be important to consult a lawyer on the applicable regulations.

Health and Safety

The health and safety of the public is not affected by the system.

Sustainability

The software system can be operated indefinitely with minimal maintenance cost.

3.2 Program Flow

A high level flow of the program follows:
1. Initialize environment and starting populations
2. Calculate the fitness of each individual of a population
3. Breed the fitter members with a small mutation rate
4. Goto 2 until stopping criterion is met

Some evolutionary algorithms present a very natural stopping criteria. However, without fitness requirements or even knowledge of the valid range of fitness, a stopping criteria of a set time or a set number of generations is used. The advantage of a set number of generations has over a set amount of time is that the results will be independent of hardware.

### 3.3 Evolutionary Considerations

#### 3.3.1 Genotype

The genotype completely describes an individual of the population in a manner that allows breeding between different individuals. In biological studies the phenotype is a subset of the genotype that represents observable features of the organism. The phenotype is the filter types and their coefficients. It is best represented by a tree structure with each node a filter that takes one or two input signals and produces one output signal. The tree will use a max depth parameter, which will limit the maximum depth of the tree. A binary tree is used in order to limit the search space and simplify the coding of the multiple input filters. The non-observable part of the genotype is the parameters that affect breeding. While a biological system is usually limited to one or two parents, an evolutionary algorithm has been shown to perform better with more parents [9]. However, this adds additional complexity to the system, two parents will be used. In order to take advantage of the genetic programming structure, the filters must be composed of small component pieces.

A binary tree of a fixed depth can be implemented as an array. This allows easy traversal between children and parents when using element 1 as the top level parent and $2i$ for the left child and $2i + 1$ for the right child, where $i$ is the level of the tree.
3.3.2 Population

A population is a collection of individuals with a common fitness function. It would be possible to create an environment that contained multiple populations. In the simplest case, the populations do not exchange genetic material and an environment is simply a parallelization of the algorithm. However, genetic material can be exchanged which would reintroduce genetic diversity [10]. This could help move a population out of a local minimum. It may be even more interesting to introduce more diversity by using slightly different fitness functions between the populations and sharing the genetic information.

3.3.3 Breeding

The fittest individuals of a population are breed. An evolutionary algorithm is naturally a distributed process, but the breeding can be tweaked to support
even more distribution of the computation. The most basic implementation of a generation requires all individuals to be evaluated before breeding. Instead, a generation can be defined as a certain number of matings. This allows the mating to occur whenever individuals are evaluated and join the mating pool. The individuals in the population are added to the mating pool as soon as they are evaluated for fitness. There must be a minimum number of evaluated individuals to start mating. As long as this minimum number is met, when an individual is added to the mating pool, a new random breeding is done to create a new individual. In order to conserve resources individuals must eventually die, that is be removed from the available mating pool. Individuals with poor fitness will be killed off to keep the total population constant and the individual to remove from the pool is randomly selected favoring less fit individuals and never the most fit individual. Unlike in nature where old age starts to affect the fitness of an individual the fitness will not change no matter how many generations an individual lasts. In all cases roulette wheel selections will be made as described below. This randomizes the choice based on percentage as seen in figure 3.3. Using a mating pool inherently favors filters that are computed quickly, which may or may not be desirable.

\[
\text{Matings Per Generation} = \frac{\text{IndividualsInPopulation}}{3}
\]
Minimum Individuals To Mate

\[ \frac{\text{Individuals In Population}}{3} \]

Max In Pool - number individuals in starting population

Mutation Rate - start at 0.6 percent, an accepted standard [11]

Roulette wheel selection [10] is a term given to random selection that allows a parameter to control the distribution. For example, random selection of mates would use fitness as the parameter.

\[ \text{BreedingSelectionChance} = \frac{\text{Fitness Of Individual}}{\text{Total Fitness In Mating Pool}} \]

Figure 3.3: The uneven pie of roulette selection

3.4 Signal

There is another important data structure within the system, the signal. The inputs and outputs are signals, but there is also a large number of intermediary signals used by each member of the population. In order to avoid recalculating complex filters the signal each filter produces should be persistent. The program must be able to randomly access the signal \( (t = -5) \) and output a subsection of the signal \( (t = -5 : -15) \). An undefined signal will be considered 0.
3.5 The Inputs

3.5.1 Fundamental Inputs
Fundamental inputs for financials are used to evaluate the strength of the underlying business. A few examples are price to earning ratio, price to book ratio, and earnings forecasts.

3.5.2 Technical Inputs
Technical inputs have been used for many years by chartists to predict future stock movements. Price and volume are the two most used quantities. From Google Finance a large set of open, close, high, low, and volume entries are available for publically traded stocks.

3.5.3 Other Inputs
Ultimately the filters do not care where signals come from. Signals such as rainy days or the lunar cycle could be included, however the less noise in the pool of available inputs the quicker a good filter can be found. News stories would likely be a useful input.

3.6 Filters: A Function Approximation Engine

3.6.1 Pre-processing
The inputs must be normalized to time-series signals of floating-point numbers. Most financial data already exists in such a form, and historical data at one day resolution is readily available from sources such as Google Finance. However, other signals such as number of news articles or news articles themselves may require processing to turn into a usable form. The signals should not be required to be the same length, but should use the same timestep. Interpolation may be required to process the data and turn it into usable signals.

3.6.2 Filters
\( P \) is the feed forward filter order
\( b_i \) are the feed forward filter coefficients
$Q$ is the feedback filter order
$a_i$ are the feedback filter coefficients
$x[n]$ is the input signal
$y[n]$ is the output signal

**FIR Filters**

A finite impulse response filter or FIR filter can be represented by a difference equation:

$$y[n] = \frac{1}{a_0} (b_0 x[n] + b_1 x[n-1] + \cdots + b_P x[n-P])$$

**IIR Filters**

An infinite impulse response (IIR) filter can be represented by the difference equation:

$$y[n] = \frac{1}{a_0} (b_0 x[n] + b_1 x[n-1] + \cdots + b_P x[n-P] - a_1 y[n-1] - a_2 y[n-2] - \cdots - a_Q y[n-Q])$$

The infinite impulse filter incorporates feedback from previous output signals. It will be limited to second order feedback.
Transforms

Transforms to the signal can also be applied, such as an FFT.

Combinational Filters

Since the complex chained filter will be represented by a binary tree, filters will be limited to two inputs. The system will use two types of multi-input filters: accumulator and comparator. The accumulator sums the two inputs. The comparator uses logic (such as greater than) to pass one of the inputs as the output.

3.6.3 Filter Factory

Parameters

In order to be introduced into the population a function factory will be used. This is a typical object-oriented software design pattern [12]. The starting population will be created based on average depth, and the distribution of the types of filters FIR, IIR, transform, and combinational and order of the IIR (FIR is IIR order 0) will be controlled by the factory. The tuning of the factory allows the tuning of the percentage of different types of filters.

- **Average Depth** - the average depth of the factory created starting tree
- **IIR\(_0\)Weight** - the proportion of IIR 0th order filters. This value divided by the total filter weights is the percentage. An IIR 0th order filter is a FIR filter.
- **IIR\(_1\)Weight** - the proportion of IIR 1st order filters.
- **IIR\(_2\)Weight** - the proportion of IIR 2nd order filters.
- **Accumulator Weight** - the proportion of two input filters that are accumulators. This weight divided by the sum of all two input weights is the random percentage this is used.
- **Comparator Weight** - the proportion of two input filters that are comparators.
Optimization

These parameters must be tuned. Each population is bound to one filter factory, but populations could be evolved with different factories to tune the parameters. It would be possible to create a factory that specified the percent of certain types of FIR and IIR filters such as gain, integration, moving average, and differentiation.

3.6.4 Mating Filters

The mating of two filters combines subtrees of the parents in a random way. The maximum depth and other parameters are maintained and the combining of subtrees is termed crossover [11]. There is a certain rate of mutation - the rate at which the genetic material is randomly modified to produce a child, as stated above this system will start with a 0.6 percent mutation rate.

A graphic example follows:

In Figure 3.5, two parents have been selected using the roulette method that favors more fit individuals.

In Figure 3.6 an attachment point has been randomly selected on the left parent and a branching point on the right. These are shown in yellow.

In Figure 3.7 the branching point and subtrees replace the attachment point. A entirely new filter is created.

However we have must make sure our graft complies with the rules of the tree. It must terminate in a source and it must not exceed the max depth. If the max depth is exceeded then it must be reconciled. This could be avoided by first picking the level then randomly picking a node at that level to graft. However, as shown in Figure 3.8, the final child for a max depth five would have to be trimmed.

3.6.5 Building Blocks

A complex filter can be composed of cascaded simpler filters. The basic building blocks of this system, where $A$, $B$, and $out$ are signals and $x$ is a real number and $i$ is an integer, are as follows:

- **source**: outputs a historical data signal
- **gain**: $out = x \cdot A$
- **FIR**: $out = x_{FIR} \cdot A[-i_{FIR}]$
IIR: \( \text{out} = A + x_{IIR} \cdot \text{out}[−i_{IIR}] \)

delay: \( \text{out} = A[−i_{\text{delay}}] \)

sum: \( \text{out} = A + B \)

### 3.7 Fitness

The fitness needs to represent how well the complete filter tree predicts the future values. One fitness that represents a function approximation well is a coefficient of determination. This is also known as an \( R^2 \) value, a higher \( R^2 \) value is better. \( R^2 \) is used in statistics when the main purpose is the
prediction of future outcomes on the basis of other related information. It is the proportion of variability in a data set that is accounted for by the statistical model. The fitness will be explored with a prediction of different steps in the future, ranging from one to ninety steps.

$$Fitness = 1 - \frac{SS_{\text{err}}}{SS_{\text{tot}}}$$

$$SS_{\text{tot}} = \sum_i (x_i - \bar{x})^2$$

$$SS_{\text{err}} = \sum_i (x_i - f_i)^2$$

where $SS_{\text{tot}}$ is known as total sum of squares (proportional to the sample variance), $SS_{\text{err}}$ is known as the sum of squared errors. $x$ is the input signal.
Figure 3.7: The resulting child filter

Figure 3.8: The max depth 5 child filter

(price of a certain security) and $f$ is the output of the prediction filter. $\bar{x}$ is the estimated mean of the input.

However, when applied to stock trend prediction this function is too polarizing. There are very few good results and many very bad ones. A fitness function that produces a better distribution of fitness is percent correct. The filter’s sources use a window of the previous one hundred days and a block of predictions are made. As shown in Figure 3.9 The number of correct trend predictions over this block is the fitness.

$$Fitness = \frac{CorrectPredictions}{TotalPredictions}$$
Figure 3.9: The system makes a prediction based on the previous 100 data points.
Chapter 4

System Testing

4.1 Test Data

The system will be tested with known historical data to benchmark its predictive capabilities. The results of the tests are recorded in chapter 5. The original testing plan was as follows:

Simple Tests

The simplest test would be to use a source delayed by one timestep as the target signal. Then the ideal filter would be only that source. This test was conducted with random signals A, B, and C available in the pool of signals and with the Dow Jones 30 group of stocks as the pool of signals. The expected result that the best filter should be only the source or a combination of filters that reduce to only that source, as the system does not naturally prefer smaller filter chains.

Derived Filters

Simple filters were used to create the focus signal. An integration of one source, a derivative of one source, and a gain of one source were all be tested. The best fitness value as a function of the number of generations was explored. This tested the systems ability to evolve two types of filters that are often seen in nature. The expected performance that a close approximation accurate to 5% of the target should be produced within one hundred generations.
Complicated Filter

The performance of the system on a more complicated and known target signal was explored. A randomly generated depth four binary filter tree was created in a similar manner to how individuals are randomly initialized. This created the target signal. It was expected that the system’s fitness would approach the theoretical maximum over a number of generations. An approximation that is accurate to 99% of the focus was expected within 1000 generations.

Performance with Noise

The next step was to introduce noise into the target signal. This was similar to random future information which would affect stock signals, assuming good news is as likely as bad news. Noise from five to seventy-five percent of the focus’s median value was tested with both the simple filters, derived filters, and the complicated filters. It was expected that the performance of the system should decrease as noise is introduced.

4.2 Output on Stock Data

The system does not seek the globally optimal solution, merely a good solution. A good solution for financial data is one that can be exploited for profit, taking into account the real cost of money. It must be right to a greater degree than it is wrong, and must make up for any brokers fee for trading. The solution will likely change over time, but in order to have confidence in the system, the system must be able to produce a statistically significant profit over a significant period of time. This may not be possible for all stocks or even any stocks with this system. The Dow Jones 30 - the thirty stocks in the Dow Jones Industrial Average were used to provide the historic data. The effectiveness of the system was analyzed for each member stock. Twenty percent annualized returns have been demonstrated using time series prediction and the German stock index [13], achieving such a return with a simpler model on a larger market would be considered an excellent result. In fact, assuming a transaction cost of 1%, any return over prime (currently 0.50% [14]) would be considered good.
Chapter 5

Results

5.1 Test Results

The first system test completed was predicting a ramp using the $r^2$ sum fitness using that ramp as a source. This is a test of the filter factory’s ability to create diverse starting filters when using three as the maximum starting filled depth. One hundred filters were tested three times. The results were:

<table>
<thead>
<tr>
<th>Total Fitness</th>
<th>Best Fitness</th>
<th>Individuals with Positive Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.46</td>
<td>1.0</td>
<td>7</td>
</tr>
<tr>
<td>7.53</td>
<td>0.998</td>
<td>8</td>
</tr>
<tr>
<td>9.06</td>
<td>1.0</td>
<td>10</td>
</tr>
</tbody>
</table>

*The system is able to generate reasonable initial filters*

The system’s compartmentalization of source data was tested. Since historical data was being used to simulate predictions it was important that for any given timestep the filter could only use past data and was not using the future value that was used to check for accuracy. The above test was repeated but with 10% standard deviation on the source values. If the system was using future values the prediction accuracy would again be near 1.0, otherwise if the system was behaving as designed it would be near 0.9.

<table>
<thead>
<tr>
<th>Total Fitness</th>
<th>Best Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>0.91</td>
</tr>
<tr>
<td>10.2</td>
<td>0.91</td>
</tr>
<tr>
<td>5.3</td>
<td>0.90</td>
</tr>
</tbody>
</table>

*The system copes with noise as expected*
The next system test was approximating an exponential function with 10% standard deviated noise. This is the test of the ability to breed better children. Using a ramp, constant, and exponential sources we focused the evolution on predicting the next exponential time step. The results were:

<table>
<thead>
<tr>
<th>Generation</th>
<th>Best Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.49</td>
</tr>
<tr>
<td>1</td>
<td>0.55</td>
</tr>
<tr>
<td>2</td>
<td>0.82</td>
</tr>
<tr>
<td>3</td>
<td>0.82</td>
</tr>
<tr>
<td>4</td>
<td>0.82</td>
</tr>
<tr>
<td>5</td>
<td>0.82</td>
</tr>
<tr>
<td>6</td>
<td>0.82</td>
</tr>
<tr>
<td>7</td>
<td>0.82</td>
</tr>
<tr>
<td>8</td>
<td>0.82</td>
</tr>
<tr>
<td>9</td>
<td>0.82</td>
</tr>
</tbody>
</table>

For a simple problem there is limited improvement beyond the third generation

Figure 5.1: The system $r^2$ sum fitness performance with multiple sources

These results are encouraging. They show the ability to breed better filters on slightly noisy data. The fitness is achieved rather quickly and does not improve after the third generation which means this evolutionary method might not be the best search method for this problem. However, it is a fairly simple search so a quick maximum is expected.
5.2 Performance Enhancements

The evolutionary approach is computationally intensive. Several performance enhancements to the code base were benchmarked. The sources were changed to a modified singleton pattern. This means that a source was created only once when it was first needed and subsequent individuals that used the same source received a reference to the original object. Over 10 generations of 10 individuals this improved performance from 530s to 510s. Over 3 generations of 150 individuals this improved performance from 2603s to 50s.

In order to further increase performance a lower level of logging was created which removed the println from every evaluation. Coupled with minor memory modifications, this reduced execution time of 60,000 evaluations from 6581s to 232s.

5.3 Stock Prediction

Initial results using $r^2$ sum for fitness produced solutions with extremely low fitness and did not appear to be able to breed better filters with any reliability. The fitness was modified to trend prediction with three exclusive predictions (up more than one percent, same, and down more than one percent). The 1% value choosen to divide the trends was selected because it was close to the average violatility of the DOW Jones 30, meaning the trends should be approximentaly equally distributed. Using 100 generations of 250 individuals with 5% hypermutation (brand new filter), and 3% mutation when breeding the maximum fitness is recorded in the Figure 5. A two year block of data between 2005 and 2007 was examined with a 100 trailing day window available to the sources.
Figure 5.2: The system performance using trend fitness on historical data

The best filters were correct over 80% of the time during that two year block while the expected rate of a random choice would be 33%. The limit between up, down, and same predictions was set at 1%.

Figure 5.3: The trend fitness is a trinary prediction
Chapter 6

Conclusions

It was demonstrated that the system worked as expected for simple test cases. Furthermore, on real data from the DJIA digital filters could be found using an evolutionary process with greater than 80% correct prediction rate over a two year period. However, it was not demonstrated that these filters were good filters. A good filter must be able to be used to earn a return on investment greater than the real cost of money. There are various trading strategies that could be employed - the simplest of which would be to buy the stocks predicted to go up and short sell the stocks predicted to go down. There are option\(^1\) trading strategies that are typically used to bet on trends that could most likely be made more effective than the simple strategy.

An evolutionary algorithm is a useful search tool. However, its development cost should not be underestimated. It is an expensive technique to develop and test partly because it is a random process - there is no single expected result but a range of expected conditions. It is non-deterministic, running it multiple times is nearly guaranteed to give different results. Evolutionary algorithms bring additional design considerations. It is important to deal with diversity in order to avoid local maxima and bloat in order to get efficient results in a reasonable period of time. This system takes care of these considerations by using a maximum filter size and by using hyper mutation and mutation during breeding.

More work is required to continue this research and evaluate the amount of information available from historical stock data using this method. The results thus far are promising in that a certain amount of information can be procured from historical data but far from conclusive.

\(^1\)the option to buy or sell stocks at a certain price in the future
6.1 Future Work

While the results are encouraging they require more verification. Especially because the system is a random process it must be tested on many other blocks of time. The results (see Figure 5) for the tested time block may be an anomaly. Additionally, different length block should be tested. It may be worthwhile to bring the size of the window that is used for the fitness evaluation inside the evolutionary search as a parameter.

It is unclear exactly how effective the evolutionary method of search is for finding good filters. The fitness progression over the generations must be recorded and analyzed, if the results plateau after only a few generations it is unlikely the method is effective. This test must be done over multiple blocks of time just like the final result.

Good filters were found, but the returns possible with the filters depend greatly on the trading strategy that capitalizes on this information. Different trading rules should be simulated. It may be possible to incorporate the trading rules into the evolutionary algorithm which would take advantage of its derivative-free nature.

While the current results were produced with fundamental technical inputs like price and volume better indicators may be available. Expanding the available inputs a few at a time and comparing the results over multiple blocks would be a worthwhile investigation.
Bibliography


