Evolving a Low Price Recovery Strategy for Distressed Securities

Robert E. Marmelstein(✉), Alexander L. Hunt, and Christoper Eroh

Computer Science Department, East Stroudsburg University,
East Stroudsburg, PA 18301, USA
rmarmelstein@esu.edu, {ahunt,ceroh1}@live.esu.edu

Abstract. This paper investigates methods to evolve an automated agent that executes a niche trading stock strategy. Unlike trading strategies that seek to exploit broad market trends, we choose a very specific strategy on the assumption that it will be easier to learn, require less input data to do so, and more straightforward to evaluate the agents performance. In this case, we select a Low Price Recovery Strategy (LPRS), which involves picking stocks that have a high likelihood of quickly recovering after a steep, one day decline in share price. A series of intelligent agents are evolved through the use of a Genetic Programing approach. The inputs to our algorithms included traditional stock performance metrics, sentiment indicators available from online sources, and associated classification rules. The essential aspects of the research discussed include: identification of opportunities, feature selection and extraction, design of various genetic programs for evolving the agent, and testing approaches for the agents. We demonstrate that the evolved agent yields results outperform a randomized version of the LPRS and the benchmark Standard & Poor's 500 (S&P500) stock market index.

1 Introduction

The last two decades have seen substantial work done in development of algorithms to perform rapid, automated trading of securities. Given the need to adapt to dynamically changing market conditions and the tremendous amount of data available to make trading decisions, the machine learning community has been in the forefront of these efforts. For many trading firms, the use of neural networks has proved extremely popular [6][8]. Such "black box" techniques excel in performance and the underlying design is much easier to keep secret for the purposes of competitive advantage. However, from a knowledge acquisition perspective, it is preferable to learn rules which can explain why a given decision was made. Once promising rules are identified, they can form the basis for more sophisticated, intelligent agents which use those underlying rules to make trading decisions. Evolutionary Algorithm techniques, such as Genetic Algorithms (GA) and Genetic Programming (GP), are a proven way to combine existing rules or discover new ones in order to construct such trading agents.

P. Perner (Ed.): MLDM 2016, LNAI 9729, pp. 1–14, 2016.
DOI: 10.1007/978-3-319-41920-6_1

1.1 Problem Description

This research evolves automated agents to trade using a Low Price Recovery Strategy. The main idea of the LPRS is to identify and purchase selected stocks which suffer a steep decline in share price, but have a high likelihood of rebounding within a short period of time. The candidate stocks selected for this strategy possess three characteristics. First, the stock share price must decline by at least 10% in value on a single trading day. While a lower percent decline would yield more candidates, we chose 10% as our decline threshold in order to ensure a resonable profit if a rebound occurs within a sixty (60) day time period. The trading day on which the decline occurs is known as the baseline day.

A second critera is that the company must have a pre-decline market capitalization value of at least $10B dollars. We incorporated this rule because we wanted to evaluate the impact of online sentiment on the likelihood of recovery. While using small capitalization stocks would have increased the size of our candidate data set, these tend to have little social media chatter relating to the decline event. Measurable levels of chatter are more likely with larger capitalization companies than smaller ones. Third, the stock share type must be common stock only.

Once candidates have been identified, they are then classified as either recovered or non-recovered. A "successful" recovery occurs when a stock gains back 75% of the value it lost within 60 days after the baseline day. Note that a stock just has to breach this threshold once to be successful. Thus, a stock that rebounds and then declines further within the 60 day period is still labeled as recovered. Any candidate stock that does not meet this condition is classified as non-recovered. We chose the 60 day period because it is long enought to give the stock adequate time to recover and short enough to reinvest the proceeds up to six (6) times in a given year. Figure 1 shows an example recovery situation for the Best Buy Corporation (stock symbol: BBY).

In our LPRS, the agent must decide to purchase the stock (or not) the day after the baseline day; recommended purchases are executed immediately. This snap decision is based on the available information at that time. If a stock is purchased, its target price (current price + 75% of the absolute decline amount) is computed. The stock is then held in the portfolio until the target price is met or until the 60 day hold period has passed. If the former condition occurs, the stock holding is considered to be sold the first time the target price is reached. If the latter condition occurs, the holding is sold for the stock's closing price on the last day of the holding period. Note that a non-recovered stock may result in either a capital loss or gain. If a gain, it will be less than the gain if the stock had met our recovery criteria.

2 Approach

We employ the Genetic Programming paradigm to evolve several LPRS agent variations. GP can be viewed as an extension of the GA, a biologically inspired

Fig. 1. Low Price Recovery Example (BBY)

model for testing and selecting the best choice among a set of results, each represented by a string. However, GP goes a step farther–it generates a program to solve a problem. The utility of the generated program is tested using a predefined fitness function. Two steps are employed over multiple generations to evolve a successful program. The first is selection of the fittest set of programs, using a competitive approach (such as a tournament). The second step is breeding the selected program, using crossover and mutation techniques, to create a new population of even fitter solutions for the next generation.

A crucial aspect of any GP is designing the fitness function to measure the degree to which a program is achieving the desired goal. In this experiment, we use different fitness function to evolve multiple variations of the LPRS agent. The GP chromosome in our application encodes trading rules as a function in the form of a tree. Figure 2 shows an example of such a tree structure. Each chromosome is evaluated and the value returned by the function (root node) is converted into a Boolean value. In our case, if the function returns a positive value, it is interpreted as a buy signal (true); when a negative value is returned, no purchase is made. Note the tree structure is composed of a combination of primitive functions (e.g., +, -, sin, exp, etc.), variables and variables. The complete set of these primative functions are described in section 2.3.

We chose GP as our evolutionary computation paradigm because it has more flexibility in exploring the algorithmic search space to creating novel trading strategies than a GA. Extensive tutorials on both GAs and GP can be found at [2] [7] [11] [12] [18].

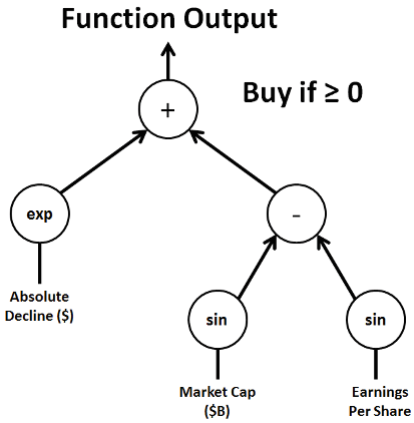

Fig. 2. Function represented as a GP tree structure

2.1 LPRS Agent Fitness Functions

Our GP utilized two types of fitness functions to evolve the LPRS agent. The first function type is based on a Confusion Matrix (CM) [10], which measures the agents ability to correctly classify each candidate stock as recovered or non-recovered. In machine learning, a CM contains the true positives, false positives and false negatives for every class in the data set. In our approach, the classification results over the training set are put in a confusion matrix (X). Each diagnonal value of the matrix (x_{ii}) represents the true positives the ith class–that is, the percentage of class i that was correctly classfied. The fitness value is computed by multiplying the diagonal values of the matrix together (see Equation 1). Thus, the fitness metric reflects the agents ability to correctly discriminate between classes, even if it comes at the expense of overall classification accuracy.

$$Fitness = \prod_{i=1}^{n} x_{ii} \tag{1}$$

The second type of fitness function returns the overall profit (P) ratio earned by the agent. For this case, a simulation is elaborated with the LPRS agent's portfolio having an initial cash balance and no stocks on a predetermined start date. Beginning with this date, trade decisions are made in temporal order, as candidates are identified. Stock holdings are sold according to the rules enumerated in section 1.1. Capital gains on trades are added to the available cash reserve in the account; likewise, capital losses are subtracted. Using this approach, purchases are only carried out if the sufficient free cash exists in the portfolio to cover the transaction. The fitness result is returned based on the percent increase in portfolio value within 60 days after the last security pur-

chase is made. The overall fitness of the LPRS is the ratio of the starting and ending portfolio balances (see Equation 2).

$$Fitness = \left(\frac{PortfolioValue(\$) - Ending}{PortfolioValue(\$) - Starting} \right) - 1 \tag{2}$$

Unlike the CM fitness, P fitness is highly dependent on the time-sensitive aspects of trading, both in terms of order and spacing of stock purchases. For example, if there is not sufficient cash available to buy a stock, it is not purchased regardless of what the buy signal is. Thus, performance is dependent on the kinds of resource and timing constraints that real world traders must contend with.

2.2 Types of LPRS Agents

Each LPRS agent evaluates a candidate stock using some set of features. The set of features utilized is dependent on the agent type. For any given candidate, the output of the LPRS agent is a buy signal. A positive buy signal indicates the stock should be purchased; no purchase is made for a negative signal. In cases where a profit (P) fitness function is used, the stock is not necessarily purchased even if the LPRS agent emits a positive buy signal. For this experiment, the following types of LPRS agents were used:

- Random. This agent selects candidate stocks for purchase at random; each candidate has a 50% probability that the agent will generate a positive buy signal. This agent-type is used only with the P fitness function. Note that this agent type is solely intended as a baseline for performance comparison with other agent types.
- Metric-based. This agent is evolved from a combination of fundamental stock metrics, features derived from social media sources, and low level primitive operators.
- Rule-based. For this type of agent, each metric in the dataset is evaluated using a Classification and Regression Tree (CART) [13] technique on its ability to discriminate between the two classes. The most promising rules, described in section 2.3, are input as features to evolve the GP.
- Majority-rule. For this type of agent, we select the nine (9) most promising agents for a majority-rules scheme. That is, if a majority of the selected LPRS agent output a positive buy signal, then the buy-signal of the conglomerate is positive; otherwise, it is negative.

2.3 Genetic Program Characteristics

In this section, we describe the inputs to the Genetic Programming framework. For our experiment, we chose the AForge.Net framework [9], which provides a wide variety of machine learning C# libraries, including GA and GP. The primitive functions supported by the frameworks default GP chromosome are: addition, subtraction, multiplication, division, sine, cosine, exponent, natural logarithm, and square root. These primitives operate on a set of features, which

appear as leaves on the tree (see Figure 2). The choice of features is very important to the success of a GP; in some cases, the features themselves may be implemented as functions. The feature set employed for the LPRS agents are described in the subsections that follow.

Candidate Stocks. For this research, we surveyed the biggest decliners on the New York Stock Exchange (NYSE) for the period from Jan 1, 2011 through April 30, 2015. From the total list of decliners, we identified 207 transactions that met the criteria described in Section 1.1. These cases were almost evenly split between the two classes, with 101 (48.8%) in the recovery class and 106 (51.2%) in the non-recovery class. We partitioned this set of candidates into two subsets for training and testing the agents, respectively.

Stock Metrics. For each candidate transaction, the following metrics were available as features for GP training:

- Decline date
- Closing share price on decline date
- Absolute price decline (on decline date)
- Percentage price decline
- Recovery or Sell Date
- Final Share Price
- Trade volume in millions of trades (on decline date)
- Market Capitalization of company in billions of $ (pre decline)
- Earnings per Share (prior quarter)
- Current Ratio [16]
- Dividend Per Share (quarterly)
- Price-Earnings (PE) Ratio [16]

Several other features, derived from online sources, were used as well (below). The purpose of these features was to get a snapshot of the prevailing sentiment about each security on the day of its decline.

- Google Trend Index. The weighted stock symbol value from Google Trends on day of decline. This is the trend value on the decline day divided by the average Google trend value for the previous three (3) months.
- Message Post Index. The normalized number of message board posts on Yahoo Finance about the stock on the day of the decline. This value is normalized with respect to the mean percentage increase in posts noted for all candidate stocks on the day of decline.
- Analyst Count. The total number of financial analysts following the company that issued the security.
- Analyst Sentiment. The cumulate change in sentiment of analysts over the past 120 days. Each update to analyst ratings on the stock changed the sentiment value, on a scale between -1 and $+1$ depending on the specific ratings change.

Buying Rules. The following are decision rules that were generated using the CART decision tree algorithm. These rules were chosen based on their ability to discriminate each candidate into the correct class. Each of the below four (4) rules were used as input to the GP to evolve the rule-based LPRS agents.

A. Buy signal true if: absolute price decline is greater than $4.01; false otherwise.
B. Buy signal true if: PE ratio is less than 9.3; false otherwise.
C. Buy signal true if: volume is greater than 40.26M OR less than 0.7707M shares; false otherwise.
D. Buy signal true if: the percent price decline is between 10.1% and 11.5%; false otherwise.

Additionally, for the rule-based LPRS agent, the following metrics were also input to the GP: percentage price decline, market capitalization, volume, sentiment count, Message Post Index and Google Trend Index.

3 Related Work

Over the past two decades, both GAs and GPs have been applied to a wide range of financial trading problems[5]. In many cases, the resulting evolved solutions have been shown to outperform both human traders and benchmark trading indices (e.g., S&P500) for specific problems. This approach differed from ours in that the pool of candidate stocks was much smaller[1] used GPs to evolve stock trading rules for the S&P500 index as a whole from 1929 through 1995. After transaction costs were factored in, it was found the rules did not earn consistent excess returns over a simple buy-and-hold strategy in the out-of-sample test periods. The rules were able to identify periods to be invested in the index when daily returns were positive and volatility is low and out when the reverse was true. Becker & Seshadri [3] were later able to improve upon their work and actually outperform a buy-and-hold strategy for the S&P500 index. Potvin, Soriano & Vallée [15] employed a GP approach to trade a small pool of fourteen (14)l Canadian stocks, with each stock selected from a distinct commercial sector. Like [1], their evolved rules were unable to outperform a buy-and-hold strategy. In contrast to these researchers, our approach did not use either indices or a specific pool of stock. Instead, we utilized the LPRS criteria to continuously indentify specific buying opportunities from the entire NYSE. The only real decision our evolved agent makes is the buy decision; unlike these other approaches, our sell decision is automatic. Thus our approach does not result in excessive churning of stock purchases. Further, when our P fitness function is utilized, the cash balance of the portfolio serves as a limiting factor for potential stock purchases.

A number of researchers have also experimented with using public sentiment as a basis for predicting the performance of markets and individual stocks. Bollen, Mao, & Zeng [4] showed that the general public mood, as derived from Twitter feeds, was correlated to the Dow Jones Industrial Average (DJIA) over time. Nuij et al. [14] incorporated mined specific categories of news events and incorporated them into a GP to evolve trading rules for the FTSE350 and

S&P500 indices. They found that augmenting news information with more traditional technical financial indicators generated higher returns than if the news events had not been included. While our approach likewise utilizes sentiment to make buying decisions, we attempt to measure the prevailing sentiment for individual stocks (vs. global stock indices). This drove our decision to ignore stocks which had a low market capitalization since there appeared to be insufficient sentiment information available for these stocks. Vu, Chang, Ha & Collier [17] had considerable success in using Twitter sentiment to predict the daily up/down changes of widely held technology stocks (AMZN, APPL, GOOG, and MSFT). A key advantge of this approach is that the large size of the companies helps ensure a critical mass of sentiment was available. Even so, large companies tend to have fairly stable stock prices; as such, they are typically not LPRS candidates.

4 Experimentation and Analysis of Results

In this section, we describe the setup of the experiment, including all test cases. We then present and analyze the results of each test case. The primary objective of our analysis is to compare approaches for evolving the most effective LPRS agent. As part of our analysis, we also compare the performance of our evolved agents against the performance of the S&P500 index and a random purchase decision for LPRS candidates.

4.1 Experiment Setup

We experimented with a number of different strategies for evolving each type of LPRS agent. In general, these strategies varied by:

- Agent type
- Fitness function type
- Buying strategy (buy all or selective)

For some cases, the fitness function type was alternated, such that one was used for training and the other for testing. The motivation here was to see how effective an agent trained on one fitness function would be on the other. Table 1 provides a summary of test cases.

The breakdown of transactions by class was discussed in section 2.3. For this experiment, the training period was from January 1, 2011 through 31 May 2014. The test period was from 1 June 2014 through 30 April 2015. The training and test data sets had 165 and 42 transactions, respectively. The parameters for the evolved GP were as follows:

- Population size = 100
- Maximum generations = 100
- Crossover Rate = 75%
- Mutation Rate = 10%
- Selection method is Roulette wheel

The results reported for each test case are based on ten (10) agents evolved using the GP.

Table 1. Summary of Test Cases

Exp ID	Metrics Type: Stock	
	Train	Test
SM-CM-All	Fitness: CM; BuyMode: All	Fitness: CM; BuyMode: All
SM-CM-Sel	Fitness: CM; BuyMode: All	Fitness: CM; BuyMode: Selective using GP-Output
SM-CM-Vote	Fitness: CM; BuyMode: All	Fitness: CM; BuyMode: Selective using GP-Vote
SM-P-Sel	Fitness: Profit ; BuyMode: Selective using GP-Output	Fitness: Profit ; BuyMode: Selective using GP-Output
SM-P-Vote	Fitness: Profit ; BuyMode: Selective using GP-Output	Fitness: Profit ; BuyMode: Selective using GP-Vote
Exp ID	**Metrics Type: Rule-Based**	
	Train	**Test**
RB-CM-All	Fitness: CM; BuyMode: All	Fitness: CM; BuyMode: All
RB-CM-Sel	Fitness: CM; BuyMode: All	Fitness: CM; BuyMode: Selective using GP-Output
RB-CM-Vote	Fitness: CM; BuyMode: All	Fitness: CM; BuyMode: Selective using GP-Vote
RB-P-Sel	Fitness: Profit ; BuyMode: Selective using GP-Output	Fitness: Profit ; BuyMode: Selective using GP-Output
RB-P-Vote	Fitness: Profit ; BuyMode: Selective using GP-Output	Fitness: Profit ; BuyMode: Selective using GP-Vote

4.2 Experiment Results

For the CM-ALL experiment, we evolved a series of ten GPs using the CM fitness function and the training data set. In each case, the final, evolved GP was then selected as the LPRS agent and run against the test data set. This was done for both the Stock Metric (SM) and Rule-based (RB) feature sets. The averaged results of these runs are shown in Table 2. The REC and NonRec columns in the table indicate the percentage of items in each class that were correctly classified by the GP. Though not shown in Table 2, the fitness for each test case may be computed using Equation 1.

Table 2. Confusion Matrix Training and Test Results

Test Case	Statistic	Training		Test	
		REC %	NonREC %	REC %	NonREC %
SM-CM-All	Mean	67.6%	62.9%	57.9%	49.6%
	STDev	6.6%	6.5%	13.0%	16.0%
RB-CM-All	Mean	70.8%	67.9%	70.0%	48.1%
	STDev	6.3%	3.8%	4.5%	7.8%

When we use the CM fitness function, we are most concerned with evolving an LPRS agent that can distinguish a recovery situation from a non-recovery one. While such an LPRS agent is not explicitly trained to maximize profit, it is reasonable to expect it to be profitable. Table 3 shows the extent to which this may be true. Here we run the LPRS agents evolved for the SM-CM-ALL and RB-CM-ALL cases to determine the increase in portfolio value each would reap.

The P fitness function, given in Equation 2, is reported against both the training and test datasets (labeled in Table 3 as Test-1 and Test-2, respectively). Note that there is no need to provide the data for the REC and NoREC columns in Table 3, because the results would be identical to those in Table 2.

Table 3. CM Training and Profit Test Results

Test Case	Statistic	Test-1 (on Training data)			Test-2 (Test Data)		
		Fitness	REC %	NonREC %	Fitness	REC %	NonREC %
SM-CM-Sel	Mean	0.624	N/A		0.164	N/A	
	STDev	0.078			0.051		
RB-CM-Sel	Mean	0.607	N/A		0.182	N/A	
	STDev	0.094			0.124		

Next, we selected the top 9 out of 10 LPRS agents from the previous experiment based on the Table 2 results. We then combined these to create a composite LPRS agent, where the REC/NoREC determination is made based on what the majority of agents decide. The results of this test case are shown in Table 4. As is the case for Table 3, the fitness is computed using the P fitness function and we also track the CM classification accuracy for each composite LPRS. Since we employ majority rule to make this decision on the data, this experiment consisted of a single run for each test case. For this reason, there was no need to report the standard deviation statistic.

For the last two experiments, we focused on LPRS agents evolved to maximize profit during training. The results for the SM and RB feature set are given in Table 5. Even though profit is the guiding factor here, the REC/NoREC results are included to give a sense of how critical class discrimination is to profit.

Table 4. Best of Nine (9) Votes - CM Training and Profit Test Results

Test Case	Statistic	Test-1 (on Training data)			Test-2 (Test Data)		
		Fitness	REC %	NonREC %	Fitness	REC %	NonREC %
SM-CM-Vote	Mean	0.7355	76.7%	65.1%	0.2289	64.3%	63.0%
	STDev	N/A					
RB-CM-Vote	Mean	0.4724	67.8%	69.7%	0.3519	67.9%	51.9%
	STDev	N/A					

Table 5. Profit Training and Test Results

Test Case	Statistic	Training			Test		
		Fitness	REC %	NonREC %	Fitness	REC %	NonREC %
SM-P-SEL	Mean	0.816	90.4%	12.9%	0.188	85.4%	25.5%
	STDev	0.035	3.5%	5.2%	0.035	5.9%	6.8%
RB-P-SEL	Mean	0.832	91.2%	21.5%	0.181	88.1%	17.0%
	STDev	0.034	5.5%	10.6%	0.030	8.7%	13.3%
Random	Mean	0.315	N/A		0.137	N/A	
	STDev	0.129			0.028		
S&P 500	Mean	0.530	N/A		0.086	N/A	
	STDev	N/A					

Table 6. Best of Nine (9) Votes - Profit Training and Test Results

Test Case	Statistic	Test-1 (on Training data)			Test-2 (Test Data)		
		Fitness	REC %	NonREC %	Fitness	REC %	NonREC %
SM-P-Vote	Mean	0.8174	91.3%	10.6%	0.1725	85.7%	29.2%
	STDev	N/A					
RB-P-Vote	Mean	0.902	98.8%	17.7%	0.2098	95.2%	4.8%
	STDev	N/A					

Finally, Table 6 shows the performance of the majority rule approach for LPRS agents evolved for the profit fitness function. Similar to Table 4, we selected the top nine LPRS agents from those evolved for the Table 5 test case. Thus, the results in Table 6 reflect a single, composite LPRS agent.

4.3 Analysis of Results

While Tables 2 through 6 contain the raw experiment results, Figure 3 compares the performance results for each approach side by side. For this figure, we converted the raw profit fitness performance into an annualized Return on Investment (ROI) percentage. Taking duration out of the training/test performance results enables us to better compare how the LPRS results generalize to new data.

In the CM-ALL experiment (Table 2), we were only concerned with class discrimination performance. Given that the classes were nearly equally represented, the agents were able to marginally improve the apriori class probability ($\approx 50\%$), with a training classification accuracy SM-CM-ALL and RB-CM-ALL of 65% and 69% respectively. For the test data, these percentages decreased to 54% and 59%, respectively. For these cases, the discrimination performance did not generalize well and the classification accuracy on the test data was barely better than guessing.

For the P-SEL cases, the evolved agents also did poorly in terms of training classification, with accuracy rates of SM-P-SEL and RB-P-SEL of 50% and 55%, respectively. However, unlike the CM-ALL approach, these rates generalized well with the test data. Here, the bias was heavily skewed in favor of classifying cases as REC. For this case, fitness was measure in terms of profit. In terms of the increase in overall portfolio value, this approach achieved better fitness than both Random and the S&P500. This indicates that the LPRS strategy is more profitable if the agent is biased toward stock recovery.

The best overall results were achieved with P-Vote approach. As Figure 3 shows, these LPRS agents did well in terms of profit on the training data and as well, or better, on the test data (generalized well). For these two cases, the RB-P-Vote did slightly better than the SM-P-Vote agent. Based on the classification accuracy, it appears that both these agents are biased toward classifying stocks as REC. This characteristic supports the counter-intuitive assertion that accurate class discrimination is not essential to good portfolio performance.

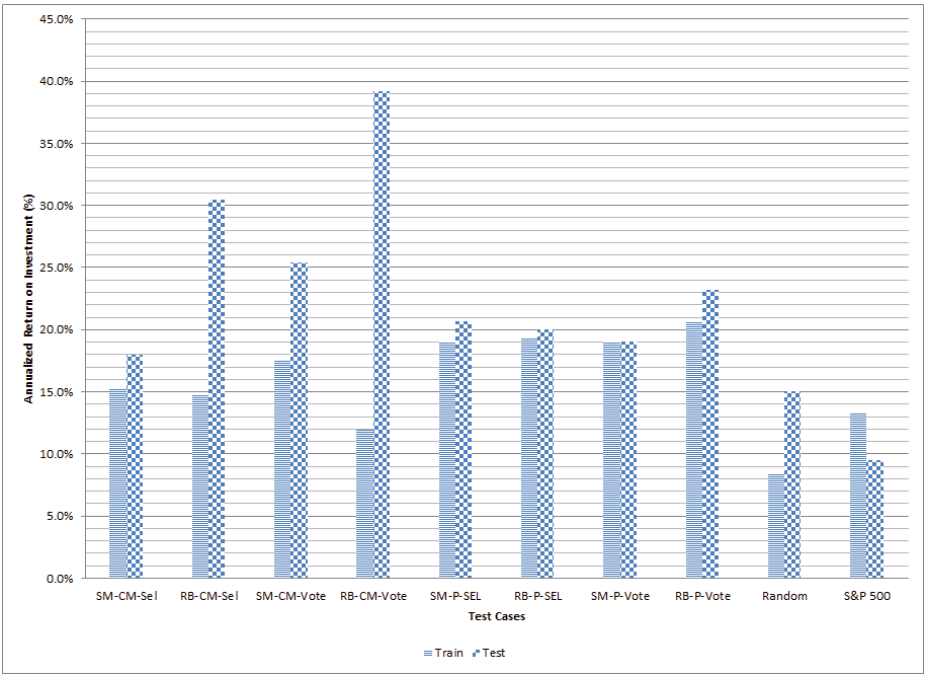

Fig. 3. Comparison View of Annualized ROI Results

For the most part, the Random approach was outperformed by all the LPRS agents. That being said, the SM-CM-Sel results (Table 3) did not convincingly outperform Random on the test data, especially when taking the large fitness standard deviations into account. Likewise, the results for SM-P-SEL cases (Table 5) were only marginally better than the Random results. The performance of the S&P500 index was competitive with the LPRS agents on the training data. However, with regard to the test data, all the LPRS approaches (including the Random approach) outperformed this benchmark index by substantial margins.

These results also yielded a number of surprises. For the RB-CM-SEL case, Buying Rule A (from section 2.3) was very prominent in the evolved solutions, appearing repeatedly. In fact, only two distinct solutions were involved. This phenomenon indicates that Buying Rule A is a very powerful discrimination rule. Another surprise was that the evolved LPRS agents consistently performed better on test cases than on the training cases.

5 Summary

In this paper, we investigated a number of different approaches for evolving agents to conduct a Low Price Recovery Strategy for stock trading. We targeted

the LPRS because our observation had been that stocks experiencing major declines will often bounce back in a relatively short period of time. This phenomenon has been shown to be fairly common in our research, with nearly 50% of the candidate stocks recovering. Given this, our evolved LPRS agents were all able to beat a strategy of buying random candidates. Additionally, their performance also bested the S&P500 index, which is considered an important benchmark when evaluating the performance of mutual stock funds. In general, the best LPRS performance was observed when the agents were trained to maximize the return of the portfolio. From a profit perspective, this approach proved superior to simply training agents to discriminate between the REC and NoREC classes. Within those subgroups, the majority rules strategy yielded the best results, with the agent evolved for the RB-P-Vote case having the best overall performance, both in terms of fitness and generalization of results (from training to test). It was also found that the absolute decline in stock price (Buying Rule A) was an especially important factor in choosing stocks that would eventually recover.

This research has demonstrated the utility of evolving agents for specific trading strategies. The fact that these agents can outperform benchmarks like the S&P500 index is especially promising. Our future work in this area will focus on variations on this technique to determine if performance can be improved still further. In particular, instead of a binary (yes/no) trading decision, we want to use the output of the evolved function as a basis for prioritizing buying opportunities. Additionally, we will continue to investigate popular sentiment metrics which potentially have short-term predictive value for investments.

References

1. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules. Journal of Financial Economics **51**(2), 245–271 (1999)
2. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic programming: an introduction, vol. 1. Morgan Kaufmann San Francisco (1998)
3. Becker, L.A., Seshadri, M.: Gp-evolved technical trading rules can outperform buy and hold (2003)
4. Bollen, J., Mao, H., Zeng, X.: Twitter mood predicts the stock market. Journal of Computational Science **2**(1), 1–8 (2011)
5. Chen, S.-H.: Genetic algorithms and genetic programming in computational finance. Springer Science & Business Media (2012)
6. Enke, D., Thawornwong, S.: The use of data mining and neural networks for forecasting stock market returns. Expert Systems with Applications **29**(4), 927–940 (2005)
7. Golberg, D.E.: Genetic algorithms in search, optimization, and machine learning, 1st edn., vol. 1989 (1989)
8. Kimoto, T., Asakawa, K., Yoda, M., Takeoka, M.: Stock market prediction system with modular neural networks. In: 1990 IJCNN International Joint Conference on Neural Networks, pp. 1–6. IEEE (1990)
9. Kirillov, A.: Aforge.net framework (2013)

10. Kohavi, R., Provost, F.: Glossary of terms. Machine Learning **30**(2–3), 271–274 (1998)
11. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press (1992)
12. Koza, J.R.: Survey of genetic algorithms and genetic programming. In: Wescon Conference Record, pp. 589–594. Western Periodicals Company (1995)
13. Lewis, R.J.: An introduction to classification and regression tree (cart) analysis. In: Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, California, pp. 1–14 (2000)
14. Nuij, W., Milea, V., Hogenboom, F., Frasincar, F., Kaymak, U.: An automated framework for incorporating news into stock trading strategies. IEEE Transactions on Knowledge and Data Engineering **26**(4), 823–835 (2014)
15. Potvin, J.-Y., Soriano, P., Vallée, M.: Generating trading rules on the stock markets with genetic programming. Computers & Operations Research **31**(7), 1033–1047 (2004)
16. Rosenberg, C.N.: Stock Market Primer. Grand Central Publishing (1991)
17. Vu, T.-T., Chang, S., Ha, Q.T., Collier, N.: An experiment in integrating sentiment features for tech stock prediction in twitter (2012)
18. Whitley, D.: A genetic algorithm tutorial. Statistics and Computing **4**(2), 65–85 (1994)