Robust Stock Value Prediction using Support Vector Machines with Particle Swarm Optimization

Trevor M. Sands, Deep Tayal, Matthew E. Morris, and Sildomar T. Monteiro
Electrical and Microelectronic Engineering
Rochester Institute of Technology
Rochester, NY 14623 USA
{tms6485, dt2804, mem9836, sildomar.monteiro}@rit.edu

***Abstract*—Attempting to understand and characterize trends in the stock market has been the goal of numerous market analysts, but these patterns are often difficult to detect until after they have been firmly established. Recently, attempts have been made by both large companies and individual investors to utilize intelligent analysis and trading algorithms to identify potential trends before they occur in the market environment, effectively predicting future stock values and outlooks.**

In this paper, three different classification algorithms will be compared for the purposes of maximizing capital while minimizing risk to the investor. The main contribution of this work is a demonstrated improvement over other prediction methods using machine learning; the results show that tuning support vector machine parameters with particle swarm optimization leads to highly accurate (approximately 95%) and robust stock forecasting for historical datasets.

I. Introduction

Before automated systems, the financial industry was dominated by career businessmen and analysts; while today market data is almost entirely handled by computerized systems with minimal human intervention. While this means the modern market is incredibly efficient, it is also now a closed system, dominated by competing algorithms. For everyday investors, this makes the stock market difficult to reliably invest in. The goal of this paper is to propose a relatively simple and robust intelligent system that would allow for individual investors to make informed decisions based on historical market trends and predicted future trends.

In this paper, the current outlook of a particular stock on the public market will be described by a state machine; a Bull market describes a generally positive trend in the value of a particular stock, while a Bear market describes a generally negative trend. The current state of the market is evaluated by comparing a given trading day's adjusted closing price to that of the opening price.

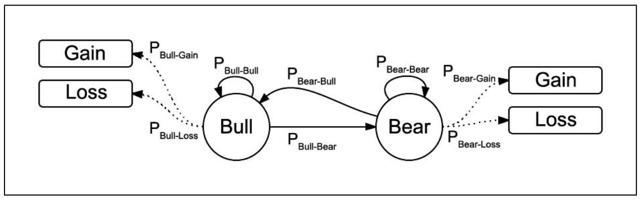

Fig. 1. Market state machine representation

Support vector machines (SVM) are a model used for classification and data analysis. SVM became a popular machine learning method in the late 1990s, spurred on by the need for a robust method for data classification [1], [2]. For the purposes of short-term stock value prediction, the support vector machine model will be boosted by particle swarm optimization (PSO) [3]. The performance of the proposed method is compared to the naïve Bayes classifier and artificial neural network models in terms of prective accuracy and computation time required.

II. Related Works

Attempting to apply automated analysis and trading systems to the public stock market has been gaining popularity since the early 1980s, where dedicated traders and analyists would design algorithms to watch for major fluctuations in individual stock prices. Since then, algorithms for identifying and expoiting market trends have become far more sophisticated; through the use of classification and regression machine learning methods, researchers have been able to uncover some of the underlying structure in the highly fluid market [4].

Xia, Liu, and Chen [5] propose using support vector machines for the purposes of building a regression model of historical stock data. This work identifies one of the key benefits of SVM-based prediction models: minimized risk due to the optimality of the hyperplane margin. The regression model utilized is able to learn historical trends with nearly negligible mean-squared error and high correlation between values; however, the model fails to provide meaningful insight into potential future behaviors of a stock's value and thus cannot stand alone as a tool for the use of a beginning investor. Yang [6] suggests using support vector machines for the purposes of pattern recognition in stock trends, an approach that will be explored further in this work. The behaviors of a stock's price are categorized based on its difference between the opening and closing values in one of four ways: extremes (decidedly Bear or Bull market), neutral (little change between opening and closing), climbers (buyers dominate), and drifers (sellers dominate.) By identifying these trend archetypes, a SVM can be trained to recognize these patterns.

Ernawait and Subanar [7] utilize sequence matching techniques boosted by particle swarm optimization for market forecasting. Their application shows initial promise in determining market trends, but falls short in terms of predictive accuracy.

These works indicate that identifying trends in the market alone is not enough for effective forecasting of future values. The most promising results come from Yang's work with support vector machines [6], where a peak predictive accuracy of 70% is achieved. This work intends to improve upon this benchmark by applying particle swarm optimization to the support vector machine classification parameters.

III. Classifiers for Stock Market Prediction

A. Support Vector Machines

Support vector machines are supervised learning algorithms, used for the purpose of classification and regression analysis. The basic SVM is a non-probabilistic binary linear classifier.

The SVM takes into consideration the training data tied to a specific class and builds a model that assigns new data into one category or other. The SVM model is a representation of the data as points marked in space, mapped so that the data of the separate categories are divided clearly by a margin; the algorithm constructs a hyperplane which can be used for the purpose of classification or regression. For an input vector x, Equation (1) [2], describes the linear separation function for an empirically determined optimal separation parameter α_i and kernel function $k(x, x_i)$. A standard linear kernel function is of the form: $k(x, x_i) = x \cdot x_i$

$$h(x) = \text{sign}\,(\sum_i \alpha_i k(x, x_i)) \tag{1}$$

Boser, Guyon and Vapnik [1] suggested a way to create nonlinear classifiers by applying the so-called 'kernel trick,' based on Aizerman's work [8]. The resulting method has each dot product replaced by a nonlinear kernel function, defined in Equation (2), allowing the algorithm to fit the maximum-margin hyperplane in a transformed feature space. This is shown visually in Figure 2.

$$k(x, x_i) = (\gamma x x_i + b)^d \tag{2}$$

b: Kernel polynomial coefficient
d: Order of polynomial
γ: Moderator of sample influence

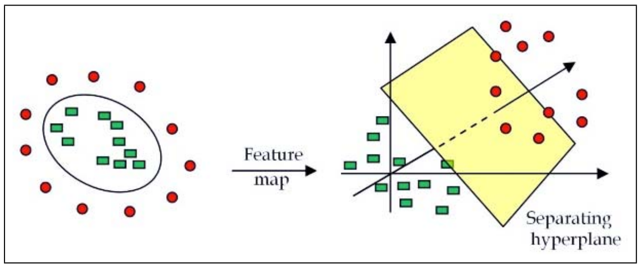

Fig. 2. Utilizing the kernel trick to transform non-seperable data into a higher-dimensional space [9]

In addition to linear and polynomial kernel functions, additional common functions include Gaussian radial basis functions and sigmoid functions. For this application, the polynomial kernel in Equation 2 will be utilized.

The baseline SVM method will utilize the least squares cost function, defined in Equation 3. The cost function measures the difference between training set input and target vectors, x and y respectively, when given a current estimate of the vector of parameters to be solved for, θ. The training set consists of n samples of dimension D.

$$J(x, y, \theta) = \frac{1}{n} \sum_{i=1}^{n} (\theta x_i - y_i)^2 \tag{3}$$

The cost function in Equation 3 can be modified to include a 'balancing' parameter, C. This parameter dictates the balance between a model's complexity and the degree of allowable misclassifcation. The modified cost function is shown in Equation 4, assuming the output of $J(x, y, \theta)$ is constrained to two classes by a sigmoid hypothesis function. Note that C must be determined empirically.

$$J(x, y, \theta, C) = \min_\theta C[\frac{1}{n} \sum_{i=1}^{n} (\theta x_i - y_i)^2] + \frac{1}{n} \sum_{j=1}^{D} \theta_j^2 \tag{4}$$

B. Particle Swarm Optimization

Particle swarm optimization is a stochastic approach for solving continuous and discrete optimization problems. Using this optimization technique, simple software agents, called particles, move in the search space of an optimization problem; the position of a particle represents a candidate solution to the optimization problem at hand. Each particle searches for more optimal positions in the search space by changing its velocity according to rules originally inspired by behavioral models of bird flocking.

The PSO algorithm is population-based: a set of potential solutions develops to approach an appropriate solution (or set of solutions) for a problem. Being an optimization method, the goal is finding the global optimum of a real-valued fitness function defined in a given search space. The social representation that led to this algorithm can be summarized as follows: the individuals that are part of a society hold a view that is part of a 'belief space' (search space) shared by every possible individual. Individuals may modify this 'opinion state' based on three factors: the optimality of the environment (i.e. its fitness value), the values of previously occupied states, and the values of neighboring states [3].

Particles have a limited memory, which allows them to revert to previous and more advantageous positions. Each particle's movement is the combination of its own initial velocity and the two randomly weighted influences: individualism, the tendency to return to its best previous position, and the tendency to move towards the neighborhood's best previous position [10]. This movement is expressed mathematically in Equation 5 [11], [12]:

$$v_{id}^{t+1} = w v_{id}^t + c_1 \psi_1 (p_{id}^t - x_{id}^t) + c_2 \psi_2 (p_{gd}^t - x_{gd}^t) \tag{5}$$

v_{id}^t: Velocity component in the d dimension of the i^{th} particle
x_{id}^t: Position component in the d dimension of the i^{th} particle

c_1,c_2: Individual and social constant weighted factors
p_i: Best position achieved thus far by the particle
p_g: Best position achieved thus far by the particle's neighbors
ψ_1,ψ_2: Random weight factors between 0 and 1
w: Inertia weight

C. Artificial Neural Network

Artificial neural networks (ANN) behave in a similar manner to their biological counterparts, where the neuron is given an evaluation function and a weighted set of input values. Based on these parameters and inputs, the neuron will decide whether or not to activate. Haykin [13] mathematically represents the weighted input and activation function relationship in the following manner:

$$x_i = \sum_{j=0}^{n} W_{j,i}, a_j \tag{6}$$

x_i: The i^{th} input from data series x
$W_{j,i}$: The weight associated with j past inputs up to and including input i
a_j: Activation response at node j

The activation response, a_j, at each node is calculated by an activation function as applied to the input in the series:

$$a_i = gx_i = g(\sum_{j=0}^{n} W_{j,i}, a_j) \tag{7}$$

g: Threshold function, typically a $sign$ or a $sigmoid$ function is used

The initial bias weight, $W_{0,j}$ determines the threshold between the neuron firing or not, as a_0 determines what value the threshold function, g uses as a center. Generally, $sign$ and $sigmoid$ functions are centered about 0, so $a_0 = -1$ is a typical bias threshold; that is: if $a_i > -1$, the neuron will activate.

The strength of artificial neural networks is due to the randomly generated interconnections between layers, where a typical parameter to optimize for is the sum of squared errors (SSE) between the target and actual output values. The neural network will reconfigure itself with the goal to optimize this target performance parameter. The validation and testing sets allow for the net to assign weights associated with each input [14]. During the testing phase, the net attempts to minimize the SSE by using the trained weights; if the network performance does not improve between subsequent trials, then the weights are reconfigured.

Even when considering small time periods, stock value approximation functions are nonlinear by nature. In order to account for the nonlinearity of stock values over time, a nonlinear autoregressive network (NARX) structure is employed, which is comprised of an external input, target input values, and a time delayed feedback line; the NARX structure is shown in Figure 3 While in the closed-loop configuration, values are taken from the output layer and are fed back to the input layer with a finite time delay along with an external input value array. The network is trained and tested under open-loop conditions [15], [16].

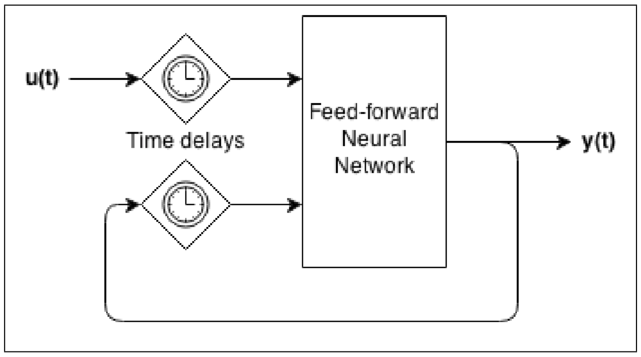

Fig. 3. Nonlinear autoregressive network configuration

D. Naïve Bayes Classifier

The naïve Bayes classifier (NBC) is a method for classifying a subject based on given feature attributes, which are assumed to be independent of one another (thus the classifier is 'naïve' to the features' relationships.) The naïve Bayes classifier exploits Bayes' theorem to approximate the prior probability of the class. These features are assumed to be independent of one-another, resulting in the decision rule highlighted in Equation (8) [17].

$$NBC(f_1, \ldots, f_n) = \arg\max_{c} P(C = c) \prod_{i=1}^{n} P(f_i|c) \tag{8}$$

The naïve Bayes classifier is suitable for the purposes of stock value prediction for the following reasons: prior knowledge of a stock's behavior can be utilized by probablistic methods; requires little computation time for large datasets due to linear classifier nature; retraining only requires updating the prior probabilities when new data becomes available.

While the NBC offers a number of desirable features for acting as a baseline comparison method, the assumptions it makes about the independence of features in stock data (opening price, closing price, volume, etc.) are likely to negatively impact the accuracy of the value prediction.

IV. Experiments

In order to provide a meaningful, real-world implementation of the proposed prediciton methods, two well-known companies with public stock options are selected: Apple, Inc. (AAPL) and Google (GOOG.) In addition to being giants of the consumer electronics industry, both Apple and Google are immensely popular 'big-ticket' stocks for traders. These two stock profiles will allow for practical tests of the predictive accuracy of the proposed methods. The stock data shown in Figure 4 and Figure 5 are publically available on websites such as Yahoo! Finance and Google Finance.

The methods are implemented in MATLAB utilizing nine years worth of daily stock data (opening price and adjusted closing price), spanning from January 2005 to March 2014.

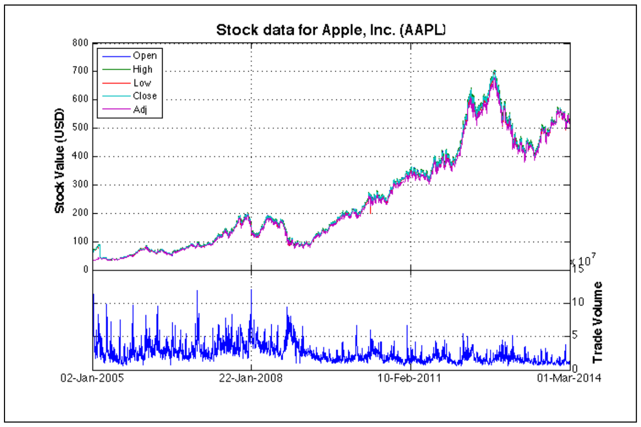

Fig. 4. Apple stock data for 2005-2014

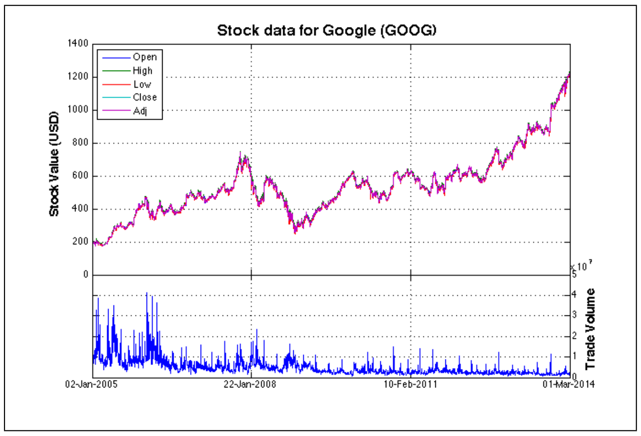

Fig. 5. Google stock data for 2005-2014

The data is separated into three sets for the purposes of cross-validation; this will help prevent the model from overfitting to the data.

The SVM method considers multiple features of the data: if a trading day results in a gain or loss, the maximum and minimum values of a stock in a single trading day, the difference between the closing price and adjusted closing price, and the volume of stocks traded on a particular day.

The proposed method of combining PSO with SVM will be compared against a standard SVM implementation using the least squares method (LS-SVM). PSO will be utilized to optimize the SVM parameters from Equations (2) and (4) for classification accuracy. These parameters will be left as typical values for evaluating performance with LS-SVM, shown in Table I, where N is the number of samples of the dataset.

TABLE I. LEAST SQUARES SUPPORT VECTOR MACHINE PARAMETERS

Parameter	Value
C	1
γ	N^{-1}
b	0
d	3

The particle swarm optimization parameters are detailed in Table II; these parameters are determined empircally after multiple trial runs on the given datasets. The PSO weights are resampled on every iteration of the classifier.

TABLE II. PARTICLE SWARM OPTIMIZATION PARAMETERS

Parameter	Value
c_1	12.6
c_2	1.5
Swarm size	20
Iterations	5

V. RESULTS

Table III shows the methods' average performances for the Apple dataset, Table IV shows the methods' average performances for the Google dataset. Accuracy is determined by comparing if a prediction results in a gain or a loss as compared to the historical ground truth.

TABLE III. AVERAGE PERFORMANCE RESULTS FOR APPLE DATASET

Algorithm	Accuracy (%)	Time (s)
NBC	49.4 ± 1.65	0.158
ANN	85.1 ± 0.263	20.4
LS-SVM	50.3 ± 1.57	0.163
SVM + PSO	96.1 ± 0.503	23.5

TABLE IV. AVERAGE PERFORMANCE RESULTS FOR GOOGLE DATASET

Algorithm	Accuracy (%)	Time (s)
NBC	47.7 ± 2.09	0.145
ANN	84.2 ± 0.343	24.2
LS-SVM	47.4 ± 2.57	0.327
SVM + PSO	97.5 ± 0.429	16.2

VI. DISCUSSION

For naïve Bayes classifiers, the inherent simplicity of the algorithm is both its greatest strength and most severe weakness. NBC and similar algorithms work well for predicting the output of a system or environment with a relatively small amount of input data; in these limited cases, the 'naïve' assumptions about the independence of individual elements does not greatly impact the probabilities of an output prediction matching a sampled output.

In time-series sets with a large amount of data, such as stock trends, the assumption of feature independence no longer holds approximately. From the results of Table III and Table IV, NBC is shown to be no more reliable than performing a coin-flip for predicting if tomorrow's close will result in a gain or a loss.

Given the popularity of constructing and optimizing artificial neural networks for the purposes of stock prediction, the accurate and consistent performance of the NARX configuration comes as little surprise. The inherent problems with using artificial neural networks stem from the "black-box" nature of the connections, meaning reconfiguring and retraining a network is more often than not a trial-and-error process for the user. Additionally, the ANN method took a fairly long time to complete, which is further exacerbated by the large datasets that market analysts often use.

The support vector machines algorithm alone yields poor results when its parameters are not carefully selected; this selection can be time-intensive for a user and also may require quite a bit of trial-and-error. For this reason, particle swarm optimization proves itself to be an extremely useful tool for selecting the C and γ values. By allowing PSO to automatically choose these values, the predictive accuracy of the system becomes very high, with little deviation in subsequent trials. This comes at the cost of computation time.

VII. Conclusion and Future Works

Attempting to identify historical trends in large datasets for the purpose of predicting future values is not a simple task, especially not for something as complicated as the 21st century's public market. Despite this complexity, technology has advanced sufficiently far enough to allow even an inexperienced stock trader with some technical knowledge to make a bit of extra income. In particular, support vector machines boosted by particle swarm optimization provide a relatively easy system to understand and implement for the purposes classifying stock performance and in predicting future outcomes.

The findings in this work show that the utilization of properly tuned particle swarms to optimize support vector machine parameters yields a highly accurate predictor for stock behavior prediction. These predictions are reliable for the short term, which is useful for casual interday traders.

This comparison has given an informative insight into the world of algorithm-based financial endeavors and opens the door for future machine learning methods to excel in the financial market. Future extensions to increase the robustness of this system include: the ability to update inputs based on current market values; dynamic gain or loss magnitude prediction, the ability to select multiple stocks or options to track, and the ability to include public information feeds (Twitter, Wall Street Journal, etc.) Additionally, improvements can be made to the particle swarm optimization algorithm to reduce computation time. Such an improved system would be a powerful tool in the hands of an investor.

Acknowledgment

The authors would like to thank R. Ptucha for his help in the preparation of this work. Additionally, the authors would like to thank R. Bowen, S. Kumar, and L. Tanwar for their constant support and input in developing the source code for testing the algorithms presented in this work.

References

[1] B. E. Boser, I. M. Guyon,and V. N. Vapnik, *A training algorithm for optimal margin classifiers*, Proceedings of the fifth annual workshop on Computational learning theory, 1992, 144-152.

[2] B. Schölkopf, P. Simard, A.J. Smola, and V. Vapnik, *Prior knowledge in support vector kernels*, Advances in neural information processing systems, 1998, 640-646.

[3] J. Kennedy and R. C. Eberhart, *Particle swarm optimization*, Proceedings of the 1999 Congress on Evolutionary Computation, 1995, 1942–1948.

[4] R.G. Palmer, W. B. Arthur, J. H. Holland, B. LeBaron, and P. Tayler, *Artificial economic life: a simple model of a stockmarket*, Physica D: Nonlinear Phenomena, vol. 75, 1994, 264-274.

[5] Y. Xia, Y. Liu, and Z. Chen, *Support Vector Regression for prediction of stock trend*, 6th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII), 2013, vol. 2, 123-126.

[6] Y. Yang, *Pattern Recognition Based on Support Vector Machine: Computerizing Expertise for Predicting the Trend of Stock Market*, WRI World Congress on Computer Science and Information Engineering, 2009, vol. 6, 60-66.

[7] Ernawati and Subanar, *Using particle swarm optimization to a financial time series prediction*, International Conference on Distributed Framework and Applications (DFmA), 2010, 1-6.

[8] M. A. Aizerman, E.A. Braverman, and L. Rozonoer, *Theoretical foundations of the potential function method in pattern recognition learning*, Automation and Remote Control, 1964, 821-837.

[9] M. Elmezain, A. Al-Hamadi, O. Rashid, B. Michaelis, *Advanced Technologies*, 2009.

[10] Y. Shi and R. C. Eberhart, *Empirical study of particle swarm optimization*, Proceedings of the 1999 Congress on Evolutionary Computation, 1999, vol. 3.

[11] Y. Shi and R. C. Eberhart, *A modified particle swarm optimizer*, IEEE International Conference on Evolutionary Computation Proceedings, 1998, 69-73.

[12] H. J. Escalante, M. Montes, and L. E. Sucar, *Particle Swarm Model Selection*, Journal of Machine Learning Research, 2009, 405-440.

[13] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2 ed., Prentice Hall PTR, 1998.

[14] K. Schierholt and C. H. Dagli, *Stock market prediction using different neural network classification architectures*, Proceedings of the IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering, 1996, 72-78.

[15] M. C. Medeiros, T. Teräsvirta, and G. Rech, *Building Neural Network Models for Time Series: A Statistical Approach*, SSE/EFI Working Paper Series in Economics and Finance, No. 508, 2002.

[16] J. T. Connor, R. D. Martin, and L. E. Atlas, *Recurrent Neural Networks and Robust Time Series Prediction*, IEEE Transactions on Neural Networks, Vol. 5, No. 2, 1994, 240-254.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag New York, Inc., 2006.