LG-Trader: Stock trading decision support based on feature selection by weighted localized generalization error model

Wing W.Y. Ng\textsuperscript{a}, Xue-Ling Liang\textsuperscript{a}, Jincheng Li\textsuperscript{a,b,*}, Daniel S. Yeung\textsuperscript{a}, Patrick P.K. Chan\textsuperscript{a}

\textsuperscript{a} Machine Learning and Cybernetic Research Center, School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
\textsuperscript{b} College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, China

\textbf{Abstract}

Stock trading is an important financial activity of human society. Machine learning techniques are adopted to provide trading decision support by predicting the stock price or trading signals of the next day. Decisions are made by analyzing technical indices and fundamental analysis of companies. There are two major machine learning research problems for stock trading decision support: classifier architecture selection and feature selection. In this work, we propose the LG-Trader which will deal with these two problems simultaneously using a genetic algorithm minimizing a new Weighted Localized Generalization Error (wl-GEM). An issue being ignored in current machine learning based stock trading researches is the imbalance among buy, hold and sell decisions. The feature selection based on wl-GEM helps to select most useful technical indices among choices for each stock. Experimental results demonstrate that the LG-Trader yields higher profits and rates of return in both stock and index trading.

© 2014 Elsevier B.V. All rights reserved.

\section{Introduction}

It is impossible to have full knowledge about changes of stock market in real time, so making correct investment decision is a challenging and meaningful task for machine learning researchers. According to the Efficient Market Hypothesis (EMH), stock prices always reflect all information in real time immediately \cite{1}. One cannot make correct prediction on stock market to gain extra returns consistently. So, traditional wisdom finds that the buy-and-hold strategy yields consistent profit when the market is in uptrend overall. However, there is no guarantee for the market to keep in uptrend everyday. A variety of methods and techniques have been proposed to forecast the stock market for obtaining high profits in recent decades and have been proved that underlying assumptions of the EMH turns out to be unrealistic in many cases \cite{2, 3}.

Current machine learning based methods focus on the use of historical information, e.g. technical indices, to predict future stock price or trend. Some researchers predict the future stock price based on the time series of previous close prices \cite{4, 5}. But, forecasting the exact stock price is difficult and may not be necessary to make profits from the stock market. Profit can be made by simply predicting the time when the stock price turns from downtrend to uptrend or vice versa. So, authors in \cite{6} combine the MultiLayer Perceptron Neural Network (MLPNN) and the case based reasoning method for turning point prediction of stock price. However, not all turning points in the time series of stock price represent a proper trading opportunity. Authors of \cite{7} propose an improved trading strategy designing approach based on a turning point prediction to find real trading points.

This is no doubt that finding out proper turning points for the optimal trading opportunity is more direct and feasible for stock investments. However, common phenomenon being ignored is that both buy and sell decisions occur much less frequent when compared to the hold decisions. The imbalance between hold and trade (buy or sell) exists in stock investment environment naturally and causes an immeasurable impact to the machine learning based method for stock trading decision prediction. In current literatures, the imbalance problem is commonly ignored. Prediction system built with this ignorance is likely to be biased to classify samples into the hold decision which hinders the correct judgment on other minority classes, i.e. buy or sell. Therefore, such imbalance problem should be taken into account when we construct the prediction system to increase its capability.
The problem of imbalance between classes will lead to a biased classifier because the number of samples in the majority class dominates the whole training dataset. No learning for minority classes is needed and just predicting all samples to the majority class is enough to yield a very high training accuracy. Therefore, we are motivated to propose a new trading decision support algorithm based on the Weight Localized Generalization Error Model (WL-GEM). The WL-GEM provides a cost sensitive based objective function to balance the error made by each class according to their imbalance ratios. The algorithm will predict the trading suggestion for the next day based on technical indices and prices in previous days. Furthermore, owing to the large number of technical indices available and variety of choices, both MLPNN architecture and its input features (indices, prices and volume in previous days) are selected using the WL-GEM via a Pareto-based multi-objective optimization method.

The rest of the paper is organized as follows: a literature survey on stock market prediction methods is given in Section 2. Section 3 describes the new turning point prediction algorithm, i.e. the LG-Trader. The experimental results and discussion will be shown in Section 4. Finally, we conclude this work in Section 5.

2. Related works

With the rapid development of machine learning techniques in recent decades, many different classifiers and algorithms have been proposed or applied to provide decision support to human stock trading investors: such as radial basis function neural network [8], genetic network programming [9] and support vector machines [10], etc. Most of them apply a standard classifier directly with input features (e.g. technical indices) selected by authors with experience. For the purpose of pursuing higher returns in stock investment, researchers have proposed a variety of algorithms and techniques to construct prediction models with high capabilities. Different researchers concentrate on different forecasting targets. A measurement is also proposed in [8] for tracing long term trend of stock market. Some researchers only focus on the prediction of price fluctuation [11] and attempt to achieve precise models to predict future prices of stocks. In [12], fuzzy models combined with coevolutionary methods are applied for stock price forecasting. On the other hand, an integrated independent component analysis based denoising scheme with neural network for stock price prediction is proposed in [13].

The forecasting of future stock prices is one of the most challenging tasks in stock market because stock prices are inherently noisy and non-stationary. More importantly, predicting the exact stock price in future may not be necessary to make profits from the stock market. Instead, turning points in the time series of stock prices are treated as optimal opportunities of stock trading turning point classiﬁcation in stock trading. The LG-Trader is a two-phase algorithm to select optimal input features (phase 1) and architecture of MLPNN (phase 2) for trading decision classiﬁcation. Fig. 1 shows the ﬂowchart of the LG-Trader. Both phases are conducted via a multi-objective optimization of WL-GEM. In this work, the improved Non-dominant Sorting Genetic Algorithm (NSGA-II) [27] is adopted as the Pareto-based multi-objective optimization method for both phases. The WL-GEM will be derived in Section 3.1. Sections 3.2 and 3.3 will introduce the optimization of feature selection and architecture, respectively. Optimizing both features and architecture separately may yield sub-optimal solution. However, optimizing them boundary of the prediction model is optimized by a roulette wheel based genetic algorithm with elitism.

To analyze the stock market comprehensively, researchers tend to adopt different information from historical stock data. There are plenty of input features or indices available for describing a stock from its historical data. The time series of stock prices is the most widely used data in this field [4,5,12,13]. Technical analysis is an important tool being used to predict stock price in both machine learning methods and human investors [10,14,15]. Some popular technical indices reﬂect the future trend of the dynamic stock market well. For example, candlestick patterns [16] are often applied to ﬁnd the optimal buy and sell opportunities. The crossover of moving average curves over different numbers of days is another commonly adopted decision supporting index for both machine learning and human investor [17]. There are a lot of different types of technical indices and different periods for computing different technical indices, so researchers select technical indices using either feature selection techniques via fractal feature selection or evolutionary computation methods [10,18]. In [23], a stepwise regression analysis (SRA) is applied to select important input features. The SRA relies on correlation between inputs and outputs which may ignore nonlinear relationship between inputs and outputs.

Although holding and trading (buy and sell) signals are inherently imbalance, current machine learning based stock trading researches usually ignore this problem. Authors in [23] have not considered the imbalance problem directly, but they applied an output threshold moving based on GA. In imbalance classiﬁcation researches, threshold moving is applied to relief the imbalance issue by moving the decision boundary towards the majority class. The GA adopted in [23] does not provide this functionality directly. Our previous work [26] shows the attempt of applying GA with cost sensitive based objective function trains better MLPNN in dealing with imbalance trading problems. However, in [26], we only select the architecture of MLPNNs for stocks which performances still heavily depend on the choice of input features. Therefore, in this work, we propose a two-phase genetic algorithm based optimization on the WL-GEM to solve feature selection, architecture selection and imbalance problems simultaneously for turning point classiﬁcation in stock trading.

3. LG-Trader

The LG-Trader is a two-phase algorithm to select optimal input features (phase 1) and architecture of MLPNN (phase 2) for trading decision classiﬁcation. Fig. 1 shows the ﬂowchart of the LG-Trader. Both phases are conducted via a multi-objective optimization of WL-GEM. In this work, the improved Non-dominant Sorting Genetic Algorithm (NSGA-II) [27] is adopted as the Pareto-based multi-objective optimization method for both phases. The WL-GEM will be derived in Section 3.1. Sections 3.2 and 3.3 will introduce the optimization of feature selection and architecture, respectively. Optimizing both features and architecture separately may yield sub-optimal solution. However, optimizing them...
together will take much longer training time because of much larger search space. Stock market changes quickly and usually a large number of stocks are under investigation in the same time, so fast training of turning point classifier is needed. Experimental results also show that the current approach yields better profit making ability in comparison to current state-of-the-art research results. Readers may optimize them simultaneously by a simple re-arrangement of the chromosome of the genetic algorithm to obtain optimal classifier for both tasks simultaneously, if time allows.

In this work, we use several commonly used technical indices, stock price and volume within a day as candidate feature set. In the LG-Trader, we use feature collected at day $t$ to predict the trading opportunity of day $t+1$.

The L-GEM model in [19] treats all misclassification to have the same cost. To deal with the imbalance problem between holding and trading decisions, the L-GEM model is modified to yield different costs for misclassification of different classes. Higher costs are assigned to misclassifications of buy and sell classes as hold class. Finally, the LG-Trader gives trading suggestion to user based on the classification of the MLPNN.

3.1. Weighted L-GEM for MLPNN

The wL-GEM is derived in Section 3.1.1 and we will discuss why wL-GEM is useful to improve the generalization capability of classifier for trading decision support in 3.1.2.

3.1.1. Derivation of the wL-GEM for MLPNN

Training a MLPNN ($g(.)$) minimizing the training error only easily suffers from overfitting. Therefore, we propose the L-GEM to train a MLPNN minimizing both the training error and the stochastic sensitivity (ST-SM) [24]. The purpose of minimizing the ST-SM is to minimize the fluctuation of MLPNN output with respect to small input changes. However, one cannot minimize both the training error and the ST-SM. Therefore, we apply a Pareto based multi-objective optimization to minimize the L-GEM. This follows the bias/variance dilemma that a classifier yielding good generalization capability should have a good balance between its error and variance of outputs.

Given a training dataset $D = \{x_b, G(x_b)\}_{b=1}^{N}$ where $N$, $x_b$ and $G(x_b)$ denote the number of training samples, the input feature vector and true classification of $x_b$, respectively. A MLPNN is defined as:

$$z_b = g_b(x) = f(\text{net}_b(x))$$  

$$\text{net}_b(x) = \sum_{j=1}^{M} w_{bj} f\left( \sum_{i=1}^{n} w_{ij} x_{i} \right) = \sum_{j=1}^{M} w_{bj} f(\text{net}_j(x))$$

$$f(y) = \frac{1}{1 + \exp(-y)}$$

where $M$, $n$, $g(.)$, $w_{bj}$ and $w_{ij}$ denote the number of hidden neurons in the hidden layer, the number of neurons in the input layer, the $k$th output of the MLPNN, a connection weight between output and hidden layers and a connection weight between input and hidden layers. In a standard MLPNN training, mean square error between $G$ and $g$ for all training samples is minimized using a back-propagation algorithm. The training error of a MLPNN ($R_{\text{emp}}$) is defined as follows:

$$R_{\text{emp}} = \frac{1}{N} \sum_{b=1}^{N} (G(x_b) - g(x_b))^2$$

For a given problem, future unseen samples could be expected to deviate from training samples no more than a particular percentage or value. We denote the maximum expected deviation by $Q$ which is a user defined variable. For a given $Q$ value, the Q-Neighborhood of a training sample is defined as follows:

$$S_Q(x_b) = \{x|x = x_b + \Delta x : \max_{i=1}^{n} |\Delta x_i| \leq Q\}$$

The mean square error of a MLPNN for unseen samples within a union of Q-Neighborhood (i.e. Q-Union) is defined as follows.

$$w_{\text{emp}}^Q(Q) = \frac{1}{N} \sum_{b=1}^{N} C(x_b) \int_{S_{Q(b)}} (G(x_b) - g(x_b))^2 p(x) dx$$

With probability $1 - \eta$, the L-GEM ($w_{\text{SM}}(Q)$) of a MLPNN is defined as follows:

$$w_{\text{SM}}(Q) = (\sqrt{w_{\text{emp}} + \sqrt{w_{\text{ESQ}}((\Delta x)^2)}} + A)^2 + \epsilon$$

The derivation of Eq. (7) from Eq. (6) follows steps as shown in [19] for the L-GEM. The wL-GEM of MLPNN ($w_{\text{Remp}}(Q)$) consists of three major components: weighted training error ($w_{\text{Remp}}$), weighted ST-SM ($w_{\text{ESQ}}((\Delta x)^2)$) and constants. Constants $A$ and $\epsilon$ depend on the choice of confidence of the bound ($\eta$) and characteristics of the training dataset. Therefore, the effective components in the $w_{\text{Remp}}$ are the weighted training error and the weighted ST-SM.

The L-GEM was firstly proposed for Radial Basis Function Neural Networks (RBFNN) [19] and had been extended to LS-SVM [32]. The L-GEM for RBFNN was used as a single objective function and has been applied to credit risk evaluation [31] and feature selection for pattern classification problems [30]. A combination of weighted training error and sensitivity measure has been applied as a single objective function for image steganalysis in [29]. In our previous works [16,19,20], the L-GEM is used as a single objective function to optimize network parameters. In this work, we formulate the MLPNN training problem as a multi-objective learning problem with two terms (training error and ST-SM) in the L-GEM as in [24]. The L-GEM in [24] is a standard L-GEM for training MLPNN for standard pattern classification problems while the wL-GEM in this work is designed for imbalance problems using a cost sensitive based L-GEM. Input features, architecture and connection weights of a MLPNN are optimized via a 2-phase Pareto based multi-objective minimization of training error and ST-SM in the wL-GEM as follows:

$$\min(\sqrt{w_{\text{Remp}}}, \sqrt{w_{\text{ESQ}}((\Delta x)^2)})$$

The training error of a MLPNN ($w_{\text{Remp}}(Q)$) is defined as follows:

$$w_{\text{Remp}}(Q) = \frac{1}{N} \sum_{b=1}^{N} C(x_b)(G(x_b) - g(x_b))^2$$

The $C(x_b)$ is the cost matrix when a sample belongs to class $i$ is misclassified as class $j$. Costs are computed by the ratio of each class in the training set. For example, if the ratio of each class in the training samples is $F_{\text{sell}}:F_{\text{buy}}:F_{\text{hold}}$, then the cost matrix can be derived as follow:

$$C(x) = C(G(x), g(x)) = \begin{bmatrix} 0 & F_{\text{sell}} & F_{\text{buy}} \\ F_{\text{sell}} & 0 & F_{\text{hold}} \\ F_{\text{buy}} & F_{\text{hold}} & 0 \end{bmatrix}$$

For a MLPNN with $k$ output neurons, the output perturbation of the $k$th output neuron of MLPNN is defined as follows:

$$\Delta z_k = g_k(x + \Delta x) - g_k(x)$$

Then, the weighted ST-SM is defined as the expectation of the squared difference between the output and its perturbation.

$$w_{\text{ESQ}}((\Delta z)^2) = \frac{1}{N} \sum_{b=1}^{N} C(x_b)E[(g_k(x_b + \Delta x) - g_k(x_b))^2]$$
The hyper-rectangle model of computing ST-SM in [21] requires a very high time complexity for problems with large numbers of input features. So, we propose a Quasi-Monte Carlo (QMC) based method to calculate the ST-SM of MLPNN. The QMC method is efficient in computing high dimensional hyper-rectangle regions. Here, we adopt an n-dimension Halton Sequence [22] to compute the ST-SM of an output neuron of MLPNN as shown in Algorithm 1. If the MLPNN has multiple output neurons (e.g. 3 for the LG-Trader for each decision), the ST-SM of each output neuron is computed by Algorithm 1 individually. Then, the average of all ST-SM yields the final ST-SM of the whole MLPNN.

**Algorithm 1. Computation of ST-SM**

1. Step 1: Generate $H$ Halton points $\Delta x_h \in R^n$, $h = 1, ..., H$ with each coordinate range from $[-Q, Q]$;
2. Step 2: Compute the ST-SM for each training sample $x_i$:
   $$ F_i(x_h) = \frac{1}{H} \sum_{h=1}^{H} (g(x_h + \Delta x_h) - g(x_h))^2; $$
3. Step 3: Compute the ST-SM of output neuron:
   $$ wE_{x_h}((\Delta x)^2) = \frac{1}{N} \sum_{i=1}^{N} f(x_h)F_i(x_h) $$

3.1.2. Why wL-GEM can enhance the generation capability?

The wL-GEM is based on mean square error instead of classification error. However, as stated in [19,24], a classifier minimizing classification error directly is sensitive to small changes of inputs and may not generalize well. The major reason is that samples located near the decision boundary are sensitive and will be misclassified to another class when there is a small change in inputs. In contrast, a MLPNN trained via a minimization of MSE will minimize the difference between classifier’s real-value outputs and target class values. This creates a margin between two classes in the output space. Therefore, we train MLPNN in this work via a minimization of mean square error.

Fig. 2 shows distributions of training and testing samples of the stock AOU in Buy, Hold and Sell classes in a two dimensional representation (volume and open price). One could observe that most testing samples are located closely to training samples. One of the basic assumptions of machine learning is that training and testing samples are sampled from the same distribution. If training and testing samples are very dissimilar, machine learning methods may not generalize well. The major reason is that samples located near the decision boundary are sensitive and will be misclassified to another class when there is a small change in inputs. In contrast, a MLPNN trained via a minimization of MSE will minimize the difference between classifier’s real-value outputs and target class values. This creates a margin between two classes in the output space. Therefore, we train MLPNN in this work via a minimization of mean square error.

There are many different technical indices could be used to predict stock trends. In this work, we select a set of technical indices according to current literatures [23] and readers could change the set of indices by their own experiences. This set of indices serves as the initial full set of features for the NSGA-II to select the optimal feature set. There are 22 input features in total, including the open price (open), the close price (close), the highest price (high), the lowest price (low), the volume of trading and 17 technical indices being collected for day $t$.

The 17 indices are listed below:

(1) Moving averages (5MA, 10MA, 20MA)
   The moving average emphasizes the direction of trend. We use 5MA, 10MA and 20MA to capture 5-day, 10-day and 20-day moving averages, respectively. Moving averages with different number of days show different resolution of the current trend. The major drawback of moving average is the ignorance of the trading volume which may vary significantly across days.
   $$ nMA_t = \frac{\sum_{i=1}^{n} \text{Close}_i}{n} \quad (n = 5, 10, 20) $$
   where close$_i$ denotes the close price of the $i$th day.

(2) Bias (5BIAS, 10BIAS)
   5BIAS and 10BIAS measure differences between the close price and 5-day and 10-day moving averages, respectively.
   $$ nBIAS_t = \frac{\text{Close}_t - nMA_t}{nMA_t} \times 100\% \quad (n = 5, 10) $$

(3) Relative strength index (6RSI, 9RSI, 12RSI)
   RSI shows the ratio between averages of close price of rising and falling days. The close price of a rising (falling) day is

\[ \text{rise} \]
\[ \text{fall} \]

\[ \text{relative strength index} \]

\[ \text{averages} \]

\[ \text{ratio} \]
higher (lower) than that of the open price. It helps to
determine overbought and oversold conditions of a stock.
\[
n_{RS} = 100 \times \frac{100}{1 + n_{RS}^{10}} \quad (n = 6, 9, 12) 
\]
(15)
\[
n_{RS} = \frac{\text{rise}(n_{avg})}{\text{fall}(n_{avg})} \quad (n = 6, 9, 12) 
\]
(16)
where \(\text{rise}(n_{avg})\) and \(\text{fall}(n_{avg})\) denote the average of close
prices arising and falling days in a period \(n\) days before the
day \(t\), respectively.

(4) Nine day stochastic lines (\(K, D, J\))
The nine day stochastic lines consist of 3 different indices: \(K\),
\(D\) and \(J\), which use the price fluctuations to reflect the
strength of price movements, overbought and oversold.
\[
R_{SV} = \frac{\text{close}_{t} - \min \{\text{low}_{t-1}, \ldots, \text{low}_{t-n}\}}{\max \{\text{high}_{t-1}, \ldots, \text{high}_{t-n}\} - \min \{\text{low}_{t-1}, \ldots, \text{low}_{t-n}\}} \times 100 
\]
(17)
\[
K_{t} = \frac{2}{3} \times K_{t-1} + \frac{1}{3} \times R_{SV}_{t} 
\]
(18)
\[
D_{t} = \frac{2}{3} \times D_{t-1} + \frac{1}{3} \times K_{t} 
\]
(19)
\[
J_{t} = 3D_{t} - 2K_{t} 
\]
(20)
The \(\text{low}_{t}, \text{high}_{t}\) denote the lowest price and the highest price
of day \(t\), respectively.

(5) 9-day moving average convergence and divergence (9MACD)
The 9MACD is based on moving averages, which shows the
difference between a “fast” exponential moving average
(EMA) and a “slow” EMA of close prices.
\[
D_{lt} = \frac{\text{high}_{t-1} - \text{low}_{t-1} + 2 \times \text{close}_{t}}{4} 
\]
(21)
\[
\text{EMA}(n)_{t} = \frac{1}{n} \sum_{i=n-1}^{t} D_{li} \quad (n = 12, 26) 
\]
(22)
\[
\text{DIF}_{t} = \text{EMA}(12)_{t} - \text{EMA}(26)_{t} 
\]
(23)
\[
\text{9MACD}_{t} = \frac{1}{n} \sum_{i=t-n-1}^{t} \text{DIF}_{i} 
\]
(24)

(6) Williams %R (10W%R, 12W%R)
Williams %R is used to show the estimated position of recent
close price with respect to the range between recent \(\text{high}\)
and \(\text{low}\) in \(n\) days.
\[
n_{W%R}_{t} = \max \{\text{low}_{t}, \text{low}_{t-1}, \ldots, \text{low}_{t-n+1}\} - \text{close}_{t} \]
\[
\min \{\text{high}_{t}, \text{high}_{t-1}, \ldots, \text{high}_{t-n+1}\} - \text{close}_{t} 
\]
(25)
(7) On-balance volume (OBV)
OBV is computed based on the cumulative volume of a stock.
The \(\text{SIGN}\) in the computation of OBV indicates the day \(t\) is a
rising day or falling day.
\[
\text{OBV}_{t} = \text{OBV}_{t-1} + \text{SIGN} \times \text{volume}_{t} 
\]
(26)
\[
\text{SIGN} = \begin{cases} 
+1 & \text{close}_{t} > \text{close}_{t-1} \\
0 & \text{close}_{t} = \text{close}_{t-1} \\
-1 & \text{close}_{t} < \text{close}_{t-1} 
\end{cases} 
\]
(27)
(8) Rate of change (SROC, 10ROC)
ROC provides the slope of the close price chart of a stock
with a step size of \(n\) days.
\[
\text{nROC}_{t} = \frac{\text{close}_{t} - \text{close}_{t-n}}{\text{close}_{t-n}} \quad (n = 5, 10) 
\]
(28)

The chromosome consists of two matrices: connection matrix
(Boolean) and weight matrix (real value). Both are \((N^{*} + M^{*} + K) \times
\((N^{*} + M^{*} + K)\) matrices, where \(N^{*}\), \(M^{*}\) and \(K\) denote the maximum
number of input features (i.e. 22 in this work), the number of
maximum hidden neurons and the number of classes (outputs),
respectively. For the connection matrix, 1 and 0 represent connect
and disconnect, respectively. When an element in connection
matrix is 0, the corresponding element in weight matrix will be
set to zero also. Otherwise, the real value in the weight matrix
represents the connection weight value of a corresponding weight.

In Phase 1, mutation is applied to the input features (neurons)
only. Mutations include adding a feature and deleting a feature
randomly. Mutation rate is 20% of the population. When a feature
needs to be deleted, its corresponding input neuron and all
connection weights will become zero in connection and weight
matrices. When a feature needs to be added, the corresponding
input neuron in the connection matrix will become one and values
of all connected connection weights will be randomly selected.
The rest of the Phase 1 is performed according to the standard
NSGA-II method. After iteration of generations, the solution yielding
the smallest \(\text{w}_{\text{best}}(Q)\) is selected as the final solution.
Algorithm 2 shows the detailed algorithm.

Algorithm 2. Phase 1 Feature Selection

Step 1: Initialize population. 50 individuals are generated
randomly using the chromosome scheme mentioned above.
Weight values are randomly generated from \([-1, 1]\).

Step 2: Fitness Calculation. Calculate the weighted ST-SM
and the weighted \(R_{emp}\) values of each individual (parent
generation).

Step 3: Crossover and Mutation. Crossover and mutation are
performed to create a new generation of population with 50
individuals. For each parent, mutation of either deleting a
feature or adding a feature is performed.

Step 4: Individuals Selection. Calculate the weighted ST-SM and
the weighted \(R_{emp}\) values of each offspring individual. Then,
the best 50 individuals among both the parent and the
offspring populations are selected by applying the non-
dominate sorting algorithm for the next generation (as the
new parent generation).

Step 5: Repeat Steps 1 to 4 for fifty times. Select the optimal
individual yielding the highest \(w_{best}\) so the best features
group is fixed finally.

3.3. Phase 2: Architecture selection

Phase 2 starts with the final solution from Phase 1. The set of
features selected in Phase 1 is treated as the full set of features in
Phase 2. The number of input neurons is reset to the number of
features being selected in Phase 1. In Phase 2, both input and
output neurons are fixed. Procedures of Phase 2 are the same as the
training algorithm being described in [24].

Fifty individuals with features selected in Phase 1 are initialized
randomly selecting all elements in both connection and
weight matrices. The multi-objective optimization for both Phase
1 and Phase 2 is the same. But, the set of mutation operations is
different in Phase 2: 1 mutation for the weight matrix and
4 mutations for the connection matrix. In each generation, one
of these 5 operations is randomly selected with an equal prob-
ability. For weight mutation, Gaussian mutation operator with
zero mean and variance equal to a “temperature” is used to
generate random numbers being added to weights in the MLPNN.
The “temperature” begins with a large magnitude and decreases
when the number of iterations increases. The other four mutation
operators are: add connections, delete connections, add a hidden
neuron and delete a hidden neuron. When the add (delete) connection operator is applied, a connection is added (deleted). The chances of adding or deleting a connection in input to hidden and in hidden to output layers are 20% and 5%, respectively. The algorithm is the same as feature selection being shown in Algorithm 2, but with Step 3 replaced by the aforementioned 5 operations.

4. Experiments and discussions

We will test the performance of LG-Trader for stock and index trading in Sections 4.1 and 4.2, respectively. We compare the proposed LG-Trader for stock trading with two turning point based trading methods in [7,23] in Sections 4.1.1 and 4.1.2, respectively. The Buy-and-Hold Strategy (B&H Strategy) will also be compared. Parameters of LG-Traders being used in experiments are shown in Table 1.

As in [7,23], the LG-Trader will make investment decisions (buy, sell, or hold) for the next day (day $t+1$) by using the data extracted from the current day (day $t$). Both buy and sell decision will be carried out at the $t+1$ day, so the price of buy or sell is the open price of the $t+1$ day. This is to simulate the real situation that trading immediately at the next opening of the stock market at $t+1$ day while LG-Trader yields either a buy or a sell decision. Similar to other literatures, the counting of $t$ does not include holidays of stock markets.

During experimental comparison, we use simple trading rules. Readers could use LG-Trader as a decision support system and make use of more sophisticated trading rules to increase profits. For all methods under comparison, each buy decision will purchase one share of the stock while each sell decision will sell out all shares being bought since last sell decision or the beginning of trading. Profits presented in Tables 2, 4 and 5 are computed by the difference between the amount of money received for selling the stocks and the accumulated amount of money for buying stocks including tax and handling fee during transactions. Rates of return per stock being shown in Tables 3-5 are computed by the following formula [23]:

$$Rate = \frac{\sum_{i=1}^{k} (1 - a - b) \times sell_i - (1 + a) \times buy_i \times 100\%}{(1 + a) \times buy_i}$$

where $a$, $b$, $k$, $sell_i$, and $buy_i$ denote the tax rate, the handling fee, the total number of transactions, the selling price of the stock in the $i$th transaction and the buying price of the stock in the $i$th transaction, respectively. In experiments, the training and the testing samples of AUO, EPSTAR, COMPAL, UMC, FOXLINK, SENAO, SIS and D-LINK are the same as in paper [23]. The training and testing samples of TESCO and DJIA are the same as in [7].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Quasi-Point ($H$)</td>
<td>50</td>
</tr>
<tr>
<td>GA Generation number ($L_{\text{max}}$)</td>
<td>50</td>
</tr>
<tr>
<td>Population size ($N_p$)</td>
<td>50</td>
</tr>
<tr>
<td>Minimum-Initial-Maximum Hidden neuron number for architecture mutation</td>
<td>5-10-30</td>
</tr>
<tr>
<td>$\text{Q-value (Q)}$</td>
<td>$0.1:0.01:0.2$</td>
</tr>
<tr>
<td>Local search Iteration ($L_{\text{max}}$)</td>
<td>30</td>
</tr>
<tr>
<td>Weight mutation initial temperature</td>
<td>0.02</td>
</tr>
<tr>
<td>Input to hidden layer connection mutation percentage</td>
<td>20%</td>
</tr>
<tr>
<td>Hidden to output layer connection mutation percentage</td>
<td>5%</td>
</tr>
</tbody>
</table>

Table 2

Features selected by the LG-Trader for different stocks and corresponding profits yielded by the LG-Trader.

<table>
<thead>
<tr>
<th>Stock</th>
<th>Features</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUO</td>
<td>Open, low, volume, 6RSI, 12RSI, 10W%R, 9MACD, OBV, 5ROC</td>
<td>76.03</td>
</tr>
<tr>
<td>EPSTAR</td>
<td>Open, 5BIAS, 12RSI, 9MACD, K, D, OBV, 10ROC</td>
<td>264.60</td>
</tr>
<tr>
<td>COMPAL</td>
<td>High, 10MA, 20MA, 5BIAS, 6RSI, 9RSI, 12W%R, 9MACD, K, OBV, 5ROC, 10ROC</td>
<td>25.18</td>
</tr>
<tr>
<td>UMC</td>
<td>Open, high, close, volume, 10MA, 6RSI, 9RSI, 10W%R, 12W%R, 5MACD, K, D, OBV, 5ROC</td>
<td>36.68</td>
</tr>
<tr>
<td>FOXLINK</td>
<td>Open, high, low, close, volume, 10MA, 20MA, 10BIAS, 6RSI, 9RSI, 12RSI, 9MACD, K, D, OBV, 5ROC</td>
<td>186.11</td>
</tr>
<tr>
<td>SENAO</td>
<td>High, low, close, 6RSI, 12RSI, 5ROC, 10W%R, 12W%R, 12W%R, K</td>
<td>30.05</td>
</tr>
<tr>
<td>SIS</td>
<td>Open, volume, 20MA, 9RSI, 10ROC, 12W%R</td>
<td>21.90</td>
</tr>
<tr>
<td>D-LINK</td>
<td>Open, low, volume, 5MA, 10MA, 20MA, 5BIAS, 9RSI, 12RSI, 5ROC, 12W%R, K, D, OBV</td>
<td>35.00</td>
</tr>
<tr>
<td>TESCO</td>
<td>Close, volume, 5MA, 5BIAS, 10BIAS, 6RSI, 10W%R, 12W%R, 9MACD, K</td>
<td>48.41</td>
</tr>
</tbody>
</table>
4.1. LG-Trader for stock trading

Table 2 shows feature sets selected by the LG-Trader for the stocks during the experimental comparison of [7,23]. The LG-Trader selects different features for different stocks or indices. Among 22 features, J, 10BIAS, 5MA and 10ROC are selected for 2 or less out of 9 stocks only. These show that these three features are not important for decision making for stock trading in our experiments. J is selected for FOXLINK only. J depends on both D and K, so it does not provide additional information to the LG-Trader. In contrast, K is selected for 7 stocks while open, volume, 6RSI, 12W%R and OBV are selected for 6 stocks. These features are important in decision making. K is a feature derived from the combination of high, low and close during 9 trading days and provides a relative view of the stock price with respect to maximum difference between open and close in 9 trading days. 12W%R computes has a similar formula with the K which shows that the close price with respect to price range of the stock in a recent time period is important. 10W%R and 12W%R only differ for 2 days in computation which may be overlapping heavily and such that 10W%R is only selected for 4 stocks. OBV is a feature derived from volume which shows that the volume of trading is important to trading decision. Volume provides strong complementary information to other important features because K, 6RSI and 12W%R ignore the volume of trading which is important indicator of the liquidity of the stock. Interestingly, the close price is not an important feature while it is always shown in financial charts for demonstration of trends of stocks. Widely used moving averages (20MA, 10MA and 5MA) are not important features for decision making in the LG-Trader. Among them, moving average over a long period (20MA) is more important than that over a short period (5MA). In contrast, 6RSI is more important than 9RSI and 12RSI.

In Section 4.1.1, the LG-Trader is compared with the Piecewise Linear Representation Method (PLR), the Intelligent PLR (IPLR) [23], the B&H Strategy and the Rule Based BPN. The B&H Strategy and the TPP-based strategy in [7] are compared in Section 4.1.2.

4.1.1. Comparison to the IPLR

In this section, we use the data of 8 stocks in Taiwan stock market for simulation. Training and testing samples are the same as in [23]. i.e., the training data is sampled from 2004/01/02 to 2005/9/30 while the testing data is sampled from 2005/10/01 to 2006/04/12. Experimental results of Rule Based BPN, PLR and IPLR are directly copied from [23] for fair comparison. As is shown in Table 2, the rates of return by applying LG-Trader are much higher than other methods. For stock trading, we test the LG-Trader on 8 stocks for each method. The LG-Trader yields the best rates of return in 7 stocks. It outperforms IPLR, PLR, Rule Based BPN and B&H Strategy by 69.21%, 95.21%, 122.83% and 190.58%, respectively, over the 8 stocks for each method. The LG-Trader may not perform very well in this situation. In such a case, the LG-Trader still make 128.57% positive rate of return, about 33.43% and 10.43% less than that of the IPLR and PLR, respectively. In such case, IPLR and PLR are better choices of trading decision support. However, in real situation, we cannot foresee the price trend of a stock will be very different in future period in comparison to the past few months. So, overall, the LG-Trader is still the best choice of user.

Then, we further analyze the distributions of training and testing samples of SIS and UMC. For visualization, we selected two features that are selected for both stocks by the LG-Trader feature selection: open and volume. Figs. 4 and 5 show the plots of SIS and UMC, respectively. As aforementioned, the training and the testing samples of SIS are very different, so training samples of SIS locate mainly at lower-left and lower-right corners while testing samples mainly located in the lower-middle and right hand side of the figure. On the other hand, training and testing samples of UMC located closely, i.e. they are similar to each other. Therefore, the LG-Trader yields significantly better results in comparison to IPLR and PLR.

4.1.2. Comparison to the TPP-based strategy

The TPP is proposed in [7]. Experiments in [7] use a stock TESCO and an index DJIA only. So, we compare the LG-Trader with TPP based Strategy and the B&H Strategy on these two datasets. Again, the experimental results of the TPP method is copied from [7] and we use the same periods of training and testing as in [7]. Table 4 shows the experimental results of different methods on TESCO and DJIA. In both datasets, the LG-Trader outperforms the TPP-based Strategy.

4.2. LG-Trader for index trading

To demonstrate the LG-Trader is good for both stock and index trading, we test the LG-Trader on 5 indices of major stock markets on the world: UK Financial Times Stock Exchange 100 Index (FTSE), Hong Kong Hang Seng Index (HSI), Japan Nikkei Index (NIKKEI), China Stock Exchange Component Index (SZI) and USA
Dow Jones Industrial Average Index (DJIA). We compare the LG-Trader for indices of different markets. The LG-Trader provides both functionality and is adaptable to different indices. 5BAIS is never selected while stock or index trading in overall. This also supports our view that we enhance to better trading rules with decision support from the LG-Trader: feature selection and architecture selection of the neural network. Both feature and architecture are optimized using a Pareto based multi-objective optimization method with the NSGA-II. Both the training error and the ST-SM are used as the multi-objective of the optimization. Experimental results show that the proposed LG-Trader outperforms Buy-and-Hold Strategy and state-of-the-art turning point based trading methods.

The performance of the LG-Trader could be further enhanced by using more powerful multi-objective optimization algorithms (e.g. [28]) and increase the features in the full set for selection (e.g. adding candlestick patterns or other financial indices). Replace the MLPNN by support vector machines is another alternative.

5. Conclusion

To deal with the imbalance problem between trading and holding decisions of stock trading, the LG-Trader is proposed. The imbalance problem is relieved by a weighted L-GEM consisting of a weighted training error and a weighted ST-SM. The error of misclassifying either a buy or a sell to a hold decision is penalized heavier than that of reversing by the weight of the generalization error model. There are two phases in the training of the LG-Trader: feature selection and architecture selection of the neural network. Therefore, feature and architecture are optimized using a Pareto based multi-objective optimization method with the NSGA-II. Both the training error and the ST-SM are used as the multi-objective of the optimization. Experimental results show that the proposed LG-Trader outperforms Buy-and-Hold Strategy and state-of-the-art turning point based trading methods.

The performance of the LG-Trader could be further enhanced by using more powerful multi-objective optimization algorithms (e.g.) and increase the features in the full set for selection (e.g. adding candlestick patterns or other financial indices). Replace the MLPNN by support vector machines is another alternative.

Table 6

<table>
<thead>
<tr>
<th>Stock index</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTSE</td>
<td>Open,low,volume,5MA,20MA,9RSI,12RSI,K,D,OBV,10ROC</td>
</tr>
<tr>
<td>HSI</td>
<td>Open,low,5MA,10BIAS,10V3,10KD,5RSOC,10ROC</td>
</tr>
<tr>
<td>NKKEI</td>
<td>High,low,volume,20MA,9RSI,10W7,RSMACD,12W7,5RSOC</td>
</tr>
<tr>
<td>SZI</td>
<td>Close,10MA,9BIAS,9RSI,12W7,3L5ROC,9MACD,K,D,J</td>
</tr>
<tr>
<td>DJIA</td>
<td>Open,volume,10MA,20MA,10BIAS,12RSI,OVB,9MACD,K,D</td>
</tr>
</tbody>
</table>

Fig. 5. Distribution of Training and Testing Samples of UMC.

Table 6

<table>
<thead>
<tr>
<th>Feature selected by the LG-Trader for indices of different markets.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stock index</strong></td>
</tr>
<tr>
<td>FTSE</td>
</tr>
<tr>
<td>HSI</td>
</tr>
<tr>
<td>NKKEI</td>
</tr>
<tr>
<td>SZI</td>
</tr>
<tr>
<td>DJIA</td>
</tr>
</tbody>
</table>

Acknowledgements

This work is supported by National Natural Science Foundation of China (61030171, 61272201 and 61003172) and a Program for New Century Excellent Talents in University (NCET–11–0162) of China.

References

W.W.Y. Ng et al. / Neurocomputing 146 (2014) 104–112


(http://finance.yahoo.com/).


Ms. Xue-ling Liang received her bachelor degree in Guangdong University of Technology in 2011. She is currently a master student in South China University of Technology, China. Her major research interest is stock prediction with machine learning techniques.

Dr. Jincheng Li received his Ph.D. degree from South China University of Technology in 2013. His current research interests include machine learning and pattern recognition, especially deep learning and image retrieval.

Professor Daniel S. Yeung is currently a Chair Professor with School of Computer Science and Engineering, South China University of Technology. He has been a faculty member of Hong Kong Polytechnic University, Hong Kong, and Rochester Institute of Technology, USA. Prof. Yeung has also worked for TRW Inc., General Electric Corporation R&D Centre and Computer Consoles Inc. in USA. He is a Fellow of the IEEE and served as the President of IEEE SMC Society in 2008–2009.

Dr. Patrick P.K. Chan received the B.Sc. degree in Information Technology and the Ph.D. degree from Hong Kong Polytechnic University. He is currently an Associate Professor in South China University of Technology, China. His current research interests include adversary learning, multiple classifier system and localized generalization error model. He is a member of the Board of Governors of IEEE SMC society and also the Chairman of IEEE SMC Hong Kong Chapter.

Dr. Wing W. Y. Ng received his B.Sc. and Ph.D. degrees from Hong Kong Polytechnic University in 2001 and 2006, respectively. He is now an Associate Professor in the School of Computer Science and Engineering, South China University of Technology, China. His major research interests include business intelligence, large scale image retrieval and machine learning techniques. Dr. Ng is currently an associate editor of the International Journal of Machine Learning and Cybernetics. He is the principle investigator of two China National Nature Science Foundation projects and a Program for New Century Excellent Talents in University from China Ministry of Education. Dr. Ng served as the Board of Governor of IEEE Systems, Man and Cybernetics Society (SMCS) in 2011–2013.