Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 34 (2008) 620–627
Expert Systems with Applications
www.elsevier.com/locate/eswa

Intelligent stock trading system by turning point confirming and probabilistic reasoning

Depei Bao *, Zehong Yang

State Key Lab of Intelligent Technology and Systems, Computer Science Department, Tsinghua University, Beijing, China

Abstract

Financial engineering such as trading decision is an emerging research area and also has great commercial potentials. A successful stock buying/selling generally occurs near price trend turning point. Traditional technical analysis relies on some statistics (i.e. technical indicators) to predict turning point of the trend. However, these indicators can not guarantee the accuracy of prediction in chaotic domain. In this paper, we propose an intelligent financial trading system through a new approach: learn trading strategy by probabilistic model from high-level representation of time series – turning points and technical indicators. The main contributions of this paper are two-fold. First, we utilize high-level representation (turning point and technical indicators). High-level representation has several advantages such as insensitive to noise and intuitive to human being. However, it is rarely used in past research. Technical indicator is the knowledge from professional investors, which can generally characterize the market. Second, by combining high-level representation with probabilistic model, the randomness and uncertainty of chaotic system is further reduced. In this way, we achieve great results (comprehensive experiments on S&P500 components) in a chaotic domain in which the prediction is thought impossible in the past.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Intelligent stock trading system; Turning point; Technical indicators; Markov network

1. Introduction

The stock market is a complex and dynamic system with noisy, non-stationary and chaotic data series (Peters, 1994). Stock movement is affected by the mixture of two types of factors: determinant (e.g. gradual strength change between buying side and selling side) and random (e.g. emergent affairs or daily operation variations). Generally, the random factors are regarded as noise in stock data.

Due to complication and uncertainty of stock market, stock prediction is one of the most challenging problems. Along with the development of artificial intelligence, especially machine learning and data mining, more and more researchers try to build automatic decision-making systems to predict stock market (Kovalerchuk & Vityaev, 2000). Among these approaches, soft computing techniques such as fuzzy logic, neural networks, and probabilistic reasoning draw most attention because of their abilities to handle uncertainty and noise in stock market (Vanstone & Tan, 2003, 2005).

However, most of them just utilize low-level price data as their training set in classification or rule induction (Vanstone & Tan, 2003). Though soft computing can somewhat reduce the impact of random factors, low-level data are so uncertain that they even behave purely randomly at some time (Peters, 1994).

High-level representation integrates human understanding of stock market into representative model, which can largely reduce the randomness embedded in the low-level data. A financial time series moves generally in fluctuant way because of continuous changes of strength between buying side and selling side. Therefore, data series is comprised of a series of turning points. Good investors will sell stocks at the top of the range and buy stocks at the bottom of the range within the stock trends. Obviously, the top and the bottom are near the turning points. We make use of

* Corresponding author. Tel.: +86 010 62777087.
E-mail address: baodepei@gmail.com (D. Bao).

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.09.043

this high-level representation by predicting or confirming turning points at the time close to its occurrence instead of predicting the precise price in the future. Another kind of high-level representation is technical indicators, which are some statistics derived from recent price data. In technical analysis (Pring, 2002), the central idea is to look for peaks, bottoms, trends, and indicators to estimate the possibility of current trend reversal and then making buy/sell decisions based on those factors. In fact, technical indicators and turning point representation are consistent. In this paper, we combine these two high-level representations together to make buy/sell decision.

However, there still exists uncertainty in high-level representations. For example, single indicator generally cannot predict trend reversal with high accuracy. More effectively they should be viewed in probabilistic manner, i.e. more indicators indicate the same reversal, and more probable it will occur. Thus, the probability model is used to further integrate two representations, and parameters of the model are learned from historical data.

In Section 2, the architecture of our system is presented. Then, we describe the details of training and testing of probabilistic model in Section 3. The experiments in Section 4 show the promising of our system. Finally, we conclude.

2. Architecture of our system

Due to fluctuation, there are many local maximal/minimal points in stock data series (we will call them critical points for simplicity). Some of them are real trend reversal points and others are just noise. The high-level representation is to represent the time series as well as training examples by these real turning points. Moreover in technical analysis, all indicators can produce signals at certain time to indicate the reversal of current trend. The signals of most technical indicators occur after turning point (of course, they are close to the turning point and still in the top or bottom range of the trend). Therefore, in our model, we use technical indicators to confirm whether a critical point is real turning point of current trend along with the movement of price.

The architecture of our system is shown in Fig. 1. Like most machine learning algorithms, there are training and testing phase. The purpose of training is parameter selection for our probabilistic model from historical data. Then, in testing phase, we use our model to make buy/sell decision.

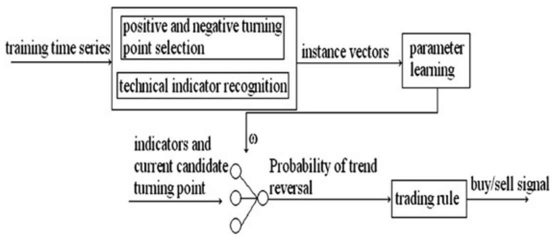
training time series
positive and negative turning point selection
technical indicator recognition
instance vectors
parameter learning
ϖ
indicators and current candidate turning point
Probability of trend reversal
trading rule
buy/sell signal

Fig. 1. Architecture of the trading system.

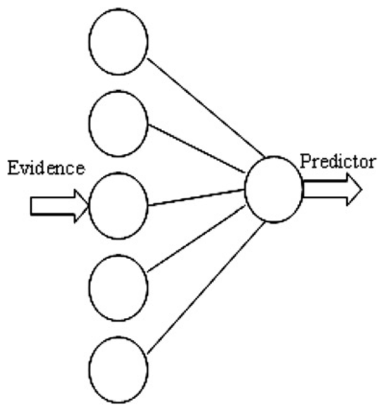

Fig. 2. Probabilistic structure.

The probabilistic model is a simplified Markov Network (Pearl, 1988) with one node as predictor variable and other nodes as evidence variables shown in Fig. 2. Parameter vector $\varpi = \{\omega_1, \ldots \omega_n\}$is assigned to every link between predicator and evidence.

Every evidence node represents a technical indicator and the predicator represents whether current candidate turning point is the real trend reversal point (true/false). When an indicator produce signal, its corresponding node will take true, or else take false. The result of the model is probabilities of predicator taking true and false.

In Markov Network, parameters on the link between evidence and predicator are learned from database with discrete positive and negative instances. We take a real turning point and signals of indicators near it as a positive instance (vector) in the training set. Real turning point is selected depending on investors' trading cycle and strategy (long term or short term investment, etc). In our training process, we measure the cycle by the duration and

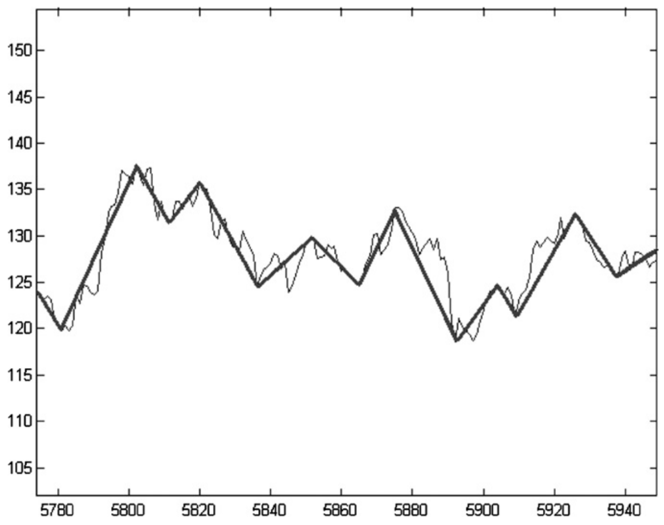

Fig. 3. Stock time series with thin lines as original data and bold lines as trend.

oscillation between a local maximal point and a local minimal point. Then, if these two parameters exceed some thresholds determined by trading cycle between two consecutive critical points, these two points will be the turning points of the current trend and previous trend, as shown in Fig. 3.

3. Training and testing of the model

3.1. Measure criteria of turning points

As shown in Fig. 4, the criterion is defined as follows (Perng, Wang, Zhang, & Parker, 2000): Given a sequence of critical points $(x_1,y_1),\ldots(x_n,y_n)$, ($x_i$ and y_i are the time point and price data of the ith critical point, respectively), a minimal time interval T and a minimal vibration percentage P, (we call them together as distance), remove (x_i,y_i) and (x_{i+1},y_{i+1}) if

$$x_{i+1} - x_i < T \text{ and } \frac{|y_{i+1} - y_i|}{(y_i + y_{i+1})/2} < P.$$

The thresholds of P and T have intuitive meaning. For example, if a stock trader trades once a week (5 business days) and regards a 5% gain or loss as significant, then he/she simply uses $P = 0.05$ and $T = 5$ to recognize turning point. In our experiment, we assign different thresholds for different stocks in terms of the volatility of historical data.

3.2. Positive and negative instances selection

As stated in previous section, we define positive instance (i.e. turning point) as critical point that exceeds specified threshold and negative instance as critical points, which fail to exceed the given threshold. However, directly following this definition will make the number of negative instances

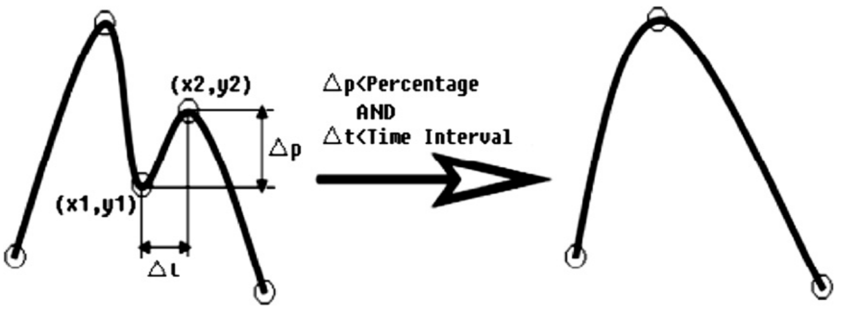

Fig. 4. Criterion of the model.

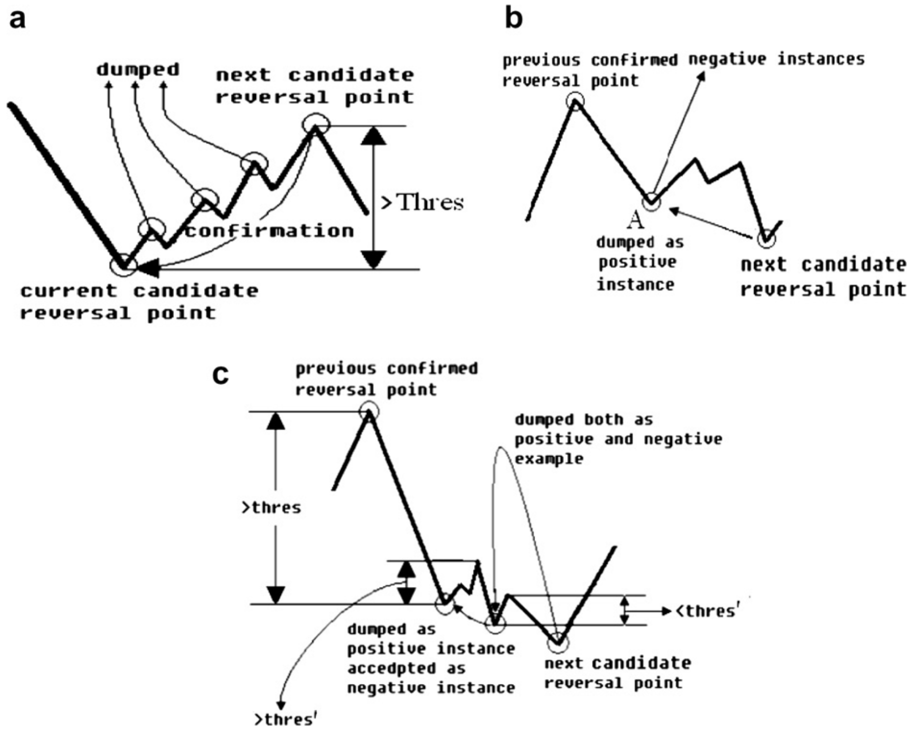
a
dumped
next candidate reversal point
>Thres
confirmation
current candidate reversal point
b
previous confirmed reversal point
negative instances
A
dumped as positive instance
next candidate reversal point
c
previous confirmed reversal point
dumped both as positive and negative example
>thres
<thres'
dumped as positive instance accedpted as negative instance
next candidate reversal point
>thres'

Fig. 5. (a) Positive instance (b) negative instance (c) modified negative instance.

far more than that of positive ones and make the parameter learning algorithm invalid.

To address such problem, suppose we move along all critical points in training set to select critical point one after another like realistic trading process (testing). In Fig. 5a, the current candidate critical point is a real turning point and previous trend is downtrend. We cannot confirm this reversal until we reach the critical point with the oscillation and time interval between it and candidate turning point exceeding threshold. Note that a local maximal point and a minimal point act as turning point alternately. Hence, any local maximal/minimal point between these two points cannot be regarded as candidate reversal points (i.e. neither as positive or negative instances) without previous candidate point confirmation. In other words, no critical point will be regarded as candidate turning point (positive or negative instance) until distance between current critical point and candidate turning point exceeds threshold (if it is the real turning point). On the other hand, in Fig. 5b if current candidate turning point A (current is downtrend) is a false turning point (we have not known that at current point), then never can we find another critical point satisfying above conditions in the future. In such case, it will be regarded as "false" turning point, i.e. negative instance, when we get to the critical point higher (for uptrend reversal) or lower (for downtrend reversal) than it (new candidate is more higher or lower in the neighborhood. Before new candidate, current trend is not reversed).

However, even by such disposal, negative instances outnumber positive ones a lot. In practice, we generally do not concern whether a critical point is a turning point, when it does not behave "like" a turning point, i.e. has oscillation and time interval to next critical point to some extent. This means there is another threshold (*thres'*) between a certain future opposite critical point before next candidate turning point and the current point, which should be exceeded (Fig. 5c). Then only critical point meeting this extra condition will be regarded as negative instance.

3.3. Signal recognition

In our system, we adopt 4 common technical indicator systems including 30 indicators (we can enhance our system by more indicator systems in the future), they are:

Moving average system.
RSI oscillator system.
Stochastic slowK–slowD crossover/oscillator system.
Trendline system.

The signals of the indicator system are recognized and then assigned to their closest positive or negative instances according to the properties of indicators, i.e. leading or lagging. A leading signal will be assigned to next instance while a lagging signal will be assigned to previous instance. Then, every instance can be represented as a Boolean vector.

A comprehensive discussion of technical analysis and indicator computation can be found in the book by Pring (2002).

3.4. Probabilistic model

In Markov Network, for each clique C_i (namely the maximal subgraphs, whose nodes are all adjacent to each other) assign a nonnegative and real-valued potential function (also called compatibility function) $g_i(c_i)$, where c_i is an assignment to the variables in C_i. Then the joint distribution of the Markov Network is given by

$$P(X=x)=\frac{1}{Z}\prod_i g_i(c_i), \tag{1}$$

where partition function is $Z=\sum_{x\in\chi}\prod_i g_i(c_i)$. $P(X=x)$ is normalized product P over all possible value combinations of the variables in the system. Richardson and Domingos (2004) represent the joint distribution in log-linear form, with each clique potential function replaced by an exponentiated weighted sum of features of the state, leading to

$$P(X=x)=\frac{1}{Z}\exp\left(\sum_i \omega_i f_i(x)\right). \tag{2}$$

Comparing Eq. (1) with Eq. (2), we get $\log g_i(c_i)=\omega_i f_i(x)$.

A feature $f_i(x)$ may be any real-valued function of the state. In fact, our model defines $f_i(x)$ in logic (hence $f_i(x)\in\{0,1\}$), i.e. we regard connections between nodes as propositional rules $L_{i1}\rightarrow L_{i2}$ ($\sim L_{i1}\vee L_{i2}$) and each rule corresponds to one clique.

If $x\in\{TRUE,FALSE\}^2$ in each clique makes the corresponding rule take $TRUE$ value, then $f_i(x)=1$, or else $f_i(x)=0$.

3.4.1. Inference

Inference in Markov networks equals to answer arbitrary queries of the form: What is the probability that set of variables V_1 holds given that another set of variables V_2 does? That is

$$P(V_1|V_2)=\frac{P(V_1\wedge V_2)}{P(V_2)}=\frac{\sum_{x\in\chi_{V_1}\cap\chi_{V_2}}P(X=x)}{\sum_{x\in\chi_{V_2}}P(X=x)}, \tag{3}$$

where χ_{V_i} is the set of truth values, where V_i holds, and $P(X=x)$ is given by Eq. (2).

In our problem, the conditional probabilities required to calculate take only one form, according to Eq. (3):

$$P(Y=y|X'=TRUE) = \frac{\sum\limits_{x,y\in V_{X'=TRUE}\cap V_{Y=y}} P(Y=y,X=x)}{\sum\limits_{x,y\in V_{X'=TRUE}\cap V_{Y=TRUE}} P(Y=TRUE,X=x)+\sum\limits_{x,y\in V_{X'=TRUE}\cap V_{Y=FALSE}} P(Y=FALSE,X=x)}, \tag{4}$$

where $X'\subseteq X$.

However, computing Eq. (4) directly is intractable. Some approximation algorithms should be taken. Gibbs sampling, a special Monte Carlo Markov Chain (MCMC) algorithm is adopted. A more detailed introduction to Gibbs sampling and MCMC can be found in lecture note by Walsh (2002).

3.4.2. Parameter learning

The dependency parameter learning is carried out on all instances. By closed world assumption, we assume that any variable without explicit assignment will be considered as *FALSE*.

In probabilistic model parameter estimation, the usual method is maximum likelihood estimation. For our model, the log-likelihood function can be written as

$$L(W) = \log \prod_{j=1}^{n} P_{\omega}(X = x_j) = \sum_{j=1}^{n} \sum_{i} \omega_i f_i(x_j) - n \log Z, \tag{5}$$

where W is weight vector $W = (\omega_1, \omega_2, \ldots \omega_{|n|})$, n is the number of all instances.

The derivative of the log-likelihood with respect to its weight is

$$\frac{\partial}{\partial_{\omega_i}} L(W) = \sum_{j=1}^{n} f_i(x_j) - n \sum_{j=1}^{n} P_{\omega}(X = x_j) \cdot f_i(x_j), \tag{6}$$

where the sum is over all possible instances, and $P_{\omega}(X = x_j)$ is $P(X = x_j)$ computed using the current weight vector. Unfortunately, due to Richardson and Domingos (2004), calculating the second item in Eq. (6), which requires inference over the model is intractable.

An alternative is proposed by Singla et al. (2005) called discriminative training. The parameter estimation through optimizing conditional likelihood function is more convenient than optimizing likelihood function according to the structure of our model. The conditional likelihood function of Y given X is

$$\prod_{j=1}^{n} P(Y = y_j | X = x_j) = \prod_{j=1}^{n} \frac{1}{Z_x} \exp\left(\sum_{i \in V_y} \omega_i \cdot f_i(x_j, y_j) \right), \tag{7}$$

where V_Y is the set of cliques that contains at least one variable in Y.

The gradient of the conditional log-likelihood (CLL) is

$$\frac{\partial}{\partial_{\omega_i}} \log \prod_{j=1}^{n} P_{\omega}(Y = y_j | X = x_j) = \sum_{j=1}^{n} \left[f_i(x_j, y_j) - n \cdot \sum_{y' \in V_Y} P_{\omega}(Y = y' | X = x_j) \cdot f_i(x_j, y') \right], \tag{8}$$

where V_Y is the set of all possible values of Y.

Despite of its reduction of dimension, the CLL cannot be calculated precisely in general case. However, the CLL and its gradient can be transformed into tractable form because there is only one query variable and two possible values in our model. Then the gradient of CLL becomes

$$\frac{\partial}{\partial_{\omega_i}} \log \prod_{j=1}^{n} P_{\omega}(Y = y_j | X = x_j) = \sum_{j=1}^{n} f_i(x_j, y_j) - \sum_{j=1}^{n} [P_{\omega}(Y = TRUE | X = x_j) \cdot f_i(x_j, y_j = TRUE) + P_{\omega}(Y = FALSE | X = x_j) \cdot f_i(x_j, y_j = FALSE)], \tag{9}$$

where

$$P_{\omega}(Y = TRUE | X = x_j) = \frac{\exp(\sum_i \omega_i f_i(x_j, y_j = TRUE))}{\exp(\sum_i \omega_i f_i(x_j, y_j = TRUE)) + \exp(\sum_i \omega_i f_i(x_j, y_j = FALSE))},$$

$$P_{\omega}(Y = FALSE | X = x_j) = \frac{\exp(\sum_i \omega_i f_i(x_j, y_j = FALSE))}{\exp(\sum_i \omega_i f_i(x_j, y_j = TRUE)) + \exp(\sum_i \omega_i f_i(x_j, y_j = FALSE))}.$$

(derived under close world assumption).

The gradient of CLL can be obtained with ease and no approximating algorithm is required. Then, with conditional likelihood function and its gradient we can use any unconstrained optimization algorithm (e.g. Quasi-Newton method) to achieve optimized ω_i.

4. Experiment

We test our system on two main aspects: probabilistic model computing cost and trading result.

4.1. Computing cost of probabilistic model

The cost for learning and inference is low. Parameter learning (with 1000 iterations in local searching) with train set of 30 years (7800 daily points, around 500 instances) takes only about 75 s. Inference on a candidate turning points takes average 73.188 s on a P4 2.4G computer. Though there is no benchmark for this problem, the time cost is much lower than other approaches. The reason for this efficiency is that we take high-level representation for the learning and inference.

4.2. Test on automatic stock trading

A simple trading rule is adopted: when we confirm current reversal with certain probability, we sell/buy, i.e. when current turning is from uptrend to downtrend, sell and vice versa. Suppose \$1000 initial fund and trade all funds/stocks at each operation.

4.2.1. Typical stock movements and their trades

In this experiment, each stock will be trained and tested individually. We employ all historical price of one stock, and all data are partitioned into train set and test set (only dealing with stock with over 3000 points).

Fig. 6 shows a movement in a fluctuant and overall loss market and long period trade (4 years). In such market, our trading system is still able to profit up to 71.2% while the stock loss is about 31.6%.

Movement in Fig. 7 is an overall bull market and short period trade (2 years). In such market, our trading system is able to profit up to 79% while the market gains about 61.3%. In other word, our system can outperform buy-and-hold strategy in bull market.

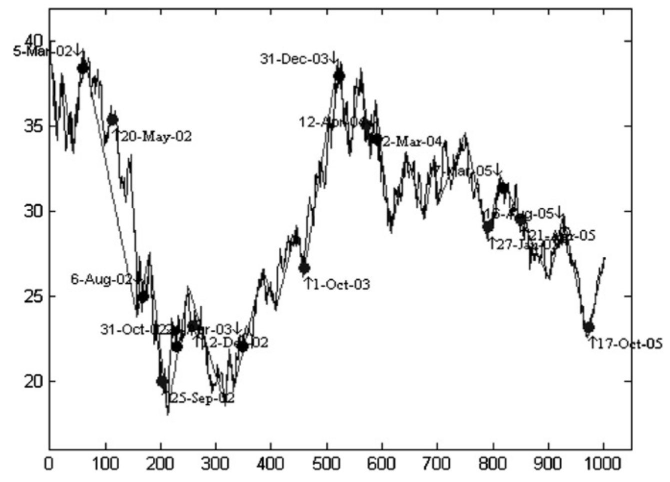

Fig. 6. Trading log (black: sell, grey: buy) of ALCOA INC for 4 years, with probability set to 0.8 and profit 71.2%.

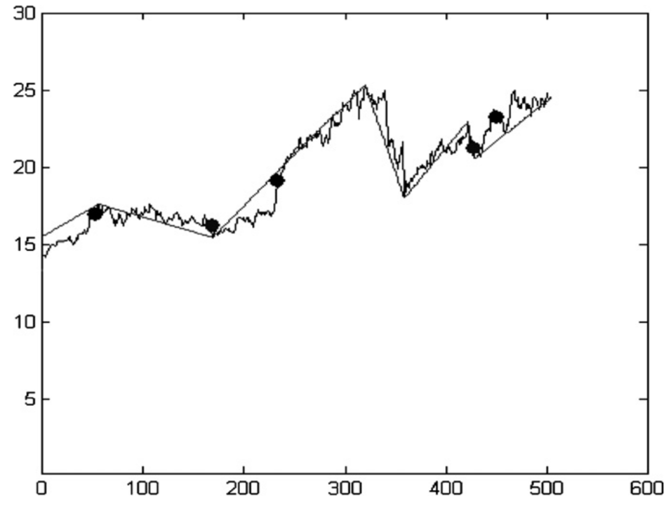

Fig. 7. Trade log of AMERISOURCEBERGEN CP for 2 years, with probability set to 0.8 and profit 79%.

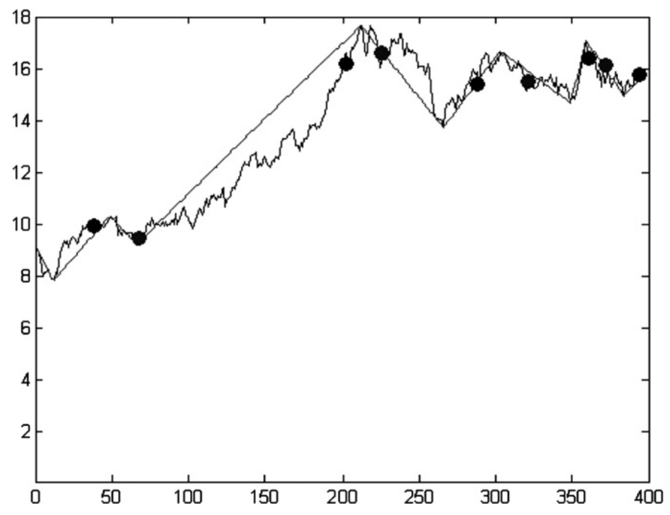

Fig. 8. Trade log of AES CP INC for 2 years, with probability set to 0.8 and profit 104%.

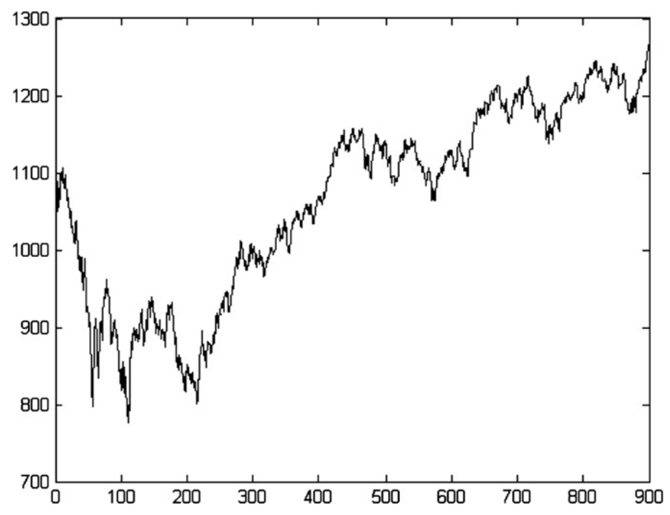

Fig. 9. S & P500c from May 3, 2002 to November 25, 2005 (900 points).

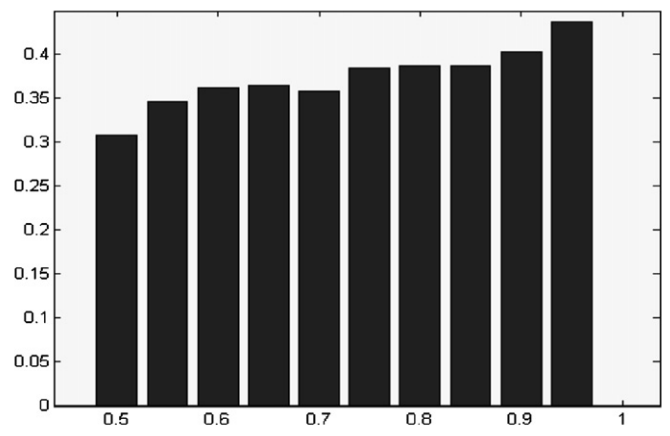

Fig. 10. Average profit rate of S & P500 components on different probabilities.

Table 1
Three periods profit of 10 arbitrary selected stock in S & P500

Symbol	2000–2002 (%)	2002–2004 (%)	2004–2006 (%)
ASN	15.2	64.9	14.5
BHI	71.6	45.4	5.4
CMVT	34.3	35.7	229.5
CVS	9.2	58.4	70.3
HAS	20.8	5.7	201.3
LEN	48.3	37.2	−20.7
LPX	7.0	55.6	12.9
MUR	20.7	32.1	64.4
PGL	−5.9	13.3	33.3
PHM	−26.8	77.8	24.8

Movement in Fig. 8 is an overall bull market and short period trade (2 years). In such market, our trading system achieves excellent result up to 104%.

4.2.2. Overall market performance

Most trading systems evaluate their performance on selected stock. However, such test is unreliable to prove effectiveness due to lack of generalization. To validate our system extensively, we test gain or loss of 500 stocks (S & P500 components) and check whether their average profit outperform S & P500c index.

Fig. 9 shows S & P500c index over three years (900 points), it gains about 18.15%.

The system performs simulated trading on 454 stocks (with over 3000 daily data for training and testing) and gets profits on 416 stocks. The average profit rates with different probabilities in trading rule are exhibited in Fig. 10. The maximal profit rate is up to 43.6%. On average, the profit is improved with more certainty on trend reversal.

Table 2
Missed opportunities and false operation of IBM

Probabilities	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Actual opportunity	36	36	36	36	36	36	36	36	36	36
Missed opportunity	4	4	4	4	4	4	4	10	14	18
Missed %	11.1	11.1	11.1	11.1	11.1	11.1	11.1	27.8	38.9	50
Total operation	16	16	16	16	16	16	16	13	11	9
False operation	7	7	5	5	5	3	3	4	3	3
F.O %	43.8	43.8	31.3	31.3	31.3	18.8	18.8	30.8	27.3	33.3
Profit %	24.4	23.3	93.7	92.8	98.9	145.1	160.3	63.0	34.9	53.1

Table 3
Missed opportunities and false operation of microsoft

Probabilities	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Actual opportunity	48	48	48	48	48	48	48	48	48	48
Missed opportunity	4	4	6	8	8	8	14	24	34	42
Missed %	8.3	8.3	12.5	16.7	16.7	16.7	29.2	50	70.8	87.5
Total operations	22	22	21	20	20	20	17	12	7	3
False operation	9	9	7	5	5	5	5	3	2	1
F.O %	40.9	40.9	33.3	25	25	25	29.4	25	28.6	33
Profit %	−27.1	−23.6	69.5	87.5	75.9	74.3	36.0	21.1	−9.2	−26.9

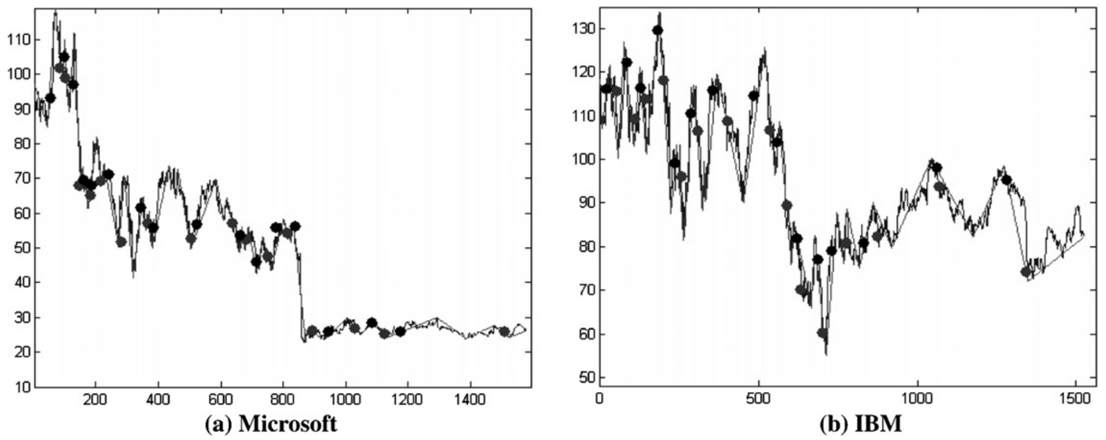

Fig. 11. Trading log (buy: grey sell: black) of IBM and Microsoft.

Second, we pick 10 stocks in random and examine their profit on three periods of time (two years (252 ∗ 2 points) each) with probability 0.8. The results are listed in Table 1.

According to Table 1, most stocks can profit with only few loss (due to the overall downtrend during that period).

4.2.3. Experiments on individual stock

We test the system on three specified criteria: missed opportunity, false operation and profit. The experiment is carried out on the common selected IBM and Microsoft of many papers.

Missed opportunity is defined as the number of real tuning point without trading operation. *False operation* is the number of loss operations (buy high and sell low) in total operations (one operation contains a buy and a sell). Tables 2 and 3 list the results of IBM and Microsoft stock tested from January 3, 2000 to December 30, 2005 with different probabilities. Trading logs of IBM and Microsoft are shown in Fig. 11. Though there are some losses of Microsoft at some probabilities, the overall price of Microsoft drops about 77.5% during the period. In addition, our raw trading rule is not so elaborate as to adapt to different probabilities. We can see IBM also experienced drastic decrease (−29.1%), but we can still profit a lot even by our simple trading rule. Moreover, two criteria: *missed opportunity*and *false operation* are satisfactory.

5. Conclusion

In this paper, we present a novel implementation of stock trading system, which combines high-level representations with probabilistic model. We regard the financial market as a dynamic system with uncertainty. Therefore a high-level representation and probabilistic model are more robust to such problem. Some simulated experiments are carried out to test the efficiency of our system, showing the universally profitability for most stocks in the market.

Further work includes augmenting the system with other soft computing techniques and developing more reasonable trading rules. The applicability of our system is to facilitate people in trading decision by providing them with high-level decision.

References

Kovalerchuk, Boris & Vityaev, Evgenii. (2000). *Data mining in finance: advances in relational and hybrid methods*. Kluwer Academic.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA: Morgan Kaufmann.

Perng, C.S., Wang, H., Zhang, S.R., & Parker, D.S. (2000). Landmarks: a new model for similarity based pattern querying in time series databases, In *Proceedings of the sixteenth international conference on data engineering*. San Diego, USA.

Peters, Edgar E. (1994). *Fractal market analysis: applying chaos theory to investment and economics*. New York: Willey & Sons, Inc.

Pring, Martin. (2002). *Technical analysis explained*, (4th ed.). Paperback McGraw-Hill Company, ISBN0071226699.

Richardson, M., & Domingos, P. (2004). *Markov logic networks*, Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA.

Singla, Parag and Domingos, Pedro. (2005). Discriminative training of Markov logic networks. In. *Proceedings of the twentieth national conference on artificial intelligence* (pp. 868–873). Pittsburgh, PA: AAAI Press.

Vanstone, B., & Tan, C. (2003). A survey of the application of soft computing to investment and financial trading, In *Proceedings of the eighth Australian & New Zealand intelligent information systems conference (ANZIIS 2003)*. Sydney, Australia, December 10–12.

Vanstone, B., & Tan, C.N.W. (2005). Artificial neural networks in financial trading, In M. Khosrow-Pour (Ed.), *Encyclopedia of information science and technology*. Idea Group.

Walsh, B. (2002). Markov chain Monte Carlo and Gibbs sampling, lecture. *Notes for EEB, 596z*.