Day Trading the Emerging Markets Using Multi-Time Frame Technical Indicators and Artificial Neural Networks

Alexandru Stan

Abstract This chapter addresses the topic of automated day trading systems based on artificial neural networks and multi-timeframe technical indicators, a very common market analysis technique. After we introduce the context of this study and give a short overview of day trading, we set out our approach and methodological framework. Then, we present the results obtained through these procedures on several of the most liquid stocks in the Romanian stock market. The final section of the chapter concludes the study and brings some insight about possible future work in the area.

Keywords Day trading systems · Neural networks · Multi-time frame technical analysis · Market forecast · Automatic trading

1 Introduction

Artificial neuronal networks have many applications [1] in the financial field as interesting and diverse as the identification of firms in financial distress, the assessment of credit rates for individuals, the detection of fraudulent behavior in bank cards usage, the forecasting of losses in insurance companies or the valuation of enterprises. In this chapter, we intend to address the topic of day trading emerging markets with artificial neuronal networks [10, 11] and multi-time frame technical indicators.

We opted for the analysis of the emerging markets [8] since many of them obviously don't satisfy the Efficient Market Hypothesis even in its weakest form, that is to say, future price movements can be predicted by using historical stock

A. Stan (✉)
Babes-Bolyai University, Cluj-Napoca, Romania
e-mail: alexandru.stan@econ.ubbcluj.ro

B. Iantovics and R. Kountchev (eds.), *Advanced Intelligent Computational Technologies and Decision Support Systems*, Studies in Computational Intelligence 486,
DOI: 10.1007/978-3-319-00467-9_16, © Springer International Publishing Switzerland 2014

prices and technical analysis. If we accept the existence of these deterministic patterns, then the artificial neural networks can be put to work to find them. Naturally, the inputs for the artificial neuronal networks will be the inputs used by the technical analysts: a myriad of classical technical indicators during different time periods at different sampling frequency. Since we discuss very short term trading strategies [2] the sampling frequency will be counted in minutes rather than hours or days.

The main problem we identified when implementing trading strategies is that although some offer good accuracy over well chosen time horizons the stochastic components [2, 4] contained in financial time series may completely jeopardize their accuracy and effectiveness as the time span varies [3, 5, 9]. Thus the dynamics of an algorithmic strategy [7] may differ significantly according to the time frame we are looking at. For instance, when on a weekly basis the market may be strongly up trending while in the very short term the stock prices could be ranging between ephemeral support and resistance levels the trading automaton should adapt its behaviors to the sampling frequency. As a consequence, we decided to have a multi-timeframe approach and dedicate an autonomous artificial neural network for each time frame to identify more accurately the deterministic patterns of the corresponding sampling frequency [6]. An important characteristic when using neural networks in this context is the neural network dimensionality [12, 15], that is to say, how many steps are used to predict the best trading operations. If the dimensionality is too small the learning process may suffer and the trading strategies prove inefficient. If the dimensionality is too large an over fitting may prove even more harmful. A multiple timeframe approach can be an interesting solution since it allows for different dimensionalities in the different time frame neural networks.

2 Day Trading

This technique consists in rather making small gains by using highly leveraged transactions scaled so as to maximize financial performance. The day trader benefits from market volatility and often gains or losses on each transaction only from a 0.1 % to a few percents of the invested capital. Most often a day trader performs dozens of orders per day. All positions are closed at the end of the market session, even if losses must be taken, avoiding by this considerable overnight losses. The goal is to consistently engage in more winners than losers and ensures that losers are as small as possible.

While many trading techniques such as trend following, range trading, scalping, news trading, contrarian investing are being used by the day traders, neural network automated strategies are a good fit for intraday strategies since a well trained network can effectively detect intraday patterns, enter and exit signals and assure the short and medium term statistical consistency in non efficient emerging

markets. Like any automated strategy is also eliminates any psychological bias that a human trader may present.

The main obstacle to financial effectiveness when frequently performing numerous transactions are the costs related with brokerage, commissions, spreads and slippage. For this reason, we incorporated in our strategies anti churning parameters.

3 Methodological Framework

3.1 Data Preprocessing

The inputs for the artificial neuronal networks will be the inputs used by the technical analysts: the stock prices, the traded volumes, and a plethora of technical indicators that may reveal interesting and effective in the forecasting process. These input data will have to go through a preprocessing so as to fit in the intervals [0,1] or [−1,1]. For instance, all the bounded technical oscillators having values between 0 and 100 will be scaled down by dividing their values by 100.

$$Input^{t}_{Oscilator} = \frac{OscillatorValue^{t}}{100}$$

Thus, for instance,

$$Input^{t}_{RSI} = \frac{RSIValue^{t}}{100}$$

More generally,

$$Input^{t}_{BoundedOscillator} = \frac{OscillatorValue^{t} - LowerBoundry}{UpperBoundry - LowerBoundry}$$

The unbounded oscillators are standardized if we have access to their historical values, and generally we do. After computing their estimated mean and standard deviation we can scale the input as follows:

$$Input^{t}_{BoundedOscillator} = \Phi\left(\frac{OscillatorValue^{t} - \overline{Oscillator}}{s_{oscillator}}\right)$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt$ is the normal cumulative distribution function.

If the data doesn't follow a normal distribution, statistical tests may be performed and the normal cumulative distribution function may be substituted accordingly.

For the moving average indicators, it is important to teach the neural networks not their absolute values at a given time but rather their relative dynamics (the way

their first and second derivatives move) and relative values as compared to stock prices and other time moving averages. This way the ANN will be able to generalize some profitable behaviors and not be stuck with absolute and often meaningless values.

$$Input^t_{MovingAverage} = \frac{MovingAverageValue^t}{price^t * \mathrm{Pri}ceScallingFactor} - \frac{1}{\mathrm{Pri}ceScalingFactor}$$

The scaling factor will mainly depend on the length of time frame of the sampling and the upper boundary of the estimated variance.

$$\mathrm{Pri}ceScalingFactor \approx s_{price} * \sqrt{LengthOfTheTimeframe}$$

In penny markets, characterized by huge levels of variance, the scaling factor can take big values, but in our case, since our strategies are on the very short term and the volatility moderate, scaling factors 2 turn out to be a good choice.

For the price and volume inputs we are most often interested in their dynamics than in their absolute values. That's why we use percentage changes rather than the differences of absolute values.

$$Input^t_{price} = \frac{\Delta price^t}{price^t \times \mathrm{Pr}\, iceScalingFactor}$$

$$Input^t_{Volume} = \frac{\Delta volume^t}{volume^t \times VolumeScalingFactor}$$

If it is deemed that support and resistance levels do play an important role in improving the trading strategy then additional volume and price inputs may be added as follows:

$$Input^t_{absoluteprice} = \frac{1}{price^t}$$

$$Input^t_{absolutevolume} = \frac{1}{volume^t}$$

The percentage input prices will also help at detecting the tangent of the trend channels.

Sometimes during the preprocessing stage we also need to get rid of some basic non stochastic components such as seasonality and trends. For instance, when the trends are linear this can easily be done by first differences on the level series and when exponential by first difference on log series.

In order to get rid of complex periodic components we may use discrete time low- and high-pass frequency filters or some linear combination of them. Low- and high-pass frequency filters attenuates signals with frequencies higher and, respectively, lower than some threshold cutoff frequency. Thus, the initial sampling

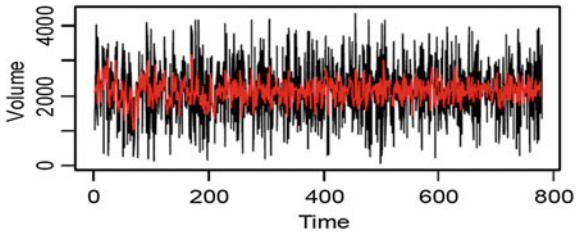

Fig. 1 A 5 min low-pass volume filtering example

vector $\begin{bmatrix} x_1 \\ x_2 \\ x_n \end{bmatrix}$ of the prices will be transformed through filtering into a vector $\begin{bmatrix} y_1 \\ y_2 \\ y_n \end{bmatrix}$ free of the non stochastic periodic components.

The general form of these linear filters is:

$$y_t = a_0 x_t + a_1 x_{t-1} + \cdots + a_{Dimensionality} x_{t-Dimensionality}$$

Using these filters could be quite useful in our approach based on multiple timeframes with different sampling frequencies.

This way each filter could have different dimensionalities enhancing the learning effectiveness of the corresponding artificial neural network. In a stock market time series the dimensionality stands for the number of previous pieces of information that are potentially relevant to the forecasting of the next value.

In order to determine the dimensionality of the networks in the regression models we analyzed auto correlation and partial auto correlation functions and identified past data which may cause variation in forecasting process.

The filters can also be applied on the volume data in order to focus on some aspects of the frequency domain (see Fig. 1).

3.2 The Cost of Transactions

The costs of transaction in our model will be the costs of brokerage and slippage. Since we do not always have their values at any given time we've seen them as prices, the brokerage cost as the price for a service and the slippage cost as the price for closing a position as fast as possible. Therefore we modeled them as log-normal distributions Log-N(μ_{spread},σ^2_{spread}) and Log-N(μ_{slippage},$\sigma^2_{\text{slippage}}$). While the spread is always there, we will suppose that the slippage cost will appear with the intensity of a homogenous Poisson process.

3.3 The Activation Function

According to the data preprocessing we used 2 activation functions. If the inputs were scaled so as to be positive we used the sigmoid function a special logistic function. Otherwise we used a hyperbolic tangent function.

If inputs $\in [0,\ 1]$, then the activation function is:

$$Act(x) = \frac{1}{1 + e^{-x}}$$

If inputs $\in [-1,\ 1]$, then the activation function is:

$$Act(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

3.4 The Neural Networks Training

We used a bootstrapping technique to feed for instruction the different timeframe foreword neural networks. We randomly chose the beginning of the each window element in the training set. This way the training set is more homogeneous since we avoid regime changing and structural breaks in the data [13]. The downturn is that the training and validation sets can intermingle. For an element window to be valid all its points must pertain to the same daily trading session. That's why the granularity of the convolution window should rather be small.

Feeding algorithm

1. For each sampling point x_i in the data set obtained according with the sampling granularity, calculate $q_i = \frac{i}{\text{cardinalityOfTheData}}$
2. While the size of the training set is not attained do
3. Generate an random number $g \in [0, 1]$
4. If $g \in [0, q_1]$ then feed the network with the element window $[x_1, x_2, \ldots, x_p]$ and with the related technical indicators
5. If $g \in [q_{i-1}, q_i]$ and x_i and x_{i+p-1} belong to the same market session then feed the network with the element window $[x_i, x_{i+1}, \ldots, x_{i+p-1}]$ and with the related technical indicators
6. End while

3.5 Error Calculation

According to the data preprocessing and the level of financial leverage we used 3 error calculation methods. If the inputs were scaled so as to be positive and the

leverage moderate we used mean squared error. If the inputs had alternating sings and the leverage was moderate we used root mean squared error. If the inputs were alternating or positives and the level of leverage was high we used arctangent root mean squared error.

We used arctangent mean squared error since it exaggerates and gives more weight to errors far away from origin and thus is useful for highly leveraged strategies.

3.6 The Networks Structure

Our data stemmed from the historical prices of the titles compounding the Bucharest Stock Exchange (BET) index for a 4 years period. We used 6 time-frames feed foreword neural networks with 5, 10, 15, 20, 25 and 30 min periods between samplings. The inputs were sliding window based. The structure of the each network for every given set of technical parameters was a multiple layer preceptor (MLP). The output layer of the MLP contains only one neuron trying to forecast the value of the stock at time t + 1.

After the computation of the 6 forecasting estimators obtained from the 6 neural networks (30, 25, 20, 15, 10 and respectively 5 min sampling), the strategy is to engage in a short or a long transaction only when a majority of estimates forecast the same direction for the value of the stock.

During the election process we granted more vote power to frequencies that dominated the normalized prices evolution spectrogram (Fig. 2).

Finally, as an alternative to Kelly criterion or Vince's optimal f position sizing, we resorted to an efficient frontier neural network algorithm which used all these different time frame algorithms separately as inputs for the efficient frontier combination (see Figs. 3 and 4).

3.7 Results

We used a wide range of performance and risk metrics in order to assess the performances of this technique. The most significant facts:

1. We compared the results with a set of outcomes obtained from technical indicator strategies and noticed that the algorithm had results of at least 30 % better.
2. We results were 46 % higher than a buy and hold strategy: $r_{\text{Buy\&Hold}} = \left(\frac{price_{sell}}{price_{buy}}\right)^{\frac{360}{daysnumber}}$

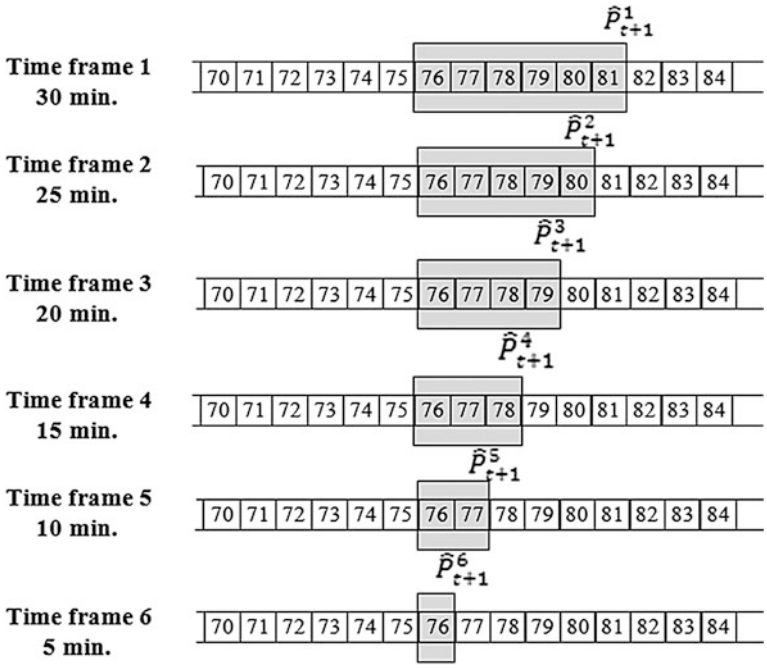

Fig. 2 The six ANN forecasting estimators

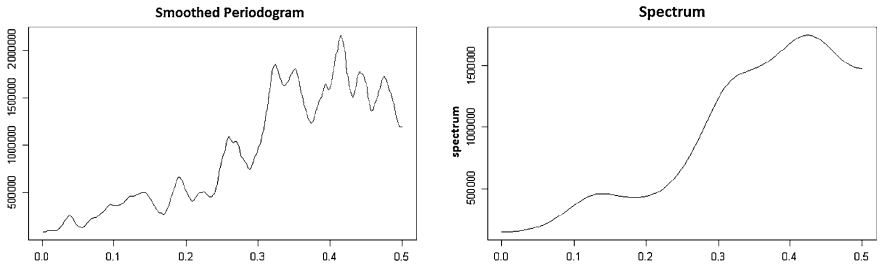

Fig. 3 Smoothed periodogram and spectrum for a share

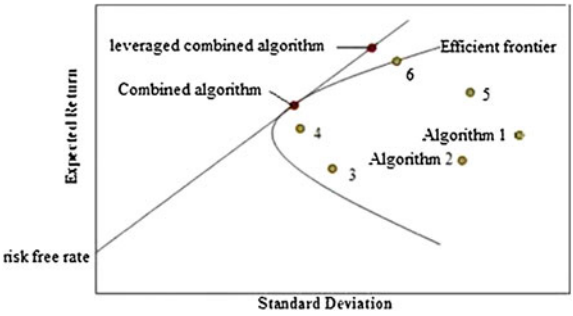

Fig. 4 An efficient frontier combination of the 6 algorithms

3. The maximum drawdown of the algorithm was 14 % smaller when compared with the best drawdown in the set of technical indicator strategies.
4. The length of the maximum interrupted loss was also shorter as compared to those of the other strategies.
5. We also noticed that the best performing technical indicators were those based on trend following; this is rather normal since the Hurst exponent [14] H ($(R/S)t = c*t^H$) was estimated to be around $0.624 > 0.5$, so the series were persistent and trend reinforcing.

4 Conclusions

This article confirms that technical indicators combined with neural artificial networks can produce effective trading strategies in emerging markets which exploit this inefficiency much better than the standard technical analysis strategies do. They also seem to automatically detect the type of indicators that usually best fit the market's acting mode and give them more weight and credit in the price prediction algorithms. Furthermore, the non linear structure of some ANNs provides us with adaptive filters which are more performing than standard filters applied on raw technical indicators.

Possible future evolution of the current strategy could rely upon further digital signal processing of the input technical indicators, using more elaborate filtering methods based, for instance, on Fisher and Hilbert transforms.

Acknowledgments This work is a part of CNCSIS TE_316 Grant « Intelligent Methods for Decision Fundamentation on Stock Market Transactions Based on Public Information », manager V.P. Bresfelean, Assistant Professor PhD.

References

1. Enke, D., Thawornwong, S.: The use of data mining and neural networks for forecasting stock market returns. Expert Syst. Appl. **29**(4), 927–940 (2005)
2. Farmer, D., Sidorowich, J.: Predicting chaotic time series. Phys. Rev. Lett. **59**, 845–848 (1987)
3. Fernández-Rodríguez, F., Sosvilla-Riveiro, S., García-Artile, M.: An empirical evaluation of non-linear trading rules (Working paper), FEDEA., **16** (2001)
4. Gallagher, L., Taylor, M.P.: Permanent and Temporary Components of Stock Prices: Evidence from Assessing Macroeconomic Shocks. South. Econ. J. **69**, 345–362 (2002)
5. Gately, E.: Neural networks for financial forecasting. Wiley, New York (1996)
6. de Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. Int. J. Forecast. **22**, 443–473 (2006)
7. Hayati, M., Shirvany, Y.: Artificial neural network approach for short term load forecasting for Illam Region. Proc. World Acad. Sci. Eng. Technol. **22** (2007)

8. Kavussanos, M.G., Dockery, E.: A multivariate test for stock market efficiency: the case of ASE. Appl. Finan. Econ. **11**(5), 573–579 (2001)
9. Lawrence, R.: Using neural networks to forecast stock market prices (1997) (Course project, University of Maritoba, Canada)
10. McNelis, D.P.: Neural networks in finance: gaining predictive edge in the market. Elsevier Academic Press, USA (2005)
11. Moreno, D., Olmeda, I.: The use of data mining and neural networks for forecasting stock market returns. Eur. J. Oper. Res. **182**(1), 436–457 (2007)
12. Priddy, L.K., Keller, E.P.: Artificial Neural Network: an introduction. SPIE Press, Washington–Bellington (2005)
13. Rotundo, G., Torozzi, B., Valente, M.: Neural networks for financial forecast, private communication (1999)
14. Qian, B., Rasheed, K.: Hurst exponent and financial market predictability, Proc. Finan. Appl. Eng. **437** (2004)
15. Walczak, S.: An empirical analysis of data requirements for financial forecasting with neural networks. J. Manage. Inf. Syst. **17**(4), 203–222 (2001)