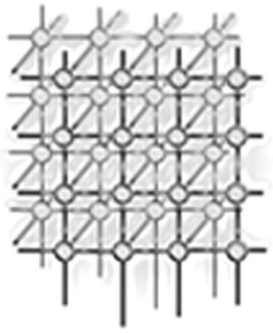
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2011; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

Fast Learning and Predicting of Stock Returns with VG-RAM Weightless Neural Networks



Alberto Ferreira De Souza[1,*,†], Fabio Daros Freitas [2,‡] and André Gustavo Coelho de Almeida[1,§]

[1] *Departamento de Informática, Universidade Federal do Espírito Santo, Vitória - ES, Brazil*
[2] *Receita Federal do Brasil, Vitória - ES, Brazil*

SUMMARY

We employ Virtual Generalized Random Access Memory weightless neural networks, VG-RAM WNN, for predicting future stock returns. We evaluated our VG-RAM WNN stock predictor architecture in predicting future weekly returns of the Brazilian stock market and obtained the same error levels and properties of baseline autoregressive neural network predictors; however, our VG-RAM WNN predictor runs 5,000 times faster than autoregressive neural network predictors. This allowed us to employ VG-RAM WNN predictors to build a high frequency trading system able to achieve a monthly return of approximately 35% in the Brazilian stock market.

KEY WORDS: high performance time series prediction; weightless neural networks; high frequency trading;

1. INTRODUCTION

Neural networks-based predictors have been successfully applied in predicting future stock returns and other financial variables, exhibiting many advantages over alternative methods [1–9]; however, little attention has been given to their computational performance requirements. Recently, trading firms started using sophisticated computer algorithms that must place hundreds of millions, even billions, of buy and sell orders a day—in some cases spending less than ten microseconds per order. Today, *High Frequency Trading* is a multibillion-dollar business, accounting for an estimated 50 to 70 percent

*Correspondence to: Alberto Ferreira De Souza, Universidade Federal do Espírito Santo, Centro Tecnológico, Departamento de Informática, Av. Fernando Ferrari 514, Goiabeiras, 29075-910 - Vitoria, ES - Brazil.
†E-mail: alberto@lcad.inf.ufes.br
‡E-mail: freitas@computer.org
§E-mail: andre@lcad.inf.ufes.br
Contract/grant sponsor: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil; contract/grant number: PQ 309831/2007-5, FACADOIC 620185/2008-2, PQ 308096/2010-0
Contract/grant sponsor: Fundação de Apoio à Ciência e Tecnologia do Espírito Santo (FAPES), Brazil; contract/grant number: PRONEX 48511579/2009

Received 25 March 2011

of the total U.S. equity market volume, and 28 percent of European and 16 percent of Asian order flows [10–12].

The standard approach for neural network time series predictor is the *Autoregressive Neural Network* (ARNN) predictor [13], also known as *focused time lagged feedforward network* [14], with p inputs—the present value and the $p-1$ past values of the series—whose output is an estimate of the series value for the next time period. After being trained, the ARNN predictor implements a non-linear multiple regression model of the time series using its past values as explanatory variables and network's weights as regression coefficients. But, their training procedure is intrinsically time consuming and difficult to parallelize. On the other hand, in RAM-based neural networks, also known as *weightless neural networks* (WNN) [15], training can be made in one shot and basically consists of storing the desired output in a memory position associated with the input of the neuron [16], thus being more time efficient and easier to parallelize than ARNN.

In this work, we present a new WNN-based time series predictor that uses Virtual Generalized Random Access Memory weightless neural network (VG-RAM WNN) [15] to predict future stock returns. Using stocks of the Brazilian stock market, we compared the performance of our VG-RAM WNN predictor with that achieved by ARNN predictors that were evaluated in previous works [17,18]. Our results showed that VG-RAM WNN predictors can produce predictions of future stock returns with the same error levels and properties of baseline ARNN predictors; however, running $5,000$ times faster than ARNN predictors. This allowed us to employ VG-RAM WNN predictors to build a straightforward high frequency trading system. A preliminary evaluation of our trade system showed that it can achieve a monthly return of 35% in the Brazilian stock market.

The remainder of this paper is organized as follows. After this introduction, Section 2 briefly presents the ARNN predictor used in our previous works [17,18], while Section 3 presents our VG-RAM WNN-based approach for predicting stock returns. Our experimental methods for comparing the VG-RAM WNN and ARNN predictors and the achieved results are presented in Section 4. In Section 5, we present our high frequency trading system and preliminary experiments to evaluate its performance. Our conclusions and directions for future works are presented in Section 6.

A preliminary version of this work appeared in [19].

2. ARNN PREDICTORS

The one-period stock return at time t, r_t, can be defined as:

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}} \qquad t \geq 1, \tag{1}$$

where P_t and P_{t-1} are the stock prices at times t and $t-1$, respectively, and the series of R past returns of a stock, $\mathbf{r}^{'}$, can be defined as:

$$\mathbf{r}^{'} = (r_1, r_2, \ldots, r_R). \tag{2}$$

The baseline ARNN predictor that we used in this work is the *Autoregressive Neural Network* (ARNN) predictor [13,14]. The ARNN predictor receives as input the present return and the past $p-1$ returns of a series and produces a future return as output. An $ARNN(p)$, then, implements a non-linear prediction system, $\mathcal{S}_{\mathcal{R}}$, that has regression order p and inputs and outputs as shown in Equation 3:

$$r_{t-(p-1)}, \ldots, r_t \rightarrow \mathcal{S}_{\mathcal{R}} \rightarrow \hat{r}_{t+1}. \tag{3}$$

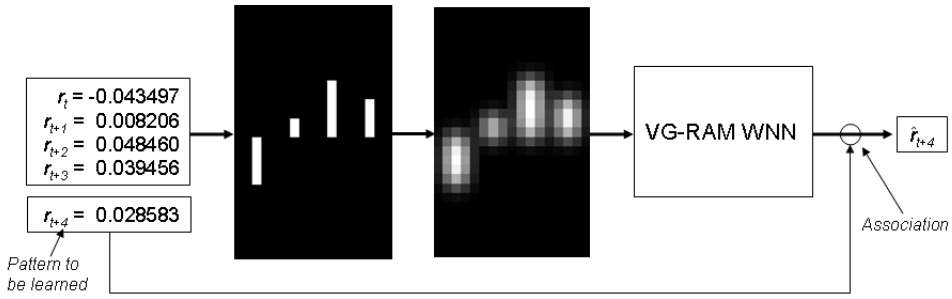


Figure 1. Schematic diagram of WNN-SPA.

We compare VG-RAM WNN with ARNN predictors trained with the *back-propagation algorithm* [14]. The training methods employed with our ARNN predictors are detailed in [19].

3. VG-RAM WNN PREDICTORS

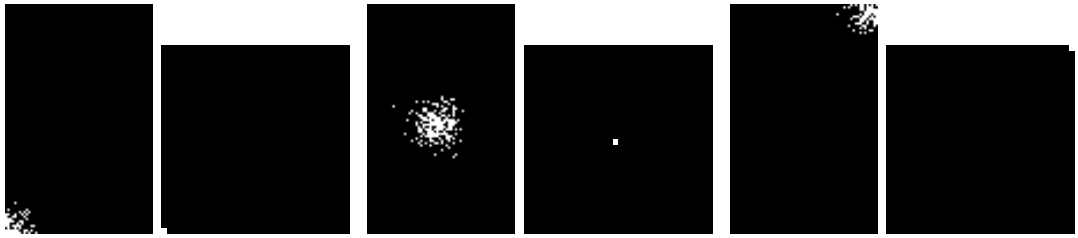
Despite of their remarkable simplicity, RAM-based neural networks are very effective as pattern recognition tools, offering fast training and test, in addition to easy implementation [20]. However, if the network input is too large, the memory size becomes prohibitive, since it must be equal to 2^n, where n is the input size [15].

Virtual Generalizing Random Access Weightless Neural Networks (VG-RAM WNN) are RAM-based neural networks that only require memory capacity to store the data related to the training set [15]. In the neurons of these networks, the memory stores the input-output pairs shown during training, instead of only the output. In the test phase, the memory of VG-RAM WNN neurons is searched associatively by comparing the input presented to the network with all inputs in the input-output pairs learned. The output of each VG-RAM WNN neuron is taken from the pair whose input is nearest to the input presented—the distance function employed by VG-RAM WNN neurons is the *hamming distance*. If there is more than one pair at the same minimum distance from the input presented, the neuron's output is chosen randomly among these pairs.

3.1. VG-RAM WNN stock predictor architecture

We propose a VG-RAM WNN stock predictor architecture (WNN-SPA) that is similar to one we have designed for face recognition [21]. In that architecture, neurons were trained to inform a person's code from her image; similarly, in the WNN-SPA, neurons are trained to inform a return time index from an image equivalent to a set of p past returns, as shown in Figure 1 for $p = 4$. With this approach, we try and mimic a way one typically inspects a time series graph to guess its future value. In this task, one tends to concentrate the visual attention in the last points of the graph, creating an imaginary frame that delimits this region of the graph and offers an image with, hopefully, sufficient visual information for extrapolating the next value of the series.

The WNN-SPA has many neurons and, during training, for a given set of p returns, all of these neurons are trained to associate the image of this set with the time index of the next return; i.e., the



(a) (b) (c)

Figure 2. The synaptic interconnection pattern of WNN-SPA's VG-RAM WNN. (a) Left, input Φ: in white, the elements $\varphi_{k,l}$ of the input Φ that are connected to neuron $n_{1,1}$ of N via $\Omega_{1,1}(W)$. Right, neuron array N: in white, the neuron $n_{1,1}$ of N. (b) Left: in white, the elements $\varphi_{k,l}$ of Φ connected to $n_{\frac{m}{2},\frac{n}{2}}$ via $\Omega_{\frac{m}{2},\frac{n}{2}}(W)$. Right: in white, the neuron $n_{\frac{m}{2},\frac{n}{2}}$ of N. (c) Left: in white, the elements of Φ connected to $n_{m,n}$ via $\Omega_{m,n}(W)$. Right: in white, the neuron $n_{m,n}$.

neurons try to "remember" a similar situation in the past to produce the prediction of the future return, $\hat{r}_{t+1}$. During test, an image of the present return and the past $p-1$ returns is presented to the network and each neuron outputs the time index of a return seen during training. These indexes are used to get returns from the training database, which are averaged to produce the output of the network, that is the prediction of the future return, $\hat{r}_{t+1}$.

Figure 1 shows a diagram that illustrates how the WNN-SPA operates. In the figure, a series of four returns is transformed into a bar graph image where the size of each bar is equivalent to the magnitude of a return of the series—bars representing negative returns starts in the middle of the image and grow downwards, while bars representing positive returns starts in the middle of the image grow upwards. To make the bar graph look even more like an image, we filtered it with a Gaussian filter and the output of this filter that is actually the input of WNN-SPA.

Virtual generalized random access memory weightless neural network stock predictor architecture has a single bidimensional array of $m \times n$ VG-RAM WNN neurons, N, where each neuron, $n_{i,j}$, has a set of synapses $W = \{w_1, \ldots, w_{|W|}\}$, which are randomly connected to the network's bidimensional input, Φ, of $u \times v$ inputs, $\varphi_{k,l}$ (see Figures 2 and 3). The random synaptic interconnection pattern of each neuron $n_{i,j}$, $\Omega_{i,j,\sigma}(W)$, follows a bidimensional Normal distribution with variance σ^2 centered at φ_{μ_k,μ_l}, where $\mu_k = \frac{i.u}{m}$ and $\mu_l = \frac{j.v}{n}$; i.e., the coordinates k and l of the elements of Φ to which $n_{i,j}$ connect via W follow the probability density functions:

$$\omega_{\mu_k,\sigma^2}(k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(k-\mu_k)^2}{2\sigma^2}}, \tag{4}$$

$$\omega_{\mu_l,\sigma^2}(l) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(l-\mu_l)^2}{2\sigma^2}}, \tag{5}$$

where σ is a parameter of the architecture. This synaptic interconnection pattern mimics that observed in many classes of biological neurons [22], and is created when the network is built and does not change afterwards.

Virtual generalized random access memory weightless neural network synapses can only get a single bit from the input. Thus, in order to allow our VG-RAM WNN to deal with images, in which a pixel

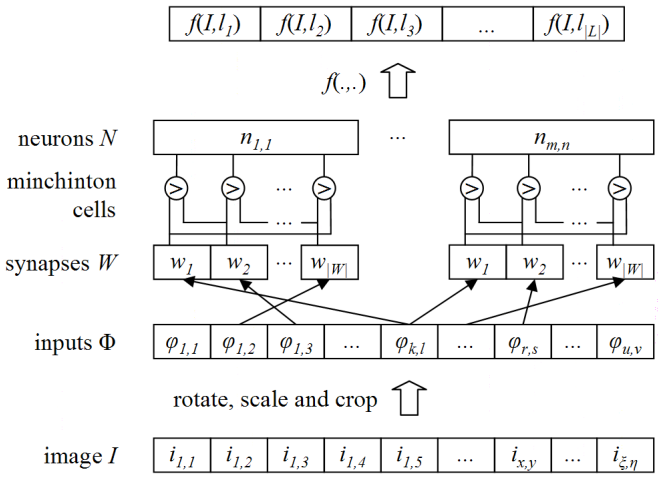


Figure 3. Schematic diagram of WNN-SPA's VG-RAM WNN.

may assume a range of different values, we use *minchinton cells* [23]. In the proposed VG-RAM WNN architecture, each neuron's synapse, w_t, forms a minchinton cell with the next, w_{t+1} ($w_{|W|}$ forms a minchinton cell with w_1). The type of the minchinton cell we have used returns 1 if the synapse w_t of the cell is connected to an input element, $\varphi_{k,l}$, whose value is larger than the value of the element $\varphi_{r,s}$ to which the synapse w_{t+1} is connected, i.e., $\varphi_{k,l} > \varphi_{r,s}$; otherwise, it returns 0 (see the synapses w_1 and w_2 of the neuron $n_{m,n}$ of Figure 3).

During training, the bar graph image I_x of the p past values of r_x, $r_{x-p}, \ldots, r_{x-1}$, is copied to the VG-RAM WNN's input Φ and all $n_{i,j}$ neurons' outputs are set to the time index of r_x. All neurons are then trained to output the time index of the input image I_x. This procedure is repeated for all images I_x in the training data set. During testing, each bar graph image I_y of the p past values of r_y, $r_{y-p}, \ldots, r_{y-1}$, in the testing data set is also copied to the VG-RAM WNN's input Φ. After that, all neurons' outputs are computed and the returns associated with these outputs are averaged. The WNN-SPA outputs this average return as the predicted return $\hat{r}_y$.

4. VG-RAM WNN PREDICTIONS EVALUATION

This section presents the experiments we have used to evaluate the performance of our VG-RAM WNN predictors. This evaluation consisted in predicting future stock returns of the same data set we have used in previous works that evaluated ARNN stock predictors [17, 18] and comparing its error levels and properties.

4.1. Data

We selected a subset of 46 stocks from those that participated in the IBOVESPA index on the first quarter of 2005 and had long enough time series for training the neural networks. We have chosen

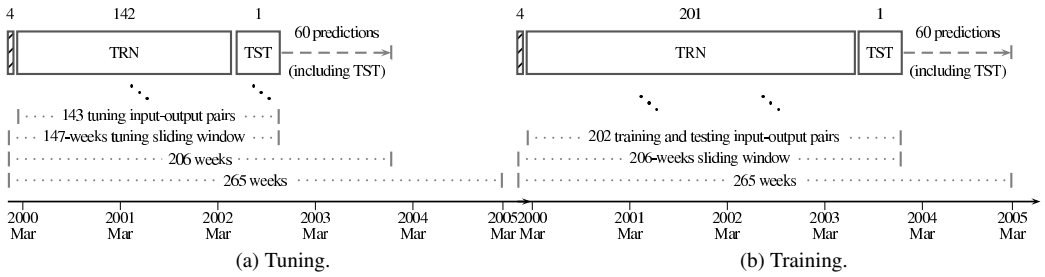


Figure 4. Sliding windows for (a) tuning and (b) training the VG-RAM WNN predictors.

stocks of the IBOVESPA because it is the main index of the BM&FBOVESPA stock exchange, the main Brazilian stock market.[†] IBOVESPA is widely used to gauge the state of the Brazilian stock market and we are very acquainted to it, which was very important during the preliminary studies that lead to this work; currently, BM&FBOVESPA is the 3rd largest stock market in the World. We believe that any set of stocks of an equivalent index of any major stock market could be used for the experiments described in this paper and would show equivalent results.

For each one of the 46 stocks, we computed the 265 weekly returns of the period between 16-February-2000 and 9-March-2005 using closing prices sampled at Wednesdays to avoid beginning-of-the-week and end-of-the-week effects [24, 25]. We have used the last 60 observations of our data set, between 21-January-2004 and 9-March-2005, to evaluate the performance of our predictors. In all cases of missing data, we used the last daily closing price available.

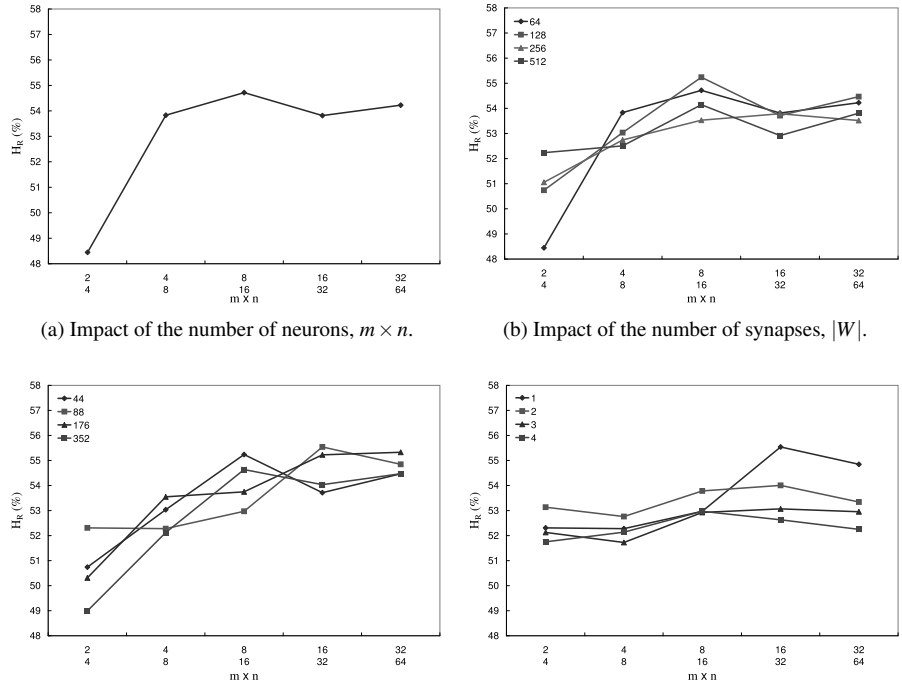
4.2. Parameters tuning of virtual generalized random access memory weightless neural network

We run tuning experiments to empirically try and find proper topological parameters of the WNN-SPA. As mentioned in Section 3.1, the WNN-SPA has a single bidimensional array of $m \times n$ VG-RAM WNN neurons, and each neuron has a set of synapses of size $|W|$ which are randomly connected to the network's bidimensional input Φ of $u \times v$ pixels according to a synaptic interconnection pattern that follows a bidimensional Normal distribution with variance σ^2.

We have used the H_R performance metric in the tuning experiments. In short, H_R measures the percentage of WNN-SPA predictions that are different from zero and have the same sign of the return of the series being predicted. H_R is formally presented in Section 4.4 (see Equation 10).

We have searched for the best parameter values for $m \times n$ in the set $\{2 \times 4, 4 \times 8, 8 \times 16, 16 \times 32, 32 \times 64\}$, for $|W|$ in the set $\{64, 128, 256, 512\}$, for $u \times v$ in the set $\{17 \times 44, 17 \times 88, 17 \times 176, 17 \times 352\}$ (we have fixed u to reduce the search space), and for σ in the set $\{1, 2, 3, 4\}$—a total of 400 parameter set possibilities. To train and validate (search for the best parameters) the WNN-SPA in tuning mode, we used a sliding window with the first 147 of the 265 weekly returns available. This allowed 60 predictions for choosing the best parameters within the 400 possibilities; i.e., 206

[†]See BMF&BOVESPA—The New Exchange at `http://www.bmfbovespa.com.br/en-us/home.aspx?idioma=en-us`



(a) Impact of the number of neurons, $m \times n$.

(b) Impact of the number of synapses, $|W|$.

(c) Impact of v.

(d) Impact of σ.

Figure 5. Impact of the parameters of the WNN-SPA on its performance.

returns (from ARNN training sliding window size) minus 147 (WNN-SPA tuning sliding window size, including the four returns necessary as the first four inputs of the WNN-SPA and the first predicted return), plus one (the prediction of the initial window). This sliding window scheme is shown in Figure 4(a), which illustrates the first sliding window position and its training (TRN) and testing (TST) segment sizes (in weeks). Therefore, we had $1,104,000$ training sessions during the WNN-SPA tuning. (60 train and validate cycles $\times$ 46 stocks $\times$ 400 parameter combinations $= 1,104,000$). Figure 5 shows the results of our tuning experiments.

The graph of Figure 5(a) allows one to appreciate the impact of the number of neurons on the performance of WNN-SPA. In Figure 5(a), the x-axis is the number of neurons ($m \times n$), while the y-axis is performance of WNN-SPA in terms of H_R. To produce the results in this graph, we set the values of $|W|$, v and σ to their minimum (64, 44, and 1, respectively; see the ranges considered for each parameter above) and varied the value of $m \times n$ in the range $\{2 \times 4, 4 \times 8, 8 \times 16, 16 \times 32, 32 \times 64\}$. As the graph of Figure 5(a) shows, the performance of WNN-SPA increases as the number of neurons increases, but reaches a plateau at about 8×16 neurons for $|W| = 64$, $v = 44$ and $\sigma = 1$.

The graph of Figure 5(b) presents the impact of the number of synapses, $|W|$, on the performance of WNN-SPA. It has the same format of Figure 5(a), but includes one curve for each value of $|W|$ considered (see Figure 5(b)'s legend). To produce the results in this graph, we set the values of v and σ to their minimum (44 and 1, respectively), varied the value of $|W|$ in the range $\{64, 128, 256, 512\}$, and plotted one curve for each value of $|W|$. As the graph of Figure 5(b) shows, the performance of WNN-

SPA increases as the number of synapses increases, but reaches a maximum at about 128 synapses and then starts to decrease. So, we choose 128 as the number of synapses of the WNN-SPA.

The graph of Figure 5(c) presents the impact of the height of the input Φ, v, on the performance of WNN-SPA. It includes one curve for each value of v considered (see Figure 5(c)'s legend). To produce the results in this graph, we set the value of $|W|$ to 128, of σ to its minimum ($\sigma = 1$), varied the value of v in the range $\{44, 88, 176, 352\}$, and plotted one curve for each value of v. As the graph of Figure 5(c) shows, the performance of WNN-SPA overall increases as v increases, but reaches a maximum at about 88, with $m \times n$ equal to 16×32, and then starts to decrease. So, we choose 88 as value of v.

The graph of Figure 5(d) presents the impact of σ on the performance of WNN-SPA. To produce the results in this graph we set the value of $|W|$ to 128, of v to 88, and plotted one curve for each value of σ considered. As the graph of Figure 5(d) shows, the best performance of WNN-SPA occurs when $\sigma = 1$. So, we choose one as value of σ.

Hence, from the tuning experiments we found that the best WNN-SPA parameters on our set-up are: $m \times n$ equal to 16×32 neurons, $|W|$ equal to 128 synapses, $u \times v$ equal to 17×88 pixels, and σ equal to one pixel. It is important to note that, for poor parameter values (the left side of the graphs of Figure 5), the WNN-SPA performance is close to chance ($H_R = 50\%$). However, with tuning, the WNN-SPA is able to predict returns with accuracy suitable for many interesting stock market applications.

4.3. Virtual generalized random access memory weightless neural network predictions

To train and test (use for predictions) the tuned WNN-SPA, we used a sliding window of 206 of the 265 weekly returns available. This allowed 60 predictions, i.e. the 265 returns (from our full data set) minus 206 (sliding window size, including the four returns necessary as the first four inputs of the WNN-SPA and the first predicted return), plus one (the prediction of the initial window). Therefore, we had also 2,760 training sessions (60 train and test cycles $\times$ 46 stocks $= 2{,}760$). The training and testing procedure was repeated for all 60 predictions by advancing the sliding window of 206 weeks, one week at a time. This sliding window scheme is shown in Figure 4(b), which illustrates the first sliding window position and its training (TRN) and testing (TST) segment sizes (in weeks).

It is also important to note that training at the arrival of new data leads to better predictions.

4.4. Evaluation metrics

We used the *Mean Error*, *Root Mean Square Error*, *Mean Absolute Percentage Error*, *Theil's U*, and *Hit Rates* metrics to evaluate our predictor's performance.

Mean Error (ME) is the average difference between the realized and predicted returns, defined as:

$$ME = \frac{1}{n} \sum_{t=1}^{n} r_t - \hat{r}_t, \tag{6}$$

where n is the length of the time series, and r_t and $\hat{r}_t$ are the realized and predicted returns at time t, respectively.

Root Mean Square Error (*RMSE*) is a standard metric for evaluating the differences between two time series and it is defined for the case of differences between returns and predicted returns as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(r_t - \hat{r}_t)^2}. \tag{7}$$

RMSE may be interpreted as the standard deviation of the errors of prediction $(r_t - \hat{r}_t)$ with respect to a zero mean, and it shows the distance of these errors from the ideal situation of zero mean error—see Equation 6. *RMSE* has low outlier protection, good sensitivity for small changes in data, and does not display data asymmetry [26].

Mean Absolute Percentage Error (*MAPE*) is defined as:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{r_t - \hat{r}_t}{r_t}\right|. \tag{8}$$

MAPE is a unit-free measure, has good sensitivity for small changes in data, does not display data asymmetry and has very low outlier protection [26]. It is important to note that, when r_t is very close to zero and $\hat{r}_t$ is not, *MAPE* can average to a large value.

Theil's U (U) is defined as:

$$U = \sqrt{\frac{\sum_{t=1}^{n-1}\left(\frac{\hat{r}_{t+1} - r_{t+1}}{r_t}\right)^2}{\sum_{t=1}^{n-1}\left(\frac{r_{t+1} - r_t}{r_t}\right)^2}}. \tag{9}$$

Theil's U compares the performance of the evaluating predictor with that of the *naïve* predictor. If $U = 0$, the evaluating predictor is a perfect one; if $U = 1$, it is performing like the *naïve* predictor; and if $U > 1$, it is performing worse than the *naïve* predictor [27].

Hit Rates H_R, H_{R+}, and H_{R-} measure the percentage of predictions whose signs of $\hat{r}$ and r coincide: H_R is the percentage of predictions were both have the same sign and are different from zero, H_{R+} is the same percentage were both are positive, and H_{R-} is the same percentage were both are negative. These metrics are suitable for evaluating predictors as trading signals generators [28], and can be written as:

$$H_R = \frac{Count_{t=1}^{n}(r_t\,\hat{r}_t > 0)}{Count_{t=1}^{n}(r_t\,\hat{r}_t \neq 0)}, \tag{10}$$

$$H_{R+} = \frac{Count_{t=1}^{n}(r_t > 0\ AND\ \hat{r}_t > 0)}{Count_{t=1}^{n}(\hat{r}_t > 0)}, \tag{11}$$

$$H_{R-} = \frac{Count_{t=1}^{n}(r_t < 0\ AND\ \hat{r}_t < 0)}{Count_{t=1}^{n}(\hat{r}_t < 0)}, \tag{12}$$

where, $Count(\cdot)$ is the counting of occurrences of its argument.

	min.	max.	avg.	σ
ME	−0.0248	0.0100	−0.0047	0.0067
RMSE	0.0400	0.0800	0.0543	0.0119
MAPE	0.7500	6.7100	1.6415	0.9830
U	0.2700	3.6500	1.0489	0.4633
H_R	0.4035	0.6552	0.5376	0.0613
H_{R+}	0.3529	0.6250	0.5249	0.0772
H_{R-}	0.0000	1.0000	0.3655	0.2266

(a) VG-RAM WNN predictions.

	min.	max.	avg.	σ
ME	−0.0142	0.0090	−0.0037	0.0052
RMSE	0.0380	0.0887	0.0544	0.0120
MAPE	0.7861	307.0927	20.4742	64.2316
U	0.4200	1.7700	1.0113	0.2555
H_R	0.3900	0.6600	0.5365	0.0696
H_{R+}	0.3500	0.6600	0.5313	0.0833
H_{R-}	0.1100	1.0000	0.4780	0.1492

(b) $ARNN(4)$ predictions.

Table I. Summary of the predictions obtained for the 46 stocks ($n = 60$).

4.5. Experimental results

4.5.1. Performance of the virtual generalized random access memory weightless neural network predictions

One VG-RAM WNN and, using the same procedure (see [19] for details), one $ARNN(4)$ predictor, were trained and tested for each of the 46 stocks and used for obtaining each of the 60 predictions of future returns, totalizing 5,520 predictions of future returns ($2 \times 46 \times 60 = 5,520$). Table I summarizes the performance of VG-RAM WNN and $ARNN(4)$ predictions with the evaluation metrics of Section 4.4 and is organized as follows. The results for VG-RAM WNN predictions are shown at Table Ia, while the results for $ARNN(4)$ predictions are shown at Table Ib. The achieved results for the evaluation metrics of Section 4.4 are shown in the rows of the table, with its minimum (min.), maximum (max.), average (avg.), and standard deviation (σ) values shown in each respective column.

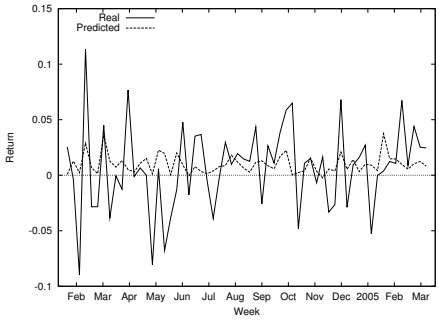
As Table I shows, the average *ME* of both predictors was close to zero with a low standard deviation, thus indicating unbiased predictions.

RMSE, *MAPE* and H_R achieved typical levels for this application [28, 29]. *RMSE* of both neural models achieved very similar performances, while VG-RAM WNN performed better than $ARNN(4)$ in *MAPE* results, reflected by its lower maximum and average values. Both predictors produced average H_R around 53% and maximum values near 66%, thus achieving a peak performance of 16 percentage points above chance (50%). H_{R+} performances were also similar, and VG-RAM WNN exhibited H_{R-} lower minimum values than $ARNN(4)$.

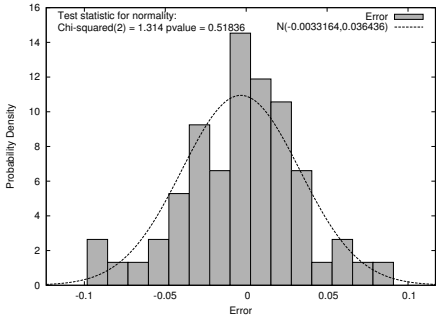
H_R reported in literature are typically near 55% [28], and in our experiments 21 of 46 VG-RAM WNN and $ARNN(4)$ predictors exceeded this performance level; also seven VG-RAM WNN and nine $ARNN(4)$ predictors exceeded a H_R of 60%.

Apart from *ME* and peak H_R, the performance of the predictions was similar and modest—correctly predicting the time series of stock returns is recognized as a difficult task [28].

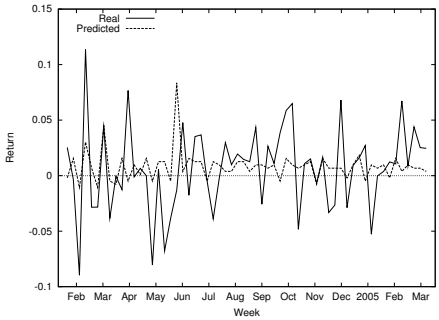
Figure 6 shows a sample of the results of VG-RAM WNN and $ARNN(4)$ predictions for the PETROBRAS PN (PETR4) stock. Figure 6(a) and Figure 6(c) show the real and predicted values for the 60 weeks between 21-January-2004 and 9-March-2005, and Figure 6(b) and Figure 6(d) show the frequency distributions for the prediction errors, respectively, for VG-RAM WNN and $ARNN(4)$ predictors. As Figure 6 shows, both predictors exhibited similar behavior with respect to predicted values and frequency distributions.



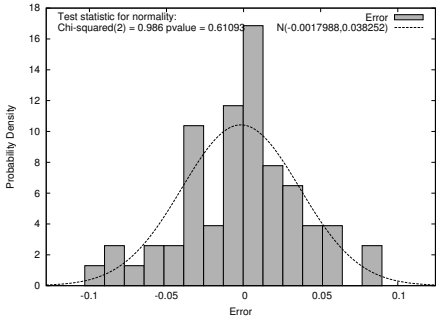
(a) VG-RAM WNN predictions.



(b) Distribution of VG-RAM WNN prediction errors.



(c) $ARNN(4)$ predictions.



(d) Distribution of $ARNN(4)$ prediction errors.

Figure 6. VG-RAM WNN and $ARNN(4)$ predictions for PETROBRAS PN (PETR4): (a) and (c) real and predicted weekly returns, and (b) and (d) frequency distribution of prediction errors, respectively.

4.5.2. Computational performance of the virtual generalized random access memory weightless neural network predictors

Our benchmark experiments were executed on PC machines with Intel Core 2 Duo Processor T8100 (3M Cache, 2.10 GHz, 800 MHz FSB) and 3 GB RAM (DDR-2 800 MHz). $ARNN(4)$ predictors spent approximately $18,000$ seconds (i.e. five hours), on average, to run the 60 training sessions for a stock (i.e. to train and predict the 60 future returns—see [19] for details). VG-RAM WNN predictors, on the other hand, spent only 3.6 seconds on average to do the same task using the methods of Section 4.3, thus obtaining a $5,000$ times speedup ($18,000/3,6 = 5,000$). This means that a VG-RAM WNN predictor can perform a training session for predicting a future stock return in only 6 milliseconds.

It is important to note that, as the VG-RAM WNN parameters tuning procedure of Section 4.2 was conducted only once, it did not contribute to VG-RAM WNN's computing time.

5. A VIRTUAL GENERALIZED RANDOM ACCESS MEMORY WEIGHTLESS NEURAL NETWORK-BASED TRADING SYSTEM

This section presents a preliminary study of a trading system that employs VG-RAM WNN predictors in high frequency trading. Our trading system buys and sells a single stock at every four minutes (it does not perform short selling). We selected four minute intervals because, within this period of time, there is enough price variation of the stocks of interest to make a profit with a modest wealth (considering trading costs).

5.1. Data

We selected a subset of 49 stocks from those that participated in the IBOVESPA index in September 2010. We used each stock's minute-by-minute prices to compute four-minute returns of 21 trading days between 27-August-2010 and 30-September-2010[‡]. In this period, the BM&FBOVESPA stock exchange had regular sessions between 10 am and 5 pm, allowing a total of 105 four-minute ticks a day (7 hours $\times$ 60 minutes $/4$ minutes $= 105$). Thus, we obtained a total of $108,045$ four-minute returns as our data set for the trading system experiment (49 stocks $\times$ 21 days $\times 105 = 108,045$). In all cases of missing data, we used the last available price.

5.2. Trading strategy

Our trading strategy for intraday trading using VG-RAM WNN predictors was straightforward. At each time t during a stock exchange's regular session, we analyze a set of stocks and predict its next four-minute return, $\hat{r}_{t+1}$, searching for the best investment opportunity (i.e., the best stock to invest). We then buy this stock at market price at time t and sell it four minutes later, at time $t+1$, also at market price.

In order to select the best stock to invest, we compute a rank measure for each stock that is based on its predicted return value, $\hat{r}_{t+1}$, and on the H_{R+} performance metric of its predictor (see Section 4.4). This rank measure, $R(t)$, is defined as:

$$R(t) = \frac{3\,\hat{r}^*_{t+1} + H^*_{R+}(t)}{4} \tag{13}$$

where $R(t)$ is the stock's rank measure for time t, $\hat{r}^*_{t+1}$ is its normalized predicted return for time $t+1$, and $H^*_{R+}(t)$ is its normalized H_{R+} measure calculated on a validation set containing previous predictions. These normalized values were obtained through normalizing the respective variables values of all stocks in $[0,1]$. Thus, to produce $\hat{r}^*_{t+1}$ of each stock at time t, the highest $\hat{r}_{t+1}$ observed for all stocks in time t is adjusted to 1 and the lowest to 0, while the remaining are adjusted according to their proportions. The same is done with $H_{R+}(t)$ to produce $H^*_{R+}(t)$.

At each time t, the stock with the highest rank value, $R(t)$, is selected for investing; i.e., it is bought and, four minutes later, at time $t+1$, it is sold. The gross realized return is then computed as in

[‡]There were a total of 24 regular sessions in the BM&FBOVESPA stock exchange between 27-August-2010 and 30-September-2010, but we obtained data for just 21 of them.

Equation 1. From this gross realized return we subtract all incident transaction costs related to the buy and sell orders and compute the net realized return of the trade.[§]

We repeat this procedure for all four-minute ticks of every day's regular session and, from the net realized returns of all trades of a day, we compute the day's total realized return.

5.3. Virtual generalized random access memory weightless neural network intraday predictions

We employed VG-RAM WNN predictors with the best WNN-SPA parameters found in Section 4.2. Thus, we have used VG-RAM WNN predictors with $m \times n$ equal to 16×32 neurons, $|W|$ equal to 128 synapses, $u \times v$ equal to 17×88 pixels, and σ equal to one pixel (see Section 3.1 for parameters description).

To train, validate and test (use for on-line predictions) the VG-RAM WNN predictors employed in the trading strategy of Section 5.2 in a given day, we used the 49 stocks' time series of four-minute returns of the previous three days. The first two of these three days are used only for training, while the last is used for training and validation; i.e., in this day, we make predictions and compare them with the observed returns in order to compute the $H^*_{R+}(t)$ of each stock. At this point we can start on-line predictions and trading. For example, to predict the four-minute on-line returns and trade in 1-September-2010, we use data from 27-August-2010, 30-August-2010, and 31-August-2010. The first two days' data, from 27-August-2010 and 30-August-2010, are used to train the VG-RAM WNN predictors as described in Section 3; while the last day's data, from 31-August-2010, are used as a validation set for computing the $H^*_{R+}(t)$ metric used in the stocks' rank measure, $R(t)$ (see Equation 13), required by the trading strategy of Section 5.2.

Thus, for the first prediction of a given day, we train the VG-RAM WNN predictors with a training set of 210 four-minute returns (two days) and a validation set of 105 four-minute returns (one day). For the second prediction, we use a training set with 211 four-minute returns and a validation set with 105 four-minute returns that preceded the current observation. The remaining predictions of that day were obtained in the same way, augmenting the training set window and advancing the validation set's fixed-size sliding window one observation at a time.

The first four returns of the 105 four-minute daily returns are used as the first four inputs that the stocks' VG-RAM WNN predictors use to produce the first day's prediction (see Section 3.1). This first day's prediction is used by the trading system to select the stock for the first day's trade; thus, we have 101 daily predictions for each stock and 101 trades (a buy order followed by a sell order) a day. This allowed a total of $103,929$ predictions (49 stocks $\times 21$ days $\times 101 = 103,929$) that were employed to produce the $3,636$ buy and sell trades (18 days $\times 2$ orders $\times 101 = 3,636$) simulated in the period of the experiments.

[§]The transaction costs considered were broker commissions, stock exchange's fees and incident taxes. Brazilian income tax was not considered because it has a complex variable tax rate and deduction structure.

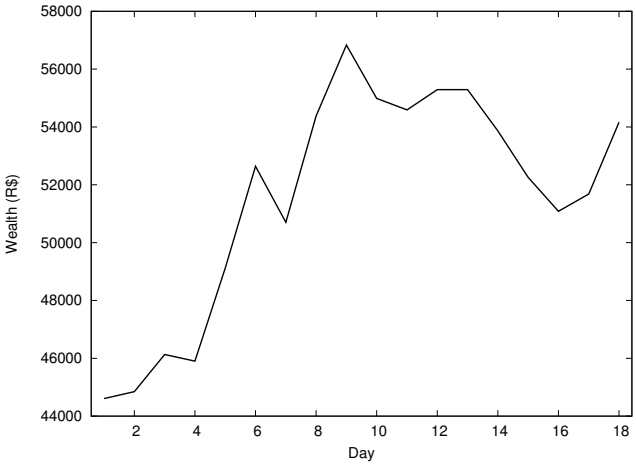


Figure 7. Wealth accumulation with the VG-RAM WNN-based trading system over the 18 trading days.

5.4. Experimental results

Our trading simulations used real data from Brazilian stock market, so all results are expressed in Brazilian Real currency (R$). We believe that in any other similar market these experiments would show equivalent results.

We started 1-September-2010, day $d = 1$, investing an initial amount, $W_i(d)$, of R$ 40,000 (U$ 23,812.93 in that date) in the first selected stock. After selling this first stock four minutes later, we compute the current obtained wealth, $W_t(d)$, considering a broker commission of R$ 5 per order (R$ 10 per trade), a good and services tax rate of 2% over the broker commission, and BM&FBOVESPA stock exchange fees of 0.026% over the traded volume. We repeat this procedure, as described in Section 5.2, until the last trade of that day, when we compute the final obtained wealth, $W_f(d)$, for that day. We then save the earnings of this day and start over the next day investing the same R$ 40,000, proceeding the same way until the last day of our experimental period, day $d = 18$ (30-September-2010). Figure 7 shows the result of our experiment.

In Figure 7, the x-axis is the day of trade, whereas the y-axis is the accumulated wealth. As the graph of Figure 7 shows, our trading system was able to capture good opportunities of trade in the period considered—from an initial wealth of R$ 40,000, our trading system provided a final wealth of R$ 54,164.21, an increase of 35.4% in one month.

During the period considered, a daily volume of approximately R$ 8,000,000 was generated from a wealth of R$ 40,000 (R$ 40,000 $\times$ 202 trades). In spite of this daily volume, only a return of approximately 2% per day—about R$ 800 of the R$ 40,000 invested daily—was obtained. This is important because the volume generated contributes for market liquidity—the daily wealth invested hops from stock to stock during the day—but does not produce a large market impact.

The graph of Figure 7 also shows that there are several days with negative returns. In fact, we obtained an average daily return of 0.0198 (1.98%) with a standard deviation of 0.0523. But, a one

sample t-test of this mean shows that it is significantly larger than zero (one sample t-test for mean at 5% significance level—p-value equal to 0.06451).

6. CONCLUSION

In this work, we proposed a new WNN-based time series predictor that uses Virtual Generalized Random Access Memory weightless neural networks (VG-RAM WNN) for predicting future stock returns. Our VG-RAM WNN stock predictor architecture (WNN-SPA) is similar to one we have designed for face recognition. In that architecture, neurons were trained to inform a person's code from her image; similarly, in the WNN-SPA, neurons are trained to inform a return time index from an image equivalent to a set of past returns.

This new VG-RAM WNN-based time series predictor was evaluated in predicting future weekly returns of stocks of the Brazilian stock market. Our results showed that VG-RAM WNN predictors can produce predictions of future stock returns with the same error levels and properties of baseline autoregressive neural network predictors; however, running $5,000$ times faster—a VG-RAM WNN predictor performed a training session for predicting a future stock return in only 6 milliseconds. This allowed us to employ VG-RAM WNN predictors to build a high frequency trading system. Our trade system achieved a monthly return of approximately 35%.

REFERENCES

1. Sharda R, Patil RB. A connectionist approach to time series prediction: An empirical test. *Journal of Intelligent Manufacturing* 1992; **3**(5):317–323.
2. Moody J. Prediction risk and architecture selection for neural networks. *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, Cherkassky V, Friedman JH, Wechsler H (eds.). Springer, NATO ASI Series F, 1994.
3. Moody J. Economic forecasting: Challenges and neural solutions. *International Symposium on Artificial Neural Networks*, Hsinchu, Taiwan, 1995. Keynote Talk.
4. Hansen J, Nelson R. Neural networks and traditional time series methods: a synergistic combination in state economic forecasts. *IEEE Trans. Neural Netw.* Jul 1997; **8**(4):863–873, doi:10.1109/72.595884.
5. Chen Y, Yang B, Dong J, Abraham A. Time-series forecasting using flexible neural tree model. *Information Sciences* Aug 2005; **174**(3–4):219–235.
6. Chen Y, Yang B, Abraham A. Flexible neural trees ensemble for stock index modeling. *Neurocomputing* Jan 2007; **70**(4-6):697–703.
7. Pantazopoulos K, Tsoukalas L, Bourbakis N, Brun M, Houstis E. Financial prediction and trading strategies using neurofuzzy approaches. *IEEE Trans. Syst., Man, Cybern. B* Aug 1998; **28**(4):520–531, doi:10.1109/3477.704291.
8. Ferreira TA, Vasconcelos GC, Adeodato PJ. A new intelligent system methodology for time series forecasting with artificial neural networks. *Neural Process. Lett.* 2008; **28**(2):113–129, doi:http://dx.doi.org/10.1007/s11063-008-9085-x.
9. Freitas FD, De Souza AF, Almeida AR. Prediction-based portfolio optimization model using neural networks. *Neurocomputing* Jun 2009; **72**(10–12):2155–2170.
10. Brabazon A, O'Neill M, Dempsey I. An introduction to evolutionary computation in finance. *IEEE Comput. Intell. Mag.* 2008; **3**(24):42–55, doi:10.1109/MCI.2008.929841.
11. Peltz M. Inside the machine: A journey into the world of high-frequency trading. *Institutional Investor* Jun 2010; **45**(5):42–48, 90–93.
12. Hendershott T, Jones CM, Menkveld AJ. Does algorithmic trading improve liquidity? *The Journal of Finance* Feb 2011; **66**(1):1–33, doi:10.1111/j.1540-6261.2010.01624.x.
13. White H. Economic prediction using neural networks: The case of IBM daily stock returns. *Proceedings of the IEEE International Conference on Neural Networks*, 1988; 451–458.
14. Haykin S. *Neural Networks: A Comprehensive Foundation*. 2 edn., Prentice-Hall, Inc., 1999.

15. Ludermir TB, de Carvalho A, Braga AP, de Souto MCP. Weightless neural models: A review of current and past works. *Neural Computing Surveys* 1999; **2**:41–61.
16. Aleksander I. Self-adaptive universal logic circuits. *IEE Electronic Letters* 1966; **2**(8):231–232.
17. Freitas FD, De Souza AF, Almeida AR. Autoregressive neural network predictors in the Brazilian stock market. *VII Simpósio Brasileiro de Automação Inteligente (SBAI)/II IEEE Latin American Robotics Symposium (IEEE-LARS)*, São Luis, Brasil, 2005; 1–8.
18. Freitas FD, De Souza AF, Almeida AR. A prediction-based portfolio optimization model. *5th International Symposium On Robotics and Automation—ISRA 2006*, Hidalgo, Mexico, 2006; 520–525.
19. De Souza AF, Freitas FD, de Almeida AGC. High performance prediction of stock returns with VG-RAM weightless neural networks. *High Performance Computational Finance (WHPCF), 2010 IEEE Workshop on*, New Orleans, LA, USA, 2010; 1–8, doi:10.1109/WHPCF.2010.5671832.
20. Aleksander I. *RAM-Based Neural Networks*, chap. From WISARD to MAGNUS: a Family of Weightless Virtual Neural Machines. World Scientific, 1998; 18–30.
21. De Souza AF, Badue C, Pedroni F, Oliveira E, Dias SS, Oliveira H, de Souza SAF. Face recognition with VG-RAM weightless neural networks. *Proceedings of the 18th International Conference on Artificial Neural Networks (ICANN'08)*, 2008; 209–216.
22. Kandel ER, Schwartz JH, Jessell TM. *Principles of Neural Science*. 4 edn., Prentice-Hall International Inc., 2000.
23. Mitchell RJ, Bishop JM, Box SK, Hawker JF. *RAM-Based Neural Networks*, chap. Comparison of Some Methods for Processing Grey Level Data in Weightless Networks. World Scientific, 1998; 61–70.
24. Malkiel BG. The efficient market hypothesis and its critics. *The Journal of Economic Perspectives* 2003; **17**(1):59–82.
25. Elton EJ, Gruber MJ, Brown SJ, Goetzmann WN. *Modern Portfolio Theory and Investment Analysis*. 7 edn., John Wiley & Sons, Inc., 2007.
26. Armstrong JS, Collopy F. Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting* Jun 1992; **8**(1):69–80.
27. Makridakis SG, Wheelwright SC, Hyndman RJ. *Forecasting: Methods and Applications*. 3 edn., John Willey & Sons, 1997.
28. Hellström T. Data snooping in the stock market. *Theory of Stochastic Processes* Oct 1999; **5**(21):33–50.
29. Liu F, Ng GS, Quek C. RLDDE: A novel reinforcement learning-based dimension and delay estimator for neural networks in time series prediction. *Neurocomputing* Mar 2007; **70**(7–9):1331–1341.