Weightless Neural Network for High Frequency Trading

Samara A. Alves
Department of Computer Science
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
samara.alvarez@ufrj.br

Wouter Caarls
Department of Electrical Engineering
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
wouter@caarls.org

Priscila M.V. Lima
Tercio Pacitti Institute
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
priscilamvl@gmail.com

***Abstract*—High frequency trading depends on quick reactions to meaningful information. In order to identify opportunities in intraday negotiation in the stock markets, we propose a weightless neural network autonomous trader agent composed by forecasting and decision modules. The forecasting module uses ridge regression, which compared favorably against recursive least squares with exponential forgetting. The decision model applies the predicted prices to compute technical indicators based on a set of relative strength indicators evaluated by back-testing, which are then used to train the weightless neural network WiSARD in deciding whether to buy or sell stocks. Experimental results on a real dataset from the Brazilian stock market showed that it is feasible encode the back-testing in WiSARD in order to improve trading rules in a way that is compatible with the reaction time required by online market updates.**

***Index Terms*—WiSARD, high frequency trading, relative strength indicator, ridge regression.**

I. Introduction

A typical algorithm in high frequency trading operates at the millisecond time scale. In this kind of trade, the success of investors is based not only on the quality of the information they use to support decision making but also on how fast they take decisions. The approach commonly used to predict the future prices of a stock and assist profitable decisions in short term trading is *technical analysis*: treating the historical behavior of a stock as a time series [1].

In order to minimize computation time, online algorithms have been proposed to support high frequency trading applications, especially the class of one-pass algorithms [2]. Important examples of this class include exponential smoothing, exponentially weighted variance, and exponentially weighted regression, also called recursive least squares with exponential forgetting [3].

In this context, we propose an autonomous trader agent that is able to negotiate in the high frequency trading scenario. The trader agent identifies trading opportunities in the market based on a set of trading rules and technical indicators that use the future price predictions, and implements a weightless neural network model WiSARD (Wilkie, Stonham and Aleksander's Recognition Device) [4] that evaluates these trading strategies, trained by back-testing. WiSARD is quite well suited to such applications because of its low time complexity. To avoid overfitting problems, we opted to predict the future price of stocks by ridge regression.

To validate our trader agent, we have tested it on PETR4, one of the most negotiated stocks in the Brazilian market, BM&FBovespa. The BM&FBovespa exchange uses the PUMA system to trade stocks. This system is comprised of a financial information exchange protocol, which is composed by specific messages that enable the online electronic communication between market makers and the BM&FBovespa in a standardized way. We compare the forecasting accuracy on PETR4 to recursive least squares with exponential forgetting and evaluated trading strategies by back-testing, using the methodology described in [5].

Although high frequency trading still faces questions about governmental regulations, the major financial markets have established their own. Several papers have addressed the impact of these practices on market quality [6]. Nevertheless, according to the Securities and Exchange Commission (SEC) this practice currently dominates the majority of trading in the U.S. market [7]. In the Brazilian case, BM&FBovespa has allowed this type of trading since 2009.

The remainder of this paper is organized as follows. Section 2 describes work related to our research. Section 3 explains our proposed trader agent in detail. Section 4 describes the experiments, while Section 5 discusses their results. Section 6 summarizes the main contributions and points to directions for future work.

II. Related Work

Several approaches have been proposed in order to forecast financial time series and to provide decision-making support systems. The two most important models in this field are statistical models and soft computing approaches [8]. Statistical models assume that the time series are generated from a linear process [9]. However, the nature of financial time series nature is non-linear, complex, highly noisy and chaotic [1].

Artificial neural networks have been one of the most frequently applied soft computing mechanisms in financial forecasting and trading problems, since they perform very well in uncertain and noisy environments. In this context, the development of high-frequency trading strategies begins with identification of recurrent profitable trading opportunities

present in high frequency data. However, it is hard to quantify the returns of such strategies at different frequencies due to a lack of available data. Most successful traders choose not to publish their strategies, instead using them for their own profit [5].

In the work proposed in [10], the authors implemented a multilayer perceptron to predict positive oscillations in short time periods (5, 10 or 15 minutes) as a trigger of a market making process to be applied in high frequency trading, using intraday technical indicators as input to the neural network. The output indicated whether the market making process should or should not be initiated predicting an uptrend (positive oscillation).

In [11], the authors proposed a trading agent based on a neural network ensemble that predicts if one stock is going to rise or fall instead of predicting its future values. In that method the experiments used two completely different scenarios: the North American stock market with a daily granularity and the Brazilian stock market with a 15 minute granularity.

In [12] designed and evaluated some models of automated agents for stock market intraday trading, which modeled strategies inspired by the elliot wave principle and some technical concepts commonly adopted by stock market analysis in short time periods (1, 5, 10 or 15 minutes), using a dataset from the Brazilian Stock Exchange .

In the work proposed in [13], the authors improved the risk-adjusted trading performance of AI models using three representative milestone models from Neurocomputing literature namely ANN, ANFIS and DENFIS, by combining different risk-adjusted objective functions. The authors demonstrated the effectiveness of using an ANFIS ensemble architecture and integration method with a span of less aggressive high-frequency trading window (5 minutes).

III. Weightless Trader Agent

In this work, we propose a trader agent that is able to autonomously negotiate in the high frequency trading scenario. The trader is composed of a forecasting module and a decision module. The first module predicts the future prices and feeds the decision module. The second module applies the predicted prices to compute technical indicators used to identify trading opportunities in the market based on a set of trading rules. These trading strategies are evaluated by back-testing, which is then used to train the weightless neural network WiSARD in deciding whether to buy or sell stocks, as illustrated in Fig. 1.

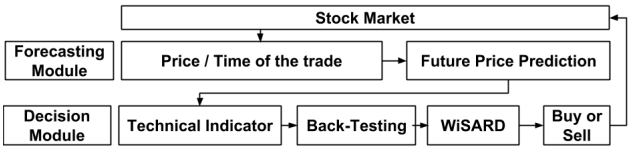

Fig. 1. Weightless Trader Agent.

A. Forecasting Module

In an online scenario, new input data are disclosed sequentially, meaning that the algorithm must be updated whenever new data points are observed. Moreover, in high frequency trading algorithms the agent also needs to react extremely fast to market updates. As such, efficient computation and memory usage are a necessity. In order to overcome the problem of limited fast storage, [2] proposes the use of one-pass algorithms. This type of algorithms receives one data point at a time and updates a set of factors. Only the updated factors are kept in accessible memory, while the actual data point is discarded thus the algorithm uses only a minimal amount of memory and parameters.

The exponentially weighted linear regression model is very useful in high frequency trading strategies [2]. We include an l2-norm regularization term in the loss function, to reduce the variance of the regression coefficients and thus, the forecast error variance. This method is commonly known as ridge regression.

1) Ridge Regression.: The model considers a two-dimensional time series (X_t, Y_t) and conjectures that the variables X and Y are related via a linear relation. These data points are collected in a vector $\boldsymbol{y} = (y_0, y_1, ..., y_t)^T$ and a matrix $\boldsymbol{X}$. In our model we defined $\boldsymbol{y}$ as the prices vector and $\boldsymbol{X}$ as the hypotheses matrix. A major advantage of this approach is the ability to choose the hypotheses that best fit the study scenario. Our matrix is composed by a long moving average (MA_l), a short moving average(MA_s), the trading time, and the exponentially weighted moving average of the prices (EXP).

To calculate the exponentially weighted moving average, let $\widehat{y_i}$ denote the estimated value of the original data point y_i, $i = 1, ..., t$. The exponentially weighted moving average can be computed via the following recursion: $\widehat{y_i} = \alpha\ y_{i-1} + (1 - \alpha)\widehat{y}_{i-1}$.

Supposing $\boldsymbol{y}$ is linearly dependent on $\boldsymbol{X}$, their relationship can be written in matrix notation as $y = \beta X + \varepsilon$. Where $\boldsymbol{\varepsilon}$ is a vector of stochastic noise terms with zero mean. The approach to estimating the parameter vector $\boldsymbol{\beta}$ uses ordinary least squares with a penalty factor λ, where $\lambda \geq 0$, on the size of coefficients. Thus, the ridge coefficients minimize a penalized residual sum of squares

$$\min_{\beta} \sum_{j=0}^{t} (y_j - \beta_0 - \beta_1 x_j - ... - \beta_k x_j)^2 + \lambda \sum_{j=0}^{k} \beta_j^2 \qquad (1)$$

To validate our forecasting module, we compared the forecasting accuracy with the methods proposed in [2]: recursive least square with exponential forgetting and exponential smoothing. In both algorithms a single parameter α, $0 < \alpha \leq 1$, controls the rate at which old information is forgotten.

2) Recursive Least Squares with Exponential Forgetting.: This approach considers a recursive method that updates $\boldsymbol{\beta}$ sequentially and minimizes

$$\min_{\beta} \sum_{j=0}^{t} \alpha^{(t-j)}(y_j - \beta_0 - \beta_1 x_j - ... - \beta_k x_j)^2 \qquad (2)$$

In this algorithm a matrix $\mathbf{M}$ and a vector $\mathbf{v}$, at each step t, are updated with a new data point as: $\mathbf{M_t} = \alpha \mathbf{M_{t-1}} + \mathbf{X_t^T X_t}$ and $\mathbf{v_t} = \alpha \mathbf{v_{t-1}} + \mathbf{X_t^T y_t}$. After each time t the best estimate is $\boldsymbol{\beta} = \mathbf{M_t^{-1}} \mathbf{v_t}$. The advantage of recursive least squares is that it is computationally more efficient than regular least squares. Ridge regression as described above can also be made recursive, by adding the regularization factor from (1) into (2).

B. Decision Module

In high frequency trading algorithms the agent needs to react extremely fast to market updates to be ahead of other market participants, making efficient computation and memory usage a necessity. To build the decision-making module of the agent, we used the weightless neural network model WiSARD, because two of the main advantages of this model are its fast training and classification times. The goal of WiSARD is to learn if the trading strategy based on the predicted relative strength indicator (RSI), moving average convergence/divergence (MACD), rate of change (ROC), commodity channel index (CCI) and moving average values results in profitable or unprofitable buys and sells in order to decide whether or not to change the strategy.

1) RSI: RSI is a momentum oscillator that provides signals to buy when the price is oversold, described as a period of time where there has been a significant and consistent downward move in price, and to sell when it is overbought, described as a period of time where there has been a significant and consistent upward move in price. For this, let t_1 and t_2 be subsequent trading periods and Pt_i the last price in trading period i, $i = 1, 2$. An upward change (U) is characterized by $Pt_2 - Pt_1 > 0$, on the other hand a downward change (D) is characterized by $Pt_2 - Pt_1 < 0$. The calculation of the RSI is described as follows: $\mathrm{RSI} = 100 - \frac{100}{1+\mathrm{RS}}$. Here, $\mathrm{RS} = \frac{\mathrm{EMA}(U)}{\mathrm{EMA}(D)}$, where $\mathrm{EMA}(U)$ is the exponential moving average of the upward changes, and $\mathrm{EMA}(D)$ is the exponential moving average of the downward changes.

2) ROC: The ROC indicator shows trends by remaining positive while an uptrend is sustained, or negative while a downtrend is sustained. Its calculation is: $\mathrm{ROC} = \frac{P_{t2} - P_{t1}}{P_{t1}}$

3) MACD: The MACD indicator is a trend-following momentum indicator composed by three time series: the MACD series, the signal series and the difference between them. The MACD series is the difference between a short and a longer period exponential moving average of historical price. The signal series is an exponential moving average of the MACD series itself. So, $\mathrm{MACD} = \mathrm{EMA_L} - \mathrm{EMA_S}$. Here, $\mathrm{EMA_L}$ and $\mathrm{EMA_S}$ are the exponential moving average of a long and a short period of the historical price.

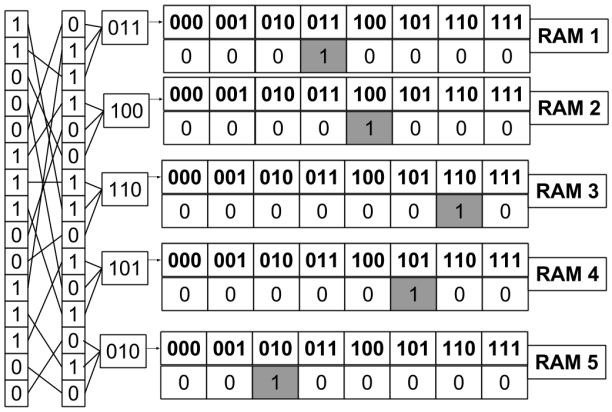

Fig. 2. WiSARD discriminator during the training phase.

4) CCI: The CCI indicator identifies cyclical trends. The calculation is described as follows: $\mathrm{CCI} = \frac{TP - SMA}{0.015 * \sigma}$. Here, SMA is the simple moving average, and σ is the mean absolute deviation and $TP = \frac{H+L+C}{3}$, where H, L and C are the maximum value, the minimal value and the last price in the period t_1, respectively.

5) WiSARD: The model is a weightless neural network used for pattern classification which has been successfully applied to several areas [14], [15], [16], [17]. The model is composed of units called discriminators, each of them corresponding to one output class of the problem and composed by a set of RAM units. The RAMs have a fixed number of address bits randomly connected to the binary input pattern. Nonbinary input must first be converted to a binary representation.

Fig. 2 shows a discriminator during the training phase, with a set of 5 RAMs with a 3 bit address. The input pattern, at this phase, is presented only to the discriminator of its corresponding class, as depicted in Fig. 2. The position addressed by the input bits of each RAM unit receives the value 1. The RAM positions are initialized the value 0.

Fig. 3 details the same discriminator of Fig. 2 during the classification phase. At this stage, a new pattern is presented to all discriminators and the value stored during the training phase in each RAM unit is returned. The output of the discriminator is the sum of the contents of the positions accessed in each of its RAMs. In the example shown in Fig. 3 the output of this discriminator is 3. The class of the discriminator with the highest output is selected as the output of the model.

Therefore, the main difference when compared to traditional approaches lies in the way knowledge is represented internally. Unlike feedforward neural network models, which uses weight matrices, WiSARD uses a set of RAM units connected to the (binary) input pattern. One can see that, because of the simple operations involved in both training and classification phases, WiSARD's complexity is very low and particularly suitable for online algorithms.

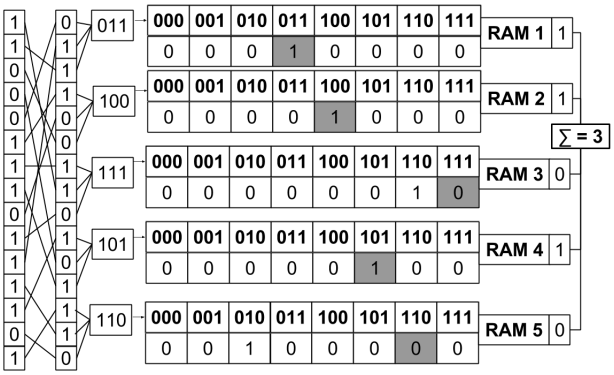

Fig. 3. Discriminator Output.

IV. Experiments

A. Data Set Description

The dataset was collected by GetHFData [18] and consists of tick-by-tick transaction data of the preferred Petrobras share PETR4, in the equity market of Bovespa, from 2015-09-30 until 2015-10-31. In [5] it is asserted that one month is enough to experiment in high frequency trading because the small granularity already results in a very large dataset.

Tick data are irregularly spaced at time intervals [5] which means that at each timestamp the database records the price traded. Therefore, the proposed system uses the granularity of 1, 5, 10 and 15 minute time intervals. The goal is to evaluate the influence of temporal granularity in predicting fluctuations in the price of an asset.

Furthermore, in order to adopt a suitable structure for this scenario, we used a sliding window for training and testing. So, the training starts at the first window – which represents how many values the agent has to look in the past to reasonably predict the future – and slides by moving this window one new data point each time. As such, the training is updated whenever a new transaction occurs.

B. Forecasting Evaluation

In order to be able to forecast future price movements using past data, most linear models require that the distributional properties of the data remain approximately constant through time, or are stationary [5]. However, financial data are frequently non-stationary. Therefore the financial literature often analyzes financial data defined as follows, where P_f is the first price of a time interval and P_l is last price of the time interval:

$$R_t = \frac{P_l}{P_f} - 1 \tag{3}$$

The market efficiency theory [19] defines that all information of the stock market is incorporated instantaneously into its price, such that prices in the market follow a random walk [19]. For this purpose, the market movement can be defined as a stochastic price process. Then, P_t is a *martingale* if the best forecast of P_{t+1} based on current information is P_t itself. Hence, we decided to test for an efficient market by defining an approach which sets out the next stock price as the same as the last price observed and comparing its forecasting error with ours.

To evaluate the prediction of the prices we compared the results of the ridge regression and recursive least square with exponential forgetting. All algorithms were implemented in Python 2.7 and used `scikit-learn`[1] to support the ridge regression algorithm and `padasip`[2] to implement the recursive least square algorithm. We evaluated the generalization performance of the forecaster by the Root Mean Square Error (RMSE).

C. Decision Evaluation

The decision-making module of the agent used a static rule based on RSI that advises the trading system to buy stocks if this indicator is lower than 30 and to sell if it is higher than 70. The weightless neural network model WiSARD, implemented in PyWANN [20], then learned if the decision is an unprofitable or profitable one in order to accept or change this decision. The network was fed with the inputs of the technical indicators RSI, MACD, ROC, CCI and moving averages converted into a binary concatenated representation that recorded the indicators' values and its outputs were the predicted trade profitability: *buy loss*, *sell loss*, *buy profit* or *sell profit*. If the network predicts an unprofitable transaction, the original decision is reversed.

The retina therefore consists of five parts. The first part is a 3-bit vector where the first bit equals 1 if $RSI \leq 70$; the second bit equals 1 if $30 < RSI < 70$; and the third bit equals 1 if $RSI \geq 70$. The second part is a 3-bit vector where the first bit of each vector equals 1 if $P > SMA_{short}$; the second equals 1 if $P < SMA_{long}$; and the third equals 1 if $SMA_{short} < SMA_{long}$. Here, P is the stock price, SMA_{short} and SMA_{long} are the simple moving average in a short and a long period. The third part is a 1-bit vector that equals 1 if Signal $> MACD$. The fourth part is a 3-bit vector where the first bit is 1 if $ROC \leq 0.02$; the second bit is 1 if $-0.02 < ROC < 0.02$; and the third bit equals 1 if $ROC \geq -0.02$. The fifth part is a 3-bit vector where the first bit equals 1 if $CCI \leq 25$; the second bit equals 1 if $15 < CCI < 25$; and the third bit equals 1 if $CCI \geq 15$. The total length of the structure is 13 bits. In total, the model is composed of 4 discriminators (*buy loss*, *sell loss*, *buy profit* or *sell profit*) represented in Fig 2 by a set of 7 RAMs with 2 bit addresses.

The network output is an evaluation of the trading strategies based on a methodology presented in [5]. This approach consists in dividing all trading opportunities into profitable and unprofitable buys and sells. The profitable opportunities are determined based on *stop-gain* and *stop-loss* parameters decided before training. The *stop-gain* and *stop-loss* strategies are defined as an advance order to sell an stock when it reaches

[1] http:///scikit-learn.org/stable/modules/linear-model.html
[2] http://matousc89.github.io/padasip/sources/filters/rls.htm

a particular price limit in order to limit the losses. In our trader, we defined the *stop-gain* as the point where the price reached the last price traded plus z times the standard deviation of the historical prices. On the other hand, we defined the *stop-loss* as the point where the price reached the last price traded minus z times the standard deviation of the historical prices. *Stop-gain* and *stop-loss* are parameters defined to assist the system in closing the position. The situation where the price reached the *stop-gain* before the *stop-loss* was defined as a profitable trading opportunity. This way, the outputs were if the trading decisions is a profitable buy, a profitable sell, an unprofitable buy or an unprofitable sell.

We evaluated if WiSARD is able to identify if the decision rules is an unprofitable or a profitable decision. Thereby, we used the firsts data to train the network and the last ones, in sequence, to test. Moreover, as the RAM units are randomly connected to the input pattern, we performed 100 runs in order to calculate the accuracy standard deviation. After that, we verified whether the processing times of the WiSARD, during training and recognition, were compatible with the time of trading in this high frequency trading scenario.

V. Results and Analysis

A. Data Set Evaluation

In order to evaluate whether our price series, as most financial data, was non-stationary we applied the Augmented Dickey Fuller test [21], which tested the null hypothesis of a unit root was present in a time series sample. We failed to reject the null in all these granularity time windows since all *p-values* were greater then 20%. However, after preprocessing our data as described in (3), the *p-value*, in all these cases, were less than 0.00001. This meant that we could reject the null hypothesis and concluded that our return series is stationary. Consequently its statistical properties do not change over time which guarantees the convergence of the predictor.

B. Forecasting Module Evaluation

The optimal parameters were estimated by comparing the RMSE in a grid search algorithm. In this experiment, using the raw dataset (tick by tick), the best exponential forgetting factor was 0.001 and 0.099, chosen by a line search between 0.0001 and 0.1 with fixed step size 0.01, for the least squares with exponential forgetting and the ridge regression, respectively. For the short moving average and the long moving average, the best parameters were 2 and 5, respectively, from a range between 2 and 15 with step size 1.

In order to select the best regularization factor, λ, one must tune this parameter in such a way that a balance between model fit and model complexity is maintained, since if λ is very large, the regularization effect dominates the squared loss function and the coefficients tend to zero. Whereas, if λ is very small the solution tends towards ordinary least squares, where coefficients exhibit big oscillations. In this sense, we choose to use the Aikaikes information criterion (AIC) [22], which measures the balance between model fit and model complexity. To support this decision, we used the *RidgeCV* implementation by `scikit-learn`, which built an efficient form of leave-one-out cross-validation. The optimal lambda estimation using the AIC criterion was $1e-05$.

TABLE I
Covariance Matrix of Ridge Coefficient

coefficients	log(Time)	EXP	MMA_s	MMA_l
log(Time)	1	0.3572	0.3596	0.3681
EXP	0.3572	1	0.9979	0.9892
MMA_s	0.3596	0.9979	1	0.9940
MMA_l	0.3681	0.9892	0.9940	1

TABLE II
Ridge Coefficient (1 minute time window granularity)

Model	RMSE	MAE	AIC
log(Time)	0.0018613	0.0013511	-12542.3
EXP	0.0018613	0.0013511	-12542.4
MMA_s	0.0018611	0.0013511	-12543.1
MMA_l	**0.0018611**	**0.0013510**	**-12543.1**
MMA_l+log(Time)	0.0021743	0.0015379	-11706.9
$MMA_l + EXP$	0.0021656	0.0015307	-11728.4
MMA_l+log(Time) $+EXP$	0.0021595	0.0015364	-11741.6

The matrix of the ridge regression model is composed of the following hypotheses (coefficients): $\log(Time)$, EXP (which represents exponential smoothing of prices), MMA_s and MMA_l. We observe from Table I, that the independent variables EXP, MMA_s and MMA_l are strongly linearly correlated. The existence of multicollinearity may induce inaccurate estimates of the regression coefficients. Thus, in order to select an appropriate model we compared 7 different models combining these variables using a 1 minute time window granularity. As one can see from Table II, an appropriate model for our dataset in this particular experiment scenario is the Model MMA_l, which presents the smallest RMSE, MAE and AIC values.

The last experiment for the forecasting module is presented in Table III. In this we evaluated the regression approaches using the RMSE, which measures the predictor accuracy, for the model selected in Table II. In terms of generalization performance, ridge regression (Ridge) presents better results for all time window *granularities*, compared to the recursive least squares with exponential forgetting (RLS) approach proposed in [2]. Additionally, we used the *Welch's t-test* to compare the Mean Absolute Error (MAE) of Ridge to RLS. As the p-values in all these cases are less than 0.0001, we have enough evidence to reject the hypotheses that Ridge has statistically similar MAE as RLS.

TABLE III
Comparison of the Forecasters using RMSE.

Granularity	1 min	5 min	10 min	15 min
RMSE *Ridge*	0.00160	0.00298	0.00403	0.00467
RMSE RLS	0.00223	0.00403	0.00541	0.00607
MAE *Ridge*	0.00117	0.00216	0.00303	0.00349
MAE RLS	0.00159	0.00290	0.00401	0.00463
Welch test	-13.17	-5.66	-4.15	-3.44
Welch test *P-value*	3.93e-39	1.78e-08	3.75e-05	6.25e-04

TABLE IV
TRADES SUMMARY WITH RIDGE REGRESSION

Experiment	A	B	A	B	A	B	A	B
Granularity (min)	1	1	5	5	10	10	15	15
ACC (%)	-	55	-	59	-	60	-	68
Profitable Buy	523	611	84	110	40	47	12	30
Unprofitable Buy	654	405	111	90	43	37	34	31
Profitable Sell	708	957	191	212	112	118	82	85
Unprofitable Sell	787	699	142	116	69	62	42	24
Win (%)	46	59	52	61	58	63	55	68
Loss (%)	54	41	48	39	42	38	45	32
Win Loss Ratio	0.85	1.42	1.09	1.56	1.35	1.66	1.23	2.08
Expectancy	-0.08	0.17	0.04	0.22	0.15	0.25	0.10	0.35

C. Decision Evaluation

In order to evaluate the profitability of the trading system we considered trades with a a fixed volume of shares to calculate trading statistics, such as: win ratio, win loss ratio and expectancy. The win ratio refers to the percentage of winning trades among all trades taken, and is an important aspect of risk management. Based on this concept the trader will be considered profitable if it can hit a win ratio higher than 50%, as the evaluation method used in [11]. The win loss ratio is a comparison between the number of wins and the number of the losses. Considering this the trader will be profitable if this ratio is greater than 1, which indicates that, on average, it wins more often than it loses. The third statistic is the expectancy, which is calculated by taking the product of average winning trade size and the win ratio then subtracting the product of the average losing trade size and the loss ratio. Therefore trading strategy is considered profitable if it has an expectancy greater than zero.

In the first experiment in the decision module, based on the RSI trading rules, we simulated a forecaster with 100% accuracy, which we call the Oracle. This Oracle was built by using the real traded prices of the dataset instead of using the prices provided by the forecasters. These results used parameter 0.9 for the RSI exponential forgetting factor. The win ratio of the proposed strategy was superior for all *granularities* when using price predictions at time $t + 1$ and with z equal to 1. This meant that the agent performed best for short-term predictions, and when the *stop-gain* and *stop-loss* formed a small variation band to assist the system to close the position.

In the second experiment we evaluated the agent in two parts: experiments A and B. In both of them, we used the best parameters found in the Oracle experiment. In the experiment A the agent decision was based on the RSI rules in order to decide whether to buy or sell stocks. In experiment B, the WiSARD was used to predict profitability and alter the RSI decision if necessary. After each prediction it was trained on-line for that same input, using a back-testing oracle to generate the label 1. Table IV lists the number of profitable and unprofitable buys and sells, its win percentage, the win loss ratio, the expectancy and the WiSARD accuracy. As one can see, the trader improved its win ratio and expectancy from experiment A to B. Moreover, Table IV shows that the win loss ratio is greater than one for all *granularities* in experiment B. This implies that the agent performance was improved by training the network using actual profitability from data instead of fixed trading rules, since the WiSARD was able to identify some of the failed RSI trading strategy signals. Besides that the training and classification mean times at each point in the dataset were 0.08 and 0.20 milliseconds, respectively, considering only the time needed for decision making.

We executed the classification phase 100 times and the WiSARD showed accuracy greater than 55% in the experiment B, with a zero standard deviation. We can observe that the level of granularity affects the overall trades in the strategies. For example, for the 1 minute granularity, the occurrences of identified opportunities by the strategies are checked in increments of 1 minute; for the granularity of 5 minutes, the occurrences are checked every 5 minutes. This directly influences the number of opportunities identified by each strategy.

```
for i ← 1 to n do
    if price_i ≥ stop gain :
        if RSI decision rule == sell :
            real back testing label = sell loss ;
        else:
            real back testing label = buy profit ;
        break
    if price_i ≤ stop loss :
        if RSI decision rule == sell :
            real back testing label = sell profit ;
        else:
            real back testing label = buy loss ;
        break
end
if WiSARD result = buy loss or = sell profit :
    if real back testing label = buy loss or = sell profit :
        final result = sell profit
    else:
        final result = sell loss
else:
    if real back testing label = buy profit or = sell loss :
        final result = buy profit
    else:
        final result = buy loss
train WiSARD with real back testing label
```

Algorithm 1: Experiment B

VI. CONCLUSION

In this work, we proposed a trader agent, composed of two modules, that was able to negotiate in the high frequency trading scenario. The first module predicted the future prices and fed the decision module. We compared ridge regression, the strategy adopted in the first module, with that proposed in [2] and it achieved better generalization performance. The second module applied the predicted prices to compute technical indicators used to identify trading opportunities in the market based on a set of trading rules. These trading strategies were evaluated by back-testing, which were then used to train the weightless neural network WiSARD in deciding whether to buy or sell stocks. The results applied on a real dataset from the Brazilian stock market showed meaningful performance of the neural network in order to improve trading rules in a

way that is compatible to the reaction time required by online market updates.

Few published works have addressed the problem of trading in high frequency scenario using technical indicators. A major part of the studies focused on trading strategies that have tended to trade in bid-ask spread analysis. Although, the most important limitation of that field of research lies in data availability, many successful traders choose not to publishes their strategies, using them for their own profit [5]. As a next step, we will seek to find the best WiSARD architecture for each time interval, since the same architecture was employed at the four time intervals leading to different performance for each time interval. Another possible topic of future investigation could be the relation of the WiSARD, adjusted by backtesting, to a more sophisticated regression function representing the behavior of the time series, as well as using only causal labels during on-line training.

REFERENCES

[1] J. J. Murphy, *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[2] J. Loveless, S. Stoikov, and R. Waeber, "Online algorithms in high-frequency trading," *Communications of the ACM*, vol. 56, no. 10, pp. 50–56, 2013.

[3] M. E. Salgado, G. C. Goodwin, and R. H. Middleton, "Modified least squares algorithm incorporating exponential resetting and forgetting," *International Journal of Control*, vol. 47, no. 2, pp. 477–491, 1988.

[4] I. Aleksander, W. Thomas, and P. Bowden, "WisardŁ a radical step forward in image recognition," *Sensor review*, vol. 4, no. 3, pp. 120–124, 1984.

[5] I. Aldridge, *High-frequency trading: a practical guide to algorithmic strategies and trading systems*. John Wiley & Sons, 2013, vol. 604.

[6] M. OHara, "High frequency market microstructure," *Journal of Financial Economics*, vol. 116, no. 2, pp. 257–270, 2015.

[7] U. Securities, E. Commission *et al.*, "Equity market structure literature review part ii: High frequency trading," *Staff of the Division of Trading and Markets*, 2014.

[8] R. C. Cavalcante and A. L. Oliveira, "An autonomous trader agent for the stock market based on online sequential extreme learning machine ensemble," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 1424–1431.

[9] D. A. Kumar and S. Murugan, "Performance analysis of indian stock market index using neural network time series model," in *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on*. IEEE, 2013, pp. 72–78.

[10] E. Silva, D. Castilho, A. Pereira, and H. Brandao, "A neural network based approach to support the market making strategies in high-frequency trading," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 845–852.

[11] F. Giacomel, R. Galante, and A. Pereira, "An algorithmic trading agent based on a neural network ensemble: a case of study in north american and brazilian stock markets," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE/WIC/ACM International Conference on*, vol. 2. IEEE, 2015, pp. 230–233.

[12] E. Jabbur, E. Silva, D. Castilho, A. Pereira, and H. Brandão, "Design and evaluation of automatic agents for stock market intraday trading," in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 03*. IEEE Computer Society, 2014, pp. 396–403.

[13] V. Vella and W. L. Ng, "Enhancing risk-adjusted performance of stock market intraday trading with neuro-fuzzy systems," *Neurocomputing*, vol. 141, pp. 170–187, 2014.

[14] D. O. Cardoso, D. S. Carvalho, D. S. Alves, D. F. Souza, H. C. Carneiro, C. E. Pedreira, P. M. Lima, and F. M. França, "Financial credit analysis via a clustering weightless neural classifier," *Neurocomputing*, vol. 183, pp. 70–78, 2016.

[15] H. C. Carneiro, F. M. França, and P. M. Lima, "Multilingual part-of-speech tagging with weightless neural networks," *Neural Networks*, vol. 66, pp. 11–21, 2015.

[16] B. P. Grieco, P. M. Lima, M. De Gregorio, and F. M. França, "Producing pattern examples from mental images," *Neurocomputing*, vol. 73, no. 7-9, pp. 1057–1064, 2010.

[17] S. A. Alves, F. Rangel, F. F. Faria, and P. M. Lima, "Análise de séries temporais financeiras utilizando wisard," in *Congresso Brasileiro de Inteligência Computacional-Volume 01*, 2015.

[18] M. Perlin and H. Ramos, "Gethfdata: A r package for downloading and aggregating high frequency trading data from bovespa," *Brazilian Review of Finance*, vol. 14, no. 3, pp. 443–478, 2016.

[19] E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[20] F. Firmino, "PyWANN," https://github.com/firmino/PyWANN, 2017.

[21] Y.-W. Cheung and K. S. Lai, "Lag order and critical values of the augmented dickey–fuller test," *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 277–280, 1995.

[22] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.