Trading Strategy in Foreign Exchange Market Using Reinforcement Learning Hierarchical Neuro-Fuzzy Systems

Marcelo F. Corrêa[1], Marley Vellasco[1], Karla Figueiredo[1], and Pedro Vellasco[2]

[1] Dept. of Electrical Engineering, PUC-Rio, Brazil
[2] Dept. of Structural Engineering, State University of Rio de Janeiro, RJ, Brazil

Abstract. This paper evaluates the performance of the new hybrid neuro-fuzzy model, Reinforcement Learning Hierarchical Neuro-Fuzzy System (RL-HNFP), in a trade decision application. The proposed model was tested with the Euro/Yen negotiated in Foreign Exchange Market. The main objective of the trading system is to optimize the resource allocation, in order to determine the best investment strategy. The performance of the RL-HNFP was compared with different benchmark models. The results showed that the system was able to detect long term strategies, obtaining more profitability with smaller number of trades.

1 Introduction

Financial markets are influenced by several factors, including economical, political and psychological issues. That is why its movements are extremely hard to forecast. During the last decades, research in this field has been growing very fast. These studies can be divided into two main areas: fundamental and technical analysis. The first is based on factors such as: company's financial statements; management and competitive advantages; competitors; and markets. On the other hand, technical analysis tries to predict the prices´ future direction based on the behavior of past market data [1].

Since the early 90s, neural networks models have been successfully applied to financial time series forecasting [2-7]. The most popular model is the Multi-Layer Perceptron (MLP) with backpropagation algorithm [8]. However, many researchers have suggested the use of reinforcement learning models to create trading strategies and to optimize asset allocation [9-12]. This type of learning works through an input-output mapping, built from the continuous environment interaction, trying to minimize a performance index [8].

One of the main advantages of RL in this kind of application is the fewer number of position changes or trades in the strategies. Generally, it results in less costs and higher profitability. For example, Neurier [9] states that a strategy to invest in the German Stock Index DAX based on RL attained better results than a MLP based model doing just 33% of the trades. The RL strategy kept the investment out of the market when there was not significant trend to follow.

Therefore, this paper evaluates the performance of the new Reinforcement Learning Hierarchical Neuro-Fuzzy System (RL-HNFP) [13] as an intelligent agent

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 461–468, 2009.

that learns the best trading strategy. The intelligent trading system was tested with the Euro/Yen exchange rate between 01/28/1999 and 05/18/2006. The performance of the RL-HNFP system was compared with different benchmark models: MLP neural network, Dynamic Regression, Box-Jenkins model, and the standard buy-and-hold strategy.

This paper is divided into four additional sections. Section 2 summarizes the RL-HNFP system, section 3 presents the proposed intelligent trading system and section 4 discusses the results obtained. Finally, the conclusions that have been achieved from this study are presented in last section.

2 RL-HNFP System

The RL-HNFP model is composed of various standard cells called RL-neuro-fuzzy BSP (RL-NFP). These cells are arranged in a tree hierarchical structure. The cells outputs in the lower levels are the consequents of higher levels cells.

An RL-NFP cell is a mini-neuro-fuzzy system that performs politree partitioning (2^n partitions, where n = number of input variables) of a given space, using the complementary membership functions described in equation (1) in each input dimension, where ρ(x) and μ(x) represent the low and high membership functions, respectively.

$$\mu(x) = \frac{1}{1+e^{-(a(x-b))}} \quad and \quad \rho(x) = 1-\mu(x) \tag{1}$$

where b is the sigmoid inflexion point and α is the sigmoid inclination at $x=b$.

The RL-NFP cell generates a precise output after defuzzification [14-15]. Each cell receives all inputs that are being considered in the problem. These inputs are inferred in the antecedents' fuzzy sets (ρ (x) and μ (x)). Figure 1(a) depicts a cell with two inputs – x_1 and x_2 -, where each partitioning is generated by the combination of the two membership functions, ρ (*low*) and μ (*high*) of each input variable, and is associated with a set of actions (a_1, a_2 ... a_t).

The consequents of the cell's partitions may be a singleton or the previous level stage output. Although the singleton consequent is simple, it is not previously known, since each consequent is associated with an action that has not been defined a priori.

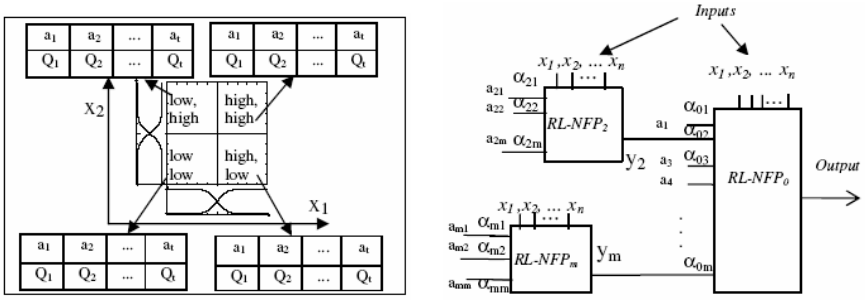

Fig. 1. (a) Internal representation of the RL-NFP cell with two inputs; (b) Example of architecture of the RL-HNFP model

Each partition has a set of possible actions, as shown in Figure 1(a), where each action is associated with a Q-value function. The Q-value is defined as being the sum of the expected values of the rewards obtained by the execution of action *a* in state *s*, in accordance with a policy π [16].

By means of the RL-based learning algorithm, one action of each poli-partition (for example, a_i, and a_q) is selected as the one that represents the desired behavior of the system whenever it is in a given state. Thus, the consequents are the actions that the agent must learn along the process.

The linguistic interpretation of the mapping implemented by the RL-NFP cell is given by fuzzy rules. Each rule corresponds to one partition generated by the politree partitioning and has the following structure:

$$\text{If } x_1 \text{ is low and } \ldots. \; x_n \text{ is high then } y = a_i.$$

RL-HNFP models can be created based on the interconnection of basic cells described above. The cells form a hierarchical structure that results in the rules that compose the agent's reasoning. Figure 1(b) exemplifies this architecture.

Neuro-fuzzy learning process is generally divided into two parts: structure identification and parameter adjustment [17]. RL-HNFP performs these learning tasks in a single algorithm. Basically, each partition chooses an action from its set of actions; the resultant action is calculated by the defuzzification process and represents the action that will be executed by the agent's output. After the resultant action is carried out, the environment is read once again. This reading enables calculation of the environment reinforcement value that will be used to evaluate the action taken by the agent. The reinforcement is calculated for each partition of all active cells, by means of its participation in the resulting action. Thus, the environment reinforcement calculated by the evaluation function is backpropagated from the root-cell to the leaf-cells. Next, the Q-values associated to the actions that have contributed to the resulting action are updated, based on the SARSA algorithm [16]. More details can be found in [15].

3 The RL-HNFP Trading System

The RL-HNFP Trading System was built using the Euro/Yen exchange rate data, from 01/28/1999 to 05/18/2006. The series contained 2130 registers and the following variables were used: closing price in time t; difference between the 3 and 20-day moving averages; difference between the 5 and 20-day moving averages; 6-day momentum (difference between the price in t and t-6); 3, 5 and 20-day moving averages; open price; maximum and minimum prices; closing prices in t-1 and t-2; and the negotiated volume in t.

The least-square estimator (LSE) [18] was used to evaluate the relevance of each input. After that, just 8 variables remained at the model and were used to define the state in t. In order of relevance, the variables considered were: closing price in t, difference between the 3 and 20-day moving averages, difference between the 5 and 20-day moving averages, 6-day momentum, 5 and 3-day moving averages, closing prices in t-1 and maximum price in t.

The RL-HNFP objective is to learn the best action to be taken at each state t, after receiving, as environment reward, the signal of the price variation. To avoid sending

undetermined signals and confusing the learning process, variations smaller than |0,3%| were changed to zero during the learning phase.

Trading strategies were composed by a sequence of three possible actions (*buy*, *sell* and *hold*). The *hold* signals were generated when the output belongs to a certain threshold interval. For example, if the threshold is defined between -0.02 and +0.02, and during 5 periods of time the RL-HNFP's output is 0.3, -0.2, 0.01, 0.25 and -0.01, then the system actions would be "buy", "sell", "hold sell position", "buy" and "hold buy position".

4 Results

The test set was composed of 213 patterns, corresponding to the interval 09/13/2005 to 05/18/2006. The results obtained can be observed in Table 1. The only cost considered was the spread cost estimated in 0.04% (difference between the bid and ask prices). As expected, by increasing the threshold, a higher number of hold signals are generated. Consequently, fewer trades are performed.

Without considering trading costs, the best performance was achieved by the strategy composed just by two actions (buy and sell) and no threshold. After 21 trades, the system reached profits of 7.52%, against 4.47% of a *buy-and-hold* benchmark strategy. However, if trading costs are included, the most profitable result was obtained by model 3. In this strategy, when the output of the RL-HNFP is between -0.02 and 0.02, the hold action is chosen and the previous position is maintained.

For a better understanding of the RL-HNFP Trading System's results, Figure 2 shows exactly when each decision of model 3 was taken. The value of +1 indicates a *buy position* and -1 a *sell position*. During a *buy position*, the investment return rate is the same as the Euro/Yen variation, and it can be positive or negative. In a *sell position*, no return is obtained. As can be observed, the system detected long term trends, without considering so much short term variations. This characteristic made the model more profitable during high long term periods, such as 09/13/2005 to 10/06/2005, 10/13/2005 to 11/18/2005, 11/27/2005 to 12/13/2005, and 01/08/2006 to 02/03/2006. The system indicated that the investor should be in a buy position during these intervals. Also, a graph with the evolution of a hypothetical initial investment of $100 is shown in Figure 3. The graph shows a consistent increase of the value invested during the time.

Table 1. RL-HNFP results: strategies with different actions and thresholds

Model	Actions	Threshold	# Trades	% Profit (without costs)	% Profit (with costs)
Buy and Hold	-	-	-	4,47%	4,47%
1	-1, 1	-	21	7.52%	6.64%
2	-1, 0, 1	±0.01	19	7.43%	6.67%
3	-1, 0, 1	±0.02	17	7.53%	6.86%
4	-1, 0, 1	±0.05	15	4.66%	4.08%
5	-1, 0, 1	±0.10	9	0.68%	0.35%
6	-1, 0, 1	±0.15	5	3.30%	3.14%

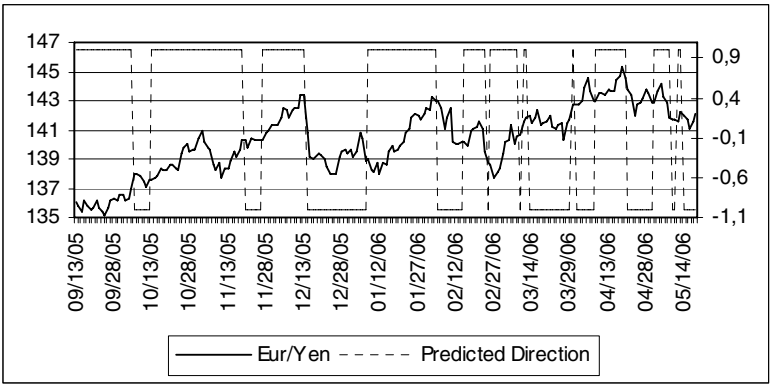

Fig. 2. RL-HNFP System strategy. +1 means a *buy position* and -1 a *sell position.*

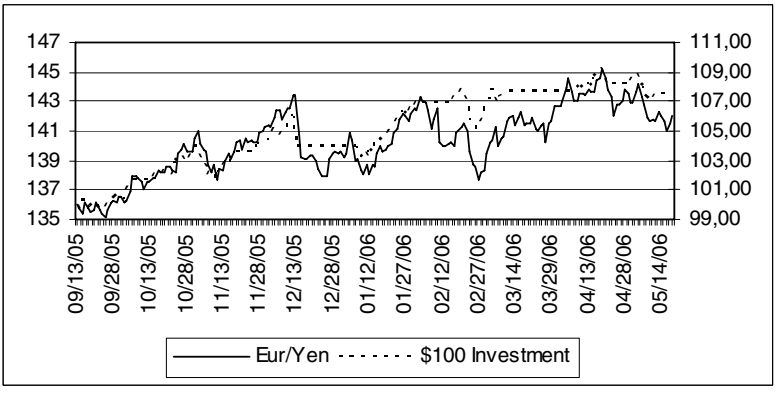

Fig. 3. Hypothetical $100 investment evolution using the RL-HNFP System

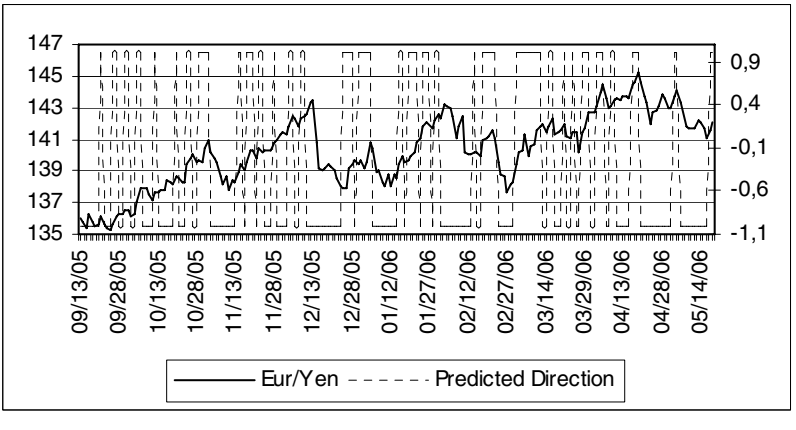

Fig. 4. MLP network strategy. +1 means a *buy position* and -1 a *sell position*.

Table 2. MLP results: strategies with different actions and thresholds

Model	Actions	Threshold	# Trades	% Profit (without costs)	% Profit (with costs)
Buy and Hold	-	-	-	4.47%	4.47%
MLP 1	-1, 1	-	104	7.56%	4.99%
MLP 2	-1, 0, 1	±0.01	98	7.83%	5.42%
MLP 3	-1, 0, 1	±0.02	84	7.80%	5.64%
MLP 4	-1, 0, 1	±0.05	70	4.28%	2.40%
BJ 1	-1, 1	-	102	-26.66%	-30.15%
BJ 2	-1, 0, 1	±0.01	56	-24.02%	-25.98%
BJ 3	-1, 0, 1	±0.02	18	-9.75%	-10.41%
BJ 4	-1, 0, 1	±0.05	2	-0.54%	-0.58%
DR 1	-1, 1	-	71	2.43%	-0.38%
DR 2	-1, 0, 1	±0.01	51	0.39%	-1.62%
DR 3	-1, 0, 1	±0.02	38	1.96%	0.45%
DR 4	-1, 0, 1	±0.05	6	0.86%	0.62%

For comparison purposes, a MLP network was developed. The test set and the input variables were the same for both systems. The MLP architecture was composed of a single hidden layer. The transfer function in all layers was the hyperbolic tangent. Three different topologies were created, with different number of neurons in the hidden layer (7, 9 and 11). The final architecture was selected by a validation set. The learning process was interrupted after 3000 epochs or by the early stopping strategy. Table 2 shows its results. The best performance was obtained by the model with 9 neurons in the hidden layer. So, this network was used for comparisons with the RL-HNFP Trading System.

Strategies based on Box-Jenkins ARIMA (BJ) and Dynamic Regression (DR) predictions were also developed using the same data. Results obtained are presented in Table 2.

The strategies created by BJ and DR were worst than the *buy-and-hold* benchmark. This was already expected, since both are linear methods. Additionally, since BJ is an autoregressive model, other inputs were not considered, which could to improve the forecast.

Analyzing the RL and MLP models, it can be verified that they provide a very distinct behavior. The number of trades of the RL-HNFP Trading System was significantly smaller. This can be explained by the different learning methods used in the MLP and RL-HNFP models. While reinforcement learning works through an input-output mapping, the supervised learning approximates the implicit function between the input variables and the output. The MLP model was also able to anticipate the price movement several times (see Figure 4), but the excessive number of operations made the system less profitable in long term trends. For example, during the intervals of 09/13/2005-11/02/2005 and 11/10/2005-12/13/2005 the gains were lower than in a buy-and-hold strategy. On the other hand, the strategy avoided loss in low trends, as it can be observed during the periods 11/02/2005-11/10/2005, 12/13/2005-12/26/2005, 01/05/2006-01/12/2006, 02/03/2006-02/13/2006 and 02/21/2006-02/27/2006.

However, during undefined trends, for example between 12/18/2005 and 01/18/2006, the MLP could take advantage of small variations. The MLP could also detect faster strong price falls, such as during the period of 12/13/2005-02/05/2006. Figure 4 shows actions taken by the system step by step.

5 Conclusions

In this paper, a Reinforcement Learning Hierarchical Neuro-Fuzzy System [14] was used for an investment decision problem. The main target was to optimize the resource allocation in defining the best Euro/Yen parity trade strategy. A MLP neural network was also developed, to be used as benchmark.

The results showed that the RL-HNFP Trading System could reach a better performance, doing just 20% of the trades done by the MLP model. When considering trading costs, it represents a significant profitability difference.

As future works, tests with other time series can be done to evaluate the RL-HNFP model in investment allocation problems. Also, it could be considered more than one asset to invest, using more actions to be optimized.

References

1. Damodaran, A.: Investment Valuation, 1st edn. John Wiley and Sons, Chichester (1996)
2. Amilon, H.: A Neural Network Versus Black-Scholes: A Comparison of Pricing and Hedging Performances. Journal of Forecasting 22, 317–335 (2003)
3. Azoff, E.M.: Neural Network Time Series Forecasting of Financial Markets. John Wiley & Sons Ltd., Chichester (1994)
4. Cheh, J., Weinberg, R.: An Application of an Artificial Neural Network Investment System to Predict Takeover Targets. Applied Business Research 15, 33–45 (1999)
5. Kutsurelis, J.E.: Forecasting Financial Markets Using Neural Networks: An Analysis of Methods and Accuracy. Master Thesis, Naval Postgraduate School (September 1998)
6. Refenes, A., et al.: Stock Performance Modeling Using Neural Networks: A Comparative Study with Regression Models 7(2), 375–388 (1994)
7. Yao, J., Tan, C.L.: Option Price Forecasting Using Neural Networks. Omega 28, 455–466 (2000)
8. Haykin, S.: Neural Networks — A Comprehensive Foundation. Prentice-Hall, Englewood Cliffs (1999)
9. Neunier, R.: Optimal asset allocation using adaptive dynamic programming. In: Advances in Neural Information Processing Systems, vol. 8. MIT Press, Cambridge (1996)
10. Moody, J., et al.: Performance Functions and Reinforcement Learning for Trading Systems and Portfolios. Journal of Forecasting 17, 441–470 (1998)
11. Lee, J.W.: Stock Price Prediction Using Reinforcement Learning. In: Proc. IEEE Int. Symposium on Industrial Electronics, vol. 1, pp. 690–695 (2001)
12. Lee, O.J., et al.: Adaptive stock trading with dynamic asset allocation using reinforcement learning. Information Sciences 176(15), 2121–2147 (2006)
13. Vellasco, M.M., Pacheco, M., Figueiredo, K., Souza, F.J.: Hierarchical Neuro-Fuzzy Systems - Part II, Encyclopedia of Artificial Intelligence. In: Rabuñal, J.R., Dorado, J., Pazos, A. (eds.) Information Science Referente (2008)

14. Figueiredo, K., Vellasco, M., Pacheco, M.: Hierarchical Neuro-Fuzzy Models based on Reinforcement Learning for Intelligent Agents. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 424–432. Springer, Heidelberg (2005)
15. Figueiredo, K., et al.: Reinf. Learning Hierarchical Neuro-Fuzzy Politree Model for Control of Autonomous Agents. In: 4th Int. Conf. on Hybrid Intelligent Systems (2004)
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
17. Jang, J.S.R.: ANFIS: Adaptive-network-based fuzzy inference system. IEEE Trans. on SMC 23(3), 665–685 (1993)
18. Chung, F.-L., Duan, J.-C.: On Multistage Fuzzy Neural Network Modeling. IEEE Trans. on Fuzzy Systems 8(2), 125–142 (2000)