Rock-paper-scissors WiSARD

Diego F. P. de Souza
Hugo C. C. Carneiro
and Felipe M. G. França
System Engineering and Computer Science Program - COPPE
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
Email: diegosouza@cos.ufrj.br
hcesar@cos.ufrj.br
felipe@cos.ufrj.br

Priscila M. V. Lima
Department of Mathematics - ICE
Universidade Federal Rural do Rio de Janeiro
Seropédica, Brazil
Email: priscilamvl@gmail.com

***Abstract*—This paper presents some strategies used for creating intelligent players of rock-paper-scissors using WiSARD weightless neural networks and results obtained therewith. These strategies included: (*i*) a new approach for encoding of the input data; (*ii*) three new training algorithms that allow the reclassification of the input patterns over time; (*iii*) a method for dealing with incomplete information in the input array; and (*iv*) a bluffing strategy. Experiments show that, in a tournament of intelligent agents, WiSARD-based agents were ranked among the 200 best players, one of them achieving 9th place for about three weeks.**

***Keywords*-Weightless neural networks, game, bots, intelligent agents, adaptiveness**

I. Introduction

A common requirement for the development of intelligent agents is the need to adapt their behavior according to the actions of other agents. As seen in [1], such a behavior is especially important for the gaming industry, in which it is desirable to create players adaptable to the strategy of each opponent, increasing this way the realism and difficulty of the game. In general, strategy games, whether electronic or not, tend to provide a good environment for the study and development of such agents. Rock-paper-scissors [2], [3], in special, constitutes a quite interesting game, because its simplicity would be analogous to situations in which agents do not have a wide range of action possibilities. Furthermore, this game analogy has been successfully applied to represent several communities of subpopulations [4], [5], [6], [7]. In this work, we create adaptive agents who can play this game, predicting its opponent strategy and defeating it.

This work assumes that every move performed in a rock-paper-scissors game is influenced by the recent results in the game and, therefore, can be inferred by an adaptive intelligent agent. As it is easily seen, in cases where at least one of the two players uses randomness as a strategy, the number of wins of both players will be on average equal to one third of the number of rounds played, thereby generating a result very close to a draw. Several machine learning techniques have been adopted to provide agents with this adaptiveness [8], [9], but weightless artificial neural networks (WANNs) had not yet been applied in this context.

Because it is a mechanism that combines a high potential adaptiveness and an extremely simple architecture [10], [11], [12], the WiSARD neural network has been chosen as the basic paradigm in the proposal of rock-paper-scissors players/agents. The network shall receive as input the game history of last H rounds and thus try to predict the next move of its opponent.

The major problem of the application of that model to this context, is that the game strategies of the opponents tend to change over time. Thus it is necessary to readjust the knowledge of the network according to opponent strategy. For this reason, this work uses variations during the network training so as to allow that its classification for a particular input pattern can evolve with time.

The remainder of this paper is organized as follows. After this introduction, the architecture and functioning of the WiSARD neural model are summarized in Section II. Sections III and IV present, respectively, the strategies used to create the agents and experimental results. Section V concludes the text pointing at future steps of this research.

II. The WiSARD Neural Model

Technological advances on the area of integrated circuits that occurred in the '70s, enabled Wilkie, Stonham and Aleksander to create a general-purpose device, composed of small units of RAM (Random Access Memory). This device, called WiSARD [10], [12], was capable of recognizing and classifying different patterns which were presented to it, with a certain degree of generalization.

WiSARD's basic unit of memory, the RAM, consists of N input bits and 2^N memory locations, capable of storing only one bit each (see Fig. 1). Initially, all memory locations are filled with the value 0. During its training, when a binary input pattern of size N is presented to that memory unit, one of its memory locations will be addressed and filled with the value 1. In the end, all locations that are filled with the value 1 represent the patterns learned by the RAM.

However, there is no generalization degree inside a single RAM, i.e., it only recognizes input patterns identical

978-1-4799-3194-1/13 $31.00 © 2013 IEEE
DOI 10.1109/BRICS-CCI-CBIC.2013.38

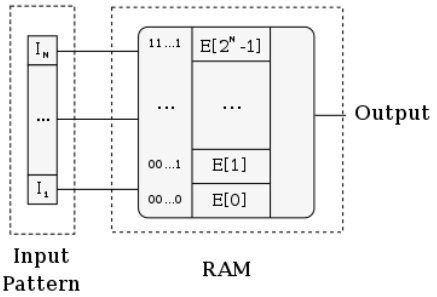

Figure 1. A RAM node with an input pattern of N bits.

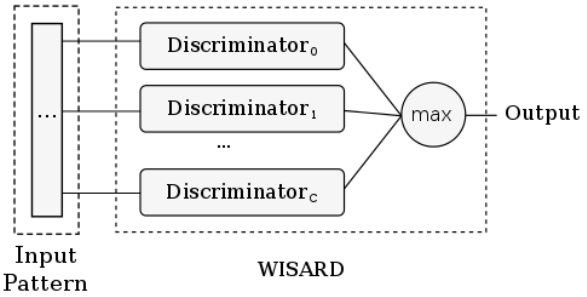

Figure 3. A WiSARD representation with C possible classifications. It's output is the class whose discriminator is most activated.

to those previously presented. In order to achieve generalization levels in the network, each response class must be represented by a discriminator, which is a structure composed of several RAMs and where each of these RAMs randomly connects to the input array of the network, depicted in Fig. 2. Thereupon the network response for this class will be the sum of the outputs of all the RAMs of this discriminator that were addressed by the input. As consequence of that, if the input pattern is slightly different from the pattern presented during the training, only a few RAMs will not be activated, slightly reducing the reliability of the response of the discriminator.

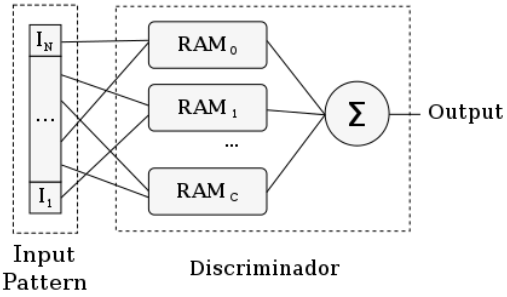

Figure 2. A graphical discriminator representation with K RAMs.

A WiSARD network consists of as many discriminators as classes one wants to classify patterns into. The network final result is the class whose discriminator received the highest reliability response (see Fig. 3).

Although RAM nodes use only binary inputs, these networks can also accept real or integer inputs, if a proper coding technique is employed [13], [14], [15]. Such a technique should preserve the properties of the original input. In any way, the resulting WiSARD network architecture remains extremely simple and its classification and learning operations are basically read/write operations on memory locations. This constitutes one of the main factors that make the model so effective.

III. RPS-WiSARD

RPS-WiSARD heuristics are based on four characterics: (*i*) base-N addressing; (*ii*) several different training strategies; (*iii*) a technique for handling incomplete input; and (*iv*) a bluff technique. These characteristics are explained in Sections III-A to III-D.

A. *Base-*N *addressing*

In general, WiSARD uses binary input patterns. Because of this, converting the entries to the binary format is usually mandatory. In order to bypass this limitation, a $H \times 2$ matrix was adopted as the input retina of the network. Each cell of this matrix can assume three values (0, 1 or 2). Thus, the only change in the network architecture is in the way the RAMs are addressed, i.e., The RAMs now are addressed by a ternary input instead of the regular binary one.

This addressing mechanism can be extended to any value of N, as there are some variations of rock-paper-scissors that use more than the three standard options [16], [17]. However, higher values of N should be used with parsimony, because, as N increases, so does the size of the network RAMs. The main advantage of using an N-ary instead of binary codifications lies on allowing any possible combination to be a valid one and on not existing any similarity between the three possible input value of each trit (a base-3 digit, by analogy with 'bit').

B. *Training strategies*

Rock-paper-scissors is a very adaptive game, i.e., the game strategies used by each player vary over time. Therefore, RPS-WiSARD must be capable of learn new behavior patterns and forget old ones. Three different training strategies were created: training with no-decay, with single-decay and with double-decay.

1) No-decay training: In order to make RPS-WiSARD capable of learning new patterns, it is mandatory to inhibit the network belief on wrong answers. For instance, if one desires to associate in the network a response r to an input H, a value **1** is put in every memory location addressed by H in the discriminator corresponding to r, whereas the memory locations addressed by H belonging to any other discriminator will be set to **0**. The ability of the network to forget old patterns is ensured by writing **0** on memory locations that were previously set to **1**.

2) Single-decay training: Although the training with no-decay strategy has a fast convergence rate and allows the network to adapt itself to new patterns, it is extremely noise-sensitive. In order to mitigate this issue, a more sofisticated training strategy is proposed.

This strategy takes into account the memory content, by either reinforcing or inhibiting it. The main idea consists of being capable of keep an old response even if the last sample of the current game history H is different. In the example of Fig. 4 ($|H| = 1$) it is preferable that the network predict R instead of P.

Such a desired behavior can be emulated if the content of the addressed memory locations are either reinforced or inhibited in a smoothen fashion. Equation 1 is the one used to solve the noise-sensitiveness issue.

$$E' = e + d_1 \cdot E \tag{1}$$

where E' is the new value to be stored in the addressed memory location, e is the value initially intended to be stored (0 or 1), d_1 is the single-decay rate (that varies according to the problem in question) and E is the value previously stored in the memory location.

To avoid that the contents of the RAM locations grow indefenitely, d_1 needs to be a real value that lies in the interval $[0, 1[$. When $d_1 = 0$, this strategy is degenerated to the no-decay one.

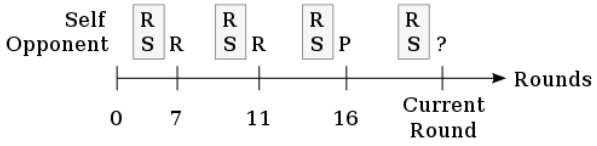

Figure 4. Trace example where an R could be the desirable output

3) Double-decay training: This approach extends the single-decay training by considering the amount of time that a given piece of information is stored in a memory location, thus, reducing its reliability. The main goal of using the double-decay instead of the single one is exemplified in the case depicted in Fig. 5 ($|H| = 1$).

Fig. 5.a shows the case where R is the response of all but the last round, where P is chosen as output. In Fig. 5.b all $\{R, S\}$ input samples are recent, all but the last indicating R as response, and the last indicating P. Thus it is desirable that RPS-WiSARD outputs P as response for the former case, since the first samples are too early and it is very likely that the opponent has already changed its game strategy, whereas it is expected that RPS-WiSARD outputs R for the latter case, as it would normally occur in the single-decay strategy.

In order to emulate this behavior, a time-decaying parameter was added to Equation 1, resulting in Equation 2.

$$E' = e + d_1 \cdot d_2^{\alpha} \cdot E \tag{2}$$

where d_2 is the time-decaying rate, α the number of rounds past since the last training to the memory location addressed in this round and E', E, e and d_1 remain the same.

Two characterics are important to note about the double-decay strategy: *i*) it is mandatory to store the number of the round in which occurred the last update to the current

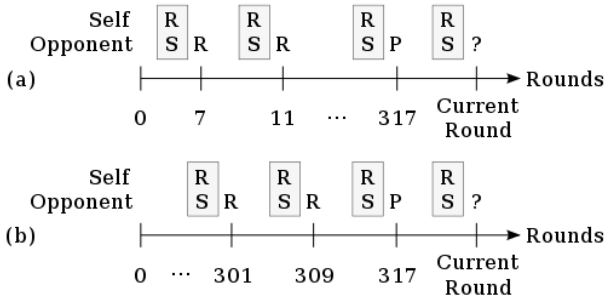

Figure 5. Trace example taking into account: (a) the time passed, where a P could be the desirable answer and (b) another one where R could be more adequate.

memory location and *ii*) d_2 needs to be a real value that lies in the interval $[0, 1]$. When $d_2 = 1$, this strategy is degenerated to the single-decay one.

C. Incomplete input

Although networks that use some kind of decay in their training are more resistant to both noise and bluff, they adapt more slowly than those that use no decay during their training. It is, therefore, disadvantageous for the former to play against the latter. In order to solve this problem, RPS-WiSARD needs to receive as input not only the game history H, but also the result of the last time in which input H appeared. This way, it is possible to identify whether or not the opponent uses quick-learning strategies.

That new piece of information is stored in two other WiSARD networks. They store the last moves of each player given a game history H. No-decay strategy is used during training of these networks. Both of them have 4 discriminators, one for each possible move (R, P or S) and a last one, which represents an "unknown trit" (any of the three possible moves), which is used when the input H has not been presented to the networks yet. Moreover, every RAM location of the last discriminator is initialized by being set to **1**, whereas the RAM locations of all other discriminators are initialized by being set to **0**.

The opponent move is predicted by yet another WiSARD network, whose input consists of array H concatenated to the outputs of the two aforementioned WiSARD networks. Its input may contain "unknown trits" and thus, several memory locations may be accessed in a single read/write operation. This access ideally occurs in a non-unbalanced fashion and thus, mapping the degree of doubt of the network to a function of the number of "unknown trits". A *confidence* measure γ is proposed in order to represent the reliability of accessing one of the several addressed memory locations. Its value is calculated according to Equation 3.

$$\gamma = N^{-b} \tag{3}$$

where γ is the confidence measure, N is the base used by the network addressing mechanism (3 in the rock-paper-scissors case) and b is the amount of "unknown trits".

Table I
FINAL SCORE OF EACH AGENT IN BOTH LOCAL TOURNAMENTS, AND IN RPSCONTEST. VALUES COLLECTED ON APRIL 15TH 2013.

Agents	50 Best		50 Random		RPSContest		
	Win Rate	Rank	Win Rate	Rank	Win Rate	Rating	Rank
wisard1	46.27%	35	68.83%	8	71.73%	7122	191
wisard2	44.54%	38	69.44%	5	79.66%	7983	14
wisard3	44.83%	37	68.84%	7	73.68%	7309	146
wisard4	48.28%	29	71.22%	3	76.12%	7759	50

During the memory writing procedure, all RAM contents must be updated proportionally to the confidence value γ. This can be done by storing in their addressed location the value of E' obtained through Equation 4. In a similar fashion, array T, which indicates the time (round number) when a memory location had its last update, must be updated according to Equation 5.

$$E' = (e + d_1 \cdot d_2^{\alpha} \cdot E) \cdot \gamma + E \cdot (1 - \gamma) \tag{4}$$

$$T' = T_n \cdot \gamma + T \cdot (1 - \gamma) \tag{5}$$

In Equation 4, γ is the confidence value and E', E, e, d_1, d_2 and α represent the same parameters of Equation 2. In Equation 5, T' is new update time to be stored, T_n is the number of the current round and T is the value previously stored in that location.

The memory reading procedure occurs in a very similar manner. Initially the confidence parameter γ is obtained. It is then used to calculate the content stored in the addressed locations of the RAM in a weighted form, according to Equation 6. The response of the RAM is the sum of all r_i's, thereby keeping the weight of the contribution of the RAM for calculating the reliability of the discriminator.

$$r_i = E \cdot \gamma \cdot d_2^{\alpha} \tag{6}$$

D. Bluff

In rock-paper-scissors games, it is important to adopt strategies to lure the opponent, thus hindering one's adaptation strategies. This way, bluffing strategy adopted by RPS-WiSARD varies according to the difficulty of the game set. In this context, bluffing means only that if a given player predicts that his opponent will draw an X, he could either draw a Y (Y wins X) and win the set or draw an X, depending on a probability to be defined.

The bluff occurs randomly, as a function of the results of the last 50 rounds. If the number of correct predictions in this interval is higher than 35, the probability of RPS-WiSARD bluffing is 0%. Otherwise, if the number of correct predictions is higher than 25, that probability increases to 30%. In case the number of correct predictions is lower than 25, the probability of bluffing will be 45%.

IV. RESULTS: EXPERIMENTS AND RPSCONTEST

In order to evaluate the capability of RPS-WiSARD four players/agents (wisard1, wisard2, wisard3 and wisard4) were created and their implementations were sent to RPSContest [18] site, so that these agents could be compared among themselves and also with hundreds of other bots. All four agents use the base-*N* addressing and the bluffing technique. Wisard1, wisard2 and wisard3 training adopts, respectively, no, single and double decays. Wisard4 is capable of dealing with incomplete inputs and is composed by two WiSARD networks trained with single-decay and a third one trained with double-decay. Additionally, two local tournaments were performed.

The first local tournament was composed by the 4 agents and the best 50 bots in RPSContest ranking. The second one was composed by all 4 agents and 50 bots randomly chosen among the 1299 present in RPSContest. In both local tournaments every agent/bot played against each other in matches composed by 1000 sets, each one of them composed by 1000 rounds. By the end of each match the average victory percentage of each network was calculated and considered as being its score. The final score of the tournament was the average of all scores obtained in each match.

Table II
MATCHES AMONG DIFFERENT IMPLEMENTATIONS. CELL (i, j) REPRESENTS THE PERCENTAGE OF WINS OF THE AGENT OF LINE i AGAINST THE ONE OF COLUMN j.

	wisard1	**wisard2**	**wisard3**	**wisard4**
wisard1	–	64.60%	59.70%	3.20%
wisard2	33.90%	–	45.30%	9.60%
wisard3	39.00%	52.80%	–	10.50%
wisard4	97.30%	89.20%	88.50%	–

Table II contains the results obtained by the four agents in the first local tournament. The results in the second local tournament is very similar to those of the first one. In this table it can be noted that wisard1 outperforms both wisard2 and wisard3, as it was expected. However, wisard4 strategy can easily defeat all other agents, especially wisard1, which uses a quick-learning strategy due to the application of no-decay during its training.

The final scores achieved by each wisard in the three tournaments can be seen in Table I. It can be noted that wisard4 was the agent that had the best average

performance in both tournaments, ranking 29th in the first one and 3rd in the second one. The other three agents obtained very similar results in both local tournaments. In RPSContest, the network that presented the best performance result was wisard2, ranking 14th, and the second best was wisard4 appearing in the 50th place. All 4 agents were sent to RPSContest with "Diego Souza" as author.

V. Conclusion

This paper described the implementation of four agents capable of playing the rock-paper-scissors game. In order to improve the quality of the agents, a new technique was developed. It consists of a new approach to the prediction of moves in the rock-paper-scissors game. Apart from a novel data input representation in WiSARD neural networks, three new training strategies were introduced, by stimulating the pattern relearning in the network and a technique to deal with incomplete information in the network input array. RPS-WiSARD proved to be very powerful, as those agents achieved a high performance (above 70% wins) when compared to several other agents.

Similar results could also be applied to a bilateral closed negotiation agent [19], that can perceive its current environment state and change it into a more desirable one. Furthermore, additional efforts into the development of new techniques capable of dinamically updating the training decay rates could possibly result in more efficient players. Finally, these strategies could be used in other problems that involve time-series forecasting.

References

[1] P. Spronck, M. Ponsen, and E. Postma, "Adaptive game AI with dynamic scripting," in *Machine Learning*. Kluwer, 2006, p. 217–248.

[2] D. Walker and G. Walker, *The Official Rock Paper Scissors Strategy Guide*. Touchstone, 2004.

[3] G. Dance and T. Jackson. (2013, May) Rock-Paper-Scissors: You vs. the Computer. [Online]. Available: http://www.nytimes.com/interactive/science/rock-paper-scissors.html

[4] B. Sinervo and C. M. Lively, "The rock-paper-scissors game and the evolution of alternative male strategies," *Nature*, vol. 380, no. 6571, p. 240–243, 1996.

[5] R. Bose, "Effect of swarming on biodiversity in non-symmetric rock-paper-scissor game," *Systems Biology, IET*, vol. 4, no. 3, p. 177–184, 2010.

[6] B. Kerr, M. A. Riley, M. W. Feldman, and B. J. M. Bohannan, "Local dispersal promotes biodiversity in a real-life game of rock-paper-scissors," *Nature*, vol. 418, p. 171–174, 2002.

[7] M. Frean and E. R. Abraham, "Rock-scissors-paper and the survival of the weakest," *Proceedings of the Royal Society B: Biological Sciences*, vol. 268, p. 1–5, 2001.

[8] F. F. Ali, Z. Nakao, and C. Yen-Wei, "Playing the rock-paper-scissors game with a genetic algorithm," in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, 2000, p. 741–745.

[9] F. Salvetti, P. Patelli, and S. Nicolo, "Chaotic time series prediction for the game, rock-paper-scissors," *Appl. Soft Comput.*, vol. 7, no. 4, p. 1188–1196, Aug. 2007.

[10] I. Aleksander and H. Morton, *An introduction to neural computing*. New York, NY, USA: Van Nostrand Reinhold Co., 1990.

[11] C. S. Pattichis, C. N. Schizas, A. Sergiou, and F. Schnorrenberg, "A hybrid neural network electromyographic system: incorporating the wisard net," in *1994 IEEE International Conference on Neural Networks*, vol. 6, 1994, p. 3478–3483.

[12] I. Aleksander, M. D. Gregorio, F. M. G. França, P. M. V. Lima, and H. Morton, "A brief introduction to weightless neural systems." in *ESANN*, 2009, p. 299–305.

[13] F. Gray, "Pulse code communication," Patent US 2 632 058, Mar 15, 1953.

[14] A. Kolcz and N. M. Allinson, "Application of the CMAC input encoding scheme in the n-tuple approximation network," *IEE Proceedings on Computers and Digital Techniques*, vol. 141, no. 3, p. 177–183, 1994.

[15] H. C. C. Carneiro, "A Função do Índice de Síntese das Linguagens na Classificação Gramatical com Redes Neurais Sem Peso (In Portuguese)," Master's thesis, Universidade Federal do Rio de Janeiro, Brazil, 2012.

[16] K. A. Hawick, "Cycles, diversity and competition in rock-paper-scissors-lizard-spock spatial game agent simulations," *Proceedings of the 2011 International Conference on Artificial Intelligence, ICAI 2011*, p. 115–121, 2011.

[17] P. P. Avelino, D. Bazeia, L. Losano, and J. Menezes, "von neummann's and related scaling laws in rock-paper-scissors-type games," *Phys. Rev. E*, vol. 86, p. 031119, Sep 2012.

[18] B. Knoll, D. Lu, and J. Burdge. (2013, May) Rock Paper Scissors Programming Competition. [Online]. Available: http://www.rpscontest.com

[19] T. Baarslag, K. Hindriks, C. Jonker, S. Kraus, and R. Lin, "The first automated negotiating agents competition (ANAC 2010)," in *New Trends in Agent-Based Complex Automated Negotiations*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, S. Fatima, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2012, vol. 383, p. 113–135.