Improving VG-RAM Neural Networks Performance Using Knowledge Correlation

Raphael V. Carneiro[1], Stiven S. Dias[1], Dijalma Fardin Jr.[1], Hallysson Oliveira[1], Artur S. d'Avila Garcez[2], and Alberto F. De Souza[1,*]

[1] Universidade Federal do Espírito Santo, Programa de Pós-Graduação em Informática
Av. Fernando Ferrari, 514, 29075-910, Vitória, ES, Brazil
[2] Department of Computing, City University, London EC1V OHB, UK
albertodesouza@gmail.com

Abstract. In this work, the correlation between input-output patterns stored in the memory of the neurons of Virtual Generalizing RAM (VG-RAM) weightless neural networks, or knowledge correlation, is used to improve the performance of these neural networks. The knowledge correlation, detected using genetic algorithms, is used for changing the distance function employed by VG-RAM neurons in their recall mechanism. In order to evaluate the performance of the method, experiments with several well-known datasets were made. The results showed that VG-RAM networks employing knowledge correlation perform significantly better than standard VG-RAM networks.

1 Introduction

RAM-based neural networks, also known as n-tuple classifiers or weightless neural networks, do not store knowledge in their connections but in Random Access Memories (RAM) inside the network's nodes, or neurons. These neurons operate with binary input values and use RAMs as lookup tables: the synapses of each neuron collect a vector of bits from the network's inputs that is used as the RAM address, and the value stored at this address is the neuron's output. Training can be made in one shot and basically consists of storing the desired output in the address associated with the input vector of the neuron [1].

In spite of their remarkable simplicity, RAM-based neural networks are very effective as pattern recognition tools, offering fast training and easy implementation [2]. However, if the network input is too large, the memory size becomes prohibitive, since it must be equal to 2 to the power of the input size. Virtual Generalizing RAM (VG-RAM) networks are RAM-based neural networks that only require memory capacity to store the data related to the training set [3]. In the neurons of these networks, the memory stores the input-output pairs shown during training, instead of only the output. In the recall phase, the memory of VG-RAM 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 neuron is taken from the pair whose input is nearest to the input presented – the distance function employed by VG-RAM

* Corresponding author.

I. King et al. (Eds.): ICONIP 2006, Part I, LNCS 4232, pp. 427 – 436, 2006.

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.

In this paper, the correlation between input-output patterns stored in the memory of VG-RAM neurons is used to improve the performance of VG-RAM neural networks. The correlation between input-output pairs, or knowledge correlation, is detected analyzing the input-output pairs learned during training using genetic algorithms. This knowledge correlation is used for changing the distance function employed in the associative search used during recall. The new function ignores those elements of the input vector that do not correlate to the expected output of the network. In order to evaluate the performance of knowledge correlated VG-RAM networks, experiments have been performed with several well-known datasets. The experiments have shown that knowledge correlated VG-RAMs outperform standard VG-RAM, achieving accuracy up to 25.23 percentage points higher.

2 VG-RAM Generalization

Fig. 1 shows the lookup table of a VG-RAM neuron with three inputs (X_1, X_2 and X_3). This lookup table contains three entries (input-output pairs), which were stored during the training phase (entry #1, entry #2 and entry #3). During the recall phase, when an input vector (input) is presented to the network, the VG-RAM recall algorithm calculates the distance between this input vector and each input of the input-output pairs stored in the lookup table. In the example of Fig. 1, the Hamming distance from the input to entry #1 is two, because both X_2 and X_3 bits do not match the input vector. The distance to entry #2 is one, because X_1 is the only non-matching bit. The distance to entry #3 is three, as the reader may easily verify. Hence, for this input vector, the algorithm evaluates the neuron's output, Y, as zero, since it is the output value stored in entry #2.

lookup table	X_1	X_2	X_3	Y
entry #1	1	1	0	1
entry #2	0	0	1	0
entry #3	0	1	0	0
input	1	0	1	**0**

Fig. 1. Example of operation of a VG-RAM neuron

The Hamming distance function gives every input bit the same weight; however, this may cause erroneous generalization in some cases. To illustrate this, let's assume that, on the previous example, both X_2 and X_3 input bits are irrelevant to the desired evaluation of the output Y (they are not necessary for modeling what the network is expected to model). In this case, a proper distance function should return zero as the distance between the input vector and entry #1 (distance to #2 remains unchanged), and Y should be one instead of zero. This shows that the knowledge about the relevance of each input bit is essential to good generalization in VG-RAM networks.

To improve the performance of VG-RAM-based classifiers it would be helpful to try and discover which input vector bits are relevant and which are not. This can be achieved analyzing the knowledge acquired by the neural network during the training phase. In this work, we propose the use of genetic algorithms for discovering and exploiting the knowledge correlation of input-output patterns stored in the lookup tables of VG-RAM neurons.

3 Knowledge Correlation in VG-RAMs with Genetic Algorithms

Genetic algorithms (GA) seek global optimization of a given cost function using natural selection principles [4]. This searching process tries to find the values, or genes, of the input parameters of this cost function by generating specific instances of them, or individuals, which best fit to an objective. In other words, the goal is to maximize the individual fitness. From the initial and subsequent generations, crossover and mutation operations are performed on the genes of the individuals and those with the highest fitness perpetuate their genes. The optimization stops after a specified number of generations or when a specific optimization goal is achieved.

To use genetic algorithms to solve a problem, or optimize a cost function, one must describe this function in accordance with the genetic algorithm paradigm. As mentioned above, each input variable of the cost function is defined as a gene. The set of all inputs is a chromosome, and an individual is an instantiated chromosome. A fitness function establishes the individuals' evaluation criteria (the cost). The solution of the problem is the individual that optimizes the cost function, or the fittest individual. In this paper, we define the problem of discovering and taking advantage of the knowledge correlation of input-output patterns stored in the lookup tables of VG-RAM neurons as a genetic algorithm optimization problem.

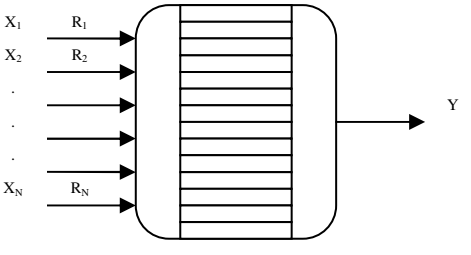

Fig. 2. VG-RAM neuron with an input vector (X_1 ... X_N), an output value (Y) and a relevance vector (R_1 ... R_N) that indicates which input elements are relevant to the determination of the neuron's output

In VG-RAM neurons, during the training phase, sample pairs of input vectors (X_1 ... X_N) and associated output values (Y) are stored into the neurons' lookup table, while, during the recall phase, inputs are compared against each input of the stored input-output pairs to find the nearest pairs, according to the Hamming distance. To compute the Hamming distance, each bit of a given input vector is compared against each bit of the inputs of the stored input-output pairs. The number of bits that differs

corresponds to the Hamming distance. We propose to associate to each comparator a relevance flag, R_i (Fig. 2). Each relevance flag may be set on (value 1) or off (value 0), indicating that the corresponding bit is either relevant or irrelevant to the proper evaluation of the neuron's output. The set of relevance flags defines a relevance vector (R_1 ... R_N), which is the chromosome of the genetic algorithm optimization problem if the neural network is composed of a single VG-RAM neuron (Fig. 2). If the neural network possesses multiple neurons, the chromosome is the collection of all relevance vectors (one per neuron). The solution of the genetic algorithm optimization problem is the individual for which the relevance bits are on only for those inputs bits which are correlated with the desired outputs, for the set of input-output pairs presented during training.

An optimal individual should not cause the network to produce an erroneous output with an input learned during training. Therefore, the fitness function must guarantee that the network produces the same correct results, after optimization, with the inputs learned during training. But it must also favor discarding irrelevant input bits by turning off the corresponding relevance bits, since, the more irrelevant bits are discarded, the more effective is the network generalization. Hence, a good fitness function should minimize the sum of relevant bits and minimize the network errors caused by the relevance bits turned off. A fitness function that meets these criteria is shown in Equation (1).

$$fitness(\underline{R},T) = \left(\frac{1}{N} \sum_{i=1}^{N} R_i + \sum_{k \in T} lookup\_error(k,\underline{R},T) \right)^{-1} \quad (1)$$

In Equation 1, $\underline{R}$ is a relevance vector instance, i.e., an individual whose fitness is to be evaluated; N is the number of genes in the chromosome $\underline{R}$, i.e., the number of relevance flags in the vector; T is the VG-RAM lookup table, i.e., the neuron's knowledge, learned during training; and k is a lookup table entry. The function *lookup_error* has the following pseudocode:

1. Find entry k in T;
2. Change the distance function according to the relevance vector R;
3. Search T, looking for the nearest and equally distant neighbors of k, and, if the output of any of them is different to that of k, returns one, otherwise, zero.

The first term of the fitness function deals with the number of relevant elements, while the second term deals with the number of errors produced by the new generalization engine. Since the objective is to maximize the fitness and minimize both terms, the equation calculates the inverse of the sum of the terms. That is, the genetic algorithm searches an individual with as less relevant elements as possible, providing that the network accuracy does not deteriorate. It is important to note that simply examining which relevance bits can be turned off in sequence without producing network errors may easily result in a suboptimal solution. The use of a genetic algorithm in the search favors (although does not guarantee) finding an optimal solution.

4 Methodology

We used experiments to evaluate knowledge correlated VG-RAM neural networks. To implement these neural networks, we employed the Event Associative Machine

(MAE), an open source framework for modeling VG-RAM neural networks developed at Universidade Federal do Espírito Santo [5]. MAE is similar to the Neural Representation Modeler (NRM) [6], developed by the Neural Systems Engineering Group at Imperial College London and commercialized by Novel Technical Solutions. MAE differs from NRM on three main aspects: it is open source, runs on UNIX (currently, Linux), and uses a textual language to describe weightless neural networks (WNN).

MAE allows designing, training and analyzing the behavior of modular WNNs whose basic modules are bidimensional neural layers and bidimensional filters. Neural layers' neurons may have several attributes (type, input sensitivity, memory size, etc) and the user can freely design filters using the C programming language. The MAE user specifies modular WNNs using the MAE Neural Architecture Description Language (NADL). NADL source files are compiled into MAE applications, which have a built-in graphical user interface and an interpreter of the MAE Control Script Language (CDL). The user can train, recall, and alter neural layers' contents using the graphical interface or scripts in CDL.

Knowledge correlation was added to MAE via: (i) a new neuron type, referred to, in NADL source, as "correlate neuron"; and (ii) a new CDL script command called "correlate network". The correlate network script command triggers the genetic algorithm that performs the knowledge correlation. This genetic algorithm was implemented using the LibGA open source library [7]. We configured the parameters of LibGA as shown in Table 1 for all experiments described in this paper.

Table 1. LibGA configuration parameters

Parameter	Value	Description
initpool	random	Initial population is generated randomly
pool_size	100	Population size is equal to 100 individuals
stop_after	5000 use_convergence	Algorithm stops after 5000 generations or when convergence is reached [7]
selection	roulette	Individual selection is done according to the roulette method
crossover	uniform	Genes are permutated uniformly
mu_rate	0.3	Mutation rate is equal to 30%
elitism	true	The individual with best fitness remain in the next generation

Four datasets were selected from the Machine Learning Repository of the Department of Information and Computing Science of the University of California, Irvine [8] to test if the use of knowledge correlation improves the performance of VG-RAM neural networks: The Monk's problems 3 datasets, and the Wisconsin breast cancer dataset. These well-known datasets have been used in many academic works, providing material for performance comparisons. We divided each of them into training and test sets, and measured the performance, in terms of the percentage of test samples classified correctly, of the standard VG-RAM and knowledge correlated VG-RAM. These performance measurements were then compared against each other and with that of other classification techniques. Our results are presented in the next section.

5 Experiments

This section briefly describes the datasets, the VG-RAM neural network modeling, the experimental results and the comparisons performed.

5.1 The Monk's Datasets

The Monk's problems are discrete classification problems defined in a domain of robot appearance descriptions [9]. The appearance of the robots may be described by the six attributes shown in Table 2.

Table 2. Robot's appearence attributes

	Attribute	Domain
a_1	head shape	{ round, square, octagon }
a_2	body shape	{ round, square, octagon }
a_3	is smiling	{ yes, no }
a_4	holding	{ sword, balloon, flag }
a_5	jacket color	{ red, yellow, green, blue }
a_6	has tie	{ yes, no }

The Monk's problems consist of asserting if a given robot belongs to a class or not, according with their appearance. The (logical) description of each class differentiates each problem: #1, robot head and body shapes are equal or robot jacket color is red; #2, exactly two of the six robot's attributes possess their first value; #3, the color of the jacket of the robot is green and it is holding a sword, or the color of the jacket of the robot is not blue and its body shape is not octagon. Three datasets, one for each problem (dataset #1, dataset #2 and dataset #3), contain instances of possible descriptions of robots, which are classified as belonging or not belonging to the class associated with each dataset (there are six misclassifications due to noise in the training set of dataset #3). The classification is, then, a binary classification task with no previous knowledge of the logical description of each class. There are 432 instances in all datasets, and 50% of them belong to the class defined in each problem. The instances used for training were selected randomly: 124 from the dataset #1, 169 from #2, and 122 from #3. All instances were used for testing. We have used the datasets as described by Thrun et al. in [9].

The VG-RAM neural network architecture used in this experiment is like the one shown in Fig. 2. It has a single VG-RAM neuron with 17 binary inputs, one for each of the 17 possible values of the robot's attributes (Table 2). If an attribute is present, the corresponding input is set to one, otherwise to zero. The output of the neuron should be set to one if the robot being analyzed by the network belongs to the class associated with the dataset, or zero otherwise. The experimental results are shown in Table 3 and Table 4.

Table 3 shows the best individuals obtained by the GA for each dataset. The genes (or relevance flag) with value equal to one are very much related to the logical description of each problem. For instance, in dataset #1, head and body shapes are defined by the first six genes and jacket color equal to red is defined by the 12^{th} gene. In

dataset #2 the logical rule is defined by the 1st, 4th, 7th, 9th, 12th and 16th genes, but the 17th gene could perfectly replace the 16th, since one is always the opposite of the other. In dataset #3, the noise present in the training data caused the appearance of extra active genes.

Table 3. GA results for the Monk's problems

Monk's Problem	GA iterations	Best individual (17 genes)
dataset # 1	13	1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0
dataset # 2	14	1 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1
dataset # 3	36	1 0 0 1 1 0 0 1 1 1 0 0 0 1 1 0 0

Table 4. The Monk's problems experimental results

Monk's Problem	Standard VG-RAM		VG-RAM with Knowledge Correlation	
	Accuracy	Errors	Accuracy	Errors
dataset # 1	83.56 %	71 out of 432	100.00 %	None
dataset # 2	71.30 %	124 out of 432	96.53 %	15 out of 432
dataset # 3	85.88 %	61 out of 432	90.28 %	42 out of 432

As Table 4 shows, the knowledge correlated VG-RAM performs significantly better than the standard VG-RAM in the Monk's problems – it is superior in all 3 datasets, achieving accuracy gains from 4.40 (dataset #3) to 25.23 (dataset #2) percentage points.

We have also compared the VG-RAM performance on the solution of Monk's problems with that of other classification techniques: Backpropagation with weight decay [10]; Cascade-Correlation with the Quickprop algorithm [11]; and AQ17-HCI with hypothesis-driven constructive induction [12] [13]. The results are shown in Fig. 3. As the graphs in the figure show, VG-RAM with knowledge correlation has the same performance of the other techniques in dataset #1, similar performance in dataset #2 (it outperforms AQ17-HCI), and a somewhat inferior performance with dataset #3.

5.2 The Wisconsin Breast Cancer Dataset

The Wisconsin breast cancer dataset contains results of breast cytology tests and diagnosis colleted at University of Wisconsin Hospitals [14]. Each instance of the dataset belongs to one of two possible classes: benign or malignant. Nine different attributes are used to represent the instances: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. Each attribute has a normalized scalar value between 1 and 10. There are 699 instances in the dataset, 65.5% benign and 34.5% malignant – 559 instances (80%), selected randomly, were used for training, while the remaining 140 (20%) for testing. There are 16 instances in the dataset with the value of one attribute missing.

The classification task is, again, binary, but there is no pre-known logical class description.

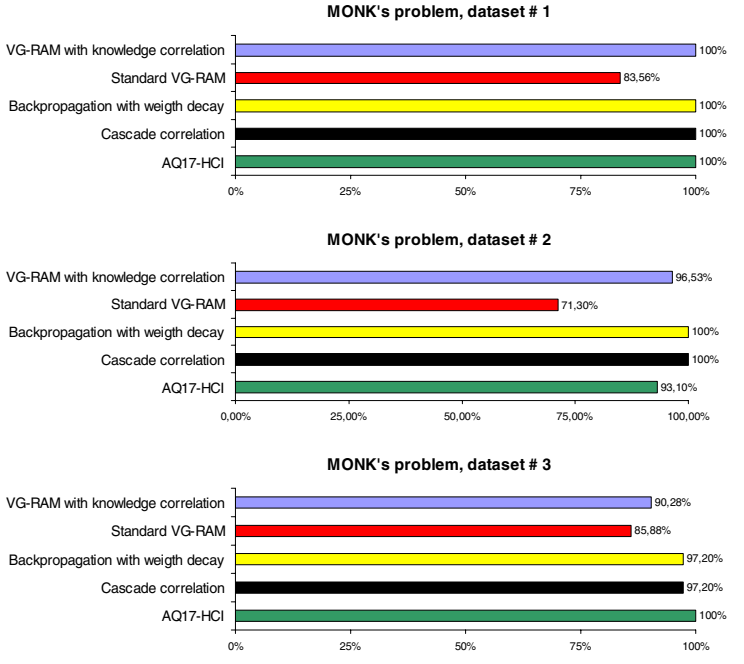

Fig. 3. Performance comparison on the Monk's problems

The VG-RAM neural network architecture used in this experiment is like the one shown in Fig. 2. The architecture has a single neuron that has nine 10-bit vector inputs (a_1 … a_9), each corresponding to one of the nine normalized cytological attributes (the neuron has 9x10 binary inputs). These attributes were coded using the thermometer coding technique [15], in which an integer of value X, in the range 1-N, is mapped to a string of N bits, where the first X bits are set to one and the remaining set to zero (missing attributes were coded as 0101010101). The neuron's output is the binary class value. The experimental results obtained with Wisconsin breast cancer dataset are shown in Table 5.

Table 5. Wisconsin breast cancer experimental results

Standard VG-RAM		**VG-RAM with Knowledge Correlation**	
Accuracy	**Errors**	**Accuracy**	**Errors**
99.29 %	1 out of 140	99.29 %	1 out of 140

As Table 5 shows, the performance of the knowledge correlated VG-RAM and the standard VG-RAM are equally good – only one out of 140 dataset instances was misclassified. Note that, as this example shows, knowledge correlated VG-RAM will not always outperform standard VG-RAM, since it only improves the generalization of standard VG-RAM.

The performance of VG-RAM was also compared with other algorithms presented in the literature for solving the Wisconsin breast cancer problem: Case-Based Reasoning (CBR) with rough sets [16]; Case-Based Reasoning (CBR) with sample correlation [16]; and the C4.5X post-processing algorithm [17]. The Fig. 4 shows this comparison.

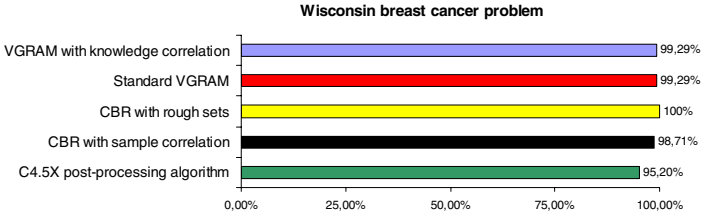

Fig. 4. Performance comparison on the Wisconsin breast cancer problem

As Fig. 4 shows, VG-RAM compares well with other algorithms.

6 Conclusion

In this paper we have used genetic algorithms for finding which bits of the inputs of memorized input-output pairs of VG-RAM neurons correlate to the memorized outputs, and used this information to implement a new distance function that improves VG-RAM generalization performance. We have called this new VG-RAM knowledge correlated VG-RAM. Our experimental evaluation of knowledge correlated VG-RAM networks have shown that they outperform standard VG-RAM or have the same performance. We have obtained classification results, with knowledge correlated VG-RAMs, up to 25.23 percentage points better than standard VG-RAMs using well known datasets.

Our experiments also shown that knowledge correlated VG-RAM networks compares well with other top performing classification techniques, such as Backpropagation with weight decay [10], Cascade-Correlation with the Quickprop algorithm [11], AQ17-HCI with hypothesis-driven constructive induction [12] [13], Case-Based Reasoning (CBR) with rough sets [16], Case-Based Reasoning (CBR) with sample correlation [16], and C4.5X post-processing algorithm [17]. VG-RAM with knowledge correlation has superior performance in some cases, the same performance or equivalent performance in most cases, and inferior performance in a few cases. However, it is important to note that, although knowledge correlated VG-RAMs have not shown superior performance in some cases, they are very easy to implement and have a faster recall than the other alternatives. Knowledge correlated VG-RAM training may nevertheless be slower, but it can be made as a second stage off-line procedure – knowledge correlated VG-RAM may be initially trained and used as standard VG-RAM and the knowledge correlation may be scheduled for an appropriate future moment.

References

1. Aleksander, I.: Self-adaptive Universal Logic Circuits (Design Principles and Block Diagrams of Self-adaptive Universal Logic Circuit with Trainable Elements). IEE Electronic Letters 2 (1966) 231-232
2. Ludermir, T.B., Carvalho, A., Braga, A.P., Souto, M.C.P.: Weightless Neural Models: A Review of Current and Past Works. Neural Computing Surveys 2 (1999) 41-61
3. Aleksander, I.: From WISARD to MAGNUS: a family of weightless virtual neural machines. In: Austin, J. (ed.): RAM-Based Neural Networks. World Scientific (1998) 18-30
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)
5. Komati, K.S., De Souza, A.F.: Using Weightless Neural Networks for Vergence Control in an Artificial Vision System. Applied Bionics and Biomechanics 1 (2003) 21-32
6. Aleksander, I., Browne, C., Dunmall, B., Wright, T.: Towards Visual Awareness in a Neural System. In: Amari, S., Kasabov, N. (eds.): Brain-Like Computing and Intelligent Information Systems. Springer-Verlag, Berlin Heidelberg New York (1997) 513-533
7. Corcoran, A.L. Wainwright, R.L.: LibGA: A User-friendly Workbench for Order-based Genetic Algorithm Research. Proceedings of the ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice (1993) 111-117
8. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html] University of California, Department of Information and Computer Science, Irvine, CA (1998)
9. Thrun, S.B., et al.: The MONK's Problems: A Performance Comparison of Different Learning Algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University (1991)
10. Haykin, S.: Neural Networks: A Comprehensive Foundation. 2nd edn. Prentice Hall (1998)
11. Fahlman S.E., Lebiere, C.: The Cascade-correlation Learning Architecture. In: Touretzky, D.S. (ed.): Advances in Neural Information Processing Systems, Vol. 2. Morgan Kaufmann (1990) 524-532
12. Vafaie, H., De Jong, K.A.: Improving the Performance of a Rule Induction System Using Genetic Algorithms. Proceedings of the First International Workshop on Multistrategy Learning. Harpers Ferry, W. Virginia (1991) 305-315
13. Wnek, J., Michalski, R.S.: Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments. Machine Learning 14:2 (1994) 139-168
14. Wolberg, W.H., Mangasarian, O.L.: Multisurface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology. Proceedings of the National Academy of Sciences, U.S.A., Vol. 87 (1990) 9193-9196
15. Rohwer, R., Morciniec, M.: A Theoretical and Experimental Account of N-tuple Classifier Performance. Neural Computing 8 (1995) 629-642
16. Salamó, M., Golobardes, E.: Vernet, D., Nieto, M.: Weighting Methods for a Case-based Classifier System. Proceedings of Learning'00, Madrid, Spain (2000)
17. Webb, G.I.: Further Experimental Evidence Against the Utility of Occam's Razor. Journal of Artificial Intelligence Research 4 (1996) 397-417