Neural Network Architecture Selection: Size Depends on Function Complexity

Iván Gómez, Leonardo Franco, José L. Subirats, and José M. Jerez

Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga, 29071 Málaga, Spain
{ivan, lfranco, jlsubirats, jja}@lcc.uma.es

Abstract. The relationship between generalization ability, neural network size and function complexity have been analyzed in this work. The dependence of the generalization process on the complexity of the function implemented by neural architecture is studied using a recently introduced measure for the complexity of the Boolean functions. Furthermore an association rule discovery (ARD) technique was used to find associations among subsets of items in the whole set of simulations results. The main result of the paper is that for a set of quasi-random generated Boolean functions it is found that large neural networks generalize better on high complexity functions in comparison to smaller ones, which performs better in low and medium complexity functions.

1 Introduction

The learning and generalization properties of artificial neural networks confer to these models a wide applicability in pattern recognition and classification tasks. The cornerstone of the neural network modeling is the selection of the network architecture for a determined application, since there is not a theoretical formula giving clear insight to this problem. Nevertheless, some general theoretical results have been published about the size of the network needed to implement a desired function [1, 2, 3], but at the time of the implementation the theory is not always accurate.

In particular, some authors [3, 4] have demonstrated that very large networks perform sometimes better than smaller ones, whereas others [8] state that the generalization process depends mainly on the weight values distribution and not on the network size. Moreover, most of the practical results based on simulations [6, 7] concentrate on the use of very few functions (up to 30 different functions), and also in general the complexity of the analyzed functions is ignored [3, 9].

In fact, the complexity of Boolean functions has been studied for a long time, the aim of that being to have a criterion for deciding if a problem is easier to solve or implement than another [12, 3].

This work studies the relationship among learning, generalization, network architectures, and complexity over a set of one thousand different boolean functions. With this aim, we use a recently introduced measure for the complexity of Boolean functions [10, 11] to analyze how the network architecture affects the

S. Kollias et al. (Eds.): ICANN 2006, Part I, LNCS 4131, pp. 122–129, 2006.

generalization ability in different classes of functions grouped by their levels of complexity. Also, the huge amount of simulation data led the authors to use a data mining technique (association rules) to find an statistical correlation among generalization, network architecture and functions complexity. Association rule discovery (ARD) is a particular technique widely used in data mining problems [19, 20, 17, 18] to find interesting associations and/or correlation relationships among large sets of data items. This algorithm extracts relevant information from the data and provides rules in the form of "if-then" statements, and, unlike the if-then rules of Logic, association rules are probabilistic in nature.

2 Methods

2.1 Measure for the Complexity of Boolean Functions

The measure proposed in [9] is based on the results obtained in [15, 16], where a close relationship between the number of examples needed to obtain valid generalization and the number of neighboring examples with different outputs was found. The complexity measure $\mathcal{C}[f]$ is obtained from the number of pairs of neighboring examples having different outputs. The complexity measure used in this paper is defined as:

$$\mathcal{C}[f] \;=\; \mathcal{C}_1[f] + \mathcal{C}_2[f], \tag{1}$$

where $\mathcal{C}_i[f]$, $i = 1, 2$ are the terms taking into account pairs of examples at a Hamming distance one and two. The first term can be written as:

$$\mathcal{C}_1[f] = \frac{1}{N_{ex} * N_{neigh}} \sum_{j=1}^{N_{ex}} \left(\sum_{\{l | Hamming(e_j, e_l)=1\}} (|f(e_j) - f(e_l)|) \right), \tag{2}$$

where the first factor, $\frac{1}{N_{ex} * N_{neigh}}$, is a normalization one, counting for the total number of pairs considered, N_{ex} is the total number of examples equals to 2^N, and N_{neigh} stands for the number of neighbor examples at a Hamming distance of 1. The second term $\mathcal{C}_2[f]$ is constructed in an analogous way. The complexity of the Boolean functions using the measure of Eq. 1 ranges from 0 to 1.5 [9].

2.2 Association Rules Discovery

Data mining, also known as Knowledge Discovery in Databases(KDD), has been defined as the extraction of implicit previously unknown useful information from data. Motivated by the huge amount of information provided by the simulation results, data mining techniques were applied to the process of finding interesting relationships among generalization, complexity and network architecture.

ARD technique is a data mining method used in many applications to discover associations patterns between subsets of items in transactions databases. It detects set of elements that co-occur frequently, creating relations of the form

$X- > Y$, which means that when the antecedent of the rule, X, occurs, it is probably for the consequent, Y, also to be occur. This technique has been extensively used in market analysis [19, 20] and analysis of gene expression data [17, 18].

In addition to the antecedent (*if* condition) and the consequent (*then* condition), an association rule provides two numbers that express the degree of uncertainty about the rule. In association analysis, the antecedent and consequent are disjointed items sets. The first number, named support for the rule, is simply the number of rows that include all items in both antecedent and consequent parts of the rule. The second number represents the lift, another parameter of interest in the association analysis, that gives the ratio of confidence to expected confidence. Expected confidence is the number of transactions, that include the consequent divided by the total number of transactions.

In the context of this work, we define transactions that contains function complexity, network architecture (number of hidden neurons) and generalization capability. Each item included in the transaction was discretized into four categories as follows: $[0, 0.25)$, $[0.25, 0.50)$, $[0.50, 0.75)$ and $[0.75, 1]$ for functions complexity; $[0.5, 0.70)$, $[0.70, 0.85)$, $[0.85, 1.0)$ for generalization ability; and $[0, 10)$, $[10, 20)$, $[20, 30)$, $[30, 100]$ for the number of hidden neurons.

One of the limitations of ARD is the large amount of rules that can be generated, which becomes a major problem in many applications. Some post-processing pruning methods have been proposed to reduce the number of rules [18]. One of the limitations of ARD is the large amount of rules that can be generated. Some pruning methods have been proposed to reduce the number of rules. In this work, we have used a filter designed to eliminate those rules that present low confidence or does not have the generalization item in, what reduces drastically the number of association rules.

2.3 Set of Quasi-random Functions

In a previous work [21], the authors analyzed the generalization ability of different network architectures studying how the generalization ability is affected by the complexity of the functions being implemented and by the size of the architecture. More exactly, generalization ability was tested in six different architectures in size, and a set of 512 symmetric boolean functions was considered. The results showed that was necessary the introduction of a second layer of neurons to improve the generalization ability of very complex functions.

In this paper we extend the study to a set of quasi-random functions. There exists 2^{2^N} boolean functions of N inputs, making their general study very complicated except for very simple and small cases. Totally random functions are very complex with an average complexity around 1.0 [10]. To analyze a large set of different functions we generate functions with different complexity by modifying with a certain probability K outputs of the constant function. The value of K is related to the complexity of the generated function, and the most complex generated function is the parity one. This procedure let us to obtain functions that are asymmetric in the number of outputs and with a complexity in the range

1.0 to 0. One hundred functions were analyzed for each of 10 levels of complexities in which the functions where grouped, with average complexity from 0.1 to 1.0 in steps of 0.1.

2.4 Neural Network Architectures and Simulations

To analyze the generalization ability of different architectures and to study how this property change with network size and functions complexities, we carried intensive numerical simulations for quasi-random Boolean functions generated according to the procedure described before. All the networks were trained with back-propagation with momentum. An early stopping procedure was implemented with the aim of avoid overtraining and improve generalization. The validation error was monitored during training and at the end of the maximum number of epochs permitted, the generalization error was taken at the point where the validation error was minimum.

The number of examples available for training was 256 as in all cases we consider input functions of size N=8. We divided the examples in a training set containing 128 examples, and into validation and generalization test sets containing 64 examples each one. The learning rate constant was fixed during training and equal to 0.01 and the momentum term was set to 0.05. The maximum number of epochs allowed for training was set to 10000.

3 Results

We performed numerical simulations on one hidden layer neural networks using the whole set of quasi-random functions (up to 1000 different functions) with $N = 8$ input variables. The number of neurons for the analyzed cases ranged from 5 to 100 in steps of 5 hidden units. The best results for each complexity level, in terms of the generalization ability obtained, were computed simulating for every network architecture and averaging over the 100 quasi-random functions. In the case of a coincidence in the generalization ability from two architectures with a different number of hidden neurons, a simple and general idea about what network size come from Occam's razor: the simpler the solution the better. In Fig. 1 mean values for architecture sizes are represented vs function complexity, and standard deviations were also computed for every complexity group.

It is possible to observe from Fig. 1 that the most appropriate network size for obtaining the best generalization ability remains small for low level complexity groups (from 0.0 to 0.55), noting also that the standard deviation is also relatively low. As the complexity of the functions increase, the number of hidden neurons grows up fast for groups with a complexity between 0.64 and 0.97. In table 1, the average generalization ability obtained for the different architectures used in every complexity group is shown together with the training error at the point in which the minimum validation error was found.

In Fig. 2 the number of iterations as a function of the number of hidden neurons, for each complexities group, is shown. In this figure the lowest level

Table 1. Average generalization ability and final training error (with standard deviations between parenthesis) obtained for the network architectures constructed to compute all Boolean quasi-random functions with N=8 inputs

Complexity group (mean value)	Generalization ability	Training error
0.07	0.976 (0.009)	0.015 (0.015)
0.14	0.968 (0.010)	0.031 (0.017)
0.27	0.929 (0.019)	0.078 (0.033)
0.34	0.898 (0.023)	0.078 (0.029)
0.45	0.875 (0.022)	0.125 (0.038)
0.55	0.835 (0.026)	0.171 (0.039)
0.64	0.804 (0.029)	0.203 (0.050)
0.75	0.757 (0.032)	0.250 (0.052)
0.86	0.695 (0.041)	0.296 (0.056)
0.97	0.585 (0.067)	0.421 (0.072)

complexities groups are located at the top of the figure, being the highest level complexity group labeled as 10. Figure 2 demonstrates that the time of training (number of epochs) in very complex functions (labeled as $8-10$) is proportional to the network size. Moreover, in the case of low complexity functions, the time

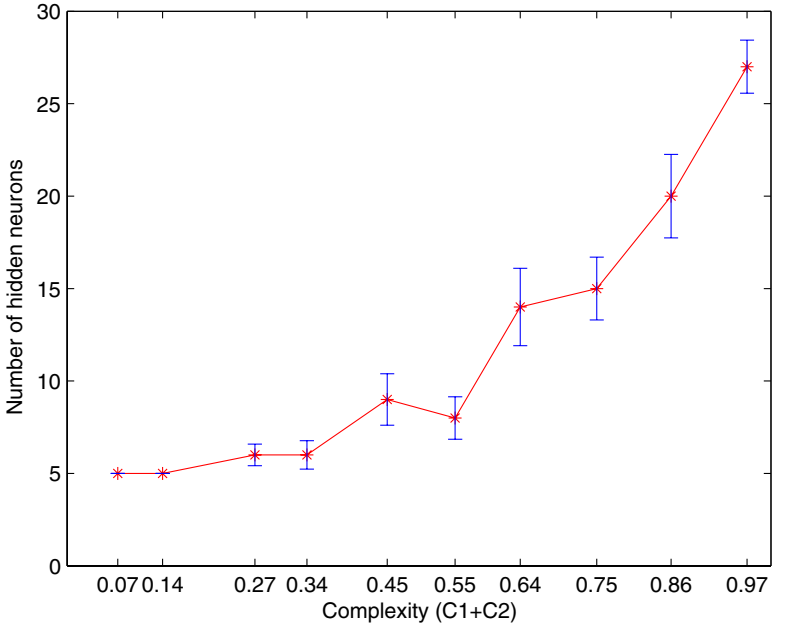

Fig. 1. Number of hidden neurons vs function complexity for quasi-random functions with N=8 inputs. The x-axis labels identify the mean values for each level of complexity.

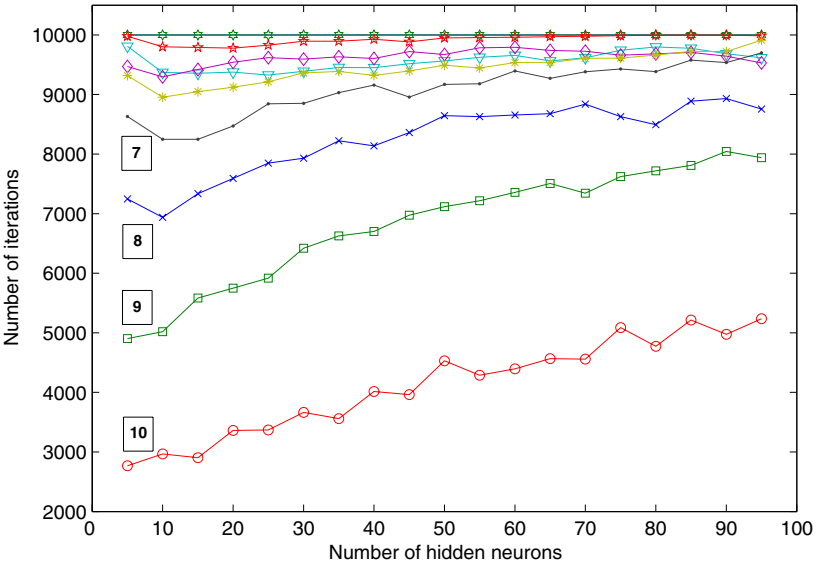

Fig. 2. Training time, in epochs, as a function of the network size, in number of hidden neurons, for every complexity group of Boolean functions

necessary to train the network is approximately constant and independent of the network size selected to implement that function.

The ARD technique was applied to the whole set of 20.000 simulations results. Association rules were extracted with absolute minimum support value of 10, minimum confidence of 40% and minimum improvement of one, obtaining a total of 35 association rules. After filtering and analyzing the set of rules extracted, it is possible to observe that a correlation exists among the architecture size, the complexity level and the generalization ability. That is confirmed through association rules with the form: { low complexity } and { low network size } → { high generalization ability }, with confidence 0.98 and lift 1.66; and others as { high complexity } and { medium network size } → { low generalization ability }, with confidence 0.93 and lift 1.60. In this case, the ARD technique confers statistical precision to graphical results, since the confidence value can be interpreted as the a posteriori probability for the consequent given the antecedent. Moreover, a lift value greater than 1.0 is interpreted as a positive correlation between the antecedent and the consequent.

4 Discussion

We have analyzed through numerical simulations the relationship between neural network size and function complexity. It was found that for groups of low complexity functions, smaller architectures have a better generalization ability than larger ones. Also, it was found that the optimal value for the number of

hidden neurons (between the analyzed values) grows with function complexity. This result is what is expected from theoretical reasons but the interestingly, up to our knowledge, is one of the first times that this problem is systematically studied and reported. We have also made use of a extraction rule technique (ARD) to analyze the result of the simulations and this technique confirmed our analysis giving also statistical confidence. Regarding the architecture selection process, much work remains to be done and we are in the process of studying it in multi-layers networks and modular architectures.

Acknowledgement. This work was supported by the "Comisión Interministerial de Ciencia y Tecnología" (Spain) through grant TIN2005-02984, and by FEDER funds. Leonardo Franco acknowledges support from the Spanish Ministry of Education and Science through a "Ramón y Cajal" fellowship.

References

1. Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan/IEEE Press.
2. Baum, E.B. & Haussler, D. (1989) What size net gives valid generalization ? *Neural Computation*, *1*, pp. 151-160..
3. Lawrence, S., Giles, C. L., & Tsoi, A. C. (1996). What Size Neural Network Gives Optimal Generalization ? Convergence Properties of Backpropagation. In Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, Univ. of Maryland.
4. Caruana, R., Lawrence, S., & Giles, C.L. (2001). Overfitting in Neural Networks: Backpropagation, Conjugate Gradient, and Early Stopping. In Leen, T. K., Dietterich, T. G. & Tresp, V. editors, Advances in Neural Information Processing Systems, MIT Press, *13*, pp. 402-408.
5. Krogh, A. & Hertz,J.A. (1992) A simple weight decay can improve generalization. In J.E. Moody, S. J. Hanson, & R. P. Lippmann editors, Advances in Neural Information Processing Systems Morgan Kaufmann, San Mateo, CA, *4*, pp. 950 957.
6. Prechelt, L. (1998). Automatic Early Stopping Using Cross Validation: Quantifying the Criteria. *Neural Networks*, *11*, pp.761-767.
7. Setiono,R. (2001) Feedforward neural network construction using cross-validation, *Neural Computation*, *13*, pp. 2865-2877.
8. Bartlett,P.L. (1997). For valid generalization the size of the weights is more important than the size of the network. In M.C. Mozer, M. I. Jordan, & T. Petsche, editors, Advances in Neural Information Processing Systems, MIT Press, *9*, pp. 134-140 .
9. Franco, L. & Anthony, M. (2004). On a generalization complexity measure for Boolean functions. In *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, IEEE Press, pp. 973-978.
10. Franco, L. Generalization ability of Boolean functions implemented in feedforward neural networks. Neurocomputing. (2006). In Press.
11. Franco, L. and Anthony, M. The influence of oppositely classified examples on the generalization complexity of Boolean functions. IEEE Transactions on Neural Networks. (2006). In Press.

12. Wegener, I. (1987) *The complexity of Boolean functions*. Wiley and Sons Inc.
13. Siu, K.Y., Roychowdhury, V.P., & Kailath, T. (1991) Depth-Size Tradeoffs for Neural Computation *IEEE Transactions on Computers*, *40*, pp. 1402-1412.
14. Franco, L. & Cannas, S.A. (2004). Non glassy ground-state in a long-range antiferromagnetic frustrated model in the hypercubic cell *Physica A*, *332*, pp. 337-348.
15. Franco, L. & Cannas, S.A. (2000). Generalization and Selection of Examples in Feedforward Neural Networks. *Neural Computation*, *12*, 10, pp. 2405-2426.
16. Franco, L. & Cannas, S.A. (2001). Generalization Properties of Modular Networks: Implementing the Parity Function. *IEEE Transactions on Neural Networks*, *12*, pp. 1306-1313.
17. Becquet, C. & Blachon, S. & Jeudy, B. & Boulicaut, J.F. & Gandrillon, O. (2002). Strong association rules mining for large-scale gene-expression data analysis: A case study on human SAGE data. *Genome Biology*, *3*, pp. 1-16.
18. Creighton, C. & Hanash, S. (2003). Mining gene expressions databases for association rules. *Bioinformatics*, *19*, pp. 79-86.
19. Agrawal, R. & Imielinski, T. & Swami, A. (1993). Mining associations rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD international conference on Management of data*, Washignton D.C., pp. 207-216.
20. Brian, S. & Motwani, R. & Silverstein, C. (1997). Beyond Market baskets: Generalizing associations rules to correlations. In *Proceedings of the ACM SIGMOD conference*, Tucson, pp. 265-276.
21. Franco, L. & Jerez, J.M. & Bravo, J.M (2005). Role of function complexity and network size in the generalization ability of feedforward networks. *LNCS*, *3512*, pp. 1-8.