Extension of the Generalization Complexity Measure to Real Valued Input Data Sets

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

Departamento de Lenguajes y Ciencias de la Computación
E.T.S.I. Informática, Campus de Teatinos, Málaga, Spain
{ivan,lfranco,jja,jlsubirats}@lcc.uma.es

Abstract. This paper studies the extension of the Generalization Complexity (GC) measure to real valued input problems. The GC measure, defined in Boolean space, was proposed as a simple tool to estimate the generalization ability that can be obtained when a data set is learnt by a neural network. Using two different discretization methods, the real valued inputs are transformed into binary values, from which the generalization complexity can be straightforwardly computed. The discretization transformation is carried out both through a very simple method based on equal width intervals (EW) and with a more sophisticated supervised method (the CAIM algorithm) that use much more information about the data. A study of the relationship between data complexity and generalization ability obtained was done together with an analysis of the relationship between best neural architecture size and complexity.

Keywords: Neural network, Architecture size, Real-valued function, CAIM, Discretization algorithms.

1 Introduction

Deciding how many nodes to include in the hidden layer of a feed forward neural network in order to classify a set of patterns or to approximate a given data set is a controversial issue. Using a variety of mathematical methods and approximations, theoretical results that give an indication about how to solve this problem have been obtained [1,3,4,5,6,7,8,9,10], but at the time of the implementation some of them offer not clear help or are very difficult to implement. As a consequence, more practical approaches, with results supported by numerical simulations have been proposed [19,17,18]. Despite all these approaches, still the main tendency at the time of selecting a neural network architecture for a given problem is the trial-and-error approach.

Recently, a new point of view to the architecture selection problem have been tried by Franco and colleagues [14,13,12], who have introduced a new measure for the estimating the complexity of Boolean functions. The measure named Generalization Complexity (GC) tries to estimate from the set of available data for a given problem what it will be the generalization ability expected, as several tests have shown that a high correlation exists between the GC measure and the prediction accuracy obtained when Boolean data is used to train a Feed-forward Neural Network (FFNN). As mentioned, the method to select proper neural architectures based on the GC complexity of the data has been only applied to Boolean input data [12], as the GC measure is defined for Boolean inputs. In this work, through the use of two different discretization (or binarization) methods [28], we compute the GC measure of real-valued input data from a well known benchmark data set. 20 classification problems from the standardized PROBEN1 [2] benchmark collection were used as testing data for analyzing the relationship between the GC measure and the generalization ability obtained when the problems are learnt by FFNN, and also we have analyze the relationship between the size of the optimal-found network and the generalization ability obtained. We have used two very different discretization methods, a very simple and unsupervised one, that discretize the inputs using equal width intervals (EW) [24], and a more complex supervised discretization algorithm, named Class-Attribute Interdependence Maximization (CAIM) based on the maximization of the interdependence between class and attributes [27].

The paper is structured as follows: section 2 contains the details of the methods and data sets used, followed by the results obtained from the extensive numerical simulations that are presented in section 3 to finally discuss the results in section 4.

2 Methods and Benchmark Data Sets

2.1 The GC Measure

The GC measure was introduced by Franco and colleagues [14,13], and was derived from evidence that pairs of bordering examples, those lying closely at both sides of separating hyperplanes that classify different regions, play an important role on the generalization ability obtained in classification problems when neural networks are used as predictive methods [15,16] . The GC measure of a Boolean function f, $\mathcal{C}[f]$ can be simply computed by counting the number of pairs of neighboring examples having opposite outputs located at Hamming distances 1 and 2.

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

where $\mathcal{C}_i[f]$, $i = 1, 2$ are the two terms of the measure taking into account pairs of examples at a Hamming distance one and two. Explicitly, 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.

In order to apply the GC measure for the estimation of the size of a proper neural network architecture, a fit of the relationship between architecture and GC has to be obtained for a set of training functions, to then use this fit to estimate an adequate architecture according to the GC complexity of a new data set. This work was carried out in [12] and the results compared to other existing approaches, finding that the GC-based method works close to the optimal.

2.2 Binarization of the Data Set: Equal Width and CAIM Algorithms

As mentioned before, the GC measure has only be defined for binary input data, and thus in order to transform real-valued inputs to binary values a discretization (or binarization) algorithm should be applied.

Usually, discretization algorithms are classified in two main categories: unsupervised and supervised algorithms.

- Unsupervised algorithms discretize attributes without taking into account respective class labels. They are the simplest to use and implement. The most representative algorithms of this category are equal-width and equal-frequency methods. The equal-width discretization algorithm find the minimum and maximum values for each attribute, and then divides this range into a number n_{F_i} of user specified equal width intervals. The equal-frequency discretization algorithm determines the minimum and the maximum values of the attribute, sort all values in ascending order, and divides the range into a user-defined number of intervals so that every interval contains the same numbers of sorted values.
- Supervised algorithms discretize attributes by taking into account the interdependence between class labels and the attribute values. The representative algorithms are: maximum entropy [20], Patterson and Niblett [21], information entropy maximization (IEM) [22], and other information-gain or entropy-based algorithms [23], statistics-based algorithms like ChiMerge [24], and clustering-based algorithms like K-means discretization [25].

  CAIM [27](Class-Attribute Interdependence Maximization) is an algorithm designed to work with supervised data. The goal of the CAIM algorithm is to maximize the class-attribute interdependence and at the same time generate a minimal number of discrete intervals. A main advantage of the CAIM algorithm is that does not require to predefine the number of intervals, as opposed to some other discretization algorithms, as CAIM automatically selects the number of intervals without any user supervision.

In this work, we implemented and applied one unsupervised and one supervised algorithms. The unsupervised algorithm used was an equal width one (EW) that we tried with 3 and 5 intervals. Among the supervised algorithms we choose the CAIM one, as good results have been reported from its performance [27].

2.3 The Benchmark Data Set

In order to investigate the performance of the GC measure over real-valued input functions, we have created several binary classification data sets from real world problems, specifically from the PROBEN1 benchmark data set[2]. This benchmark is a collection of problems for neural network learning in the field of pattern classification and function approximation plus a set of rules and conventions for carrying out benchmark tests. All data sets are represented with real-valued attributes, and they are realistic problems presented in the same simple format. The problem representation in PROBEN1 is one of the best improvements made in the benchmark, being a fixed representation that improves the comparability of results, reducing drastically the work to be done for running the benchmark. The data sets included in the present analysis are: **cancer1, card1, diabetes1, gene1, heart1, heartc1, horse1 and thyroid1**. For simplicity we have considered m-class output data as m different two-class problems and have performed this process for every function in the benchmark for which m was larger than 2. The final data used for the study comprise the 20 two-class classification problems shown in table 1.

The first column in Table 1 shows a problem identifier from 1 to 20, followed by the source function name, the percentage of examples with output class 0 or 1, the number of inputs of the source problem, the number of inputs obtained when the CAIM binarization algorithm is applied and finally the complexity values obtained using the GC measure for the three cases considered. The last three columns show the result of the application of the GC measure to the data after applying the CAIM algorithm and the equal-width method for 3 and 5 intervals. The number of inputs used for the case of the equal width algorithm can be straightforward multiplying by 3 and 5 the source number of inputs, as the EW method creates, for every single input, a number of equivalent inputs equal to the number of intervals considered.

3 Simulations

We carried intensive numerical simulations for the 20 2-class functions described in table 1 analyzing the generalization ability of FFNN as the number of neurons in the single hidden layer is varied between 2 and 30. All the networks were trained with scale conjugate gradient back propagation algorithm [26], due to its modest requirements of memory and the good performance of this algorithm for problems with large number of weights. The maximum number of epochs allowed for the training procedure was set to 5000 and the minimum performance gradient used was $1e-5$, stopping the training if the magnitude of the gradient drops below this value. In order to avoid the problem of overfitting that degrades the generalization ability, we implemented the well known early stopping procedure [11], in which the validation error is monitored during training to then choose the synaptic weights observed at the point where the validation error took its minimum value. A stratified 10-cross validation method was used to train the different networks, with the aim of preserving the percentage of examples with

Table 1. Description of the 20 functions generated from the PROBEN1 benchmark used in the present study. The table shows in the columns a function identifier (id), the original source problem name, the distribution of the output class in 0's and 1's, the source number of inputs, the number of discretized inputs using the CAIM binarization algorithm and in the last three columns the values of the GC measure. The three GC values corresponds to the values obtained from the binarized version of the problem using the CAIM algorithm and the equal width (EW) algorithm for 3 and 5 intervals.

id	**Source**	% **class output (0-1)**	# source inputs	# CAIM inputs	GC-Complexity		
					CAIM	EW-3	EW-5
f_1	cancer1	$65 - 35$	9	18	0.0525	0.0640	0.0470
f_2	card1	$45 - 55$	51	100	0.2033	0.2079	0.1969
f_3	diabetes1	$35 - 65$	8	16	0.3761	0.3911	0.3351
f_4	gene1	$76 - 24$	120	240	0.1617	0.1617	0.1617
f_5	gene1	$76 - 24$	120	240	0.1764	0.1764	0.1764
f_6	gene1	$48 - 52$	120	240	0.2417	0.2417	0.2417
f_7	glass1	$67 - 33$	9	18	0.3105	0.3449	0.3676
f_8	glass1	$64 - 36$	9	18	0.4048	0.4100	0.3562
f_9	glass1	$92 - 8$	9	18	0.1570	0.1440	0.1452
f_{10}	glass1	$94 - 6$	9	18	0.0591	0.0650	0.0339
f_{11}	glass1	$96 - 4$	9	18	0.0566	0.0653	0.0651
f_{12}	glass1	$86 - 14$	9	18	0.0573	0.0919	0.0425
f_{13}	heart1	$44 - 56$	35	67	0.2412	0.2660	0.2252
f_{14}	heartc1	$54 - 46$	35	59	0.2333	0.3048	0.2256
f_{15}	horse1	$38 - 62$	58	116	0.3097	0.3505	0.3075
f_{16}	horse1	$75 - 25$	58	116	0.2454	0.2582	0.2328
f_{17}	horse1	$85 - 15$	58	116	0.1861	0.1799	0.1850
f_{18}	thyroid1	$97 - 3$	21	42	0.0120	0.0449	0.0441
f_{19}	thyroid1	$95 - 5$	21	42	0.0553	0.0996	0.0993
f_{20}	thyroid1	$7 - 93$	21	42	0.0543	0.1394	0.1383

different output for each fold. Each 10-fold cross validation was set up choosing iteratively a test fold, then a random validation fold, and the remaining eight folds used for training. The cross validation process was repeated 5 times with a different random seed value, and the final result for each considered network was the mode of these 5 values and then the median of the 10-fold combination. The simulations were run on Matlab code under the Linux operating system in a cluster of 10 blades interconnected with infiniband, each one equipped with 2 Xeon Quadcore processors.

4 Results

In table 1 the values of the GC measure obtained with the different discretization methods used are shown in the last three columns. As it can be appreciated from the table, in almost all cases the complexity values are very similar for the three cases considered. Some differences can be observed for the three last cases analyzed, for which the output class distribution is highly unbalanced.

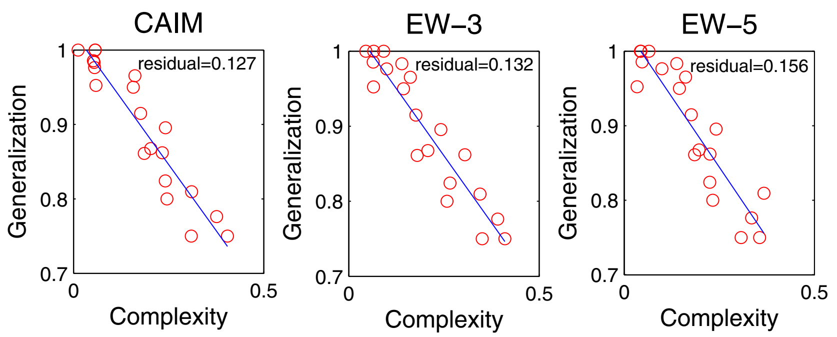

Fig. 1. Generalization ability vs function complexity (GC) for the 20 real valued input functions analyzed for the case of applying the CAIM algorithm and the equal width algorithm with 3 and 5 intervals. The solid curves shown are a linear fit of the results, and the corresponding residual errors are indicated on top of the individual figures.

An analysis of these cases have shown that, for this kind of highly unbalanced output functions, a simple method like the equal interval one, can overestimate the "true" complexity value.

Figure 1 shows the generalization ability obtained as a function of the GC measure for the whole set of test functions using the CAIM and the equal width algorithm with 3 and 5 intervals. A linear fit of the generalization ability as a function of the GC complexity is indicated by a solid line and the residual error obtained indicated on top of each individual graph. The residual error is lower for the CAIM method, indicating a better linear fit between generalization and GC for the case of using the CAIM algorithm. These results confirm the previously ones obtained with true Boolean input functions [14,13,12] and are expected as a lower generalization ability should be obtained for problems with a higher complexity.

Regarding the different discretization methods applied (CAIM and EW with 3 and 5 intervals), we can observe a slightly better fit when the CAIM algorithm is used, but not big differences are observed, especially considering that the CAIM algorithm use much more information about the function than the rather simple EW algorithm.

We have further analyzed the relationship between adequate neural architectures and problem complexity, studying the generalization ability obtained when the problems are implemented in FFNN of varying size. In Figure 2 the values of the number of neurons in the hidden layer of the neural architectures used vs the GC-measure of the data sets are represented. The shown values are for the architectures that produced the best generalization ability when varying the number of hidden neuron in the single hidden layer considered between 2 and 30. In the case of obtaining the same value for the generalization ability, the smaller architecture was chosen following Occam's razor principle: the simpler the solution the better.

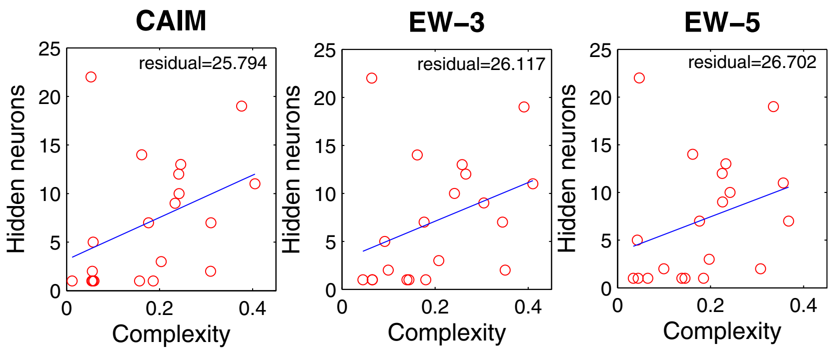

Fig. 2. Best architecture size vs function complexity (GC) for the 20 real world functions described in table 1. The curves are a linear fits of the results obtained when the functions were discretized with the CAIM algorithm, and the 3 and 5 equal-width discretization method.

For the three cases considered and shown in separate graphs in Figure 2, the relationship between the GC measure and the best architecture size was similar, with a slightly better fit found for the case of using the CAIM algorithm as discretization process, as shown by the residual value indicated in the figures. The residual values are somewhat large in all three cases as the fit it is not extremely accurate due to the variability existing in the chosen neural sizes. In particular, note that in each of the three the graphs there is a problem with low GC complexity for which a very large preferred architecture with 22 neurons lead to the best generalization results. This particular value is far apart from the linear fit leading to large residual error. We have observed previously these particular cases, that indicate that the relationship between best architecture and function complexity holds on average but some caution is necessary in individual cases. Nevertheless, we note that in most of these particular cases the difference on generalization between the preferred size network and a linear predicted one is fairly small.

5 Conclusions

Through the application of two different discretization procedures, we have been able to compute the generalization complexity of 20 different benchmark data sets. We have also verified that the relationship, previously obtained only for Boolean input data, between generalization and complexity is preserved, obtaining a clear linear fit (cf. Figure 1). Furthermore, on average, we also found that a better generalization ability can be obtained if a larger number of neurons in the neural architectures is used for more complex functions, even if large deviations can be observed for particular cases. Regarding the two different binarization procedures used, we have found no much difference between their results, even if we have compared a very simple method (equal width algorithm) to a more complex one using extra information from the data (the CAIM algorithm). For data sets with an unbalanced distribution of output class, some differences in GC values were obtained, implying that in these cases the CAIM algorithm should be the preferred method.

The overall conclusion of the present results should be that in principle the application of the generalization complexity measure to real valued input functions for the problem of finding an adequate size architecture is feasible. We plan to do a more in-depth analysis of the architecture selection method that can be derived from this work, in order to analyze its application in a real practical case.

Acknowledgements

The authors acknowledge support from the Spanish Ministry of Education and Science (MICINN) through grants TIN2008-04985 and from Junta de Andalucía through grants P06-TIC-01615 and P08-TIC-04026 (all including FEDER funds).

References

1. Baum, E.B., Haussler, D.: What Size Net Gives Valid Generalization? Neural Comput. 1(1), 151–160 (1990)
2. Prechelt, L.: A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical Report 21/94. Fakultat fur Informatik, U. Kalrsruhe, Germany (1994)
3. Barron, A.R.: Approximation and Estimation Bounds for Artificial Neural Networks. Mach. Learn. 14(1), 115–133 (1994)
4. Camargo, L.S., Yoneyama, T.: Specification of Training Sets and the Number of Hidden Neurons for Multilayer Perceptrons. Neural Comput. 13(12), 2673–2680 (2001)
5. Teoh, E., Xiang, C., Chen, K.: Estimating the Number of Hidden Neurons in a Feedforward Network Using the Singular Value Descomposition. In: Wang, J., Yi, Z., Żurada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3971, pp. 858–865. Springer, Heidelberg (2006)
6. Mirchandani, G., Cao, W.: On Hidden Nodes for Neural Nets. IEEE Trans. on Circuits and Systems 36(5), 661–664 (1989)
7. Arai, M.: Bounds on the Number of Hidden Units in Binary-Valued Three-Layer Neural Networks. Neural Networks 6(6), 855–860 (1993)
8. Zhang, Z., Ma, X., Yang, Y.: Bounds on the Number of Hidden Neurons in Three-Layer Binary Neural Networks. Neural Networks 16(7), 995–1002 (2003)
9. Yuan, H.C., Xiong, F.L., Huai, X.Y.: A Method for Estimating the Number of Hidden Neurons in Feedforward Neural Networks Based on Information Entropy. In: Computers and Electronics in Agriculture, pp. 57–64 (2003)
10. Liu, Y., Janusz, A., Zhu, Z.: Optimizing the Number of Hidden Neurons in Neural Networks. In: AIAP 2007, Proceedings of the 25th IASTED International Multi-Conference, pp. 121–126 (2007)
11. Haykin, S.: Neural Networks: A Comprehensive Foundation. Macmillan, New York (1994)

12. Gómez, I., Franco, L., Jerez, J.: Neural Network Architecture Selecion. Can Function Complexity Help? Neural Processing Letters 30, 71–87 (2009)
13. Franco, L., Anthony, M.: The Influence of Oppositely Classified Examples on the Generalization Complexity of Boolean Functions. IEEE Transactions on Neural Networks 17, 578–590 (2006)
14. Franco, L.: Generalization Ability of Boolean Functions Implemented in FeedForward Neural Networks. Neurocomputing 70, 351–361 (2006)
15. Franco, L., Cannas, S.A.: Generalization and Selection of Examples in Feedforward Neural Networks. N. Comp. 12(10), 2405–2426 (2000)
16. Franco, L., Cannas, S.A.: Generalization Properties of Modular Networks: Implementing the Parity Function. IEEE Trans. on Neural Networks 12, 1306–1313 (2001)
17. Masters, T.: Practical Neural Network Recipes in C++. Academic Press Professional, Inc., London (1993)
18. Neuralware, Inc.: The Reference Guide (2001)
19. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2005)
20. Wong, A.K., Chiu, D.K.: Synthesizing Statistical Knowledge from Incomplete Mixed Mode Data. IEEE Trans. Pattern Analysis and Machine Intelligence 9, 796–805 (1987)
21. Paterson, A., Nibblet, T.B.: ACLS Manual. Int. Teminals Ltd., Edinburgh (1987)
22. Fayyad, U.M., Irani, K.B.: Multi-Interval Discretization of Continuous Valued Attributes for Clasification Learning. In: Proceedings of the 13th International Joint Conference Artificial Intelligence, pp. 1022–1027 (1993)
23. Dougherty, J., Hohavi, R., Sahami, R.: Supervised and Unsupervised Discretization of Continuous Features. In: Proceedings 12th International Conference Machine Learning, pp. 194–202 (1995)
24. Kerber, R.: Chimerge. Discretization of Numeric Attributes. In: Proceedings of Ninth International Conference of Artificial Intelligence, pp. 123–128 (1992)
25. Liu, H., Setiono, R.: Feature Selection Via Discretization. IEEE Knowledge and Data Eng. 9(4), 642–645 (1997)
26. Moller, M.F.: Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Neural Networks 6(4), 525–533 (1993)
27. Kurgan, L.A., Cios, K.J.: CAIM Discretization Algorithm. IEEE Transactions of Knoweledge and Data Eng. 16(2), 145–153 (2004)
28. Maimon, O., Rokach, L.: Data Mining and Knoweledge Discovery Handbook. Springer, New York (2005)