Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 815156, 9 pages
http://dx.doi.org/10.1155/2014/815156

*Research Article*

# The Generalization Complexity Measure for Continuous Input Data

**Iván Gómez,[1] Sergio A. Cannas,[2] Omar Osenda,[2] José M. Jerez,[1] and Leonardo Franco[1]**

[1] *Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, 29071 Málaga, Spain*
[2] *Facultad de Matemática, Astronomía y Física, Universidad Nacional de Córdoba, 5000 Córdoba, Argentina*

Correspondence should be addressed to Iván Gómez; ivan@lcc.uma.es

Received 18 December 2013; Accepted 5 March 2014; Published 10 April 2014

Academic Editors: B. Liu and T. Zhao



We introduce in this work an extension for the generalization complexity measure to continuous input data. The measure, originally defined in Boolean space, quantifies the complexity of data in relationship to the prediction accuracy that can be expected when using a supervised classifier like a neural network, SVM, and so forth. We first extend the original measure for its use with continuous functions to later on, using an approach based on the use of the set of Walsh functions, consider the case of having a finite number of data points (inputs/outputs pairs), that is, usually the practical case. Using a set of trigonometric functions a model that gives a relationship between the size of the hidden layer of a neural network and the complexity is constructed. Finally, we demonstrate the application of the introduced complexity measure, by using the generated model, to the problem of estimating an adequate neural network architecture for real-world data sets.

## 1. Introduction

Feed-forward neural networks trained by back-propagation have become a standard technique for classification and prediction tasks given their good generalization properties. However, the process of selecting adequate neural network architecture for a given problem is still a controversial issue. Several important contributions regarding the number of hidden neurons needed to implement a given function in a neural architecture have been made using different methods. Baum and Haussler [1] obtained some bounds on the number of neurons in an architecture related to the number of training examples that can be learnt using networks composed of linear threshold networks. Barron [2] made an important contribution about the approximation capabilities of feed-forward networks, computing an estimation of the number of hidden nodes necessary to optimize the approximation error. Camargo and Yoneyama [3] obtained a result for estimating the number of nodes needed to implement a function using Chebyshev polynomials and previous results from Scarselli and Chung Tsoi [4] about the number of nodes needed for approximating a given function by polynomials. Hunter et al. [5] focused on the importance of selecting the learning algorithm to train closer to optimal architectures. Methods based on the geometry of output classes [6–8], single value decomposition [9], information entropy [10], and the signal to noise ratio [11] have been used to obtain an approximation to the size of hidden layer in a neural architecture.

Some of the previous studies tried to determine the adequate architecture depending on the complexity of the data set available for a given problem, but as expected measuring the complexity of data is a difficult task. Firstly, it has to be clearly defined what exactly the measure tries to quantify, as complexity can be related to several aspects of the data. Even if different complexity measures related to the size of the architectures needed to implement the data or to the complexity of learning have been proposed in the past [12–14], they have not been applied to the neural network architecture selection problem, in principle because they have not been proposed with this focus.

Moreover, several approaches have been proposed within the learning theory area to analyze the relationship between generalization and complexity. Ho et al. [15, 16] studied the

complexity that characterizes the difficulty of a classification problem, and they suggest using this value to guide the selection of classifier. Sánchez et al. [17] tried to characterize the behavior of the k-NN rule when working under certain situations. More specifically, their analysis focused on the use of some data complexity measures to describe class overlapping, feature space dimensionality, and class density and discover their relation with the practical accuracy of this classifier. Duch et al. [18] suggested that the identification of datasets with high complexity is important to test new methods in computational intelligence.

But most of these analyses focused on the complexity of the architectures and on the error obtained at the end of the training process rather than on the intrinsic complexity of the data. Recently, Franco and colleagues [19, 20] have proposed a complexity measure named "generalization complexity" (GC) that aims to quantify the level of generalization ability that can be expected when Boolean data are used in a classification algorithm. The measure has been also used in the process of architecture selection involved in the implementation of a neural network, as it is expected that for more complex data larger neural network architectures might be more adequate [21]. Nevertheless, the proposed measure can only be applied to Boolean input data so, in this work, the Boolean generalization complexity is first extended to the continuous input case, to then perform a series of tests to validate the proposal using a set of continuous functions with parametrized complexity. Also, by using the set of orthonormal Walsh functions, we extend the proposal for its use with patterns of data. Finally, a model is built from which it is possible to estimate the adequate feed-forward neural network architecture for real-world benchmark data sets by choosing the number of neurons to include in the hidden layer, as the size of the input and output layers is determined by the problem.

## 2. The Generalization Complexity Measure and Its Extension to Real Input Values

Our main goal in this work is to extend the GC measure defined in $f : \{0,1\}^D \to \{0,1\}$ for real input and real output functions $f : [0,1]^D \to [-1,1]$. The choice of the intervals $[0,1]$ for the input and $[-1,1]$ for the output is arbitrary and it is used for simplicity with no restrictions for the general case. We will analyze the more general case of having a continuous output as this case can later be easily particularized to the Boolean output case, more related to classification problems.

The original definition of the GC measure [19, 20] comprises two terms accounting for the first and second nearest neighbor pairs of input data points ($\{e_i\}$), where the neighborhood is defined in terms of their Hamming distance. Let $N_{\text{ex}}$ be the total number of examples (or equivalently patterns) considered and $N_{\text{neigh}}$ the number of first nearest neighbors that every example $(e_i, f(e_i))$ has; that is, examples that are the closest Hamming distance. The first term of the GC measure, $C_1$, known to be the more influential, is defined in Boolean space as

$$C_1[f] = \frac{1}{N_{\text{ex}} N_{\text{neigh}}} \sum_{j=1}^{N_{\text{ex}}} \left( \sum_{\text{Hamming}(e_i,e_j)=1} \left| f(e_i) - f(e_j) \right| \right), \tag{1}$$

where the first factor is a normalization one taking into account the number of pairs considered. Essentially, (1) measures the proportion of neighboring pairs that have different output, that is, belong to different output classes.

In the previous equation, the distance between pairs of inputs is measured by the Hamming distance, but this measure is not applicable for real valued input data. Instead, we will opt for a straightforward choice and use the Euclidean distance. We consider first the 1-dimensional (1D) case corresponding to a single continuous input variable, starting the process by discretizing the input interval $[0,1]$ in $N$ subintervals of length $h = 1/N$. In this way a data point, $e_i$, will be indicated by the subinterval in which its coordinates are included $(x_{i-1}, x_i]$, where $x_i = ih$ $(i = 1, 2, \ldots, N)$, with $x_0 = 0$ and $x_N = 1$. The total number of examples in the 1D case is equal to $N$, while, for an arbitrary dimension $D$, the discretization of every variable in the same way leads to $N^D$ examples.

Let us define $f_i$ for 1D as the value of the function at the center of subinterval $i$: $f_i \equiv f((x_{i-1} + x_i)/2)$, and also we assume that $d(e_i, e_j) \equiv |x_i - x_j|$ and $d_{\min} = \min\{d(e_i, e_j)\} = h$. For fixed $h$, we will say that two input data points are first nearest neighbors if they are at distance $d_{\min}$ (this would be the equivalent of Hamming distance 1 in Boolean space).

In this way, (1) can be generalized as

$$\mathscr{C}_1[f] = \frac{1}{N_{\text{ex}} N_{\text{neigh}} \Delta f} \sum_{j=1}^{N_{\text{ex}}} \left( \sum_{d(e_i,e_j)=d_{\min}} \left| f(e_i) - f(e_j) \right| \right), \tag{2}$$

where $\Delta f = f_{\max} - f_{\min}$. For $D = 1$ we can obtain the first term of the complexity measure, $\mathscr{C}_1[f]$, for continuous input data using a grid with $N$ subintervals:

$$\mathscr{C}_1[f] = \frac{1}{2N} \sum_{i=1}^{N} \left| f_i - f_{i-1} \right|, \tag{3}$$

where we used $\Delta f = 2$, $N_{\text{ex}} = N$, and substituted the sum over the two neighboring pairs by a forward sum over the sites. Defining the complexity measure density $\mathscr{C}'_1[f] \equiv \mathscr{C}_1[f]/d_{\min}$, we can write

$$\mathscr{C}'_1[f] = \frac{1}{2} \sum_{i=1}^{N} \left| \frac{f_i - f_{i-1}}{h} \right| h, \tag{4}$$

which in the limit $h \to 0$ $(N \to \infty)$ converges to

$$\mathscr{C}'_1[f] \longrightarrow \frac{1}{2} \int_0^1 \left| \frac{df(x)}{dx} \right| dx. \tag{5}$$

In terms of notation we will use $C_1$ for the first term of the original Boolean GC measure, $\mathscr{C}_1$ for the discretized version for continuous functions, and $\mathscr{C}'_1$ will denote continuous generalization complexity density (CGC).

Equation (5) will be our proposal for the first term of the GC for continuous value input data for $D = 1$. Clearly, this function will be larger for more fluctuating functions as expected. For $D = 2$, we have

$$\mathscr{C}_1[f] = \frac{h^2}{8}\sum_{i=1}^{N}\sum_{j=1}^{N}\Big[\left|f_{i,j} - f_{i-1,j}\right| + \left|f_{i,j} - f_{i+1,j}\right| + \left|f_{i,j} - f_{i,j+1}\right| + \left|f_{i,j} - f_{i,j-1}\right|\Big], \tag{6}$$

where $f_{i,j}$ is the value of the function within the square with coordinates $x = ih$, $y = jh$. The previous expression can be written more compactly as

$$\mathscr{C}_1[f] = \frac{h^2}{4}\left(\sum_{i=1}^{N}\sum_{j=1}^{N-1}\left|f_{i,j+1} - f_{i,j}\right| + \sum_{j=1}^{N}\sum_{i=1}^{N-1}\left|f_{i+1,j} - f_{i,j}\right|\right). \tag{7}$$

If $f$ takes alternatively the maximum and minimum values ($\pm 1$) on neighboring sites, $\mathscr{C}_1[f] = 1$, taking care of counting only once the difference between neighboring sites. Defining the complexity measure density $\mathscr{C}'_1[f] \equiv \mathscr{C}_1[f]/d_{\min}$ as before, and following the same steps, we get

$$\mathscr{C}'_1[f] = \frac{1}{4}\int_0^1 dx \int_0^1 dy \left[\left|\frac{\partial f(x,y)}{\partial x}\right| + \left|\frac{\partial f(x,y)}{\partial y}\right|\right]. \tag{8}$$

The above procedure can be straightforwardly generalized to arbitrary dimension $D$ obtaining

$$\mathscr{C}'_1[f] = \frac{1}{2D}\int_0^1 dx_1 \int_0^1 dx_2 \cdots \int_0^1 dx_D \sum_{i=1}^{D}\left|\frac{\partial f(\vec{x})}{\partial x_i}\right|. \tag{9}$$

We observe that (9) is not bounded; that is, there is not a function with maximum complexity. This seems to be an intrinsic difficulty as for a real function the number of maxima and minima can grow indefinitely. In any case, (8) can be useful because it can measure complexities relative to a given function.

Along similar lines, we can build the continuous version of the second term of the complexity measure, $C_2$. In its original version for Boolean functions this term accounts for the output difference of pair of data points located at Hamming distance 2:

$$C_2[f] = \frac{1}{N_{\text{ex}} N_{\text{neigh}} \Delta f}\sum_{j=1}^{N_{\text{ex}}}\left(\sum_{d(e_i,e_j)=2}\left|f(e_i) - f(e_j)\right|\right). \tag{10}$$

For the continuous case we can write, for $D = 1$,

$$\begin{aligned} C_2[f] &= \frac{1}{2}\sum_{i=1}^{N}\left|\frac{f_{i+2} - f_i}{h}\right| h \\ &= \frac{1}{2}\sum_{i=1}^{N}\left|\left(\frac{f_{i+2} - f_{i+1}}{h}\right) - \left(\frac{f_{i+i} - f_i}{h}\right) + 2\left(\frac{f_{i+1} - f_i}{h}\right)\right| h. \end{aligned} \tag{11}$$

Defining the second-order complexity density as $\mathscr{C}'_2[f] \equiv \mathscr{C}_2[f]/d_{\min}$, we obtain in the $h \to 0$ limit

$$\mathscr{C}'_2[f] \longrightarrow \int_0^1 \left|\frac{df(x)}{dx}\right| dx. \tag{12}$$

Hence, for $D = 1$, we have that $\mathscr{C}'_2[f] = 2\mathscr{C}'_1[f]$. For $D = 2$, we have

$$\begin{aligned} \mathscr{C}_2[f] = \frac{h^2}{8}\sum_{i=1}^{N}\sum_{j=1}^{N}\Big(&\left|f_{i,j} - f_{i-1,j+1}\right| + \left|f_{i,j} - f_{i-1,j-1}\right| \\ &+ \left|f_{i,j} - f_{i+1,j-1}\right| + \left|f_{i,j} - f_{i+1,j+1}\right| \\ &+ \left|f_{i,j} - f_{i,j-2}\right| + \left|f_{i,j} - f_{i,j+2}\right| \\ &+ \left|f_{i,j} - f_{i-2,j}\right| + \left|f_{i,j} - f_{i+2,j}\right|\Big), \end{aligned} \tag{13}$$

that in the $N \to \infty$ limit leads to

$$\begin{aligned} \mathscr{C}'_2[f] \longrightarrow \frac{1}{4}\int_0^1 dx \int_0^1 dy \Bigg(&\left|\frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y}\right| \\ &+ \left|\frac{\partial f(x,y)}{\partial x} - \frac{\partial f(x,y)}{\partial y}\right| \\ &+ \left|\frac{\partial f(x,y)}{\partial x}\right| + \left|\frac{\partial f(x,y)}{\partial y}\right|\Bigg). \end{aligned} \tag{14}$$

Equation (14) will be our proposal for the continuous version of the second term of the GC measure.

*2.1. Testing the Generalization Complexity on a Set of Continuous Functions.* Having introduced an extension of the complexity measure for a set of continuously distributed data (9) and (14), we now would like to test the proposal, and for that we will use a set of trigonometric functions with parametrized complexity. The set in dimension $D$ is defined by

$$f_n^D(\vec{x}) = \prod_{j=1}^{D}\sin\left(2\pi n x_j\right), \tag{15}$$

with $n$ taking integer values $n = 1, 2, \ldots$, even if real values can be also considered (e.g., $n = 1/\lambda$). Dividing the $D$-dimensional hypercube by using a grid of spacing $1/2n$ leads

to a function that cancels at the borders of the hypercubes of side $h = 1/2n$, taking alternatively the values $\pm 1$ on nearest neighbour cells. This function is precisely the well-known parity Boolean function, having a very high complexity among the set of Boolean functions [19]. Measured by the first term of the GC measure, the parity function achieves maximum complexity of 1, and thus, given a value of the discretization spacing of $h = 1/N$, it makes sense to consider only values of $n$ up to a maximum value $n_{\max} = 1/2h = N/2$.

From the definition of the first term ($\mathscr{C}_1'$) of the continuous GC measure (CGC) (9), the complexity of the set of trigonometric functions defined by (15) can be obtained:

$$\mathscr{C}_1'\left[f_n^D\right] = \frac{2^D n}{\pi^{D-1}}. \tag{16}$$

We observe that the complexity of the set of functions grows linearly to $n$, which is proportional to the density of points where the function cancels, a sensitive measure of the variation of the function.

The family of functions (15) can be generalized to consider different variation indexes according to the spatial direction; namely,

$$f_{\mathbf{n}}^D(\vec{x}) = \prod_{j=1}^{D} \sin\left(2\pi n_j x_j\right), \tag{17}$$

where $\mathbf{n} = (n_1, n_2, \ldots, n_D)$. The complexity $\mathscr{C}_1'$ can also be easily computed and leads to

$$\mathscr{C}_1'\left[f_{\mathbf{n}}^D\right] = \frac{2^D}{\pi^{D-1}} \frac{1}{D} \sum_{j=1}^{D} n_j. \tag{18}$$

We use the family of functions (15) to compare the behavior of the discrete and continuous complexity measures introduced in the previous section. To do that we computed numerically the discrete complexities $\mathscr{C}_1$ and $\mathscr{C}_2$ as a function of $n/n_{\max}$ for $D = 1$ and 2, for a fixed value of the discretization $h$. Figure 1 shows the complexity values obtained for the continuous and discrete first terms ($\mathscr{C}_1'h$ and $\mathscr{C}_1'$, resp.) for one and two dimensions (Figures 1(a) and 1(b)), noting that for relatively low values of $n/n_{\max}$, that is, when $h \ll 1/2n$, the agreement is quite good, while for larger values, the discrete version underestimates the true complexity. A similar behaviour is observed for both plotted dimensions, noting that as the dimension increases the maximum complexity decreases by a factor $2^D/\pi^{D-1}$ (cf. (18)). The evaluation of the second term of the continuous complexity measure ($\mathscr{C}_2'$) is more cumbersome but it can be obtained with the aid of numerical integration software. In particular, for $D = 2$, the calculations lead to

$$\mathscr{C}_2'\left[f_n^2\right] = 2\left(1 + \frac{2}{\pi}\right) n. \tag{19}$$

Figure 2 shows the results for the second term of the complexity measure for the 2D set of functions. In the figure $h\mathscr{C}_2'[f_n]$ and $\mathscr{C}_2[f_n]$ are shown as a function of $n/n_{\max}$. The continuous complexity $\mathscr{C}_2'$ grows linearly according to what has been obtained in (19), showing a different behaviour with respect to the discrete version counterpart with a nonmonotonic curve. The quadratic-like shape of $\mathscr{C}_2$ (in Boolean space) has been previously analyzed [19] and its behaviour independently of $\mathscr{C}_1$ does not hold for the continuous case. The fact that the value of $\mathscr{C}_2'$ is proportional to $\mathscr{C}_1'$ (for the set of sinusoidal benchmark functions, cf. (15)) implies that the second term does not contain independent information from what is provided by the first term.

## 3. Use of Walsh Functions for Testing and Estimation of GC

The set of Walsh functions introduced by Walsh in 1923 [22] is a set of orthonormal binary functions with continuous input. Walsh functions have been widely applied in signal processing [23, 24] and are also well known because their relationship to the Hadamard transform [25]. The approach developed in the previous section cannot be applied to a set of patterns (the standard case for practical problems) as it requires knowing the analytic expression of the underlying function. In this section, we first compute the complexity of the set of Walsh functions showing that it leads to sensitive results for the estimation of GC. After this test, we apply the set of Walsh functions for carrying out the approximation of the GC for a set of patterns. The choice of the set of Walsh functions is motivated by the fact that the original GC defined in Boolean space can be computed almost straightforwardly for this set given its discrete output. Also, the intrinsic discretization of the input space as the order of the Walsh functions is increased favors their application to continuous input problems.

*3.1. The GC of the Set of Walsh Functions.* The proposed complexity measure (9) can be applied to the set of Walsh functions by introducing an appropriated limit procedure. Let us consider first the one-dimensional case, namely, the set of Walsh functions $W_n(x)$ defined on the real interval $[0, 1]$, where the index $n = 0, 1, 2, \ldots$ is chosen so that it coincides with the number of nodes of the function. For instance, $W_0(x) = 1$ for all $x$, $W_1(x) = 1$ if $0 \le x < 1/2$, $W_1(x) = -1$ if $1/2 \le x < 1$, and so forth.

We will introduce a set of continuous parametric functions $G_n(x, \beta)$ to approach the Walsh functions. $G_n(x, \beta)$ can be constructed in such a way that it has the same nodes as $W_n(x)$; it is differentiable in the neighborhood of all the nodes of $W_n(x)$ and $\lim_{\beta \to \infty} G_n(x, \beta) = W_n(x)$. The functions $G_n(x, \beta)$ can be constructed by combining sigmoidal functions centered at the nodes of $W_n(x)$ and constant functions taking values $\pm 1$ between them, joined smoothly by any interpolation procedure, such as a spline or polynomial method. Figure 3 shows two Walsh functions approximated by using hyperbolic tangent functions combined with constant ones.

Let us consider for simplicity a finite set of Walsh functions up to order $N = 2^m$ (for some fixed integer

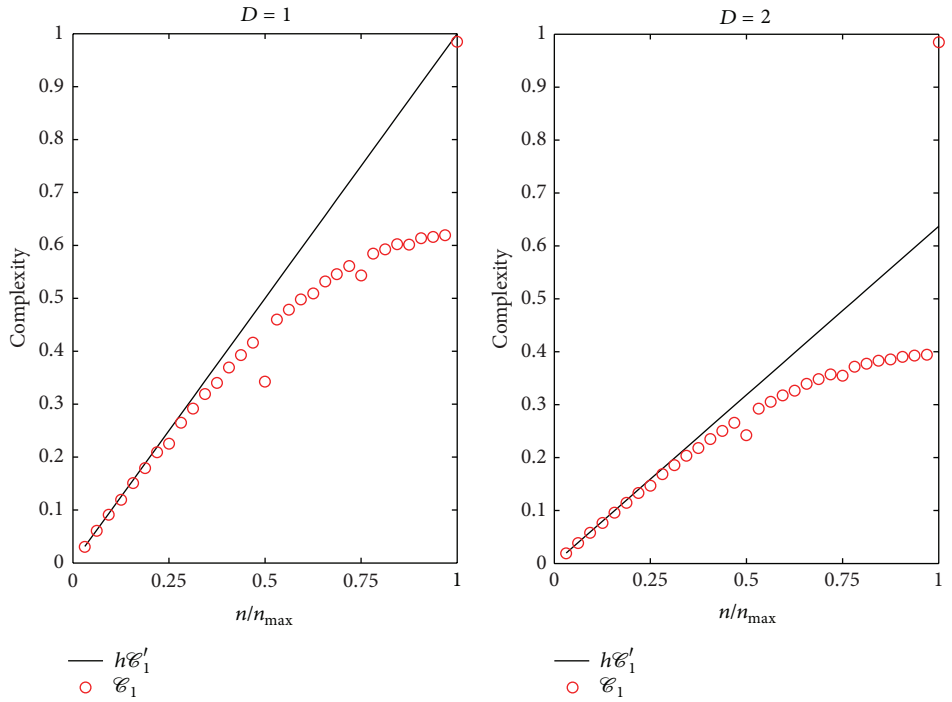


FIGURE 1: A comparison of the continuous and discrete versions of the first-order term generalization complexities for the $D = 1$ and $D = 2$ set of functions from (15) using $N = 100$. The discrete GC $C_1$ is computed over a grid with spacing $h$ and so the continuous input complexity $C_1'$ is plotted multiplied by $h$.

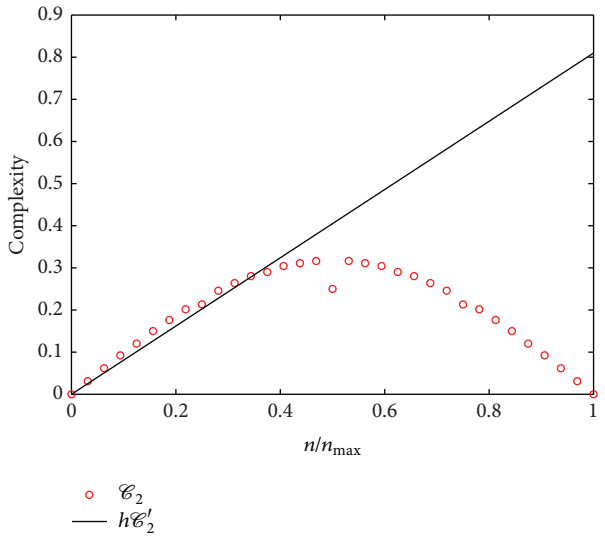


FIGURE 2: Comparison of the second terms of the complexities in their continuous and discrete versions, $h\mathscr{C}_2'$ and $\mathscr{C}_2$, for the two-dimensional set of trigonometric functions from (15) as a function of $n/n_{\max}$.

value of $m$). Then, the location of the nodes of every one of these functions belong to the set of values $x_i^* = i/N$, $i = 1, 2 \ldots, N-1$. Let $[a, b]$ be an arbitrary interval enclosing only one particular node $x_i^*$. Then the following properties hold:

$$\lim_{\beta \to \infty} \int_a^b \frac{\partial G_n(x,\beta)}{\partial x} dx = \lim_{\beta \to \infty} \left(G_n(b,\beta) - G_n(a,\beta)\right) = \pm 2,$$
$$\lim_{\beta \to \infty} G_n(x,\beta) = 0 \quad \text{if } x \neq x_i^*. \tag{20}$$

Hence, we can write

$$\frac{\partial G_n(x,\beta)}{\partial x} = 2 \sum_{i=1}^{N-1} g_i^n \phi(x - x_i^*, \beta), \tag{21}$$

where the coefficients $g_i^n$ can take the values 0 (if $W_n$ has no node at $x_i^*$) and $g_i^n = \pm 1$; otherwise $\phi(x, \beta)$ is a real function sharp peaked around $x = 0$ which satisfies $\lim_{\beta \to \infty} \phi(x, \beta) = \delta(x)$, $\delta(x)$ being a Dirac delta function [26]. Then, we can define the complexity of the Walsh functions as

$$\mathscr{C}_1'[W_n(x)] \equiv \lim_{\beta \to \infty} \mathscr{C}_1'[G_n(x,\beta)]. \tag{22}$$

From (5), (21), and (22), it follows that $\mathscr{C}_1'[W_n(x)] = n$. The extension to higher dimension is straightforward. Let $W_{\mathbf{n}}(\vec{x}) = \prod_{j=1}^{D} W_{n_j}(x_j)$ be a D-dimensional Walsh function,where $\mathbf{n} = (n_1, \ldots, n_D)$ is a set of one-dimensional

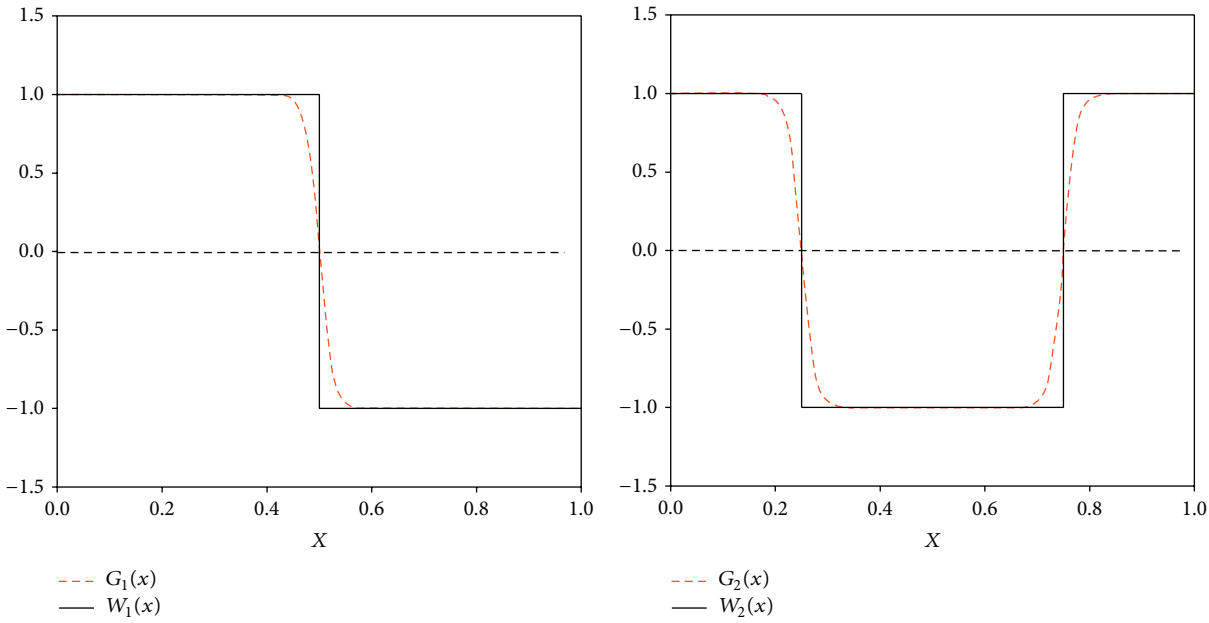


FIGURE 3: Approximation of two Walsh functions ($W_1(x)$ and $W_2(x)$) using hyperbolic tangent functions combined with constant ones ($G_1(x)$ and $G_2(x)$).

Walsh indexes, defined as before. From (9) we obtain

$$\mathscr{C}_1' \left[W_{\mathbf{n}}(\vec{x})\right] = \frac{1}{D}\sum_{j=1}^{D} n_j. \tag{23}$$

*3.2. GC Estimation for a Set of Data Points Using the Base of Walsh Functions.* Suppose that we want to compute the coefficients, $C_n$, for a given function $F$ using a set of Walsh functions $W_n(\vec{x})$ defined in the $[0,1]^D$

$$F(\vec{x}) \simeq \sum_{n=0}^{N-1} C_n W_n(\vec{x}) \tag{24}$$

given a limited set of sampling data points $(f_j, \vec{x}_j, j = 1, \ldots, M)$. We will solve the estimation of the coefficients solving a minimization problem of the square error ($S$):

$$S\left(\vec{C}\right) = \sum_{j=1}^{M}\left[f_j - F\left(\vec{x}_j\right)\right]^2 = \sum_{j=1}^{M}\left[f_j - \sum_{n'=0}^{N-1} C_{n'} W_{n'}\left(\vec{x}_j\right)\right]^2, \tag{25}$$

where $\vec{C} \equiv (C_0, C_1, \ldots, C_{N-1})$.

To find the minimum of the error function, $S$, we compute the first derivative and make it equal to 0:

$$\frac{\partial S}{\partial C_n} = -2\sum_{j=1}^{M}\left[f_j - \sum_{n'=0}^{N-1} C_{n'} W_{n'}\left(\vec{x}_j\right)\right] W_n\left(\vec{x}_j\right) = 0 \tag{26}$$

from which

$$\sum_{j=1}^{M} f_j W_n\left(\vec{x}_j\right) = \sum_{n'=0}^{N-1} C_{n'} \sum_{j=1}^{M} W_{n'}\left(\vec{x}_j\right) W_n\left(\vec{x}_j\right). \tag{27}$$

Define the vector $\vec{A} \equiv (a_0, a_1, \ldots, a_{N-1})$ as

$$a_n \equiv \sum_{j=1}^{M} f_j W_n\left(\vec{x}_j\right) \tag{28}$$

and matrix $B = \{b_{n,n'}\}$ with

$$b_{n,n'} \equiv \sum_{j=1}^{M} W_{n'}\left(\vec{x}_j\right) W_n\left(\vec{x}_j\right). \tag{29}$$

Equation (27) takes the lineal form $\vec{A} = B\vec{C}$, whose solution is given by

$$\vec{C} = B^{-1}\vec{A}. \tag{30}$$

A practical issue of the previous procedure is the computational cost involved; as for D-dimensional input data a matrix of size $N_h^D \times N_h^D$ has to be inverted (cf. (30)), where $N_h = 1/h$ is the maximum spacing used for the construction of the 1D set of Walsh functions. Nevertheless, such computation has to be done only once for given values of $D$ and $h$, being independent of the data.

Once the Walsh coefficients of a function (or data) have been obtained, the CGC can be approximated by the same

limiting procedure of the previous section. For instance, in one dimension we have

$$\begin{aligned}\mathrm{CGC}_w[F] &= \lim_{\beta\to\infty} \mathscr{C}_1' \left[\sum_{n=0}^{N-1} C_n G_n(x,\beta)\right] \\ &= \lim_{\beta\to\infty} \int_0^1 \left|\sum_{n=0}^{N-1} C_n \sum_{i=1}^{N-1} g_i^n \phi(x - x_i^*, \beta)\right| dx \\ &= \sum_{i=1}^{N-1} \left|\sum_{n=0}^{N-1} C_n g_i^n\right|,\end{aligned} \tag{31}$$

where we have used (21). For an expansion of a D-dimensional function on a finite set of $N$ Walsh functions $W_{\mathbf{n}}$ with $n_j = 0, 1, \ldots, N'$ $(j = 1, \ldots, D, N = N'^D)$, we obtain similarly

$$\mathrm{CGC}_w[F] = \frac{1}{D} \sum_{i=1}^{N'-1} \sum_{j=1}^{D} \left|\sum_{\mathbf{n}} C_{\mathbf{n}} g_i^{n_j}\right|, \tag{32}$$

where $\mathrm{CGC}_w$ indicates the approximation of the CGC using the set of Walsh basis functions. We carried out an experiment where we analyzed the accuracy of the proposed approximation to obtain a similar graph to the one shown in Figure 1(a), indicating that the approximation is working correctly. The fact that the graph obtained is almost exact to the one obtained in Figure 1(a) is consistent with what can be expected, as both are discrete approximations of the continuous value of the complexity.

## 4. Application to Real-World Input Data

In order to test practically the developed procedures, we first construct a model based on the extension of the complexity measure proposed previously, to then apply this model for the estimation of adequate neural network architecture to real-world problems. The model was estimated using the set of trigonometric functions defined by (15) for $D = 4$. For each of the analyzed data set we calculated the complexity with the above method and we found values in the range between 0 and 0.5, and the generalization ability was computed for a set of single hidden layer neural architectures with a number of neurons in the hidden layer between 2 and 50, choosing the one that leads to the lowest validation error computed in a cross-validation procedure to avoid overfitting (early stopping), where the training is performed by the standard back-propagation algorithm. From the obtained number of neurons for each of the analyzed cases, a quadratic fitting was applied to obtain the final model, shown in Figure 4 by the solid line.

Figure 4 shows the application of the developed method, described in Section 3.2, to obtain the value of CGC for a given data set. Using the constructed model (the solid line in the Figure 4), it is then possible to use the obtained CGC value to get an estimate of an adequate neural architecture to implement the function. The figure also shows the best architecture found by intensive numerical simulations (see Table 1 for the numerical values).

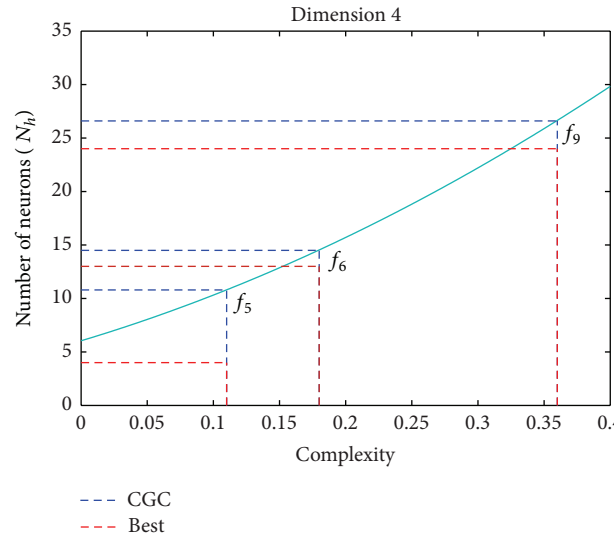


Figure 4: The model constructed for $N = 4$ input dimensions and its application to estimate an adequate size neural network for three test benchmark functions. The continuous line represents the model estimated from a set of trigonometric functions of variable complexity and the blue dashed line indicates the size estimated by the model (using the $Y$-axis values), while the red dashed line is the best size obtained from exhaustive numerical simulations.

Table 1: Results of the application of the model constructed by approximating the CGC of 10 benchmark data sets from the UCI repository.

| ID | Data set | $\mathrm{CGC}_w$ | $N_{h_{\mathrm{est}}}$ | $N_{h_{\mathrm{Best}}}$ |
|---|---|---|---|---|
| $f_1$ | Balance Scale[2,3,4,5] | 0.001 | 6.08 | 4 |
| $f_2$ | Ecoli[2,4,6,7] | 0.03 | 7.1 | 10 |
| $f_3$ | Blood[1,2,3,4] | 0.06 | 8.47 | 4 |
| $f_4$ | TicTacToe[1,2,5,8] | 0.1 | 9.84 | 4 |
| $f_5$ | Liver Disorders[1,2,5,6] | 0.11 | 10.8 | 4 |
| $f_6$ | Mammografic[2,3,4,5] | 0.18 | 14.5 | 13 |
| $f_7$ | Hayes-Roth[2,3,4,5] | 0.22 | 16.9 | 4 |
| $f_8$ | Spectf[2,3,5,7] | 0.26 | 17.8 | 23 |
| $f_9$ | Vertebral Column[1,2,3,4] | 0.36 | 26.6 | 24 |
| $f_{10}$ | Haberman[1,2,3,4] | 0.43 | 31.8 | 26 |

The table shows the identifier of the function, the name of the data set with superscripts indicating the 4 input used variables, the estimated $\mathrm{CGC}_w$, the estimated size of an adequate neural network according to the model ($N_{h_{\mathrm{est}}}$), and the best architecture found from intensive simulations ($N_{h_{\mathrm{Best}}}$).

Table 1 shows the results obtained by applying the developed method to 10 four-dimensional benchmark data sets. The data set problems are taken from the UCI repository and for each problem 4 input variables were selected. The columns show the identifier of the function, the name of the benchmark dataset with the 4 input variables used (indicated as a superscript), the estimated Generalization complexity obtained from (22), the number of neurons in

the hidden layer estimated by the model ($N_{h_{\text{est}}}$), and the best number of neurons found from exhaustive simulations ($N_{h_{\text{Best}}}$). The results obtained shown a quite good correlation between the estimated and best found values ($\rho = 0.84$, $P$ value = 0.002), suggesting the validity of the approach, even if there are some cases, like the function indicated in the table by $f_7$ for which the estimation is not extremely accurate. Nevertheless, some discrepancies are always expected as the problem of choosing an adequate neural architecture is a complex problem with no exact solution, as it depends on the particular set of patterns presented and the training process used, and thus it is an intrinsically noisy process.

## 5. Discussion and Conclusions

We have introduced in this work an extension for the generalization complexity (GC) measure for continuous input data. The analysis of the new measure on a parametrized complexity set of trigonometric functions shows that the new proposal is consistent with the expected results and with the spirit of the original measure, as the GC essentially measures for a set of data the output variations as the inputs are modified. Nevertheless, a difference between the continuous and discrete cases exists in relationship to the role of the second term of the GC, as in the continuous case this term is no longer independent from the first term (at least for the set of trigonometric functions), and thus it does not add extra information about the complexity of the data. We have also introduced an approach based on the use of the set of Walsh functions for computing the CGC measure for data expressed as a set of patterns, the typical case in most practical applications. By fitting a model that relates architecture size to function complexity, a model is built and then it is applied to the problem of selecting an adequate neural network architecture in ten real-world benchmark problems. The application of the method to the benchmark data shows that the estimated neural architectures are quite close to the optimal values, indicating the suitability of the developed approach to the architecture selection problem. The method is clearly more efficient than the trial-and-error alternative for choosing a proper neural network architecture, as the computationally heavy part of the procedure is related to a matrix inversion that has to be done only once for a given dimension and thus, once computed, it can be reused with different data sets. The GC measure provides an estimate of the complexity of the data, and as such can possibly be used not only for the case of choosing the adequate architecture for neural networks, but also when using other predictive models (like SVM, decision trees, etc.), for example, for choosing the magnitude of the penalization term of the model complexity (regularization).

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The authors acknowledge support from CICYT (Spain) through Grants TIN2008-04985 and TIN2010-16556 (including FEDER funds), from Junta de Andalucía through Grants P08-TIC-04026 and P10-TIC-5770, and from CONICET (Argentina) and SECyT Universidad Nacional de Córdoba (Argentina).

## References

[1] E. B. Baum and D. Haussler, "What size net gives valid generalization?" *Neural Computation*, vol. 1, no. 1, pp. 151–160, 1990.

[2] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," *Machine Learning*, vol. 14, no. 1, pp. 115–133, 1994.

[3] L. S. Camargo and T. Yoneyama, "Specification of training sets and the number of hidden neurons for multilayer perceptrons," *Neural Computation*, vol. 13, no. 12, pp. 2673–2680, 2001.

[4] F. Scarselli and A. Chung Tsoi, "Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results," *Neural Networks*, vol. 11, no. 1, pp. 15–37, 1998.

[5] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures—a comparative study," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 228–240, 2012.

[6] G. Mirchandani and W. Cao, "On hidden nodes for neural nets," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 5, pp. 661–664, 1989.

[7] M. Arai, "Bounds on the number of hidden units in binary-valued three-layer neural networks," *Neural Networks*, vol. 6, no. 6, pp. 855–860, 1993.

[8] Z. Zhang, X. Ma, and Y. Yang, "Bounds on the number of hidden neurons in three-layer binary neural networks," *Neural Networks*, vol. 16, no. 7, pp. 995–1002, 2003.

[9] M. Bacauskiene, V. Cibulskis, and A. Verikas, "Selecting variables for neural network committees," in *Advances in Neural Networks—ISNN*, J. Wang, Z. Yi, J. M. Zurada, B.-L. Lu, and H. Yin, Eds., vol. 3971 of *Lecture Notes in Computer Science*, pp. 837–842, Springer, 2006.

[10] H. C. Yuan, F. L. Xiong, and X. Y. Huai, "A method for estimating the number of hidden neurons in feed-forward neural networks based on information entropy," *Computers and Electronics in Agriculture*, vol. 40, no. 1–3, pp. 57–64, 2003.

[11] Y. Liu, J. A. Starzyk, and Z. Zhu, "Optimizing number of hidden neurons in neural networks," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA '07)*, pp. 121–126, ACTA Press, Anaheim, Calif, USA, February 2007.

[12] I. Wegener, *The Complexity of Boolean Functions*, John Wiley & Sons, 1987.

[13] J. Hastad, "Almost optimal lower bounds for small depth circuits," *Advanced Computer Research*, vol. 5, pp. 143–170, 1989.

[14] I. Parberry, *Circuit Complexity and Neural Networks*, MIT Press, 1994.

[15] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 289–300, 2002.

[16] M. Basu and T. K. Ho, *Data Complexity in Pattern Recognition (Advanced Information and Knowledge Processing)*, Springer, New York, NY, USA, 2006.

[17] J. S. Sánchez, R. A. Mollineda, and J. M. Sotoca, "An analysis of how training data complexity affects the nearest neighbor classifiers," *Pattern Analysis and Applications*, vol. 10, no. 3, pp. 189–201, 2007.

[18] W. Duch, N. Jankowski, and T. Maszczyk, "Make it cheap: learning with o(nd) complexity," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '12)*, pp. 1–4, 2012.

[19] L. Franco, "Generalization ability of Boolean functions implemented in feedforward neural networks," *Neurocomputing*, vol. 70, no. 1–3, pp. 351–361, 2006.

[20] L. Franco and M. Anthony, "The influence of oppositely classified examples on the generalization complexity of Boolean functions," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 578–590, 2006.

[21] I. Gómez, L. Franco, and J. M. Jerez, "Neural network architecture selection: can function complexity help?" *Neural Processing Letters*, vol. 30, no. 2, pp. 71–87, 2009.

[22] J. L. Walsh, "A closed set of normal orthogonal functions," *The American Journal of Mathematics*, vol. 45, pp. 5–24, 1923.

[23] K. G. Beauchamp, *Walsh Functions and Their Applications*, Academic Press, 1975.

[24] W. A. Evans, "Sine-wave synthesis using walsh functions," *IEE Proceedings G*, vol. 134, no. 1, pp. 1–6, 1987.

[25] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proceedings of the IEEE*, vol. 57, pp. 58–68, 1969.

[26] R. E. Mickens, *Mathematical Methods for the Natural and Engineering Sciences*, vol. 65 of *Series on Advances in Mathematics for Applied Sciences*, World Scientific, 2004.