A Minimax Method for Learning Functional Networks

E. Castillo[1], J. M. Gutiérrez[1], A. Cobo[1] and C. Castillo[2]

[1]Department of Applied Mathematics and Computational Sciences.
[2]Ocean and Coastal Research Group
University of Cantabria, Spain.

ABSTRACT

In this letter a minimax method for learning functional networks is presented. The idea of the method is to minimize the maximum absolute error between predicted and observed values. In addition, the invertible functions appearing in the model are assumed to be linear convex combinations of invertible functions. This guarantees the invertibility of the resulting approximations. The learning method leads to a linear programming problem and then: (a) the solution is obtained in a finite number of iterations, and (b) the global optimum is attained. The method is illustrated with several examples of applications, including the Hénon and Lozi series. The results show that the method outperforms standard least squares direct methods.

Correspondence author:

Enrique Castillo
Departamento de Matemática Aplicada
y Ciencias de las Computación
Universidad de Cantabria
39005 Santander
Spain

e-mail: castie@ccaix3.unican.es
Phone: (34) (942) 201722
Fax: (34) (942) 201703

A Minimax Method for Learning Functional Networks

E. Castillo (castie@ccaix3.unican.es)*, J. M. Gutiérrez (gutierjm@ccaix3.unican.es) and A. Cobo (acobo@besaya.unican.es)
Department of Applied Mathematics and Computational Sciences. University of Cantabria, Spain.

C. Castillo (carmen@puer.unican.es)
Ocean and Coastal Research Group University of Cantabria, Spain.

Keywords: Functional equations, Functional networks, Learning, Neural networks.

1. Introduction

Functional networks have been introduced by Castillo [3] and Castillo, Cobo, Gutiérrez and Pruneda [5]. The main advantage of functional networks is that they can use domain knowledge together with data knowledge. In fact, the initial topology of the network is derived from the properties the real world is assumed to have. Next, functional equations (see Aczél [1] and Castillo and Ruiz Cobo [2]) allow simplifying the network to obtain a much simpler topology, where the initially multivariate neural functions can be written in terms of unidimensional neuron functions. Once the uniqueness of representation of this networks has been analyzed and sets of linearly independent functions have been selected for approximating the resulting neuron functions, least squares methods allow estimating the parameters of the model, thus, guaranteeing the global optimum value is attained. Details of this process is given in Castillo, Cobo, Gutiérrez and Pruneda [5], pp. 61–68.

In this paper we present a new minimax method for learning functional networks. The idea of the method is to minimize the maximum absolute error between predicted and observed values. The learning method leads to a linear programming problem and then the global optimum is attained in a finite number of iterations.

We consider two of the most popular functional network architectures: the uniqueness and the separable models.

* The authors are grateful to the University of Cantabria, the Dirección General de Investigación Científica y Técnica (DGICYT) (project TIC96-0580), and to Iberdrola for partial support of this research.

2. The Uniqueness Functional Network

The functional network in Figure 1 is known as the uniqueness model, an extension of the well known associative model. The architecture of this functional network shows that the output z can be written, as a function of the inputs x and y as

$$z = f_3^{-1}(f_1(x) + f_2(y)), \tag{1}$$

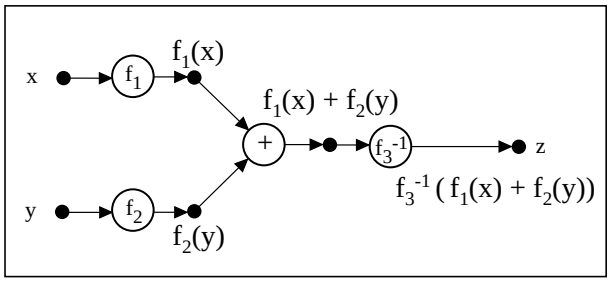

Figure 1. The uniqueness functional network.

The problem of the uniqueness of representation deals with the constraints relating the neural functions $\{f_1, f_2, f_3\}$ and $\{g_1, g_2, g_3\}$ of two different copies of the same functional network for them to give the same outputs for identical inputs; that is, it is equivalent to solving the functional equation:

$$z = f_3^{-1}(f_1(x) + f_2(y)) = g_3^{-1}(g_1(x) + g_2(y)),$$

which solution is given by (see Castillo and Ruiz-Cobo [2]):

$$f_1(x) \;=\; ag_1(x) + b;\;\; f_2(y) \;=\; ag_2(y) + c;\;\; f_3^{-1}(u) \;=\; g_3^{-1}(\frac{u - b - c}{a}) \tag{2}$$

where a, b and c are arbitrary constants. Thus, constants a, b and c are not identifiable. Thus, to have uniqueness of solution we only need to fix functions f_1, f_2 and f_3 at a point. Note that these three conditions allow determining the values of a, b and c.

Learning the function $F(x, y) = f_3^{-1}[f_1(x) + f_2(y)]$ is equivalent to learning the functions $f_1(x)$, $f_2(x)$ and $f_3(x)$. To this end, we can approximate $f_s(x); s = 1, 2, 3$ by

$$\hat{f}_s(x) = \sum_{i=1}^{m_s} a_{si}\phi_{si}(x);\;\; s = 1, 2;\;\; \hat{f}_3(x) = -\sum_{i=1}^{m_3} a_{3i}\phi_{3i}(x) \tag{3}$$

where the $\{\phi_{si}(x); i = 1, \ldots, m_s\}$; $s = 1, 2, 3$ are given sets of linearly independent functions capable of approximating $f_s(x)$; $s = 1, 2, 3$ to the desired accuracy.

To estimate the coefficients $\{a_{si}; i = 1, \ldots, m; s = 1, 2, 3\}$, we use some data set consisting of triplets $\{(x_{j1}, x_{j2}, x_{j3}) | x_{j3} = F(x_{j1}, x_{j2}); j = 1, \ldots, n\}$. Thus, the error of the approximation can be measured by

$$e_j = x_{j3} - \hat{f}_3^{-1}(\hat{f}_1(x_{j1}) + \hat{f}_2(x_{j2})); j = 1, \ldots, n \tag{4}$$

2.1. The Least-Squares Learning Method

This learning algorithm consists of minimizing the sum of squared errors, i.e., $\sum_j e_j^2$. This leads to a nonlinear minimization problem which can be solved with any standard algorithm (see, e.g., [12]). However, we can write (1) as

$$x_3 = f_3^{-1}(f_1(x_1) + f_2(x_2)) \equiv f_3(x_3) = f_1(x_1) + f_2(x_2) \tag{5}$$

and consider the errors

$$e_j = \hat{f}_1(x_{j1}) + \hat{f}_2(x_{j2}) - \hat{f}_3(x_{j3}); j = 1, \ldots, n. \tag{6}$$

Then, a simple linear minimization problem results, because to estimate the coefficients $\{a_i; i = 1, \ldots, m\}$, we minimize the sum of square errors

$$Q = \sum_{j=1}^{n} e_j^2 = \sum_{j=1}^{n} \left(\sum_{i=1}^{m_s} \sum_{s=1}^{3} a_{si} \phi_{si}(x_{js}) \right)^2 \tag{7}$$

subject to

$$\hat{f}_k(x_0) \equiv \sum_{i=1}^{m_k} a_{ki} \phi_{ki}(x_0) = \alpha_k; \; k = 1, 2, 3. \tag{8}$$

where $\alpha_k; \; k = 1, 2, 3$ are arbitrary but given real constants, which are necessary to obtain a unique solution.

Using the Lagrange multipliers we build the auxiliary function

$$Q_\lambda = \sum_{j=1}^{n} \left(\sum_{i=1}^{m_s} \sum_{s=1}^{3} a_{si} \phi_{si}(x_{js}) \right)^2 + \sum_{k=1}^{3} \lambda_k \left(\sum_{i=1}^{m_k} a_{ki} \phi_{ki}(x_0) - \alpha_k \right). \tag{9}$$

The minimum is obtained by solving the system of linear equations:

$$\begin{cases} \dfrac{\partial Q_\lambda}{\partial a_{tr}} = 2 \displaystyle\sum_{j=1}^{n} \left(\sum_{i=1}^{m_s} \sum_{s=1}^{3} a_{si} \phi_{si}(x_{js}) \right) \phi_{tr}(x_{jt}) + \lambda_t \phi_{tr}(x_0) = 0, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad r = 1, \ldots, m, t = 1, 2, 3 \\ \dfrac{\partial Q_\lambda}{\partial \lambda_t} = \displaystyle\sum_{i=1}^{m_t} a_{ti} \phi_{ti}(x_0) - \alpha_t = 0, t = 1, 2, 3 \end{cases} \tag{10}$$

Table I. Data obtained from an associative operation F.

x	y	$F(x,y)$	x	y	$F(x,y)$	x	y	$F(x,y)$
0.6378	0.0682	0.6710	0.2141	0.6465	0.9538	0.2112	0.2679	0.8791
0.3566	0.0266	0.8108	0.1743	0.1901	0.8824	0.5349	0.4655	0.7863
0.9051	0.2706	0.4655	0.6312	0.8303	0.8543	0.9463	0.1881	0.3934
0.9941	0.7795	0.5718	0.2937	0.0980	0.8367	0.6857	0.4590	0.7006
0.6558	0.0297	0.6582	0.4716	0.8124	0.9180	0.4445	0.7618	0.9111
0.1150	0.7858	1.0208	0.2702	0.5716	0.9171	0.5800	0.3202	0.7329
0.3650	0.3010	0.8294	0.9488	0.4899	0.4834	0.4187	0.1129	0.7880
0.9547	0.7103	0.5768	0.1250	0.0148	0.8889	0.2689	0.2513	0.8581
0.4692	0.9850	0.9842	0.7973	0.4388	0.6149	0.0246	0.2232	0.9266
0.6822	0.6529	0.7617	0.7544	0.6515	0.7161	0.1022	0.3327	0.9196
0.3893	0.3505	0.8276	0.1533	0.8427	1.0293	0.9706	0.2376	0.3671
0.1985	0.1323	0.8707	0.8455	0.2227	0.5211	0.9296	0.8810	0.6858
0.3763	0.2377	0.8166	0.1323	0.4422	0.9298	0.3517	0.0144	0.8126
0.4500	0.7892	0.9183						

This learning algorithm has been successfully applied in problems such as nonlinear time series or fitting two-dimensional functions (see [4, 7]). An analysis of the performance of this model compared with standard feed-forward neural networks trained with the back-propagation algorithm has been also reported in [8].

One of the main limitations of the above linear least-squares algorithm is that, in some cases, the function f_3 obtained from the learning process is not invertible (note that this condition is required in (1)) and, therefore, a nonlinear least-squares algorithm is needed. In this case, the existence of a single optimal value is not guaranteed.

For example, consider the data given in Table I, obtained from the non-linear function $F(x,y) = \sqrt{1.3 + 0.4y^2 - 0.45e^x}$.

If we consider a polynomial family of functions $\phi = \{1, x, x^2\}$ (i.e., a Taylor expansion) to approximate the neuron functions f_1, f_2 and f_3 in (1), and apply the above linear least-squares learning algorithm we get a non-invertible function $f_3(x) = -12.467x + 13.467x^2$. Therefore, the nonlinear minimization procedure is needed in this case.

In the following section we present a novel alternative without these shortcomings.

2.2. The Minimax Learning Method

In this method we minimize

$$\max_{j=1,\ldots,n} |\hat{f}_1(x_{1j}) + \hat{f}_2(x_{2j}) - \hat{f}_3(x_{3j})| \tag{11}$$

and we approximate each of the functions $f_s; s = 1, 2, 3$ as before. In addition, since f_3 must be invertible, we assume that the functions in the set $\boldsymbol{\phi}_3$ are invertible and the corresponding coefficients $a_{3i}; i = 1, \ldots, m_3$ are non-negative and add up to one. This guarantees the invertibility of the resulting approximation.

Since in linear programming problems it is customary to work with non-negative variables, we introduce the following change of variables

$$a_{si} = a^*_{si} - b;\ a^*_{si} \geq 0;\ b \geq 0;\ i = 1, \ldots, m_s;\ s = 1, 2; \tag{12}$$

Then, the problem can be stated as the linear programming problem:

Minimize ϵ subject to

$$\begin{cases} -\sum_{i=1}^{m_1}(a^*_{1i} - b)\phi_{1i}(x_{1j}) - \sum_{i=1}^{m_2}(a^*_{2i} - b)\phi_{2i}(x_{2j}) + \sum_{i=1}^{m_3} a_{3i}\phi_{3i}(x_{3j}) + \epsilon \geq 0; \forall j \\ \sum_{i=1}^{m_1}(a^*_{1i} - b)\phi_{1i}(x_{1j}) + \sum_{i=1}^{m_2}(a^*_{2i} - b)\phi_{2i}(x_{2j}) - \sum_{i=1}^{m_3} a_{3i}\phi_{3i}(x_{3j}) + \epsilon \geq 0; \forall j \\ \sum_{i=1}^{m_3} a_{3i} = 1 \\ a^*_{si} \geq 0, \quad \forall i,\ s = 1, 2; \quad a_{3i} \geq 0, \quad \forall i; \quad b \geq 0 \end{cases} \tag{13}$$

For example, consider again the data in Table I and the family of functions $\phi = \{1, x, x^2\}$. Solving the linear programing problem (13), we get the functions $\hat{f}_1(x) = -0.382 - 0.382x - 0.381x^2$, $\hat{f}_2(y) = 1.225 + 0.0077y + 0.391y^2$, and $\hat{f}_3(x) = x^2$, where $f_3(x)$ can be easily inverted thus obtaining the approximate model $\hat{F}(x, y) = f_3^{-1}(f_1(x) + f_2(y)) = \sqrt{-0.382 - 0.382x - 0.381x^2 + 1.225 + 0.0077y + 0.391y^2}$. The obtained errors are shown in Figure 2. Figures 3 and 4 show the quality of the obtained model not only on the training set, but on the whole domain $(0, 1) \times (0, 1)$.

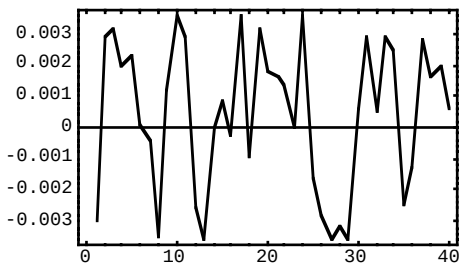

Figure 2. Errors on the 40 training samples shown in Table I.

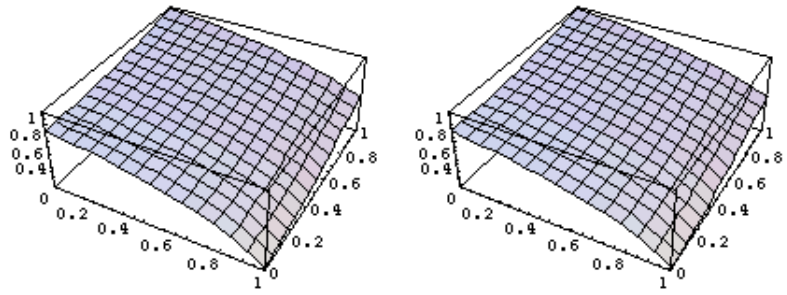

Figure 3. Real (left) and estimated (right) functions.

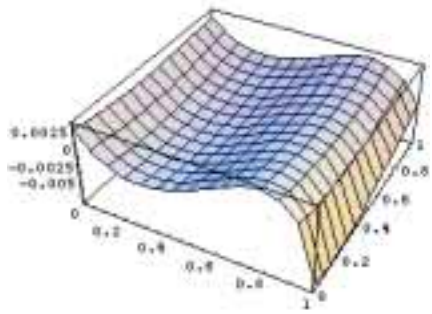

Figure 4. Error surface.

2.3. Comparison of Least-Squares and Minimax Algorithms

With the aim of comparing the performance of least-squares and minimax learning algorithms, we shall consider a time series from the Hénon chaotic map which has been previously analyzed using both functional networks (trained with the least-squares algorithm [3, 4]), and standard feed-forward neural networks (trained with the backpropagation algorithm [11]). For instance, Castillo and Gutiérrez [4] showed that functional networks with polynomial or trigonometric neuron functions trained with the least-squares method outperform neural networks with sigmoidal functions. In the following we show that even better results can be obtained using the minimax algorithm.

The Hénon map is given by the following iterative equation (see [9]):

$$x_{n+1} = 1 - ax_n^2 + 0.3x_{n-1}. \tag{14}$$

It can be shown that, depending on the values of the parameter a, the system exhibits different behaviors, ranging from periodic regimes to deterministic chaos, where orbits are sensible to small perturbations making more difficult the analysis of the model. In particular we consider a chaotic time series consisting of 500 points generated from the

initial conditions $x_0 = 0.1, x_1 = 0.3$ for the parameter value $a = 1.4$ (see Figure 5). The first hundred iterations are used for the training process and the remaining data for validating, or testing, the obtained model.

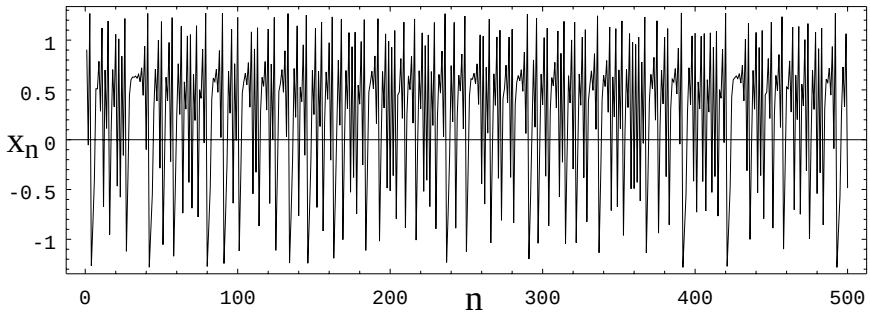

Figure 5. Time series of a chaotic orbit of the Hénon map.

If we choose a polynomial family $\phi = \{1, x, x^2\}$, for all the neuron functions, then we get the exact Hénon map given in (14). In this case we have used some information about the structure of the underlying model, i.e., the most convenient family of functions for the polynomial dynamics of the time series. However, if we use a different family for the neuron functions, such as a Fourier expansion given by $\phi = \{sin(x), cos(x), \ldots, sin(m\,x), cos(m\,x)\}$, then we obtain an approximate model. Table II shows the performance of the least-squares and the minimax learning algorithm for obtaining an approximate model for different choices of m. The maximum errors of the resulting approximate models trained over a 100-point time series are shown in Table II. Since the number of involved parameters is large, it is important to check whether the resulting models overfit to the training sample. Thus, we evaluate the model with the following 400 terms in the Hénon series. In all the cases, the train and test errors are very similar indicating that no overfitting is produced during the training process.

Note that the difference between the errors obtained with the least-squares and the minimax algorithms shown in Table II is of two orders of magnitude indicating a great difference between both methods. For instance, Figure 6 shows the training errors in the case $m = 3$ obtained with both algorithms. Note that the errors are more uniform for the minimax algorithm, since the maximum error is being minimized in this case.

Table II. Performance of several Fourier functional networks for the Hénon time series using the least-squares and the minimax learning algorithms.

Network	Maximum Training Error		Maximum Test Error	
	SSE	Minimax	SSE	Minimax
m=2	0.279	$3.40\ 10^{-3}$	0.288	$4.55\ 10^{-3}$
m=3	0.048	$6.42\ 10^{-4}$	0.049	$2.72\ 10^{-4}$
m=4	$6.36\ 10^{-3}$	$2.14\ 10^{-5}$	$5.97\ 10^{-3}$	$3.48\ 10^{-5}$
m=5	$9.17\ 10^{-4}$	$1.72\ 10^{-6}$	$9.81\ 10^{-4}$	$5.56\ 10^{-6}$

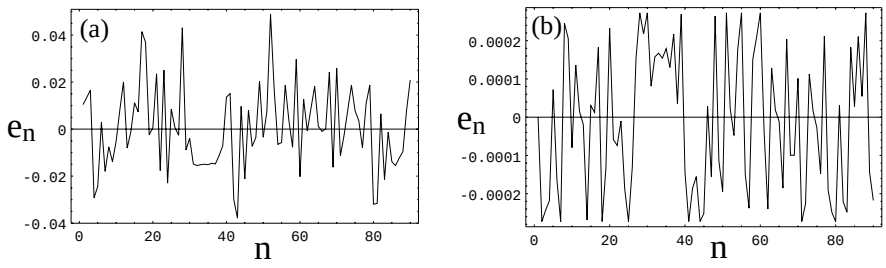

Figure 6. Training errors e_n for the (a) least-squares and (b) minimax learning algorithms.

3. The Separable Functional Network

Another interesting family of functional network architectures with many applications is the so called separable functional networks (see [4]), which has associated a functional expression which combines the separate effects of input variables. For the case of two inputs, x and y, and one output, z, we have:

$$z = F(x, y) = \sum_{i=1}^{n} f_i(x) g_i(y). \tag{15}$$

For illustrative purposes, we consider the simplest architecture from this family, which neglects double interactions by separating the contributions of each of the inputs in the form

$$z = F(x, y) = f(x) + g(y). \tag{16}$$

Note that this functional corresponds to (15) with $n = 2$ and $g_1 = f_2 = 1$.

In this case, the uniqueness of representation problem among the functions of two different representations of (16), say,

$$F(x,y) = f_1(x) + g_2(y) = f_1^*(x) + g_2^*(y). \tag{17}$$

gives the constraints (see [4]):

$$f_1^*(x) = f_1(x) - c,\ g_2^*(y) = g_2(y) + c,$$

where c is an arbitrary constant. Therefore, during the learning process, an initial functional condition for one of the functions have to be given in order to have a unique representation of the functional network.

3.1. Learning a Separable Functional Network

A least-squares learning method similar to the one presented in Section 2.1 is also available for separable networks. In this case, the functions f and g in (16) have to be approximated by considering a linear combination of known functions from a given family:

$$\hat{f}(x) = \sum_{j=1}^{m_1} a_{1j}\phi_{1j}(x), \quad \hat{g}(x) = \sum_{j=1}^{m_2} a_{2j}\phi_{2j}(x),$$

Then, the error can be measured by

$$e_i = x_{3i} - \hat{f}(x_{1i}) - \hat{g}(x_{2i});\ i = 1, \ldots, n. \tag{18}$$

Thus, to find the optimum coefficients we minimize the sum of squared errors

$$Q = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left(x_{3i} - \sum_{k=1}^{2} \sum_{j=1}^{m_k} a_{kj}\phi_{kj}(x_{ki}) \right)^2. \tag{19}$$

subject to the condition

$$\hat{f}(x_0) \equiv \sum_{j=1}^{m_1} a_{1j}\phi_{1j}(x_0) = \alpha_0, \tag{20}$$

where x_0 and α_0 are given constants.

Then, the minimum can be obtained by solving the following system of linear equations, where the unknowns are the coefficients a_{kj} and the

multiplier λ:

$$\begin{cases} \dfrac{\partial Q_\lambda}{\partial a_{1r}} = -2\displaystyle\sum_{i=1}^{n}\left(x_{3i} - \sum_{k=1}^{2}\sum_{j=1}^{m_k} a_{kj}\phi_{kj}(x_{ki})\right)\phi_{1r}(x_{1i}) + \lambda\phi_{1r}(x_0) = 0, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad r = 1,\ldots,m_1, \\ \dfrac{\partial Q_\lambda}{\partial a_{2r}} = -2\displaystyle\sum_{i=1}^{n}\left(x_{3i} - \sum_{k=1}^{2}\sum_{j=1}^{m_k} a_{kj}\phi_{kj}(x_{ki})\right)\phi_{2r}(x_{2i}) = 0, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad r = 1,\ldots,m_2, \\ \dfrac{\partial Q_\lambda}{\partial \lambda} = \displaystyle\sum_{j=1}^{m_1} a_{1j}\phi_{1j}(x_0) - \alpha_0 = 0. \end{cases} \tag{21}$$

The minimax algorithm can also be easily adapted for this new architecture. In this method we minimize

$$\max_{j=1,\ldots,n} |x_{3j} - (\hat{f}_1(x_{1j}) + \hat{f}_2(x_{2j}))|. \tag{22}$$

As before, the problem we have the linear programming problem:

Minimize ϵ subject to

$$\begin{cases} -x_{3j} + \displaystyle\sum_{i=1}^{m_1}(a_{1i}^* - b)\phi_{1i}(x_{1j}) + \sum_{i=1}^{m_2}(a_{2i}^* - b)\phi_{2i}(x_{2j}) + \epsilon \geq 0, \ \forall j \\ x_{3j} - \displaystyle\sum_{i=1}^{m_1}(a_{1i}^* - b)\phi_{1i}(x_{1j}) - \sum_{i=1}^{m_2}(a_{2i}^* - b)\phi_{2i}(x_{2j}) + \epsilon \geq 0, \ \forall j \\ a_{si}^* \geq 0, \quad \forall i, \ s = 1,2; \qquad a_{3i} \geq 0, \quad \forall i \qquad b \geq 0 \end{cases} \tag{23}$$

3.2. Example: A Non-Differentiable Map

With the aim of illustrating the performance of the above learning algorithms with maps involving non-differentiable functions, we consider the Lozi map [10]:

$$x_n = 1 - 1.7\,|x_{n-1}| + 0.5\,x_{n-2}. \tag{24}$$

We have considered a 100-point time series obtained with the initial conditions $x_0 = 0.5$ and $x_1 = 0.7$ and trained a separable functional network using both the least-squares and minimax algorithms. We obtained similar results to the previous example. For instance, Figure 7 shows the errors obtained when considering a $m = 4$ Fourier family for the functions f and g. From this figure, we can easily check how the minimax method produces more uniform errors, since the maximum error is being minimized.

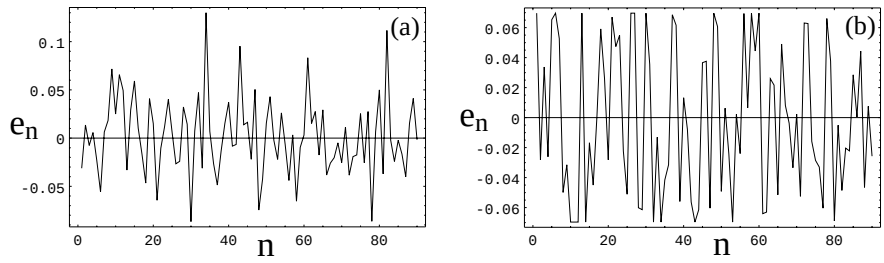

Figure 7. Training errors e_n for the (a) least-squares and (b) minimax learning algorithms.

References

1. J. Aczél. *Lectures on Functional Equations and Their Applications*. Vol. 19. Mathematics in Science and Engineering. Academic Press, 1966.
2. E. Castillo and R. Ruiz-Cobo. *Functional Equations in Science and Engineering.* Marcel Dekker, 1992.
3. E. Castillo. Functional Networks. *Neural Processing Letters* 7, 151-159, 1998.
4. E. Castillo and J. M. Gutiérrez. Nonlinear Time Series Modeling and Prediction Using Functional Networks. Extracting Information Masked by Chaos. *Physics Letters A*, Vol. 244, 71–84, 1998.
5. E. Castillo, A. Cobo, J. M. Gutiérrez and E. Pruneda. *An Introduction to Functional Networks with Applications*. Kluwer Academic Publishers, 1998.
6. E. Castillo, A. Cobo, J. M. Gutiérrez and R. E. Pruneda. Working with Differential, Functional and Difference Equations Using Functional Networks. *Applied Mathematical Modeling (in press)*, 1998.
7. E. Castillo, A. Cobo, J. M. Gutiérrez and R. E. Pruneda. Functional Networks. A New Neural Network Based Methodology. *Computer-Aided Civil and Infrastructure Engineering (in press)*, 1998.
8. E. Castillo and J. M. Gutiérrez. A Comparison of Functional Networks and Neural Networks. IASTED International Conference on Artificial Intelligence and Soft Computing, 1998, Cancún, Mexico.
9. M. Hénon. A Two Dimensional Mapping with Strange Attractor. *Communications in Mathematical Physics*, 50, 69-77.1976.
10. M. Misiurewicz. The Lozi Mapping has a Strange Attractor. In *Nonlinear Dynamics (Ed.: Hellemann, R.H.G.)*, Annals of the NY Academy of Science, 357, 348-358, 1980.
11. H. S. Stern. Neural Networks in Applied Statistics. *Technometrics*, 38, 205-214.1996.
12. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes 2^{nd} edition*. Cambridge University Press, 1992.