Parametric Learning of Associative Functional Networks Through a Modified Memetic Self-Adaptive Firefly Algorithm

Akemi Gálvez[1,2][0000−0002−2100−2289], Andrés Iglesias[1,2][0000−0002−5672−8274], Eneko Osaba[3][0000−0001−7863−9910], and Javier Del Ser[3,4][0000−0002−1260−9775]

[1] Department of Information Science, Faculty of Sciences, Toho University, 2-2-1, Miyama 274-8510, Funabashi, Japan
[2] Department of Applied Mathematics and Computational Sciences, University of Cantabria, 39005 Santander, Spain
{galveza,iglesias}@unican.es
[3] TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Spain
[4] University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain
{eneko.osaba,javier.delser}@tecnalia.com

Abstract. Functional networks are a powerful extension of neural networks where the scalar weights are replaced by neural functions. This paper concerns the problem of parametric learning of the associative model, a functional network that represents the associativity operator. This problem can be formulated as a nonlinear continuous least-squares minimization problem, solved by applying a swarm intelligence approach based on a modified memetic self-adaptive version of the firefly algorithm. The performance of our approach is discussed through an illustrative example. It shows that our method can be successfully applied to solve the parametric learning of functional networks with unknown functions.

Keywords: Artificial intelligence · Functional networks · Associative model · Parametric learning · Swarm intelligence · Firefly algorithm

1 Introduction

Models in science and engineering are usually expressed in the form of mathematical equations representing the reality with a given quality level. In addition to the free variables determining the degrees of freedom of the system, such equations usually involve some particular parameters accounting for the conditions of the problem. Learning such parameters is of paramount importance for an accurate and realistic description of the observed behavior of the system. This is also a very challenging task, typically demanding a lot of expertise, time, and effort from a human expert in order to get reliable results. Seeking to overcome this limitation, several approaches and methodologies have been devised to address this issue automatically. A classical example arises in neural networks, where several methods for parametric learning have been reported in the literature.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

In this work, we focus our attention on the functional networks, which are a powerful extension of the standard artificial neural networks (see our discussion in Sect. 2 for further details).

In this paper, we consider the problem of parametric learning of a classical model of functional networks, the so-called associative model, which is used to represent the associativity operator. This problem can be formulated as a non-linear continuous least-squares minimization problem. We solve it by applying a swarm intelligence approach based on a modified memetic self-adaptive version of the firefly algorithm. The paper is organized as follows: functional networks and their main components are described in Sect. 2. Sections 3 and 4 discuss the problem to be solved, and the firefly algorithm and its variants, respectively. Sect. 5 describes the method used to solve this optimization problem. Sect. 6 discuss an illustrative example. The paper closes with the main conclusions and a brief discussion about future work.

2 Functional networks

In short, *functional networks* can be regarded as a generalization of the standard artificial neural network in which the classical scalar weights of the neural networks are replaced by neural functions. This methodology was firstly described in 1998 by E. Castillo in [2] as a way to extend neural networks with new capabilities. Since then, they have been successfully applied to several problems in science and engineering. The interested reader is referred to [3] for a detailed explanation about functional networks, several examples and applications. Next paragraphs describe the main components of a functional network as well as the differences between neural networks and functional networks.

2.1 Components of a Functional Network

As the functional networks generalize the neural networks, they share several common features, including a close graphical representation. Fig. 1(a) shows a functional network called *associative functional network* (discussed in detail in Section 3.1), which represents the associativity operator. Following this figure, we can identify the main components of a functional network. They are:

1. Some layers of storing units: in Fig. 1(a), we can see a first layer of input units, which contains the input information. In our example, this input layer consists of the units ξ, ζ and v. We also have some intermediate layers of storing units. We point out that they are not neurons, but units storing some intermediate information. This set is optional and it is used to allow more than one neuron output to be connected to the same unit. For instance, in Fig. 1(a),we can see one layer with 4 intermediate units, represented by small circles in black. Finally, we have a layer of output units. In Fig. 1(a), it consists only of the unit μ.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

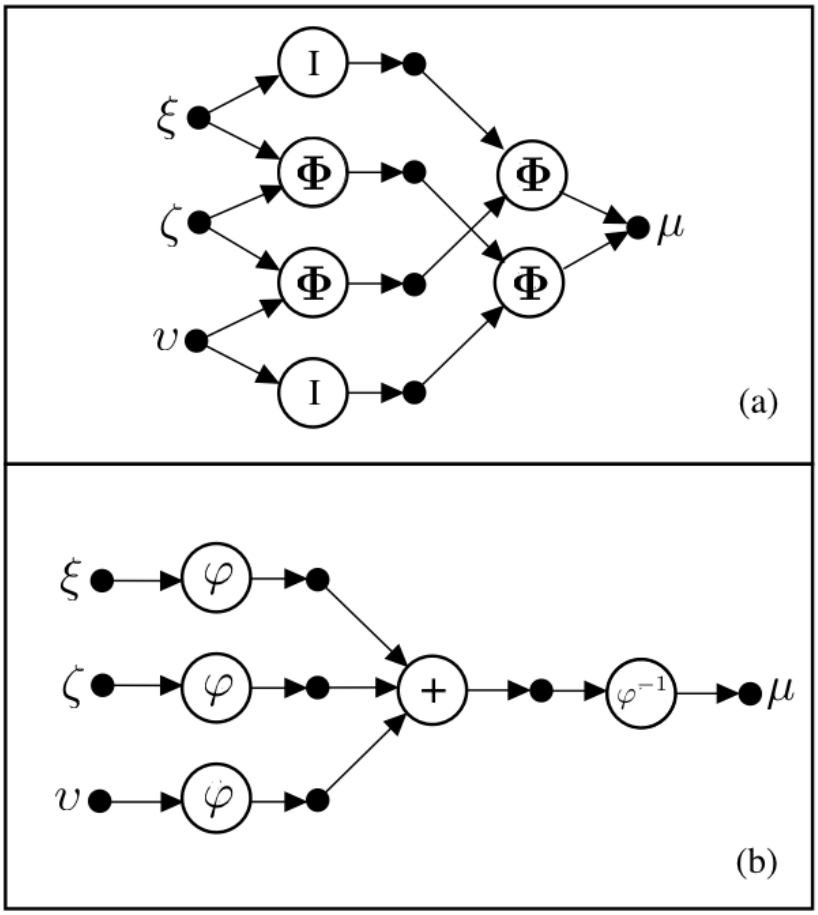

Fig. 1. Associative functional network: (a) original network; (b) simplified network.

2. One or more layers of computing units called neurons. Each neuron receives a set of input values, coming from the previous layer, makes some computations with them and returns a set of output values to the next layer. Neurons are represented graphically by circles, with the name of the corresponding neural function written inside. For example, in Fig. 1(a), we have 6 neurons arranged in two layers, where **Φ** and I represent the associative operator and the identity function, respectively.
3. A set of directed links, represented graphically by directional arrows. These arrows connect the input layer (or any intermediate layer, in general) to its adjacent layer of neurons, and neurons of one layer to its adjacent intermediate layers, or to the output layer. Note that the information flow is

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

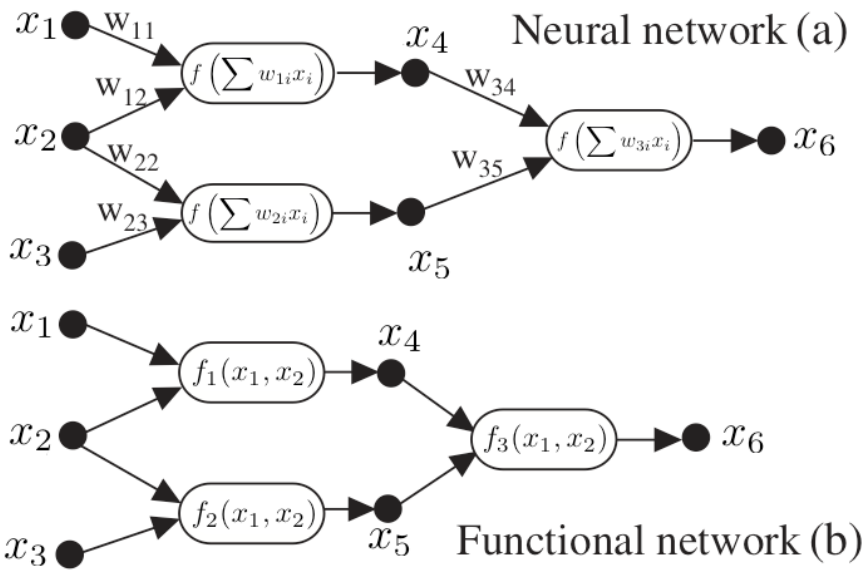

Fig. 2. Graphical differences between: (top) neural networks; (bottom) functional networks.

exclusively unidirectional: it always flows from the input layer to the output layer.

The collection of all these features define the so-called *network architecture* or *topology* of the functional network, which determines the functional capabilities of the network.

2.2 Differences Between Neural Networks and Functional Networks

In spite of their similarities, there is a number of differences between neural networks and functional networks. The most important ones are:

1. Each neuron of a standard neural network returns an output value $y = f(\sum w_{ik}x_k)$ that depends only on the value $\sum w_{ik}x_k$, where $x_1, x_2, \ldots, x_n$ are the received inputs (see Fig. 2 (top)). This means that each neural function is always univariate, as opposed to the case of functional networks, in which the neural functions are multivariate, as shown in Fig. 2 (bottom).
2. The neural functions of the functional networks can be *different* (such as functions f_1, f_2 and f_3 in Fig. 2 (bottom)). In contrast, the neural functions in neural networks are generally all *identical*.
3. The neural networks contain scalar values called weights, which must be learned. This component is not part of the functional networks, where neural functions have to be learned instead.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

4. The neuron outputs of the neural networks are usually different. On the contrary, the neuron outputs of the functional networks can be coincident. In such cases, we obtain a set of functional equations, which have to be solved through specialized techniques, such as those reported in [2, 3]. As a consequence, the neural functions in functional networks can be reduced in dimension or can be expressed as functions of lower dimension.

All these differences and features show that functional networks are more general and exhibit more interesting capabilities than the neural networks.

3 Problem to be Solved

3.1 The Associative Functional Network

In this paper we consider the associative functional network, which represents the associativity operator $\mathbf{\Phi}$ between two real numbers, given by:

$$\mathbf{\Phi}(\mathbf{\Phi}(\xi, \zeta), \upsilon) = \mathbf{\Phi}(\xi, \mathbf{\Phi}(\zeta, \upsilon)). \tag{1}$$

Our goal is to learn the function $\mathbf{\Phi}$ by using functional networks. To this purpose, we consider the network topology shown in Fig. 1(a), which replicates the mathematical structure of Eq. (1). Initially, it seems that a two-argument function $\mathbf{\Phi}$ is to be learned. However, Eq. (1) puts some constraints on it. In fact, it can be proved that the general solution of the functional equation (1) takes the form (see [3] for details):

$$\mathbf{\Phi}(\xi, \zeta) = \varphi^{-1}[\varphi(\xi) + \varphi(\zeta)] \tag{2}$$

where $\varphi(\xi)$ is a continuous and strictly monotonic (but otherwise arbitrary) function, which can only be replaced by $\eta\,\varphi(\xi)$,with η being an arbitrary constant. Replacing now Eq. (2) in Eq. (1), the two sides of Eq. (1) can be written as:

$$\varphi^{-1}[\varphi(\xi) + \varphi(\zeta) + \varphi(\upsilon)] \tag{3}$$

which means that the functional network in Fig. 1(a) is equivalent to the functional network in Fig. 1(b), where only a one-argument function φ has to be learned. This observation leads to two important conclusions:

1. This is the unique functional form for $\mathbf{\Phi}$ that satisfies Eq. (1). So, the neurons $\mathbf{\Phi}$ cannot be replaced by any other neurons.
2. The initial two-dimensional function $\mathbf{\Phi}$ is fully determined by a univariate function φ. This means that the effect of the functional equation (1) is to reduce the initial degrees of freedom of $\mathbf{\Phi}$ from a bivariate function to a univariate function φ.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

3.2 Parametric Learning of the Associative Functional Network

Suppose now that we are provided with a set of α data points $\{(\xi_j, \zeta_j, \upsilon_j)\}_{j=1,\ldots,\alpha}$, obtained from a certain unknown function $\upsilon = \mathbf{\Phi}(\xi, \zeta)$. Let us also assume that no information is available about the form of the function, but we still know that it is associative, i.e., it follows Eq. (1). To learn this associative operator, we can take pairs of numbers and their operated values as triplets $(\xi_j, \zeta_j, \upsilon_j)$ such that $\upsilon_j = \mathbf{\Phi}(\xi_j, \zeta_j) = \xi_j \oplus \zeta_j$, for $j = 1, \ldots, \alpha$. From Eq. (2) we get:

$$\omega = \mathbf{\Phi}(\xi_j, \zeta_j) \Longleftrightarrow \varphi(\omega) = \varphi(\xi) + \varphi(\zeta) \tag{4}$$

an interesting relation to be exploited for learning $\varphi(\xi)$. Therefore, learning the associative functional network is equivalent to learning the function $\varphi(\xi)$. To this end, we can approximate $\varphi(\xi)$ by a function:

$$\psi(\xi) = \sum_{i=1}^{\beta} \delta_i \psi_i(\xi), \tag{5}$$

where the $\{\psi_i(\xi)\}_{i_1,\ldots,\beta}$ is a given set of linearly independent functions, with the ability to approximate $\varphi(\xi)$ to the desired accuracy, and the coefficients δ_i are the parameters of the functional network. Note that this means that they assume the role played by the weights in a neural network.

in order to estimate the coefficients $\{\delta_i\}_{i=1,\ldots,\beta}$, we use the available data in the form of triplets $(\xi_j, \zeta_j, \upsilon_j)$. According to Eq. (4) we must have

$$\varphi(\upsilon_j) = \varphi(\xi_j) + \varphi(\zeta_j) \tag{6}$$

for $j = 1, \ldots, \alpha$. Then, the error of the approximation can be measured as:

$$\chi_j = \psi(\xi_j) + \psi(\zeta_j) - \psi(\upsilon_j) \tag{7}$$

To estimate the coefficients $\{\delta_i\}_{i=1,\ldots,\beta}$, we minimize the sum of squared errors:

$$\sum_{j=1}^{\alpha} \chi_j^2 = \sum_{j=1}^{\alpha} \left(\sum_{i=1}^{\beta} \delta_i \left[\psi_i(\xi_j) + \psi_i(\zeta_j) - \psi_i(\upsilon_j) \right] \right)^2 \tag{8}$$

subject to $\varphi(\xi_\kappa) \equiv \sum_{i=1}^{\alpha} \delta_i \psi_i(\xi_\kappa) = \lambda$, where λ is an arbitrary but given real constant, required to identify the constant η discussed above.

To summarize, learning the associative functional network finally reduces to perform parametric learning on this approximation function $\psi(\xi)$. This requires to solve a least-squares minimization problem:

$$\mathbf{\Lambda} = \underset{\{\delta_i\}_i, \lambda}{\text{minimize}} \left[\sum_{j=1}^{\alpha} \left(\sum_{i=1}^{\beta} \delta_i \left[\psi_i(\xi_j) + \psi_i(\zeta_j) - \psi_i(\upsilon_j) \right] \right)^2 + \left(\sum_{i=1}^{\alpha} \delta_i \psi_i(\xi_\kappa) - \lambda \right)^2 \right] \tag{9}$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

Unfortunately, this is a difficult multivariate nonlinear continuous optimization problem. In this paper, we address this issue by applying a swarm intelligence approach based on a modified memetic self-adaptive version of the firefly algorithm. The original firefly algorithm and the modifications introduced in this paper are briefly explained in next section.

4 The Firefly Algorithm

4.1 Original Algorithm

In this work we rely on the firefly algorithm, a bio-inspired computational algorithm for optimization [14, 15]. The basic inspiration for the algorithm is the observed flashing behavior of fireflies in nature; in particular, the variation of the intensity of light and the concept of attractiveness, which is assumed to be related with the encoded target function. The interested reader is referred to [16] for further details on the firefly algorithm. See also [4] for an updated review on bio-inspired computation at large.

The firefly algorithm is a population- based method in which the individuals (fireflies) are randomly distributed over the search space and perform exploration searching for the best location, related to the quality of the solution. The motion at iteration $t+1$ of a firefly i which is attracted by a more attractive (i.e., brighter) firefly j is governed by the following evolution equation:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \beta_0 e^{-\gamma r_{ij}^{\mu}} (\mathbf{X}_j^t - \mathbf{X}_i^t) + \alpha \left(\sigma - \frac{1}{2} \right) \tag{10}$$

where the three terms on the right-hand side of the equation account respectively for the current position of the firefly, the attractiveness of the firefly to light intensity seen by neighbor fireflies, and a random movement of the firefly if it is the brightest one. Coefficients α and σ are random numbers uniformly distributed on the interval $[0, 1]$.

Since its appearance, the firefly algorithm has been successfully applied to several problems in many different fields (see, for instance, [1, 6–9] for some illustrative applications). Also, several modifications and enhanced versions on the original algorithm have been developed [10, 11, 13]. We refer the reader to the paper in [5] for a review and taxonomic classification of several firefly algorithms and its variants and applications.

4.2 Modified Memetic Self-Adaptive Firefly Algorithm

A promising line of research nowadays is given by the so-called *memetic algorithms*. Basically, they consist of the hybridization of a global search method with a local search procedure. In agreement with this, we consider here a modified version of the firefly algorithm that enhances the original algorithm with three additional features: the use of self-adaptation schemes on some control parameters, a new elitist population model, and the hybridization with a heuristics for local search.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

The first modification consists of the application of self-adaptation schemes on some control parameters; in our particular case, this strategy is applied on the randomization parameter α, the attractiveness β, and the light absorption coefficient γ. For parameter α it is convenient to consider relatively large values at initial stages, thus promoting the explorative ability at the beginning of the simulation. Consequently, we apply a self-adaptive perturbation driven by:

$$\alpha^{t+1} = \alpha^t \left(1 - \frac{t-1}{\mathcal{T}_{max}}\right)^2$$

with $\alpha^0 = 0.9$. This means that we start with a high randomization parameter value, e.g. $\alpha = 0.9$, which corresponds to a system where fireflies are affected by a relatively large perturbation leading to a wide-range exploration, and slowly reduce it to lower values near to 0, where the system intensifies the exploitation around the local optima.

At their turn, parameters β and γ undergo a process of uncorrelated mutation where each control parameter is perturbed additively according to the normal distribution modulated by the mutation strength of that parameter, as:

$$\beta^{t+1} = \beta^t + \zeta_\beta^t N(0,1) \quad ; \quad \gamma^{t+1} = \gamma^t + \zeta_\gamma^t N(0,1) \tag{11}$$

where the mutation strength ζ_i^t, $i = \beta, \gamma$, also undergoes mutation determined by a characteristic time called the learning rate:

$$\zeta_i^{t+1} = \zeta_i^t \, exp\left[\tau' N(0,1) + \tau N_i(0,1)\right] \tag{12}$$

where $\tau' \propto 1/\sqrt{2d}$ and $\tau \propto 1/\sqrt{2\sqrt{d}}$.

The second modification concerns the population model. In the original firefly algorithm, the population of $N_{\mathcal{P}}$ individuals is entirely replaced in each generation. Consequently, it misses some valuable features typically found in the evolutionary algorithms, such as the selection pressure for survival of individuals. As a matter of fact, even the best firefly in each generation is not preserved for the next generation. In our approach, at each generation we select a percentage p of the best fireflies to be preserved unaltered to the next generation. Similarly, we select a percentage q of the worst fireflies of the swarm and split it up into two subgroups of the same size, formed respectively by fireflies that are replaced by random solutions to increase the exploratory capacity of the swarm, and by fireflies that are copies (clones) of the best members of the swarm but then undergo mutation through an additive single-point, inductive uniform mutation at a single coordinate while all other coordinates remain unaltered.

Finally, this modified firefly algorithm is enhanced by its hybridization with a local search heuristics. In this work we apply the Luus-Jaakola local search method, a heuristic proposed in 1973 to solve nonlinear programming problems [12]. The method begins with an initialization step, in which random uniform values are chosen within the search space. This is achieved by computing the upper and lower bounds for each dimension. Then, a random uniform value within these bounds is sampled for each component. This value is additively

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

added to the current position of the firefly location to generate a new candidate solution. This new solution replaces the current one if and only if this leads to a improvement of the fitness at the new position. Otherwise, the sampling space is multiplicatively decreased by a factor, freely chosen by the user. This workflow is repeated iteratively. With each iteration, the size of the neighborhood of the point is reduced, until eventually collapsing to a point.

5 Our Method

The modified firefly algorithm described in previous section has been applied to solve the parametric learning optimization problem described in Sect. 3.2. To this purpose, we need to determine some important choices. First of all, we need an adequate representation of the unknowns of the problem. The fireflies in our method correspond to real-coded vectors of length $\beta+1$ corresponding to the free variables, $\{\delta_i\}_{i=1,\ldots,\beta}$ and λ, of the least-squares minimization problem in Eq. (9). All individuals (fireflies) are initialized with uniformly distributed random numbers on the parametric domain for each coordinate. On the other hand, the fitness function corresponds to the evaluation of the least-squares function, given by Eq. (9).

Regarding the modified memetic self-adaptive firefly algorithm, some control parameters should be determined. As is is usual in the field of metaheuristic techniques, the choice of suitable values for the control parameters becomes an important issue, as it affects the performance of the method at large extent. It is also a challenging problem, since it is strongly problem-dependent. In this work, our choice is mainly based on a large collection of empirical results. These control parameters and their values for this work are:

- the number of fireflies, n_f: we set this value to $n_f = 100$ fireflies in this paper. We also tested larger populations of fireflies (up to 300 individuals) at the expense of higher computation times, without any significant improvement, so we found this value to be appropriate in our simulations.
- the number of iterations, n_{iter}. Through numerical simulations, we found that $n_{iter} = 5000$ is a suitable value, as convergence is achieved in all our simulations and higher values for n_{iter} do not lead to any improvement in our results.
- the initial attractiveness, β_0: some theoretical results indicate that $\beta_0 = 1$ is a suitable value for many optimization problems. Accordingly, we consider this value in this paper, with positive results, as shown in next section.
- the absorption coefficient, γ: its value is set up to $\gamma = 0.5$ in this work, since it provides a quick convergence of the algorithm to the optimal solution.
- the potential coefficient, μ: in principle, any positive value can be used for this parameter. However, it is noticed that the intensity of light varies according to the inverse square law. Therefore, we decided to choose $\mu = 2$ accordingly.
- the randomization parameter, α. This parameter, which varies on the interval $[0, 1]$, is used to decide the degree of randomization introduced in

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

the algorithm, which, in turn, is used in order to generate new solutions and avoid to getting stuck in a local minimum. However, it has been noticed that larger values introduce strong perturbations on the evolution of the firefly and, eventually, delay convergence to the global optima. Consequently, it is preferable to select values between these extreme ends of the spectrum. In this work, we select $\alpha = 0.5$.
- the percentage p of best solutions selected for elitism: it is set to $p = 0.1$, with the meaning that 10% of the best solutions are preserved to the next generation unaltered.
- the percentage q of worst solutions for replacement: it is taken as $q = 0.2$, meaning that 20% of the worst solutions are replaced in our population for each generation. Among them, 10% are replaced by random fireflies to promote exploration, while the rest are replaced by copies of the best individuals selected for elitism and then further mutated, as explained in Sect. 4.2.

After the selection of suitable values for its parameters, the modified firefly algorithm is executed iteratively for the given number of iterations. With the purpose to remove the stochastic effects, and also to avoid premature convergence, 30 independent executions have been exectued for each experiment. Then, the firefly with the best (minimum) fitness value is taken as the best solution to the problem.

6 Computational Simulations and Experimental Results

6.1 Computational Simulations

To check the performance of our approach, it has been applied to a practical example of an associative model with functional networks. The corresponding benchmark is given by the collection of data points shown in Table 1. The table displays a collection of 100 training points applied to learn the functional network. The parametric learning is achieved by solving the minimization problem in Eq. (9) through the method described in Sect. 5.

For the learning process in Eq. (5), we consider the family of Bernstein polynomials of degree ρ (which are linearly independent), given by:

$$\psi_i(\xi) = B_i^{\rho}(\xi) = \binom{\rho}{i} \xi^i \ (1-\xi)^{\rho-i} \quad ; \quad i = 0, \ldots, \rho$$

used to approximate the neuron function φ in Eq. (2), and perform parametric learning of the functional network.

6.2 Experimental Results

We have tested our results for different values of the polynomial degree ρ, starting with the simplest case ($\rho = 1$). The best solution obtained for the approximating function is given by the expression:

$$\varphi(\xi) = 0.234\, B_0^1(\xi) + B_1^1(\xi)$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

Table 1. Benchmark used for parametric learning of the associate functional network.

ξ	ζ	$\mathbf{\Phi}(\xi,\zeta)$	ξ	ζ	$\mathbf{\Phi}(\xi,\zeta)$	ξ	ζ	$\mathbf{\Phi}(\xi,\zeta)$	ξ	ζ	$\mathbf{\Phi}(\xi,\zeta)$
0.376	0.608	1.240	0.869	0.981	1.660	0.934	0.897	1.650	0.566	0.641	1.350
0.230	0.811	1.300	0.174	0.291	0.984	0.439	0.848	1.400	0.911	0.858	1.620
0.569	0.860	1.460	0.244	0.173	0.960	0.714	0.106	1.200	0.952	0.163	1.365
0.240	0.682	1.230	0.093	0.742	1.210	0.170	0.075	0.876	0.816	0.606	1.460
0.762	0.778	1.510	0.305	0.997	1.450	0.686	0.089	1.170	0.116	0.293	0.959
0.377	0.138	1.010	0.337	0.565	1.201	0.938	0.101	1.340	0.184	0.684	1.210
0.598	0.152	1.150	0.355	0.867	1.380	0.903	0.041	1.301	0.951	0.412	1.460
0.995	0.468	1.501	0.140	0.189	0.916	0.190	0.669	1.202	0.818	0.512	1.421
0.907	0.521	1.470	0.960	0.687	1.570	0.252	0.199	0.976	0.517	0.541	1.269
0.726	0.714	1.461	0.739	0.274	1.280	0.710	0.527	1.360	0.296	0.724	1.280
0.073	0.649	1.151	0.402	0.607	1.252	0.619	0.964	1.541	0.031	0.512	1.050
0.471	0.121	1.061	0.518	0.882	1.462	0.902	0.983	1.681	0.835	0.373	1.368
0.123	0.292	0.962	0.145	0.057	0.854	0.426	0.199	1.071	0.299	0.331	1.060
0.144	0.206	0.926	0.437	0.888	1.432	0.032	0.050	0.794	0.383	0.070	0.988
0.371	0.172	1.031	0.696	0.754	1.471	0.192	0.241	0.967	0.767	0.012	1.200
0.558	0.791	1.420	0.543	0.852	1.450	0.793	0.762	1.521	0.797	0.382	1.350
0.928	0.192	1.372	0.208	0.819	1.301	0.182	0.858	1.319	0.397	0.285	1.089
0.883	0.427	1.421	0.472	0.984	1.499	0.641	0.755	1.440	0.664	0.146	1.180
0.001	0.126	0.818	0.468	0.397	1.181	0.703	0.884	1.539	0.723	0.187	1.230
0.425	0.718	1.330	0.694	0.623	1.401	0.905	0.348	1.398	0.480	0.956	1.482
0.454	0.719	1.341	0.261	0.490	1.131	0.181	0.250	0.966	0.331	0.516	1.172
0.628	0.314	1.230	0.352	0.681	1.269	0.823	0.388	1.370	0.357	0.442	1.151
0.570	0.698	1.381	0.894	0.434	1.429	0.832	0.831	1.570	0.826	0.324	1.350
0.555	0.755	1.402	0.283	0.612	1.209	0.431	0.427	1.180	0.185	0.892	1.340
0.742	0.178	1.240	0.248	0.191	0.971	0.536	0.584	1.299	0.471	0.121	1.061

Table 2. RMSE and maximum errors for five different approximate models.

		Training phase		Testing phase	
β	par.	RMSE	Max.	RMSE	Max.
2	3	1.93×10^{-1}	4.57×10^{-1}	1.98×10^{-1}	5.16×10^{-1}
3	4	1.49×10^{-2}	4.33×10^{-2}	1.51×10^{-2}	7.63×10^{-2}
4	5	1.28×10^{-3}	3.47×10^{-3}	1.51×10^{-3}	5.72×10^{-3}
6	7	7.56×10^{-6}	2.81×10^{-5}	8.38×10^{-6}	4.56×10^{-5}
11	12	4.98×10^{-8}	1.09×10^{-7}	5.71×10^{-8}	2.63×10^{-7}

Proceeding in the same way, we also tested our method for larger values of η. The approximate models associated with $\rho = 2$ and $\rho = 3$ are given by:

$$\varphi(\xi) = 0.423B_0^2(\xi) + 0.521B_1^2(\xi) + B_2^2(\xi)$$

and

$$\varphi(\xi) = 0.395B_0^3(\xi) + 0.5276B_1^3(\xi) + 2.06B_2^3(\xi) + 1.049B_2^3(\xi),$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

respectively. The performance of these models can be better measured in terms of the RMSE (root mean squared error), given by: RMSE $= \sqrt{\frac{\mathbf{\Lambda}}{\alpha+1}}$, which takes into account not only the approximation error but also the sample size used for training.

Table 2 reports our experimental results. The table shows (in columns): the number of approximating functions, ρ, the number of free parameters, $\alpha + 1$, and the averaged RMSE and maximum error values for the 100 training points (columns 3 and 4) and testing points (coumns 5 and 6) in Table 1. As the reader can see, the RMSE shown in third column decreases as the number of approximating functions increases. We also performed cross-validation of our results to check for over-fitting. To this aim, we predicted the values of 500 data points and computed the prediction errors. The results for the RMSE are shown in columns 5 and 6 of Table 2. We can see that the RMSE and the maximum errors for the training and testing data are comparable. As a result, we can conclude that no over-fitting happens and our results can be safely validated.

All the computational work in this paper has been carried out on a 3.4 GHz Intel Core i7 processor, with 16 GB of RAM. The programming code has been implemented by the authors in the programming language of the popular scientific program *Matlab*, particularly on its version 2018b.

7 Conclusions and Future work

In this paper we addressed the parametric learning problem of functional networks by considering a classical model: the associative functional network, which represents the associativity operator. We showed that learning this functional network for an unknown functions can be transformed into the problem of learning the parameters of an approximating function, leading to a multivariate nonlinear continuous minimization problem. We solved it by applying a modified memetic self-adaptive version of the firefly algorithm. The experimental results on an illustrative example used as a benchmark show that the method performs well and is able to obtain the learn all parameters of the model and hence, replicate the approximating model with good accuracy. Our results also show that the accuracy increases with the number of approximating functions. We also performed cross-validation by using two sets of data for training and testing, respectively. Since the obtained results are comparable, we concluded that no over-fitting occurs.

Our future work includes extending this methodology to other types of functional networks and to more sophisticated approximating functions. For instance, it is still an open problem to determine whether or not the method improves for other choices of the basis functions, such as shifted step functions, logistic functions, or the like. Applying this approach to some practical problems in different domains of science and engineering is also included in the plans for future work in the field.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

Acknowledgments

The first two authors would like to thank the financial support provided by the project PDE-GIR of the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 778035, and also from the Agencia Estatal de Investigación (AEI) of the Spanish Ministry of Science, Innovation and Universities (Computer Science National Program) under grant #TIN2017-89275-R and European Funds FEDER (AEI/FEDER, UE). The last two authors wish to thank the Basque Government for its funding support through the EMAITEK and ELKARTEK programs. Javier Del Ser also received funding support from the Consolidated Research Group MATHMODE (IT1294-19), granted by the Department of Education of the Basque Government.

References

1. Campuzano, A., Iglesias, A., Gálvez, A.: Applying firefly algorithm to data fitting for the Van der Waals equation of state with Bézier curves. *Proc. of Int. Conf. on Cyberworlds, CW 2019*. IEEE Computer Society Press, Los Alamitos, CA, 211–214 (2019).
2. Castillo, E. Functional networks. *Neural Processing Letters*, **7**, 151–159 (1998).
3. Castillo, E., Iglesias, A., Ruiz-Cobo, R.: *Functional Equations in Applied Sciences*. Elsevier, Amsterdam (2005).
4. Del Ser, J., Osaba, E., Molina, D., Yang, X.S., Salcedo-Sanz, S., Camacho, D., Das, S., Suganthan, P.N., Coello, C.A., Herrera, F.: Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*, **48**, 220–250 (2019).
5. Fister I., Yang, X.S., Brest, J., Fister Jr., I.: A comprehensive review of firefly algorithms. In: Yang, X.S., Cui, Z., Xiao, R., Gandomi, A.H., Karamanoglu, M. (Eds.): *Swarm Intelligence and Bio-Inspired Computation. Theory and Applications*, Elsevier (2013) 73-102.
6. Gálvez, A., Iglesias, A.: Firefly algorithm for polynomial Bézier surface parameterization. *Journal of Applied Mathematics*. **2013**, Article ID 237984, 9 pages (2013).
7. Gálvez, A., Iglesias, A.:Firefly algorithm for explicit B-spline curve fitting to data points. *Mathematical Problems in Engineering*. **2013**, Article ID 528215, 12 pages (2013).
8. Gálvez, A., Iglesias, A.: Modified memetic self-adaptive firefly algorithm for 2D fractal image reconstruction. *Proc. of IEEE 42nd Annual Computer Software and Applications Conference, IEEE COMPSAC 2019*. IEEE Computer Society Press, Los Alamitos, CA, 165–170 (2018).
9. Gálvez, A., Fister, I., Osaba, E., Del Ser, J., Iglesias, A.: Hybrid modified firefly algorithm for border detection of skin lesions in medical imaging. *Proc. of IEEE Congress on Evolutionary Computation, IEEE CEC 2019*. IEEE Computer Society Press, Los Alamitos, CA, 111–118 (2019).
10. Iglesias, A., Gálvez, A.: Memetic firefly algorithm for data fitting with rational curves. In: *Proc. of IEEE Congress on Evolutionary Computation, CEC'2015*. IEEE Computer Society Press, Los Alamitos, CA, 507–514 (2015).
11. Iglesias, A., Gálvez, A.: New memetic self-adaptive firefly algorithm for continuous optimisation. *Int. Journal of Bio-Inspired Computation*, **8**(5), 300–317 (2016).

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42

12. Luus, R. Jaakola, T.H.I.: Optimization by direct search and systematic reduction of the size of search region. *American Institute of Chemical Engineers Journal* (AIChE),19(4), 760–766 (1973).
13. Tilahun, S.L., Ong, H.C.: Modified firefly algorithm. *Journal of Applied Mathematics*, (2012) Article ID 467631.
14. Yang, X.S.: Firefly algorithms for multimodal optimization. *Lectures Notes in Computer Science*, **5792** (2009) 169-178.
15. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. *Int. Journal of Bio-Inspired Computation*, **2**(2) (2010) 78-84.
16. Yang, X.-S.: *Nature-Inspired Metaheuristic Algorithms (2nd. Edition)*. Luniver Press, Frome, UK (2010).

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50426-7_42